## UPDATE NOTICE 1

# PDP–11 MACRO–11 Language Reference Manual

AD–V027A–T1

**May 1984**

**NEW AND CHANGED INFORMATION**

This update contains changes and additions to the *PDP–11 MACRO–11 Language Reference Manual*, AA–V027A–TC.

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

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

digital equipment corporation · maynard, massachusetts

# INSTRUCTIONS

The enclosed pages are replacements for or additions to current pages of the *PDP–11 MACRO–11 Language Reference Manual*. On replacement pages, changes and additions are indicated by vertical bars ( ▌ ); deletions are indicated by bullets (•).

Keep this notice in your manual to maintain an up-to-date record of changes.

© Digital Equipment Corporation 1984.
All Rights Reserved.
Printed in U.S.A.

| Old page | New page |
|---|---|
| Title/Copyright | Title/Copyright |
| v/vi and vii/viii | v/vi and vii/viii |
| 1–1/1–2 | 1–1/1–2 |
| 2–3/2–4 | 2–3/2–4 |
| 3–1/3–2 | 3–1/3–2 |
| | 3–2.1/blank |
| 3–11/3–12 | 3–11/3–12 |
| 5–3/5–4 | 5–3/5–4 |
| 6–15/6–16 thru 6–19/6–20 | 6–15/6–16 thru 6–19/6–20 |
| 6–25/6–26 and 6–27/6–28 | 6–25/6–26 and 6–27/6–28 |
| 6–55/6–56 | 6–55/6–56 |
| 6–61/blank | 6–61/blank |
| 7–5/7–6 | 7–5/7–6 |
| 7–9/7–10 | 7–9/7–10 |
| 7–13/7–14 thru 7–19/7–20 | 7–13/7–14 thru 7–19/7–20 |
| 8–7/8–8 | 8–7/8–8 |
| 9–3/9–4 | 9–3/9–4 |
| A–1/A–2 | A–1/A–2 |
| A–5/blank | A–5/A–6 |
| J–1/J–2 | J–1/J–2 |
| | J–2.1/blank |
| Index–1/Index–2 thru Index–5/blank | Index–1/Index–2 thru Index–5/Index–6 |
| Reader's Comments/Mailer | Reader's Comments/Mailer |

# PDP–11 MACRO–11
# Language Reference Manual

AA–V027A–TC

**March 1983**

This document describes how to use the MACRO–11 relocatable assembler to develop PDP–11 assembly language programs. Although no prior knowledge of MACRO–11 is required, the user should be familiar with the PDP–11 processor addressing modes and instruction set. This manual presents detailed descriptions of MACRO–11's features, including source and command string control of assembly and listing functions, directives for conditional assembly and program sectioning, and user-defined and system macro libraries. The chapters on operating procedures previously were found in two separate manuals (the *PDP–11 MACRO–11 Language Reference Manual* and the *IAS/RSX MACRO–11 Reference Manual*). This manual should be used with a system-specific user's guide as well as a Linker or a Task Builder manual.

This manual supersedes previous editions, AA–5075B–TC, published 1980, AA–5075A–TC, published 1977, and DEC–11–OIMRA–B–D, published 1976. This manual contains Update Notice 1, AD–V027A–T1.

**Operating System:** IAS Version 2
MICRO/RSX Version 1
MICRO/RSTS Version 1
VAX/VMS Version 4
RSTS/E Version 8
RSX–11M Version 4
RSX–11M–PLUS Version 2
RT–11 Version 5

**Software:** MACRO–11 Version 5.2

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

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

digital equipment corporation · maynard, massachusetts

First Printing, August 1977
Revised, January 1980
Updated, December 1981
Revised, March 1983
Updated, May 1984

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.

© Digital Equipment Corporation 1977, 1980, 1981, 1983, 1984.
All Rights Reserved.

Printed in U.S.A.

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

The following are trademarks of Digital Equipment Corporation:

| | | |
|---|---|---|
| | digital™ | |
| DEC | MASSBUS | RT–11 |
| DECmate | PDP | UNIBUS |
| DECsystem–10 | P/OS | VAX |
| DECSYSTEM–20 | Professional | VMS |
| DECUS | Rainbow | VT |
| DECwriter | RSTS | Work Processor |
| DIBOL | RSX | |

CHAPTER 7 MACRO DIRECTIVES 7-1

7.1 DEFINING MACROS 7-1
7.1.1 .MACRO Directive 7-1
7.1.2 .ENDM Directive 7-2
7.1.3 .MEXIT Directive 7-3
7.1.4 MACRO Definition Formatting 7-4
7.2 CALLING MACROS 7-4
7.3 ARGUMENTS IN MACRO DEFINITIONS AND MACRO CALLS 7-4
7.3.1 Macro Nesting 7-6
7.3.2 Special Characters in Macro Arguments 7-7
7.3.3 Passing Numeric Arguments as Symbols 7-7
7.3.4 Number of Arguments in Macro Calls 7-8
7.3.5 Creating Local Symbols Automatically 7-8
7.3.6 Keyword Arguments 7-10
7.3.7 Concatenation of Macro Arguments 7-11
7.4 MACRO ATTRIBUTE DIRECTIVES: .NARG, .NCHR, AND .NTYPE 7-12
7.4.1 .NARG Directive 7-12
7.4.2 .NCHR Directive 7-13
7.4.3 .NTYPE Directive 7-14
7.5 .ERROR AND .PRINT DIRECTIVES 7-16
7.6 INDEFINITE REPEAT BLOCK DIRECTIVES: .IRP AND .IRPC 7-17
7.6.1 .IRP Directive 7-17
7.6.2 .IRPC Directive 7-18
7.7 REPEAT BLOCK DIRECTIVE: .REPT, .ENDR 7-20
7.8 MACRO LIBRARY DIRECTIVE: .MCALL 7-20
7.9 MACRO DELETION DIRECTIVE: .MDELETE 7-21

PART IV OPERATING PROCEDURES

CHAPTER 8 IAS/RSX-11M/RSX-11M-PLUS OPERATING PROCEDURES 8-1

8.1 RSX-11M/RSX-11M-PLUS OPERATING PROCEDURES 8-1
8.1.1 Initiating MACRO-11 Under RSX-11M/RSX-11M-PLUS 8-2
8.1.1.1 Method 1 - Direct MACRO-11 Call 8-2
8.1.1.2 Method 2 - Single Assembly 8-2
8.1.1.3 Method 3 - Install, Run Immediately, and Remove On Exit 8-2
8.1.1.4 Method 4 - Using the Indirect Command Processor 8-3
8.1.2 Default File Specifications 8-3
8.1.3 MCR Command String Format 8-4
8.1.4 DCL Operating Procedures 8-8
8.1.5 MACRO-11 Command String Examples 8-13
8.2 IAS MACRO-11 OPERATING PROCEDURES 8-14
8.2.1 Initiating MACRO-11 Under IAS 8-14
8.2.2 IAS Command String 8-14
8.2.3 IAS Indirect Command Files 8-16
8.2.4 IAS Command String Examples 8-16
8.3 CROSS-REFERENCE PROCESSOR (CREF) 8-17
8.4 IAS/RSX-11M/RSX-11M-PLUS FILE SPECIFICATION 8-19
8.5 MACRO-11 ERROR MESSAGES UNDER IAS/RSX-11M/RSX-11M-PLUS 8-20

CHAPTER 9 RSTS/RT-11 OPERATING PROCEDURES 9-1

9.1 MACRO-11 UNDER RSTS 9-1
9.1.1 RT-11 Through RSTS 9-1
9.1.2 RSX Through RSTS 9-1

| | | | |
|---|---|---|---|
| | 9.2 | INITIATING MACRO-11 UNDER RT-11 | 9-2 |
| | 9.3 | RT-11 COMMAND STRING | 9-2 |
| | 9.4 | FILE SPECIFICATION OPTIONS | 9-4 |
| | 9.5 | CROSS-REFERENCE (CREF) TABLE GENERATION OPTION | 9-5 |
| | 9.5.1 | Obtaining a Cross-Reference Table | 9-5 |
| | 9.5.2 | Handling Cross-Reference Table Files | 9-7 |
| | 9.5.3 | MACRO-11 Error Messages Under RT-11 | 9-7 |
| APPENDIX | A | MACRO-11 CHARACTER SETS | A-1 |
| | A.1 | ASCII CHARACTER SET | A-1 |
| | A.2 | RADIX-50 CHARACTER SET | A-4 |
| | A.3 | DEC MULTINATIONAL CHARACTER SET | A-6 |
| APPENDIX | B | MACRO-11 ASSEMBLY LANGUAGE AND ASSEMBLER DIRECTIVES | B-1 |
| | B.1 | SPECIAL CHARACTERS | B-1 |
| | B.2 | SUMMARY OF ADDRESS MODE SYNTAX | B-1 |
| | B.3 | ASSEMBLER DIRECTIVES | B-3 |
| APPENDIX | C | PERMANENT SYMBOL TABLE (PST) | C-1 |
| | C.1 | OP CODES | C-1 |
| | C.2 | MACRO-11 DIRECTIVES | C-5 |
| APPENDIX | D | ERROR MESSAGES | D-1 |
| APPENDIX | E | SAMPLE CODING STANDARD | E-1 |
| | E.1 | LINE FORMAT | E-1 |
| | E.2 | COMMENTS | E-1 |
| | E.3 | NAMING STANDARDS | E-2 |
| | E.3.1 | Registers | E-2 |
| | E.3.1.1 | General Purpose Registers | E-2 |
| | E.3.1.2 | Hardware Registers | E-2 |
| | E.3.1.3 | Device Registers | E-2 |
| | E.3.2 | Processor Priority | E-2 |
| | E.3.3 | Symbols | E-3 |
| | E.3.3.1 | Symbol Examples | E-3 |
| | E.3.3.2 | Local Symbols | E-4 |
| | E.3.3.3 | Global Symbols | E-4 |
| | E.3.3.4 | Macro Names | E-4 |
| | E.3.3.5 | General Symbols | E-4 |
| | E.4 | PROGRAM MODULES | E-5 |
| | E.4.1 | The Module Preface | E-5 |
| | E.4.2 | The Module | E-5 |
| | E.4.3 | Module Example | E-7 |
| | E.4.4 | Modularity | E-8 |
| | E.4.4.1 | Calling Conventions (Inter-Module/ Intra-Module) | E-9 |
| | E.4.4.2 | Exiting | E-9 |
| | E.4.4.3 | Success/Failure Indication | E-9 |
| | E.4.4.4 | Module Checking Routines | E-9 |
| | E.5 | CODE FORMAT | E-10 |
| | E.5.1 | Program Flow | E-10 |
| | E.5.2 | Common Exits | E-11 |
| | E.5.3 | Code with Interrupts Inhibited | E-12 |
| | E.5.4 | Code in System State | E-12 |
| | E.6 | INSTRUCTION USAGE | E-13 |
| | E.6.1 | Forbidden Instructions | E-13 |
| | E.6.2 | Conditional Branches | E-14 |
| | E.7 | PROGRAM SOURCE FILES | E-14 |

| | | | |
|---|---|---|---|
| | E.8 | PDP-11 VERSION NUMBER STANDARD | E-14 |
| | E.8.1 | Displaying the Version Identifier | E-15 |
| | E.8.2 | Use of the Version Number in the Program | E-15 |
| APPENDIX | F | ALLOCATING VIRTUAL MEMORY | F-1 |
| | F.1 | GENERAL HINTS AND SPACE-SAVING GUIDELINES | F-1 |
| | F.2 | MACRO DEFINITIONS AND EXPANSIONS | F-2 |
| | F.3 | OPERATIONAL TECHNIQUES | F-3 |
| APPENDIX | G | WRITING POSITION-INDEPENDENT CODE | G-1 |
| | G.1 | INTRODUCTION TO POSITION-INDEPENDENT CODE | G-1 |
| | G.2 | EXAMPLES | G-2 |
| APPENDIX | H | SAMPLE ASSEMBLY AND CROSS-REFERENCE LISTING | H-1 |
| APPENDIX | I | OBSOLETE MACRO-11 DIRECTIVES, SYNTAX, AND COMMAND LINE OPTIONS | I-1 |
| | I.1 | OBSOLETE DIRECTIVES AND SYNTAX | I-1 |
| | I.2 | OBSOLETE COMMAND LINE OPTION | I-1 |
| APPENDIX | J | RELEASE NOTES | J-1 |
| | J.1 | CHANGES -- ALL VERSIONS OF MACRO-11 | J-1 |
| | J.1.1 | V5.2 Update Changes | J-1 |
| | J.1.2 | V5.1 Changes | J-2 |
| | J.2 | CHANGES -- MACRO-11/RSX VERSION ONLY | J-3 |
| | J.3 | CHANGES -- MACRO-11/RT VERSION ONLY | J-3 |


## FIGURES


| | | | |
|---|---|---|---|
| FIGURE | 3-1 | Assembly Listing Showing Local Symbol Block | 3-12 |
| | 3-2 | Sample Assembly Results | 3-13 |
| | 6-1 | Example of Line Printer Assembly Listing | 6-5 |
| | 6-2 | Example of Teleprinter Assembly Listing | 6-7 |
| | 6-3 | Listing Produced with Listing Control Directives | 6-13 |
| | 6-4 | Assembly Listing Table of Contents | 6-16 |
| | 6-5 | Example of .ENABL and .DSABL Directives | 6-21 |
| | 6-6 | Example of .BLKB and .BLKW Directives | 6-39 |
| | 6-7 | Example of .SAVE and .RESTORE Directives | 6-50 |
| | 7-1 | Example of .NARG Directive | 7-13 |
| | 7-2 | Example of .NCHR Directive | 7-14 |
| | 7-3 | Example of .NTYPE Directive in Macro Definition | 7-15 |
| | 7-4 | Example of .IRP and .IRPC Directives | 7-19 |
| | 8-1 | Sample CREF Listing | 8-19 |
| | G-1 | Example of Position-Dependent Code | G-3 |
| | G-2 | Example of Position-Independent Code | G-3 |


