PDP-11 FORTRAN-77
Documentation Supplement

AA–JQ94A–TK

June 1987

REVISION/UPDATE INFORMATION: This is a new manual.

OPERATING SYSTEM AND VERSION: See the Preface for details.

SOFTWARE VERSION: FORTRAN–77 Version 5.2

digital equipment corporation · maynard, massachusetts

First Printing, June 1987

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

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

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

The postage-paid 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:

DEC	DECsystem-10	PDT
DECUS	DECSYSTEM-20	RSTS
DIGITAL	DECwriter	RSX
PDP	DIBOL	VMS
UNIBUS	Edusystem	VT
VAX	IAS	digital
DECnet	MASSBUS	

ZK4341

CONTENTS

PREFACE

MANUAL OBJECTIVES

This manual supplements the existing FORTRAN-77 documentation set. It provides information not in the current documentation set and points out known documentation errors.

OPERATING SYSTEMS AND VERSIONS

FORTRAN-77 runs on the following operating systems and versions:

RSX-11M V4.2 or higher
RSX-11S V4.2 or higher
RSX-11M-PLUS V3.0 or higher
Micro/RSX V3.0 or higher
RSTS/E V9.2 or higher
Micro/RSTS V2.0 or higher
VAX/VMS V4.4 with VAX-11 RSX V2.2 or higher
P/OS V3.0 with PRO/Tool Kit V3.0 or higher

INTENDED AUDIENCE

This manual is intended for PDP-11 FORTRAN-77 users.

STRUCTURE OF THIS DOCUMENT

This manual contains three chapters and four appendixes.

- Chapter 1 describes known PDP-11 FORTRAN-77 documentation errors and omissions.
- Chapter 2 explains the effects of selecting certain compiler options. It also describes the optional Object Time System (OTS) modules distributed on the release media.
- Chapter 3 describes how to create and use OTS resident libraries.
- Appendix A contains additional information on character arrays and strings.

- Appendix B contains additional information on virtual arrays.
- Appendix C contains additional information on I- and D-space.
- Appendix D provides compiler task-build file listings for each system (i.e., RSX-11M/M-PLUS and RSTS/E) and a system combination (VAX to RSX).

ASSOCIATED DOCUMENTS

The following documents provide related information:

- PDP-11 FORTRAN-77 User's Guide
- PDP-11 FORTRAN-77 Language Reference Manual
- PDP-11 FORTRAN-77 Object Time System Reference Manual
- PDP-11 FORTRAN-77 Installation Guide

CHAPTER 1

DOCUMENTATION ERRORS, CLARIFICATIONS, AND ADDITIONS

This chapter lists known documentation errors and omissions in the following manuals:

- PDP-11 FORTRAN-77 User's Guide
- PDP-11 FORTRAN-77 Language Reference Manual
- PDP-11 FORTRAN-77 Object Time System Reference Manual

1.1 CORRECTIONS AND ADDITIONS TO THE USER'S GUIDE

The PDP-11 FORTRAN-77 User's Guide contains the following errors and omissions:

1. Section 1.2.4 - The following sentences supersede the sentence "Some switches are appended to the F77 Command, others to the specification for the input or output file to be affected by the switch.":

   Append switches that affect listings (e.g., /SP, /LI:n) to the specification for the input or output file they will affect. Append all other switches to the FORTRAN-77 command line.

2. Section 1.2.4 - The following paragraphs describe a new switch, the /DS switch, that should appear in the alphabetized list of compiler switches.

   Overrides the default compiler's provisions for I- and D-space. These provisions date from installation, when the decision for in-line or I- and D-space code generation was made. (I- and D-space code generation functioned as the default.)

   When the default compiler generates in-line code, the /DS switch forces I- and D-space code generation. When the default compiler generates I- and D-space code, the /-DS switch forces in-line code generation.

3. Section 1.2.5.1 - The following paragraphs supersede the information listed under /ID switches:

   Specifies that the task use I- and D-space. You can build an I- and D-space task on RSX-11M-PLUS (Version 2.0 or higher), Micro/RSX (Version 3.0 or higher), RSTS/E (Version 9.0 or higher), and Micro/RSTS (Version 2.0 or higher).

The default FORTRAN-77 compiler supports I- and D-space. This may cause some tasks to grow slightly.

The RSX-11M-PLUS and RSTS/E systems allow you to turn off this support. (The Micro systems do not provide this option.) To turn off I- and D-space support, edit the configuration file during installation. Change the DSPACE parameter in this file to 0.

Then re-task-build the compiler, following the instructions listed in the PDP-11 FORTRAN-77 Installation Guide.

4. Section 2.5.3 - There is a missing restriction: FCS does not allow error checking with the FIND statement. (Note that RMS does allow error checking with the FIND statement.)

5. Section 3.7.1 - This section should read as follows:

   Virtual arrays are limited by the number of elements, not by the available storage. The maximum number of elements in a VIRTUAL array is 32767; there is no limit to the total size of the VIRTUAL arrays a program can access. The limit on elements is 32767 because PDP-11 FORTRAN-77 requires that the number of elements in an array not exceed the size of a signed integer*2, which is 2**15-1.

   For example, the largest LOGICAL*1 VIRTUAL array is 1 byte x 32767 elements (or 32767 bytes). The largest REAL*8 VIRTUAL array is 8 bytes x 32767 elements (or 262136 bytes).

6. Section 4.1.2 (Table 4-1) - The Generic Function JIFIX is incorrectly spelled as JIFXI.

7. Section 4.2.4 - The intrinsic function IFIX does not (as implied in this section) support DOUBLE PRECISION arguments. The function INT is a more appropriate example; it does support DOUBLE PRECISION arguments.

8. Section 6.3 - An additional section, Section 6.3.1, has been added to clarify character data usage. This section appears in Appendix A.

9. Section C.3.2 - Append the following sentence to information for Error 2 (Task initialization failure), Error 3 (Odd address trap), Error 4 (Segment fault), and Error 7 (Reserved instruction trap):

   Check for incorrect usage of virtual arrays.

1.2 CORRECTIONS AND ADDITIONS TO THE LANGUAGE REFERENCE MANUAL

The PDP-11 FORTRAN-77 Language Reference Manual contains the following errors and omissions:

1. Section 2.3.4 - Complex constants cannot consist of a pair of integer constants separated by a comma and enclosed in parentheses (contrary to the first sentence in this section).

2. Section 3.1 - The following sentence should be appended to the paragraph that begins, "The expression must yield a value ...":

   Similarly, if you assign an Integer*4 variable to an Integer*2 variable, only the lower word of the Integer*4 variable is moved to the Integer*2 variable. There is no overflow check.

3. Section 5.13 - You can now use PARAMETER and SAVE statements within a BLOCK DATA subprogram.

4. Section 7.2.1.3 - You can use the format specifier [FMT=]* only for an external read or write, not for an internal read or write, as stated in this section.

5. Section 9.1.2 - There is a restriction to the ASSOCIATEVARIABLE keyword. It cannot be a dummy argument to the routine in which the OPEN statement appears.

6. Section 9.1.6 - There is a restriction on using the CARRIAGECONTROL keyword with the OPEN statement. You should use the CARRIAGECONTROL keyword in the OPEN statement only when you create a new file. If you open an old file that has a carriage control attribute different from the one you specified with the CARRIAGECONTROL keyword, the old file retains the carriage control attribute with which it was created.

   The attribute that you specified with the CARRIAGECONTROL keyword does not override the carriage control attribute specified in the header of the old file. You will receive no warning.

   If in doubt as to the carriage control format in an existing file, open it using the USEROPEN keyword. (See Section 2.3.12 in the User's Guide for more information.) The USEROPEN keyword causes an external MACRO routine to open the file; this routine can access the file header including the carriage control attribute.

1.3 CORRECTIONS AND ADDITIONS TO THE OTS REFERENCE MANUAL

The PDP-11 FORTRAN-77 Object Time System Reference Manual contains the following errors and omissions:

1. Section 3.1 - The example in this section should include the following line:

```
.GLOBL $FIO ; F77 FORMAT processor
```

   A corresponding note (5.) should be added to the list of notes closing this section:

   The $FIO reference allows the MACRO program to call an internal FORTRAN OTS routine; this routine handles format processing.

2. Section 6.2.1 - The INITIALSIZE keyword is incorrectly spelled as INITIALIZE.

3. Section 7.1.1 - The following note should be appended to this section.

NOTE

Since requesting additional memory requires RQMEM$ to issue the EXTK$ directive to extend tasks, tasks built with memory-resident overlays may fail (if a task extension is requested, either by the user task or by the OTS). See the RSX-11M/M-Plus Task Builder Manual for more information.

4. Section 10.2.3 - Insert the following information at the beginning of this section:

This section describes virtual array initialization, virtual array access, and use of virtual arrays with I- and D-space. For more information on virtual arrays, see the FORTRAN-77 User's Guide and the FORTRAN-77 Language Reference Manual.

Virtual arrays are assigned space outside a program's normal virtual address space. When a task is initiated, the OTS initializes virtual arrays by performing the following steps:

1. Determine which D-space Active Page Register (APR) to use to map the virtual array area by examining the APR usage bit map set by the Task Builder.

2. Set up window parameters (window size=8KB, region ID=0, write access, map, 32-word alignment, D-space, no send/receive buffer).

3. Create an address window using the CRAW$S directive.

4. Set up the window boundaries.

5. Section 10.2.3 - Append the material in Appendix B to this section.

6. Section 11.1 - It is not completely true, as this section states, that the assembly options can be combined. The EIS instruction set option (F77EIS.OBS) is incompatible with the floating-point format conversion option (F77CVF.OBJ). The EIS option is for machines without a floating-point processor, whereas the CVF option requires the use of a floating-point processor.

7. Chapter 12 - An additional chapter, Chapter 12, has been added to clarify I- and D-space concepts and tasks. This information appears in Appendix C.

CHAPTER 2

SYSTEM TAILORING

This chapter describes options you can choose when building PDP-11 FORTRAN-77 into your system. It includes factors affecting compiler performance and information about optional OTS modules that you can use to tailor PDP-11 FORTRAN-77 to your particular applications.

2.1 OPTIONS AFFECTING COMPILE-TIME PERFORMANCE

The following two options affect compile-time performance:

- You can choose one, two, or three temporary disk files for the compiler to use to store information during the compilation process.
- You can change the size of the dynamic storage area in the compiler.

The PDP-11 FORTRAN-77 compiler uses temporary disk files for storing information during the compilation process. The compiler requires at least one temporary file, called the work file.

The work file contains information that the compiler normally accesses at random (for example, the symbol table and the constants table). The dynamic storage area within the compiler is used to manipulate this information. (Only part of the work file is in memory at any given time. Software paging techniques move information back and forth between the dynamic storage area and the work file.)

