

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.

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

Title/Copyright

v/vi and vii/viii

1-1/1-2

2-3/2-4

3-1/3-2

3-11/3-12

5-3/5-4

6-15/6-16 thru 6-19/6-20

6-25/6-26 and 6-27/6-28

6-55/6-56

6-61/blank

7-5/7-6

7-9/7-10

7-13/7-14 thru 7-19/7-20

8-7/8-8

9-3/9-4

A-1/A-2

A-5/blank

J-1/J-2

Index-1/Index-2 thru Index-5/blank

Reader's Comments/Mailer

New page

Title/Copyright

v/vi and vii/viii

1-1/1-2

2-3/2-4

3-1/3-2

3-2.1/blank

3-11/3-12

5-3/5-4

6-15/6-16 thru 6-19/6-20

6-25/6-26 and 6-27/6-28

6-55/6-56

6-61/blank

7-5/7-6

7-9/7-10

7-13/7-14 thru 7-19/7-20

8-7/8-8

9-3/9-4

A-1/A-2

A-5/A-6

J-1/J-2

J-2.1/blank

Index-1/Index-2 thru Index-5/Index-6

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:

DEC	digital ™	RT-11
DECmate	MASSBUS	UNIBUS
DECsystem-10	PDP	VAX
DECSYSTEM-20	P/OS	VMS
DECUS	Professional	VT
DECwriter	Rainbow	Work Processor
DIBOL	RSTS	
	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: .NAME, .NAME, AND .NTYPE	7-12
	7.4.1	.NAME Directive	7-12
	7.4.2	.NAME Directive	7-13
	7.4.3	.NTYPE Directive	7-14
	7.5	.ERROR AND .PRINT DIRECTIVES	7-16
	7.6	INDEFINITE REPEAT BLOCK DIRECTIVES: .INT AND .IREP	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-58 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	3-2	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 .BLKN 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 .NCHS Directive	7-14
	7-3	Example of .NTYPE Directive in Macro Definition	7-15
	7-4	Example of .IRP and .IRPC Directives	7-18
	8-1	Sample CREF Listing	8-19
	8-1	Example of Position-Dependent Code	G-3
	8-2	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	6-13
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.

THE MACRO-11 ASSEMBLER

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 0.3 in the Preface).

SOURCE PROGRAM FORMAT

The legal characters for defining labels are:

A through Z
0 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-RAD50 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-50 characters.

SOURCE PROGRAM FORMAT

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, `ROUGHOUT`, 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 5.2.1, .ENABL IC).
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 0 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)

SYMBOLS AND EXPRESSIONS

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	Tab	Item or field terminator.
Space	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.

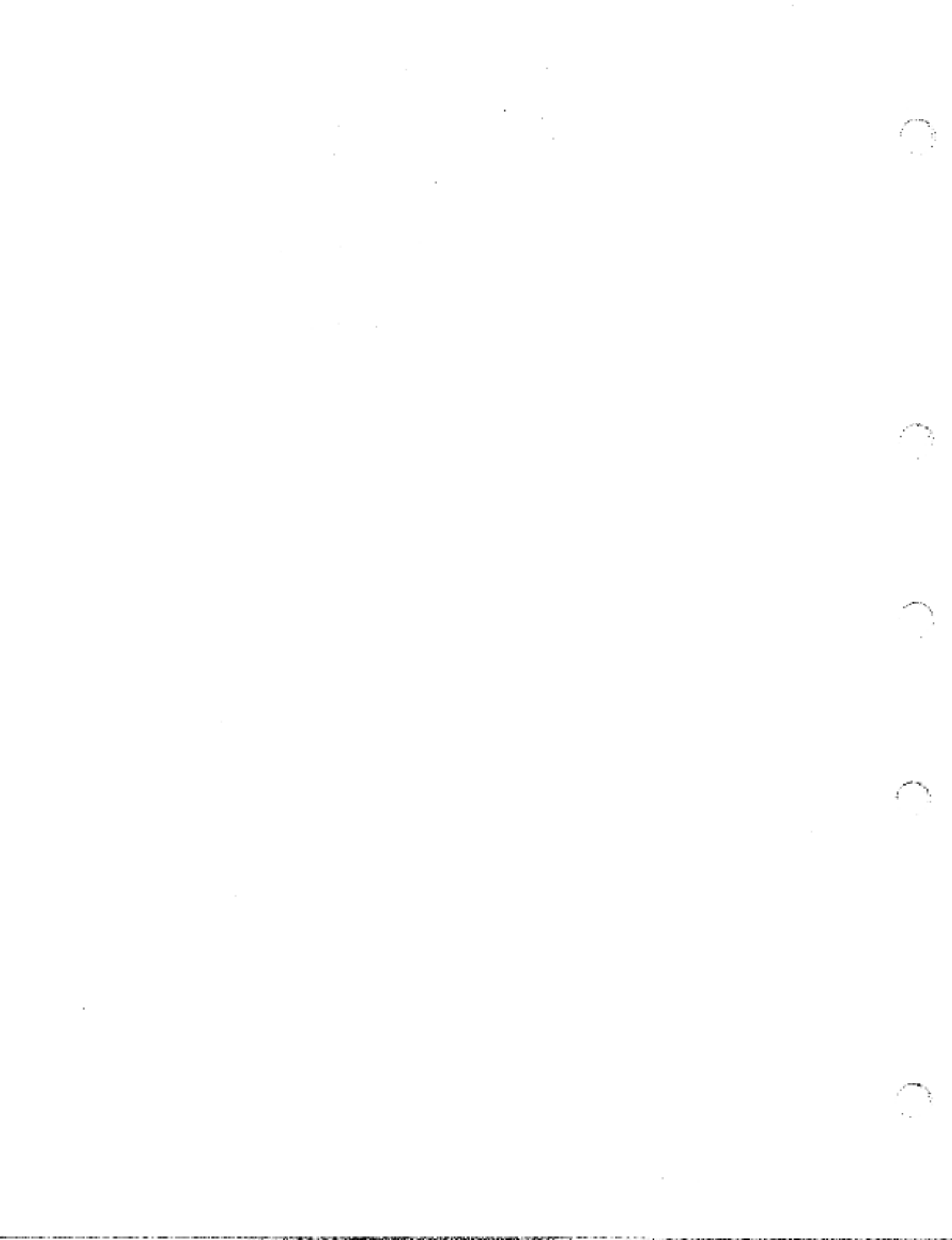
(continued on next page)

SYMBOLS AND EXPRESSIONS

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)



SYMBOLS AND EXPRESSIONS

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

```
ACB=%B
```

the statement

```
MULF (R0),ACB
```

multiplies the contents of floating-point accumulator register ACP 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 5.2.1)

SYMBOLS AND EXPRESSIONS

Ending Delimiter: .ENABL LSH

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 LSH (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 18 to 299999 first. Local symbols within the range 300000 through 655359, 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 188 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 XCFPAS.
4          *
5
6 000000 012700 XCFPRL MOV     $IMPURE,RC      *Point to IMPURE area
7          000006
8 000004 005020 181     CLR     (R014          *Clear a word
9 000006 020027       CMP     R0,4IMPUR1      *Test if at top of area
10          000008
11 000017 001374       BNE     18             *Iterate if not
12          *Fall in to perform pass initialization
13
14 000014 012700 XCFPAS MOV     $IMPUR3,RC      *Point to save storage area
15          000008
16 000020 005020 181     CLR     (R014          *Clear the area
17 000022 020027       CMP     R0,4IMPUR3      *Test if at top of area
18          000008
19 000024 001374       BNE     18             *Iterate if not
20 000026 000007       RTS     PC              *Return if so
  
```

Figure 3-1 Assembly Listing Showing Local Symbol Block

ADDRESSING MODES

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	(R0)+	;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 R2 originally contains 100(8):

MOV	R0,(R0)+	;The quantity 100 is moved	
		;to location 100.	
MOV	R0,-(R2)	;The quantity 102 is moved	
		;to location 100.	

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.

ADDRESSING MODES

5.4 AUTOINCREMENT DEFERRED MODE

Format:

$\theta(RR)+$

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

Example:

```
CLR     $\theta(R3)+$            ;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:

$-(RR)$

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

Examples:

```
CLR     $-(R0)$              ;Decrement the contents of the speci-
                        ;fied register (0, 1, 2, or 3) 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:

$\theta-(RR)$

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

Example:

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

.TITLE

6.1.2 .TITLE Directive

Format:

`.TITLE string`

where string represents:

An identifier of one or more Radix-58 characters.

An identifier of one or more 8-bit DBC Multinational character set (MCS) characters. Any MCS character must be preceded by six Radix-58 characters.

Appendix A.2 contains a table of Radix-58 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-58 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-58 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-58 character, the directive is flagged with an error code (A) in the assembly listing.

.SBTTL

6.1.3 .SBTTL Directive

Format:

`.SBTTL string`

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

GENERAL ASSEMBLER DIRECTIVES

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 `.LIST 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.

