DEC-11- SFDB-D

PDP-11

GETTING FORTRAN ON THE AIR

AUGUST 1971

THIS DOCUMENT IS FOR INFORMATION PURPOSES ONLY, AND IS SUBJECT TO CHANGE WITHOUT NOTICE. IT IS PRIMARILY A SUPPLEMENT TO THE PDP-11 FORTRAN IV MANUAL.

YOUR ATTENTION IS INVITED TO THE SPECIAL NOTE ON THE NEXT PAGE.

pdp11 PROGRAM LIBRARY

SPECIAL NOTE

SOFTWARE PERFORMANCE REPORT

If you have any problem or discover any inadequacy in your FORTRAN software or its documentation, please report it using the Software Performance Report forms enclosed in your software kit.

Give the Software Performance Report to your DEC Software Specialist. In most cases he will be able to provide an immediate answer to your problem, as he is kept informed of new information as soon as it becomes available. If yours is an original problem, the Software Specialist will ensure that all necessary details, examples, and supporting material are included in the Report, and then he will forward the complete report to DEC's Software Information Service Group in Maynard, Massachusetts for a thorough investigation of your problem. As soon as the investigating programmer has the answer to your problem, it will be sent to you via the Software Specialist.

This procedure is intended to provide fast replies to your Software Performance Reports either by an immediate answer from your Software Specialist or as the result of concentrating our software maintenance effort on well-documented original problems.

Your inputs are most appreciated in our continuing effort to improve our software, and with your help our commitment to good software support will remain apparent.

If you have any questions on this procedure, please contact your Software Specialist.

READER'S COMMENTS CARD

Your attention is invited to the last page of this document. The "Reader's Comments" page, when filled in and mailed, is beneficial to both you and DEC; all comments received are acknowledged and are considered when documenting subsequent manuals.

PREFACE

This document contains information which should expedite the integration of FORTRAN into a PDP-11 DOS system. Please read this document before attempting to put FORTRAN onto the system.

Chapter 1 is a description of how to load and operate the FORTRAN system. Chapter 2 contains a number of programming cautions and useful data on the current version of FORTRAN (1B, Compiler; 11A, Library). Chapter 3 contains advice on using the FORTRAN system, Chapter 4 contains additions and corrections to the current FORTRAN IV manual. Chapter 5 describes the FORTRAN Library Functions.

For more detailed information on the PDP-11 FORTRAN language and its implementation, see the FORTRAN IV manual (DEC-11-KFDA-D).

All of the following directions assume:

a. The system device is DF:

b. The user is logged in under [1,1]

c. The user is familiar with the use of the PDP-11 DOS system.

Monitor prints a period (.) or dollar sign ($) to which the user can issue a direct command. System programs print a number sign (#) to which the user can issue a command. CTRL/C causes the Monitor to print a period and accept a command.

NOTE

> The characters ., #, and $ are underlined in examples to indicate that they are printed by the DOS Monitor. All characters printed by the system are underlined; user input is not.

CONTENTS

CONTENTS

CHAPTER 1 STARTING FORTRAN ON THE PDP-11

CHAPTER 2 PROGRAMMING NOTES AND CAUTIONS

~~CHAPTER 1~~

STARTING FORTRAN ON THE PDP-11

1.1 SYSTEM REQUIREMENTS TO USE FORTRAN

1.1.1 Software Requirements

The FORTRAN system must be run with the Disk Operating System (DOS) Monitor, version 4A; PIP-11, version 4A; PAL-11R, version 5A; LINK-11, version 5A; and Librarian, version 2A. All of these programs are available from the DEC Program Library. Later versions of the above programs are also usable.

1.1.2 Hardware Requirements

FORTRAN programs can be compiled and run on any hardware configuration which supports the PDP-11 DOS Monitor.

1.2 PRINCIPLES OF CREATING A FORTRAN SYSTEM

The FORTRAN System has been supplied on either paper tape or DECtape. Briefly, the suggested sequence of operations in building a FORTRAN system is as follows:

a. Put the Compiler overlays on the system device.
b. Put the Compiler load module on the system device.
c. Put the Compiler diagnostic file on the system device.
d. Put the FORTRAN Library on the system device.
e. It is further advised that the user create a backup copy (with PIP) of the FORTRAN system on DECtape, if available on the system.

1.3 PRELIMINARY OPERATIONS

Before starting the loading process, delete any files from the system device which have the following names:

FTNØØØ.OVL
FTNØØ1.OVL
FTNØØ2.OVL
FTNØØ3.OVL
FTNØØ4.OVL
FTNØØ5.OVL
FTNØØ6.OVL
FORCOM.DGN
FORTRN
FORLIB.OBJ
EAELIB.OBJ

1.4 CREATING THE FORTRAN SYSTEM FROM A FORTRAN DECTAPE

FORTRAN systems can be built with either the 8K or 12K Compiler. Both cannot reside on the system device simultaneously. If the system has 12K or more, the 12K Compiler should be used. The 8K Compiler presently has several operational restrictions documented in Chapter 2. All directions regarding DECtape assume that the FORTRAN Compiler DECtape (and, later, the FORTRAN Library DECtape) are placed on DECtape unit Ø. DT: is synonymous with DTØ: and is shown in all DECtape examples in the following sections. The FORTRAN Compiler DECtape (DEC-11-DFFB-UC) contains the following files:

FORCOM.DGN	FORTRAN Diagnostic File
FORTRN.Ø8K	8K Compiler
OVLØ8K.LDA	
OVL18K.LDA	
OVL38K.LDA	
OVL48K.LDA	
OVL58K.LDA	
OVL68K.LDA	
FORTRN.12K	12K Compiler
OVLØ .LDA	
OVL1 .LDA	
OVL2 .LDA	
OVL3 .LDA	
OVL4 .LDA	
OVL5 .LDA	
OVL6 .LDA	

The FORTRAN Library DECtape (DEC-11-SFFB-UC) contains the following files:

FORLIB.OBJ	FORTRAN Non-EAE Library
EAELIB.OBJ	FORTRAN EAE Library
IOLØ1 .OBJ	Optional Files (see Section 2.2.4, 2.1.5, and 3.4)
DEFIN .OBJ	
DVTBØ6.PAL	

In general, the philosophy of the FORTRAN system is that if the system device is an RF11 disk, the entire system can be kept on the system disk. If the system device is an RK11 disk, the entire system except for the FORTRAN diagnostic file (FORCOM.DGN) can be kept on the system device. If the system device is an RC11 disk, the FORTRAN Compiler overlays should be kept on the system disk and all other components of the system run from DECtape.

The following instructions on loading FORTRAN from DECtape describe the various permutations in loading procedure for 12K and 8K systems as well as for system devices: DF:, DK:, and DC: (RF11 disk, RK11 disk, and RC11 disk, respectively).

1.4.1 Creating the FORTRAN System in 12K or More of Core

The user should go to section 1.4.1.1 if he has an RF11 disk, to section 1.4.1.2 if he has an RK11 disk, and to section 1.4.1.3 if he has an RC11 disk.

1.4.1.1 The 12K FORTRAN System on an RF11 Disk

The general sequence of steps to be followed consists of loading the Compiler overlays, loading the FORTRAN Compiler load module, loading the FORTRAN diagnostic file, and then loading the FORTRAN Library (FORLIB or EAELIB, depending upon the system). Details are as follows:

1. Log into the system under UIC 1,1. Perform the following two commands to the Monitor:

```
$ASSIGN
$AS DF:,OVL
```

2. Run each of the individual Compiler overlay builders as follows:

```
$RU DT:OVLØ
$RU DT:OVL1
$RU DT:OVL2
$RU DT:OvL3
$RU DT:OVL4
$RU DT:OVL5
$RU DT:OVL6
```

At this point, the contiguous files FTNØØØ.OVL through FTNØØ6.OVL are present on the system disk.

3. Place the FORTRAN Compiler load module on the system device after changing the version number of the Compiler (from VØØ1A to VØØ1B) as follows:

```
$GE DT:FORTRN.12K
$MO 26466
Ø26466/Ø4Ø461: 41Ø61
$SAVE FORTRN
$↑C
.KI
```

4. Run PIP and copy the FORTRAN diagnostic files onto the system disk, changing the protection on the diagnostic file:

```
$RU PIP


PIP-11  VØØ4A

#DF:<DT:FORCOM.DGN/CO

#DF:FORCOM.DGN/PR:2ØØ
```

5. With PIP, load the appropriate FORTRAN Library object module: FORLIB.OBJ if a non-EAE system, or EAELIB.OBJ if an EAE system. (FORLIB.OBJ can be used on an EAE system, but EAELIB.OBJ cannot be used on a non-EAE system.)

```
#DF:<DT:FORLIB.OBJ
```

or

```
#DF:<DT:EAELIB.OBJ
```

1.4.1.2 The 12K FORTRAN System on an RK11 Disk

The general sequence of steps to be followed consists of loading the Compiler overlays, loading the FORTRAN Compiler load module, and then loading the appropriate FORTRAN Library. Details are as follows:

1. Log into the system under UIC 1,1. Perform the following two commands to the Monitor:

```
$ASSIGN
$AS DK:,OVL
```

2. ~~Make the following patch to the file OVLØ (which~~ changes the name of the system device, as built into the Compiler, from DF: to DK:):

```
$GET DT:OVLØ
$MO 36332
Ø36332/Ø1476Ø: 1527Ø
$BE
```

3. Run the remaining Compiler overlay builders as follows:

```
$RU DT:OVL1

$RU DT:OVL2

$RU DT:OVL3

$RU DT:OVL4

$RU DT:OVL5

$RU DT:OVL6
```

4. Place the FORTRAN Compiler load module on the system device after changing the version number of the Compiler (from VØØ1A to VØØ1B) as follows:

```
$GE DT:FORTRN.12K
$MO 26466
Ø26466/Ø4Ø461: 41Ø61
$SAVE FORTRN
$↑C
.KI
```

5. With PIP, load the appropriate FORTRAN Library object module: FORLIB.OBJ if a non-EAE system or EAELIB.OBJ if an EAE system. (FORLIB.OBJ can be used on an EAE system, but EAELIB.OBJ cannot be used on a non-EAE system.)

```
        $RU PIP


        PIP-11 VØØ4A

        #DK:<DT:FORLIB.OBJ
or
        #DK:<DT:EAELIB.OBJ
```

6. The FORTRAN diagnostic file (FORCOM.DGN) does not function correctly at the present time on the RK11 disk and should not be loaded onto the disk (see section 2.1.1 and section 2.2.6).

~~1.4.1.3 The 12K FORTRAN System on an RC11 Disk~~

Due to the size limitations of the RC11 disk, only the Compiler overlay files should be kept on the system disk. All other files should remain DECtape resident and run from DECtape as needed. The process is as follows:

1. Log into the system under UIC 1,1. Perform the following two commands to the Monitor:

```
$ASSIGN
$AS DC:,OVL
```

2. Make the following patch to the file OVLØ which changes the name of the system device, as built into the Compiler, from DF: to DC:):

```
$GE DT:OVLØ8K
$MO 32332
Ø32332/Ø1476Ø: 1457Ø
$BE
```

3. Run the remaining Compiler overlay builders as follows:

```
$RU DT:OVL1

$RU DT:OVL2

$RU DT:OVL3

$RU DT:OVL4

$RU DT:OVL5

$RU DT:OVL6
```

4. Maintain a copy of the FORTRAN Compiler load module on DECtape (unit n) after changing the version number of the Compiler (from VØØ1A to VØØ1B) as follows:

```
$GE DT:FORTRN.12K
$MO 26466
Ø26466/Ø4Ø461: 41Ø61
$SAVE DTn:FORTRN
$↑C
.KI
```

When starting the Compiler the user can type:

```
$RU DT:FORTRN
```

which allows the Compiler to bring the overlays into core from the disk.