Information must be moved into the dynamic storage area when needed by the compiler. Therefore, increasing the size of the dynamic storage area increases compilation speed by reducing the number of disk I/O operations (see Section 2.1.2).

2.1.1 Number of Temporary Files

The /WF:w compiler switch specifies the number of temporary disk files that are available to the compiler. If you specify /WF:1, the compiler stores internal representations in just one file, the work file. However, if you specify /WF:2 (or /WF:3), the compiler stores some (or all) of these representations in the one or two other temporary files. The /WF:2 option is the default.

Using additional temporary files slows the compilation process, but it significantly increases the capacity of the compiler. For instance, with three temporary files (/WF:3), the compiler can compile a program that is approximately three times larger than any it can compile with only one temporary file (/WF:1).

No significant change occurs in the compilation rate if you place the temporary files on a fixed-head disk, because these files are written and read sequentially.

2.1.2 Size of the Dynamic Storage Area

Increasing the size of the dynamic storage area increases the rate of compilation. Figure 2-1 illustrates the correlation between compile time and the size of the dynamic storage area. The compile time of four different FORTRAN programs, varying in length from 90 to 450 statements, was measured on a PDP-11/60. The compiler used two temporary files (/WF:2), with the work file residing on the system moving-head disk (RP04). The dynamic storage area varied in size from 4 to 26 pages (when the listing file was suppressed).

The measurements at the end points of each curve denote the approximate compilation rate measured in statements compiled per minute. Continuation and comment lines were not counted.

As Figure 2-1 shows, compilation speed is approximately three times greater when 26 pages of dynamic storage are used than it is when only 4 pages are used. However, using 14 pages results in optimal compiler performance. Building the compiler with more than 14 pages of dynamic storage achieves minimal improvement in the rate of compilation. The default size of the dynamic storage area is 12 pages.

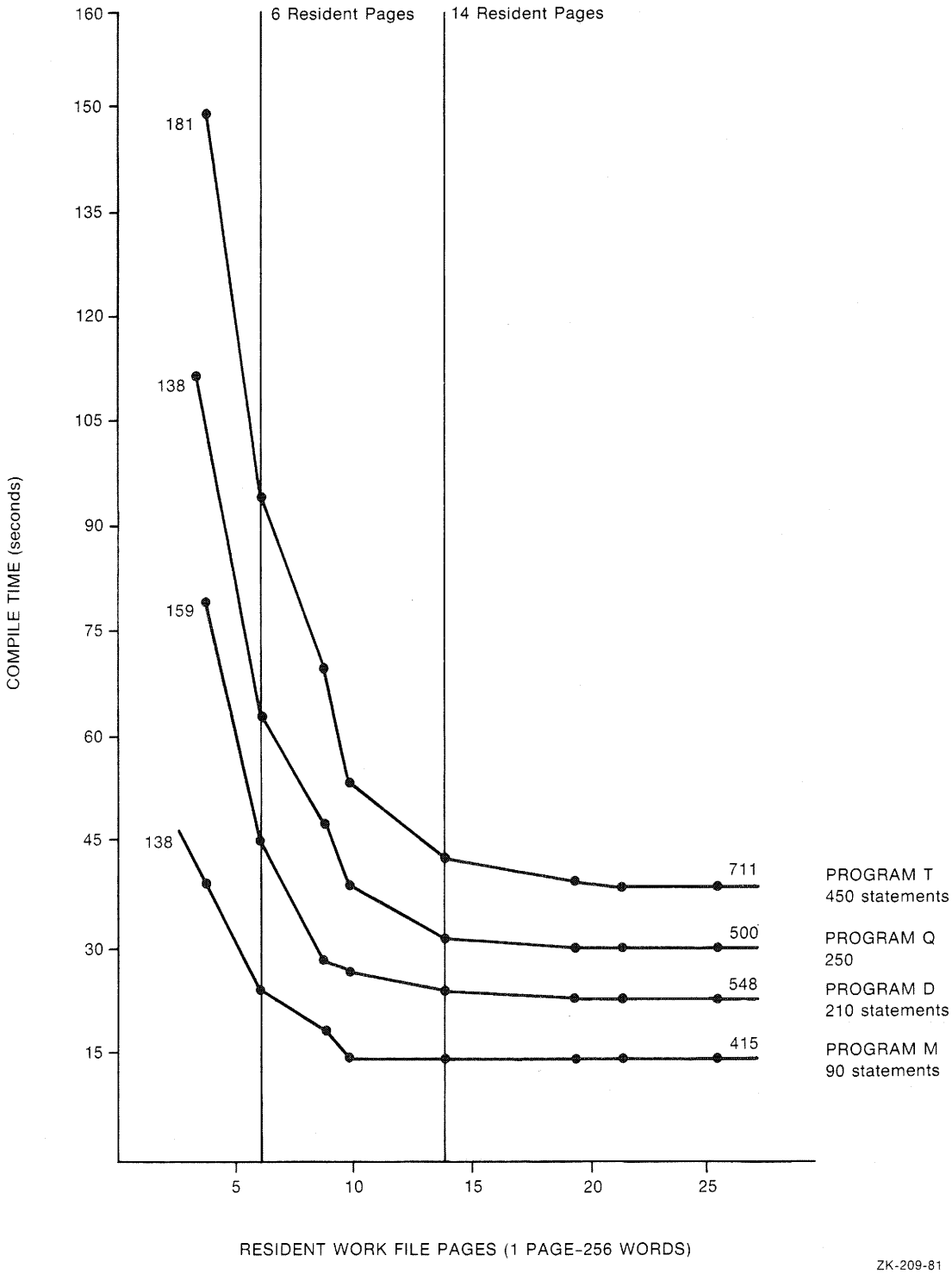

Figure 2-1: Compiler Performance

2.1.2.1 Operating Systems Supporting Dynamic Memory Allocation

Under a RSTS/E, RSX-11M-PLUS, or RSX-11M system with dynamic memory allocation, you specify the size of the PDP-11 FORTRAN-77 compiler's dynamic storage area by using the EXTTSK option in the task-build command file. The value specified by EXTTSK is the size of the dynamic storage area in decimal words. The size of the dynamic storage area is computed as follows:

```
256*(n+w+1)
```

n

The number of pages for the dynamic storage area.

w

The value specified in the /WF:w switch.

You can override the dynamic storage area specified by EXTTSK at installation by means of the INC switch on the INSTALL (INS) command. The task extension size is specified in decimal words.

The following table shows the correlation between the compiler task size, the EXTTSK value, and the number of pages for the dynamic storage area under /WF:2.

Number of Pages	EXTTSK - Value INS/INC Value	Size of Compiler Task (Words)
4	1792	22K
8	2816	23K
12	3840	24K
16	4864	25K

For RSX-11M/M-PLUS installations whose default is the ANSI magnetic tape version of FCS-11 (LB:[1,1]ANSLIB.OLB), the compiler task size increases by approximately 500 words.

2.1.2.2 RSX-11M Without Dynamic Memory Allocation

On an RSX-11M system without dynamic memory allocation, the PDP-11 FORTRAN-77 compiler determines the size of the partition in which it is operating and uses all of the memory in that partition. Install the compiler in a partition large enough for the compiler to run with the desired number of pages of dynamic storage.

2.2 OTS OPTIONS

The distribution kit includes a number of optional OTS modules. After building the OTS library, you can add one or more of these optional modules to the library, or you can maintain these modules separately and refer to them only as needed. The installation procedures copy these modules to LB:[1,1] (or, on RSTS/E systems, to LB:). The PDP-11 FORTRAN-77 system does not require any of the optional modules for normal use.

2.2.1 F7711S

Module F7711S.OBS consists of a set of concatenated object modules that contain alternate versions of FORTRAN sequential I/O support modules. These I/O support modules, designed for use with RSX-11S, provide sequential I/O to non-file-structured devices (for example, terminals, nonspooled card readers, and line printers). These modules do not use the file system but perform direct I/O operations; they reduce task size by approximately 2500 words.

You can use F7711S.OBS in two ways:

- You can include it as an object module at task-build time, as follows:

  ```
  TKB>MAIN/FP=MAIN,LB:[1,1]F7711S.OBS

  (On RSTS/E systems, replace LB:[1,1] with LB:.)
  ```

- Or you can build a separate F77 OTS library for RSX-11S use, LB:[1,1]F7711S.OLB, in addition to the host operating system's OTS library. To do this, when building the OTS, substitute module LB:[1,1]F7711S.OBS (or, for RSTS/E, LB:F7711S.OBS) for the file system module you selected in the installation procedure. For example, replace the reference to FCS11M.OBS with F7711S.OBS.

  Use this OTS library, rather than the host operating system's OTS library, when building tasks for RSX-11S, as follows:

  ```
  TKB>MAIN/FP=MAIN,LB:[1,1]F7711S/LB

  (On RSTS/E systems, replace LB:[1,1] with LB:.)
  ```

2.2.2 Short Error Text - RSX-11M/M-PLUS and RSTS/E Only

For error messages, the PDP-11 FORTRAN-77 OTS references an error-text module containing ASCII text. If your operating system is RSX-11M, RSX-11M-PLUS, or RSTS/E, you can use a long or a short error-text module. The long error-text module requires approximately 1000 words of memory, whereas the alternate version (SHORT.OBJ) requires only one word of memory.

A task with the short error-text module built into it generates complete error reports, but omits the one-line description of the error condition. The PDP-11 FORTRAN-77 User's Guide contains a complete list of OTS error numbers and message text. The F77 OTS uses the long error-text module by default. You can build a task using the short error-text module by loading module $SHORT from the library.

2.2.3 F77MAP

Module F77MAP.OBS consists of a set of concatenated object modules that can be used to transform intrinsic function names into internal names at task-build time. (The PDP-11 FORTRAN-77 compiler transforms intrinsic function names into internal names at compile time.)

Without F77MAP.OBS, if a program written in MACRO-11 attempts to reference a PDP-11 FORTRAN-77 intrinsic function with the FORTRAN name of the function instead of the internal name, an unresolved reference will occur during task build.

For example, F77MAP.OBS maps the FORTRAN name SIN using the following module:

```
          .TITLE    $MSIN
SIN::     JMP       $SIN
          .END
```

F77MAP.OBS contains an object module similar to the preceding module for each of the PDP-11 FORTRAN-77 intrinsic functions.

You can build an F77MAP library as follows:

```
LBR>LB:[1,1]F77MAP.OLB/CR:40.=LB:[1,1]F77MAP.OBS
```

(On RSTS/E systems, replace LB:[1,1] with LB:.)