## TABLES


| | | | |
|---|---|---|---|
| TABLE | 3-1 | Special Characters Used in MACRO-11 | 3-1 |
| | 3-2 | Legal Separating Characters | 3-3 |
| | 3-3 | Legal Argument Delimiters | 3-4 |
| | 3-4 | Legal Unary Operators | 3-4 |

| | | |
|---|---|---|
| 3-5 | Legal Binary Operators | 3-5 |
| 5-1 | Addressing Modes | 5-1 |
| 5-2 | Symbols Used in Chapter 5 | 5-2 |
| 6-1 | Directives in Chapter 6 | 6-1 |
| 6-2 | Symbolic Arguments of Listing Control Directives | 6-10 |
| 6-3 | Symbolic Arguments of Function Control Directives | 6-19 |
| 6-4 | Symbolic Arguments of .PSECT Directive | 6-41 |
| 6-5 | Program Section Default Values | 6-48 |
| 6-6 | Legal Condition Tests for Conditional Assembly Directives | 6-54 |
| 6-7 | Subconditional Assembly Block Directives | 6-57 |
| 8-1 | File Specification Default Values | 8-4 |
| 8-2 | MACRO-11 File Specification Switches | 8-6 |
| 8-3 | DCL Command Qualifiers | 8-8 |
| 8-4 | DCL Parameter Qualifiers | 8-13 |
| 9-1 | Default File Specification Values | 9-3 |
| 9-2 | File Specification Options | 9-4 |
| 9-3 | /C Option Arguments | 9-6 |
| I-1 | Old and New Directives and Syntax | I-1 |

CHAPTER 1

# THE MACRO-11 ASSEMBLER

MACRO-11 provides the following features:

1. Source and command string control of assembly functions
2. Device and filename specifications for input and output files
3. Error listing on command output device
4. Alphabetized, formatted symbol table listing; optional cross-reference listing of symbols
5. Relocatable object modules
6. Global symbols for linking object modules
7. Conditional assembly directives
8. Program sectioning directives
9. User-defined macros and macro libraries
10. Comprehensive system macro library
11. Extensive source and command string control of listing functions.

MACRO-11 assembles one or more source files containing MACRO-11 statements into a single relocatable binary object file. The output of MACRO-11 consists of a binary object file and a file containing the table of contents, the assembly listing, and the symbol table. An optional cross-reference listing of symbols and macros is available. A sample assembly listing is provided in Appendix H.

## 1.1 ASSEMBLY PASS 1

During pass 1, MACRO-11 locates and reads all required macros from libraries, builds symbol tables and program section tables for the program, and performs a rudimentary assembly of each source statement.

In the first step of assembly pass 1, MACRO-11 initializes all the impure data areas (areas containing both code and data) that will be used internally for the assembly process. These areas include all dynamic storage and buffer areas used as file storage regions.

MACRO-11 then calls a system subroutine which transfers a command line into memory. This command line contains the specifications of all files to be used during assembly. After scanning the command line for proper syntax, MACRO-11 initializes the specified output files. These files are opened to determine if valid output file specifications have been passed in the command line.

MACRO-11 now initiates a routine which retrieves source lines from the input file. If no input file is open, as is the case at the beginning of assembly, MACRO-11 opens the next input file specified in the command line and starts assembling the source statements. MACRO-11 first determines the length of the instructions, then assembles them according to length as one word, two words, or three words.

At the end of assembly pass 1, MACRO-11 reopens the output files described above. Such information as the object module name, the program version number, and the global symbol directory (GSD) for each program section are output to the object file to be used later in linking the object modules. After writing out the GSD for a given program section, MACRO-11 scans through the symbol tables to find all the global symbols that are bound to that particular program section. MACRO-11 then writes out GSD records to the object file for these symbols. This process is done for each program section.

## 1.2 ASSEMBLY PASS 2

On pass 2 MACRO-11 writes the object records to the output file while generating both the assembly listing and the symbol table listing for the program. A cross-reference listing may also be generated.

Basically, assembly pass 2 consists of the same steps performed in assembly pass 1, except that all source statements containing MACRO-11-detected errors are flagged with an error code as the assembly listing file is created. The object file that is created as the final consequence of pass 2 contains all the object records, together with relocation records that hold the information necessary for linking the object file.

The information in the object file, when passed to the Task Builder or Linker, enables the global symbols in the object modules to be associated with absolute or virtual memory addresses, thereby forming an executable body of code.

The user may wish to become familiar with the macro object file format and description. This information is presented in the applicable system manual (see Section Ø.3 in the Preface).

The legal characters for defining labels are:

A through Z
Ø through 9
. (Period)
$ (Dollar Sign)

NOTE

By convention, the dollar sign ($) and period (.) are reserved for use in defining DIGITAL system software symbols. Therefore these characters should not be used in defining labels in MACRO-11 source programs.

A label may be any length; however, only the first six characters are significant and, therefore, must be unique among all the labels in the source program. An error code (M) is generated in the assembly listing if the first six characters in two or more labels are the same.

A symbol used as a label must not be redefined within the source program. If the symbol is redefined, a label with a multiple definition results, causing MACRO-11 to generate an error code (M) in the assembly listing. Furthermore, any statement in the source program which references a multi-defined label generates an error code (D) in the assembly listing.

### 2.2.2 Operator Field

The operator field specifies the action to be performed. It may consist of an instruction mnemonic (op code), an assembler directive, or a macro call. Chapters 6 and 7 describe these three types of operators.

When the operator is an instruction mnemonic, a machine instruction is generated and MACRO-11 evaluates the addresses of the operands which follow. When the operator is a directive MACRO-11 performs certain control actions or processing operations during the assembly of the source program. When the operator is a macro call, MACRO-11 inserts the code generated by the macro expansion.

Leading and trailing spaces or tabs in the operator field have no significance; such characters serve only to separate the operator field from the preceding and following fields.

An operator is terminated by a space, tab, or any non-RAD5Ø character*, as in the following examples:

```
MOV A,B         ;The space terminates the operator MOV.

MOV     A,B     ;The tab terminates the operator MOV.

MOV@A,B         ;The @ character terminates the operator MOV.
```

* Appendix A.2 contains a table of Radix-5Ø characters.

Although the statements above are all equivalent in function, the second statement is the recommended form because it conforms to MACRO-11 coding conventions.

### 2.2.3 Operand Field

When the operator is an instruction mnemonic (op code), the operand field contains program variables that are to be evaluated/manipulated by the operator. The operand field may also supply arguments to MACRO-11 directives and macro calls, as described in Chapters 6 and 7, respectively.

Operands may be expressions or symbols, depending on the operator. Multiple expressions used in the operand field of a MACRO-11 statement must be separated by a comma; multiple symbols similarly used may be delimited by any legal separator (a comma, tab, and/or space). An operand should be preceded by an operator field; if it is not, the statement is treated by MACRO-11 as an implicit .WORD directive (see Section 6.3.2).

When the operator field contains an op code, associated operands are always expressions, as shown in the following statement:

```
MOV     R0,A+2(R1)
```

On the other hand, when the operator field contains a MACRO-11 directive or a macro call, associated operands are normally symbols, as shown in the following statement:

```
.MACRO  ALPHA  SYM1,SYM2
```

Refer to the description of each MACRO-11 directive (Chapter 7) to determine the type and number of operands required in issuing the directive.

The operand field is terminated by a semicolon when the field is followed by a comment. For example, in the following statement:

```
LABEL:  MOV     A,B     ;Comment field
```

the tab between MOV and A terminates the operator field and defines the beginning of the operand field; a comma separates the operands A and B; and a semicolon terminates the operand field and defines the beginning of the comment field. When no comment field follows, the operand field is terminated by the end of the source line.

### 2.2.4 Comment Field

The comment field normally begins in column 33 and extends through the end of the line. This field is optional and may contain any 7-bit ASCII or 8-bit DEC Multinational characters except null, RUBOUT, carriage-return, line-feed, vertical-tab or form-feed. All other characters appearing in the comment field, even special characters reserved for use in MACRO-11, are checked only for ASCII legality and then included in the assembly listing as they appear in the source text.

CHAPTER 3

# SYMBOLS AND EXPRESSIONS

This chapter describes the components of MACRO-11 instructions: the character set, the conventions observed in constructing symbols, and the use of numbers, operators, terms and expressions.

## 3.1 CHARACTER SET

The following characters are legal in MACRO-11 source programs:

1. The letters A through Z. Both upper- and lower-case letters are acceptable, although, upon input, lower-case letters are converted to upper-case (see Section 6.2.1, .ENABL LC).

2. Characters in the DEC Multinational character set (MCS). A chart showing the MCS is located in Appendix A, with a list of directives that support the MCS. Specific support for the MCS is included with the description of each directive.

3. The digits Ø through 9.

4. The characters . (period) and $ (dollar sign). These characters are reserved for use as Digital Equipment Corporation system program symbols.

5. The special characters listed in Table 3-1.

Table 3-1
Special Characters Used in MACRO-11

| Character | Designation | Function |
|---|---|---|
| : | Colon | Label terminator. |
| :: | Double colon | Label terminator; defines the label as a global label. |
| = | Equal sign | Direct assignment operator and macro keyword indicator. |
| == | Double equal sign | Direct assignment operator; defines the symbol as a global symbol. |

(continued on next page)

Table 3-1 (Cont.)
Special Characters Used in MACRO-11

| Character | Designation | Function |
|---|---|---|
| =: | Equal sign colon | Direct assignment operator; macro keyword indicator; causes error (M) in listing if an attempt is made to change the value of the symbol. |
| ==: | Double equal sign colon | Direct assignment operator; defines the symbol as a global symbol; causes error (M) in listing if an attempt is made to change the value of the symbol. |
| % | Percent sign | Register term indicator. |
| | Tab | Item or field terminator. |
| | Space | Item or field terminator. |
| # | Number sign | Immediate expression indicator. |
| @ | At sign | Deferred addressing indicator. |
| ( | Left parenthesis | Initial register indicator. |
| ) | Right parenthesis | Terminal register indicator. |
| . | Period | Current location counter. |
| , | Comma | Operand field separator. |
| ; | Semicolon | Comment field indicator. |
| < | Left angle bracket | Initial argument or expression indicator. |
| > | Right angle bracket | Terminal argument or expression indicator. |
| + | Plus sign | Arithmetic addition operator or autoincrement indicator. |
| - | Minus sign | Arithmetic subtraction operator or autodecrement indicator. |
| * | Asterisk | Arithmetic multiplication operator. |
| / | Slash | Arithmetic division operator. |

(continued on next page)

Table 3-1 (Cont.)
Special Characters Used in MACRO-11

| Character | Designation | Function |
|---|---|---|
| & | Ampersand | Logical AND operator. |
| ! | Exclamation point | Logical inclusive OR operator. |
| " | Double quote | Double ASCII character indicator. |

(continued on next page)

The % character may be used with any legal term or expression to specify a register. For example, the statement

```
CLR     %3+1
```

is equivalent in function to the statement

```
CLR     %4
```

and clears the contents of register 4.

In contrast, the statement

```
CLR     4
```

clears the contents of virtual memory location 4.

The accumulator registers used in floating-point instructions can be defined in a similar manner. For example, with the definition

```
AC0=%0
```

the statement

```
MULF (R0),AC0
```

multiplies the contents of floating-point accumulator register AC0 by the floating-point number addressed by R0.

## 3.5 LOCAL SYMBOLS

Local symbols are specially formatted symbols used as labels within a block of coding that has been delimited as a local symbol block. Local symbols are of the form n$, where n is a decimal integer from 1 to 65535, inclusive. Examples of local symbols are:

```
  1$
 27$
 59$
104$
```

A local symbol block is delimited in one of three ways:

1. The range of a local symbol block usually consists of those statements between two normally-constructed symbolic labels (see Figure 3-1). Note that a statement of the form:

   ALPHA=EXPRESSION

   is a direct assignment statement (see Section 3.3) but does not create a label and thus does not delimit the range of a local symbol block.

2. The range of a local symbol block is normally terminated upon encountering a .PSECT, .CSECT, .ASECT, or .RESTORE directive in the source program (see Figure 3-1).

3. The range of a local symbol block is delimited through MACRO-11 directives, as follows:

   Starting delimiter: .ENABL LSB (see Section 6.2.1)

Ending delimiter: .ENABL LSB

or

one of the following:

Symbolic label (see Section 2.2.1)
.PSECT (see Section 6.7.1)
.CSECT (see Section 6.7.2)
.ASECT (see Section 6.7.2)
.RESTORE (see Section 6.7.4)

encountered after a .DSABL LSB (see Section 6.2.1).

Local symbols provide a convenient means of generating labels for branch instructions and other such references within local symbol blocks. Using local symbols reduces the possibility of symbols with multiple definitions appearing within a user program. In addition, the use of local symbols differentiates entry-point labels from local labels, since local symbols cannot be referenced from outside their respective local symbol blocks. Thus, local symbols of the same name can appear in other local symbol blocks without conflict. Local symbols do not appear in cross-reference listings and require less symbol table space than other types of symbols. Their use is recommended.

When defining local symbols, use the range from 1$ to 29999$ first. Local symbols within the range 30000$ through 65535$, inclusive, can be generated automatically as a feature of MACRO-11. Such local symbols are useful in the expansion of macros during assembly (see Section 7.3.5).

Be sure to avoid multiple definitions of local symbols within the same local symbol block. For example, if the local symbol 10$ is defined two or more times within the same local symbol block, each symbol represents a different address value. Such a multi-defined symbol causes an error code (P) to be generated in the assembly listing.

