UNISYS

OS 1100 Procedure Definition Processor (PDP)

Operations Reference Manual

Relative to Release Level 13R1

January 1988

Printed in U S America

Priced Item

UP–10070.1–A

Contents

About This Manual

Purpose

This guide describes the OS 1100 Procedure Definition Processor (PDP) level 13R1 software. PDP is used to create and update symbolic entry points in a program file's procedure table.

Organization

This manual is organized as follows:

Section 1. Overview

This section provides an overview of the Procedure Definition Processor (PDP).

Section 2. Symbolic Input

This section describes the procedure definition directives used as symbolic input to PDP.

Section 3. Symbolic Output

This section describes the symbolic output produced by PDP.

Section 4. Copying Procedures

This section discusses the transfer of PDP symbolic elements.

Appendix A. PDP Diagnostic Messages

This appendix lists the four types of PDP diagnostic messages.

Appendix B. Restrictions

This appendix lists the restrictions for the current level of PDP.

Related Product Information

The following related documents apply to this guide when you use the OS 1100 software. Use the version that corresponds to the level of software at your site. Consult the *2200 Systems Product Information Library Directory* (7831 1578) for specific levels and document numbers.

***OS 2200 Exec System Software Executive Control Language (ECL) End Use and Programming Reference Manual, SB5R2* (7830 7949)**

This manual provides a complete reference for all ECL statements. It describes the purpose, format, options, and individual fields of each statement. The manual also includes guidelines for using ECL statements for specific situations, such as creating and naming files.

***OS 1100 ASCII COBOL Programming Reference Manual* (UP–8582.2–A)**

This manual describes OS 1100 ASCII COBOL and is intended for programmers using COBOL under control of the OS 1100 Executive.

***ASCII FORTRAN Programming Reference Manual* (UP–8244.4–B)**

This manual provides the necessary information for you to write programs in ASCII FORTRAN. It describes all the features of the American National Standard FORTRAN, X3.9–1978 (also known as FORTRAN 77), as well as the extensions that provide increased programming capabilities.

***OS 1100 Universal Compiling System (UCS) PLUS Programming Reference Manual Volume 1, PLUS Statements* (7831 0497)**

This manual is a detailed reference for the UCS PLUS language. It describes the language structure and compiler operations of UCS PLUS and contains syntax information and instructions for writing systems programs.

***OS 1100 Universal Compiling System (UCS) PLUS Programming Reference Manual Volume 2, Compiler and System Interface* (7831 2287)**

This manual is a detailed reference for the UCS PLUS language. It describes how to compile, link, and execute a UCS PLUS program, using the appropriate compiler options. It also describes how UCS PLUS interacts with other system software, such as Executive requests.

***OS 1100 Meta-Assembler (MASM) Programming Reference Manual* (7830 8269)**

This manual provides programmers and systems analysts with a detailed reference manual for the OS 1100 Meta-Assembler (MASM) processor and language. It includes comprehensive information about this release level of MASM. It is not a tutorial and does not attempt to teach anyone how to write MASM programs.

***OS 1100 System Service Routines Library (SYSLIB) Programming Reference Manual* (UP–8728)**

This manual describes the routines in the System Service Routines Library (SYSLIB).

***OS 2200 FURPUR End Use Reference Manual* (7831 1271)**

This manual serves as a reference for the use of FURPUR on 2200 Series systems. The manual describes FURPUR commands and command options that are used to control and move data within 2200 Series files and elements.

Section 1
Overview

1.1. General

This section provides an overview of the Procedure Definition Processor (PDP). It explains the format of the PDP control statement and discusses the options.

1.2. @PDP Format

PDP is normally a standard processor in the system absolute library file SYS$*LIB$; however, it may reside in a user file. (See the *ECL End Use and Programming Reference Manual* for additional information on processor control statements.)

The @PDP control statement is used to call the PDP. It must precede any procedure source images appearing in the runstream (I option only).

Format:

```
@label :PDP,options     si ,so ,,time-and-date
```

where:

options

Are the following:

B

Processes one or more Assembler (ASM) procedures.

C

Processes one or more COBOL PDP library entries.

E

Lists only those line images in which errors are detected and all error messages.

F

Processes one or more FORTRAN PDP procedures.

I

Creates a symbolic source language procedure element from the symbolic input in the runstream following the @PDP control statement. The *si* parameter specifies the name to be given to the symbolic source language procedure element. *so* is ignored and should be omitted. The procedure element is subtyped according to the procedure type option specified, that is, B = ASM, C = COB, F = FOR. J = PSP, and M = MSM.

J

Processes one or more PLUS PDP procedures.

L

Produces a complete listing of the symbolic input including Conversational Time Sharing (CTS) line numbers, if present.

M

Processes one or more MASM procedures (default option).

N

Suppresses all symbolic listing (default option).

T

Creates or updates the procedure table without generating a symbolic procedure element. This requires the procedure type option to specifically match the type of the source input element. This option cannot be used in conjunction with the I option.

V

Produces base and Permanent Correction File (PCF) relative line numbers along with new line numbers in a listing. This option is used in conjunction with updating a procedure element and the listing options. This option cannot be used in conjunction with the I option.