2.2.4 F77EIS

Module F77EIS.OBS consists of a set of concatenated object modules that contain extended instruction set (EIS) versions of certain integer functions that normally use a floating-point processor. This module allows FORTRAN programs that do not perform floating-point arithmetic to run on a machine that has the extended instruction set but not a floating-point processor. The modules provided in the F77 OTS use a floating-point processor for maximum efficiency in certain INTEGER*4 computations.

Use one of the following commands at task-build time to replace the normal modules in file INTEGER with their EIS versions:

```
TKB>INT/-FP=INT,LB:[1,1]F77EIS.OBS,              <--  for FCS OTS
                LB:[1,1]F77FCS/LB
TKB>INT/-FP=INT,LB:[1,1]F77EIS.OBS,              <--  for RMS OTS
                LB:[1,1]F77RMS/LB,
                LB:[1,1]RMSLIB/LB
```

(On RSTS/E systems, replace LB:[1,1] with LB:.)

You may, instead, substitute the F77EIS module for the default conversion module as follows:

```
LBR>LB:[1,1]F77FCS/RP=LB:[1,1]F77EIS              <--  for FCS OTS
LBR>LB:[1,1]F77RMS/RP=LB:[1,1]F77EIS              <--  for RMS OTS
```

(On RSTS/E systems, replace LB:[1,1] with LB:.)

No changes in the Task Builder command line are necessary if this substitution is performed.

This module cannot be used with optional OTS module F77CVF.

2.2.5 F77CVF

Module F77CVF.OBJ is an alternative module for performing formatted output of floating-point values under control of the D, E, F, and G field specifiers. The standard module provided as part of the F77 OTS uses multiple-precision, fixed-point integer techniques to maintain maximum accuracy during the conversion of data (FPP hardware is not used). The alternative module performs the same functions using the FPP hardware. It is approximately twice as fast as, but in some cases slightly less accurate than, the standard module.

Use one of the following commands at task-build time to replace the normal modules in file OUTR with their F77CVF FPP versions:

```
TKB>OUTR=OUTR,LB:[1,1]F77CVF,LB:[1,1]F77FCS/LB    <--  for FCS OTS
TKB>OUTR=OUTR,LB:[1,1]F77CVF,LB:[1,1]F77RMS/LB,  <--  for RMS OTS
        LB:[1,1]RMSLIB/LB
```

(On RSTS/E systems, replace LB:[1,1] with LB:.)

You may, instead, substitute the F77CVF module for the default conversion module as follows:

```
LBR>LB:[1,1]F77FCS/RP=LB:[1,1]F77CVF              <--  for FCS OTS
LBR>LB:[1,1]F77RMS/RP=LB:[1,1]F77CVF              <--  for RMS OTS
```

(On RSTS/E systems, replace LB:[1,1] with LB:.)

No changes in the Task Builder command line are necessary if this substitution is performed.

This module cannot be used with optional OTS module F77EIS.

2.2.6 F77NER

Module F77NER.OBS consists of a set of concatenated object modules for reporting run-time errors. If you use this module, the error-message text report is suppressed. However, error processing and calls to ERRSET, ERRSNS, and ERRTST continue to operate normally; only the logging of the message on the user's terminal is suppressed. The STOP and PAUSE statement messages are also suppressed. F77NER.OBJ reduces task size by approximately 375 words less than the standard module.

If you use F77NER with optional OTS modules F7711S or F77NIO, a multiply-defined symbol error may result during task-build. Two correct ways to use F77NER with F7711S or F77NIO follow:

- Build F7711S (or F77NIO) and F77NER as separate libraries and use them as follows:

```
TKB>MAIN/FP=MAIN,LB:[1,1]F77NER/LB:$NERRL,
            LB:[1,1]F7711S/LB,LB:[1,1]F77FCS/LB
```

(On RSTS/E systems, replace LB:[1,1] with LB:.)

- Build an OTS by incorporating F7711S.OBS instead of FCS11M.OBS into F77FCS. Name the resulting library F7711S.OLB, and build F77NER as a separate library. Use those libraries as follows:

```
TKB>MAIN/FP=MAIN,LB:[1,1]F77NER/LB:$NERRL,
            LB:[1,1]F7711S/LB
```

(On RSTS/E systems, replace LB:[1,1] with LB:.)

2.2.7 F77NIO

Module F77NIO.OBS consists of a set of concatenated object modules that contain alternative versions of certain OTS routines that are always present in the user task and that provide support for FORTRAN I/O operations. The alternate routines in F77NIO.OBS do not support FORTRAN I/O and reduce task size by approximately 1000 words for programs that do not require FORTRAN I/O (such as process control).

2.2.8 F77RAN

Module F77RAN.OBS consists of a set of concatenated object modules that contain an alternative random-number generator that is compatible with previous releases of PDP-11 FORTRAN. If you require this random-number generator for compatibility purposes, include file LB:[1,1]F77RAN.OBS (or, for RSTS/E systems, LB:F77RAN.OBS) at task-build time.

2.2.9 OTS Overlay Description Files

The two OTS overlay description files are:

FCS11M.ODL - FCS-11 support for RSX-11M/M-PLUS and RSTS/E
RMS11M.ODL - RMS support for RSX-11M/M-PLUS and RSTS/E

Each file is an ODL fragment file that you can use for overlaying the PDP-11 FORTRAN-77 OTS modules. Each file contains documentation that describes OTS options and the procedures for using the file.

NOTE

If you are using the RSTS/E system, files FCS11M.ODL and RMS11M.ODL contain references to LB:[1,1]. Change all occurrences of LB:[1,1] to LB:.

2.2.10 OTS Modules Chart

The following chart lists the optional OTS modules and indicates whether they require a floating point processor (FPP) and whether they can replace the standard OTS modules in your OTS library.

	REQUIRES FPP	CAN REPLACE MODULES IN STANDARD OTS
F7711S.OBS	NO	YES *
F77MAP.OBS	NO **	NO
F77EIS.OBS	NO ***	YES *
F77CVF.OBJ	YES	YES
F77NER.OBJ	NO	YES
F77NIO.OBS	NO	YES *
F77RAN.OBS	NO **	YES
FCS11M.ODL	NO	NO
RMS11M.ODL	NO	NO
SHORT.OBJ	NO	---

* - Use this module instead of FCS11M to build the OTS.

** - This module does not require an FPP, though a program using the results might.

*** - This module is for processors without an FPP and cannot be used with optional module F77CVF.OBJ.

CHAPTER 3

OTS RESIDENT LIBRARIES

This chapter describes how to create OTS resident libraries. First, it discusses factors that influence your choice of library. (Note in particular the new vectored default.) Then, it provides instructions for building the resident library of your choice.

A resident library has the following characteristics:

- It resides in memory and must be installed before a task that references it can be installed or run.
- It can be shared by multiple tasks. However, it occupies virtual address space in each task to which it is linked.

The FORTRAN-77 OTS has the following general limitations:

- It does not contain position-independent code (PIC) and, therefore, cannot be built into a PIC resident library.
- It cannot be built into a supervisor-mode library. (However, it can be linked to the FCSFSL supervisor-mode library; see Section 3.1.1.)

For more information on resident libraries, refer to the Task Builder manual for your operating system.

3.1 TYPES OF RESIDENT LIBRARIES

There are two general types of resident libraries: noncluster and cluster. Within each of these two types, you can employ various schemes of organizing your libraries, and you can choose either File Control Services (FCS) or Record Management Services (RMS). The following sections describe the various combinations possible and some of the considerations involved in choosing a resident library organization.

3.1.1 Noncluster Libraries

In a simple noncluster library, all of the library code takes up virtual address space in the task. That is, the size of the library (and hence the amount of address space it requires) consists of the total amount of space required by OTS routines and the file system (typically FCS) routines. A resident library of this type may be faster than an equivalent cluster library organization, but it also takes up much more space. You must use this organization, however, if you have an RSX system that contains no support for memory-mapping directives or for supervisor-mode libraries. (Section 3.4.2.2 contains a sample command file to build a nonclustered FCS library.)

For RSX-11M-PLUS or Micro/RSX systems that support supervisor-mode libraries, you can link the OTS resident library with the supervisor-mode FCS library (FCSFSL). With this organization, only the OTS library takes up virtual address space in your task. When available, this configuration is recommended because it is slightly faster than the equivalent cluster organization. Note, however, that you cannot link the OTS resident library with the RMS supervisor-mode library. (Section 3.4.2.3 describes how to build a nonclustered library linked to the supervisor-mode FCS library.)

3.1.2 Cluster Libraries

Cluster libraries are sets of two or more resident libraries that share the same portion of virtual memory. Conceptually, cluster libraries are like memory-resident overlays; the two or more cluster libraries form a single memory-resident overlay tree in your task's virtual address space.

When your task is linked with cluster libraries, only one of the two or more libraries is mapped by your task at one time. Therefore, the amount of virtual address space dedicated to libraries equals the largest of the cluster libraries, rather than their total. When a call is made to a routine in a library other than the one currently mapped, the task automatically remaps to the new library. This process incurs some overhead; cluster libraries are slower than their noncluster or supervisor-mode counterparts.

Nevertheless, if your task uses RMS, a cluster library organization is recommended. Using this scheme, you can include all of RMS and a great deal of the FORTRAN-77 OTS in 8K words of your task's virtual address space. (Section 3.4.2.4 shows how to build a resident library clustered with RMSRES.)

If your task uses FCS, you should use a cluster library organization for these same reasons, unless your system supports supervisor-mode libraries. The supervisor-mode library (as discussed in Section 3.1.1) gains for your task the same advantages as those of a cluster library and is faster. (Section 3.4.2.5 shows how to build a resident library clustered with FCSRES.)

3.2 SUPPORT FOR RESIDENT LIBRARIES

The chart below indicates the support available for the types of OTS libraries described in Section 3.1. Unless noted, support is extended regardless of operating system.

	FCS OTS	RMS OTS
NONCLUSTER LIBRARY	SUPPORTED	NOT SUPPORTED
CLUSTER LIBRARY	SUPPORTED	SUPPORTED
LINKED AGAINST SUPERVISOR-MODE LIBRARY	SUPPORTED*	NOT SUPPORTED

* - This combination is not supported on the RSTS/E operating system.

3.3 VECTORED RESIDENT LIBRARIES

No matter which of the above types of resident libraries you favor for your task, it will be a vectored library. A vectored resident library references a fixed entry point to an OTS routine, rather than the address to an OTS routine.

This means that tasks built using a vectored resident library do not need to be rebuilt if routines in the resident library change. When modifications to the OTS resident library routines occur, the OTS routine addresses may shift, but the vectored entry pointers in the resident library do not.

It also means that resident libraries must remain the same size (see Section 3.4.2.1 for information about optimal sizes). A task linked against a resident library of one size fails when run against a resident library of another size.

