Update Notice #2

June 1982

RSTS/E Programmer's Utilities Manual

AD–D749A–T2

NEW AND CHANGED INFORMATION

This update contains changes and additions made to the *RSTS/E Programmer's Utilities Manual* for RSTS/E V7.2. New information in this update includes changes to the MACRO chapter with new information on DCL.

INSTRUCTIONS

The enclosed pages are to be placed in the *RSTS/E Programmer's Utilities Manual* as replacements for, or additions to, current pages. The changes made on replacement pages are indicated in the outside margin by change bars (▌). Because all of Chapter 2 is being replaced, a change bar appears only in the Table of Contents.

Old Page	New Page
Title page/Copyright	Title page/Copyright
iii to vii	iii to viii
Chapter 2	Chapter 2
(2–1 to 2–9)	(2–1 to 2–14)
Index	Index
(Index–1 to Index–4)	(Index–1 to Index–4)
Ordering Instructions	Ordering Instructions
Reader's Comments/Mailer	Reader's Comments/Mailer

RSTS/E
Programmer's Utilities Manual

Order No. AA–D749A–TC

Including AD–D749A–T1, T2

June 1982

This document describes the RSX–based utilities available to the RSTS/E programmer. It contains information on the MACRO Assembler, Librarian, Patch, and MAKSIL utilities.

OPERATING SYSTEM AND VERSION:	RSTS/E	V7.2
SOFTWARE VERSION:	RSTS/E	V7.2

digital equipment corporation, maynard, massachusetts

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 your critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

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

digital

Commercial Engineering Publications typeset this manual using DIGITAL's TMS–11 Text Management System.

Contents

Appendix A MACRO–11 Diagnostic Error Message Summary

Appendix B Librarian Utility Program (LBR) Files and Formats

Index

Figures

Tables

Preface

This manual describes four RSX–based utility programs available on RSTS/E: MAC, LBR, PAT, and MAKSIL.

- MAC – The MACRO–11 Assembler processes assembly language programs and produces single relocatable binary object files.
- LBR – The Librarian Utility is a library maintenance program that provides a means for creating, modifying, updating, listing, extracting, and maintaining library files.
- PAT – The Object Module Patch Utility is used to modify code in a relocatable binary object module.
- MAKSIL – The Make a Save Image Library Utility is used to create a resident library, a run-time system, or a multi-user task.

Audience Description

To use this manual, you should be familiar with the MACRO computer language and have a working knowledge of the RSTS/E operating system.

Document Structure

This manual has five chapters and two appendixes:

- Chapter 1 Introduction

  Introduces you to the RSX–11M utilities as they are used on a RSTS/E system.

- Chapter 2 Using the MACRO–11 Utility Program

  Describes how to run the MACRO assembler to convert MACRO source code into object code.

- Chapter 3 Using the Librarian Utility Program (LBR)

  Shows how to create, modify, maintain, and use library files containing MACRO modules.

- Chapter 4 Using the Object Module Patch Utility Program (PAT)

  Shows how to modify code in a relocatable binary object module.

- Chapter 5 Using the MAKSIL Utility Program

  Describes how to create a resident library, a run-time system, or a multi-user task.

- Appendix A MACRO–11 Diagnostic Error Message Summary

  Describes the single character codes that identify MACRO programming errors. (Appendix A does not, however, contain a description of the MACRO input-output error messages. You must refer to Chapter 2 for that information.)

- Appendix B Librarian Utility Program (LBR) Files and Formats

  Contains detailed information on the formats and contents of library files.

Associated Documents

The *PDP–11 MACRO–11 Language Reference Manual* describes how to use the MACRO–11 relocatable assembler to develop PDP–11 assembly language programs.

The *RSTS/E DCL User's Guide* describes the use of DCL (DIGITAL Command Language) on RSTS/E.

See the *RSTS/E Documentation Directory* for more information on RSTS/E manuals.