W

SIR$ option; list line–change statements and change images only.

X

Takes error exit (ER ERR$; see the *ER Programming Reference Manual* from PDP if errors are detected.

The B, C, F, J, and M options indicate the type of procedure source images input (and output, if output is specified). PDP assumes the Meta-Assembler (MASM) procedure source image type if no option is given.

The E, L, and N options indicate the type of listing PDP prints. PDP suppresses all symbolic listings if no listing option is specified.

The remaining options control the input, output, and processing of PDP.

si

Normally specifies the source input element. However, when the I option is specified, *si* specifies the source output procedure element.

When only the *si* field is given and the T option is not used, PDP only reads the symbolic input. This enables the user to obtain a listing of the procedures, using the E or L options, without updating the procedure table or generating a new symbolic procedure element.

so

Specifies the source output procedure element. *so* is specified only if PDP is to create or update a procedure element.

When the I option or T option is specified, this field is ignored.

time-and-date

This field is used to time stamp the symbolic output procedure element. This field is not required and is only recognized with the I option. If this field is omitted, the preceding commas must also be omitted. In addition, if the *time-and-date field* is omitted and symbolic output is specified, then the time and date that the symbolic element is entered in the program file is used as a time stamp.

This field is produced in the @PDP,I call statement generated by FURPUR when an @PCH,S statement is performed on a PDP procedure element. The time stamp indicates the date and time the symbolic element was created. *time-and-date* is used to retain this date when an element is recreated in another file by @PDP,I.

The *time–and–date* are specified in octal, in the following format:

```
tttttmmddyy
```

where *ttttt* is the number of seconds past midnight, *mm* is the month, *dd* is the date, and *yy* is the last two digits of the year (modulo 1964). Six digits must appear in the first field and two digits in each of the other three. Zeros should be used to fill the fields where necessary.

Note: *This field is not checked for validity or syntax; therefore, extreme care must be taken in coding this field.*

si and *so* must follow the conventions for file and element notation as described in the *ECL End Use and Programming Reference Manual.* Also, if only the file name is specified in the *so* field, the element name that is specified in the *si* field is assumed for *so*.

1.3. Files

If the *si* and *so* files are not already assigned to the run, PDP will exclusively assign these files. This is necessary when accessing the *si* and *so* files with a write function. PDP attaches an internal name to the file, if needed. When processing is complete, PDP restores these files to their original assign state and removes the internal file name.

1.4. Reusability

PDP is a reusable processor. This means that if successive calls on PDP are separated by the control statements @ADD, @EOF, @HDG, @JUMP, @LOG, @MSG, @SETC, and @TEST, PDP is not reloaded from mass storage but reads its own control statement, reinitializes itself, and processes the next source input element. This saves considerable time and I/O resources. This feature is also available if PDP is called from a user file (rather than from SYS$*LIB$) provided that all calls on PDP after the first in a sequence do not specify the user file (in other words, the call is @PDP...) from which PDP was obtained. If a reload of PDP is desired, any other control statement (other than those listed above), such as @ENDX, may be used to terminate the reusability sequence. When PDP is reused, it resets the storage size of its workspace to the original minimum value.

Section 2
Symbolic Input

2.1. General

The symbolic input and generated symbolic output are both accessible to the user through other system processors such as @ELT, @ED, and @CTS.

The symbolic input to PDP consists of source language statements and PDP procedure definition directives, coded in the Fieldata, ASCII/ISO, or the Attributed Character Data (ACD) character sets.

This symbolic input may be either in a program file SDF element or in the runstream following the @PDP statement (I option only).

If the symbolic input is in the runstream, it must be followed by an end-of-file (@EOF) or other control statement.

The symbolic input may contain one or more PDP procedure definitions; however, all the procedures must contain source language statements of the same type as specified by the procedure type option on the @PDP statement.

The character strings ‘ΔPROCΔ’, ‘ΔEND’, ‘#PROCΔ’, ‘#ENDPROCΔ’, and ‘DEF’ (in uppercase, lowercase, or a combination of both) are considered reserved words for PDP. Therefore, these character strings may not appear within the symbolic input in any con other than to define the start or end of a PDP procedure. However, these character strings may appear in comment statements or character string constants within the symbolic input.

Each procedure must be coded following the PDP procedure format for the symbolic input source language, as specified in 2.2 through 2.6.

2.2. COBOL PDP Procedures (Library Entries)

The symbolic input defining a COBOL PDP library entry has the following format:

```
label∆PROC

    source statements

∆END
```

where:

label

is a character string, beginning in column one, that does not exceed 30 characters in length.

The *label* may contain alphabetic characters A,B,...,Z, digits 0,1,...,9, and the minus character (-).

The *label* must contain at least one alphabetic character and must not have a minus character (-) as its first or last character.

The character string specified by *label* is the name of the COBOL PDP library entry entered into the COBOL procedure table.

PROC7

is a PDP directive separated from the *label* by at least one space. PROC indicates that a COBOL PDP library entry is beginning.