3.4 CREATING AN OTS RESIDENT LIBRARY

Once you have identified the type of OTS resident library you need, you must choose between a default or a tailored version of this library.

- A default version offers you ease and speed in building, as well as a standard library size (8K-words).
- A tailored version offers you your choice of library modules and some choice in library size (a 4K-word multiple).

Section 3.4.1 provides instructions for building a default library; Section 3.4.2 provides instructions for building a tailored library. Section 3.4.2 also provides Task Builder command file examples for each type of resident library; these examples will build appropriate libraries in most situations.

NOTE

The command files listed in this chapter do not link either of the two OTS error-message modules, $ERTXT and $SHORT, into the resident library. You may include one of these modules when building your library if you wish to force long or short error-message text to be used by programs that link to that library.

3.4.1 The Default Library

Your FORTRAN-77 installation kit contains the command files identified in the following list. You build a default OTS resident library by invoking the command file that corresponds to the type of library you desire.

Command File Name	Type of OTS Resident Library Built
F7FRES.BLD	Noncluster with FCS Routines Included
F7SRES.BLD	Noncluster Linked to FCSFSL
F7RCLS.BLD	Cluster with RMSRES
F7FCLS.BLD	Cluster with FCSRES

These command files perform the following steps:

1. Compile the vector module twice, once to create F77VEC (without OTS routine pointers) and once to create F77REC (with OTS routine pointers).

2. Replace the .OLB vector module in the OTS resident library with F77VEC.

3. Build an 8K-word resident library.

4. Install the library in a partition.

These steps make a default OTS resident library available to you.

3.4.2 The Tailored Library

Your FORTRAN-77 installation kit contains the MACRO-11 files identified in the following list. You can build a tailored OTS resident library by editing the MACRO-11 file that corresponds to the type of library you desire.

MACRO-11 File	Type of OTS Resident Library Built
F7FRES.MAC	Noncluster with FCS Routines Included
F7SRES.MAC	Noncluster Linked to FCSFSL
F7RCLS.MAC	Cluster with RMSRES
F7FCLS.MAC	Cluster with FCSRES

(Section 3.4.2.1 provides more information on editing these files.)

Depending on the degree of tailoring desired, you can then invoke the appropriate command file (see Section 3.4.1), or you can issue commands and create command files of your own, as follows:

1. Compile the MACRO-11 file you just edited. Use .OBJ as the file extension in your output file.

2. Use a text editor to create an appropriate Task Builder command file to build your resident library. (The command files in Sections 3.4.2.2 through 3.4.2.5 should work as shown or with slight modifications.)

   Make sure the number in your EXTSCT=$$OTSI:n statement is correct; it should extend your task's size to 8K.

   Make sure that the OTS object module library you specify contains file system modules (FCS or RMS) that match the file system you intend to use.

3. Invoke the Task Builder and pass to it the command file you just created.

4. Inspect the map file resulting from the task-build. If the resident library is too large or is not large enough, edit (or re-edit) your MACRO-11 file and repeat the steps outlined above.

5. Purge any task, map, or STB files resulting from previous task-builds.

6. Install the library in memory, following the instructions in the documentation for your particular operating system.

These steps make a tailored OTS resident library available to you.

3.4.2.1 Editing the MACRO-11 File

The four MACRO-11 files (F7FRES.MAC, F7SRES.MAC, F7FCCLS.MAC, and F7RCLS.MAC) contain global references to OTS entry points. The modules referenced in the MACRO-11 file you choose will make up your OTS resident library. You can edit these files to include modules that your tasks use frequently, or to exclude modules that are used infrequently. (Editing instructions are included in the file.) You can also use each file as is.

If you edit one of these files, your goal should be to create an OTS resident library that appropriately balances the requirements of size and functionality. If your library is very large, the virtual address space available for your task may be unreasonably small. (A library size of 8K words is recommended.) On the other hand, the Task Builder places in your task the object code for any modules it references that are not in the OTS resident library; thus, it does not make sense to exclude commonly used modules from the library.

To make the best use of available virtual memory, the OTS resident library should be nearly equal to, but slightly below, a multiple of 4K words. Each time the size of the library exceeds a 4K multiple, an additional APR is required; this has the effect of reducing the virtual address space available to the task by 4K words. The following table illustrates this relationship:

Size in Words	Number of APRs	Size in Octal Bytes
4096	1	20000
8192	2	40000
12288	3	60000
16384	4	100000

In the case of an OTS resident library that will be clustered with a file system library, there is an additional consideration. Remember that with clustered libraries, the virtual address space occupied by the libraries is equal to the size of the largest of the libraries. For example, if your OTS resident library occupies 4K words and the file system library occupies 8K words, the libraries occupy a full 8K words of virtual address space. In this situation, there is no advantage to limiting the size of your OTS resident library to 4K words; you might as well use the full 8K words and have a richer library.

The file system cluster libraries occupy space as follows:

File system	Library name	Number of APRs
FCS	FCSRES	1
RMS	RMSRES	2

When you create the command file that will build the OTS resident library, include a PAR option as follows:

```
PAR=pname:base:length
```

The pname is the partition name; it must be the same as the name of the resident library. For RSX-11M systems, this partition must exist in the system; for RSX-11M-PLUS, the INSTALL command places the

library in another partition if the specified partition does not exist. The values you supply for base and length depend on the number of APRs that the resident library occupies, as follows:

Number of APRs	Base	Length
1	160000	20000
2	140000	40000
3	120000	60000
4	100000	100000

When you use these values, the OTS resident library occupies the highest virtual addresses of your task.

3.4.2.2 Building a Noncluster Library with FCS Routines

The command files in this section build 8K-word OTS resident libraries that include FCS modules. You can tailor your library by modifying F7FRES.MAC.

RSX Systems

The following command file builds an 8K-word OTS resident library (F7FRES.TSK) that includes FCS modules referenced by the OTS:

```
F7FRES/-HD/LI/-PI,F7FRES/-SP/MA,F7FRES=F77RES
LB:[1,1]F77FCS/LB
/
STACK=0
UNITS=0
PAR=F7FRES:140000:40000
EXTSCT=$$OTSI:1000
@LB:[1,1]F77GBL.XCL
//
```

Note that the number 1000 in the EXTSCT statement is a variable, not a constant. It represents the difference between actual library size and the recommended 8K library size.

Note also that the FORTRAN-77 OTS library (LB:[1,1]F77FCS.OLB) is referenced in this command file. This library is the FCS version of the FORTRAN-77 OTS.

When you link a task to this library, make sure that you include LB:[1,1]F77FCS/LB:[1,1]F77VEC in the task's root. Then, use the following Task Builder option:

```
LIBR=F7FRES:RO
```

RSTS/E Systems

The following command file builds an 8K-word OTS resident library (F7FRES.TSK) that includes FCS modules referenced by the OTS.

```
F7FRES/-HD/LI/-PI,F7FRES/-SP/MA,F7FRES=F77RES
LB:F77FCS/LB
/
STACK=0
UNITS=0
PAR=F7FRES:140000:40000
EXTSCT=$$OTSI:1000
@LB:F77GBL.XCL
//
```

Note that the number 1000 in the EXTSCT statement is a variable, not a constant. It represents the difference between actual library size and the recommended 8K library size.

Note also that the FORTRAN-77 OTS library (LB:F77FCS.OLB) is referenced in this command file. This library is the FCS version of the FORTRAN-77 OTS.

When you link a task to this library, make sure that you include LB:F77FCS/LB:F77VEC in the task's root. Then, use the following Task Builder option:

```
LIBR=F7FRES:RO
```

3.4.2.3 Building a Noncluster Library Linked to FCSFSL

The following command file builds an OTS resident library (F7SRES.TSK) for which FCS routines reside in a separate supervisor-mode library. As supplied, F7SRES.MAC builds a 8K-word library to link to FCSFSL. You can tailor your library by modifying F7SRES.MAC.

You can use this configuration only on RSX-11M-PLUS systems that support supervisor-mode libraries.

```
F7SRES/-HD/LI/-PI,F7SRES/-SP/MA,F7SRES=F77RES
LB:[1,1]F77FCS/LB
/
STACK=0
UNITS=0
SUPLIB=FCSFSL:SV
PAR=F7SRES:140000:40000
GBLDEF=.FSRCA:0
GBLXCL=.FSRCA
EXTSCT=$$OTSI:1000
@LB:[1,1]F77GBL.XCL
//
```

Note that the number 1000 in the EXTSCT statement is a variable, not a constant. It represents the difference between actual library size and the recommended 8K library size.

Note also that the FORTRAN-77 OTS library (LB:[1,1]F77FCS.OLB) is referenced in this command file. This library is the FCS version of the FORTRAN-77 OTS.

When you link a task to this library, make sure that you include LB:[1,1]F77FCS/LB:F77VEC in the task's root. Then, use the following Task Builder option:

```
LIBR=F7SRES:RO
```

3.4.2.4 Building a Library to Cluster with RMSRES

The command files in this section build 8K-word OTS resident libraries that cluster with RMSRES, the RMS file system resident library. You can tailor your library by modifying F7RCLS.MAC

RSX Systems

The following command file builds an 8K-word OTS resident library (F7RCLS.TSK) that clusters with RMSRES:

```
F7RCLS/-HD/LI/-PI,F7RCLS/-SP/MA,F7RCLS=F77RES
LB:[1,1]F77RMS/LB
LB:[1,1]RMSLIB/LB:R0RMSC
/
STACK=0
UNITS=0
PAR=F7RCLS:140000:40000
GBLXCL=.SAVR1
EXTSCT=$$OTSI:10000
@LB:[1,1]F77GBL.XCL
//
```

Note that the number 1000 in the EXTSCT statement is a variable, not a constant. It represents the difference between actual library size and the recommended 8K library size.

Note also that the FORTRAN-77 OTS library (LB:[1,1]F77RMS.OLB) is referenced in this command file. This library is the RMS version of the FORTRAN-77 OTS.

When you link a task to this library, make sure that you include the following modules in the task's root:

```
LB:[1,1]F77RMS/LB:F77VEC
LB:[1,1]RMSLIB/LB:ROAUTL:ROIMPA:ROEXSY:RMSSYM
```

Then, use the following Task Builder option:

```
CLSTR=F7RCLS,RMSRES:RO
```

RSTS/E Systems

The following command file builds an 8K-word OTS resident library (F7RCLS.TSK) that clusters with RMSRES:

```
F7RCLS/-HD/LI/-PI,F7RCLS/-SP/MA,F7RCLS=F77RES
LB:F77RMS/LB
LB:RMSLIB/LB:R0RMSC
/
STACK=0
UNITS=0
PAR=F7RCLS:140000:40000
GBLXCL=.SAVR1
EXTSCT=$$OTSI:1000
@LB:F77GBL.XCL
//
```

Note that the number 1000 in the EXTSCT statement is a variable, not a constant. It represents the difference between actual library size and the recommended 8K library size.