5. Maintain a copy of the appropriate FORTRAN Library on DECtape: FORLIB.OBJ if a non-EAE system or EAELIB.OBJ if an EAE system. (FORLIB.OBJ can be used on an EAE system, but EAELIB.OBJ cannot be used on a non-EAE system.)

```
        $RU PIP

        PIP-11 VØØ4A

        #DTn:FORLIB<DT:FORLIB.OBJ
or
        #DTn:EAELIB<DT:EAELIB.OBJ
```

6. The user can put the diagnostic file, FORCOM.DGN on the system disk if he so desires, but size limitations on the disk make this undesirable in most cases (see section 2.1.1).

1.4.2 Creating the FORTRAN System in 8K of Core

The user should go to section 1.4.2.1 if he has an RF11 disk, or to section 1.4.2.2 if he has an RC11 disk. (RK11 disks are not supported on 8K machines.) The 8K Compiler does not currently support the diagnostic file, FORCOM.DGN (see section 2.1.1).

1.4.2.1 The 8K FORTRAN System on an RF11 Disk

The general sequence of steps to be followed consists of loading the 8K Compiler overlays, loading the 8K FORTRAN Compiler load module, and then loading the appropriate FORTRAN Library file. Details are as follows:

1. Log into the system under UIC 1,1. Perform the following two commands to the Monitor:

```
$ASSIGN
$AS DF:,OVL
```

2. Perform the following patch to the file OVLØ8K.LDA:

```
$GE DT:OVLØ8K
$MO 31314
Ø31314/ØØ5Ø67: 12737
Ø31316/ØØ1Ø36: 1
Ø31320/Ø12746: 32356
Ø31322/Ø32324: 137
Ø31324/1Ø4ØØ6: 3134Ø
$BE
```

3. Run the remaining 8K Compiler overlay builders as follows:

```
$RU DT:OVL18K

$RU DT:OVL28K

$RU DT:OVL38K

$RU DT:OVL48K

$RU DT:OVL58K

$RU DT:OVL68K
```

4. Place the FORTRAN Compiler load module on the system device after changing the Compiler version number (from VØØ1A to VØØ1B) as follows:

```
$GE DT:FORTRN.Ø8K
$MO 22466
Ø22466/Ø4Ø461: 41Ø61
$SAVE FORTRN
$↑C
.KI
```

5. With PIP, load the appropriate FORTRAN Library Object module: FORLIB.OBJ if a non-EAE system or EAELIB.OBJ if an EAE system. (FORLIB.OBJ can be used on an EAE system, but EAELIB.OBJ cannot be used on a non-EAE system.)

```
     $RU PIP


     PIP-11 VØØ4A

     #DF:<DT:FORLIB.OBJ
or
     #DF:<DT:EAELIB.OBJ
```

1.4.2.2 The 8K FORTRAN System on an RC11 Disk

Due to the size limitations of the RC11 disk, only the 8K Compiler overlay files should be kept on the system disk. All other files should remain DECtape resident and run from DECtape as needed. The process is as follows:

1. Log into the system under UIC 1,1. Perform the following two commands to the Monitor:

```
$ASSIGN
$AS DC:,OVL
```

2. Make the following patches to the file OVLØ8K (which changes the name of the system device, as built into the Compiler, from DF: to DC: as well as patching the 8K system):

```
$GE DT:OVLØ8K
$MO 32332
Ø32332/Ø1476Ø: 1457Ø
$MO 31314
Ø31314/ØØ5Ø67: Ø12737
Ø31316/ØØ1Ø36: 1
Ø3132Ø/Ø12746: 32356
Ø31322/Ø32324: 137
Ø31324/1Ø4ØØ6: 3134Ø
$BE
```

~~3. Run the remaining 8K Compiler overlay builders as~~ follows:

```
        $RU DT:OVL18K

        $RU DT:OVL28K

        $RU DT:OVL38K

        $RU DT:OVL48K

        $RU DT:OVL58K

        $RU DT:OVL68K
```

4. Maintain a copy of the FORTRAN Compiler load module on DECtape (unit n) after changing the version number of the Compiler (from VØØ1A to VØØ1B) as follows:

```
        $GE DT:FORTRN.12K
        $MO 22466
        Ø22466/Ø4Ø461: 41Ø61
        $SAVE DTn:FORTRN
        $↑C
        .KI
```

When starting the Compiler the user can type:

```
        $RU  DT:FORTRN
```

which allows the Compiler to bring the overlays into core from the disk.

5. Maintain a copy of the appropriate FORTRAN Library on DECtape: FORLIB.OBJ if a non-EAE system or EAELIB.OBJ if an EAE system. (FORLIB.OBJ can be used on an EAE system, but EAELIB.OBJ cannot be used on a non-EAE system).

```
        $RU PIP


        PIP-11 VØØ4A

        #DTn:FORLIB<DT:FORLIB.OBJ
   or
        #DTn:EAELIB<DT:EAELIB.OBJ
```

1.5 CREATING THE FORTRAN SYSTEM FROM PAPER TAPES

FORTRAN systems can be built with either the 8K or 12K Compiler. Both cannot reside on the system device simultaneously. If the system has 12K or more, the 12K Compiler paper tapes are supplied with the system. If the system has 8K, the 8K Compiler tapes are supplied. The 8K Compiler presently has several operational restrictions documented in Chapter 2. Paper Tapes delivered with the hardware include:

FORTRAN Compiler Paper Tapes (one set provided)

8K Compiler	12K Compiler	
DEC-11-KF1B-PL1	DEC-11-KF2B-PL1	FORTRAN Compiler
DEC-11-KF1B-PL2	DEC-11-KF2B-PL2	Compiler Overlay Builders
DEC-11-KF1B-PL3	DEC-11-KF2B-PL3	
DEC-11-KF1B-PL4	DEC-11-KF2B-PL4	
DEC-11-KF1B-PL5	DEC-11-KF2B-PL5	
DEC-11-KF1B-PL6	DEC-11-KF2B-PL6	
DEC-11-KF1B-PL7	DEC-11-KF2B-PL7	
DEC-11-KF1B-PL8	DEC-11-KF2B-PL8	
	DEC-11-KF2B-PL9	Diagnostic File Builder
	DEC-11-KF2B-PL10	Diagnostic File

FORTRAN Library (both sets provided if EAE present on system)

EAE Version	Non-EAE Version
DEC-11-SFEB-PL1	DEC-11-SFNB-PL1
DEC-11-SFEB-PL2	DEC-11-SFNB-PL2
DEC-11-SFEB-PL3	DEC-11-SFNB-PL3
DEC-11-SFEB-PL4	DEC-11-SFNB-PL4
DEC-11-SFEB-PL5	DEC-11-SFNB-PL5
DEC-11-SFEB-PL6	DEC-11-SFNB-PL6
DEC-11-SFEB-PL7	DEC-11-SFNB-PL7
DEC-11-SFEB-PL8	DEC-11-SFNB-PL8
DEC-11-SFEB-PL9	DEC-11-SFNB-PL9

Storage philosophy is the same as for loading the system via DECtape. If the system device is an RF11 disk, the entire system can be kept on the system disk. If the system device is an RK11 disk, the entire system except for the FORTRAN diagnostic file (and the builder, DEC-11-KFxB-PL9 and 10) can be kept on the system device. If the system device is an RC11 disk, the FORTRAN Compiler overlays should be kept on the system disk and the other components run from the paper tapes.

If Compiler errors occur, the Compiler extracts an appropriate error message from the diagnostic file, which it normally finds on the system device. If the file does not exist, an error code is output without a message. The diagnostic file is not used on systems having an RK11 disk or having only 8K of core; see sections 2.1.1 and 2.2.6. A diagnostic file build program is supplied with the FORTRAN Compiler tapes to create the diagnostic file, FORCOM.DGN, and give a listing of that file.

The following instructions on loading FORTRAN from paper tape describe the various permutations in loading procedures for 12K and 8K systems as well as for system devices: DK:, DF, and DC: (RK11 disk, RF11 disk, and RC11 disk, respectively).

1.5.1 Creating the FORTRAN System in 12K or More of Core

The user should go to section 1.5.1.1 if he has an RF11 disk, to section 1.5.1.2 if he has an RK11 disk, and to section 1.5.1.3 if he has an RC11 disk.

1.5.1.1 The 12K FORTRAN System on an RF11 Disk

The general sequence of steps to be followed consists of loading the Compiler overlays, loading the FORTRAN Compiler load module, building and listing the FORTRAN diagnostic file, and then loading the FORTRAN Library. Details are as follows:

1. Log into the system under UIC 1,1. Perform the following two commands to the Monitor:

```
$ASSIGN
$AS DF:,OVL
```

2. Run each of the individual Compiler overlay builders (DEC-11-KF2B-PL2 to PL8) from the high-speed paper tape reader giving the following command each time:

```
$RU PR:
```

3. Place the FORTRAN Compiler load module (DEC-11-KF2B-PL1) on the system device after changing the Compiler version number (from VØØ1A to VØØ1B). Place the tape in the high-speed reader and proceed as follows:

```
$GE PR:
$MO 26466
Ø26466/Ø4Ø461: 41Ø61
$SAVE FORTRN
```

~~4. Load the diagnostic file build program by loading~~ DEC-11-KF2B-PL9 in the high-speed reader and proceeding as follows:

```
$GE PR:
```

Put DEC-11-KF2B-PL1Ø in the high-speed reader. If the system has a line printer, make sure it is on-line, otherwise give the following command:

```
$AS KB:,5
```

Now give the following command:

```
$BE
```

The paper tape in the reader is read, the file FORCOM.DGN is created on the disk, and a listing of the diagnostic file is generated on the line printer or terminal. Now change the protection on the diagnostic file as follows:

```
$↑C
.KI
$RU PIP


PIP-11 VØØ4A

#DF:FORCOM.DGN/PR:2ØØ
```

5. With PIP, load the appropriate FORTRAN Library tapes: the SFEB set if the system has EAE and the SFNB set if the system does not have EAE. Place DEC-11-SFxB-PL1 in the high-speed reader and proceed as follows:

```
#EAELIB<PR:/FB,/FB,/FB,/FB,/FB,/FB
```

or

```
#FORLIB<PR:/FB,/FB,/FB,/FB,/FB,/FB
```

After each individual tape is read, the message:

```
AØØ2 Ø6332Ø
$
```

is printed. Load the next tape in the high-speed reader and type CO followed by the RETURN key.

6. Once the Library tapes are loaded, LIBR-11 must be run to create a library from the file created by PIP. Proceed as follows:

```
$↑C
.KI
$RU LIBR


LIBR-11  VØØ2A

#EAELIB.OBJ,LP:EAELIB.LST<,EAELIB
```

or

```
#FORLIB.OBJ,LP:FORLIB.LST<,FORLIB
```

The filename of the listing file is used as the title of the listing.

1.5.1.2 The 12K FORTRAN System on an RK11 Disk

The general sequence of steps to be followed consists of loading the Compiler overlays, loading the FORTRAN Compiler load module, and then loading the appropriate FORTRAN Library. Details are as follows:

1. Log into the system under UIC 1,1. Perform the following two commands to the Monitor:

```
$ASSIGN
$AS DK:,OVL
```

   Put the tape DEC-11-KFZB-PL2 into the high-speed reader.