The word address of the symbolic image containing this directive is saved in the COBOL procedure table.

source statements

are COBOL source statements. *source statements* may include data declarations and COBOL directives. The COBOL COPY directive is not allowed.

Note: *PDP does not check these source statements for syntax or validity.*

∆END

is a PDP directive that must begin in column two. It indicates the last symbolic image in the COBOL PDP library entry.

Any line image in the symbolic input, defining a COBOL PDP library entry, that contains an asterisk character in column seven is considered to be a comment line image no matter what characters precede or follow the asterisk character.

COBOL PDP library entries may not be nested, either physically or implied. Physical nesting is indicated when the symbolic images defining another COBOL PDP library entry appear within the *source statements* section of a previous COBOL PDP library entry. Implied nesting is indicated when the COBOL COPY directive appears in the *source statements* section of a COBOL PDP library entry.

For information on accessing COBOL PDP library entries, see the *ASCII COBOL Programming Reference Manual.*

Example

The following example illustrates COBOL PDP procedures:

```
 1. @PDP,IC  COBOL–PROC
 2.       *
 3.       *  This procedure contains COBOL data declarations
 4.       *
 5. DATA-DECLARATION01  PROC
 6.         01  STATE–TABLE
 7.           05  STATE–CODE.
 8.             10  FILLER  PIC X(20)  VALUE '01ALABAMA               '.
 9.             10  FILLER  PIC X(20)  VALUE '02ARKANSAS              '.
10.  END
11.      *
12.      *    This procedure contains more COBOL data declarations
13.      *
14. DATA–DECLARATION02 PROC
15.      *
16.      *    This is the county table declaration
17.      *
18.           05  COUNTY–CODE.
19.             10  FILLER  PIC X(20)  VALUE '01ALBANY COUNTY         '.
20.  END
21. @EOF
```

1. This is a PDP processor call statement with the I and C options. The I option indicates that a new procedure element is inserted in the file TPF$ with a name of COBOL–PROC. The C option indicates that the images in the runstream following the processor call are processed as COBOL PDP procedures and the COBOL procedure table in the file TPF$ is updated.

2–4. COBOL comment lines.

Note: *These comments will not appear when the procedure is referenced because they do not reside within the procedure.*

5. Procedure declaration, starting in column one. The symbolic entry point name is DATA–DECLARATION01. The word address of this line image is saved by PDP as the starting address of procedure DATA–DECLARATION01.

6–9. Symbolic source lines in procedure DATA–DECLARATION01.

Note: *PDP does not check these statements for validity or syntax.*

10. End declaration, starting in column two. This declaration causes PDP to save the information about the COBOL PDP procedure DATA–DECLARATION01 for subsequent entry into the COBOL procedure table.

11–13. COBOL comment lines.

Note: *These comments will not appear when the procedure is referenced because they do not reside within the procedure text.*

14. Another procedure declaration, starting in column one. The symbolic entry point name is DATA–DECLARATION02. The word address of this line image is saved by PDP as the starting address of the procedure DATA–DECLARATION02.

15–17. COBOL comment lines.

Note: *These comments will appear when the procedure is referenced because they reside within the procedure text.*

18–19. Symbolic source lines in procedure DATA–DECLARATION02.

Note: *PDP does not check these statements for validity or syntax.*

20. End declaration, starting in column two. This declaration causes PDP to save the information about the COBOL PDP procedure DATA–DECLARATION02 for subsequent entry into the COBOL procedure table.

21. End-of-file control statement. This completes the input from the runstream and causes PDP to complete processing and terminate.

2.3. FORTRAN (FTN) PDP Procedures

The symbolic input defining a FORTRAN PDP procedure has the following format:

```
label∆PROC
 source statements
∆END
```

where:

label

is a character string, beginning in column one, that does not exceed 12 characters in length.

The *label* may contain alphabetic characters A,B,...,Z, digits 0,1,...,9, and the dollar sign character ($). (See the *ECL End Use and Programming Reference Manual* for caution on the use of '$'.)

The character string specified by *label* is the name of the FTN PDP procedure entered into the FORTRAN procedure table.

PROC

is a PDP directive separated from the *label* by at least one space. PROC indicates that an FTN PDP procedure is beginning.

The word address of the symbolic image containing this directive is saved in the FORTRAN procedure table.

source statements

are FTN source statements. These may include data declarations and FTN directives. The FTN INCLUDE directive is not allowed.

Note: *PDP does not check these statements for validity or syntax.*

ΔEND

is a PDP directive that must begin in column two. It indicates the last symbolic image in the FTN PDP procedure.

An FTN comment line of the form:

```
CxxxΔPROCΔyyyyyy ...
```

where C is in column one and *xxx* is nonexistent or a character string with the same attributes as *label* indicates the beginning of an FTN PDP procedure; for example:

```
CΔPROCΔ is the start of processing
```

is interpreted as the beginning of an FTN PDP procedure named 'C', instead of a comment line.

FTN PDP procedures may not be nested, either physically or implied. Physical nesting is indicated when the symbolic images defining another FTN PDP procedure appear within the *source statements* section of a previous FTN PDP procedure. Implied nesting is indicated when the FTN INCLUDE directive appears in the *source statements* section of an FTN PDP procedure.