Note also that the FORTRAN-77 OTS library (LB:F77RMS.OLB) is referenced in this command file. This library is the RMS version of the FORTRAN-77 OTS.

When you link a task to this library, make sure that you include the following modules in the task's root:

```
LB:F77RMS/LB:F77VEC
LB:RMSLIB/LB:ROAUTL:ROIMPA:ROEXSY:RMSSYM
```

Then, use the following Task Builder option:

```
CLSTR=F7RCLS,RMSRES:RO
```

3.4.2.5 Building a Library to Cluster with FCSRES

The following command file builds an 8K-word OTS resident library (F7FCLS.TSK) that clusters with FCSRES, the FCS file system resident library. As is, F7FCLS.MAC builds a 8K-word library to cluster with FCSRES. You can tailor your library by modifying F7FCLS.MAC

This command file is applicable to RSX systems only.

```
F7FCLS/-HD/LI/-PI,F7FCLS/-SP/MA,F7FCLS=F77RES
LB:[1,1]F77FCS/LB
LB:[1,1]SYSLIB/LB:FCSVEC
/
STACK=0
UNITS=0
PAR=F7FCLS:140000:40000
EXTSCT=$$OTSI:1000
@LB:[1,1]F77GBL.XCL
GBLINC=.FCSJT
GBLXCL=.ASLUN
GBLXCL=.CLOSE
GBLXCL=.CSI1
GBLXCL=.CSI2
GBLXCL=.CSI4
GBLXCL=.DELET
GBLXCL=.DLFNB
GBLXCL=.ENTER
GBLXCL=.EXPLG
GBLXCL=.EXTND
GBLXCL=.FCTYP
GBLXCL=.FIND
GBLXCL=.FINIT
GBLXCL=.FLUSH
GBLXCL=.GET
GBLXCL=.GETSQ
GBLXCL=.GTDID
GBLXCL=.GTDIR
GBLXCL=.MARK
GBLXCL=.MRKDL
GBLXCL=.OPEN
GBLXCL=.OPFID
GBLXCL=.OPFNB
GBLXCL=.PARSE
GBLXCL=.POINT
GBLXCL=.POSIT
GBLXCL=.POSRC
GBLXCL=.PRINT
GBLXCL=.PRSDI
GBLXCL=.PRSDV
GBLXCL=.PRSFN
```

```
GBLXCL=.PUT
GBLXCL=.PUTSQ
GBLXCL=.READ
GBLXCL=.REMOV
GBLXCL=.RENAM
GBLXCL=.SAVR1
GBLXCL=.TRNCL
GBLXCL=.WAIT
GBLXCL=.WRITE
//
```

Note that the number 1000 in the EXTSCT statement is a variable, not a constant. It represents the difference between actual library size and the recommended 8K library size.

Note also that the FORTRAN-77 OTS library (LB:[1,1]F77FCS.OLB) is referenced in this command file. This library is the FCS version of the FORTRAN-77 OTS.

When you link a task to this library, include LB:[1,1]F77FCS/LB:F77VEC in the task's root. Then, use the following Task Builder option:

```
CLSTR=F7FCLS,FCSRES:RO
```

APPENDIX A

CHARACTER STRINGS AND ARRAYS: DECLARATION AND REFERENCE

This appendix presents additional information on character strings and arrays. This material should be appended to Section 6.3 of the PDP-11 FORTRAN-77 User's Guide.

Character strings and character arrays are not interchangeable. Character strings comprise one or more characters; character arrays comprise one or more character strings.

Both must be declared and referenced uniquely to avoid compiler and run-time errors. This section describes how to declare and reference character strings and arrays correctly.

A.1 CHARACTER STRING DECLARATION

A character string declarator has the form

```
CHARACTER[*len[,]] v[*len][,v[*len]]...
```

len

The length specification, that is, the number of characters in a character variable. Len must be an unsigned, nonzero, integer constant.

A length len immediately following the word CHARACTER is the length specification for each variable in the character string statement not having its own length specification. A length specification immediately following a variable is the length specification for only that variable. If a length is not specified for a variable, its length is 1.

v

A variable name.

The following examples illustrate the length specification rules when applied to character string declaration.

CHARACTER*10	FNAM	a string FNAM of ten characters
CHARACTER*10	FNAM,MNAM	two strings FNAM and MNAM of ten characters each
CHARACTER	FNAM*10	a string FNAM of ten characters
CHARACTER	FNAM*10,MNAM*10	two strings FNAM and MNAM of ten characters each

```
CHARACTER*2  FNAM*10              a string FNAM of ten characters

CHARACTER*2  FNAM*10,MNAM         a string FNAM of ten characters
                                  and a string MNAM of two characters

CHARACTER    FNAM                 a string FNAM of one character
```

A.2 CHARACTER ARRAY DECLARATION

A character array declarator has the form

```
CHARACTER[*len[,]] v(elm)[*len] [,v(elm)[*len]]
```

len

The length specification, that is, the number of characters in a character array element. Len must be an unsigned, nonzero, integer constant.

A length len immediately following the word CHARACTER is the length specification for each array element in the character array statement not having its own length specification. A length specification immediately following an array element is the length specification for only that array element. If a length is not specified for an array element, its length is 1.

v

An array name.

elm

The array declarator, that is, the number of elements in the array.

The following examples illustrate the length specification rules when applied to character array declaration.

```
CHARACTER*10 FNAM(5)              a five-element array FNAM,
                                  each element ten characters

CHARACTER*10 FNAM(5),MNAM(5)      two five-element arrays FNAM
                                  and MNAM, each element ten
                                  characters

CHARACTER    FNAM(5)*10           a five-element array FNAM,
                                  each element ten characters

CHARACTER    FNAM(5)*10,MNAM(5)*10 two five-element arrays FNAM
                                  and MNAM, each element ten
                                  characters

CHARACTER*2  FNAM(5)*10           a five-element array FNAM,
                                  each element ten characters

CHARACTER*2  FNAM(5)*10,MNAM(5)   a five-element array FNAM,
                                  each element ten characters;
                                  a five-element array MNAM,
                                  each element two characters

CHARACTER    FNAM(5)              a five-element array FNAM,
                                  each element one character
```

A.3 CHARACTER STRING REFERENCE

A character string reference has the form

```
v([e1]:[e2])
```

v

A character variable.

e1

A numeric expression that specifies the leftmost character position of the string. Character positions within a character variable are numbered from left to right, beginning at one. If e1 is omitted, FORTRAN-77 assumes that e1 equals one.

e2

A numeric expression that specifies the rightmost character position of the string. Character positions within a character variable are numbered from left to right, beginning at one. If e2 is omitted, FORTRAN-77 assumes that e2 equals the length specification.

The following examples illustrate the character-range rules when applied to character-string references.

FNAM(7:9)	characters 7 through 9 of string FNAM
FNAM(3:3)	character 3 of string FNAM
FNAM(:6)	characters 1 (default) through 6 of string FNAM
FNAM(2:)	characters 2 through the length specification (default) of string FNAM
FNAM	all characters of string FNAM

A.4 CHARACTER ARRAY REFERENCE

A character array reference has the form

```
a(s[,s]...)([e1]:[e2])
```

a

A character array name.

s

A subscript expression.

e1

A numeric expression that specifies the leftmost character position of the string. Character positions within an array element are numbered from left to right, beginning with one. If e1 is omitted, FORTRAN-77 assumes the e1 equals one.

e2

A numeric expression that specifies the rightmost character position of the string. Character positions within an array element are numbered from left to right, beginning with one. If e2 is omitted, FORTRAN-77 assumes that e2 equals the length specification.

The following examples illustrate the character-range rules when applied to referencing character arrays.

FNAM(1)(1:10)	characters 1 through 10 of the first string of array FNAM
FNAM(2)(3:3)	character 3 of the second string of array FNAM
FNAM(3)(:4)	characters 1 (default) through 4 of the third string of array FNAM
FNAM(4)(6:)	characters 6 through the length specification (default) of the fourth string of array FNAM
FNAM(5)	all characters of the fifth string of array FNAM

A.5 ERROR 21 AND PROGRAM CORRECTIONS

ERROR 21, Missing operator or delimiter symbol, is generated at compile time if an illegal reference is made to a character string. The following program demonstrates this error.

```
PROGRAM CHAR
CHARACTER*5 ASTR,BSTR*2
ASTR(1:1) = '1'
BSTR = ASTR(1)  ! This line generates ERROR 21 at compile time
END
```

The error is generated at the indicated line because ASTR is illegally referenced as an array, not as a string. The compiler expected to see ASTR referenced with the form ASTR(1:1). That is, a ":1" was expected after the "(1" thus the missing operator or delimiter symbol error. The corrected program follows.

```
PROGRAM CHAR
CHARACTER*5 ASTR,BSTR*2
ASTR(1:1) = '1'
BSTR = ASTR(1:1)
END
```

There is an alternate way to correct this program. Instead of changing the reference to ASTR, you might change the declaration of ASTR (making it an array, as the original incorrect program assumed it was). However, the assignment to ASTR must be modified to reference the first (and only) string of the array. The second version of the program follows.

```
PROGRAM CHAR
CHARACTER*5 ASTR(1),BSTR*2
ASTR(1)(1:1) = '1'
BSTR = ASTR(1)  ! Accept defaults for range of elements
END
```

APPENDIX B

VIRTUAL ARRAY PROCESSING

This appendix presents additional information on virtual arrays. Append this information to Section 10.2.3 in the PDP-11 FORTRAN-77 Object Time System Reference Manual.

B.1 VIRTUAL ARRAYS IN SEPARATE I- AND D-SPACE

When programming virtual arrays, you should carefully consider virtual array interaction with other elements. This is particularly important when your programs run in separate I- and D-space.

I- and D-space is an advanced programming technique that allows you to effectively double your virtual task space. Normally, 32K words can be associated with a task. With I- and D-space, this number increases to 64K words. (See Appendix C for more information on I- and D-space.)

I- and D-space increases your virtual task space, but it complicates relationships among virtual arrays and resident libraries, resident commons, and other shared regions. You must pay attention to these relationships, or you might experience failing tasks and unreliable results.

NOTE

> Be especially careful if you use virtual arrays in an I- and D-space task linked to a resident library that contains data. This configuration is not supported on any operating system.

The remainder of this section explains some of the problems that may occur with tasks in I- and D-space.

B.1.1 Importance of the Active Page Register Usage Bit Map

The OTS uses information provided by the Task Builder to initialize virtual arrays. This information is coded in the Active Page Register (APR) usage bit map.