Documentation Conventions

This manual uses the following conventions:

RET	This symbol represents a carriage return. Unless the manual indicates otherwise, end all commands or command strings with a carriage return.
@	This symbol invokes an indirect command file. The at sign immediately precedes the file specification for an indirect command file.
(/)	Slashes in the file specification precede switches or qualifiers.
(.)	Periods in the file specification separate the file name and extension. When the file name is used without an extension, the period is not necessary.
UPPERCASE	In discussions of syntax, uppercase letters represent the command name, which you must type.
lowercase	Lowercase letters represent a variable, for which you must supply a value.
[ ]	Square brackets enclose an item that is optional. You may include the item in brackets, or you may omit it, as you choose.
(...)	The ellipsis symbol represents repetition. You can repeat the item that precedes the ellipsis.

Chapter 2
Using the MACRO–11 Utility Program

RSTS/E has two MACRO language assemblers, one that runs under the RT11 run-time system and one that runs under the RSX–11M run-time system. This chapter describes how to use the RSX–based MACRO assembler. Refer to the *PDP–11 MACRO–11 Language Reference Manual* for complete information about MACRO–11 assembly language and the *RSTS/E RT11 Utilities Manual* for information on the RT11–based MACRO assembler.

The RSX–based MACRO assembler (MAC) converts MACRO source code into object code. Output from the assembler can include any or all of the following:

- A binary relocatable object file that contains all the records and relocation information needed for task building
- A listing of the source input file that provides both documentation for the module and a tool for debugging the code
- A table of contents listing that contains the line sequence numbers, the page numbers, and the text accompanying each .SBTTL directive
- A symbol table listing that provides information about symbol names and program sections (.PSECTs) that are referenced in the source program

To use the MACRO assembler, you should understand how to:

- Run the assembler (including how to format command strings to specify files MACRO uses during assembly)
- Use file specification switches to override file control directives in the source program
- Interpret error messages

The following sections describe these topics.

2.1 Running the MACRO Assembler

There are two ways to run the MACRO assembler on RSTS/E:

- With the DIGITAL Command Language (DCL) MACRO command
- With the RUN command or a Concise Command Language (CCL) command

You can also use indirect command files to enter command lines.

Use the method that best suits your needs and the conditions on your system (default keyboard monitor, for example).

2.1.1 Running MACRO in DCL

You can recognize DCL by its dollar prompt ($). To run MACRO, type:

```
$ MACRO RET
```

After you press RETURN, DCL prompts for the input files:

```
Files:
```

Enter the input files in the form:

input filespec + ... + n

For example:

```
Files:     FILE1+FILE2+FILE3 RET
```

In this example FILE1, FILE2, and FILE3 are the source files to be assembled.

You can also type a complete command on one line. For example:

```
$ MACRO MAIN.MAC RET
```

MACRO creates the object file MAIN.OBJ and the list file MAIN.LST from the source file MAIN.MAC and places the files in your account on the public structure.

Note that you cannot use the wildcard characters (?) and (*) with the DCL MACRO command.

The DCL processor expands the file specifications you enter into more complete file specifications, including device and account. The expanded command line can contain no more than 80 characters. If DCL detects more than 80 characters in the expanded command line, you receive the error message "?Command too long." Note that you can use a plus sign (+) or a comma (,) to separate the input files.

By default, MACRO creates an object file and gives it the same name as the first input file. The assembler also creates a list file by default. The default file extensions are .OBJ for the object file and .LST for the list file.

Although an input file can be on a disk other than the system disk, MACRO places the object and list files in your account on the public structure by default. You can include a device specification in the command line directing MACRO to place the files on a specified device.

If you do not want MACRO to create an object file, use the /NOOBJ qualifier. Similarly, the /NOLIST qualifier tells MACRO not to create a list file. You use these qualifiers when you want to see if a MACRO program assembles, and you do not want to get an object or a list file (at least the first time you run the program).