For information on accessing FTN PDP procedures with the INCLUDE statement, see the *ASCII FORTRAN Programming Reference Manual.*

Example:

The following example illustrates FORTRAN PDP procedures:

```
 1. @ELT,I  FTN-PROC
 2. C
 3. C   This procedure contains FTN data specifications
 4. C
 5. SPECS3  PROC
 6.       COMPLEX  C
 7.       REAL  I,J,K
 8.       COMMON  /CB1/  C(5),K,O
 9.  END
10. *
11. *   This procedure contains format declarations
12. *
13. PRTFMT PROC
14. C
15. C   These are print format declarations
16. C
17.  100  FORMAT  (/,20x,13hTABLE HEADING)
18.  200  FORMAT  (5x,i3,2x,5f5.2)
19.  END
20. @PDP,FL   FTN-PROC
source statements
21. @EOF
```

1. This is an ELT processor call statement for building a symbolic input element named FTN–PROC in the file TPF$ for PDP. The ELT statement indicates that procedure text can be created by other processors, but must be processed by the @PDP statement (see item 20).

2–4. FTN comment lines.

Note: *These comment lines will not appear when the procedure is referenced because they are not within the procedure text.*

5. Procedure declaration, starting in column one. The symbolic entry point name is SPECS3. The word address of this line image is saved by PDP as the starting address for the procedure SPECS3.

6–8. Symbolic source lines in the FTN PDP procedure SPECS3.

Note: *PDP does not check these source statements for validity or syntax.*

9. End declaration, starting in column two. This causes PDP to save the information about the FTN PDP procedure SPECS3 for subsequent entry into the FORTRAN procedure table.

10–12. FTN comment lines.

Note: *These comment lines will not appear when the procedure is referenced because they are not within the procedure text.*

13. Another procedure declaration, starting in column one. The symbolic entry point name is PRTFMT. The word address of this line image is saved by PDP as the starting address for the procedure PRTFMT.

14–16. FTN comment lines.

Note: *These comment lines will appear when the procedure is referenced because they are within the procedure text.*

17–18. Symbolic source lines in the FTN PDP procedure PRTFMT.

Note: *PDP does not check these source statements for validity or syntax.*

19. End declaration, starting in column two. This causes PDP to save the information about the FTN PDP procedure PRTFMT for subsequent entry into the FORTRAN procedure table.

20. This is the PDP processor call with the F and L options. The *so* field specified on the processor call causes PDP to generate a source output procedure element (the F option types the element as a FORTRAN procedure). The F option indicates that the images in the symbolic input element are to be processed as FTN PDP procedures and the FORTRAN procedure table in the file TPF$ will be updated. The L option indicates a complete listing of the symbolic input is desired.

21. End-of-file control statement. This completes correction image input from the runstream (none in this case) and causes PDP to complete processing and terminate. No correction image input causes an exact copy of the symbolic input element to be placed in a FORTRAN procedure element.

2.4. PLUS PDP Procedures

The symbolic input defining a PLUS PDP procedure has the following format:

```
#PROC∆label

source statements

#ENDPROC
```

where:

#PROC

is a PDP directive beginning in column one. It indicates the beginning of a PLUS PDP procedure.

The word address of the symbolic image containing this directive is saved in the FORTRAN procedure table.

label

is a character string, beginning in column seven, that does not exceed 12 characters in length.

The *label* may contain alphabetic characters A,B,...,Z, digits 0,1,...,9, the minus character (-), and the dollar sign character ($). (See the *ECL End Use and Programming Reference Manual* for caution on the use of '$'.)

The character string specified by *label* is the name of the PLUS PDP procedure entered into the FORTRAN procedure table.

source statements

are PLUS source statements and may include data declarations and PLUS directives.

#ENDPROC

is a PDP directive beginning in column one. It indicates the last symbolic image in the PLUS PDP procedure.

PLUS PDP procedures may be nested either physically or implied. Physical nesting is indicated when the symbolic images defining another PLUS PDP procedure appear within the *source statements* section of a previous PLUS procedure. Each physically nested PLUS PDP procedure must be bracketed properly (in other words, matching #PROC, #ENDPROC pairs). Implied nesting is indicated when the PLUS COPY directive appears in the *source statements* section of a PLUS PDP procedure. The physical nesting level of PLUS PDP procedures may not exceed 10 levels.

For more information on accessing PLUS PDP procedures, see the COPY statement in the *UCS PLUS Programming Reference Manual Volume 1* and the *UCS PLUS Programming Reference Manual Volume 2*.