```

MTTENT - RT-11 MULTI-TERM EMT SE  M48RL JCS.OV Saturday 08/28/63 10:00
TABLE OF CONTENTS

50- 1      .MTOUT - Single character output EMT
51- 1      .MTRC'D - Reset CTRL/D EMT
52- 1      .MTATCH - Attach to terminal EMT
54- 1      .MTDICH - Detach from a terminal EMT
55- 1      .MTPRNT - Print message EMT
56- 1      .MTSTAT - Return multi-terminal system status EMT
57- 1      .MTIN - Single character input
58- 1      .MTGET - Get a character from the ring buffer
59- 1      .MTRSET - Reset terminal status bits
60- 1      .MTPUT - 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-52 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.

GENERAL ASSEMBLER DIRECTIVES

/ / 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.W0/
```

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.

.PAGE

6.1.5 .PAGE Directive/Page Ejection

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.

GENERAL ASSEMBLER DIRECTIVES

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 facsimilate 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.

.REM

6.1.6 .REM Directive/Begin Remark Lines

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 5
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.

GENERAL ASSEMBLER DIRECTIVES

.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 17). This function is useful during the debugging phase of program development.
CER	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 73-88.
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.
FST	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)

GENERAL ASSEMBLER DIRECTIVES

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. 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). 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)

GENERAL ASSEMBLER DIRECTIVES

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 120 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).

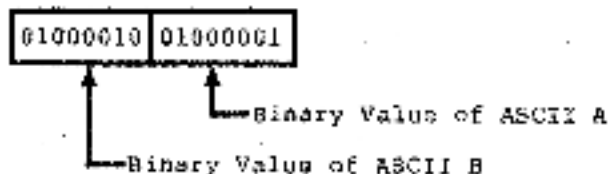
The single quote (') character must not be followed by a carriage-return, null, RUBOUT, line-feed, or form-feed character; if it is, an error code (A) is generated in the assembly listing.

GENERAL ASSEMBLER DIRECTIVES

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,R0
```

moves the following 16-bit expression value into register R0:



Thus the expression *AB results in a value of \$411E1(\$).

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.

.ASCII

6.3.4 .ASCII Directive

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.

GENERAL ASSEMBLER DIRECTIVES

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.
```

GENERAL ASSEMBLER DIRECTIVES

```
.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 (0)
                      ;is generated upon
                      ;encountering the
                      ;second = sign.
```

6.3.5 .ASCII2 Directive

.ASCII2

Format:

```
.ASCII2 /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 .ASCII2 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.

GENERAL ASSEMBLER DIRECTIVES

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

Conditions		Arguments	Assemble Block If:
Positive	Complement		
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.

GENERAL ASSEMBLER DIRECTIVES

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 (0) in the assembly listing.

MACRO-11 permits a nesting depth of 16(18) conditional assembly levels. Any statement that attempts to exceed this nesting level depth is flagged with an error code (0) in the assembly listing.

.IFF

.IFT

.IFTF

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.

.INCLUDE

6.10.2 .INCLUDE Directive

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.

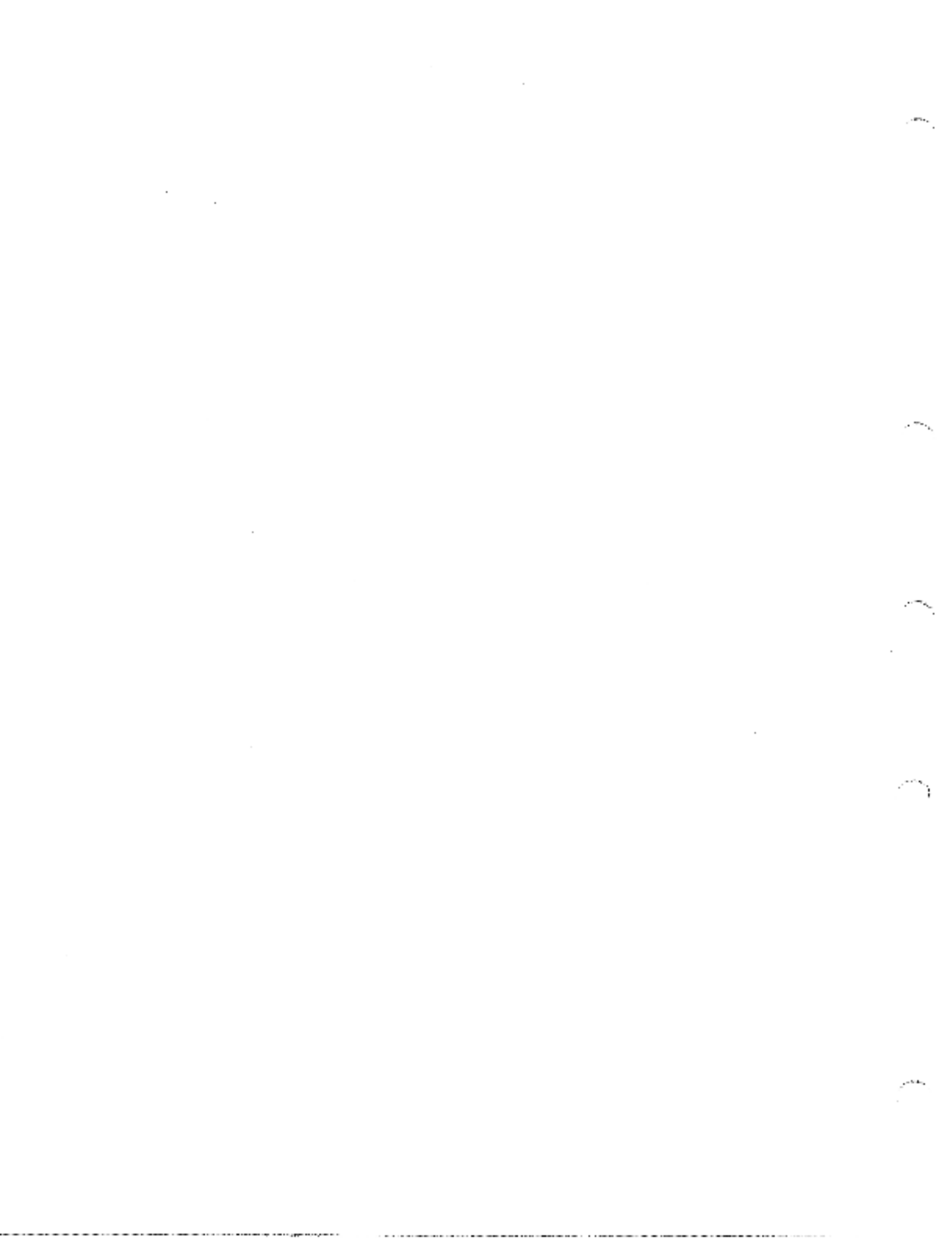
If any information is omitted from the source file argument, default values are assumed. The default file specification for MACRO-11/RT-11 is DSK:MAC, and for other systems it is SY:MAC.

The .INCLUDE directive is used as follows:

```
.INCLUDE      /DSK:[1,2]MACROS/      ;File MACROS.MAC
.INCLUDE      DSK:SYNDEF?
.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 DIRECTIVES

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  SEN 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    <^D 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/
```

MACRO DIRECTIVES

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.

MACRO DIRECTIVES

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     38888$                  ;Local symbol is created.
ADD     $5,R1
38888$:

```

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 36(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 .ENABLE 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.

MACRO DIRECTIVES

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
18 000000      000000    .WORD    A
19 000002      000000    .WORD    B
20 000004      000000    .WORD    C
21
22
23 000006      TEST    ADDRES=20,BLOCK=30,CONTRL=40
24 000006      000040    .WORD    40
25 000010      000030    .WORD    30
26 000012      000020    .WORD    20
27
28
29 000014      TEST    BLOCK=5
30 000014      000001    .WORD    1
31 000016      000005    .WORD    5
32 000020      000000    .WORD    TEMP