To give the object or list file a name other than the default, use the /OBJECT or /LIST qualifier to assign a specific file name. If you do not include file extensions, the assembler assigns them for you. For example:

```
$ MACRO MAIN.MAC/OBJECT=MODULE/LIST=MODLST RET
```

In this example:

MACRO	Invokes the assembler.
MAIN.MAC	Is the source input file.
/OBJECT=MODULE	Specifies MODULE as the object file name.
/LIST=MODLST	Specifies MODLST as the list file name.

You can also specify a library file. To do so, add the /LIBRARY qualifier to the input file specification. For example:

```
$ MACRO MAIN.MAC+SECOND.LIB/LIBRARY/OBJECT=MODULE RET
```

In this example:

MACRO	Invokes the assembler.
MAIN.MAC	Is a source input file.
SECOND.LIB	Is a second source input file.
/LIBRARY	Is the library qualifier that marks the input file SECOND.LIB as a library file created by the LBR program.
/OBJECT=MODULE	Specifies MODULE as the object file name.

If you are at the Files: prompt and want to exit the MACRO assembler, press CTRL/Z to return to the dollar prompt. If you have begun an assembly, you can stop the process and return to the dollar prompt by pressing CTRL/C. Note that CTRL/C is an abnormal exit that you should use only as a last resort.

Refer to the *RSTS/E DCL User's Guide* for complete information about DCL.

2.1.2 Running MACRO with the RUN Command or a CCL Command

On most RSTS/E systems, the MACRO assembler runs from the system library account [1,2]. You can use the RUN command to run MACRO from any keyboard monitor by typing:

```
RUN $MAC (RET)
```

The $ is a special account character representing [1,2].

Your system manager may have installed a Concise Command Language (CCL) command to run MACRO. If so, you can type:

```
MAC (RET)
```

In response, MACRO prompts with:

```
MAC>
```

Note that in DCL, typing MAC causes the Files: prompt to appear on your terminal. To use the CCL MAC in DCL, type:

```
$ CCL MAC (RET)
```

The MAC> prompt indicates that the assembler is ready for a command of the form:

output-filespec(s) = input filespec(s)

The file specifications define the object files, listing files and source files appearing in a MACRO command line.

Type output and input file specifications in the form:

object,listing/s:arg = src1,src2....srcn/s:arg

where:

object	Is the file specification of the binary object file produced by the assembly process.
listing	Is the file specification of the assembly and symbol listing produced by the assembly process.
/s:arg	Is one or more file specification switches and arguments. (Section 2.3 describes these switches and arguments.)
src1,src2....srcn	Are file specifications for one or more MACRO source files or MACRO library files.

You cannot use wildcard characters (?) and (*) in MACRO file specifications.

If you omit the object or listing output file specification, the output file is not created. For example:

```
MAC>,LIST=SRC1,SRC2 (RET)
```

In this case an object file is not created, but the list file is. Note that you must include the comma before "LIST".

If you do not enter an input file, you receive the error message "MAC – ILLEGAL FILENAME."

If you do not specify a listing file, MACRO displays on your terminal any errors found in the source program.

You can enter a complete command on one line when you use the CCL command, but you are limited to 80 characters. However, when you are at the MAC> prompt, you can have a total of 132 characters in a command line.

You can use continuation lines at the MAC> prompt, as long as the total length does not exceed 132 characters. A continuation line allows you to type a command line on more than one physical line. Use a hyphen to indicate continuation after any element in the command line. For example:

```
MAC>OBJECT,LISTING=SRC1,- (RET)
MAC>SRC2,SRC3,- (RET)
MAC>SRC4 (RET)
```

If you are at the MAC> prompt and want to exit the MACRO assembler, press CTRL/Z to return to the keyboard monitor prompt. If you have entered a command string and begun an assembly, you can stop the process and return to the keyboard monitor prompt by pressing CTRL/C.

2.1.3 Using Indirect Command Files

Besides typing command lines using the methods described in Sections 2.1.1 and 2.1.2, you can also enter command lines by using an indirect command file. When you specify an indirect command file, MACRO interprets the contents of the file as one or more command lines. The advantage of an indirect command file is that you can enter a commonly used command sequence once and store it for later use.