2. Make the following patch to the first Compiler overlay; this changes the name of the system device, as built into the Compiler, from DF: to DK:

```
$GE PR:
$MO 36332
Ø36332/Ø1476Ø: 1527Ø
$BE
```

3. Run the remaining Compiler overlay builders (DEC-11-KF2B-PL3 through PL8) from the high-speed reader, giving the following command each time:

```
$RU PR:
```

4. Place the FORTRAN Compiler load module (DEC-11-KF2B-PL1) on the system device after changing the version number of the Compiler (from VØØ1A to VØØ1B) as follows:

```
$GE PR:
$MO 26466
Ø26466/Ø4Ø461: 41Ø61
$SAVE FORTRN
$↑C
.KI
```

5. The FORTRAN diagnostic file (FORCOM.DGN) does not function correctly at the present time on the RK11 disk and should not be loaded onto the disk. Tapes DEC-11-KF2B-PL9 and PL10 should not be used with an RK11 disk. See sections 2.1.1 and 2.2.6.

6. With PIP, load the appropriate FORTRAN Library tapes: the SFEB set if the system has EAE and the SFNB set if the system does not have EAE. Place DEC-11-SFxB-PL1 in the high-speed reader and proceed as follows:

```
$RU PIP


PIP-11 VØØ4A

#EAELIB<PR:/FB,/FB,/FB,/FB,/FB,/FB
```

   or

```
#FORLIB<PR:/FB,/FB,/FB,/FB,/FB,/FB
```

   After each individual tape is read, the message:

```
AØØ2 Ø6332Ø
$
```

   is printed. Load the next tape in-the high speed reader and type CO followed by the RETURN key.

7. Once the Library tapes are loaded, LIBR-11 must be run to create a library from the file created by PIP. Proceed as follows:

   ```
   $↑C
   .KI
   $RU LIBR


   LIBR-11 VØØ2A

   #EAELIB.OBJ,LP:EAELIB.LST<,EAELIB
   ```

   or

   ```
   #FORLIB.OBJ,LP:FORLIB.LST<,FORLIB
   ```

   The filename of the listing file is used as the title of the listing.

1.5.1.3 The 12K FORTRAN System on an RC11 Disk

Due to the size limitations of the RC11 disk, only the Compiler overlay files should be kept on the system disk. All other files should be run from paper tape. The process is as follows:

1. Log into the system under UIC 1,1. Perform the following two commands to the Monitor:

   ```
   $ASSIGN
   $AS DC:,OVL
   ```

2. Put the tape DEC-11-KF2B-PL2 into the high-speed reader. Make the following patch to the first Compiler overlay; this changes the name of the system device, as built into the Compiler, from DF: to DC:

   ```
   $GE PR:
   $MO 36332
   Ø36332/Ø1476Ø: 1457Ø
   $BE
   ```

3. Run the remaining Compiler overlay builders (DEC-11-KF2B-PL3 through PL8) from the high-speed reader, giving the following command each time:

   ```
   $RU PR:
   ```

4. For storage conservation it is recommended that the user not attempt to use the FORTRAN diagnostic file with the RC11 disk (see sections 2.1.1 and 2.2.6). However, if it is desired to use the diagnostic file while learning about the system, following step 3 above, proceed as shown below:

   Load the diagnostic file build program by loading DEC-11-KF2B-PL9 in the high-speed reader and give the following command:

   ```
   $GE PR:
   ```

   Put DEC-11-KF2B-PL1Ø in the high-speed reader. If the system has a line printer, make sure it is on-line; otherwise, give the command:

   ```
   $AS KB:,5
   ```

Now give the command:

```
$BE
```

The paper tape in the reader is read, the file FORCOM.DGN is created on the disk, and a listing of the diagnostic file is generated on the line printer or terminal.

5. The user should keep a copy of DEC-11-KF2B-PL1, the FORTRAN Compiler load module on paper tape, after changing the version number of the Compiler (from VØØ1A to VØØ1B) as follows:

```
$GE PR:
$MO 26466
Ø26466/Ø4Ø461: 41Ø61
$SAVE PP:
```

6. In order to keep the FORTRAN Library on paper tape, the user has the option of creating one large library or six smaller libraries. To create one large library, follow the directions in section 1.5.1.2, step 7, specifying PP: as the output device to the LIBR-11.

   To create six smaller libraries, run LIBR-11 six times following the directions in section 1.5.1.2, step 7, specifying PP: as the output device and PR: as the input device for each of the six libraries.

1.5.2 Creating the FORTRAN System in 8K of Core

The user should go to section 1.5.2.1 if he has an RF11 disk, or to section 1.5.2.2 if he has an RC11 disk. (RK11 disks are not supported on 8K machines.) The 8K Compiler does not currently support the diagnostic file, FORCOM.DGN.

1.5.2.1 The 8K FORTRAN System on an RF11 Disk

The general sequence of steps to be followed consists of loading the 8K Compiler overlays, loading the 8K Compiler load module, and then loading the appropriate FORTRAN Library file. Details are as follows:

1. Log into the system under UIC 1,1. Perform the following two commands to the Monitor:

```
$ASSIGN
$AS DF:,OVL
```

2. Place DEC-11-KF1B-PL2 in the high-speed reader. Perform the following patch to the 8K Compiler:

```
$GE PR:
$MO 31314
Ø31314/ØØ5Ø67: 12737
Ø31316/ØØ1Ø36: 1
Ø3132Ø/Ø12746: 32356
Ø31322/Ø32324: 137
Ø31324/1Ø4ØØ6: 3134Ø
$BE
```

3. Run each of the remaining 8K Compiler overlay builders ~~(DEC-11-KF1B-PL3 through PL8) from the high-speed~~ reader giving the following command each time:

```
     $RU PR:
```

4. Place the FORTRAN Compiler load module (DEC-11-KF1B-PL1) on the system device after changing the Compiler version number (from VØØ1A to VØØ1B). Place the tape in the high-speed reader and proceed as follows:

```
     $GE PR:
     $MO 22466
     Ø22466/Ø4Ø461: 41Ø61
     $SAVE FORTRN
     $↑C
     .KI
```

5. With PIP, load the appropriate FORTRAN Library tapes: the SFEB set if the system has EAE and the SFNB set if the system does not have EAE. Place DEC-11-SFxB-PL1 in the high-speed reader and proceed as follows:

```
     $RU PIP


     PIP-11 VØØ4A

     #EAELIB<PR:/FB,/FB,/FB,/FB,/FB,/FB
or
     #FORLIB<PR:/FB,/FB,/FB,/FB,/FB,/FB
```

After each individual tape is read, the message:

```
     AØØ2 Ø363332Ø
     $
```

is printed. Load the next tape in the high-speed reader and type CO followed by the RETURN key.

6. Once the Library tapes are loaded, LIBR-11 must be run to create a library from the file created by PIP. Proceed as follows:

```
     $↑C
     .KI
     $RU LIBR


     LIBR-11 VØØ2A

     #EAELIB.OBJ,LP:EAELIB.LST<,EAELIB
or
     #FORLIB.OBJ,LP:FORLIB.LST<,FORLIB
```

The filename of the listing file is used as the title of the listing. To run the Library from paper tape, see section 1.5.2.2, step 5.

1.5.2.2 The 8K FORTRAN System on an RC11 Disk

Due to the size limitations of the RC11 disk, only the Compiler overlay files should be kept on the system disk. All other files should be run from paper tape. The details follow.

~~1. Log into the system under UIC 1,1. Perform the~~ following two commands to the Monitor:

```
$ASSIGN
$AS DC:,OVL
```

2. Put the tape DEC-11-KF1B-PL2 into the high-speed reader. Make the following patch to the first Compiler overlay; this changes the name of the system device, as built into the Compiler, from DF: to DC: and patches the 8K Compiler):

```
$GE PR:
$MO 32332
Ø32332/Ø1476Ø: 1457Ø
$MO 31314
Ø31314/ØØ5Ø67: Ø12737
Ø31316/ØØ1Ø36: 1
Ø3132Ø/Ø12746: 32356
Ø31322/Ø32324: 137
Ø31324/1Ø4ØØ6: 3134Ø
$BE
```

3. Run the remaining Compiler overlay builders (DEC-11-KF1B-PL3 through PL8) from the high-speed reader, giving the following command each time:

```
$RU PR:
```

4. The user should keep a copy of DEC-11-KF1B-PL1, the FORTRAN Compiler load module on paper tape after changing the Compiler version number (from VØØ1A to VØØ1B). Place the tape in the high-speed reader and proceed as follows:

```
$GE PR:
$MO 22466
Ø22466/Ø4Ø461: 41Ø61
$SAVE PP:
```

5. In order to keep the FORTRAN Library on paper tape, the user has the option of creating one large library or six smaller libraries. To create one large library, follow the directions in section 1.5.1.2, step 7, specifying PP: as the output device to LIBR-11.

   To create six smaller libraries, run LIBR-11 six times following the directions in section 1.5.1.2, step 7, specifying PP: as the output device and PR: as the input device for each of the six libraries.

1.6 STORAGE OF THE FORTRAN SYSTEM

The user must keep the Compiler overlay files on the system disk. The Compiler load module (FORTRN) can be kept on the [1,1] area of the system disk or, optionally, on DECtape (see section 1.4.1.3, step 4) or paper tape (see section 1.5.1.3, step 5) and run as follows:

```
        $RU  DT:FORTRN
or
        $RU  PR:
```

If compilation errors occur, the Compiler extracts an error message from the diagnostic file on the system device [1,1] area FORCOM.DGN. If the file does not exist, an error code is printed without a message. Table 3-1 contains a listing showing error codes and corresponding messages. Thus, putting this file on the system device is optional. However, it is recommended that the file be on the system at least until the user has experience using the FORTRAN Compiler.

The FORTRAN Library is supplied in two different versions: an EAE version which uses the Extended Arithmetic Element (EAE model number KE11-A) for arithmetic. This version of the Library cannot be used on a non-EAE system. The non-EAE Library does not use the EAE and runs on any FORTRAN system, whether or not the EAE is present.

The FORTRAN Library DECtape contains three other files: IOLØ1.OBJ, DEFIN.OBJ, and DVTBØ6.PAL. The use of these files is described in sections 2.1.4, 2.2.4, and 3.4.

The FORTRAN Library can be kept on the system disk either in the user's area or the [1,1] area, DECtape, or paper tape, depending upon timing and storage trade-offs determined by the user.

CHAPTER 2

Programming Notes and Cautions

2.1 FORTRAN COMPILER NOTES AND CAUTIONS (VØØ1B)

2.1.1 Storage of FORTRAN Overlays and Diagnostic File

The FORTRAN Compiler overlays FTNØØØ.OVL through FTNØØ6.OVL, as well as the diagnostic file FORCOM.DGN, can reside only on the system disk. The Compiler load module, FORTRN, can reside on any device. If space on the system disk is at a premium, FORCOM.DGN can be deleted and the Compiler run without it; in which case any source diagnostics output an error code and not the text of the error (see Table 3-1).

2.1.2 8K Version Symbol Table

Using the 8K version of the Compiler, the amount of space available for symbols is slightly less than 400 (decimal) words. A simple variable entry in the symbol table is 8 words long. A constant entry is 8 words plus the size of the constant. An array entry is 10 words plus one word for each dimension. For example, a complex constant entry is 12 words long and a 3-dimensional array entry is 13 words long. On the average, in an 8K machine, the Compiler can handle 40-45 entries in the table.