For examples of local symbols and local symbol blocks as they appear in a source program, see Figure 3-1.

```
 1                      ;+
 2                      ; Simple illustration of local symbols; the second block is delimited
 3                      ; by the label XCTPAS.
 4                      ;-
 5
 6 000000  012700  XCTPRG: MOV     #IMPURE,R0      ;Point to impure area
           000000G
 7 000004  005020  1$:     CLR     (R0)+           ;Clear a word
 8 000006  020027          CMP     R0,#IMPURT      ;Test if at top of area
           000000G
 9 000012  001374          BNE     1$              ;Iterate if not
10                                                 ;Fall in to perform pass initialization
11
12 000014  012700  XCTPAS: MOV     #IMPPAS,R0      ;Point to pass storage area
           000000G
13 000020  005020  1$:     CLR     (R0)+           ;Clear the area
14 000022  020027          CMP     R0,#IMPPAT      ;Test if at top of area
           000000G
15 000026  001374          BNE     1$              ;Iterate of not
16 000030  000207          RTS     PC              ;Return if so
```

Figure 3-1 Assembly Listing Showing Local Symbol Block

## 5.3 AUTOINCREMENT MODE

Format:

```
(ER)+
```

The contents of the register (ER) are incremented immediately after being used as the address of the operand (see Note below).

Examples:

```
CLR     (RØ)+           ;Each instruction clears
CLR     (R4)+           ;the word at the address
CLR     (R2)+           ;contained in the specified
                        ;register and increments
                        ;that register's contents
                        ;by two.
```

NOTE

Certain special instruction/address mode combinations, which are rarely or never used, do not operate the same on all PDP-11 processors, as described below.

In the autoincrement mode, both the JMP and JSR instructions autoincrement the register before its use on the PDP-11/40 but not on the PDP-11/45 or 11/10.

In double operand instructions having the addressing form Rn,(Rn)+ or Rn,-(Rn), where the source and destination registers are the same, the source operand is evaluated as the autoincremented or autodecremented value, but the destination register, at the time it is used, still contains the originally intended effective address. In the following example, as executed on the PDP-11/40, Register RØ originally contains 1ØØ(8):

```
MOV     RØ,(RØ)+        ;The quantity 1ØØ is moved
                        ;to location 1ØØ.

MOV     RØ,-(RØ)        ;The quantity 1Ø2 is moved
                        ;to location 1ØØ.
```

The use of these forms should be avoided, since they are not compatible with the entire family of PDP-11 processors.

An error code (Z) is printed in the assembly listing with each instruction which is not compatible among all members of the PDP-11 family.

## 5.4 AUTOINCREMENT DEFERRED MODE

Format:

```
@(ER)+
```

The register (ER) contains a pointer to the address of the operand. The contents of the register are incremented after being used as pointer.

Example:

```
CLR     @(R3)+          ;The contents of register 3 point
                        ;to the address of a word to be
                        ;cleared before the contents of the
                        ;register are incremented by two.
```

## 5.5 AUTODECREMENT MODE

Format:

```
-(ER)
```

The contents of the register (ER) are decremented before being used as the address of the operand (see Note in Section 5.3).

Examples:

```
CLR     -(RØ)           ;Decrement the contents of the speci-
                        ;fied register (Ø, 3, or 2) by two
CLR     -(R3)           ;before using its contents
CLR     -(R2)           ;as the address of the word to be
                        ;cleared.
```

## 5.6 AUTODECREMENT DEFERRED MODE

Format:

```
@-(ER)
```

The contents of the register (ER) are decremented before being used as a pointer to the address of the operand.

Example:

```
CLR     @-(R2)          ;Decrement the contents of
                        ;register 2 by two before
                        ;using its contents as a pointer
                        ;to the address of the word to be
                        ;cleared.
```

### 6.1.2 .TITLE Directive

.TITLE

Format:

```
.TITLE string
```

where string represents:

An identifier of one or more Radix-50 characters.

An identifier of one or more 8-bit DEC Multinational character set (MCS) characters. Any MCS character must be preceded by six Radix-50 characters.

Appendix A.2 contains a table of Radix-50 characters. Appendix A.3 contains a table of MCS characters.

The .TITLE directive assigns a name to the object module. The name assigned is the first six non-blank Radix-50 characters followed by optional characters from the MCS. MACRO-11 ignores all spaces and/or tabs up to the first non-space/non-tab character following the .TITLE directive. Any characters beyond the first six Radix-50 characters are evaluated for MCS legality.

The name of an object module (specified in the .TITLE directive) appears in the load map produced at link time. This is also the module name which the Librarian will recognize.

If the .TITLE directive is not specified, MACRO-11 assigns the default name .MAIN. to the object module. If more than one .TITLE directive is specified in the source program, the last .TITLE directive encountered during assembly pass 1 establishes the name for the entire object module.

If the .TITLE directive is specified without an object module name, or if the first non-space/non-tab character in the object module name is not Radix-50 character, the directive is flagged with an error code (A) in the assembly listing.

### 6.1.3 .SBTTL Directive

.SBTTL

Format:

```
.SBTTL string
```

where: string represents an identifier of one or more printable 7-bit ASCII or 8-bit DEC Multinational characters.

The .SBTTL directive is used to produce a table of contents immediately preceding the assembly listing and to print the text following the .SBTTL directive on the second line of the header of each page in the listing. The subheading in the text will be listed until altered by a subsequent .SBTTL directive in the program. For example, the directive:

```
.SBTTL  Conditional assemblies
```

causes the text

```
Conditional assemblies
```

to be printed as the second line in the header of the assembly listing.

During assembly pass 1, a table of contents containing the line sequence number, the page number, and the text accompanying each .SBTTL directive is printed for the assembly listing. The listing of the table of contents is suppressed whenever an .NLIST TOC directive is encountered in the source program (see Table 6-2). An example of a table of contents listing is shown in Figure 6-4.

```
MTTEMT - RT-11 MULTI-TTY EMT SE  MACRO V05.00 Saturday 08-Jan-83 10:00
TABLE OF CONTENTS

   50-   1      .MTOUT - Single character output EMT
   51-   1      .MTRCTO - Reset CTRL/O EMT
   52-   1      .MTATCH - Attach to terminal EMT
   54-   1      .MTDTCH - Detach from a terminal EMT
   55-   1      .MTPRNT - Print message EMT
   56-   1      .MTSTAT - Return multi-terminal system status EMT
   57-   1      MTTIN - Single character input
   58-   1      MTTGET - Get a character from the ring buffer
   59-   1      TTRSET - Reset terminal status bits
   60-   1      MTTPUT - Single character output
   62-   1      MTRSET  - Stop and detach all terminals attached to a Job
   63-   1      ESCAPE SEQUENCE TEST SUBROUTINE
```

Figure 6-4 Assembly Listing Table of Contents

### 6.1.4 .IDENT Directive

.IDENT

Format:

```
.IDENT  /string/
```

where: string represents a string of six or fewer Radix-50 characters which establish the program identification or version number. This string is included in the global symbol directory of the object module and is printed in the link map and librarian listing.

/ / represent delimiting characters. These delimiters may be any paired printing characters, other than the equal sign (=), the left angle bracket (<), or the semicolon (;), as long as the delimiting character is not contained within the text string itself (see Note in Section 6.3.4). If the delimiting characters do not match, or if an illegal delimiting character is used, the .IDENT directive is flagged with an error code (A) in the assembly listing.

In addition to the name assigned to the object module with the .TITLE directive (see Section 6.1.3), the .IDENT directive allows the user to label the object module with the program version number.

An example of the .IDENT directive is shown below:

```
.IDENT  /V01.00/
```

The character string is converted to Radix-50 representation and included in the global symbol directory of the object module. This character string also appears in the link map produced at link time and the Librarian directory listings.

When more than one .IDENT directive is encountered in a given program, the last such directive encountered establishes the character string which forms part of the object module identification.

The RT-11 linker allows only one .IDENT string in a program. The linker uses the first .IDENT directive encountered during the first pass to establish the character string that will be identified with all of the object modules.

The RSX-11M task builder allows an .IDENT string for each module in the program. The TASK Builder uses the first .IDENT directive in each module to establish the character string that will be identified with that module. Like the RT-11 Linker, the RSX-11M Task Builder uses the .IDENT directives encountered on the first pass.

### 6.1.5 .PAGE Directive/Page Ejection

.PAGE

Format:

```
.PAGE
```

The .PAGE directive is used within the source program to perform a page eject at desired points in the listing. This directive takes no arguments and causes a skip to the top of the next page when encountered. It also causes the page number to be incremented and the line sequence counter to be cleared. The .PAGE directive does not appear in the listing.

When used within a macro definition, the .PAGE directive is ignored during the assembly of the macro definition. Rather, the page eject operation is performed as the macro itself is expanded. In this case, the page number is also incremented.

Page ejection is accomplished in three other ways:

1. After reaching a count of 58 lines in the listing, MACRO-11 automatically performs a page eject to skip over page perforations on line printer paper and to formulate teleprinter output into pages. The page number is not changed.

2. A page eject is performed when a form-feed character is encountered. If the form-feed character appears within a macro definition, a page eject occurs during the assembly of the macro definition, but not during the expansion of the macro itself. A page eject resulting from the use of the form-feed character causes the page number to be incremented and the line sequence counter to be cleared.

3. A page eject is performed when encountering a new source file. In this case the page number is incremented and the line sequence count is reset.

### 6.1.6 .REM Directive/Begin Remark Lines

.REM

Format:

```
.REM comment-character
```

where: comment-character represents a 7-bit ASCII or 8-bit DEC Multinational character that marks the end of the comment block when the character reoccurs.

The .REM directive allows a programmer to insert a block of comments into a MACRO-11 source program without having to precede the comment lines with the comment character (;). The text between the specified delimiting characters is treated as comments. The comments may span any number of lines. For example:

```
.TITLE   Remark example
.REM     &
All the text that resides here is interpreted by MACRO-11
to be comment lines until another ampersand character is
found.  Any character may be used in place of the ampersand.&
CLR      PC
.END
```

## 6.2 FUNCTION DIRECTIVES

The following function directives are included in a source program to invoke or inhibit certain MACRO-11 functions and operations incidental to the assembly process itself.

.ENABL
.DSABL

### 6.2.1 .ENABL and .DSABL Directives

Formats:

```
.ENABL   arg
.DSABL   arg
```

where: arg represents one or more of the optional symbolic arguments defined in Table 6-3.

Specifying any argument in an .ENABL/.DSABL directive other than those listed in Table 6-3 causes that directive to be flagged with an error code (A) in the assembly listing.

Table 6-3
Symbolic Arguments of Function Control Directives

| Argument | Default | Function |
|---|---|---|
| ABS | Disable | Enabling this function produces absolute binary output in FILES-11 format. To convert this output to Formatted Binary format (as required by the Absolute Loader), use the FLX utility. |
| AMA | Disable | Enabling this function causes all relative addresses (address mode 67) to be assembled as absolute addresses (address mode 37). This function is useful during the debugging phase of program development. |
| CDR | Disable | Enabling this function causes source columns from 73 to the end of the line, to be treated as a comment. The most common use of this feature is to permit sequence numbers in card columns 73-80. |
| CRF | Enable | Disabling this function inhibits the generation of cross-reference output. This function only has meaning if cross-reference output generation is specified in the command string. |
| FPT | Disable | Enabling this function causes floating-point truncation; disabling this function causes floating-point rounding. |
| LC | Enable | Disabling this function causes MACRO-11 to convert all ASCII input to upper-case before processing it. |

(continued on next page)

Table 6-3 (Cont.)
Symbolic Arguments of Function Control Directives

| Argument | Default | Function |
|---|---|---|
| | | An example of the .ENABL LC and .DSABL LC directives, as typically used in a source program, is shown in Figure 6-5. |
| LCM | Disable | This argument, if enabled, causes the MACRO-11 conditional assembly directives .IF IDN and .IF DIF to be alphabetically case sensitive. By default, these directives are not case sensitive. |
| LSB | Disable | This argument permits the enabling or disabling of a local symbol block. Although a local symbol block is normally established by encountering a new symbolic label, a .PSECT directive or a .RESTORE directive in the source program, an .ENABL LSB directive establishes a new local symbol block which is not terminated until (1) another .ENABL LSB is encountered, or (2) another symbolic label, .PSECT directive or .RESTORE directive is encountered following a paired .DSABL LSB directive.<br><br>The basic function of this directive with regard to .PSECTS is limited to those instances where it is desirable to leave a program section temporarily to store data, followed by a return to the original program section. This temporary dismissal of the current program section may also be accomplished through the .SAVE and .RESTORE directives (see Sections 6.7.3 and 6.7.4).<br><br>Attempts to define local symbols in an alternate program section are flagged with an error code (P) in the assembly listing. |
| MCL | Disable | This argument, if enabled, causes MACRO-11 to search all known macro libraries for a macro definition that matches any undefined symbols appearing in the opcode field of a MACRO-11 statement. By default, this option is disabled. If MACRO-11 finds an unknown symbol in the opcode field, it either declares a (U) undefined symbol error, or declares the symbol an external symbol, depending on the .ENABL/.DSABL option setting of GBL (described below). |
| PNC | Enable | Disabling this function inhibits binary output until an .ENABL PNC statement is encountered within the same module. |

(continued on next page)

If an expression following the .WORD directive contains a null value, it is interpreted as a zero, as shown in the following example:

```
.=500
        .WORD   ,5,             ;Stores the values 0, 5, and 0 in
                                ;location 500, 502, and 504,
                                ;respectively.
```

A statement with a blank operator field (one that contains a symbol other than a macro call, an instruction mnemonic, a MACRO-11 directive, or a semicolon) is interpreted during assembly as an implicit .WORD directive, as shown in the example below:

```
.=440
LABEL:  100,LABEL               ;Stores the value 100 in location 440
                                ;and the value 440 in location 442.
```

NOTE

You should not use this technique to generate .WORD directives because it may not be included in future PDP-11 assemblers.