```

MACRO DIRECTIVES

```

1          .TITLE  NARG
2
3          .ENABL  _LC
4          .LIST   _MC
5
6          ;+
7          ; Example of the .NARG directive
8          ;
9          .MACRO  NULL      NUM
10         .NARG  SYN
11         .IF EQ  SYN
12         .MEXIT
13         .IFF
14         .RCPT  NUM
15         .NOP
16         .ENDM
17         .ENMC
18
19         .ENDM
20 000000      000000      NULL      &
21         .NARG  SYN
22         .IF EQ  SYN
23         .MEXIT
24         .IFF
25         .RCPT  &
26         .NOP
27         .ENDM
28         .ENMC
29
30 000000 000240      .NOP
31 000002 000240      .NOP
32 000004 000240      .NOP
33 000006 000240      .NOP
34 000010 000240      .NOP
35 000012 000240      .NOP
36         .ENMC
37
38 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).

MACRO DIRECTIVES

`<string>` represents a string of printable 7-bit ASCII or 8-bit 85C Multinational characters. If the character string contains a logical 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 MACHO-II 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          .CWARL  LC
4          .LIST  ME
5
6          !+
7          ! Illustrate the .NCHR directive
8          !-
9
10         .MACRO  STRING  MESSAG
11         .NCHR  $$$MESSAG
12         .WORD  $$$
13         .ASCII /MESSAG/
14         .EVEN
15
16 000000  MSG11  STRING  <Hello>
17         .NCHR  $$$Hello
18         .WORD  $$$
19         .ASCII /Hello/
20         .EVEN
21
22 000000  MSG11  STRING  <Hello>
23         .NCHR  $$$Hello
24         .WORD  $$$
25         .ASCII /Hello/
26         .EVEN
27
28 000000  MSG11  STRING  <Hello>
29         .NCHR  $$$Hello
30         .WORD  $$$
31         .ASCII /Hello/
32         .EVEN
33
34 000000  MSG11  STRING  <Hello>
35         .NCHR  $$$Hello
36         .WORD  $$$
37         .ASCII /Hello/
38         .EVEN
39
40 000000  MSG11  STRING  <Hello>
41         .NCHR  $$$Hello
42         .WORD  $$$
43         .ASCII /Hello/
44         .EVEN
45
46 000000  MSG11  STRING  <Hello>
47         .NCHR  $$$Hello
48         .WORD  $$$
49         .ASCII /Hello/
50         .EVEN
51
52 000000  MSG11  STRING  <Hello>
53         .NCHR  $$$Hello
54         .WORD  $$$
55         .ASCII /Hello/
56         .EVEN
57
58 000000  MSG11  STRING  <Hello>
59         .NCHR  $$$Hello
60         .WORD  $$$
61         .ASCII /Hello/
62         .EVEN
63
64 000000  MSG11  STRING  <Hello>
65         .NCHR  $$$Hello
66         .WORD  $$$
67         .ASCII /Hello/
68         .EVEN
69
70 000000  MSG11  STRING  <Hello>
71         .NCHR  $$$Hello
72         .WORD  $$$
73         .ASCII /Hello/
74         .EVEN
75
76 000000  MSG11  STRING  <Hello>
77         .NCHR  $$$Hello
78         .WORD  $$$
79         .ASCII /Hello/
80         .EVEN
81
82 000000  MSG11  STRING  <Hello>
83         .NCHR  $$$Hello
84         .WORD  $$$
85         .ASCII /Hello/
86         .EVEN
87
88 000000  MSG11  STRING  <Hello>
89         .NCHR  $$$Hello
90         .WORD  $$$
91         .ASCII /Hello/
92         .EVEN
93
94 000000  MSG11  STRING  <Hello>
95         .NCHR  $$$Hello
96         .WORD  $$$
97         .ASCII /Hello/
98         .EVEN
99
100 000000  MSG11  STRING  <Hello>
101        .NCHR  $$$Hello
102        .WORD  $$$
103        .ASCII /Hello/
104        .EVEN
105
106 000000  MSG11  STRING  <Hello>
107        .NCHR  $$$Hello
108        .WORD  $$$
109        .ASCII /Hello/
110        .EVEN
111
112 000000  MSG11  STRING  <Hello>
113        .NCHR  $$$Hello
114        .WORD  $$$
115        .ASCII /Hello/
116        .EVEN
117
118 000000  MSG11  STRING  <Hello>
119        .NCHR  $$$Hello
120        .WORD  $$$
121        .ASCII /Hello/
122        .EVEN
123
124 000000  MSG11  STRING  <Hello>
125        .NCHR  $$$Hello
126        .WORD  $$$
127        .ASCII /Hello/
128        .EVEN
129
130 000000  MSG11  STRING  <Hello>
131        .NCHR  $$$Hello
132        .WORD  $$$
133        .ASCII /Hello/
134        .EVEN
135
136 000000  MSG11  STRING  <Hello>
137        .NCHR  $$$Hello
138        .WORD  $$$
139        .ASCII /Hello/
140        .EVEN
141
142 000000  MSG11  STRING  <Hello>
143        .NCHR  $$$Hello
144        .WORD  $$$
145        .ASCII /Hello/
146        .EVEN
147
148 000000  MSG11  STRING  <Hello>
149        .NCHR  $$$Hello
150        .WORD  $$$
151        .ASCII /Hello/
152        .EVEN
153
154 000000  MSG11  STRING  <Hello>
155        .NCHR  $$$Hello
156        .WORD  $$$
157        .ASCII /Hello/
158        .EVEN
159
160 000000  MSG11  STRING  <Hello>
161        .NCHR  $$$Hello
162        .WORD  $$$
163        .ASCII /Hello/
164        .EVEN
165
166 000000  MSG11  STRING  <Hello>
167        .NCHR  $$$Hello
168        .WORD  $$$
169        .ASCII /Hello/
170        .EVEN
171
172 000000  MSG11  STRING  <Hello>
173        .NCHR  $$$Hello
174        .WORD  $$$
175        .ASCII /Hello/
176        .EVEN
177
178 000000  MSG11  STRING  <Hello>
179        .NCHR  $$$Hello
180        .WORD  $$$
181        .ASCII /Hello/
182        .EVEN
183
184 000000  MSG11  STRING  <Hello>
185        .NCHR  $$$Hello
186        .WORD  $$$
187        .ASCII /Hello/
188        .EVEN
189
190 000000  MSG11  STRING  <Hello>
191        .NCHR  $$$Hello
192        .WORD  $$$
193        .ASCII /Hello/
194        .EVEN
195
196 000000  MSG11  STRING  <Hello>
197        .NCHR  $$$Hello
198        .WORD  $$$
199        .ASCII /Hello/
200        .EVEN
201
202 000000  MSG11  STRING  <Hello>
203        .NCHR  $$$Hello
204        .WORD  $$$
205        .ASCII /Hello/
206        .EVEN
207
208 000000  MSG11  STRING  <Hello>
209        .NCHR  $$$Hello
210        .WORD  $$$
211        .ASCII /Hello/
212        .EVEN
213
214 000000  MSG11  STRING  <Hello>
215        .NCHR  $$$Hello
216        .WORD  $$$
217        .ASCII /Hello/
218        .EVEN
219
220 000000  MSG11  STRING  <Hello>
221        .NCHR  $$$Hello
222        .WORD  $$$
223        .ASCII /Hello/
224        .EVEN
225
226 000000  MSG11  STRING  <Hello>
227        .NCHR  $$$Hello
228        .WORD  $$$
229        .ASCII /Hello/
230        .EVEN
231
232 000000  MSG11  STRING  <Hello>
233        .NCHR  $$$Hello
234        .WORD  $$$
235        .ASCII /Hello/
236        .EVEN
237
238 000000  MSG11  STRING  <Hello>
239        .NCHR  $$$Hello
240        .WORD  $$$
241        .ASCII /Hello/
242        .EVEN
243
244 000000  MSG11  STRING  <Hello>
245        .NCHR  $$$Hello
246        .WORD  $$$
247        .ASCII /Hello/
248        .EVEN
249
250 000000  MSG11  STRING  <Hello>
251        .NCHR  $$$Hello
252        .WORD  $$$
253        .ASCII /Hello/
254        .EVEN
255
256 000000  MSG11  STRING  <Hello>
257        .NCHR  $$$Hello
258        .WORD  $$$
259        .ASCII /Hello/
260        .EVEN
261
262 000000  MSG11  STRING  <Hello>
263        .NCHR  $$$Hello
264        .WORD  $$$
265        .ASCII /Hello/
266        .EVEN
267
268 000000  MSG11  STRING  <Hello>
269        .NCHR  $$$Hello
270        .WORD  $$$
271        .ASCII /Hello/
272        .EVEN
273
274 000000  MSG11  STRING  <Hello>
275        .NCHR  $$$Hello
276        .WORD  $$$
277        .ASCII /Hello/
278        .EVEN
279
280 000000  MSG11  STRING  <Hello>
281        .NCHR  $$$Hello
282        .WORD  $$$
283        .ASCII /Hello/
284        .EVEN
285
286 000000  MSG11  STRING  <Hello>
287        .NCHR  $$$Hello
288        .WORD  $$$
289        .ASCII /Hello/
290        .EVEN
291
292 000000  MSG11  STRING  <Hello>
293        .NCHR  $$$Hello
294        .WORD  $$$
295        .ASCII /Hello/
296        .EVEN
297
298 000000  MSG11  STRING  <Hello>
299        .NCHR  $$$Hello
300        .WORD  $$$
301        .ASCII /Hello/
302        .EVEN
303
304 000000  MSG11  STRING  <Hello>
305        .NCHR  $$$Hello
306        .WORD  $$$
307        .ASCII /Hello/
308        .EVEN
309
310 000000  MSG11  STRING  <Hello>
311        .NCHR  $$$Hello
312        .WORD  $$$
313        .ASCII /Hello/
314        .EVEN
315
316 000000  MSG11  STRING  <Hello>
317        .NCHR  $$$Hello
318        .WORD  $$$
319        .ASCII /Hello/
320        .EVEN
321
322 000000  MSG11  STRING  <Hello>
323        .NCHR  $$$Hello
324        .WORD  $$$
325        .ASCII /Hello/
326        .EVEN
327
328 000000  MSG11  STRING  <Hello>
329        .NCHR  $$$Hello
330        .WORD  $$$
331        .ASCII /Hello/
332        .EVEN
333
334 000000  MSG11  STRING  <Hello>
335        .NCHR  $$$Hello
336        .WORD  $$$