Example:

The following example illustrates a physically nested PLUS PDP procedure.

```
 1. @PDP,IJ  PLUS-PROC
 2. ″
 3. ″  This procedure contains only PLUS declarations
 4. ″
 5. #PROC    PLUSPROC
 6.    DECLARE  test_var01  :  36 BIT SIGNED INTEGER;
 7.                              ″
 8.                              ″  This variable is for testing
 9.                              ″
10.    DECLARE  test_var02  :  MACHINE WROD LOGICAL;
11. ″
12. ″  The following is a nested PLUS PDP procedure
13. ″
14. #PROC NESTEDPROC
15. BEGIN
16. END;
17. #ENDPROC
18.        DECLARE  test_var03  :  MACHINE WORD POINTER;
```

```
19.        DECLARE  test_var04  :  18 BIT INTEGER;
20. #ENDPROC
21. @EOF
```

1. This is a PDP processor call with the I and J options.

2–4. PLUS comment lines.

Note: *These comment lines will not appear when the procedure is referenced because they do not reside within the procedure text.*

5. Procedure declaration, starting in column one. The symbolic entry point name is PLUSPROC. The word address of this line image is saved by PDP as the starting address of the PLUS PDP procedure PLUSPROC.

6. Symbolic source line in procedure PLUSPROC.

7–9. PLUS comment lines.

Note: *These comment lines will appear when the procedure is referenced because they are in the procedure text.*

10. Symbolic source line in procedure PLUSPROC. Notice the misspelling of 'WORD'; PDP does not check the syntax of the procedure source lines.

11–13. PLUS comment lines.

14. Nested procedure declaration, starting in column one. The symbolic entry point name is NESTEDPROC. The word address of this line image is saved by PDP as the starting address of the PLUS PDP procedure NESTEDPROC.

15–16. Symbolic source lines in procedure NESTEDPROC.

17. End declaration, starting in column one. This causes PDP to save the information about the PLUS PDP procedure PLUSPROC for subsequent entry into the FORTRAN procedure table, and to close out the procedure PLUSPROC.

18–19. Symbolic source lines in procedure PLUSPROC.

20. End declaration, starting in column one. This causes PDP to save the information about the PLUS PDP procedure PLUSPROC for subsequent entry into the FORTRAN procedure table, and to close out the procedure PLUSPROC.

21. End-of-file control statement. This completes the input from the runstream and causes PDP to complete processing and terminate.

2.5. MASM Procedures

This subsection discusses MASM and ASM procedures in terms of what directives PDP recognizes and how PDP processes them. All MASM directives may start in any column except column one but the standard convention is to start in column 11. See the *MASM Programming Reference Manual* for the correct format and usage of all the MASM directives that PDP recognizes and for detailed information on MASM procedures, nesting, external labels, etc.

MASM procedures have a variety of formats that PDP can recognize. This subsection describes a basic format for these procedures and each of the MASM directives that PDP recognizes.

The symbolic input defining a MASM procedure has the following format:

```
            DEF
            definitions
label       $PROC
            source statements
            $END
```

where:

DEF

is a PDP directive used to introduce values and functions into the MASM assembly.

This directive is optional and, if omitted, the *definitions* section must also be omitted. The PDP DEF directive is distinct from the MASM $DEF (known as DEF) directive. PDP does not recognize the $DEF form of this directive. PDP interprets all DEF directives as the PDP type instead of the MASM type.

This directive enables PDP to save the word address in the file of the next symbolic line image. This word address is used as the starting address of the procedure definition following the *definitions* section.

definitions

are line images containing one of the following MASM directives: $EQU, $EQUF, $FORM, or $FUNC definitions. There are no other line images (except comments) allowed. See the *MASM Programming Reference Manual* for the correct usage and form of these directives.

The *definitions* section must appear only if the DEF directive appears.

label

is the MASM label field, starting in column one, as described in the *MASM Programming Reference Manual.*

$PROC

is a MASM directive recognized by PDP as the start of a MASM procedure definition.

If the DEF directive and *definitions* section do not precede the line image containing the $PROC directive, the word address of this line image is used as the starting address of the procedure.

Note: *The MASM $PROC directive may contain additional fields, but they are ignored by PDP (see the* MASM Programming Reference Manual *for the correct form and use of this directive).*

source statements

are line images containing MASM directives, instructions, declarations, etc.

The only MASM directives that PDP recognizes in this section are the $NAME and $FUNC definitions (see discussion below).

Note: *PDP does not check these source statements for syntax or validity.*

$END

is a MASM directive that indicates the last symbolic line image in the MASM procedure.

Note: *The MASM $END directive may contain an additional field, but PDP ignores it. See the* MASM Programming Reference Manual *for the correct form and use of this directive).*

MASM procedures may be nested, either statically or dynamically. Static nesting occurs when two or more $PROC directives appear in sequence without an intervening $END directive. Dynamic nesting occurs when a procedure reference image is part of the *source statements* section of another procedure.

The DEF directive and its *definitions* section may not be nested. If they appear within the *source statements* section of a MASM procedure, PDP ignores them.

PDP detects static nesting of MASM procedures by matching $PROC and $END directives. Each procedure declaration must have only one corresponding $END directive. Each $PROC directive increases the static nesting level and each $END decreases the static nesting level. PDP does not detect dynamic nesting.

Within the *source statements* section of a MASM procedure, PDP recognizes the definition of an internal name by using the MASM $NAME directive. The format for this directive is:

```
label       $NAME
```

where *label* is a MASM label field (as described in the *MASM Programming Reference Manual* and $NAME is the MASM directive.