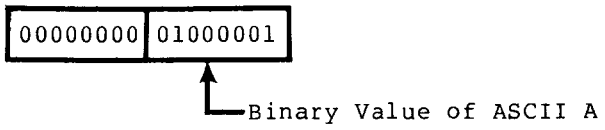
### 6.3.3 ASCII Conversion Characters

The single quote (') and the double quote (") characters are unary operators that can appear in any MACRO-11 expression. Used in MACRO-11 expressions, these characters cause a 16-bit expression value to be generated.

When the single quote is used, MACRO-11 takes the next character in the expression and converts it from its 7-bit ASCII or 8-bit DEC Multinational value to a 16-bit expression value. The high-order byte of the resulting expression value is always zero (0). The 16-bit value is then used as an absolute term within the expression. For example, the statement:

```
        MOV     #'A,R0
```

moves the following 16-bit expression value into register 0:



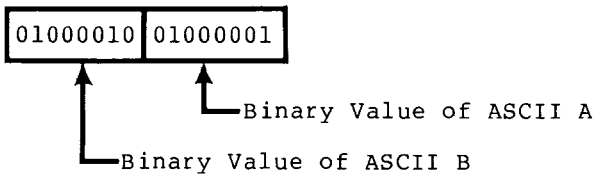

Thus the expression 'A results in a value of 101(8).

The single quote (') character must not be followed by a carriage-return, null, RUBOUT, line-feed, or form-feed character; if it is, an error code (A) is generated in the assembly listing.

When the double quote is used, MACRO-11 takes the next two characters in the expression and converts them to a 16-bit binary expression value from their 7-bit ASCII or 8-bit DEC Multinational values. This 16-bit value is then used as an absolute term within the expression. For example, the statement:

```
MOV     #"AB,RØ
```

moves the following 16-bit expression value into register Ø:




Thus the expression "AB results in a value of Ø411Ø1(8).

The double quote (") character, like the single quote (') character, must not be followed by a carriage-return, null, RUBOUT, line-feed, or form-feed character; if it is, an error code (A) is generated in the assembly listing.

The ASCII character set is listed in Appendix A.1. The DEC Multinational character set is listed in Appendix A.3.

### 6.3.4 .ASCII Directive

.ASCII

Format:

```
.ASCII  /string 1/.../string n/
```

where: string is a string of printable 7-bit ASCII or 8-bit DEC Multinational characters. The vertical-tab, null, line-feed, RUBOUT, and all other non-printable ASCII characters, except carriage-return and form-feed, cause an error code (I) if used in an .ASCII string. The carriage-return and form-feed characters are flagged with an error code (A) because these characters end the scan of the line, preventing MACRO-11 from detecting the matching delimiter at the end of the character string.

/ / represent delimiting characters. These delimiters may be any paired printing characters, other than the equal sign (=), the left angle bracket (<), or the semicolon (;) (see Note at end of section), as long as the delimiting character is not contained within the text string itself. If the delimiting characters do not match, or if an illegal delimiting character is used, the .ASCII directive is flagged with an error code (A) in the assembly listing.

The .ASCII directive translates character strings into their 7-bit ASCII or 8-bit DEC Multinational equivalents and stores them in the object module. A non-printing character can be expressed only by enclosing its equivalent octal value within angle brackets. Each set of angle brackets so used represents a single character. For example, in the following statement:

```
.ASCII  <15>/ABC/<A+2>/DEF/<5><4>
```

the expressions <15>, <A+2>, <5>, and <4> represent the values of non-printing characters. Each bracketed expression must reduce to eight bits of absolute data or less.

Angle brackets can be embedded between delimiting characters in the character string, but angle brackets so used do not take on their usual significance as delimiters for non-printing characters. For example, the statement:

```
.ASCII  /ABC<expression>DEF/
```

contains a single ASCII character string, and performs no evaluation of the embedded, bracketed expression. This use of the angle brackets is shown in the third example of the .ASCII directive below:

```
.ASCII  /HELLO/             ;Stores the binary representation
                            ;of the letters HELLO in five
                            ;consecutive bytes.

.ASCII  /ABC/<15><12>/DEF/  ;Stores the binary representation
                            ;of the characters A,B,C,carriage
                            ;return,line feed,D,E,F in eight
                            ;consecutive bytes.

.ASCII  /A<15>B/            ;Stores the binary representation
                            ;of the characters A, <, 1, 5, >,
                            ;and B in six consecutive bytes.
```

NOTE

The semicolon (;) and equal sign (=) can be used as delimiting characters in the string, but care must be exercised in so doing because of their significance as a comment indicator and assignment operator, respectively, as illustrated in the examples below:

```
.ASCII  ;ABC;/DEF/    ;Stores the binary
                      ;representation of
                      ;the characters
                      ;A, B, C, D, E, and
                      ;F in six
                      ;consecutive bytes;
                      ;not recommended
                      ;practice.
```

```
.ASCII  /ABC/;DEF;    ;Stores the binary
                      ;representations of
                      ;the characters A,
                      ;B, and C in three
                      ;consecutive bytes;
                      ;the characters D,
                      ;E, F, and ; are
                      ;treated as a
                      ;comment.

.ASCII  /ABC/=DEF=    ;Stores the binary
                      ;representation of
                      ;the characters A,
                      ;B, C, D, E, and
                      ;F in six
                      ;consecutive bytes;
                      ;not recommended
                      ;practice.
```

An equal sign is treated as an assignment operator when it appears as the first character in the ASCII string, as illustrated by the following example:

```
.ASCII  =DEF=         ;The direct
                      ;assignment
                      ;operation
                      ;.ASCII=DEF is
                      ;performed, and a
                      ;syntax error (Q)
                      ;is generated upon
                      ;encountering the
                      ;second = sign.
```

### 6.3.5 .ASCIZ Directive

.ASCIZ

Format:

```
.ASCIZ  /string 1/.../string n/
```

where: string is a string of printable 7-bit ASCII or 8-bit DEC Multinational characters. The vertical-tab, null, line-feed, RUBOUT, and all other non-printable ASCII characters, except carriage-return and form-feed, cause an error code (I) if used in an .ASCIZ string. The carriage-return and form-feed characters are flagged with an error code (A) because they end the scan of the line, preventing MACRO-11 from detecting the matching delimiter.

Table 6-6 (Cont.)
Legal Condition Tests for Conditional Assembly Directives

| Conditions | | | |
|---|---|---|---|
| Positive | Complement | Arguments | Assemble Block If: |
| IDN | DIF | Two 7-bit ASCII or 8-bit DEC Multinational macro-type arguments | Arguments are identical (or different). The .IF IDN/.IF DIF conditional directives are not alphabetically case sensitive by default. The user may enable these directives to be case sensitive by using the .ENABL option (.ENABL LCM). |

NOTE

A macro-type argument (which is a form of symbolic argument), as shown below, is enclosed within angle brackets or denoted with an up-arrow construction (as described in Section 7.3).

```
<A,B,C>
^/124/
```

An example of a conditional assembly directive follows:

```
.IF EQ  ALPHA+1      ;Assemble block if ALPHA+1=0
  .
  .
  .
.ENDC
```

The two operators & and ! have special meaning within DF and NDF conditions, in that they are allowed in grouping symbolic arguments.

| | |
|---|---|
| & | Logical AND operator |
| ! | Logical inclusive OR operator |

For example, the conditional assembly statement:

```
.IF DF  SYM1 & SYM2
  .
  .
  .
.ENDC
```

results in the assembly of the conditional block if the symbols SYM1 and SYM2 are both defined.

Nested conditional directives take the form:

```
Conditional Assembly Directive
Conditional Assembly Directive
  .
  .
  .
.ENDC
.ENDC
```

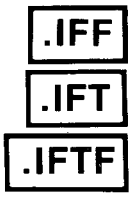
For example, the following conditional directives:

```
.IF DF  SYM1
.IF DF  SYM2
  .
  .
  .
.ENDC
.ENDC
```

can govern whether assembly is to occur. In the example above, if the outermost condition is unsatisfied, no deeper level of evaluation of nested conditional statements within the program occurs.

Each conditional assembly block must be terminated with an .ENDC directive. An .ENDC directive encountered outside a conditional assembly block is flagged with an error code (O) in the assembly listing.

MACRO-11 permits a nesting depth of 16(10) conditional assembly levels. Any statement that attempts to exceed this nesting level depth is flagged with an error code (O) in the assembly listing.




### 6.9.2 Subconditional Assembly Block Directives

Formats:

```
.IFF
.IFT
.IFTF
```

Subconditional directives may be placed within conditional assembly blocks to indicate:

1. The assembly of an alternate body of code when the condition of the block tests false.

2. The assembly of a non-contiguous body of code within the conditional assembly block, depending upon the result of the conditional test in entering the block.

3. The unconditional assembly of a body of code within a conditional assembly block.

## 6.10.2 .INCLUDE Directive

.INCLUDE

Format:

```
.INCLUDE string
```

where: string represents a delimited string that is the file specification of a macro source file.

The .INCLUDE directive is used to insert a source file within the source file currently being used. When this directive is encountered, an implicit .PAGE directive is issued, the current source file is stacked, and the source file specified by the directive is read into memory. When the end of the specified source file is reached, an implicit .PAGE directive is issued, the original source file is popped from the stack, and assembly resumes at the line following the directive. A source file can also be inserted within a source file that has already been specified by the .INCLUDE directive. In this case the original source file and the first source file specified by the .INCLUDE directive are stacked and the second specified source file is read into memory. When the end of the second source file is reached, the first specified source file is popped from the stack and assembly resumes at the line following the directive, and when the end of the first specified source file is reached, the original source file is popped from the stack and assembly of that file is started again at the line following the .INCLUDE directive. An implicit .PAGE directive precedes and follows each included source file. The maximum nesting level of source files specified by the .INCLUDE directive is five.

If any information is omitted from the source file argument, default values are assumed. The default file specification for MACRO-11/RT-11 is DK:.MAC, and for other systems it is SY:.MAC.

The .INCLUDE directive is used as follows:

```
.INCLUDE        /DR3:[1,2]MACROS/    ;File MACROS.MAC
.INCLUDE        ?DK:SYSDEF?
.INCLUDE        \CURRENT.MAC\
```

NOTE

If you are using MACRO-11 with an RT-11 operating system, the device driver for the specified device that the .INCLUDE file resides on must already be loaded, either explicitly with the KMON LOAD command, or implicitly by reference to the device on the original MACRO-11 command line.

Macro definition arguments (dummy) and macro call arguments (real) normally maintain a strict positional relationship. That is, the first real argument in a macro call corresponds with the first dummy argument in a macro definition. Only the use of keyword arguments in a macro call can override this correspondence (see Section 7.3.6).

For example, the following macro definition and its associated macro call contain multiple arguments:

```
.MACRO  REN A,B,C
 .
 .
 .
REN     ALPHA,BETA,<C1,C2>
```

Arguments which themselves contain separating characters must be enclosed in paired angle brackets. For example, the macro call:

```
REN     <MOV    X,Y>,#44,WEV
```

causes the entire expression

```
MOV     X,Y
```

to replace all occurrences of the symbol A in the macro definition. Real arguments within a macro call are considered to be character strings and are treated as a single entity during the macro expansion.

The up-arrow (^) construction allows angle brackets to be passed as part of the argument. This construction, for example, could have been used in the above macro call, as follows:

```
REN     ^/<MOV X,Y>/,#44,WEV
```

causing the entire character string <MOV X,Y> to be passed as an argument.

Because of the use of the up-arrow (^) shown above, care must be taken when passing an argument beginning with a unary operator (^O, ^D, ^B, ^R, ^F ...). These arguments must be enclosed in angle brackets (as shown below) or MACRO-11 will read the character following the up-arrow as a delimiter.

```
REN <^O 411>,X,Y
```

The following macro call:

```
REN     #44,WEV^/MOV X,Y/
```

contains only two arguments (#44 and WEV^/MOV X,Y/), because the up-arrow is a unary operator (see Section 3.1.3) and it is not preceded by an argument separator.

As shown in the examples above, spaces can be used within bracketed argument constructions to increase the legibility of such expressions.

When 8-bit DEC Multinational character set (MCS) characters are used in argument strings, they must be enclosed in angle brackets (<>) or the argument delimiter (/) must be preceded by an up-arrow (^). The following are legal uses of the MCS characters in the argument string:

```
<This string can contain MCS characters>

^/This string can contain MCS characters/
```

### 7.3.1 Macro Nesting

Macro nesting occurs where the expansion of one macro includes a call to another. The depth of nesting allowed depends upon the amount of dynamic memory used by the source program being assembled.

To pass an argument containing legal argument delimiters to nested macros, enclose the argument in the macro definition within angle brackets, as shown in the coding sequence below. This extra set of angle brackets for each level of nesting is required in the macro definition, not in the macro call.

```
.MACRO  LEVEL1 DUM1,DUM2
LEVEL2  <DUM1>
LEVEL2  <DUM2>
.ENDM

.MACRO  LEVEL2 DUM3
DUM3
ADD     #10,40
MOV     R0,(R1)+
.ENDM
```

A call to the LEVEL1 macro, as shown below, for example:

```
LEVEL1  <MOV    X,R0>,<MOV    R2,R0>
```

causes the following macro expansion to occur:

```
MOV     X,R0
ADD     #10,R0
MOV     R0,(R1)+
MOV     R2,R0
ADD     #10,R0
MOV     R0,(R1)+
```

When macro definitions are nested, the inner definition cannot be called until the outer macro has been called and expanded. For example, in the following coding:

```
.MACRO  LV1 A,B
   .
   .
   .
.MACRO  LV2 C
   .
   .
   .
.ENDM
.ENDM
```

the LV2 macro cannot be called and expanded until the LV1 macro has been expanded. Likewise, any macro defined within the LV2 macro definition cannot be called and expanded until LV2 has also been expanded.