The bit map is a flag word with bits representing APR status. Half the bits represent I-space APRs and half represent D-space APRs. When the Task Builder determines that an I-space APR is required to map instructions to a task, it marks an I-space APR bit; this indicates which I-space APR will be used. When it determines that a D-space APR is required to provide data for a task, it marks a D-space APR bit; this indicates which D-space APR will be used.

The OTS, at run-time, can then determine from the bit map which D-space APRs are free to map virtual arrays.

B.1.2 Conflicts Between the Bit Map and Actual APR Status

The bit map may not reflect actual APR status because the Task Builder makes certain assumptions in encoding the bit map. Some resident and clustered libraries (for example, the CLUSTR, LIBR, and RESLIB options of the Task Builder) contain both information and data. They require I-space APRs and D-space APRs. The Task Builder marks only the I-space APRs in the bit map. This means that the Task Builder encodes the bit map with incomplete information, and the FORTRAN-77 OTS maps virtual arrays using incomplete information.

B.1.3 Operating System Response at Run-Time

RSX systems and RSTS systems respond differently to this situation at run-time.

RSX systems employ the APRs needed as tasks execute, despite the Task Builder assumption that resident and cluster libraries will not require D-space APRs. If RSX determines that a library needs data, it uses a D-space APR to map that data when the task is initiated. When the OTS initializes a virtual array, it may remap the D-space APR used by the library. This remapping will proceed successfully. However, when the library tries to access its data, the task will fail because the data the library needs has been destroyed; in its place is the virtual array.

RSTS systems assign both I- and D-space APRs to resident or cluster libraries, whether the libraries actually need them or not. When the OTS initializes a virtual array, it may attempt to use the same D-space APR used by a resident library. However, RSTS does not permit the FORTRAN-77 OTS to remap an APR used by a resident or clustered library. The virtual array initialization will fail, and you will receive a "maximum memory exceeded" error.

On RSX systems, virtual arrays may interact correctly with resident and cluster libraries that contain only I-space. Check your operating system and layered product manuals to confirm that your resident libraries are I-space only. Make sure, also, that any other shared regions are being used correctly.

APPENDIX C

I- AND D-SPACE TASKS

This appendix presents additional information that will form part of a new PDP-11 FORTRAN-77 Object Time System Reference Manual chapter on I- and D-space.

I- and D-space ("I" represents "instruction" and "D" represents "data") is an advanced programming technique that allows you to effectively double your virtual task space from 32K words to 64K words. More specifically, you can have up to 32K words of instructions and 32K words of data associated with a task. Without I- and D-space, only 32K total words are available to your tasks.

C.1 SUPPORT FOR I- AND D-SPACE TASKS

Before using the I- and D-space technique, you must establish that I- and D-space support is available for your task.

- Make sure your processor and operating system support I- and D-space tasks. (See Section C.1.1.)
- Make sure the D-space parameter within the configuration file equals 1. (See Section C.1.2.)
- Make sure you have included the /ID switch in the task-build command line for your programs. (See Section C.1.3.)
- Make sure, if your task uses a FORTRAN-77 OTS resident library, that this library's relationships to supervisor-mode libraries, resident commons, and virtual arrays respect restrictions. (See Section C.1.4.)

C.1.1 Processor and Operating System Support

I- and D-space is supported by the processors and operating systems listed below:

- Processors:

  - PDP-11/44
  - PDP-11/45
  - PDP-11/50
  - PDP-11/55
  - PDP-11/70
  - PDP-11/73
  - PDP-11/83
  - PDP-11/84
  - J-11

- Operating systems (version noted or higher):

  - RSX-11M-PLUS V2.0
  - Micro/RSX V3.0
  - RSTS/E V9.0
  - Micro/RSTS V2.0

C.1.2 D-Space Parameter Requirement for Support

See Section 1.1 for more information about the D-space parameter and I- and D-space support.

C.1.3 ID Switch Requirement for Support

All I- and D-space tasks require an /ID switch in the Task Builder command line. For example:

```
TKB MYPROG/ID=MYPROG,LB:[1,1]F77FCS/LB
```

(On RSTS/E systems, replace LB:[1,1] with LB:.)

Instead of explicitly supplying the command line to the Task Builder, you can also use an indirect command file. In this case, edit your command file to include the /ID switch on the command line.

C.1.4 OTS Resident Library Support

When a FORTRAN-77 OTS resident library contains data, the following cases can cause invalid results or run-time errors in an I- and D-space environment:

- Using the FORTRAN-77 OTS resident library with resident commons.
- Using the FORTRAN-77 OTS resident library (or any resident library that contains data) with a virtual array. (This configuration is not supported.)

For more information on I- and D-space and how it works, see the RSX-11M/M-Plus Task Builder Manual and the PDP-11 Architecture Handbook.

APPENDIX D

COMPILER TASK-BUILD FILES

This appendix contains compiler task-build files for RSX-11M/M-PLUS, RSTS/E, and VAX-to-RSX operating systems. These files are built as shown when users employ the pre-defined defaults.

D.1 PDP-11 FORTRAN-77 COMPILER TASK-BUILD FILE FOR RSX-11M/M-PLUS (F7711M.CMD)

```
F77/CP/-FP,F77/-SP=F7711M.ODL/MP
;
; PDP-11 FORTRAN-77 COMPILER TASK BUILD FILE
;
; SUMMARY OF SYSTEM PARAMETERS:
;           REFERENCES PARTITION "GEN"
;           24K COMPILER TASK
;           512 WORD STACK
;           32  FRAMES IN EXPRESSION ANALYZER STACK
;           20  FRAMES IN DO/BLOCK IF STATEMENT STACK
;           45  COMMON BLOCKS, MAXIMUM
;           12  RESIDENT PAGES FOR WORKFILE SYSTEM
;
;
; OPTION INPUT
;
TASK        =...F77

; BUILD FOR PARTITION "GEN", MAPPED 11M SYSTEM
; PARTITION MUST BE AT LEAST 22K
;
PAR         =GEN

; SP STACK OF 1024 WORDS
; STACK MUST NEVER BE LESS THAN 512 WORDS
;
STACK       =1024

; F77 COMPILER LOGICAL UNIT ASSIGNMENTS
;            1  COMMAND INPUT
;            2  COMMAND OUTPUT
;            3  .OBJ OUTPUT
;            4  .LST OUTPUT
;            5  .FTN INPUT
;
;            6  COMPILER WORKFILE    (RANDOM ACCESS)
;           CAN BE REASSIGNED TO SWAPPING DISK IF AVAILABLE
;           DISK MUST BE MOUNTED AS WRITABLE FILES-11 VOLUME,
;           BUT THE WORKFILE DOES NOT REQUIRE A UFD ON THE VOLUME.
;
```

```
;            7  COMPILER TEMP FILES  (SEQUENTIAL ACCESS)
;            8  DISK MUST BE MOUNTED AS WRITABLE FILES-11 VOLUME,
;            BUT THE TEMP FILES DO NOT REQUIRE A UFD ON THE VOLUME.
;
;            9  COMPILER MESSAGE TEXT FILE
;
UNITS        =9
ASG          =TI:1,TI:2
ASG          =SY0:6,SY0:7,SY0:8
ASG          =LB0:9

; RESIDENT MEMORY FOR WORKFILE VIRTUAL MEMORY SYSTEM
;
; UNDER RSX-11M/M-PLUS, WORKFILE RESIDENT MEMORY IS DYNAMICALLY DETERMINED.
; IF THE OPERATING SYSTEM SUPPORTS DYNAMIC MEMORY ALLOCATION,
; THE SIZE OF THE COMPILER DYNAMIC STORAGE IS DETERMINED BY "EXTTSK".
; OTHERWISE, THE COMPILER WILL USE ALL MEMORY AVAILABLE IN THE PARTITION.
;
; INCREASING THE NUMBER OF RESIDENT WORKFILE PAGES WILL MAKE THE COMPILER
; RUN FASTER BY REDUCING PAGING I/O, BUT IT DOES NOT AFFECT THE SIZE
; OF THE MAXIMUM SOURCE PROGRAM WHICH CAN BE COMPILED.
;
EXTTSK       =3840

; F77 USES CONTROL SECTION "STACK1" FOR:
;            EXPRESSION ANALYZER STACK DURING PASS 1
;            NAMED COMMON BLOCK DEFINITIONS IN LATER PASSES
;
; AS DEFINED BELOW, STACK1 IS 312(10) WORDS, PROVIDING:
;            312/8    = 39 EXPRESSION ANALYZER STACK FRAMES
;            312/6    = 52 CONTROL SECTIONS
;                          UP TO 7 CONTROL SECTIONS MAY BE USED FOR
;                          COMPILER-GENERATED CODE AND DATA, LEAVING 45 COMMON BLOCKS.
;
EXTSCT       =STACK1:1160

; F77 USES CONTROL SECTION "DOSTK1" FOR:
;            DO STATEMENT NESTING STACK DURING PASS 1
;
; AS DEFINED BELOW, DOSTK1 IS 80(10) WORDS, PROVIDING:
;            80/4     = 20 NESTED DO/BLOCK IF STATEMENTS
;
EXTSCT       =DOSTK1:240

; DEFINE PRINTER WIDTH AND NUMBER OF SOURCE LINES PER LISTING PAGE
; F77 DEFAULT VALUES ARE:
;            55  SOURCE LINES PER PAGE (PLUS 3 LINES OF HEADING)
;            132 COLUMN LINE PRINTER
; NOTE:
;            55(10) = 67(8)
;            80(10) = 120(8) 132(10) = 204(8)
;
GBLPAT       =FORTRN:LPLINE:67
GBLPAT       =FORTRN:LPWDTH:204

; DEFINE DEFAULT OUTPUT FILE SUPERSEDE BEHAVIOR: A VALUE OF 0 (DEFAULT)
; INDICATES THAT THE COMPILER SHOULD NOT SUPERSEDE OUTPUT LISTING AND
; OBJECT FILES; A VALUE OF 1 ALLOWS SUPERSEDING.
;
GBLPAT       =COMAND:SUP00:0

; DEFINE I- AND D-SPACE SUPPORT.  A VALUE OF 1 INDICATES THAT
; OBJECT MODULES OUTPUT BY THE COMPILER CAN BE USED FOR BUILDING I-
; AND D-SPACE TASKS; A VALUE OF 0 DOES NOT ALLOW THIS.
;
GBLPAT  =FORTRN:DSPACE:1
```

```
; DEFINITION OF COMPILER SWITCH OPTION VALUES
;
; A COMPLETE DESCRIPTION OF THE EFFECTS OF THE COMPILER OPTION SWITCHES
; IS CONTAINED IN SECTION 1.2 OF THE PDP-11 FORTRAN-77 USER'S GUIDE.
;
; SWITCH      SWITCH          VALUE TO GBLPAT
; NAME        SETTING
; ------      -------         ---------------
;
; CK          /-CK            0
;             /CK             1       ARRAY SUBSCRIPT BOUNDS CHECKING
;
; CO          /CO:19.         23      NUMBER OF CONTINUATION LINES
;             /CO:N.          N
;
; DE          /-DE            0
;             /DE             1       INCLUDE DEBUG LINES
;
; I4          /-I4            0       DEFAULT INTEGER*2
;             /I4             1       DEFAULT INTEGER*4
;
; LA          /-LA            0       REINITIALIZE SWITCHES
;             /LA             1
;
; LI          /LI:0           0
;             /LI:1           1       SOURCE
;             /LI:2           2       SOURCE, MAP
;             /LI:3           3       SOURCE, MAP, GENERATED CODE
;
; RO          /-RO            0       R/W CODE SECTIONS
;             /RO             1       R/O CODE SECTIONS
;
; SP          /-SP            0       NO SPOOLING
;             /SP             1       SPOOLING
;
; TR          /-TR            0
;             /TR:NONE        0
;             /TR:NAMES       1
;             /TR:BLOCKS      3
;             /TR:ALL         7
;             /TR             7
;
; WF          /WF:2           2       NUMBER OF TEMPORARY FILES
;             /WF:N           1,2,3
;
; WR          /-WR            0       NO OPTIONAL WARNINGS
;             /WR             1
;
; F77         /-F77           0       FORTRAN 66 INTERPRETATION
;             /F77            1       FORTRAN 77 INTERPRETATION
;
; ST          /-ST            0
;             /ST:NONE        0
;             /ST:SOURCE      1
;             /ST:SYNTAX      2
;             /ST             2
;             /ST:ALL         3
;
; DB          /-DB            0       NO DEBUG INFORMATION
;             /DB             1       PRODUCE DEBUG INFORMATION
;
; EX          /-EX            0       72 COLUMN PER SOURCE LINE
;             /EX             1       132 COLUMN PER SOURCE LINE
;
```

```
; OP          /-OP            0       NO CODE OPTIMIZATION
; OP          /OP             3       WITH CODE OPTIMIZATION
;
;
; THE FOLLOWING "GBLPAT" DEFINITIONS EFFECT DEFAULTS OF:
;
; /-CK/CO:19./-DB/-DE/-EX/-I4/-LA/LI:2/OP/-RO/-SP/TR:BLOCKS/WF:2/WR/F77/-ST
;
; DEFAULT VALUES FOR SWITCH "XX" ARE DEFINED
; BY A "GBLPAT" TO GLOBAL VARIABLE "XX000".
;
GBLPAT  =FORTRN:LA000:0
GBLPAT  =COMAND:CK000:0
GBLPAT  =COMAND:CO000:23
GBLPAT  =COMAND:DE000:0
GBLPAT  =COMAND:I4000:0
GBLPAT  =COMAND:LI000:2
GBLPAT  =COMAND:RO000:0
GBLPAT  =COMAND:SP000:0
GBLPAT  =COMAND:TR000:3
GBLPAT  =COMAND:WF000:2
GBLPAT  =COMAND:WR000:1
GBLPAT  =COMAND:F7700:1
GBLPAT  =COMAND:ST000:0
GBLPAT  =COMAND:DB000:0
GBLPAT  =COMAND:EX000:0
GBLPAT  =COMAND:OP000:3
/
```