You can use EDT or any other editor to create the indirect command file. For example, the contents of an indirect command file might be:

```
MAIN,MAIN=MAIN
MACRO,MACRO=MACRO
```

Here is an example that shows the use of an indirect command file:

```
RUN $MAC (RET)
MAC>@filespec (RET)
MAC>
```

The file specified as "@filespec" contains MACRO command strings. After opening this file, MACRO reads and executes command lines until it detects the end of file. MACRO then displays the MAC> prompt. Press CTRL/Z to exit.

You can also use indirect command files in a CCL command:

```
MAC @filespec (RET)
```

MACRO reads and executes command lines until it detects the end of file. Control then returns to the keyboard monitor.

Note that you cannot use indirect command files with the DCL MACRO command.

You can nest indirect command files in MACRO (that is, call one indirect command file from another). However, you cannot nest command files to more than three levels. If you do, you receive the error message "MAC — INDIRECT FILE DEPTH EXCEEDED."

2.2 File Specification Switches

At assembly time, you can override specific MACRO directives that appear in the source program. You can also tell MACRO how to handle specific files during assembly. You do this by including special switches in the MACRO command string. Table 2–1 lists the switches and describes their effects.

Table 2–1: MACRO–11 Switches

Switch	Type	Function
/LI:arg	Listing Control	Overrides source program directive .NLIST.
/NL:arg	Listing Control	Overrides source program directive .LIST.
/SP	Listing Control	Spools listing output.
/NOSP	Listing Control	Does not spool output (default value).
/EN:arg	Function Control	Allows the use of arguments (see Table 2–3) to override source program directive .DSABL.
/DS:arg	Function Control	Allows the use of arguments (see Table 2–3) to override source program directive .ENABL.
/ML	MACRO Library	Indicates that the input file is a MACRO library file.
/PA:1	Assembly Pass	MACRO is a 2–pass assembler. /PA:1 causes the associated file to assemble during pass 1 only.
/PA:2	Assembly Pass	Causes the associated file to assemble during pass 2 only.

Attach the /ML, /PA:1 and /PA:2 switches directly to the source files they affect. Place other switches anywhere in the command string. For example, the /LI switch affects the listing file regardless of where you place it in the command string.

The next four sections describe how to use the file specification switches.

2.2.1 Listing Control Switches

Use the /LI:arg and /NL:arg switches to:

- Control the content and format of assembly listings at assembly time
- Override the arguments of .LIST and .NLIST directives in the source program

MACRO has default settings for the /LI and /NL switches when you do not include arguments:

- The /LI switch without an argument causes MACRO to ignore .LIST and .NLIST directives that have no arguments
- The /NL switch without an argument causes MACRO to list only the symbol table, the table of contents, and error messages

Table 2–2 lists the arguments you can use with /LI and /NL switches. See the *PDP–11 MACRO–11 Language Reference Manual* for more information on the meaning of these arguments.

Table 2–2: Arguments for /LI and /NL Switches

Argument	Listing Control	Default
BEX	Binary extensions.	List
BIN	Generated binary code.	List
CND	Unsatisfied conditionals, .IF and .ENDC statements.	List
COM	Comments.	List
LD	List control directives with no arguments.	No List
LOC	Address location counter.	List
MC	Macro calls, repeat range expansion.	List
MD	Macro definitions, repeat range expansion.	List
ME	Macro expansion.	No List
MEB	Macro expansion binary code.	No List
SEQ	Source line sequence number.	List
SRC	Source code.	List
SYM	Symbol table.	List
TOC	Table of contents.	List
TTM	132–column line printer format when not specified.	No List

For example, assume that the source program contains the following sequence:

```
.NLIST  MEB
        .
        .  (MACRO references)
        .
.LIST   MEB
```

In this example, you disable the listing of MEB (Macro Expansion Binary) code with the .NLIST directive and later resume MEB listing with the .LIST directive. If you include /LI:MEB in the assembly command string, however, MACRO ignores both the .NLIST MEB and the .LIST MEB directives. This enables MEB listing throughout the program.