This automatic generation is invoked on each call of a macro whose definition contains a dummy argument preceded by the question mark (?) character, as shown in the macro definition below:

```
        .MACRO  ALPHA, A,?B     ;Contains dummy argument B preceded by
                                ;question mark.
        TST     A
        BEQ     B
        ADD     #5,A
B:
        .ENDM
```

A local symbol is created automatically by MACRO-11 only when a real argument of the macro call is either null or missing, as shown in Example 1 below. If the real argument is specified in the macro call, however, MACRO-11 inhibits the generation of a local symbol and normal argument replacement occurs, as shown in Example 2 below. (Examples 1 and 2 are both expansions of the Alpha macro defined above.)

EXAMPLE 1: Create a Local Symbol for the Missing Argument:

```
        ALPHA   R1              ;Second argument is missing.
        TST     R1
        BEQ     30000$          ;Local symbol is created.
        ADD     #5,R1
30000$:
```

EXAMPLE 2: Do Not Create a Local Symbol:

```
        ALPHA   R2,XYZ          ;Second argument XYZ is specified.
        TST     R2
        BEQ     XYZ             ;Normal argument replacement occurs.
        ADD     #5,R2
XYZ:
```

Automatically created local symbols are restricted to the first 16(10) arguments of a macro definition.

Automatically created local symbols resulting from the expansion of a macro, as described above, do not establish a local symbol block in their own right.

When a macro has several arguments earmarked for automatic local symbol generation, substituting a specific label for one such argument risks assembly errors because MACRO-11 constructs its argument substitution list at the point of macro invocation. Therefore, the appearance of a label, the .ENABL LSB directive, or the .PSECT directive, in the macro expansion will create a new local symbol block. The new local symbol block could leave local symbol references in the previous block and their symbol definitions in the new one, causing error codes in the assembly listing. Furthermore, a later macro expansion that creates local symbols in the new block may duplicate one of the symbols in question, causing an additional error code (P) in the assembly listing.

### 7.3.6 Keyword Arguments

Format:

```
name=string
```

where: name represents the dummy argument,

string represents the real symbolic argument.

The keyword argument may not contain embedded argument separators unless delimited as described in Section 7.3.

Macros may be defined with, and/or called with, keyword arguments. When a keyword argument appears in the dummy argument list of a macro definition, the specified string becomes the default real argument at macro call. When a keyword argument appears in the real argument list of a macro call, however, the specified string becomes the real argument for the dummy argument that matches the specified name, whether or not the dummy argument was defined with a keyword. If a match fails, the entire argument specification is treated as the next positional real argument.

The DEC Multinational character set can be used in keyword arguments if enclosed in angle brackets (<>).

A keyword argument may be specified anywhere in the dummy argument list of a macro definition and is part of the positional ordering of argument. A keyword argument may also be specified anywhere in the real argument list of a macro call but, in this case, does not affect the positional ordering of the arguments.

```
 1                              .LIST   ME
 2                      ;
 3                      ; Define a macro having keywords in dummy argument
 4                      ; list
 5                      ;
 6                              .MACRO  TEST CONTRL=1,BLOCK,ADDRES=TEMP
 7                              .WORD   CONTRL
 8                              .WORD   BLOCK
 9                              .WORD   ADDRES
10                              .ENDM
11
12
13                      ;
14                      ; Now invoke several times
15                      ;
16
17 000000                      TEST    A,B,C
   000000  000000G             .WORD   A
   000002  000000G             .WORD   B
   000004  000000G             .WORD   C
18
19 000006                      TEST    ADDRES=20,BLOCK=30,CONTRL=40
   000006  000040              .WORD   40
   000010  000030              .WORD   30
   000012  000020              .WORD   20
20
21 000014                      TEST    BLOCK=5
   000014  000001              .WORD   1
   000016  000005              .WORD   5
   000020  000000G             .WORD   TEMP
```

```
 1                              .TITLE  NARG
 2
 3                              .ENABL  LC
 4                              .LIST   ME
 5                      ;+
 6                      ; Example of the .NARG directive
 7                      ;
 8
 9                      .MACRO  NULL    NUM
10                              .NARG   SYM
11                              .IF EQ  SYM
12                              .MEXIT
13                              .IFF
14                              .REPT   NUM
15                              NOP
16                              .ENDM
17                              .ENDC
18                      .ENDM
19
20 000000                       NULL
           000000               .NARG   SYM
                                .IF EQ  SYM
                                .MEXIT
                                .IFF
                                .REPT
                                NOP
                                .ENDM
                                .ENDC
21
22 000000                       NULL    6
           000001               .NARG   SYM
                                .IF EQ  SYM
                                .MEXIT
                                .IFF
           000006               .REPT   6
                                NOP
                                .ENDM
   000000  000240               NOP
   000002  000240               NOP
   000004  000240               NOP
   000006  000240               NOP
   000010  000240               NOP
   000012  000240               NOP
                                .ENDC
23
24         000001               .END
```

Figure 7-1 Example of .NARG Directive

## 7.4.2 .NCHR Directive

.NCHR

Format:

```
[label:]    .NCHR symbol,<string>
```

where: label represents an optional statement label.

symbol represents any legal symbol. This symbol is equated to the number of characters in the specified character string. If a symbol is not specified, the .NCHR directive is flagged with an error code (A) in the assembly listing.

represents any legal separator (comma, space, and/or tab).

<string> represents a string of printable 7-bit ASCII or 8-bit DEC Multinational characters. If the character string contains a legal separator (comma, space, and/or tab) the whole string must be enclosed within angle brackets (<>) or be delimited using the up-arrow (^) construction, explained in Section 7.3. If the delimiting characters do not match or if the ending delimiter cannot be detected because of a syntactical error in the character string (thus prematurely terminating its evaluation), the .NCHR directive is flagged with an error code (A) in the assembly listing.

The .NCHR directive, which can appear anywhere in a MACRO-11 program, is used to determine the number of characters in a specified character string. This directive is useful in calculating the length of macro arguments.

An example of the .NCHR directive is shown in Figure 7-2.

```
 1                              .TITLE  NCHR
 2
 3                              .ENABL  LC
 4                              .LIST   ME
 5                      ;+
 6                      ; Illustrate the .NCHR directive
 7                      ;-
 8
 9                      .MACRO  STRING  MESSAG
10                              .NCHR   $$$,MESSAG
11                              .WORD   $$$
12                              .ASCII  /MESSAG/
13                              .EVEN
14                      .ENDM
15
16 000000                      MSG1:   STRING  <Hello>
           000005                       .NCHR   $$$,Hello
   000000  000005                       .WORD   $$$
   000002     110                       .ASCII  /Hello/
   000003     145
   000004     154
   000005     154
   000006     157
                                       .EVEN
17
18         000001                      .END
```

Figure 7-2 Example of .NCHR Directive

### 7.4.3 .NTYPE Directive

.NTYPE

Format:

```
[label:]    .NTYPE symbol,aexp
```

where: label represents an optional statement label.

symbol represents any legal symbol. This symbol is equated to the 6-bit addressing mode of the following expression (aexp). If a symbol is not specified, the .NTYPE directive is flagged with an error code (A) in the assembly listing.

| | |
|---|---|
| , | represents any legal separator (comma, space, and/or tab). |
| aexp | represents any legal address expression, as used with an opcode. If no argument is specified, an error code (A) will appear in the assembly listing. |

The .NTYPE directive is used to determine the addressing mode of a specified macro argument. Hence, the .NTYPE directive can appear only within a macro definition; if it appears elsewhere, it is flagged with an error code (O) in the assembly listing.

An example of the use of an .NTYPE directive in a macro definition is shown in Figure 7-3.

```
 1                              .TITLE  NTYPE
 2
 3                              .ENABL  LC
 4                              .LIST   ME
 5
 6                      ;+
 7                      ; Illustrate the .NTYPE directive
 8                      ;-
 9
10                      .MACRO  SAVE    ARG
11                              .NTYPE  $$$,ARG
12                      .IF EQ  $$$&70
13                              MOV     ARG,-(SP)       ;Save in register mode
14                              .IFF
15                              MOV     #ARG,-(SP)      ;Save in non-register mode
16                      .ENDC
17                      .ENDM
18
19 000000                      SAVE    R1
           000001               .NTYPE  $$$,R1
                        .IF EQ  $$$&70
   000000  010146               MOV     R1,-(SP)        ;Save in register mode
                                .IFF
                                MOV     #R1,-(SP)       ;Save in non-register mode
                        .ENDC
20
21 000002                      SAVE    TEMP
           000067               .NTYPE  $$$,TEMP
                        .IF EQ  $$$&70
                                MOV     TEMP,-(SP)      ;Save in register mode
                                .IFF
   000002  012746               MOV     #TEMP,-(SP)     ;Save in non-register mode
           000006'
                        .ENDC
22
23 000006  000000       TEMP:  .WORD    0
24
25         000001               .END
```

Figure 7-3 Example of .NTYPE Directive in Macro Definition

For additional information concerning addressing modes, refer to Chapter 5 and Appendix B.2.

## 7.5 .ERROR AND .PRINT DIRECTIVES

**.ERROR**

Format:

```
[label:]   .ERROR [expr]    ;text
```

where:

| | |
|---|---|
| label | represents an optional statement label. |
| expr | represents an optional expression whose value is output when the .ERROR directive is encountered during assembly. |
| ; | denotes the beginning of the text string. |
| text | represents the message associated with the .ERROR directive. The text can be 7-bit ASCII or 8-bit DEC Multinational characters. |

The .ERROR directive is used to output messages to the listing file during assembly pass 2. A common use of this directive is to alert the user to a rejected or erroneous macro call or to the existence of an illegal set of conditions in a conditional assembly. If the listing file is not specified, the .ERROR messages are output to the command output device.

Upon encountering an .ERROR directive anywhere in a source program, MACRO-11 outputs a single line containing:

1. An error code (P)
2. The sequence number of the .ERROR directive statement
3. The value of the current location counter
4. The value of the expression, if one is specified
5. The source line containing the .ERROR directive.

For example, the following directive:

```
.ERROR  A    ;Invalid macro argument
```

causes a line in the following form to be output to the listing file:

```
    Seq. Loc.    Exp.
    No.  No.     Value                                  Text

P   512 005642  000076        .ERROR  A      ;Invalid macro argument
```

**.PRINT**

The .PRINT directive is identical in function to the .ERROR directive, except that it is not flagged with the error code (P).

## 7.6 INDEFINITE REPEAT BLOCK DIRECTIVES: .IRP AND .IRPC

An indefinite repeat block is similar to a macro definition with only one dummy argument. At each expansion of the indefinite repeat range, this dummy argument is replaced with successive elements of a real argument list. Since the repeat directive and its associated range are coded in-line within the source program, this type of macro definition and expansion does not require calling the macro by name, as required in the expansion of the conventional macros previously described in this chapter.

An indefinite repeat block can appear either within or outside another macro definition, indefinite repeat block, or repeat block. The rules for specifying indefinite repeat block arguments are the same as for specifying macro arguments (see Section 7.3).

### 7.6.1 .IRP Directive

**.IRP**

Format:

```
[label:]     .IRP sym,<argument list>
               .
               .
               .
          (range of indefinite repeat block)
               .
               .
               .
          .ENDM
```

where: label represents an optional statement label.

NOTE

Although it is legal for a label to appear on an .IRP directive, this practice is discouraged, especially in the case of nested macro definitions, because invalid labels or labels constructed with the concatenation character will cause the macro directive to be ignored. This may result in improper termination of the macro definition.

This NOTE also applies to .IRPC and .REPT.

| | |
|---|---|
| sym | represents a dummy argument that is replaced with successive real arguments from within the angle brackets. If no dummy argument is specified, the .IRP directive is flagged with an error code (A) in the assembly listing. |
| , | represents any legal separator (comma, space, and/or tab). |
| <argument list> | represents a list of real arguments enclosed within angle brackets that is to be used in the expansion of the indefinite repeat range. A real argument may consist of one or more 7-bit ASCII or 8-bit DEC Multinational characters; multiple arguments must be separated by any legal separator (comma, space, and/or tab). If no real arguments are specified, no action is taken. |
| range | represents the block of code to be repeated once for each occurrence of a real argument in the list. The range may contain other macro definitions, repeat ranges and/or the .MEXIT directive (see Section 7.1.3). |
| .ENDM | indicates the end of the indefinite repeat block range. |

The .IRP directive is used to replace a dummy argument with successive real arguments specified in an argument string. This replacement process occurs during the expansion of an indefinite repeat block range.

An example of the use of the .IRP directive is shown in Figure 7-4.

### 7.6.2 .IRPC Directive

.IRPC

Format:

```
[label:]  .IRPC   sym,<string>
            .
            .
            .
          (range of indefinite repeat block)
            .
            .
            .
          .ENDM
```

| | | |
|---|---|---|
| where: | label | represents an optional statement label (see Note in Section 7.6.1). |
| | sym | represents a dummy argument that is replaced with successive real arguments from within the angle brackets. If no dummy argument is specified, the .IRPC directive is flagged with an error code (A) in the assembly listing. |

| | |
|---|---|
| , | represents any legal separator (comma, space, and/or tab). |
| <string> | represents a list of 7-bit ASCII or 8-bit DEC Multinational characters, enclosed within angle brackets, to be used in the expansion of the indefinite repeat range. Although the angle brackets are required only when the string contains separating characters, their use is recommended for legibility. |
| range | represents the block of code to be repeated once for each occurrence of a character in the list. The range may contain macro definitions, repeat ranges and/or the .MEXIT directive (see Section 7.1.3). |
| .ENDM | indicates the end of the indefinite repeat block range. |

The .IRPC directive is available to permit single character substitution, rather than argument substitution. On each iteration of the indefinite repeat range, the dummy argument is replaced with successive characters in the specified string.