2.1.3 8K Version Compiler I/O

Because of buffer size constraints in the 8K Compiler, all input and output from the Compiler must be done to the disk. Thus the command string:

```
          #DF:ABC,DF:DEF<DF:XYZ
```

executes, while:

```
#DTØ:ABC,DF:DEF<DF:XYZ
```

does not.

In an 8K machine, if any of the files specified in the Compiler output command string already exist, an FØØ7 error is issued when the Compiler attempts to delete the old files.

2.1.4 DEFINE FILE Statement

The DEFINE FILE statement does not work yet. An alternate way of accomplishing the task is to use routine DEFIN as follows:

```
CALL DEFIN (a,m,l,U,v)
```

where a, m, l, U and v are of the same form as in the DEFINE FILE statement (see the FORTRAN manual, part I, page 5-9). When linking, link DEFIN.OBJ *before* searching the Library. (DEFIN.OBJ is on the FORTRAN Library tape, DEC-11-SFFB-UC.)

2.1.5 DATA Statement

In DATA statements, a constant of the form:

```
.NOT..TRUE.
```

or

```
.NOT..FALSE
```

will not be diagnosed properly. These are not acceptable constants to FORTRAN IV at any time.

In DATA statements, using octal or hexadecimal constants to preset any type except integer causes Compiler errors 105 and 107 to be issued improperly.

2.1.6 Illegal Constants

Illegal forms of constants, such as -3.D, are not yet detected as errors. They compile with a value of zero.

LOGICAL*1 (byte mode) usages currently do not always compile correct code.

COMPLEX constants and expressions currently do not compile correctly.
X = 4HABCD where X is type REAL or DOUBLE does not compile correctly.

2.1.7 FORMAT Statements

FORMAT statements whose data lists are an exact multiple of 41 characters (excluding non-significant blanks) generate an extraneous

```
        .ASCII↑↑
```

directive within the PAL assembly which in turn causes a "Q" assembler error on the offending line in the source listing. This error does not affect the proper execution of a program.

2.1.8 Compilation of Expressions

Expressions of the form:

```
        NOT.±n
```

where n is any number do not compile correctly. Diagnostic 33 is issued.

2.1.9 Main Program Caution

A main program having no executable statements will not compile correctly.

2.1.10 DIMENSION Statement Used with a TYPE Statement

Forms similar to:

```
DIMENSION R(1Ø)
LOGICAL R
```

cause bad arrays to be generated, whereas by placing the type statement ahead of the DIMENSION statement the arrays are formed correctly as follows:

```
LOGICAL R
DIMENSION R(1Ø)
```

The type processor does not currently check whether an item has been previously dimensioned.

2.1.11 EXTERNAL Statement

The EXTERNAL statement does not currently work.

2.1.12 Monitor Stack Overflow

Occasionally, (with the 8K Compiler only), a Monitor stack overflow error (FØØ1, a fatal error) is issued after compiling several programs in sequence. The current solution is to reload the Monitor and the Compiler, and recompile the last program attempted at the time of the failure. Source errors during compilation will also cause this condition.

2.1.13 CTRL/C Caution

Typing ↑C followed by BEGIN or RESTART while the Compiler is running does not work. The Compiler must be reloaded by using a RUN FORTRN command to the Monitor.

2.1.14 Implied DO

In READ and WRITE statements, any I/O list elements following an implied DO are handled incorrectly. The Compiler keeps an incorrect count of the number of elements being processed. No I/O list elements should follow an implied DO in this version of the Compiler. However, an implied DO can be followed by another implied DO.

Bad characters in implied DO loops cause a halt with a fatal error condition. For example, use of a period (.) instead of a comma (,) as follows:

```
          WRONG:      (A(I).I=1,N)
          RIGHT:      (A(I),I=1,N)
```

Reload the Monitor.

2.1.15 DATA Statements

If the user is compiling a program from cards, DATA statements are likely to cause a system failure if a DATA statement is followed by a continuation card. DATA statements should not be used with continuation cards. Where data extends onto more than one card, use more than one DATA statement.

2.1.16 EQUIVALENCE Statement

A large number of equivalences can cause the Compiler to halt the system.

2.2 FORTRAN Library Notes and Cautions

2.2.1 Unacceptable Names

A Linker error results if (1) any of the followinq names are used as a subroutine or function name and (2) the OTS routine of the same name is linked to the program containing the reference. This restriction will be removed in the future.

ABSØ1
ADJØ2
AIAGØ1
AIAXØ1
AINTØ1
ALOGØ1
AMODØ1
ATANØ1

CABSØ1
CEXPØ1
CLOGØ1
CLSEØ1
CNJGØ1
CPLXØ1
CSINØ1
CSQTØ1

DABSØ1
DATNØ1
DBLEØ1
DDCIØ1
DDCOØ1
DEXPØ1
DFILØ1
DIAXØ1
DICIØ1
DICOØ1
DIMØ1
DINTØ1
DLCIØ1
DLCOØ1
DLOGØ1
DMODØ1
DSGNØ1
DSINØ1
DSQTØ1
DVTBØ6

ENDOØ3
ERRCØ4
EXITØ2
EXPØ1

FDEVØ
FINDØ1
FLATØ1

IABSØ1
IARGØ1
IDIMØ1
IFIXØ1
INFRØ1
INRRØ1
INTØ1
IOBFØ1
IOFIØ1
IORDØ1
IORIØ1
IOUDØ1
IOUIØ1
ISETØ3
ISGNØ1

MAXØØ1
MINØØ1
MIX1Ø2
MODØ1

OPENØ1
OTSVØ1

PAUSØ1
PDMPØ1
POLHØ1
POPRØ3
PSHPØ1
PSHRØ6
PWDDØ1
PWDIØ1
PWIIØ1
PWRIØ1
PWRRØ1

READØ1
REALØ1
RNDMØ2
RNDUØ2
FWEFØ

SERRØ1
SFILØ1
SINØ1
SNCOØ1
SNGLØ1
STOPØ2
SQRTØ1
SVSPØ2

TANHØ1
TRSTØ1

VØ11A

WRITØ1

2.2.2 'END=' Feature

Use of the 'END=' feature currently results in a Monitor error at RUN time. If it is desired to use this feature, location $IOF+236 (octal) should be changed from Ø16ØØ4 to Ø164Ø4. This will be fixed in the next version.

2.2.3 SETERR Subroutine

SETERR does not work correctly.

2.2.4 I/O of Logical Elements

There is an I/O problem with logical variables and logical array elements. Use of either of these in an I/O list will result in an undefined global reference at link time. This problem can be solved by linking IOLØ1.OBJ to the main program before the FORTRAN Library (IOL01.OBJ is on the FORTRAN Library DECtape, DEC-11-SFFB-UC). The problem does not exist for I/O of entire logical arrays.

2.2.5 Complex Routines Caution

In general, the complex routines are positioned incorrectly. This will be corrected in the next release. If the user wishes to use these routines, he should search the Library twice in order to force linking.

2.2.6 Text of Error Messages

Users of RK systems should not put FORCOM.DGN, the diagnostic file, on the RK disk. The Compiler prints the correct error code, but extracts the wrong error message from the diagnostic file. See Table 3-1 in Chapter 3.

The Object Time System is unable to read the error message file at present. As a result, the English text of RUN Time error messages is missing. Table 3-2 (in Chapter 3) contains a list of diagnostic codes and their corresponding English messages.

Chapter 3

Using the FORTRAN System

3.1 FORTRAN PROGRAM EXECUTION

Figure 3-1 shows the three discrete steps required to prepare a FORTRAN source program for execution: (1) compilation, (2) assembly and (3) linking.

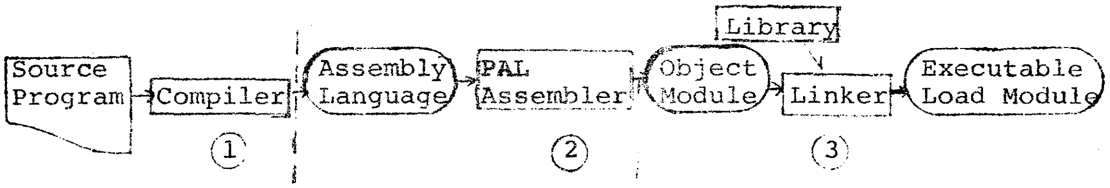


Figure 3-1

Steps in the Preparation of an Executable FORTRAN Program.

In the simplest form possible, the user could cause the compilation, assembly, and linking of a program with the commands shown in Figure 3-2. Assembly listing and a load map can be obtained with slightly different commands (see the relevant DOS manuals). Just as PAL-11R assumes .PAL extensions on input files, FORTRAN assumes .FTN extensions on input files.

Compilation

```
$RU FORTRN
FORTRAN  VØØ1B
#PROG < PROG
```

Compilation With Compilation Listing

```
$RU FORTRN
FORTRAN  VØØ1B
#PROG,LST<PROG
```

Assembly

```
$RU PAL
PAL11R  VØØ5A
#PROG < PROG
```

Linking

```
$RU LINK
LINK-11  VØØ5A
PASS1
#PROG<PROG, FORLIB/L/E
```

Figure 3-2

Example Interaction With FORTRAN System

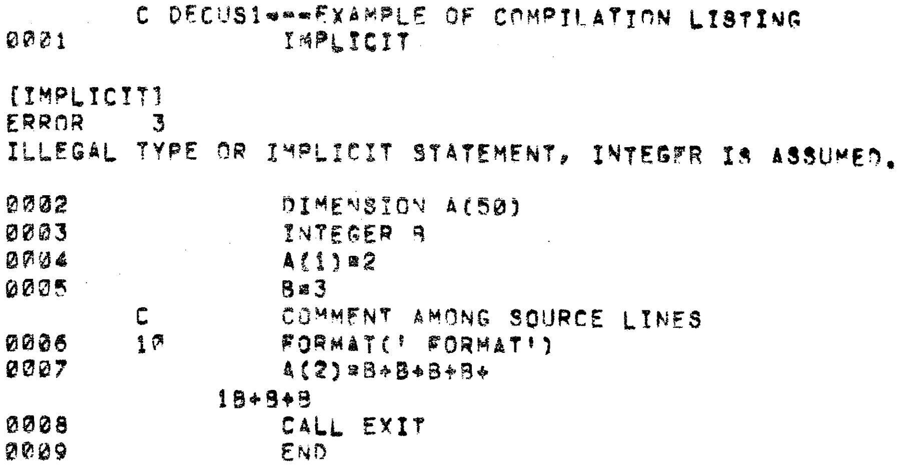
3.1.1 FORTRAN Compile Time Operation

At compile time the FORTRAN program is transformed into an assembly language program. A compilation listing can be generated. Figure 3-3 contains a compilation listing. This is useful for the following reasons:

a. Diagnostic messages are imbedded in the FORTRAN source listing. If there are any compilation errors, an error count is printed on the teleprinter.

b. All FORTRAN source statements are given sequence numbers. These numbers are referenced by any error messages printed when the program is run.