337        .ASCII /Hello/
338        .EVEN
339
340 000000  MSG11  STRING  <Hello>
341        .NCHR  $$$Hello
342        .WORD  $$$
343        .ASCII /Hello/
344        .EVEN
345
346 000000  MSG11  STRING  <Hello>
347        .NCHR  $$$Hello
348        .WORD  $$$
349        .ASCII /Hello/
350        .EVEN
351
352 000000  MSG11  STRING  <Hello>
353        .NCHR  $$$Hello
354        .WORD  $$$
355        .ASCII /Hello/
356        .EVEN
357
358 000000  MSG11  STRING  <Hello>
359        .NCHR  $$$Hello
360        .WORD  $$$
361        .ASCII /Hello/
362        .EVEN
363
364 000000  MSG11  STRING  <Hello>
365        .NCHR  $$$Hello
366        .WORD  $$$
367        .ASCII /Hello/
368        .EVEN
369
370 000000  MSG11  STRING  <Hello>
371        .NCHR  $$$Hello
372        .WORD  $$$
373        .ASCII /Hello/
374        .EVEN
375
376 000000  MSG11  STRING  <Hello>
377        .NCHR  $$$Hello
378        .WORD  $$$
379        .ASCII /Hello/
380        .EVEN
381
382 000000  MSG11  STRING  <Hello>
383        .NCHR  $$$Hello
384        .WORD  $$$
385        .ASCII /Hello/
386        .EVEN
387
388 000000  MSG11  STRING  <Hello>
389        .NCHR  $$$Hello
390        .WORD  $$$
391        .ASCII /Hello/
392        .EVEN
393
394 000000  MSG11  STRING  <Hello>
395        .NCHR  $$$Hello
396        .WORD  $$$
397        .ASCII /Hello/
398        .EVEN
399
400 000000  MSG11  STRING  <Hello>
401        .NCHR  $$$Hello
402        .WORD  $$$
403        .ASCII /Hello/
404        .EVEN
405
406 000000  MSG11  STRING  <Hello>
407        .NCHR  $$$Hello
408        .WORD  $$$
409        .ASCII /Hello/
410        .EVEN
411
412 000000  MSG11  STRING  <Hello>
413        .NCHR  $$$Hello
414        .WORD  $$$
415        .ASCII /Hello/
416        .EVEN
417
418 000000  MSG11  STRING  <Hello>
419        .NCHR  $$$Hello
420        .WORD  $$$
421        .ASCII /Hello/
422        .EVEN
423
424 000000  MSG11  STRING  <Hello>
425        .NCHR  $$$Hello
426        .WORD  $$$
427        .ASCII /Hello/
428        .EVEN
429
430 000000  MSG11  STRING  <Hello>
431        .NCHR  $$$Hello
432        .WORD  $$$
433        .ASCII /Hello/
434        .EVEN
435
436 000000  MSG11  STRING  <Hello>
437        .NCHR  $$$Hello
438        .WORD  $$$
439        .ASCII /Hello/
440        .EVEN
441
442 000000  MSG11  STRING  <Hello>
443        .NCHR  $$$Hello
444        .WORD  $$$
445        .ASCII /Hello/
446        .EVEN
447
448 000000  MSG11  STRING  <Hello>
449        .NCHR  $$$Hello
450        .WORD  $$$
451        .ASCII /Hello/
452        .EVEN
453
454 000000  MSG11  STRING  <Hello>
455        .NCHR  $$$Hello
456        .WORD  $$$
457        .ASCII /Hello/
458        .EVEN
459
460 000000  MSG11  STRING  <Hello>
461        .NCHR  $$$Hello
462        .WORD  $$$
463        .ASCII /Hello/
464        .EVEN
465
466 000000  MSG11  STRING  <Hello>
467        .NCHR  $$$Hello
468        .WORD  $$$
469        .ASCII /Hello/
470        .EVEN
471
472 000000  MSG11  STRING  <Hello>
473        .NCHR  $$$Hello
474        .WORD  $$$
475        .ASCII /Hello/
476        .EVEN
477
478 000000  MSG11  STRING  <Hello>
479        .NCHR  $$$Hello
480        .WORD  $$$
481        .ASCII /Hello/
482        .EVEN
483
484 000000  MSG11  STRING  <Hello>
485        .NCHR  $$$Hello
486        .WORD  $$$
487        .ASCII /Hello/
488        .EVEN
489
490 000000  MSG11  STRING  <Hello>
491        .NCHR  $$$Hello
492        .WORD  $$$
493        .ASCII /Hello/
494        .EVEN
495
496 000000  MSG11  STRING  <Hello>
497        .NCHR  $$$Hello
498        .WORD  $$$
499        .ASCII /Hello/
500        .EVEN
501
502 000000  MSG11  STRING  <Hello>
503        .NCHR  $$$Hello
504        .WORD  $$$
505        .ASCII /Hello/
506        .EVEN
507
508 000000  MSG11  STRING  <Hello>
509        .NCHR  $$$Hello
510        .WORD  $$$
511        .ASCII /Hello/
512        .EVEN
513
514 000000  MSG11  STRING  <Hello>
515        .NCHR  $$$Hello
516        .WORD  $$$
517        .ASCII /Hello/
518        .EVEN
519
520 000000  MSG11  STRING  <Hello>
521        .NCHR  $$$Hello
522        .WORD  $$$
523        .ASCII /Hello/
524        .EVEN
525
526 000000  MSG11  STRING  <Hello>
527        .NCHR  $$$Hello
528        .WORD  $$$
529        .ASCII /Hello/
530        .EVEN
531
532 000000  MSG11  STRING  <Hello>
533        .NCHR  $$$Hello
534        .WORD  $$$
535        .ASCII /Hello/
536        .EVEN
537
538 000000  MSG11  STRING  <Hello>
539        .NCHR  $$$Hello
540        .WORD  $$$
541        .ASCII /Hello/
542        .EVEN
543
544 000000  MSG11  STRING  <Hello>
545        .NCHR  $$$Hello
546        .WORD  $$$
547        .ASCII /Hello/
548        .EVEN
549
550 000000  MSG11  STRING  <Hello>
551        .NCHR  $$$Hello
552        .WORD  $$$
553        .ASCII /Hello/
554        .EVEN
555
556 000000  MSG11  STRING  <Hello>
557        .NCHR  $$$Hello
558        .WORD  $$$
559        .ASCII /Hello/
560        .EVEN
561
562 000000  MSG11  STRING  <Hello>
563        .NCHR  $$$Hello
564        .WORD  $$$
565        .ASCII /Hello/
566        .EVEN
567
568 000000  MSG11  STRING  <Hello>
569        .NCHR  $$$Hello
570        .WORD  $$$
571        .ASCII /Hello/
572        .EVEN
573
574 000000  MSG11  STRING  <Hello>
575        .NCHR  $$$Hello
576        .WORD  $$$
577        .ASCII /Hello/
578        .EVEN
579
580 000000  MSG11  STRING  <Hello>
581        .NCHR  $$$Hello
582        .WORD  $$$
583        .ASCII /Hello/
584        .EVEN
585
586 000000  MSG11  STRING  <Hello>
587        .NCHR  $$$Hello
588        .WORD  $$$
589        .ASCII /Hello/
590        .EVEN
591
592 000000  MSG11  STRING  <Hello>
593        .NCHR  $$$Hello
594        .WORD  $$$
595        .ASCII /Hello/
596        .EVEN
597
598 000000  MSG11  STRING  <Hello>
599        .NCHR  $$$Hello
600        .WORD  $$$
601        .ASCII /Hello/
602        .EVEN
603
604 000000  MSG11  STRING  <Hello>
605        .NCHR  $$$Hello
606        .WORD  $$$
607        .ASCII /Hello/
608        .EVEN
609
610 000000  MSG11  STRING  <Hello>
611        .NCHR  $$$Hello
612        .WORD  $$$
613        .ASCII /Hello/
614        .EVEN
615
616 000000  MSG11  STRING  <Hello>
617        .NCHR  $$$Hello
618        .WORD  $$$
619        .ASCII /Hello/
620        .EVEN
621
622 000000  MSG11  STRING  <Hello>
623        .NCHR  $$$Hello
624        .WORD  $$$
625        .ASCII /Hello/
626        .EVEN
627
628 000000  MSG11  STRING  <Hello>
629        .NCHR  $$$Hello
630        .WORD  $$$
631        .ASCII /Hello/
632        .EVEN
633
634 000000  MSG11  STRING  <Hello>
635        .NCHR  $$$Hello
636        .WORD  $$$
637        .ASCII /Hello/
638        .EVEN
639
640 000000  MSG11  STRING  <Hello>
641        .NCHR  $$$Hello
642        .WORD  $$$
643        .ASCII /Hello/
644        .EVEN
645
646 000000  MSG11  STRING  <Hello>
647        .NCHR  $$$Hello
648        .WORD  $$$
649        .ASCII /Hello/
650        .EVEN
651
652 000000  MSG11  STRING  <Hello>
653        .NCHR  $$$Hello
654        .WORD  $$$
655        .ASCII /Hello/
656        .EVEN
657
658 000000  MSG11  STRING  <Hello>
659        .NCHR  $$$Hello
660        .WORD  $$$
661        .ASCII /Hello/
662        .EVEN
663
664 000000  MSG11  STRING  <Hello>
665        .NCHR  $$$Hello
666        .WORD  $$$
667        .ASCII /Hello/
668        .EVEN
669
670 000000  MSG11  STRING  <Hello>
671        .NCHR  $$$Hello
672        .WORD  $$$
673        .ASCII /Hello/
674        .EVEN
675
676 000000  MSG11  STRING  <Hello>
677        .NCHR  $$$Hello
678        .WORD  $$$
679        .ASCII /Hello/
680        .EVEN
681
682 000000  MSG11  STRING  <Hello>
683        .NCHR  $$$Hello
684        .WORD  $$$
685        .ASCII /Hello/
686        .EVEN
687
688 000000  MSG11  STRING  <Hello>
689        .NCHR  $$$Hello
690        .WORD  $$$
691        .ASCII /Hello/
692        .EVEN
693
694 000000  MSG11  STRING  <Hello>
695        .NCHR  $$$Hello
696        .WORD  $$$
697        .ASCII /Hello/
698        .EVEN
699
700 000000  MSG11  STRING  <Hello>
701        .NCHR  $$$Hello
702        .WORD  $$$
703        .ASCII /Hello/
704        .EVEN
705
706 000000  MSG11  STRING  <Hello>
707        .NCHR  $$$Hello
708        .WORD  $$$
709        .ASCII /Hello/
710        .EVEN
711
712 000000  MSG11  STRING  <Hello>
713        .NCHR  $$$Hello
714        .WORD  $$$
715        .ASCII /Hello/
716        .EVEN
717
718 000000  MSG11  STRING  <Hello>