An example of the use of the .IRPC directive is shown in Figure 7-4.

```
 1                                      .TITLE  IRPTST
 2
 3                                      .LIST   ME
 4                              ;+
 5                              ; Illustrate the .IRP and .IRPC directives
 6                              ; by creating a pair of RAD50 tables
 7                              ;-
 8
 9 000000                       REGS:   .IRP    REG,<PC,SP,R5,R4,R3,R2,R1,R0>
10                                      .RAD50  /REG/
11                                      .ENDR
   000000  062170                       .RAD50  /PC/
   000002  074500                       .RAD50  /SP/
   000004  072770                       .RAD50  /R5/
   000006  072720                       .RAD50  /R4/
   000010  072650                       .RAD50  /R3/
   000012  072600                       .RAD50  /R2/
   000014  072530                       .RAD50  /R1/
   000016  072460                       .RAD50  /R0/
12
13 000020                       REGS2:  .IRPC   NUM,<76543210>
14                                      .RAD50  /R'NUM/
15                                      .ENDR
   000020  073110                       .RAD50  /R7/
   000022  073040                       .RAD50  /R6/
   000024  072770                       .RAD50  /R5/
   000026  072720                       .RAD50  /R4/
   000030  072650                       .RAD50  /R3/
   000032  072600                       .RAD50  /R2/
   000034  072530                       .RAD50  /R1/
   000036  072460                       .RAD50  /R0/
16
17         000001                      .END
```

Figure 7-4 Example of .IRP and .IRPC Directives

.REPT
.ENDR

## 7.7 REPEAT BLOCK DIRECTIVE: .REPT, .ENDR

Format:

```
[label:]    .REPT        exp
              .
              .
              .
          (range of repeat block)
              .
              .
              .
          .ENDR
```

where:

| | |
|---|---|
| label | represents an optional statement label (see Note in Section 7.6.1). |
| exp | represents any legal expression. This value controls the number of times the block of code is to be assembled within the program. When the expression value is less than or equal to zero (Ø), the repeat block is not assembled. If this expression is not an absolute value, the .REPT statement is flagged with an error code (A) in the assembly listing. |
| range | represents the block of code to be repeated. The repeat block may contain macro definitions, indefinite repeat blocks, other repeat blocks and/or the .MEXIT directive (see Section 7.1.3). |
| .ENDM or .ENDR | indicates the end of the repeat block range. |

The .REPT directive is used to duplicate a block of code, a certain number of times, in line with other source code.

.MCALL

## 7.8 MACRO LIBRARY DIRECTIVE: .MCALL

Format:

```
.MCALL  arg1,arg2,...argn
```

where:

| | |
|---|---|
| arg1, arg2,... argn | represent the symbolic names of the macro definitions required in the assembly of the source program. The names must be separated by any legal separator (comma, space, and/or tab). |

Table 8-2 (Cont.)
MACRO-11 File Specification Switches

| Switch | Function |
|---|---|
| /ML (Cont.) | beginning with the last user macro file specified, continuing in reverse order with each such file specified, and terminating, if necessary, with a search of the system macro library file. If a required macro definition is not found upon completion of the search, an error code (U) results in the assembly listing. This means that a user macro library file must be specified in the command line or by using the MACRO-11 .LIBRARY directive (see Section 6.10.1) prior to the source file(s) that use macros defined in the library file.<br><br>MACRO-11 does not pre-scan the command line for macro libraries; when a new source file is needed, it parses the next input file specification. If that file specification contains the /ML switch, it is appended to the front of the library file list. As a result, a user macro library file must be specified in the command line prior to the source files which require it, in order to resolve macro definitions. |
| /SP | Spool listing output (default value). |
| /NOSP | Do not spool output. |
| /CR:[arg] | Produce a cross-reference listing (see Section 8.3). |

Switches for the object file are limited to /EN and /DS; when specified, they apply throughout the entire command string. Switch options for the listing file are limited to /LI, /NL, /SP, /CR, and /NOSP. Switches for input files are limited to /ML, /EN, and /DS; the option /ML applies only to the file immediately preceding the option so specified, whereas the /EN and /DS options, as noted above, are also applicable to subsequent files in the command string.

Multiple occurrences of the same switch following a file specification must be avoided, because the accompanying values of a subsequent like switch specification override any previously-specified values. If two such switch values are desired, they can be specified in the form shown below:

```
/LI:SRC:MEB
```

### 8.1.4 DCL Operating Procedures

RSX-11M/RSX-11M-PLUS indicates its readiness to accept a command by prompting with the DCL prompt. In response to the prompt, enter the command string in one of the formats shown below:

```
>MACRO[/qualifiers]
FILE? filespec[/qualifier[s]][,filespec[/qualifier[s]]...]
```

or

```
[DCL]>MACRO[/qualifiers] filespec[/qualifier[s]][,filespec[/qualifier[s]]...]
```

where: qualifiers — affect either the entire command string (command qualifiers) or the filespec (parameter qualifiers). See Table 8-3 for a description of the command qualifiers and Table 8-4 for a description of the parameter qualifiers.

filespec — is the standard file specification shown in Section 8.4.

You use the comma (,) to separate file specifications. MACRO-11 concatenates all the files and then performs the assembly.

Table 8-3
DCL Command Qualifiers

| Qualifier | Function |
|---|---|
| /[NO]CROSS_REFERENCE | Suppresses or generates a cross-reference listing (see Section 8.3). When the cross-reference is generated, a listing file is also generated, whether or not the /LIST qualifier is present in the command string.<br><br>/NOCROSS_REFERENCE is the default. |
| /DISABLE:arg<br>/ENABLE:arg<br>/DISABLE:(arg,arg...)<br>/ENABLE:(arg,arg...) | Overrides the .DISABLE or .ENABLE assembler directives in the source program. When more than one argument is entered, arguments must be enclosed in parentheses and separated by commas.<br><br>You can specify any of the following arguments with the /DISABLE or /ENABLE qualifier.<br><br>Argument<br><br>ABSOLUTE — Enabling this function causes all relative addresses (address mode 67) to be assembled as absolute addresses (address mode 37). |

(continued on next page)

srcl, src2,... srcn — represent the ASCII source (input) files containing the MACRO-11 source program or the user-supplied macro library files to be assembled. You can specify as many as six source files.

The following command string calls for an assembly that uses one source file plus the system MACRO library to produce an object file BINF.OBJ and a listing. The listing goes directly to the line printer.

```
*DK:BINF.OBJ,LP:=DK:SRC.MAC
```

All output file specifications are optional. The system does not produce an output file unless the command string contains a specification for that file.

The system determines the file type of an output file specification by its position in the command string, as determined by the number of commas in the string. For example, to omit the object file, you must begin the command string with a comma. The following command produces a listing, including cross-reference tables, but not binary object files.

```
*,LP:/C=(source file specification)
```

Notice that you need not include a comma after the final output file specification in the command string.

Table 9-1 lists the default values for each file specification.

Table 9-1
Default File Specification Values

| File | Default Device | Default File Name | Default File Type |
|---|---|---|---|
| Object | DK: | Must specify | .OBJ |
| Listing | Same as for object file | Must specify | .LST |
| Cref | DK: | Must specify | .TMP |
| First source | DK: | Must specify | .MAC |
| Additional source | Same as for preceding source file | Must specify | .MAC |
| System MACRO Library | System device SY: | SYSMAC | .SML |
| User MACRO Library | DK: if first file, otherwise same as for preceding source file | Must specify | .MLB |

NOTE

Some assemblies need more symbol table space than available memory can contain. When this occurs the system automatically creates a temporary work file called WRK.TMP to provide extended symbol table space.

The default device for WRK.TMP is DK. To cause the system to assign a different device, enter the following command:

```
.ASSIGN dev: WF
```

where: dev is the file-structured device that will hold WRK.TMP.

## 9.4 FILE SPECIFICATION OPTIONS

At assembly time you may need to override certain MACRO directives appearing in the source programs. You may also need to direct MACRO-11 on the handling of certain files during assembly. You can satisfy these needs by using the switches described in Table 9-2.

Table 9-2
File Specification Options

| Option | Usage |
|---|---|
| /L:arg<br>/N:arg | Listing control switches; these options accept ASCII switch values (arg) which are equivalent in function and name to the arguments of the .LIST and .NLIST directives specified in the source program (see Section 6.1.1). This switch overrides the arguments of the directives and remains in effect for the entire assembly process. |
| /E:arg<br>/D:arg | Function control switches; these options accept ASCII switch values (arg) which are equivalent in function and name to the arguments of the .ENABL and .DSABL directives specified in the source program (see Section 6.2.1). This switch overrides the arguments of the directives and remains in effect for the entire assembly process. |

(continued on next page)

APPENDIX A

# MACRO-11 CHARACTER SETS

## A.1 ASCII CHARACTER SET

| Even Parity Bit | 7-Bit Octal Code | Character | Remarks |
|---|---|---|---|
| Ø | ØØØ | NUL | Null, tape feed, CONTROL/SHIFT/P. |
| 1 | ØØ1 | SOH | Start of heading; also SOM, start of message, CONTROL/A. |
| 1 | ØØ2 | STX | Start of text; also EOA, end of address, CONTROL/B. |
| Ø | ØØ3 | ETX | End of text; also EOM, end of message, CONTROL/C. |
| 1 | ØØ4 | EOT | End of transmission (END); shuts off TWX machines, CONTROL/D. |
| Ø | ØØ5 | ENQ | Enquiry (ENQRY); also WRU, CONTROL/E. |
| Ø | ØØ6 | ACK | Acknowledge; also RU, CONTROL/F. |
| 1 | ØØ7 | BEL | Rings the bell. CONTROL/G. |
| 1 | Ø1Ø | BS | Backspace; also FEO, format effector. backspaces some machines, CONTROL/H. |
| Ø | Ø11 | HT | Horizontal tab. CONTROL/I. |
| Ø | Ø12 | LF | Line feed or Line space (new line); advances paper to next line, duplicated by CONTROL/J. |
| 1 | Ø13 | VT | Vertical tab (VTAB). CONTROL/K. |
| Ø | Ø14 | FF | Form Feed to top of next page (PAGE). CONTROL/L. |
| 1 | Ø15 | CR | Carriage return to beginning of line; duplicated by CONTROL/M. |
| 1 | Ø16 | SO | Shift out; changes ribbon color to red. CONTROL/N. |
| Ø | Ø17 | SI | Shift in; changes ribbon color to black. CONTROL/O. |
| 1 | Ø2Ø | DLE | Data link escape. CONTROL/P (DCØ). |
| Ø | Ø21 | DC1 | Device control 1; turns transmitter (READER) on, CONTROL/Q (X ON). |
| Ø | Ø22 | DC2 | Device control 2; turns punch or auxiliary on. CONTROL/R (TAPE, AUX ON). |
| 1 | Ø23 | DC3 | Device control 3; turns transmitter (READER) off, CONTROL/S (X OFF). |
| Ø | Ø24 | DC4 | Device control 4; turns punch or auxiliary off. CONTROL/T (AUX OFF). |

| Even Parity Bit | 7-Bit Octal Code | Character | Remarks |
|---|---|---|---|
| 1 | 025 | NAK | Negative acknowledge; also ERR, ERROR. CONTROL/U. |
| 1 | 026 | SYN | Synchronous file (SYNC). CONTROL/V. |
| Ø | 027 | ETB | End of transmission block; also LEM, logical end of medium. CONTROL/W. |
| Ø | 030 | CAN | Cancel (CANCL). CONTROL/X. |
| 1 | 031 | EM | End of medium. CONTROL/Y. |
| 1 | 032 | SUB | Substitute. CONTROL/Z. |
| Ø | 033 | ESC | Escape. CONTROL/SHIFT/K. |
| 1 | 034 | FS | File separator. CONTROL/SHIFT/L. |
| Ø | 035 | GS | Group separator. CONTROL/SHIFT/M. |
| Ø | 036 | RS | Record separator. CONTROL/SHIFT/N. |
| 1 | 037 | US | Unit separator. CONTROL/SHIFT/O. |
| 1 | 040 | SP | Space. |
| Ø | 041 | ! | |
| Ø | 042 | " | |
| 1 | 043 | # | |
| Ø | 044 | $ | |
| 1 | 045 | % | |
| 1 | 046 | & | |
| Ø | 047 | ' | Accent acute or apostrophe. |
| Ø | 050 | ( | |
| 1 | 051 | ) | |
| 1 | 052 | * | |
| Ø | 053 | + | |
| 1 | 054 | , | |
| Ø | 055 | - | |
| Ø | 056 | . | |
| 1 | 057 | / | |
| Ø | 060 | Ø | |
| 1 | 061 | 1 | |
| 1 | 062 | 2 | |
| Ø | 063 | 3 | |
| 1 | 064 | 4 | |
| Ø | 065 | 5 | |
| Ø | 066 | 6 | |
| 1 | 067 | 7 | |
| 1 | 070 | 8 | |
| Ø | 071 | 9 | |
| Ø | 072 | : | |
| 1 | 073 | ; | |
| Ø | 074 | < | |
| 1 | 075 | = | |
| 1 | 076 | > | |
| Ø | 077 | ? | |
| 1 | 100 | @ | |
| Ø | 101 | A | |
| Ø | 102 | B | |
| 1 | 103 | C | |
| Ø | 104 | D | |
| 1 | 105 | E | |
| 1 | 106 | F | |
| Ø | 107 | G | |
| Ø | 110 | H | |
| Ø | 111 | I | |

| Single Char. or First Char. | | Second Character | | Third Character | |
|---|---|---|---|---|---|
| V | 104600 | V | 001560 | V | 000026 |
| W | 107700 | W | 001630 | W | 000027 |
| X | 113000 | X | 001700 | X | 000030 |
| Y | 116100 | Y | 001750 | Y | 000031 |
| Z | 121200 | Z | 002020 | Z | 000032 |
| $ | 124300 | $ | 002070 | $ | 000033 |
| . | 127400 | . | 002140 | . | 000034 |
| Unused | 132500 | Unused | 002210 | Unused | 000035 |
| 0 | 135600 | 0 | 002260 | 0 | 000036 |
| 1 | 140700 | 1 | 002330 | 1 | 000037 |
| 2 | 144000 | 2 | 002400 | 2 | 000040 |
| 3 | 147100 | 3 | 002450 | 3 | 000041 |
| 4 | 152200 | 4 | 002520 | 4 | 000042 |
| 5 | 155300 | 5 | 002570 | 5 | 000043 |
| 6 | 160400 | 6 | 002640 | 6 | 000044 |
| 7 | 163500 | 7 | 002710 | 7 | 000045 |
| 8 | 166600 | 8 | 002760 | 8 | 000046 |
| 9 | 171700 | 9 | 003030 | 9 | 000047 |