Table 3-1 contains a list of the FORTRAN Compiler error codes and their meaning.



```
         C DECUS1---EXAMPLE OF COMPILATION LISTING
0001                IMPLICIT

[IMPLICIT]
ERROR    3
ILLEGAL TYPE OR IMPLICIT STATEMENT, INTEGER IS ASSUMED.

0002                DIMENSION A(50)
0003                INTEGER B
0004                A(1)=2
0005                B=3
         C          COMMENT AMONG SOURCE LINES
0006     10         FORMAT(' FORMAT')
0007                A(2)=B+B+B+B+
               1B+B+B
0008                CALL EXIT
0009                END
```

Figure 3-3

Example of a Compilation Listing

~~Table 3-1~~

Compiler Error Codes

FORCOM.DGN

MSG NUM	MESSAGE
0	REDUNDANT CONTINUATION MARK; IT IS IGNORED
1	CONTINUATION MARK NOT IN RANGE 1 TO 9; IT IS IGNORED
2	ILLEGAL STMT. NUMBER, NON-NUMERIC CHAR, IN COLS. 1-5
3	ILLEGAL TYPE OR IMPLICIT STATEMENT, INTEGER IS ASSUMED.
4	NON-DECLARATIVE STATEMENT IN BLOCK DATA.
5	SYMBOL TABLE FULL
6	STATEMENT TOO LONG, REMAINDER DISCARDED
7	MISSING END STATEMENT, END ASSUMED
8	ILLEGAL UNARY OPERATOR, ONLY +, -, OR .NOT ALLOWED.
9	COMPILER ERROR - IMPOSSIBLE OCCURENCE
10	MISSING LEFT PARENTHESIS IN FUNCTION CALL
11	MISSING RIGHT PARENTHESIS IN FUNCTION CALL
12	ILLEGAL CHARACTER(S) TERMINATING STATEMENT
13	ILLEGAL FORMAT IN A NUMERIC CONSTANT
14	INSUFFICIENT INTERNAL COMPILER SPACE TO EVALUATE THIS CONSTANT
15	INTEGER CONSTANT TOO LARGE. REPLACED WITH LARGEST POS. VALUE.
16	ILLEGAL SYNTAX IN LIST ITEM.
17	ILLEGAL LIST ITEM TERMINATOR
18	WARNING--LOGICAL*1 IS USED IN AN EXPRESSION
19	TOO MANY SUBSCRIPTS IN AN EXPRESSION
20	MISSING ")" IN SUBSCRIPT EXPRESSION.
21	UNRECOGNIZED STATEMENT
22	ADJUSTABLE ARRAY NAME OR INDEX IS NOT A SUBPROGRAM PARAMETER
23	LIST ITEM IN DIMENSION STATEMENT LACKS DIMENSIONS.
24	ARRAY DIMENSIONS CONFLICT WITH THOSE IN AN EARLIER STATEMENT
25	MISMATCHED PARENTHESIS.
26	NON-ARRAY REFERENCE TO ARRAY ITEM.
27	CANNOT ASSIGN TO A CONSTANT
28	CANNOT ASSIGN TO A FUNCTION
29	ILLEGAL CHARACTER TERMINATING A STMT. OR POSSIBLE BAD OPERATOR
30	SUBSCRIPT ON NON-ARRAY VARIABLE
31	VARIABLE NAME MUST BE 6 CHARACTERS OR LESS.
32	ILLEGAL SUBSCRIPT IN AN ARRAY ASSIGNMENT
33	ILLEGAL OPERAND
34	TOO MANY SUBSCRIPTS OR NO CLOSING PAREN FOR SUBSCRIPT.
35	NO FUNCTION ARGUMENTS PRESENT, FUNCTION IGNORED.
36	UNRECOGNIZABLE PARAMETER IN FUNCTION CALL.
37	FUNCTION CALL MISSING A ")".
38	ILLEGAL ROUTINE NAME.
39	MISSING END STATEMENT, END IS ASSUMED.
40	SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
41	DO LIST OVERFLOW, NO MORE THAN 10 NESTED DO'S ARE ALLOWED.
42	ILLEGAL SYNTAX IN COMMON/EQUIVALENCE
43	TABLE OVERFLOW IN COMMON/EQUIVALENCE
44	DUMMY VARIABLE USED IN COMMON
45	VARIABLE ALREADY IN COMMON, CANNOT BE RE-DEFINED.
46	ILLEGAL DO STATEMENT SYNTAX
47	DO CONTROL VARIABLE NOT SIMPLE INTEGER VARIABLE.
48	BAD DO PARAMETER

```
~~ 49 BAD STEP VALUE IN DO, IT IS ASSUMED TO BE 1.~~
 50 ILLEGAL CONSTANT IN PAUSE/STOP
 51 ILLEGAL OR MISSING STATEMENT LABEL NUMBER
 52 ILLEGAL  SYNTAX IN GOTO/ASSIGN STATEMENT
 53 ILLEGAL DO STATEMENT NESTING
 54 ROUTINE NAME CANNOT BE A NUMERIC CONSTANT
 55 SUBROUTINE OR FUNCTION STATEMENT NOT FIRST STATEMENT OF ROUTINE
 56 ILLEGAL PARAMETER IN SUBROUTINE OR FUNCTION LIST
 57 TOO MANY PARAMETERS IN ROUTINE LIST
 58 ILLEGAL ARGUMENT LIST IN EXTERNAL.
 59  MISMATCHED PARENTHESIS IN FORMAT
 60 MISSING COMMA OR ) IN COMMON/EQUIVALENCE.
 61 MISSING ( IN COMMON/EQUIVALENCE
 62 DUMMY ARGUMENT USED IN EQUIVALENCE
 63 INCONSISTENT EQUIVALENCE
 64 TWO OR MORE COMMON ITEMS ARE EQUIVALENCED
 65 I/O UNIT IS NOT SIMPLE INTEGER VARIABLE OR CONSTANT.
 66 ARRAY OR FUNCTION NAME IS NOT ALLOWED AS A UNIT IN I/O STMT.
 67 ILLEGAL SYNTAX IN I/O STATEMENT.
 68 MISSING ARGUMENT IN FIND.
 69 ILLEGAL RECORD DESIGNATOR IN RANDOM ACCESS READ/WRITE.
 70 MISSING RIGHT PARENTHESIS IN I/O CALL
 71 ILLEGAL FORM OF END= AND/OR ERR=
 72 ILLEGAL FORM FOR LIST ITEM IN I/O STATEMENT.
 73 ILLEGAL SYNTAX OF REWIND, BACKSPACE OR ENDFILE
 74 NON-INTEGER PARAMETER IN REWIND, BACKSPACE, OR ENDFILE
 75 ILLEGAL H CONSTANT IN FORMAT
 76 H CONSTANT COUNT TOO BIG.
 77 SYNTAX ERROR IN IMPLICIT STATEMENT
 78 HOLLERITH CONSTANT IMPROPERLY TERMINATED BY END OF LINE.
 79 .NOT. MAY BE USED AS A UNARY OPERATOR ONLY
 80 EXPONENT MAY NOT BE LOGICAL*1, LOGICAL*2 OR COMPLEX
 81 INTEGER**REAL OR INTEGER**COMPLEX NOT ALLOWED.
 82 COMPLEX**REAL OR COMPLEX**DOUBLE NOT ALLOWED
 83 IMPROPER LABEL SYNTAX IN IF STATEMENT
 84 ANYTHING **COMPLEX NOT ALLOWED
 85 MISSING COMMA IN READ OR PRINT
 86 INCORRECT SYNTAX IN DEFINEFILE STATEMENT
 87 COMPLEX ARITHMETIC NOT YET SUPPORTED.
 88 ARRAY IS TOO LARGE.
 89 ILLEGAL ROUTINE NAME
 90 ILLEGAL DO SPECIFICATION IN I/O STATEMENT
 91 ILLEGAL LIST IN IMPLIED DO
 92 ILLEGAL FORMAT SPECIFICATION IN I/O STATEMENT
 93 SYNTAX ERROR IN EXPRESSION OF ASF
 94 MISSING, OR ) IN ASF
 95 MISPLACED = IN ASF
 96 ILLEGAL OR MISSING DUMMY ARGUMENT IN ASF
 97 SUBSCRIPTS OUT OF BOUNDS IN DATA OR EQUIVALENCE.
 98 ILLEGAL EXTENSION OF COMMON ORIGIN BY EQUIVALENCE.
 99 OPENING "/" MISSING FROM DATA GROUP.
100 WARNING - UNEQUAL NUMBER OF VARIABLES AND CONSTANTS.
101 DATA NOT ALLOWED IN COMMON EXCEPT IN "BLOCKDATA".
102 SUBSCRIPTS ON UNDIMENSIONED ELEMENT IN DATA.
103 ADJUSTABLE ARRAY NOT ALLOWED IN DATA.
104 PRESETTING NAMED COMMON ALLOWED ONLY IN "BLOCKDATA".
105 ILLEGAL FORM FOR CONSTANT IN DATA.
106 ILLEGAL REPEAT COUNT.
107 MISMATCHED DATA TYPES.
108 DATA MUST FOLLOW ALL OTHER DECLARATIVES.
109 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
110 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
```

```
111 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
112 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
113 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
114 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
115 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
116 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
117 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
118 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
119 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
120 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
121 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
122 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
123 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
124 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
125 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
126 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
127 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
128 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
129 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
130 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
131 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
132 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
133 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
134 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
135 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
136 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
137 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
138 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
139 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
140 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
141 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
142 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
143 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
144 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
145 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
146 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
147 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
148 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
149 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
150 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
151 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
152 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
153 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
154 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
155 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
156 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
157 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
158 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
159 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
160 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
161 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
162 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
163 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
164 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
165 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
166 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
167 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
168 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
169 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
170 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
171 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
172 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
173 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
```

```
174 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
175 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
176 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
177 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
178 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
179 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
180 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
181 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
182 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
183 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
184 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
185 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
186 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
187 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
188 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
189 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
190 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
191 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
192 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
193 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
194 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
195 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
196 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
197 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
198 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER
199 SYSTEM ERROR -- NO DIAGNOSTIC MESSAGE HAS THIS NUMBER

200MESSAGES ALLOCATED;   108 MESSAGES INPUT
```

~~As an example of the usefulness of the sequence numbers, the output~~ in Figure 3-4 shows a compilation listing of a program which intentionally causes an overflow. At run time the FORTRAN Object Time System generates a run time diagnostic indicating that error number 14 occurred. Error 14 indicates that an integer multiplication resulted in a product $>2^{15}-1$ or $<-2^{15}$. The diagnostic appears 3 times because class 3 errors have a maximum occurrence count of three before the program is terminated (this is explained further in the next section.

Figure 3-4 is an example of what is known as the trace-back feature in the FORTRAN system. At run time, diagnostics are printed by OTS with a trace of the flow of control within the user-written code. Following the error code are printed the headings NAME and SEQ below which are the names of the routines through which the call is being traced and the sequence number of the specific line in which the error occurred (or from which that subroutine was called). The first name and sequence number at the top of the list is the error location, subsequent names and numbers refer to the path through which the program reached that point.

The example in Figure 3-4 shows the same error message being printed three times, indicating that after the third occurrence of the error, the program was terminated.

3.1.2 FORTRAN Assembly Time Operation

At assembly time the assembly language program is transformed into an object module. The user can generate an optional assembly listing and symbol table listing through the Assembler.

The use of the Assembler for processing FORTRAN programs does not differ in any way from the normal use of the Assembler.

```
ØØØ1             J=2                  }
ØØØ2             DO 1Ø I=1,1ØØØ       }
ØØØ3             J=J**2               }  Compilation Listing
ØØØ4      1Ø     CONTINUE             }
ØØØ5             END                  }
```

```
$RU DECS16.LDA           }
                         }
                         }
FORTØØ3Ø14               }
NAME     SEQ             }
MAIN.    ØØØØ3           }
FORTØØ3Ø14               }
NAME     SEQ             }   Run Time Diagnostic
MAIN.    ØØØØ3           }
FORTØØ3Ø14               }
NAME     SEQ             }
MAIN.    ØØØØ3           }
                         }
I351  ØØØØØ1             }
                         }
$                        }
```

Figure 3-4
Example of Run Time Diagnostic

3.1.3 Link Time Operation

At linking time, the object module is linked with any subroutines or functions that the user has prepared, and the FORTRAN Library is searched for any necessary arithmetic functions, I/O routines, error handling routines, or math routines. A load map can be obtained which shows where in core the various programs have been placed, their lengths, entry points, and use of COMMON storage.