719        .NCHR  $$$Hello
720        .WORD  $$$
721        .ASCII /Hello/
722        .EVEN
723
724 000000  MSG11  STRING  <Hello>
725        .NCHR  $$$Hello
726        .WORD  $$$
727        .ASCII /Hello/
728        .EVEN
729
730 000000  MSG11  STRING  <Hello>
731        .NCHR  $$$Hello
732        .WORD  $$$
733        .ASCII /Hello/
734        .EVEN
735
736 000000  MSG11  STRING  <Hello>
737        .NCHR  $$$Hello
738        .WORD  $$$
739        .ASCII /Hello/
740        .EVEN
741
742 000000  MSG11  STRING  <Hello>
743        .NCHR  $$$Hello
744        .WORD  $$$
745        .ASCII /Hello/
746        .EVEN
747
748 000000  MSG11  STRING  <Hello>
749        .NCHR  $$$Hello
750        .WORD  $$$
751        .ASCII /Hello/
752        .EVEN
753
754 000000  MSG11  STRING  <Hello>
755        .NCHR  $$$Hello
756        .WORD  $$$
757        .ASCII /Hello/
758        .EVEN
759
760 000000  MSG11  STRING  <Hello>
761        .NCHR  $$$Hello
762        .WORD  $$$
763        .ASCII /Hello/
764        .EVEN
765
766 000000  MSG11  STRING  <Hello>
767        .NCHR  $$$Hello
768        .WORD  $$$
769        .ASCII /Hello/
770        .EVEN
771
772 000000  MSG11  STRING  <Hello>
773        .NCHR  $$$Hello
774        .WORD  $$$
775        .ASCII /Hello/
776        .EVEN
777
778 000000  MSG11  STRING  <Hello>
779        .NCHR  $$$Hello
780        .WORD  $$$
781        .ASCII /Hello/
782        .EVEN
783
784 000000  MSG11  STRING  <Hello>
785        .NCHR  $$$Hello
786        .WORD  $$$
787        .ASCII /Hello/
788        .EVEN
789
790 000000  MSG11  STRING  <Hello>
791        .NCHR  $$$Hello
792        .WORD  $$$
793        .ASCII /Hello/
794        .EVEN
795
796 000000  MSG11  STRING  <Hello>
797        .NCHR  $$$Hello
798        .WORD  $$$
799        .ASCII /Hello/
800        .EVEN
801
802 000000  MSG11  STRING  <Hello>
803        .NCHR  $$$Hello
804        .WORD  $$$
805        .ASCII /Hello/
806        .EVEN
807
808 000000  MSG11  STRING  <Hello>
809        .NCHR  $$$Hello
810        .WORD  $$$
811        .ASCII /Hello/
812        .EVEN
813
814 000000  MSG11  STRING  <Hello>
815        .NCHR  $$$Hello
816        .WORD  $$$
817        .ASCII /Hello/
818        .EVEN
819
820 000000  MSG11  STRING  <Hello>
821        .NCHR  $$$Hello
822        .WORD  $$$
823        .ASCII /Hello/
824        .EVEN
825
826 000000  MSG11  STRING  <Hello>
827        .NCHR  $$$Hello
828        .WORD  $$$
829        .ASCII /Hello/
830        .EVEN
831
832 000000  MSG11  STRING  <Hello>
833        .NCHR  $$$Hello
834        .WORD  $$$
835        .ASCII /Hello/
836        .EVEN
837
838 000000  MSG11  STRING  <Hello>
839        .NCHR  $$$Hello
840        .WORD  $$$
841        .ASCII /Hello/
842        .EVEN
843
844 000000  MSG11  STRING  <Hello>
845        .NCHR  $$$Hello
846        .WORD  $$$
847        .ASCII /Hello/
848        .EVEN
849
850 000000  MSG11  STRING  <Hello>
851        .NCHR  $$$Hello
852        .WORD  $$$
853        .ASCII /Hello/
854        .EVEN
855
856 000000  MSG11  STRING  <Hello>
857        .NCHR  $$$Hello
858        .WORD  $$$
859        .ASCII /Hello/
860        .EVEN
861
862 000000  MSG11  STRING  <Hello>
863        .NCHR  $$$Hello
864        .WORD  $$$
865        .ASCII /Hello/
866        .EVEN
867
868 000000  MSG11  STRING  <Hello>
869        .NCHR  $$$Hello
870        .WORD  $$$
871        .ASCII /Hello/
872        .EVEN
873
874 000000  MSG11  STRING  <Hello>
875        .NCHR  $$$Hello
876        .WORD  $$$
877        .ASCII /Hello/
878        .EVEN
879
880 000000  MSG11  STRING  <Hello>
881        .NCHR  $$$Hello
882        .WORD  $$$
883        .ASCII /Hello/
884        .EVEN
885
886 000000  MSG11  STRING  <Hello>
887        .NCHR  $$$Hello
888        .WORD  $$$
889        .ASCII /Hello/
890        .EVEN
891
892 000000  MSG11  STRING  <Hello>
893        .NCHR  $$$Hello
894        .WORD  $$$
895        .ASCII /Hello/
896        .EVEN
897
898 000000  MSG11  STRING  <Hello>
899        .NCHR  $$$Hello
900        .WORD  $$$
901        .ASCII /Hello/
902        .EVEN
903
904 000000  MSG11  STRING  <Hello>
905        .NCHR  $$$Hello
906        .WORD  $$$
907        .ASCII /Hello/
908        .EVEN
909
910 000000  MSG11  STRING  <Hello>
911        .NCHR  $$$Hello
912        .WORD  $$$
913        .ASCII /Hello/
914        .EVEN
915
916 000000  MSG11  STRING  <Hello>
917        .NCHR  $$$Hello
918        .WORD  $$$
919        .ASCII /Hello/
920        .EVEN
921
922 000000  MSG11  STRING  <Hello>
923        .NCHR  $$$Hello
924        .WORD  $$$
925        .ASCII /Hello/
926        .EVEN
927
928 000000  MSG11  STRING  <Hello>
929        .NCHR  $$$Hello
930        .WORD  $$$
931        .ASCII /Hello/
932        .EVEN
933
934 000000  MSG11  STRING  <Hello>
935        .NCHR  $$$Hello
936        .WORD  $$$
937        .ASCII /Hello/
938        .EVEN
939
940 000000  MSG11  STRING  <Hello>
941        .NCHR  $$$Hello
942        .WORD  $$$
943        .ASCII /Hello/
944        .EVEN
945
946 000000  MSG11  STRING  <Hello>
947        .NCHR  $$$Hello
948        .WORD  $$$
949        .ASCII /Hello/
950        .EVEN
951
952 000000  MSG11  STRING  <Hello>
953        .NCHR  $$$Hello
954        .WORD  $$$
955        .ASCII /Hello/
956        .EVEN
957
958 000000  MSG11  STRING  <Hello>
959        .NCHR  $$$Hello
960        .WORD  $$$
961        .ASCII /Hello/
962        .EVEN
963
964 000000  MSG11  STRING  <Hello>
965        .NCHR  $$$Hello
966        .WORD  $$$
967        .ASCII /Hello/
968        .EVEN
969
970 000000  MSG11  STRING  <Hello>
971        .NCHR  $$$Hello
972        .WORD  $$$
973        .ASCII /Hello/
974        .EVEN
975
976 000000  MSG11  STRING  <Hello>
977        .NCHR  $$$Hello
978        .WORD  $$$
979        .ASCII /Hello/
980        .EVEN
981
982 000000  MSG11  STRING  <Hello>
983        .NCHR  $$$Hello
984        .WORD  $$$
985        .ASCII /Hello/
986        .EVEN
987
988 000000  MSG11  STRING  <Hello>
989        .NCHR  $$$Hello
990        .WORD  $$$
991        .ASCII /Hello/
992        .EVEN
993
994 000000  MSG11  STRING  <Hello>
995        .NCHR  $$$Hello
996        .WORD  $$$
997        .ASCII /Hello/
998        .EVEN
999
1000 000000  MSG11  STRING  <Hello>
1001        .NCHR  $$$Hello
1002        .WORD  $$$
1003        .ASCII /Hello/
1004        .EVEN
1005
1006 000000  MSG11  STRING  <Hello>
1007        .NCHR  $$$Hello
1008        .WORD  $$$
1009        .ASCII /Hello/
1010        .EVEN
1011
1012 000000  MSG11  STRING  <Hello>
1013        .NCHR  $$$Hello
1014        .WORD  $$$
1015        .ASCII /Hello/
1016        .EVEN
1017
1018 000000  MSG11  STRING  <Hello>
1019        .NCHR  $$$Hello
1020        .WORD  $$$
1021        .ASCII /Hello/
1022        .EVEN
1023
1024 000000  MSG11  STRING  <Hello>
1025        .NCHR  $$$Hello
1026        .WORD  $$$
1027        .ASCII /Hello/
1028        .EVEN
1029
1030 000000  MSG11  STRING  <Hello>
1031        .NCHR  $$$Hello
1032        .WORD  $$$
1033        .ASCII /Hello/
1034        .EVEN
1035
1036 000000  MSG11  STRING  <Hello>
1037        .NCHR  $$$Hello
1038        .WORD  $$$
1039        .ASCII /Hello/
1040        .EVEN
1041
1042 000000  MSG11  STRING  <Hello>
1043        .NCHR  $$$Hello
1044        .WORD  $$$
1045        .ASCII /Hello/
1046        .EVEN
1047
1048 000000  MSG11  STRING  <Hello>
1049        .NCHR  $$$Hello
1050        .WORD  $$$
1051        .ASCII /Hello/
1052        .EVEN
1053
1054 000000  MSG11  STRING  <Hello>
1055        .NCHR  $$$Hello
1056        .WORD  $$$
1057        .ASCII /Hello/
1058        .EVEN
1059
1060 000000  MSG11  STRING  <Hello>
1061        .NCHR  $$$Hello
1062        .WORD  $$$
1063        .ASCII /Hello/
1064        .EVEN
1065
1066 000000  MSG11  STRING  <Hello>
1067        .NCHR  $$$Hello
1068        .WORD  $$$
1069        .ASCII /Hello/
1070        .EVEN
1071
1072 000000  MSG11  STRING  <Hello>
1073        .NCHR  $$$Hello
1074        .WORD  $$$
1075        .ASCII /Hello/
1076        .EVEN
1077
1078 000000  MSG11  STRING  <Hello>
1079        .NCHR  $$$Hello
1080        .WORD  $$$
1081        .ASCII /Hello/
1082        .EVEN
1083
1084 000000  MSG11  STRING  <Hello>
1085        .NCHR  $$$Hello
1086        .WORD  $$$
1087        .ASCII /Hello/
1088        .EVEN
1089
1090 000000  MSG11  STRING  <Hello>
1091        .NCHR  $$$Hello
1092        .WORD  $$$
1093        .ASCII /Hello/
1094        .EVEN
1095
1
```