## A.3 DEC MULTINATIONAL CHARACTER SET

| $b_4$ | $b_3$ | $b_2$ | $b_1$ | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | | | | | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| | | | | | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $b_4$ | $b_3$ | $b_2$ | $b_1$ | | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | 0 | 0 | 0 | 0 | NUL | DLE | SP | 0 | @ | P | ` | p | | DCS | | ° | À | | à | |
| 0 | 0 | 0 | 1 | 1 | SOH | DC1 | ! | 1 | A | Q | a | q | | PU1 | ¡ | ± | Á | Ñ | á | ñ |
| 0 | 0 | 1 | 0 | 2 | STX | DC2 | " | 2 | B | R | b | r | | PU2 | ¢ | ² | Â | Ò | â | ò |
| 0 | 0 | 1 | 1 | 3 | ETX | DC3 | # | 3 | C | S | c | s | | STS | £ | ³ | Ã | Ó | ã | ó |
| 0 | 1 | 0 | 0 | 4 | EOT | DC4 | $ | 4 | D | T | d | t | IND | CCH | | | Ä | Ô | ä | ô |
| 0 | 1 | 0 | 1 | 5 | ENQ | NAK | % | 5 | E | U | e | u | NEL | MW | ¥ | µ | Å | Õ | å | õ |
| 0 | 1 | 1 | 0 | 6 | ACK | SYN | & | 6 | F | V | f | v | SSA | SPA | | ¶ | Æ | Ö | æ | ö |
| 0 | 1 | 1 | 1 | 7 | BEL | ETB | ' | 7 | G | W | g | w | ESA | EPA | § | · | Ç | Œ | ç | œ |
| 1 | 0 | 0 | 0 | 8 | BS | CAN | ( | 8 | H | X | h | x | HTS | | ¤ | | È | Ø | è | ø |
| 1 | 0 | 0 | 1 | 9 | HT | EM | ) | 9 | I | Y | i | y | HTJ | | © | ¹ | É | Ù | é | ù |
| 1 | 0 | 1 | 0 | 10 | LF | SUB | * | : | J | Z | j | z | VTS | | ª | º | Ê | Ú | ê | ú |
| 1 | 0 | 1 | 1 | 11 | VT | ESC | + | ; | K | [ | k | { | PLD | CSI | « | » | Ë | Û | ë | û |
| 1 | 1 | 0 | 0 | 12 | FF | FS | , | < | L | \ | l | \| | PLU | ST | | ¼ | Ì | Ü | ì | ü |
| 1 | 1 | 0 | 1 | 13 | CR | GS | – | = | M | ] | m | } | RI | OSC | | ½ | Í | Ÿ | í | ÿ |
| 1 | 1 | 1 | 0 | 14 | SO | RS | . | > | N | ^ | n | ~ | SS2 | PM | | | Î | | î | |
| 1 | 1 | 1 | 1 | 15 | SI | US | / | ? | O | — | o | DEL | SS3 | APC | | ¿ | Ï | ß | ï | |
| | | | | | ASCII Control Set | | ASCII Graphic Character Set | | | | | | Add'l Control Set | | DEC Supplemental Graphic Set | | | | | |

DEC Multinational Character Set

Empty positions are reserved for future standardizations

APPENDIX J

# RELEASE NOTES

This appendix explains the changes that have been made to MACRO-11 since the last version release. The new features mentioned are fully documented in chapters one through nine of this manual.

## J.1 CHANGES -- ALL VERSIONS OF MACRO-11

### J.1.1 V5.2 Update Changes

MACRO-11 now provides support for the 8-bit DEC Multinational character set (MCS). A chart showing the MCS is located in Appendix A.

The following directives support the MCS. For specific support information, consult the description of each directive.

| Macro | Section |
|---|---|
| .ASCII directive | 6.3.4 |
| .ASCIZ directive | 6.3.5 |
| .ERROR directive | 7.5 |
| .IF directive | 6.9.1 |
| .IF DIF | |
| .IF IDN | |
| .IFF directive | 6.9.2 |
| .IFF DIF | |
| .IFF IDN | |
| .IRP directive | 7.6.1 |
| .IRPC directive | 7.6.2 |
| .NCHR directive | 7.4.2 |
| .PRINT directive | 7.5 |
| .REM directive | 6.1.6 |
| .SBTTL directive | 6.1.3 |
| .TITLE directive | 6.1.2 |

Further information on the 8-bit DEC Multinational character set is located in sections:

2.2.4 Comment field
6.3.3 ASCII conversion characters
7.3 Arguments in macro definitions and macro calls
7.3.6 Keyword arguments

### J.1.2 V5.1 Changes

1. The opcode, CALLR addr (Call-Return), has been added to the permanent symbol table (PST). This opcode is equivalent to the JMP addr opcode. The CALLR addr opcode was added to complement the CALL addr opcode -- which is equivalent to the JSR PC,addr opcode.

2. The previous version of MACRO-11 used a range of 64$ to 127$ for automatic local symbol generation. MACRO-11 now uses a range of 30000$ to 65535$ when generating local symbols.

3. Most assembler generated listing text is now in upper/lowercase. This change was made to increase the readability of MACRO-11 code. Lines of code that include the .SBTTL or the .TITLE directive are not converted to uppercase.

4. Lines of code that include the .SBTTL directive are listed in the table of contents of an assembly listing, even if a .NLIST statement is in effect at the time the .SBTTL lines are encountered. You may specify the .NLIST directive with the TOC argument to prevent the table of contents from being printed.

5. The symbol table is printed at the end of an assembly, even if the .NLIST directive is in effect. You may specify the .NLIST directive with the SYM argument to prevent the symbol table from being printed.

6. All page headers include the day of the week.

7. The assembler statistics information that appears at the end of the assembly listing file has been updated to include the following additional information:

   - Total number of virtual work file reads
   - Total number of virtual work file writes
   - Maximum amount of virtual memory used (in words and pages)
   - Size of physical memory freespace (in words and pages)
   - Operating system and environment that the assembler is running under
   - Total elapsed assembly time
   - MACRO-11 command line

8. The PSECT synopsis that is printed in the listing file, after the symbol table, includes the psect attributes.

9. The maximum number of relocatable terms in a complex expression has been changed. The maximum size of an .OBJ record that MACRO-11 can produce was increased from 42. bytes to 128. bytes.

Do not compare .OBJ files that have been created by different versions of MACRO-11 when verifying whether your code generation is correct. Changes that have been made for this version of MACRO-11 (mentioned above) will invalidate a direct comparison of assembler .OBJ output. Verify code generation by linking or taskbuilding the .OBJ files involved and then comparing the .SAV or the .TSK image files.

NOTE

Because the .OBJ files produced by this new version of MACRO-11 are different, users of the PAT (object file patch utility) are warned that checksums must be recomputed on any object patches assembled with this new version of MACRO-11.

10. The default for the LC argument has been changed from .DSABL LC to .ENABL LC.

11. The following .ENABL/.DSABL options have been added:

    1. .ENABL LCM/.DSABL LCM
    2. .ENABL MCL/.DSABL MCL

12. The following directives have been added to MACRO-11. These new directives are documented in this manual.

    1. .CROSS
    2. .INCLUDE
    3. .LIBRARY
    4. .MDELETE
    5. .NOCROSS
    6. .REM
    7. .WEAK

# INDEX


A error, 3-10, 3-13, 5-10, 6-15, 6-25, 6-26, 6-28, 6-29, 6-32, 6-33, 6-38, 6-40, 6-42, 6-44, 6-47, 6-56, 7-2, 7-12 to 7-14, 7-16, 7-17, 7-20
Absolute address, D-2
Absolute binary output, 6-19
Absolute expression, 3-17
Absolute mode, 5-1, 5-7, B-2, G-2, G-4
Absolute module, 6-42
Absolute program section, 6-42 to 6-45, B-4. See also .ASECT directive
ADD instruction, E-12, G-3, H-2
Addition operator, 3-2, 3-5, B-1
Address boundaries, 6-39
Addressing modes, 5-1
Apostrophe, G-4
ASCII
  character set, A-1
  conversion characters, 6-23 to 6-26
.ASCII directive, 6-1, 6-21, 6-26 to 6-28, 6-36, B-3
.ASCIZ directive, 6-1, 6-28, 6-36, B-4
.ASECT directive, 3-11, 3-13, 3-14, 6-2, 6-44 to 6-47, B-4
Assembler directives. See Permanent symbol table
  version number, 6-4
Assembly
  error. See A error
  listing symbols, 4-1
  pass 1, 1-1, 1-2, 6-12, 6-15, 6-16, 6-49, 8-10, 8-12, D-3
  pass 2, 1-2, 6-12, 6-21, 7-15, D-3
Assignment operator. See Direct assignment operator
Assignment statement. See Direct assignment statement
Autodecrement deferred mode, 5-1, 5-5, B-2, G-1
Autodecrement indicator, 3-2
Autodecrement mode, 5-1, 5-4, B-1, B-2, G-1
Autoincrement deferred mode, 5-1, 5-4, B-2, G-1
Autoincrement indicator, 3-2
Autoincrement mode, 5-1, 5-3, B-2, G-1

Base level, E-14
BCC instruction, E-13
BCS instruction, E-14
BEQ instruction, H-2
BGE instruction, E-13
BGT instruction, E-14
BHI instruction, E-14
BHIS instruction, E-13
BIC instruction, E-13
Binary operator, 3-4, 3-5, 3-16
Blank line, 2-1
BLE instruction, E-14
.BLKB directive, 3-14, 6-2, 6-36 to 6-38, B-4, D-3
.BLKW directive, 3-14, 6-2, 6-36, 6-38, 6-48, B-4, D-3
BLO instruction, E-13
BLOS instruction, E-13
BLT instruction, E-13, H-2
BNE instruction, E-10, G-3, H-2
BR instruction, E-10, E-11, G-3
Branch instruction
  addressing, 5-9, D-2
  use of, E-13
.BYTE directive, 6-2, 6-23, 6-36, B-4, D-4

C bit, E-9
CALL instruction, H-2
Calling convention, E-8
Character set
  ASCII, A-1 to A-3
  DEC Multinational, A-6
  legal, 3-1 to 3-3
  Radix-50, A-5, A-6
CLR instruction, G-3, G-3, H-2
CMP instruction, E-13, H-2
Coding standard, E-1
Comment, E-1, E-5
  delimiter, 3-2, B-1, E-12
  field, 2-1, 2-4, 2-5, E-1
Commercial instruction set, C-3
Common exit, E-11
Complex relocatable expression, 3-18

Complex relocation, 4-1, G-4
Concatenation indicator, 3-3, B-1, B-3
Conditional assembly, 6-51 to 6-56, 7-8, 7-16, D-4
  immediate, 6-56
Conditional assembly block, 7-3, B-4, B-5
Conditional assembly directive, 6-49
Copyright statement, E-5
.CROSS directive, 6-2, 6-22, B-4, C-5
Cross-reference listing, 3-12, 6-19, 8-8, 8-9, 8-14, 8-16 to 8-18, 9-2, 9-3, 9-5 to 9-7
.CSECT directive, 3-11, 3-13, 6-2, 6-44 to 6-47, 9-6, B-4
Current location counter, 2-2, 3-2, 3-12 to 3-14, 3-17, 5-8, 6-11, 6-36 to 6-38, 6-43 to 6-44, B-5, B-7, D-2, D-3

D error, 2-3
Data
  sharing, 6-45
  storage, 6-2
  storage directives, 6-23
DEC Multinational character set, A-6
  Argument strings, 7-5, 7-10
  ASCII conversion characters, 6-25 to 6-27
  Comment field, 2-4
  Directive support for
    .ASCII, 6-26
    .ASCIZ, 6-28
    .ERROR, 7-16
    .IF, 6-55
    .IFF. See .IF directive
    .IRP, 7-18
    .IRPC, 7-19
    .NCHR, 7-13
    .PRINT. See .ERROR directive
    .REM, 6-18
    .SBTTL, 4-15
    .TITLE, 6-15
Default radix, 3-14
Default register definitions, 3-10, 6-21
Deferred addressing indicator, 3-2, B-1
Delimiting characters, 3-3, 6-17, 6-29, B-3 to B-5, B-8
Device register, E-2
Direct assignment
  operator, 3-1, 3-2, 3-9, B-1
  statement, 3-6 to 3-9, 3-13, 6-37
Directives. See Permanent symbol table
DIV instruction, H-2
Division operator, 3-2, 3-5, B-1
Double ASCII character indicator, 3-2, B-1
.DSABL directive, 6-2, 6-19 to 6-21, 8-6, 8-8, 9-4, B-4, D-1
Dummy argument, 7-2, 7-11, 7-17

E error, 6-40
EMT instruction, 5-9, D-4
.ENABL directive, 6-2, 6-19 to 6-21, 8-6, 8-8, 9-4, B-4, D-1, D-2, D-4, F-2
.END directive, 6-2, 6-40, B-4, D-3, H-2
.ENDC directive, 6-2, 6-12, 6-53 to 6-56, 6-59, 7-3, B-4
.ENDM directive, 6-13, 6-21, 7-2, 7-3, 7-6 to 7-8, 7-10, 7-11, 7-17 to 7-19, B-4, B-8, F-3
.ENDR directive, 7-19, 7-20, B-5, B-8
Entry point symbol, 6-52
.ERROR directive, 7-16, B-5, D-4
Error messages, D-1 to D-5
.EVEN directive, 6-2, 6-29, 6-38, B-5
Expression, evaluation of, 3-16
Expression indicator, immediate, 3-2, B-1
External expression, 3-17
External symbol, 6-52. See also Global symbol