The FORTRAN Library is composed of the following:

a. Math routines, including all standard FORTRAN functions plus other arithmetic routines needed to do arithmetic operations (e.g., floating point);

b. Miscellaneous utility routines (PDUMP, SETERR, SETFIL);

c. I/O Routines, which handle the various types of FORTRAN I/O;

d. Error handlina routines, which handle arithmetic errors, I/O errors, and system errors;

e. Miscellaneous (Polish) routines required by the compiled code: $SBS, $DOEND, $POP.

The Library was designed as a large number of small pieces so that non-necessary routines could be omitted at link time. Thus, if the user does only sequential formatted I/O, none of the random access routines are linked to his program.

3.1.4 FORTRAN Library Usage

With the DOS Librarian the user can construct his own libraries of routines which he then searches at link time (the user is constrained to search all of his own libraries before searching the FORTRAN System Library).

Users should not add subroutines or functions to the DEC-supplied FORTRAN Library. Instead, users should create their own libraries, using the Librarian. Thus, if MATLIB is a user library containing matrix manipulation routines, and the user writes a program (PROG) which uses routines from MATLIB, a command string to the Linker might be:

```
#PROG,LP:<PROG,MATLIB/L,FORLIB/L/E
```

Do not add routines to FORLIB or EAELIB. Many routines in these libraries are position dependent and the insertion of new routines could result in undefined global references when linking FORTRAN programs. Similarly, the deletion or rearrangement of routines in FORLIB and EAELIB is likely to cause problems unless the user is familiar with the ordering dependencies of the FORTRAN Library.

3.1.5 FORTRAN Run Time Operation

FORTRAN run time errors are grouped within six error classes accordinq to the nature of the error, as follows:

CLASS Ø Includes certain severe errors, such as: no space to do I/O, subroutine directly or indirectly references itself.

CLASS 1 Includes I/O errors with respect to parity, checksum, files or devices.

CLASS 2 Format statement errors.

CLASS 3 Includes arithmetic errors and overflow conditions.

CLASS 4 Includes argument errors in function calls: SQRT called with ARG<Ø, ALOG called with ARG≤Ø.

CLASS 5 Includes errors with respect to exponential FORMAT numbers.

CLASS 6 Conversion or undefined error.

Each class of errors has a maximum occurrence count (usually equal to the class number). If the maximum occurrence count is achieved for any class during a run, the job exits. Normally, an error message is logged on the teleprinter whenever any error occurs.

The list of run time error codes and messages is found in Table 3-2. Through the use of the SETERR subroutine, which is documented in the FORTRAN manual, the user can modify the maximum error count for any error class. The count for a class can be set to any positive integer value, or the user can specify:

Ø log error and ignore, continue execution

-1 ignore error, do not log, continue execution

-2 exit to Monitor, do not log, continue execution

-3 immediate abort, results in fatal Monitor error FØ3Ø

Table 3-2

Run Time Diagnostics

The diagnostics are printed in the form

FORTØØXYYY

where X is the class of the error and YYY is the particular error code.

```
FORTØØØØØØ INVALID ERROR CALL
FORTØØØØØ1 NO SPACE TO DO I/O
FORTØØØØØ2 SUBROUTINE DIRECTLY OR INDIRECTLY REFERENCES ITSELF
FORTØØØØØ3 SYSTEM ERROR NO DIAGNOSTIC MESSAGE ASSIGNED

FORTØØ1ØØØ VALUE OUT OF BOUNDS (COMPUTED OR ASSIGNED GO TO)
FORTØØ1ØØ1 DEVICE PARITY
FORTØØ1ØØ2 CHECKSUM / PARITY ERROR OR END OF DATA ERROR (RANDOM)
FORTØØ1ØØ3 UNDIAGNOSABLE I/O ERROR
FORTØØ1ØØ4 EOF / EOM
FORTØØ1ØØ5 UNABLE TO ALLOCATE CONTIGIOUS FILE
FORTØØ1ØØ6 DEFINE FILE NOT DONE (RANDOM)
FORTØØ1ØØ7 DEFINE FILE DONE (NOT RANDOM)
FORTØØ1ØØ8 INVALID PROTECT CODE
FORTØØ1ØØ9 FILE DOES NOT EXIST / OR IS ALREADY OPEN
FORTØØ1Ø1Ø UNABLE TO OPEN
FORTØØ1Ø11 COMPATABILITY ERROR
FORTØØ1Ø12 INVALID DEVICE NUMBER
FORTØØ1Ø13 INVALID RECORD NUMBER (RANDOM)

FORTØØ2ØØØ FORMAT HAS ITEMS AND NO CONVERSION SPECS
FORTØØ2ØØ1 PARENTHESES NESTING TOO DEEP IN FORMAT
FORTØØ2ØØ2 FORMAT SYNTAX ERROR
FORTØØ2ØØ3 REFERENCE OUTSIDE OF RECORD BOUNDARIES

FORTØØ3ØØØ SYSTEM ERROR NO DIAGNOSTIC MESSAGE ASSIGNED
FORTØØ3ØØ1 $ADD EXPONENT OVERFLOW
FORTØØ3ØØ2 $ADR EXPONENT OVERFLOW
FORTØØ3ØØ3 $DVD DIVIDE CHECK
FORTØØ3ØØ4 $DVD EXPONTENT OVERFLOW
FORTØØ3ØØ5 $DVI DIVIDE BY Ø
FORTØØ3ØØ6 $DVR EXPONENT OVERFLOW
FORTØØ3ØØ7 $DVC DIVIDE CHECK
FORTØØ3ØØ8 $DVR DIVIDE BY Ø
FORTØØ3ØØ9 $MLC EXPONENT OVERFLOW
FORTØØ3Ø1Ø $MLD EXPONENT OVERFLOW
FORTØØ3Ø11 $NEG EXPONENT OVERFLOW
```

~~Table 3-2~~

Run Time Diagnostics (Cont'd.)

```
FORTØØ3Ø12 $MLR EXPONENT OVERFLOW
FORTØØ3Ø13 SYSTEM ERROR NO DIAGNOSTIC MESSAGE ASSIGNED
FORTØØ3Ø14 $MLI 2**15-1 < PRODUCT OR < -2**15
FORTØØ3Ø15 $PWII BASE = Ø , EXPONENT <=Ø
FORTØØ3Ø16 $PWDI BASE = Ø , EXPONENT <= Ø
FORTØØ3Ø17 $PWDD BASE = Ø , EXPONENT < Ø
FORTØØ3Ø18 $PWDD BASE < Ø , EXPONENT <> Ø
FORTØØ3Ø19 $PWRR BASE = Ø , EXPONENT <> Ø
FORTØØ3Ø2Ø $PWRR BASE < Ø , EXPONENT <> Ø
FORTØØ3Ø21 $PWRI BASE = Ø , EXPONENT <= Ø
FORTØØ3Ø22 $RI  2**15-1 < INTEGER OR < -2**15
FORTØØ3Ø23 $DR  EXPONENT OVERFLOW

FORTØØ4ØØØ SYSTEM ERROR NO DIAGNOSTIC MESSAGE ASSIGNED
FORTØØ4ØØ1 SYSTEM ERROR NO DIAGNOSTIC MESSAGE ASSIGNED
FORTØØ4ØØ2 DEXP USER EXPONENT > 87.
FORTØØ4ØØ3 DLOG ARGUMENT < = Ø
FORTØØ4ØØ4 DSQRT ARGUMENT < Ø
FORTØØ4ØØ5 EXP  USER EXPONENT > 87.
FORTØØ4ØØ6 SYSTEM ERROR NO DIAGNOSTIC ASSIGNED
FORTØØ4ØØ7 IABS ABS(X) > 2**15 - 1
FORTØØ4ØØ8 IDIM RESULT > 2**15 - 1 OR < - 2**15
FORTØØ4ØØ9 ISIGN RESULT > 2**15 - 1
FORTØØ4Ø1Ø ALOG X < = Ø
FORTØØ4Ø11 SQRT X < Ø
FORTØØ4Ø12 SNGL EXPONENT OVERFLOW ON ROUND
FORTØØ4Ø13 RANDU WRONG NUMBER OF ARGUMENTS
FORTØØ4Ø14 PDUMP WRONG NUMBER OF ARGUMENTS
FORTØØ4Ø15 CSQRT UNDERFLOW

FORTØØ5ØØØ SYSTEM ERROR NO DIAGNOSTIC ASSIGNED
FORTØØ5ØØ1 $ADD EXPONENT UNDERFLOW
FORTØØ5ØØ2 $ADR EXPONENT UNDERFLOW
FORTØØ5ØØ3 $DVR EXPONENT UNDERFLOW
FORTØØ5ØØ4 DEXP USER EXPONENT < -88.7
FORTØØ5ØØ5 EXP  USER EXPONENT < -88.7
FORTØØ5ØØ6 $MLD EXPONTENT UNDERFLOW
FORTØØ5ØØ7 $MLR EXPONTENT UNDERFLOW
FORTØØ5ØØ8 $DVD EXPONTENT UNDERFLOW
FORTØØ5ØØ9 SYSTEM ERROR NO DIAGNOSTIC ASSIGNED

FORTØØ6ØØØ CONVERSION ERROR
FORTØØ6ØØ1 SYSTEM ERROR NO DIAGNOSTIC ASSIGNED
```

3.2 I/O CONSIDERATIONS

~~A FORTRAN program doing output to the disk would use the normal FORTRAN~~ output statement. Since the disk is a directory structured device, FORTRAN OTS invents a filename for the data being written. The filename is of the form: FORØØn.DAT where n is the logical unit number used in FORTRAN READ or WRITE statements.

It is recognized that the user would like more control over the actual name of the file and the ability to control such parameters as the protection of the file and its user identification code. Two levels of control are available: logical assignments and the SETFIL subroutine.

The user can perform DOS logical assignments before running his linked program. If the user had a program doing a WRITE on unit 1, and wished the file to have the name SPEC, he could type:

```
          $ASSIGN DF:SPEC,1
```

and the WRITE statements referencing unit 1 would all write onto the file named SPEC on the disk. Such as ASSIGN can also be used to force the FORTRAN program to READ from an existing file with a name other than FORØØn.DAT.

Note also that:

```
          $ASSIGN LP:,1
```

would cause the output to go to the line printer rather than the disk.

Where the user wishes to build such assignments into his program permanently a FORTRAN callable subroutine, SETFIL is available which allows the user to build the device name, filename, extension, protection, and UIC (user identification code) specifications into his program. See Figure 3-5 for a description of SETFIL. This dispenses with the need for operator intervention via use of the ASSIGN command at run time. Of course, a file specification setup with a call to SETFIL can be overridden with an ASSIGN command, where desired.

FORTRAN calls to SETFIL are formatted as follows:

```
CALL SETFIL (I1,F1,I2,F2,I3,I4,I5,I6,I7,I8)
```

where:

I1 = logical device number (integer constant)

F1 = file name and extension (ASCII)

I2 = error value variable (integer), is set to -1 if I6≠1 or 2 or if I6=1 and I7 and I8 are not specified.