MACRO DIRECTIVES

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

axep 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 (Q) 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   RE
5
6          ;;
7          ; Illustrate the .NTYPE directive
8          ;-
9
10         .MACRO  SAVE      ARG
11         .NTYPE  $$$ARG
12         .IF EQ  $$$170
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      000001      SAVE      R1
20         .NTYPE  $$$R1
21         .IF EQ  $$$170
22         MOV     R1,--(SP)      ;Save in register mode
23         .IFF
24         MOV     $R1,--(SP)    ;Save in non-register mode
25         .ENDC
26
27 000002      000047      SAVE      TEMP
28         .NTYPE  $$$TEMP
29         .IF EQ  $$$170
30         MOV     TEMP,--(SP)    ;Save in register mode
31         .IFF
32         MOV     $TEMP,--(SP)  ;Save in non-register mode
33         .ENDC
34
35 000006      000000      TEMPI   .WORD  0
36
37 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.

.ERROR

7.5 .ERROR AND .PRINT DIRECTIVES

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. No.	Loc. No.	Exp. Value	Text
P	512	005642	000075	.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).

.IRP

7.6.1 .IRP Directive

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.

MACRO DIRECTIVES

<code>sym</code>	represents a dummy argument that is replaced with successive real arguments from within the angle brackets. If no dummy argument is specified, the <code>.IRP</code> directive is flagged with an error code (A) in the assembly listing.
<code>,</code>	represents any legal separator (comma, space, and/or tab).
<code><argument list></code>	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 EBC 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.
<code>range</code>	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 <code>.REPT</code> directive (see Section 7.1.3).
<code>.ENDM</code>	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.

MACRO DIRECTIVES

, 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          ;+
6          ; illustrate the .IRP and .IRPC directives
7          ; by creating a pair of RAD50 tables
8          ;
9
10         000000      REG01  .IRP   REG,<PC,SP,RS,R4,R3,R2,R1,R0>
11         000000      .RAD50  /REG/
12         000000      .ENDR
13         000000      .RAD50  /PC/
14         000002      074500  .RAD50  /SP/
15         000004      072770  .RAD50  /RS/
16         000006      072720  .RAD50  /R4/
17         000010      072650  .RAD50  /R3/
18         000012      072600  .RAD50  /R2/
19         000014      072550  .RAD50  /R1/
20         000016      072460  .RAD50  /R0/
21
22         000020      REG021  .IRPC  HIGH,<76543210>
23         000020      .RAD50  /R'NUM/
24         000020      .ENDR
25         000020      073110  .RAD50  /R7/
26         000022      073040  .RAD50  /R6/
27         000024      072770  .RAD50  /RS/
28         000026      072720  .RAD50  /R4/
29         000030      072650  .RAD50  /R3/
30         000032      072600  .RAD50  /R2/
31         000034      072550  .RAD50  /R1/
32         000036      072460  .RAD50  /R0/
33
34         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 (0), 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 6.12.1) prior to the source file(s) that use macros defined in the library file. 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 /BN 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, /BN, and /DS; the option /ML applies only to the file immediately preceding the option so specified, whereas the /BN 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

RX-11M/RX-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. /NOCROSS_REFERENCE is the default.
/DISABLE:arg /ENABLE:arg /DISABLE:(arg,arg,...) /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. You can specify any of the following arguments with the /DISABLE or /ENABLE qualifier.
Argument	
ABSOLUTE	Enabling this function causes all relative addresses (address mode 67) to be assembled as absolute addresses (address mode 37).