D.2 PDP-11 FORTRAN-77/-SP COMPILER TASK-BUILD FILE FOR RSTS/E (F77RST.CMD)

```
F77/CP/-FP,F77/-SP=FRTRN7$:F77RST.ODL/MP
;
; PDP-11 FORTRAN-77 COMPILER TASK BUILD FILE
;
; SUMMARY OF SYSTEM PARAMETERS:
;           REFERENCES PARTITION "GEN"
;           24K COMPILER TASK
;           512 WORD STACK
;           39  FRAMES IN EXPRESSION ANALYZER STACK
;           20  FRAMES IN DO/BLOCK IF STATEMENT STACK
;           45  COMMON BLOCKS, MAXIMUM
;           12  RESIDENT PAGES FOR WORKFILE SYSTEM
;
;
; OPTION INPUT
;
TASK       =...F77

; The RSX-11M Emulator must
; BUILD FOR PARTITION "GEN", MAPPED 11M SYSTEM
; PARTITION MUST BE AT LEAST 28K
;
PAR        =GEN

; SP STACK OF 1024 WORDS
; STACK MUST NEVER BE LESS THAN 512 WORDS
;
STACK      =1024
```

```
; F77 COMPILER LOGICAL UNIT ASSIGNMENTS
;          1  COMMAND INPUT
;          2  COMMAND OUTPUT
;          3  .OBJ OUTPUT
;          4  .LST OUTPUT
;          5  .FTN INPUT
;
;          6  COMPILER WORKFILE     (RANDOM ACCESS)
;             CAN BE REASSIGNED TO SWAPPING DISK IF AVAILABLE
;             DISK MUST BE MOUNTED AS WRITABLE  VOLUME,
;             BUT THE WORKFILE DOES NOT REQUIRE A UFD ON THE VOLUME.
;
;          7  COMPILER TEMP FILES  (SEQUENTIAL ACCESS)
;          8  DISK MUST BE MOUNTED AS WRITABLE FILES-11 VOLUME,
;             BUT THE TEMP FILES DO NOT REQUIRE A UFD ON THE VOLUME.
;
;          9  COMPILER MESSAGE TEXT FILE
;
UNITS      =9
ASG        =TI:1,TI:2
ASG        =SY0:6,SY0:7,SY0:8
ASG        =SY:9

; RESIDENT MEMORY FOR WORKFILE VIRTUAL MEMORY SYSTEM
;
;
; INCREASING THE NUMBER OF RESIDENT WORKFILE PAGES WILL MAKE THE COMPILER
; RUN FASTER BY REDUCING PAGING I/O, BUT IT DOES NOT AFFECT THE SIZE
; OF THE MAXIMUM SOURCE PROGRAM THAT CAN BE COMPILED.
;
;
EXTTSK     =3840

; F77 USES CONTROL SECTION "STACK1" FOR:
;          EXPRESSION ANALYZER STACK DURING PASS 1
;          NAMED COMMON BLOCK DEFINITIONS IN LATER PASSES
;
; AS DEFINED BELOW, STACK1 IS 312(10) WORDS, PROVIDING:
;          312/8   = 39 EXPRESSION ANALYZER STACK FRAMES
;          312/6   = 52 CONTROL SECTIONS
;                UP TO 7 CONTROL SECTIONS MAY BE USED FOR
;            COMPILER-GENERATED CODE AND DATA, LEAVING 45 COMMON BLOCKS.
;
EXTSCT     =STACK1:1160

; F77 USES CONTROL SECTION "DOSTK1" FOR:
;          DO STATEMENT NESTING STACK DURING PASS 1
;
; AS DEFINED BELOW, DOSTK1 IS 80(10) WORDS, PROVIDING:
;          80/4    = 20 NESTED DO/BLOCK IF STATEMENTS
;
EXTSCT     =DOSTK1:240

; DEFINE PRINTER WIDTH AND NUMBER OF SOURCE LINES PER LISTING PAGE
; F77 DEFAULT VALUES ARE:
;          55  SOURCE LINES PER PAGE (PLUS 3 LINES OF HEADING)
;          132 COLUMN LINE PRINTER
; NOTE:
;          55(10) = 67(8)
;          80(10) = 120(8) 132(10) = 204(8)
;
GBLPAT     =FORTRN:LPLINE:67
GBLPAT     =FORTRN:LPWDTH:204
```

```
; DEFINE DEFAULT OUTPUT FILE SUPERSEDE BEHAVIOR: A VALUE OF 0 (DEFAULT)
; INDICATES THAT THE COMPILER SHOULD NOT SUPERSEDE OUTPUT LISTING AND
; OBJECT FILES; A VALUE OF 1 ALLOWS SUPERSEDING.
;
;NOTE:
;
;THE RSTS/E OPERATING SYSTEM ALWAYS SUPERSEDES FILES.
;
GBLPAT      =COMAND:SUP00:0

; DEFINE I- AND D-SPACE SUPPORT.  A VALUE OF 1 INDICATES THAT
; OBJECT MODULES OUTPUT BY THE COMPILER CAN BE USED FOR BUILDING I-
; AND D-SPACE TASKS; A VALUE OF 0 DOES NOT ALLOW THIS.
;
GBLPAT  =FORTRN:DSPACE:1

; DEFINITION OF COMPILER SWITCH OPTION VALUES
;
; A COMPLETE DESCRIPTION OF THE EFFECTS OF THE COMPILER OPTION SWITCHES
; IS CONTAINED IN SECTION 1.2 OF THE PDP-11 FORTRAN-77 USER'S GUIDE.
;
; SWITCH    SWITCH          VALUE TO GBLPAT
; NAME      SETTING
; ------    -------         ---------------
;
; CK        /-CK            0
;           /CK             1       ARRAY SUBSCRIPT BOUNDS CHECKING
;
; CO        /CO:19.         23      NUMBER OF CONTINUATION LINES
;           /CO:N.          N
;
; DE        /-DE            0
;           /DE             1       INCLUDE DEBUG LINES
;
; I4        /-I4            0       DEFAULT INTEGER*2
;           /I4             1       DEFAULT INTEGER*4
;
; LA        /-LA            0       REINITIALIZE SWITCHES
;           /LA             1
;
; LI        /LI:0           0
;           /LI:1           1       SOURCE
;           /LI:2           2       SOURCE, MAP
;           /LI:3           3       SOURCE, MAP, GENERATED CODE
;
; RO        /-RO            0       R/W CODE SECTIONS
;           /RO             1       R/O CODE SECTIONS
;
; SP        /-SP            0       NO SPOOLING
;           /SP             1       SPOOLING
;
; TR        /-TR            0
;           /TR:NONE        0
;           /TR:NAMES       1
;           /TR:BLOCKS      3
;           /TR:ALL         7
;           /TR             7
;
; WF        /WF:2           2       NUMBER OF TEMPORARY FILES
;           /WF:N           1,2,3
;
; WR        /-WR            0       NO OPTIONAL WARNINGS
;           /WR             1
;
```

```
; F77       /-F77           0       FORTRAN 66 INTERPRETATION
;           /F77            1       FORTRAN 77 INTERPRETATION
;
; ST        /-ST            0
;           /ST:NONE        0
;           /ST:SOURCE      1
;           /ST:SYNTAX      2
;           /ST             2
;           /ST:ALL         3
;
; DB        /-DB            0       NO DEBUG INFORMATION
;           /DB             1       PRODUCE DEBUG INFORMATION
;
; EX        /-EX            0       72 COLUMN PER SOURCE LINE
;           /EX             1       132 COLUMN PER SOURCE LINE
;
; OP        /-OP            0       NO CODE OPTIMIZATION
;           /OP             3       WITH CODE OPTIMIZATION
;
;
; THE FOLLOWING "GBLPAT" DEFINITIONS EFFECT DEFAULTS OF:
;
; /-CK/CO:19./-DB/-DE/-EX/-I4/-LA/LI:2/OP/-RO/-SP/TR:BLOCKS/WF:2/WR/F77/-ST
;
; DEFAULT VALUES FOR SWITCH "XX" ARE DEFINED
; BY A "GBLPAT" TO GLOBAL VARIABLE "XX000".

GBLPAT  =FORTRN:LA000:0
GBLPAT  =COMAND:CK000:0
GBLPAT  =COMAND:CO000:23
GBLPAT  =COMAND:DE000:0
GBLPAT  =COMAND:I4000:0
GBLPAT  =COMAND:LI000:2
GBLPAT  =COMAND:RO000:0
GBLPAT  =COMAND:SP000:0
GBLPAT  =COMAND:TR000:3
GBLPAT  =COMAND:WF000:2
GBLPAT  =COMAND:WR000:1
GBLPAT  =COMAND:F7700:1
GBLPAT  =COMAND:ST000:0
GBLPAT  =COMAND:DB000:0
GBLPAT  =COMAND:EX000:0
GBLPAT  =COMAND:OP000:3
//
```