F2 = physical device name (ASCII)

I3 = unit number of physical device, if any (integer)

I4 = UIC (integer constant)

I5 = protection code (integer constant), I5 must be the decimal equivalent of the desired octal code (if I5=64, the protection code is 200).

I6 = allocate file value (integer constant),
if I6=2 a contiguous file is allocated for random I/O,
if I6=1 a contiguous file is allocated for unformatted I/O.

I7 = logical record length in words (integer constant), required if I6=1, otherwise I7 is ignored.

I8 = number of records to allocate (integer constant), required if I6=1, otherwise I8 is ignored.

The FORTRAN OTS allocates a contiguous file only where the user has specifically requested one through SETFIL (I6 parameter).

Figure 3-5
SETFIL Subroutine

3.3 DEBUGGING FORTRAN PROGRAMS

The use of traditional FORTRAN debugging techniques such as PDUMP are recommended for development of FORTRAN applications (see Appendix C of the FORTRAN Manual).

System error reporting and traceback information is of significant value in debugging.

It requires considerable experience to successfully use ODT with a FORTRAN program. ODT was not intended to handle the problems of FORTRAN debugging, and it is especially out of its element when trying to debug threaded code.

Remember that the "threaded code" generated by the Compiler is a sequence of addresses, rather than machine instructions. ODT breakpoints can be placed ONLY on machine instructions. Thus the user is constrained to use breakpoints only in places where the code leaves Polish Mode*, e.g., a subroutine or function call; alternatively, breakpoints can be placed in the Polish routines themselves.

There is one significant inconvenience associated with putting breakpoints in Polish routines. Polish routines are usually called from several places in a program and when the breakpoint in a Polish routine is encountered, the user must look at R4 to find the address from which the routine was called.

3.4 FORTRAN Device Table

DVTBØ6.PAL is the source file containing the FORTRAN Device Table. In order to add a device to the table:

a. Enter the address of the device entry in the device table entry vector (see the listing in Appendix A of this document).

b. Set the word at $DEVTB to reflect the number of entries in the entry vector (number of devices available to FORTRAN).

c. Insert the new entry in the table.

To delete a device from the table:

a. Delete the address of the device entry in the device table entry vector.

b. Set the word at $DEVTB to reflect the number of entries in the entry vector.

c. Delete the entry from the table.

The default physical device for a table entry can be changed by modifying the second word of the entry.

The default filename and extension for a table entry can be changed by modifying words four through six of the table entry.

Random access files are contiguous files. They can be allocated by OTS through SETFIL or through PIP. Contiguous files are described in the DOS manual (section 2.7).

3.5 FORTRAN ERRORS DETECTED ONLY AT ASSEMBLY TIME

Missing formats or statement numbers are detected at assembly time. A format number starts with a "$" and a statement number starts with a ".", so it is easy to recognize which form is missing by looking at the number (in the PAL listing). A "U" diagnostic is issued in either case.

Multiple statement and/or format numbers will cause a "D" error message to be placed at the offending line(s) and an "M" error message to be placed on lines referencing them.

Chapter 4

Corrections to the FORTRAN Manual

The following changes should be made to the FORTRAN IV Manual (DEC-11-KFDA-D):

Page no.	Corrections
V I - IV	Add: 4.7 END Statement 4-8
Part I	
2-4	Insert:

2.1.8 Hexadecimal Constants

A hexadecimal constant is a string of from one to six hexadecimal digits. The hexadecimal digits are as follows:

Decimal		Hexadecimal
Ø	=	Ø
1	=	1
2	=	2
3	=	3
4	=	4
5	=	5
6	=	6
7	=	7
8	=	8
9	=	9
10	=	A
11	=	B
12	=	C
13	=	D
14	=	E
15	=	F
16	=	10

For example, $1ØØ_{16} = 256_{1Ø}$.

The use of hexadecimal constants is preceded by the letter H. For example:

```
H24E
H8A5.B2
```

A hexadecimal constant is valid only in the context of the statements: DATA, PAUSE, and STOP. The maximum value which can be expressed as a hexadecimal constant is FFFF.

Page no. Corrections

2-6 In the list of operator precedence, the first operator should be unary minus.

4-8 Add:

4.7 END Statement

The END statement is of the form:

END

This statement is the necessary final statement in all main and subprograms. Program compilation is terminated when the END statement is encountered.

An END statement forces a CALL EXIT operation in a main program.

An END statement in a function or subroutine forces a return to the calling program if a RETURN statement is not present.

If an input file is exhausted before an END statement is encountered, a diagnostic is printed and an END statement forced.

6-1
6-7 Disregard BYTE (LOGICAL*1) type declaration for this version of the FORTRAN Compiler.

6-5 Description of DATA statement should look as follows:

DATA var list$_1$/values list$_1$/,var list$_2$/value list$_2$/,...

remove parentheses from var list in description.

6-6 9th line from bottom of page should read:

```
DATA A(1), A(2), A(3)/3*Ø./
```

6-8 Insert:

6.5 Ordering of Specification Statements

Generally speaking, the specification statements are all grouped together at the beginning of a program according to the following rules:

a. A DATA statement, where used, is the last of the specification statements. It must follow statements which might affect it, such as EQUIVALENCE, TYPE, and DIMENSION.

b. Any arithmetic statement functions to be defined in a program must occur following any and all specification statements.

7-3 Add to end of Section 7.1.12:

An END statement must be the last statement of every function subprogram and causes a return to the calling program where a RETURN statement is not present.

7-3 Add to end of Section 7.2:

An END statement must be the last statement of every subroutine and causes a return to the calling program if no RETURN statement is present.

Part II

3-2 Device Assignments are as follows:

FORTRAN DEVICE #	DOS File Name	Actual Device Mnemonic
1	FORØØ1.DAT	DTØ:
2	FORØØ2:DAT	DFØ:
3	FORØØ3.DAT	DFØ:
4	FORØØ4.DAT	PR:
5	FORØØ5.DAT	LP:
6	FORØØ6.DAT	KB:
7	FORØØ7.DAT	DFØ:
8	FORØØ8.DAT	DFØ:

Add to end of statement summary:

TYPE	TYPE V_1, V_2, V_3,...	where the variables V_n are assigned to be part of the indicated type.
END	END	Cease program compilation; equivalent to CALL EXIT in main program or RETURN in subprograms.

C-3 Delete RAN, Random Number function, from the function summary. The proper name for the random number generator is RANDU and it is a subroutine, belonging in the summary on page C-5.

CHAPTER 5

FORTRAN LIBRARY FUNCTIONS

This chapter contains a brief outline of the OTS library of FORTRAN functions which involve approximations. "Floating point" means single precision, 2-word, floating point format with a 24-bit fraction and an 8-bit binary exponent. "Double precision" means 4-word, floating point format with a 56-bit fraction and an 8-bit binary exponent. The values of the coefficients used in the various approximations may be found at the cited parts of the following references:

(1) Computer Approximations, by J. F. Hart et al, John Wiley & Sons, 1968.

(2) Approximation for Digital Computers, by C. Hastings et al, Princeton University Press, 1955.

(3) PDP-11 Paper Tape Software Programming Handbook, DEC-11-GGPB-D, Digital Equipment Corporation.

All OTS FORTRAN functions are called using the standard sequence:

```
        JSR     R5,NAME
        BR      RTN
        .WORD   #ARG1,...,#ARGn
RTN:
```

The result is returned in RØ-R1 for floating point and RØ-R3 for a double precision function. Some FORTRAN functions call other single argument functions in the course of their computation. In order for them to be reentrant, these calls are made via the routine $FCALL.

```
        MOV     ARGUMENT ADDRESS, R5
        MOV      #FUNCTION NAME, R4
        JSR     PC,$FCALL
```

$FCALL calls the FORTRAN function whose address is in R4 with the argument whose address is in R5. Control is returned to the instruction following the JSR with the function result in RØ-R1 for floating point or RØ-R3 for double precision.

5.1 ALOG(X), Floating Point Natural Logarithm

If $X \leq Ø$ call error
Let $X = y \cdot 2^a$ where $1/2 \leq y < 1$
Let $Q = (y\sqrt{2} - 1)/(y\sqrt{2} + 1)$
Then $\ln(X) = a \cdot \ln(2) + \ln(y)$
$ALOG(X) = a \cdot \ln(2) = \ln(\sqrt{2}) + Q\sum_{Ø}^{3} c_i Q^{2i}$

where the c_i are drawn from Hart #2662. The relative error is $\leq 1Ø^{-9.9}$.

5.2 ALOG1Ø(X), Floating Point Common Logarithm

Computed as $\log_{1Ø}(e) \cdot ALOG(X)$.

5.3 ATAN(X), Floating Point Arctangent

If $X < Ø$, $ATAN(X) = -ATAN(-X)$
If $|X| > 1$, $ATAN(|X|) = \pi/2 - ATAN\ (1/|X|)$
If $|X| > \tan \pi/12$, $ATAN(X) = \pi/6 + ATAN\ ((X\sqrt{3} - 1)/(X + \sqrt{3}))$
For $|X| \leq \tan \pi/12$, $ATAN(X) = X\sum_{Ø}^{4} c_i X^{2i}$

where the c_i are drawn from Hart #4941. The relative error is $\leq 1Ø^{-9.5}$.

5.4 ATAN2(X,Y), Two Argument Floating Point Arctangent

If $Y = Ø$, or $X/Y > 2^{25}$, $ATAN2(X,Y) = \pi/2\ (\text{sign } X)$.
If $Y > Ø$, and $X/Y \leq 2^{25}$, $ATAN2(X,Y) = ATAN(X/Y)$
If $Y < Ø$, and $X/Y \leq 2^{25}$, $ATAN2(X,Y) = \pi \cdot \text{sign}X + ATAN(X/Y)$

5.5 DATAN(X), Double Precision Arctangent

The analysis is the same as in that for ATAN(X) except that the polynomial approximant is of degree 8. The coefficients are drawn from Hart #4945. The relative error is $\leq 1Ø^{-16.8}$.

5.6 DATAN2(X,Y), Two Argument Double Precision Arctangent

The rules for DATAN2 are the same as those for ATAN2 except that the DATAN is used in all computations.

5.7 DLOG(X), Double Precision Natural Logarithm

The analysis for DLOG is the same as that for ALOG except that the polynomial in Q^2 is of degree 6. The c_i are drawn from Hart #2665. The relative error is $\le 1\text{Ø}^{-16.5}$.

5.8 DLOG1Ø(X), Double Precision Common Logarithm

Computed as $\log_{1\text{Ø}}(e) \cdot \text{DLOG}(X)$.

5.9 DSQRT(X), Double Precision Square Root

If $X < \text{Ø}$ call error
Let $X = A \cdot 2^B$ where $1/2 \le A < 1$
Let $Y_{\text{Ø}} = 2^{B/2} \cdot (1/2 + A/2)$ if B is even
or
$Y_{\text{Ø}} = 2^{(B+1)/2} \cdot (1/4 + A/2)$ if B is odd,

a transformation requiring only two instructions. Starting with $y_{\text{Ø}}$, four Newton-Raphson iterations are performed.

$$y_n+1 = 1/2(y_n + x/y_n).$$

The relative error is $< 1\text{Ø}^{-17}$.