(continued on next page)

RSTS/RT-11 OPERATING PROCEDURES

src1, represent the ASCII source (input) files containing the
src2,... MACRO-11 source program or the user-supplied macro
srcn 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 BINP.OBJ and a listing. The listing goes directly to the line printer.

*DK:BINP.OBJ,LP1-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.

*,LP1/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:	SYCMAC	.SML
User MACRO Library	DK: 11 first file, otherwise same as for preceding source file	Must specify	.MLR

RSTS/RT-11 OPERATING PROCEDURES

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: WP
```

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 /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.
/F:arg /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
2	000	NUL	Null, tape feed, CONTROL/SHIFT/P.
1	001	SOH	Start of heading; also SOM, start of message, CONTROL/A.
1	002	STX	Start of text; also EOA, end of address, CONTROL/R.
2	003	ETX	End of text; also EOM, end of message, CONTROL/C.
1	004	END	End of transmission (END); shuts off TWX machines, CONTROL/D.
0	005	ENQ	Enquiry (ENCRY); also WRU, CONTROL/E.
0	006	ACK	Acknowledge; also RU, CONTROL/P.
1	007	BEL	Rings the bell. CONTROL/G.
1	010	BS	Backspace; also FEO, format effector. backspaces some machines, CONTROL/H.
2	011	HT	Horizontal tab. CONTROL/I.
2	012	LF	Line feed or line space (new line); advances paper to next line, duplicated by CONTROL/J.
1	013	VT	Vertical tab (VTAB). CONTROL/K.
2	014	FF	Form feed to top of next page (PAGE). CONTROL/L.
1	015	CR	Carriage return to beginning of line; duplicated by CONTROL/M.
1	016	SO	Shift out; changes ribbon color to red. CONTROL/N.
0	017	SI	Shift in; changes ribbon color to black. CONTROL/O.
1	020	DLE	Data link escape. CONTROL/P (DC0).
0	021	DC1	Device control 1; turns transmitter (RPADER) on, CONTROL/Q (X ON).
0	022	DC2	Device control 2; turns punch or auxiliary on. CONTROL/R (TAPE, AUX ON).
1	023	DC3	Device control 3; turns transmitter (RPADER) off, CONTROL/S (X OFF).
2	024	DC4	Device control 4; turns punch or auxiliary off. CONTROL/T (AUX OFF).

MACRO-11 CHARACTER SETS

Even Parity Bit	7-Bit Octal Code	Character	Remarks
1	025	NAK	Negative acknowledge; also ERR, ERROR, CONTROL/U.
1	026	SYN	Synchronous flag (SYNC). CONTROL/V.
0	027	ETB	End of transmission block; also EOM, logical end of medium. CONTROL/W.
0	030	CAN	Cancel (CANCEL). CONTROL/X.
1	031	EM	End of medium. CONTROL/Y.
1	032	SUB	Substitute. CONTROL/Z.
0	033	ESC	Escape. CONTROL/SHIFT/X.
1	034	FS	File separator. CONTROL/SHIFT/L.
0	035	GS	Group separator. CONTROL/SHIFT/M.
0	036	RS	Record separator. CONTROL/SHIFT/N.
1	037	US	Unit separator. CONTROL/SHIFT/O.
1	040	SP	Space.
0	041	:	
0	042	"	
1	043	#	
0	044	\$	
1	045	%	
1	046	&	
0	047	'	Accent acute or apostrophe.
0	050	(	
1	051	)	
1	052	*	
0	053	+	
1	054	,	
0	055	-	
0	056	.	
1	057	/	
0	060	0	
1	061	1	
1	062	2	
0	063	3	
1	064	4	
0	065	5	
0	066	6	
1	067	7	
1	070	8	
0	071	9	
0	072	:	
1	073	;	
0	074	<	
1	075	=	
1	076	>	
0	077	?	
1	100	@	
0	101	A	
0	102	B	
1	103	C	
0	104	D	
1	105	E	
1	106	F	
0	107	G	
0	110	H	
0	111	I	

MACRO-11 CHARACTER SETS

Single Char. or First Char.		Second Character		Third Character	
V	124600	V	001560	V	000026
W	127700	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

MACRO-11 CHARACTER SETS

A.3 DEC MULTINATIONAL CHARACTER SET

				00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	1	2	3	STX	DEL	SP	0	1	2	3	4	5	6	7	8	9	A	a	
1	0	1	2	SOH	DEL	1	2	A	Q	a	q		PUL	1	-	Á	N	u	ü
2	0	1	2	STX	DEL	2	3	E	R	e	r		PUS	e	2	À	Ó	á	é
3	0	1	2	UTX	DEL	#	3	C	S	c	s		STS	4	3	À	Ô	à	è
4	0	1	2	UCY	DEL	\$	4	D	T	d	t	IND	OCH			À	Ö	ä	ö
5	0	1	2	ENQ	NAL	%	5	E	U	e	u	NEI	MW	Y	p	À	Ü	ü	ü
6	0	1	2	ACK	RYN	&	6	F	V	f	v	SSA	SPA		q	À	Û	û	ü
7	0	1	2	BEL	ETB	'	7	G	W	g	w	SSA	SPA	5		C	OE	c	oe
8	0	1	2	BS	CAN	:	8	H	X	h	x	UTS		6		E	Ø	e	ø
9	0	1	2	HY	FM	;	9	I	Y	i	y	OU		7		E	Ü	e	ü
10	0	1	2	LF	SHH	<	10	J	Z	j	z	UTR		8	8	E	U	e	é
11	0	1	2	VT	OK	=	11	K	I	k	i	PLD	CS	9	9	E	U	e	é
12	0	1	2	FF	FS	>	12	L	Y	l	y	PLU	ST	10	10	E	U	e	é
13	0	1	2	CR	GS	@	13	M	J	m	j	HO	GO	11	11	E	Y	j	y
14	0	1	2	SO	RE	A	14	N	-	n	-	SSP	PM			E			
15	0	1	2	SI	CS	B	15	O	_	o	_	SSJ	APC			E	E	E	E
				DEC Multinational Character Set															
				ASCII Control Set				ASCII Graphic Character Set								Add'l Control Set			

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.

<u>Macro</u>	<u>Section</u>
.ASCII directive	6.3.4
.ASCII2 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

RELEASE NOTES

3.1.2 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 \$16 to 127\$ for automatic local symbol generation. MACRO-11 now uses a range of 32280\$ 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 .TITL 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 SYN 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 PSPECT synopsis that is printed in the listing file, after the symbol table, includes the pspect 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.

RELEASE NOTES

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. .DELETE
 5. .NCROSS
 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, E-2
- Absolute binary output, G-19
- Absolute expression, 3-17
- Absolute mode, 5-1, 5-7, 8-2, G-3, 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
- Appetrophe, 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, 6-16, B-3
- `.ASCIIZ` directive, 6-1, 6-28, 6-36, H-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, 6-1
- 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, G-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
- BEO instruction, H-2
- BGE instruction, E-13
- BGT instruction, E-14
- BHI instruction, E-16
- BHIS instruction, E-13
- BIC instruction, Z-13
- Binary operator, 3-4, 3-5, 3-16
- Blank line, 2-1
- BLB 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, 7-10, G-1, H-2
- BR instruction, E-10, E-11, G-3
- Branch instruction
 - addressing, 5-9, D-2
 - use of, 2-13
- `.BYTES` directive, 6-2, 6-23, 6-36, B-4, D-4
- C bit, E-9
- CALL instruction, H-2
- Calling convention, H-8
- Character set
 - ASCII, A-1 to A-3
 - DEC Multinationals, A-6
 - logical, 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, 3-1, 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

INDEX

- Complex relocation, 4-1, 6-4
- Concatenation indicator, 3-1, 8-1, 8-3
- Conditional assembly, 6-51 to 6-56, 7-8, 7-16, D-4
 - immediate, 6-56
- Conditional assembly block, 7-1, 8-4, 8-5
- Conditional assembly directive, 6-49
- Copyright statement, E-5
- CROSS directive, 6-2, 6-22, 8-4, 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
- CSTART directive, 3-11, 3-13, 6-2, 6-44 to 6-47, 9-6, 9-4
- 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, 8-3, 8-7, D-2, D-5
- 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, 7-18
- ASCII conversion characters, 6-25 to 6-27
- Comment field, 2-4
- Directive support for
 - .ASCII, 6-26
 - .ASCII2, 6-28
 - .SPROR, 7-16
 - .IF, 6-55
 - .IFX, See .IF directive
 - .IRP, 7-18
 - .IRPC, 7-19
 - .MCHR, 7-13
 - .PRINT, See .ERROR directive
 - .REM, 6-18
 - .SBTTL, 4-15
 - .TITLE, 6-15