Note: *The MASM $NAME directive may contain an additional field, but PDP ignores it.*

PDP only recognizes the $NAME directive if it appears within the *source statements* section of an unnested MASM procedure.

PDP uses the symbol subfield of the label field appearing on $NAME and $PROC directives as the procedure name/symbolic entry point entered into the Assembler procedure table. PDP uses the symbol as a procedure name only if the label field contains a dictionary control character (an asterisk) and the MASM$NAME or PROC directive on which the label appears does not occur within a nested MASM procedure.

MASM $FUNC definitions may appear in either the *definitions* or *source statements* sections. PDP only recognizes $FUNC definitions for the purpose of not confusing matching $PROC and $END pairs with $FUNC and $END pairs. PDP ignores all line images within the function text (in other words, between the $FUNC and $END directives). See the *MASM Programming Reference Manual* for the correct format of the $FUNC directive and additional information on functions.

The following two examples illustrate MASM procedures.

Example 1:

```
1.  @PDP,IM  MASM-PROC-1
2.                   DEF
3.  A                $EQU                24
4.  D                $EQUF               TAG,X2
5.  P                $PROC
6.  AA*              $NAME
7.                   $END
8.  @EOF
```

1. This is the PDP processor call with the I and M options.

   Note: *The M option is the default option and need not be specified.*

2. The DEF statement directs PDP to save the word address of the symbolic line following the DEF statement; in this case line 3.

3–4. These are values to be introduced into the MASM assembly.

5. The $PROC directive indicates the beginning of the MASM procedure associated with the DEF directive encountered.

6. The procedure symbolic entry point name is AA since the MASM label field on the $NAME directive contains a dictionary control character, and the $NAME directive does not appear within a nested MASM procedure.

7. The $END statement directs PDP to build an assembly procedure item with the start address pointing to line 3.

8. End-of-file control statement. This completes the input from the runstream and causes PDP to complete processing and terminate.

Example 2:

```
1.  @PDP,IM  MASM-PROC-2
2.  P*               $PROC
3.  A                $EQU                TAG
4.  PI               $PROC
5.  N**              $NAME
6.  B                $EQU                TAG
7.                   $END
8.                   $END
9.  @EOF
```

1. This is the PDP processor call with the I and M options.

2. The $PROC statement indicates the beginning of a MASM procedure. The MASM symbol P is saved as the procedure symbolic entry point name, since the MASM label field contains a dictionary control character. The word address of this line image is saved by PDP as the starting address of the MASM procedure, since no DEF directive and *definitions* section precede this image.

3. This is a line in procedure P.

4. The $PROC directive indicates the beginning of a nested procedure. PDP increases the static nesting level.

5. PDP ignores this line image even though the MASM label field contains a dictionary control character, since it appears within a nested procedure.

6. This is a line in procedure P1.

7. The $END statement directs PDP to decrease the static nesting level and to close out procedure P1.

8. The $END statement directs PDP to decrease the static neting level and to close out procedure P.

9. End-of-file control statement. This completes the input from the runstream and causes PDP to complete processing and terminate.

2.6. ASM Procedures

PDP only recognizes the following differences between ASM and MASM procedures:

- The MASM form of the directives (in other words, beginning with the '$') is not acceptable input to ASM.
- The format of the ASM label field is slightly different than the format of the MASM label field.

For additional information on ASM procedures and the ASM incompatibilities with MASM, see the *MASM Programming Reference Manual.*

Section 3
Symbolic Output

3.1. General

The symbolic output that PDP generates is accessible to users through other system processors such as @ELT, @ED, and @CTS.

This symbolic output consists of a line-by-line copy of the symbolic input to PDP (see Section 2 for more information) or an updated copy of the symbolic input to PDP.

The symbolic input may be updated by specifying a source output procedure element (*so* field) on the PDP control statement and entering correction images in the runstream following the PDP control statement, followed by and end-of-file (@EOF) or other control statement. The format of line-change statements used to apply corrections is described in the *DATA Processor and ELT Processor Operations Reference Manual*. An exact copy of the symbolic input to PDP may be obtained by specifying a source output procedure element (*so* field) on the PDP control statement and entering an end-of-file (@EOF) or other control statement in the runstream immediately following the PDP control statement.

The symbolic output is placed in a program file symbolic procedure element specified by the *so* (*si* if I option) field on the @PDP statement. The element cycle of this symbolic element is always zero (in other words, no "updating" of the symbolic input element to a higher element cycle may be performed using PDP).

Symbolic output is the same character code as the symbolic input, except when the symbolic input is in the Fieldata character set. In that case, the symbolic output is in the ASCII/ISO character set.

3.2. Procedure Table Output

The procedure table is not directly accessible to the user but is provided for access to the PDP procedures by the language processors. This table is accessed indirectly by using the Basic Service Package (BSP$). For additional information on BSP$, see the *SYSLIB Programming Reference Manual*.