2.2.2 Function Control Switches

The /EN:arg and /DS:arg switches let you enable or disable functions at assembly time and control the form and content of the binary object file. These switches override .ENABL and .DISABL directives in the source program.

Table 2–3 summarizes the /EN and /DS function arguments, their default status, and the functions they control. See the *PDP–11 MACRO–11 Language Reference Manual* for more information on the meaning of the arguments.

Table 2–3: Arguments for /EN and /DS switches

Argument	Function	Default
ABS	Absolute binary output.	Disable
AMA	Assembly of all absolute addresses as relative addresses.	Disable
CDR	Source columns from 73 on are reserved for comments.	Disable
FPT	Floating point truncation.	Disable
GBL	Undefined symbols treated as globals.	Disable
LC	Lowercase ASCII input.	Disable
LSB	Local symbol block.	Disable
PNC	Binary output.	Enable
REG	Mnemonic definitions of registers. MACRO uses the following definitions: R0 = %0 R1 = %1 R2 = %2 R3 = %3 R4 = %4 R5 = %5 SP = %6 PC = %7	Enable

2.2.3 Macro Library Switch

Before using the macro library switch, you need to understand what macro libraries are and how the assembler and Task Builder handle them. Macro libraries are specially formatted files containing macro call definitions or object module subroutines. The MACRO assembler and the Task Builder can access these library files and extract code from them. Libraries are convenient to use because they encourage sharing of code.

RSTS/E comes with a set of standard libraries. You can also create your own libraries with LBR, described in Chapter 3. The library files you create using LBR are in the same format as those that DIGITAL supplies with the operating system.

LB:RSXMAC.SML is one of the libraries that DIGITAL provides. It contains definitions for all system directives. You do not have to specify LB:RSXMAC.SML in the command string; MACRO automatically searches it as if it were the last source file in the command string. However, if you include a macro library in the command string other than LB:RSXMAC.SML, macro definitions contained in that library override the macro definitions in LB:RSXMAC.SML.

Use the /ML switch to designate a source file as a macro library. The /ML switch specifies that macros referenced in the .MCALL directive come from a library other than LB:RSXMAC.SML. When MACRO encounters the /ML switch, it searches the designated library first. If the macro definition is not found in the designated library, MACRO then searches LB:RSXMAC.SML.

When MACRO locates an .MCALL directive in the source code, it searches macro libraries that are named in the command string in reverse order. Thus, if two or more macro libraries contain definitions of the same macro name, the macro library that appears last in the command string takes precedence. For example:

MAC><output file specification>=ALIB.MLB/ML,BLIB.MLB/ML,XIZ

Assume that each of the two macro libraries, ALIB and BLIB, contains a macro called .BIG, but with different definitions. If source file XIZ contains a macro call .MCALL .BIG, the system includes in the program the definition of .BIG that appears in the macro library BLIB.

2.2.4 Assembly Pass Switch

Use the /PA:arg switch to specify whether the source file assembles in pass 1 or pass 2. The /PA:arg switch is meaningful only if you add it to a source input file specification.

The specification /PA:1 calls for assembly of the file during pass 1 only. During the first pass, the assembler groups all symbols as either local or global, performs statement generation, locates all macro symbols, and, if necessary, reads the macro definitions from libraries. At the end of pass 1, the assembler will have processed all local references (such as undefined global symbols) that are to be resolved by the Task Builder.

Use the /PA:1 switch for files that assemble completely at the end of pass 1. Definition files (prefix files containing only symbol definitions) are a good example. By specifying /PA:1 for these files, you can cause MACRO to skip processing of these files in pass 2. In some cases, this procedure can save considerable assembly time.