Field terminator, 3-2, B-1
FILES-11, 6-19
Floating-point directives, B-5. See also .FLT2 directive
Floating-point indicator, B-3
Floating-point processor, 3-14, 6-34, 6-35, C-4
Floating-point rounding, 6-19, 6-32
Floating-point truncation, 6-19, 6-35
.FLT2 directive, 6-2, 6-35, B-5
.FLT4 directive, 6-2, 6-35, B-5
FLX, 6-19
Forbidden instructions, E-13
Format control, 2-5
Formatted binary, 6-19
FORTRAN, 6-47, E-15, G-2
Forward reference, 3-8, 3-9, 3-10, 3-13, D-4
  illegal, D-3
Function control switches. See Switches, function control
Function directive, 6-18

Global expression evaluation, 3-17
Global label, 6-51
Global reference, 6-21, 6-51, F-4, G-4
Global symbol, 1-2, 3-7, B-5, D-2, D-3, E-4
Global symbol definition, 2-2, 3-1, 3-2, 3-8, 6-51. See also .GLOBL directive
Global symbol directory, 1-2
.GLOBL directive, 3-7, 6-2, 6-51, B-5, E-4

Hardware register, E-2

I error, 6-28, 6-30
IAS, 6-48, 7-21, 8-14 to 8-17, 8-19 to 8-22, G-1
.IDENT directive, 6-2, 6-16, B-5, D-2, E-5, E-7, E-15, H-1
.IF directive, 6-2, 6-12, 6-53 to 6-59, 7-3, 7-8, B-5, D-1, D-2
.IFF directive, 6-2, 6-56 to 6-58, B-5
.IFT directive, 6-2, 6-56 to 6-58, B-5
.IFTF directive, 6-2, 6-56, 6-57, B-6
.IIF directive, 6-2, 6-59, B-6, D-1, D-2
Illegal characters, 3-3, D-2, D-3
Illegal forward reference, D-3
Immediate conditional assembly, 6-59
Immediate expression indicator, 3-2, B-1
Immediate mode, 5-1, 5-6, B-2, G-2, G-4
Implicit .PAGE directive, 6-61
Implicit .WORD directive, 2-1, 2-4, 6-25
.INCLUDE directive, 6-2, 6-61, 9-8, B-6, C-6
Indefinite repeat block. See Repeat block, indefinite
Index deferred mode, 5-1, 5-5, B-2, G-2, G-4
Index mode, 5-1, 5-5, 5-7, B-2, G-2, G-4
Initial argument indicator, 3-2, B-1
Initial expression indicator, 3-2
Initial register indicator, 3-2, B-1
Instruction set
  commercial, C-3
  PDP-11, C-1
Interrupts, E-12
.IRP directive, 7-2, 7-17 to 7-19, B-6, D-2
.IRPC directive, 7-2, 7-17 to 7-19, B-6, D-2
Item terminator, 3-2, B-1

JMP instruction, 5-3, E-13
JSR instruction, 5-3, E-9

L error, 2-1
Label
  field, 2-1 to 2-3, E-1
  multiple definition, 2-3
  terminator, 3-1, B-1
.LIBRARY directive, 6-2, 6-60, 9-9, B-6, C-6
.LIMIT directive, 6-3, 6-39, B-6
Line format, E-1
Line printer listing format, 6-5, 6-6, 6-12. See also Listing control
Linker, 1-2, 2-2, 6-17, 6-43, 6-47, 6-51, F-4, G-1, G-4
Linking, 4-1, 6-40
.LIST directive, 6-3, 6-9 to 6-14, 6-21, 8-6, 8-11, 8-13, 9-4, B-6, D-1
Listing control, 6-4 to 6-14. See also .LIST directive, .NLIST directive
Listing control switches. See Switches, listing control
Listing level count, 6-9, 6-10, 6-12, B-6, B-7
Local symbol, 3-11, 3-12, 7-8, 7-9, D-4, E-4, F-2
Local symbol block, 3-11, 3-12, 6-20, D-4, F-2
Location counter. See Current location counter
Location counter control, 6-34 to 6-36
Logical AND operator, 3-2, 3-5, 6-55, B-1
Logical inclusive OR operator, 3-2, 3-5, 6-55
Logical OR operator, B-1

M error, 2-3, 3-1, 3-2, 3-8
Macro
  argument, 7-7, 7-14, 7-15, B-3
  argument concatenation, 7-11
  attribute directive, 7-12
  definition, 6-33, 7-1 to 7-13, 7-15, 7-17, 7-18, 7-20, B-4, B-6, B-7, E-6, F-2
  directive, 7-1, 7-2, 7-4. See also .MACRO directive
  expansion, 7-1, 7-3, 7-5 to 7-7, 7-9, 7-11, 7-17, B-7, D-4, F-2

Micro (Cont.)
  expansion listing, 6-9, 6-12
  keyword argument, 7-4, 7-10
  keyword indicator, 3-1
  name, 7-1, 7-2, 7-4, D-4, E-4
  nesting, 7-2, 7-3, 7-6, 7-17
  numeric argument, 7-7
  redefinition, F-3
  symbol, 3-6
Macro call, 7-1, 7-4 to 7-11, 7-12, 7-20, B-1, B-6. See also .MCALL directive
Macro call argument, 7-4
Macro call numeric argument, 3-3
.MACRO directive, 6-13, 6-21, 7-1 to 7-9, 7-10, 7-11, 9-6, B-6, D-1, F-3
Macro library directive. See .MCALL directive
Macro symbol table, 3-6, 3-7
MACRO-11 character set. See Character set, legal
.MCALL directive, 7-20, 8-6, 8-15, 9-5 to 9-6, B-6, D-4, F-1 to F-3
.MDELETE directive, 7-21, B-7, C-7
Memory
  allocation, 6-42, 6-47, F-1, F-2
  conservation, F-1
.MEXIT directive, 7-3, 7-18 to 7-20, B-7
Modularity, 6-44, E-8, F-1
Module checking routine, E-9
Module preface, E-5
Monitor console routine, 8-1, 8-2
MOV instruction, 3-13, 3-14, 6-37, 6-58, D-1, E-13, G-2 to G-4, H-2
MOVB instruction, H-2
Multinational character set. See DEC Multinational character set
Multiple definition. See M error
Multiple expression, 2-4
Multiple label, 2-2
Multiple symbol, 2-4
Multiplication operator, 3-2, 3-5, B-1

N error, 3-15
Naming standard, E-2
.NARG directive, 7-8, 7-12, 7-13, B-7, D-2
.NCHR directive, 7-12, 7-13, B-7, D-2
Nested conditional directive, 6-55, 6-58, 7-3
.NLIST directive, 6-3, 6-9 to 6-14, 6-15, 6-21, 8-6, 8-11, 8-13, 9-4, B-7, D-1
.NOCROSS directive, 6-3, 6-22, B-7, C-6
.NTYPE directive, 7-12, 7-14, B-7, D-2
Number of arguments. See .NARG directive
Numeric argument indicator, B-1
Numeric control
  operator, 6-33
  temporary, 6-36, B-3
Numeric directive, 6-34

O error, 6-40, 6-56, 6-57, 7-4, 7-12, 7-15, 7-21
Object module name, 1-2
.ODD directive, 6-3, 6-37, 6-38, B-7
Operand field, 2-1, 2-4, E-1
Operand field separator, 3-2, B-1
Operation field, E-1
Operator field, 2-1, 2-3, 2-4
Overlay, 6-42, 6-44

P error, 6-20, 7-16
.PACKED directive, 6-3, 6-31, 6-37, B-7, C-7
.PAGE directive, 6-3, 6-17, 6-61, 7-4, B-7
Page
  header, 6-4
  number, 6-17
Patch, E-15
Permanent symbol table, C-1 to C-3, 3-6, 3-7
Position-independent code, G-1 to G-4
.PRINT directive, 7-17, B-7
Processor priority, E-2
Program counter, 5-1, E-2, G-4
Program counter definition, 3-10
Program development system, 8-14
Program module, E-5
Program section directive. See .PSECT directive
Program section name, 6-41
Program section table, 1-1
Program version number. See Version identifier, program
Programming standard, E-1
.PSECT directive, 3-12, 3-14, 6-2, 6-3, 6-20, 6-41 to 6-48, 7-9, 9-6, B-7, D-1, D-2, H-2

Q error, 6-29, 6-34, 6-38

R error, 3-10
.RAD50 directive, 6-3, 6-29, B-8, H-2

Radix control, 3-15, 6-32, 6-34, B-8
  temporary, 6-31, 6-33, B-3
.RADIX directive, 3-15, 6-3, 6-32, B-8, D-1
Radix-50, 3-5, 6-30, 6-41, B-3, B-5, B-8
  character set, A-4
  temporary operator, 6-31
Read-only access, 6-41
Read/write access, 6-41
Register
  conventions, E-9
  definitions, default, 3-10, 6-21
  expression, 5-2, B-1
  symbol, 3-10, D-4
  term indicator, 3-2, B-1
Register deferred mode, 5-1, 5-2, B-2, G-1
Register mode, 5-1, 5-2, B-2, G-1
Relative deferred mode, 5-1, 5-8, B-2, G-2, G-4
Relative mode, 5-1, 5-7, 5-8, B-2, G-2, G-4
Relocatable expression, 3-17
Relocatable module, 6-43
Relocatable program section, 6-44 to 6-47, B-4
Relocation, 4-1, 6-43
Relocation bias, 2-2, 3-17, 3-18, 4-1, 6-43
.REM directive, 6-3, 6-18, B-8, C-7
Repeat block
  directive. See .REPT directive
  indefinite, 7-3, 7-17 to 7-20, B-4, B-6
.REPT directive, 7-2, 7-17, 7-20, B-8, D-3
Reserved symbols, 2-3, 3-1, 3-7
.RESTORE directive, 3-11, 3-14, 6-3, 6-20, 6-49, B-8, C-7, D-3
.RETURN directive, H-2
RSTS, 9-1 to 9-9
RSX run-time system, 9-1, 9-2
RSX-11M, 6-17, 6-41, 6-48, 7-21, 8-1 to 8-13, 8-19 to 8-22, E-12, F-3, G-1
RSX-11M-PLUS, 8-1 to 8-13, 8-19 to 8-22, G-1
RT-11, 6-17, 6-41, 6-43, 7-21, 9-1 to 9-9
RT-11 run-time system, 9-1

.SAVE directive, 6-3, 6-20, 6-49, 6-50, B-8, C-7, D-3
.SBTTL directive, 6-3, 6-4, 6-15, B-8, H-2
Separating characters, 3-3
Sequence number, 6-19
Single ASCII character indicator, 3-3, B-1, B-3
Source line format, 2-5
Source line terminator, B-1
Special characters, 3-1 to 3-3, 7-7
Stack pointer, E-2
  definition, 3-10
Statement format, 2-1
SUB instruction, E-13
Subconditional assembly, 6-56 to 6-59
Subtraction operator, 3-2, 3-5, B-1
Success/failure indicator, E-9
Switches
  file specification, 8-6
  function control, 8-6, 9-4
  listing control, 8-6, 8-7, 9-4
Symbol name syntax, E-3
Symbol table, 1-1, 1-2, F-1
Symbolic argument, 6-41
SYSLIB, F-4
System macro library, 1-1, 7-20, 8-4, 8-14, 9-3, 9-5. See also .MCALL directive

T error, 3-15, 6-24
Table of contents, 6-12, 6-16, B-8
Task builder. See Linker
Teleprinter listing format, 6-7, 6-13. See also Listing control
Temporary numeric control. See Numeric control, temporary
Temporary radix control. See Radix control, temporary
Temporary Radix-50 operator, 6-31
Term, definition of, 3-15
Terminal argument indicator, 3-2, B-1
Terminal expression indicator, 3-2
Terminal register indicator, 3-2, B-1
Terminating directive. See .END directive
Thrashing, F-1
.TITLE directive, 6-3, 6-4, 6-13, 6-15, 6-21, B-8, D-2, E-5, E-7, E-16, H-1
TRAP instruction, 5-9, D-4
TST instruction, E-10, E-11, H-2

U error, 3-8, 3-9, 3-15, 6-21, 7-21, 8-7, 8-9, 8-15
Unary operator, 3-4, 3-16, 7-5, 7-7
  control, 6-32, 6-34
  universal, 3-3, 3-5, B-1

Unconditional assembly, 6-56
Undefined symbol, 3-8, 6-21, D-2, D-4. See also U error
Universal unary operator. See Unary operator, universal
Upper-case ASCII, 6-19
User-defined symbol, 3-6 to 3-8
User-defined symbol table, 2-2, 3-6 to 3-8, 3-15

Version identifier
  assembler, 6-4
  file, 8-20
  program, 6-17, B-5
  standard, E-14 to E16. See also .IDENT directive

.WEAK directive, 6-3, 6-52, B-8, C-7
.WORD directive, 3-13, 3-14, 6-3, 6-24, 6-34, 6-36, B-8. See also Implicit .WORD directive

Z error, 5-3

PDP–11 MACRO–11
Language Reference
Manual
AD–V027A–T1

## READER'S COMMENTS

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

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

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

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

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

Name _____ Date _____

Organization _____ Telephone _____

Street _____

City _____ State _____ Zip Code _____
or Country

— — Do Not Tear — Fold Here and Tape — — — — — — — — — — — — — — — — — —

digital

No Postage Necessary if Mailed in the United States

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

POSTAGE WILL BE PAID BY ADDRESSEE

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

— — Do Not Tear — Fold Here — — — — — — — — — — — — — — — — — —

Cut Along Dotted Line