PDP generates a procedure table entry for each valid procedure coded within the symbolic input. This entry contains information about the symbolic entry point to the PDP procedure. For additional information on procedure table entries, see the Basic Service Package (BSP$) in the *SYSLIB Programming Reference Manual*. All procedure table entries generated by one execution of PDP apply to the same procedure table (in other words, only one type of procedure table is updated per execution of PDP).

The updated procedure table resides in the Table of Contents (TOC) of the program file containing the symbolic input element, except when symbolic output is specified. In that case, the updated procedure table resides in the TOC of the program file containing the symbolic output procedure element.

If a procedure table does not exist in the program file for the procedure type being processed, PDP creates a procedure table in the program file.

Procedure names entered into the procedure tables are coded in the Fieldata character set.

If a user attempts to enter a procedure name that already exists in a procedure table, PDP will automatically delete the duplicate procedure name. This assures a unique procedure name for each procedure. However, duplicate procedure names may exist in different procedure tables. All PLUS and FORTRAN procedure names combined are considered unique because they are entered into the same table. This is also true for ASM and MASM.

3.3. Listing

PDP generates printed output (L option) that consists of the following information for every line image input:

- Field 1: Image Number

  This is a 9-digit number representing the integral line number associated with this line image, or the CTS line number associated with this image, or a 20-digit number representing the segmented line number associated with this image.

- Field 2: Nesting Level

  This field is a 2-digit number indicating the static nesting level of PDP procedures. This field does not appear if the static nesting level is zero.

A partial or error listing (E option) produces the same information as an L option but only displays the line images on which an error is detected.

A base and Permanent Correction File (PCF) listing (V option) consists of line-change statements, source input images, and correction images. Source input and correction images contain the following information:

- Field 1: Image Number

  This is a 10-digit number representing the integral line number associated with this line image, or the CTS line number associated with this image, or the segmented line number associated with this image.

- Field 2: Base/PCF Number

  This is a 9-digit number representing the integral base line number for source input images. For correction images, this is a 9-digit number representing the integral PCF line number associated with this image.

- Field 3: Nesting Level

  This field is a 2-digit number indicating the static nesting level of PDP procedures. This field does not appear if the static nesting level is zero.

The termination line image for PDP consists of two lines with the following format:

```
END PDP.  Errors:  f Fatal  s Syntax  w Warning
     Lines: i  Entry Points: e  Time: t  Storage: is/nm/fs/ha
```

where:

f

is the number of fatal errors detected by PDP.

s

is the number of syntax errors detected by PDP.

w

is the number of warnings detected by PDP.

i

is the number of line images input and processed by PDP.

e

is the number of procedure entry points created or updated in the appropriate procedure table by PDP.

t

is the SUPs used by this execution of PDP. This field has for the form: *mm:ss.msc*, where *mm* is minutes, *ss* is seconds, and *msc* is milliseconds. The minutes and seconds subfields do not appear if they are zero; the milliseconds subfield will always appear.

is

is the initial number of words in PDP's work area.

nm

is the number of ER MCORE$s performed during this execution of PDP.

fs

is the final number of words in PDP's work area.

ha

is the highest D-bank address (in octal) obtained by PDP.

3.4. Diagnostic Messages

PDP produces four types of messages: fatal errors, syntax errors, warnings, and informational messages. The messages are prefixed with either '*FATAL ERROR:', '*ERROR:', '*WARNING:', or '*NOTICE:'.

Fatal errors interrupt PDP processing, produce an error message, and terminate PDP with an error exit (ER ERR$). No procedure table entries are created or updated. Symbolic output may or may not be produced.

If a syntax error is found, a message is generated and analysis of the remaining symbolic input continues, but there is no procedure table produced or updated and no symbolic output produced (if specified).

Warning messages do not prevent normal output.

In general, the diagnostic messages include symbolic input scanning errors, preprocessor interface routine errors, Basic Service Package (BSP$) errors, Executive Request (ER) errors, and Symbolic Access Routine (SAR$) errors.

See Appendix A for a detailed description of each diagnostic message.

Section 4
Copying Procedures

When a PDP symbolic element is transferred to an element file (on tape) using the FURPUR statement @COPOUT (see the *FURPUR End Use Reference Manua*), the procedure table entries are carried along with the element. On an @COPIN statement of a PDP symbolic element, FURPUR puts the procedure table items of that element into the Table of Contents (TOC) of the program file.

When a PDP symbolic element is transferred from one program file to another program file using the FURPUR statements @COPY,P or @COPY,S, the procedure table entries are carried along with the element. Therefore, it is not necessary to reprocess a PDP symbolic element using PDP if the element has been copied into a program file using PDP if the element has been copied into a program file using the FURPUR statements @COPIN, @COPY,P, or @COPY,S. @COPY,SL (copy only latest cycle information) is not allowed.

If a symbolic procedure element is updated with the symbolic output in the same file as the symbolic input, procedure names that appear in the symbolic input but not in the symbolic output are not marked as deleted in the procedure table in the program file TOC, even though the element in which they appear is marked as deleted in the element table. An @PACK statement on the file following the PDP processing removes the deleted element along with the corresponding procedure table entries.