- Default radix, 3-14
- Default register definitions, 3-18, 6-21
- Deferred addressing indicator, 3-2, 8-1
- Delimiting characters, 3-3, 6-17, 6-39, 8-3 to 8-5, 8-8
- Device register, E-2
- Direct assignment
 - operator, 3-1, 3-2, 3-9, 8-1
 - statement, 3-6 to 3-9, 3-13, 6-37
- Directives. See Permanent symbol table
- DTV instruction, H-2
- Division operator, 3-2, 3-5, 8-1
- Double ASCII character indicator, 3-2, 8-1
- .DZABL directive, 6-2, 6-19 to 6-21, 8-5, 8-8, 9-4, 8-4, D-1
- Dummy argument, 3-2, 7-11, 7-17
- E error, 6-46
- EMT instruction, 3-9, C-4
- .ENABL directive, 6-2, 6-19 to 6-21, 8-5, 8-8, 9-4, 8-4, D-1, 7-2, D-4, F-2
- .END directive, 6-2, 6-43, 8-4, D-3, 8-2
- .ENEX directive, 6-2, 6-12, 6-53 to 6-56, 6-59, 7-3, 8-4
- .ENEM directive, 6-13, 8-21, 7-2, 7-3, 7-6 to 7-8, 7-12, 7-11, 7-17 to 7-19, D-4, 8-8, F-3
- .ENOR directive, 7-19, 7-28, 8-5, 8-9
- Entry point symbol, 6-52
- .ERROR directive, 7-16, 8-5, D-4
- Error messages, D-1 to D-5
- .EVEN directive, 6-2, 6-29, 6-38, 8-5
- Expression, evaluation of, 3-16
- Expression indicator, immediate, 3-2, 8-1
- External expression, 3-17
- External symbol, 6-52. See also Global symbol
- Field terminator, 3-2, 8-1
- FILES-11, 6-19
- Floating-point directives, 8-5. See also .FLT3 directive
- Floating-point indicator, 8-3
- Floating-point processor, 3-10, 6-34, 6-35, C-4
- Floating-point rounding, 6-19, 8-32
- Floating-point truncation, 6-19, 6-35
- .FLT2 directive, 6-2, 6-35, 8-5
- .FLT4 directive, 6-2, 6-35, 8-5
- FLX, 6-19
- Forbidden instructions, 8-13
- Format control, 2-5
- Formatted binary, 6-19
- FORTAN, 6-47, 8-15, C-2
- Forward reference, 3-8, 3-9, 3-13, 3-13, C-4
- Illegal, D-1
- Function control switches. See Switches, function control
- Function directive, 6-18

INDEX

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

- Hardware register, E-2

- I error, 6-38, 6-38
- IAS, 5-48, 7-21, 8-14 to 8-17, E-19 to E-22, G-1
- .IDENT directive, 6-2, 6-16, 8-5, D-2, E-5, E-7, E-15, H-1
- .IF directive, 6-2, 6-12, 6-43 to 6-59, 7-3, 7-8, 8-5, D-1, D-2
- .IFB directive, 6-2, 6-56 to 6-58, B-5
- .IFT directive, 6-2, 6-56 to 6-58, B-5
- .IFTT directive, 6-2, 6-56, 6-57, E-5
- .IIF directive, 6-2, 6-59, B-5, 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, 5-5, 8-2, 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-5, 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, 8-2, 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, F-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, 6-68, 9-9, B-5, C-6
- .LIMIT directive, 6-3, 6-39, E-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, 6-48
- .LIST directive, 5-3, 6-9 to 6-14, 6-21, 8-5, 8-11, 8-13, 9-4, B-6, D-1
- Listing control, 6-1 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-5, 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, 6-28, 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, D-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, 7-22, 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, 8-7, B-4, F-2

INDEX

- Micro (Cont.)
 - expansion listing, 6-9, 6-12
 - keyword argument, 7-4, 7-16
 - 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, 7-28, 8-1, 8-6. 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, 7-12, 7-11, 9-6, 8-6, D-1, F-1
- Macro library directive. See .MCALL directive
- Macro symbol table, 3-6, 3-7
- MACRO-11 character set. See Character set, legal
- .MCALL directive, 7-22, 8-6, 8-13, 9-5 to 9-6, 8-6, D-4, F-1 to F-3
- .MDELETE directive, 7-21, 8-7, C-7
- Memory
 - allocation, 6-42, 6-47, H-1, F-2
 - conservation, F-1
- .MEXIT directive, 7-3, 7-18 to 7-20, H-7
- Modularity, 6-44, H-8, F-1
- Module checking routine, E-3
- Module preface, F-5
- Monitor console routine, 8-1, 8-2
- MOV instruction, 3-13, 3-19, 6-37, 6-58, D-1, E-13, G-2 to G-4, H-2
- MOVW 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, H-1
- N error, 3-15
- Naming standard, E-2
- .NARG directive, 7-8, 7-12, 7-13, 8-7, D-2
- .NCHN directive, 7-12, 7-13, 8-7, D-2
- Nested conditional directive, 6-55, 6-58, 7-3
- .NLIST directive, 6-3, 6-3 to 6-14, 6-16, 6-21, 8-8, 8-11, 8-13, 9-4, 8-7, D-1
- .NOCROSS directive, 6-3, 6-22, 8-7, C-6
- .NTYPE directive, 7-12, 7-14, 8-7, D-2
- Number of arguments. See .NARG directive
- Numeric argument indicator, 3-1
- Numeric control
 - operator, 6-33
 - temporary, 6-36, 3-3
- Numeric directive, 6-14
- O error, 6-40, 6-56, 6-57, 7-4, 7-12, 7-15, 7-21
- Object module name, 1-7
- .ODD directive, 6-3, 6-37, 6-38, 8-7
- Operand field, 2-1, 2-4, E-1
- Operand field separator, 3-2, 8-1
- Operation field, 8-1
- Operator field, 2-1, 2-3, 2-9
- Overlay, 6-42, 8-44
- P error, 6-20, 7-16
- .PACKED directive, 6-3, 6-31, 6-37, 8-7, C-7
- .PAGE directive, 6-3, 6-17, 6-61, 7-4, 8-7
- 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, C-1 to G-4
- .PRINT directive, 7-17, 8-7
- Processor priority, E-2
- Program counter, 5-1, 8-2, 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, 6-43
- Program section table, 1-2
- 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, 8-7, D-1, D-2, H-2
- Q error, 6-29, 6-34, 6-38
- R error, 3-18
- .RAD53 directive, 6-3, 6-29, 8-8, H-2

INDEX

- Radix control, 3-15, 6-32, 6-34, B-8
 - temporary, 6-31, 6-33, B-3
- .RADIX directive, 3-15, 6-1, 6-12, B-8, D-1
- Radix-50, 3-5, 6-38, 6-41, B-3, D-5, D-8
 - character set, A-4
 - temporary operator, 6-31
- Read-only access, 6-41
- Read/write access, 6-41
- Register
 - conventions, B-9
 - definitions, default, 3-18, 6-21
 - expression, 5-2, D-1
 - symbol, 3-18, 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 7-22, B-4, B-6
- .REPT directive, 7-2, 7-17, 7-22, B-8, D-3
- Reserved symbols, 2-3, 3-1, 3-7
- .RESTORE directive, 3-11, 3-14, 6-3, 6-28, 6-49, B-8, C-7, D-3
- .RETURN directive, B-2
- RTS, 9-1 to 9-9
- REX 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, B-12, F-3, G-1
- RSX-11M-PLUS, B-1 to 8-13, 8-19 to 8-22, C-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, 6-22, 6-49, 6-50, 5-8, C-7, D-3
- .SBTTL directive, 6-3, 6-4, 6-15, B-8, B-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, 3-7
- Stack pointer, B-2
 - definition, 3-18
- Statement format, 2-1
- SUB instruction, 2-11
- Subconditional assembly, 6-56 to 6-59
- Subtraction operator, 3-2, 3-5, B-1
- Success/failure indicator, 7-9
- Switches
 - file specification, B-6
 - function control, 6-6, 9-4
 - listing control, 8-6, 8-7, 9-4
- Symbol name syntax, B-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, D-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
- Threading, F-1
- .TITLE directive, 6-3, 6-4, 6-13, 6-15, 6-21, 8-8, D-2, E-5, E-7, E-16, B-1
- TRAP instruction, 5-9, D-4
- TST instruction, B-18, B-11, B-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

INDEX

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, 6-1
 file, 8-29
 program, 6-17, 8-5
 standard, 2-14 to K16. See also
 IDENT directive

 WEAK directive, 6-3, 6-51, D-8, C-7
 WORD directive, 3-13, 3-14, 6-3,
 6-24, 6-34, 6-36, 8-9. See
 also Implicit WORD directive

 % error, 6-3

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