5.10 DSIN(X), Double Precision Sine

Let y = Integer (4 ·fraction (X/2π))
Let V = Fraction (4· fraction (X/2π))
Then DSIN(X) = P(Vπ/2) if y=Ø
= P((1-V)π/2) if y=1
= P(-Vπ/2) if y=2
= P((V-1)π/2) if y=3

where $\sin(V\pi/2) \approx P(V\pi/2) = V\sum_{\text{Ø}}^{8} c_i V^{2i}$ for $-1 \le V \le 1$

The c_i are drawn from Hart #3345. The relative error is $\le 1\text{Ø}^{-18.6}$.

5.11 DCOS(X), Double Precision Cosine

Computed as DSIN(X + π/2).

5.12 DEXP(X), Double Precision Exponential

```
If $X > 87$ call overflow
If $|X| < 2^{-60}$, $DEXP(X) = 1$
If $X < -88.7$, $DEXP(X) = 0$
Let $y$ = Integer $(X \cdot \log_2(e))$
Let $V = 16 \cdot$ Fraction $(X \cdot \log_2(e))$
Let $w = 1/16 \cdot$ Fraction$(V)$
$DEXP = 2^y \cdot 2^w \cdot 2^{\text{Integer } (V)/16}$ where $0 \leq w < 1/16$.
```

Powers of $2^{1/16}$ are obtained from a table.

$2^w \approx \dfrac{P(w^2)+wQ(w^2)}{P(w^2)-wQ(w^2)}$ where P and Q are first

degree polynomials in w^2.

The coefficients of P and Q are drawn from Hart #1121.

The relative error is $\leq 10^{-16 \cdot 4}$.

5.13 EXP(X), Floating Point Exponential

```
If $X > 87$, call overflow
If $|X| < 2^{-28}$, $EXP(X) = 1$.
If $X < -88.7$, $EXP(X) = 0$.
Let $y$ = Integer $(X \cdot \log_2(e))$
Let $V$ = Fraction $(X \cdot \log_2(e))$
Let $w = 1/2 \ln(2) \cdot V$   where $|w| \leq \ln(2)/2$
```

Then $EXP(X) = 2^y \cdot (e^w)^2$

where $e^w \approx 1 + \cfrac{2 \cdot w}{c_1 - w - \cfrac{c_2}{c_3 + w^2}}$

The c_i are drawn from DEC-11-GGPB-D, page 7-23. The relative error is $\leq 10^{-10}$.

5.14 SIN(X), Floating Point Sine

The analysis is the same as that for DSIN(X). The polynomial approximant used is of degree 4 and the coefficients are drawn from Hastings, sheet 16. The relative error is $\leq 2 \cdot 10^{-8}$.

5.15 COS(X), Floating Point Cosine

Computed as SIN(X + $\pi/2$).

5.16 SQRT(X), Floating Point Square Root

The analysis is the same as that for DSQRT(X) except that only three iterations are performed. The relative error is $\leq 10^{-8}$.

5.17 TANH(X), Floating Point Hyperbolic Tangent

If $|X| \geq 16$, TANH(X) $\approx 1 \cdot$ sign(X)
otherwise

let $y = \text{EXP}(2 \cdot X)$

$\text{TANH}(X) = (y-1)/(y+1)$.

~~Appendix A~~

FORTRAN Device Table Listing

```
        .TITLE  DVTB06
        .GLOBL  $DEVTB
        .CSECT
;
;       $DEVTB  V006A
;
; COPYRIGHT 1971, DIGITAL EQUIPMENT CORPORATION, MAYNARD,MASS
;
;THESE ARE THE FORTRAN DEVICE TABLE ENTRIES
;WITH THE DEVICE TABLE HEADER ANU ENTRY VECTOR
;
        .WORD   DEVERR          ;ADDR OF ENTRY FOR ERR MSG FILE
$DEVTB: .WORD   8.              ;NUMBER OF ENTRIES IN ENTRY VECTOR
        .WORD   6.              ;DEVICE NUM OF ERROR LOGGING DEVICE
;
;THE DEVICE TABLE ENTRY VECTOR
;
        .WORD   DEV1            ;ADDR OF DEVICE 1 ENTRY
        .WORD   DEV2            ;ADDR OF DEVICE 2 ENTRY
        .WORD   DEV3            ;ADDR OF DEVICE 3 ENTRY
        .WORD   DEV4            ;ADDR OF DEVICE 4 ENTRY
        .WORD   DEV5            ;ADDR OF DEVICE 5 ENTRY
        .WORD   DEV6            ;ADDR OF DEVICE 6 ENTRY
        .WORD   DEV7            ;ADDR OF DEVICE 7 ENTRY
        .WORD   DEV8            ;ADDR OF DEVICE 8 ENTRY
;
;
;
;ENTRY 1 OF UEVICF TABLE
;
DEV1:   .WORU   0               ;LINK BLOCK PTR
        .RAD50  /DF /           ;PHYSICAL DEVICE NAME DEFAULT
        .BYTE   0               ;HOW OPEN SWITCH
        .BYTE   0               ;UNIT NUM DEFAULT
        .RAD50  /FUR/           ;DEFAULT FILE NAME
        .RAD50  /001/
        .RAD50  /DAT/           ;DEFAULT EXTENSION
        .BYTE   233             ;NO AUTO DEL, GROUP & OTHERS READ/RUN ONLY
        .BYTE   0               ;DEVICE STATUS SWITCH
        .BYTE   0               ;MODE OF I/O = FUNCN WORD (RANDOM)
        .BYTE   0               ;STATUS OF I/O
        .WORD   0               ;RECORD COUNT = BLOCK NUM (RANDOM)
        .WORD   0               ;BUFF ADDR (RANDOM)
        .WORD   0               ;BUF LEN (RANDOM)
        .WORD   0               ;ASSOCIATED VAR ADDR (FROM DEFINE FILE)
        .WORD   0               ;NUM RECORDS IN FILE (FROM DEFINE FILE)
        .WORD   0               ;RECORD LENGTH (FROM DEFINE FILE)
        .WORU   0               ;USER ID CODE
        .WORD   0               ;ERROR VAR ADDR (FROM SETFIL)
```

```
;
;
;
;ENTRY 2 OF DEVICE TABLE
;
DEV2:   .WORD   0
        .RAD50  /DF /
        .BYTE   0,0
        .RAD50  /FOR/
        .RAD50  /002/
        .RAD50  /DAT/
        .BYTE   233,0,0,0
        .WORD   0,0,0,0,0,0,0,0
;
;
;
;ENTRY 3 OF DEVICE TABLE
;
DEV3:   .WORD   0
        .RAD50  /DF /
        .BYTE   0,0
        .RAD50  /FOR/
        .RAD50  /003/
        .RAD50  /DAT/
        .BYTE   233,0,0,0
        .WORD   0,0,0,0,0,0,0,0
;
;
;
;ENTRY 4 OF DEVICE TABLE
;
DEV4:   .WORD   0
        .RAD50  /PR /
        .BYTE   0,0
        .RAD50  /FOR/
        .RAD50  /004/
        .RAD50  /DAT/
        .BYTE   233,0,0,0
        .WORD   0,0,0,0,0,0,0,0
;
;
;ENTRY 5 OF DEVICE TABLE
;
;
DEV5:   .WORD   0
        .RAD50  /LP /
        .BYTE   0,0
        .RAD50  /FOR/
        .RAD50  /005/
        .RAD50  /DAT/
        .BYTE   233,0,0,0
        .WORD   0,0,0,0,0,0,0,0
;
;
;ENTRY 6 OF DEVICE TABLE (LOGGING DEVICE NOTE PHYS DEV NAME)
;
;
DEV6:   .WORD   0
        .RAD50  /KB /
        .BYTE   0,0
        .RAD50  /FOR/
        .RAD50  /006/
        .RAD50  /DAT/
        .BYTE   233,0,0,0
        .WORD   0,0,0,0,0,0,0,0
;
;
;ENTRY 7 OF DEVICE TABLE
;
;
DEV7:   .WORD   0
        .RAD50  /DF /
        .BYTE   0,0
        .RAD50  /FOR/
        .RAD50  /007/
        .RAD50  /DAT/
        .BYTE   233,0,0,0
        .WORD   0,0,0,0,0,0,0,0
;
;
;
;ENTRY 8 OF DEVICE TABLE
;
;
DEV8:   .WORD   0
        .RAD50  /DF /
        .BYTE   0,0
        .RAD50  /FOR/
        .RAD50  /008/
        .RAD50  /DAT/
        .BYTE   233,0,0,0
        .WORD   0,0,0,0,0,0,0,0
;
;
;SPECIAL ENTRY FOR ERROR PROCESSORS MSG FILE
;
        .WORD   0                       ;LINK BLOCK ERR RTN ADDR
DEVERR: .WORD   0                       ;LINK PTR
        .RAD50  /ERR/                   ;LOG DATA SET NAME
        .BYTE   1                       ;PHYSICAL DS NAME FOLLOWS
        .BYTE   0                       ;UNIT NUM
        .RAD50  /DF /                   ;PHYSICAL DS NAME
        .WORD   0                       ;FILE BLOCK ERROR RETURN ADDR
        .BYTE   4                       ;HOW TO OPEN (OPENI)
        .BYTE   0                       ;ERROR RTN CODE
        .RAD50  /FOR/                   ;FILE NAME
        .RAD50  /TRN/
        .RAD50  /MSG/
        .BYTE   1                       ;USER ID CODE
        .BYTE   1
        .BYTE   322,0                   ;ALLOW ONLY INPUT ACCESS
        .WORD   2                       ;FUNCTION WORD (READ)
        .WORD   0                       ;BLOCK NUM
        .WORD   0                       ;BLOCK ADDR
        .WORD   0                       ;BLOCK LENGTH
;
;
```

HOW TO OBTAIN SOFTWARE INFORMATION

Announcements for new and revised software, as well as programming notes, software problems, and documentation corrections are published by Software Information Service in the following newsletters.

Digital Software News for the PDP-8 & PDP-12
Digital Software News for the PDP-11
Digital Software News for the PDP-9/15 Family

These newsletters contain information applicable to software available from Digital's Program Library, Articles in Digital Software News update the cumulative Software Performance Summary which is contained in each basic kit of system software for new computers. To assure that the monthly Digital Software News is sent to the appropriate software contact at your installation, please check with the Software Specialist or Sales Engineer at your nearest Digital office.

Questions or problems concerning Digital's Software should be reported to the Software Specialist. In cases where no Software Specialist is available, please send a Software Performance Report form with details of the problem to:

Software Information Service
Digital Equipment Corporation
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

These forms which are provided in the software kit should be fully filled out and accompanied by teletype output as well as listings or tapes of the user program to facilitate a complete investigation. An answer will be sent to the individual and appropriate topics of general interest will be printed in the newsletter.

Orders for new and revised software and manuals, additional Software Performance Report forms, and software price lists should be directed to the nearest Digital Field office or representative. U.S.A. customers may order directly from the Program Library in Maynard. When ordering, include the code number and a brief description of the software requested.

Digital Equipment Computer Users Society (DECUS) maintains a user library and publishes a catalog of programs as well as the DECUSCOPE magazine for its members and non-members who request it. For further information please write to:

DECUS
Digital Equipment Corporation
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

PDP-11 Getting FORTRAN On the Air
DEC-11-SFDB-D, August 1971

READER'S COMMENTS

Digital Equipment Corporation maintains a continuous effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback -- your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability and readability.

Did you find errors in this manual? If so, specify by page

How can this manual be improved?

Other comments?

Please state your position. ________________ Date: ________

Name: ________________ Organization: ________________

Street: ________________ Department: ________________

City: ____________ State: ____________ Zip or Country ________

-------- 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

Digital Equipment Corporation
Software Information Services
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754