If you have conditional assemblies or use .PSECTs in your program, do not use the /PA:1 switch. Using /PA:1 in these cases can cause errors in the table of contents listing or can result in link time errors. The next two paragraphs provide more details about these errors.

The table of contents listing may contain inaccurate page numbers if a portion of a program is not assembled during pass 2 (using the /PA:1 switch). This is because page numbers are sent to the listing file during pass 1, while listing pages are sent to the listing file during pass 2. Page numbering may change as the result of conditional assemblies (assembled during pass 2), but these new page numbers will not appear in the table of contents listing.

You must also be careful in declaring .PSECTs, because the GSD (Global Symbol Definition) portion of the .OBJ file is output at the end of pass 1. If a portion of your program redefines a .PSECT, or changes the size of a .PSECT, this will not be reflected in the .OBJ file and will result in link time errors.

The specification /PA:2 calls for assembly of the file during pass 2 only. During the second pass, the assembler actually generates the object module and listing files, flagging with an error code in the listing file those source statements containing errors. This switch (/PA:2), though not very useful, is provided as part of the syntax.

2.3 Error Messages

MACRO can detect two types of errors: input-output and programming. Input-output errors occur when you specify incorrect command strings to MACRO or when problems arise with I/O devices. These errors appear on your terminal when you assemble your program. Programming errors are mistakes in source code syntax or faulty program logic. These codes automatically appear on the assembly listings.

This section describes input-output errors; see Appendix A for a description of the single character codes that identify MACRO programming errors.

All the error messages listed here, except for the "MAC — COMMAND I/O ERROR" message, cause the current assembly to end. MACRO then tries to restart by reading another command line. If a command I/O error occurs, MACRO exits because it cannot get additional command line input.

```
MAC -- COMMAND FILE OPEN FAILURE
```

Description

MACRO cannot open the indirect command file. See "OPEN FAILURE ON INPUT FILE" for meaning.

Suggested Response

Make sure the command file exists and you have read access to the file. Check spelling errors in your command line.

```
MAC -- COMMAND I/O ERROR
```

Description

The file system detects an error during MACRO's attempt to read a command line. This is a fatal error and causes MACRO to exit. The device may be off line or write-protected, or a bad block may exist in the file.

Suggested Response

Check the command file for correct contents. If the indirect command file contains a bad disk block, re-create the file.

```
MAC -- COMMAND SYNTAX ERROR
```

Description

MACRO detects an error in the syntax of the command line.

Suggested Response

Check the command line for spelling errors. Make sure you have specified an input file. Check all switches for correct spelling and correct arguments.

```
MAC -- ILLEGAL FILENAME
```

Description

The input file specification is missing or the input or output file specification contains an illegal character. You cannot use the wildcard characters (?) and (*) in MACRO file specifications.

Suggested Response

Check for spelling errors in the command line. Check file specifications for correct format.

```
MAC -- ILLEGAL SWITCH
```

Description

MACRO does not recognize the specified switch.

Suggested Response

Check for spelling errors in the command line. Make sure all switch arguments are legal.

MAC -- INDIRECT COMMAND SYNTAX ERROR

Description

The name of the indirect command file specified in the MACRO command line is syntactically incorrect.

Suggested Response

Check for spelling errors in the command line. Make sure there are no wildcards in your file specification.

MAC -- INDIRECT FILE DEPTH EXCEEDED

Description

Indirect command files are nested to more than three levels.

Suggested Response

Restructure the command files so they are not nested to more than three levels.

MAC -- INSUFFICIENT DYNAMIC MEMORY

Description

The assembler has run out of space.

Suggested Response

Segment your program and assemble the pieces or have the system manager increase the system's swap maximum.

MAC -- INVALID FORMAT IN MACRO LIBRARY

Description

The library file is corrupt (contains bad data) or the Librarian Utility Program (LBR) did not create it.

Suggested Response

Make sure you are using the correct file. If the library file is corrupt, rebuild it using LBR. (See Chapter 3 for details.)