D.3 PDP-11 FORTRAN-77 COMPILER TASK-BUILD FILE FOR VAX-TO-RSX (F77VAX.CMD)

```
F77/CP/-FP=F77VAX.ODL/MP
;
; PDP-11 FORTRAN-77 COMPILER TASK BUILD FILE
;
; SUMMARY OF SYSTEM PARAMETERS:
;       REFERENCES PARTITION "GEN"
;       22K COMPILER TASK
;       512 WORD STACK
;       39  FRAMES IN EXPRESSION ANALYZER STACK
;       20  FRAMES IN DO/BLOCK IF STATEMENT STACK
;       45  COMMON BLOCKS, MAXIMUM
;       6   RESIDENT PAGES FOR WORKFILE SYSTEM
;
;
; OPTION INPUT
;
TASK        =...F77
```

```
; BUILD FOR PARTITION "GEN", MAPPED 11M SYSTEM
; PARTITION MUST BE AT LEAST 20K
;
PAR         =GEN

; SP STACK OF 1024 WORDS
; STACK MUST NEVER BE LESS THAN 512 WORDS
;
STACK       =1024

; F77 COMPILER LOGICAL UNIT ASSIGNMENTS
;           1  COMMAND INPUT
;           2  COMMAND OUTPUT
;           3  .OBJ OUTPUT
;           4  .LST OUTPUT
;           5  .FTN INPUT
;
;           6  COMPILER WORKFILE     (RANDOM ACCESS)
;              CAN BE REASSIGNED TO SWAPPING DISK IF AVAILABLE
;              DISK MUST BE MOUNTED AS WRITABLE FILES-11 VOLUME,
;              BUT THE WORKFILE DOES NOT REQUIRE A UFD ON THE VOLUME.
;
;           7  COMPILER TEMP FILES  (SEQUENTIAL ACCESS)
;           8  DISK MUST BE MOUNTED AS WRITABLE FILES-11 VOLUME,
;              BUT THE TEMP FILES DO NOT REQUIRE A UFD ON THE VOLUME.
;
;           9  COMPILER MESSAGE TEXT FILE
;
UNITS       =9
ASG         =TI:1,TI:2
ASG         =WK0:6,WK0:7,WK0:8
ASG         =LB0:9
GBLPAT      =COMAND:MSG6WD:0

; RESIDENT MEMORY FOR WORKFILE VIRTUAL MEMORY SYSTEM
;
; UNDER RSX-11M, WORKFILE RESIDENT MEMORY IS DYNAMICALLY DETERMINED.
; IF THE OPERATING SYSTEM SUPPORTS DYNAMIC MEMORY ALLOCATION,
; THE SIZE OF THE COMPILER DYNAMIC STORAGE IS DETERMINED BY "EXTTSK".
; OTHERWISE, THE COMPILER WILL USE ALL MEMORY AVAILABLE IN THE PARTITION.
;
; INCREASING THE NUMBER OF RESIDENT WORKFILE PAGES WILL MAKE THE COMPILER
; RUN FASTER BY REDUCING PAGING I/O, BUT IT DOES NOT AFFECT THE SIZE
; OF THE MAXIMUM SOURCE PROGRAM THAT CAN BE COMPILED.
;
EXTTSK      =2048

; F77 USES CONTROL SECTION "STACK1" FOR:
;           EXPRESSION ANALYZER STACK DURING PASS 1
;           NAMED COMMON BLOCK DEFINITIONS IN LATER PASSES
;
; AS DEFINED BELOW, STACK1 IS 312(10) WORDS, PROVIDING:
;           312/8    = 39 EXPRESSION ANALYZER STACK FRAMES
;           312/6    = 52 CONTROL SECTIONS
;                  UP TO 7 CONTROL SECTIONS MAY BE USED FOR
;                  COMPILER-GENERATED CODE AND DATA, LEAVING 45 COMMON BLOCKS.
;
EXTSCT      =STACK1:1160

; F77 USES CONTROL SECTION "DOSTK1" FOR:
;           DO STATEMENT NESTING STACK DURING PASS 1
;
; AS DEFINED BELOW, DOSTK1 IS 80(10) WORDS, PROVIDING:
;           80/4     = 20 NESTED DO/BLOCK IF STATEMENTS
;
EXTSCT      =DOSTK1:240
```

```
; DEFINE PRINTER WIDTH AND NUMBER OF SOURCE LINES PER LISTING PAGE
; F77 DEFAULT VALUES ARE:
;          55  SOURCE LINES PER PAGE (PLUS 3 LINES OF HEADING)
;          132 COLUMN LINE PRINTER
; NOTE:
;          55(10) = 67(8)
;          80(10) = 120(8) 132(10) = 204(8)
;
GBLPAT     =FORTRN:LPLINE:67
GBLPAT     =FORTRN:LPWDTH:204

; DEFINE DEFAULT OUTPUT FILE SUPERSEDE BEHAVIOR: A VALUE OF 0 (DEFAULT)
; INDICATES THAT THE COMPILER SHOULD NOT SUPERSEDE OUTPUT LISTING AND
; OBJECT FILES; A VALUE OF 1 ALLOWS SUPERSEDING.
;
GBLPAT     =COMAND:SUP00:0

; DEFINE I- AND D-SPACE SUPPORT: A VALUE OF 1 INDICATES THAT
; OBJECT MODULES OUTPUT BY THE COMPILER CAN BE USED FOR BUILDING I-
; AND D-SPACE TASKS; A VALUE OF 0 DOES NOT ALLOW THIS.
;
GBLPAT  =FORTRAN:DSPACE:1

; DEFINITION OF COMPILER SWITCH OPTION VALUES
;
; A COMPLETE DESCRIPTION OF THE EFFECTS OF THE COMPILER OPTION SWITCHES
; IS CONTAINED IN SECTION 1.2 OF THE PDP-11 FORTRAN-77 USER'S GUIDE.
;
; SWITCH    SWITCH          VALUE TO GBLPAT
; NAME      SETTING
; ------    -------         ---------------
;
; CK        /-CK            0
;           /CK             1       ARRAY SUBSCRIPT BOUNDS CHECKING
;
; CO        /CO:19.         23      NUMBER OF CONTINUATION LINES
;           /CO:N.          N       (OCTAL VALUE)
;
; DE        /-DE            0
;           /DE             1       INCLUDE DEBUG LINES
;
; I4        /-I4            0       DEFAULT INTEGER*2
;           /I4             1       DEFAULT INTEGER*4
;
; LA        /-LA            0       REINITIALIZE SWITCHES
;           /LA             1
;
; LI        /LI:0           0
;           /LI:1           1       SOURCE
;           /LI:2           2       SOURCE, MAP
;           /LI:3           3       SOURCE, MAP, GENERATED CODE
;
; RO        /-RO            0       R/W CODE SECTIONS
;           /RO             1       R/O CODE SECTIONS
;
; SP        /-SP            0       NO SPOOLING
;           /SP             1       SPOOLING
;
; TR        /-TR            0
;           /TR:NONE        0
;           /TR:NAMES       1
;           /TR:BLOCKS      3
;           /TR:ALL         7
;           /TR             7
;
```

```
; WF        /WF:2           2         NUMBER OF TEMPORARY FILES
;           /WF:N           1,2,3
;
; WR        /-WR            0         NO OPTIONAL WARNINGS
;           /WR             1
;
; F77       /-F77           0         FORTRAN 66 INTERPRETATION
;           /F77            1         FORTRAN 77 INTERPRETATION
;
; ST        /-ST            0
;           /ST:NONE        0
;           /ST:SOURCE      1
;           /ST:SYNTAX      2
;           /ST             2
;           /ST:ALL         3
;
; DB        /-DB            0         NO DEBUG INFORMATION
;           /DB             1         PRODUCE DEBUG INFORMATION
;
; EX        /-EX            0         72 COLUMN PER SOURCE LINE
;           /EX             1         132 COLUMN PER SOURCE LINE
;
; OP        /-OP            0         NO CODE OPTIMIZATION
;           /OP             3         WITH CODE OPTIMIZATION
;
;
; THE FOLLOWING "GBLPAT" DEFINITIONS EFFECT DEFAULTS OF:
;
; /-CK/CO:19./-DB/-DE/-EX/-I4/-LA/LI:2/OP/-RO/-SP/TR:BLOCKS/WF:2/WR/F77/-ST
;
; DEFAULT VALUES FOR SWITCH "XX" ARE DEFINED
; BY A "GBLPAT" TO GLOBAL VARIABLE "XX000".
GBLPAT  =FORTRN:LA000:0
GBLPAT  =COMAND:CK000:0
GBLPAT  =COMAND:CO000:23
GBLPAT  =COMAND:DE000:0
GBLPAT  =COMAND:I4000:0
GBLPAT  =COMAND:LI000:2
GBLPAT  =COMAND:RO000:0
GBLPAT  =COMAND:SP000:0
GBLPAT  =COMAND:TR000:3
GBLPAT  =COMAND:WF000:2
GBLPAT  =COMAND:WR000:1
GBLPAT  =COMAND:F7700:1
GBLPAT  =COMAND:ST000:0
GBLPAT  =COMAND:DB000:0
GBLPAT  =COMAND:EX000:0
GBLPAT  =COMAND:OP000:3
/
```