MAC -- I/O ERROR ON INPUT FILE

Description

The file system detected an error while reading a record from a source input file or MACRO library file. For example, it found a line containing more than 132 characters. This message can also indicate a problem with a device, a corrupt source file, or a corrupt MACRO library.

Suggested Response

Make sure all input lines do not exceed 132 characters. Check input file specifications for errors.

```
MAC -- I/O ERROR ON MACRO LIBRARY FILE
```

Description

This message has the same meaning as I/O ERROR ON INPUT FILE, except that the file is a MACRO library file and not a source input file.

Suggested Response

Make sure the library file has the correct format and is not corrupt. Check all library file specifications.

```
MAC -- I/O ERROR ON OUTPUT FILE
```

Description

The file system detected an error in writing a record to the object output file or the listing output file. This message can also indicate that a device problem exists or the device is full.

Suggested Response

Ensure that there is enough space on the output media, the output media is on line and ready, and the device is not write-locked.

```
MAC -- I/O ERROR ON WORK FILE
```

Description

A read or write error occurred on the work file used to store the symbol table. A read or write error occurs when there is a hardware problem on a device or an attempt to write to a device that is full.

Suggested Response

Ensure that there is enough space on the output media, the output media is on line and ready, and the device is not write-locked.

```
MAC -- OPEN FAILURE ON INPUT FILE
```

Description

One of the following conditions causes this error:

1. The specified device, directory, or file does not exist.
2. The volume is not mounted.

3. You do not have access to the file.

4. A hardware problem exists with the device.

Suggested Response

Check for each error condition. Also check for spelling errors in the input file specifications.

```
MAC -- OPEN FAILURE ON OUTPUT FILE
```

Description

One of the following conditions causes this error:

1. The specified device or directory does not exist.

2. The volume is not mounted.

3. The volume is full or the device is write-protected.

4. A hardware problem exists with the device.

Suggested Response

Check for each error condition. Also check for spelling errors in the output file specifications.

```
MAC -- 64K STORAGE LIMIT EXCEEDED
```

Description

64K words of work file memory are available to MACRO. This message indicates that the assembler has generated too many symbols (13,000 to 14,000) and it has run out of space. This means either the source program is too large or it contains a condition that leads to excessive size, such as a macro expansion that recursively calls itself without a terminating condition.

Suggested Response

Check for recursive macro expansions. Try to use fewer macros. Segment the program and assemble the separate parts, using global references.

Index

HOW TO ORDER ADDITIONAL DOCUMENTATION

DIRECT TELEPHONE ORDERS

In Continental USA and Puerto Rico call **800–258–1710**

In Canada call **800–267–6146**

In New Hampshire, Alaska or Hawaii call **603–884–6660**

DIRECT MAIL ORDERS (U.S. and Puerto Rico*)

DIGITAL EQUIPMENT CORPORATION
P.O. Box CS2008
Nashua, New Hampshire 03061

DIRECT MAIL ORDERS (Canada)

DIGITAL EQUIPMENT OF CANADA LTD.
940 Belfast Road
Ottawa, Ontario, Canada K1G 4C2
Attn: A&SG Business Manager

INTERNATIONAL

DIGITAL EQUIPMENT CORPORATION
A&SG Business Manager
c/o Digital's local subsidary
or approved distributor

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Northboro, Massachusetts 01532

*Any prepaid order from Puerto Rico must be placed with the Local Digital Subsidiary:
809–754–7575

RSTS/E
Programmer's Utilities Manual
AA–D749A–TC
AD–D749A–T1, T2

Reader's Comments

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

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

_____

_____

_____

_____

_____

_____

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

_____

_____

_____

_____

_____

_____

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

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

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

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

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN: Commercial Engineering Publications MK1-2/H3
DIGITAL EQUIPMENT CORPORATION
CONTINENTAL BOULEVARD
MERRIMACK, N.H. 03054

Do Not Tear - Fold Here and Tape

Cut Along Dotted Line