Appendix A
PDP Diagnostic Messages

A.1. Introduction

The diagnostic messages in this appendix are grouped into four categories: Fatal Error, Error, Warning, and Notice. There is a separate error count in PDP's termination line for each category of errors except Notice.

- Fatal Error

  This type of message is always printed by PDP. It causes the PDP to stop processing, thus producing no procedure table entries. Fatal error messages have the prefix '*FATAL ERROR:'.

- Error

  This type of message is always printed by PDP. Analysis of the remaining symbolic input continues, but there are no procedure table entries or symbolic output produced. If the X option is specified on the @PDP control statement, this type of diagnostic causes PDP to take an error exit. Error messages have the prefix '*ERROR:'.

- Warning

  Warning messages have the prefix '*WARNING:'.

- Notice

  This is not an error message, but a normal notification of an action taken. This type of message is only printed after an error-type message. Notice messages have the prefix '*NOTICE:'.

A.2. Messages

The following messages are listed in error-type order.

`*FATAL ERROR: BSP$ Add` *`Proc-type`* `Procedure Table Item error, return status:` *`status`*

Indicates an error occurred while adding an item to the procedure table input from mass storage. *Proc-type* indicates the specific procedure table. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

`*FATAL ERROR: BSP$ Delete` *`Proc-type`* `Procedure Table Item error, return status:` *`status`*

Indicates an error occurred while deleting an item from the procedure table input from mass storage. Proc-type indicates the specific procedure table. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

`*FATAL ERROR: BSP$ Read File Table Index error, return status:` *`status`*

An error occurred while reading the File Table Index from the program file's TOC on mass storage. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

`*FATAL ERROR: BSP$ Write File Table Index error, return status:` *`status`*

An error occurred while writing the File Table Index back to the program file's TOC on mass storage. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

`*FATAL ERROR: BSP$ Write` *`Proc-type`* `Procedure Table error, return status:` *`status`*

Indicates an error occurred while writing the procedure table back to the program file's Table of Contents (TOC) on mass storage. *Proc-type* indicates the specific procedure table. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

`*FATAL ERROR: Conflicting control options – I option implies symbolic output`

Indicates that both the T and I options were specified on the PDP control statement. The T option implies no symbolic output whereas the I option implies symbolic output—PDP cannot decide which is desired.

`*FATAL ERROR: Executive Request CLIST% error, unable to terminate CLIST% mode, possible next control statement loss`

Indicates an error occurred while attempting to terminate CLIST$ mode. The next 1100 control statement following the PDP control statement may be ignored. See the *ER Programming Reference Manual* and *ECL End Use and Programming Reference Manual* for additional information about CLIST$ mode.

`*FATAL ERROR:  Executive Request PFS$ error, return status: status`

Indicates an error occurred while searching for the symbolic I/O element in the program file's Table of Contents (TOC). See the *ER Programming Reference Manual* and *ECL End Use and Programming Reference Manual* for a description of the status bits.

`*FATAL ERROR:  Missing si field on PDP call statement`

Indicates the symbolic input element was not specified on the processor call or the symbolic output element was not specified on the processor call when the I option was specified. PDP must have a symbolic element to work with.

`*FATAL ERROR:  Procedure type option conflict`

Indicates two or more procedure type options were specified on the processor call statement. Only one type of procedure may be processed by PDP at any one time.

`*FATAL EROR:  SAR$ Element Add error, write status word: status`

Indicates an error occurred in adding the symbolic output element to the symbolic output file's Table of Contents (TOC). See the *SYSLIB Programming Reference Manual* for a description of the status bits.

`*FATAL ERROR:  SAR$ SAR$-func error, read status word: status`

Indicates an internal error or a corrupted file was encountered during SAR$ processing of the symbolic input element. *SAR$–func* specifies the SAR$ Input function during which the error occurred. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

`*FATAL ERROR:  SAR$ SAR$-func error, write status word: status`

Indicates an internal error or a corrupted file was encountered during SAR$ processing of the symbolic output element. *SAR$-unc* specifies the SAR$ Output function during which the error occurred. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

`*FATAL ERROR:  SAR$ SAR$-func error, write status word: status`

Indicates an internal error or a corrupted file was encountered during SAR$ processing of the standard print file. *SAR$–func* specifies the SAR$ Print function during which the error occurred. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

`*FATAL ERROR:  SAR$ SAR$-func I/O error, read status word: status`

An I/O error occurred during SAR$ processing of the symbolic input element. *SAR$–func* specifies the SAR$ Input function during which the error occurred. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

```
*FATAL ERROR:  SAR$ SAR$–func I/O error, write status word: status
```

An I/O error occurred during SAR$ processing of the symbolic output element. *SAR$–func* specifies the SAR$ Output function during which the error occurred. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

```
*FATAL ERROR:  SAR$ SAR$–func I/O error, write status word: status
```

An I/O error occurred during SAR$ processing of the standard print file. SAR$–func specifies the SAR$ Print function during which the error occurred. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

```
*FATAL ERROR:  SAR$ Open Input error – Bad Label Control Record, read status
word: status
```

An invalid SDF file or element label control record was read in the symbolic input *file.element*. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

```
*FATAL ERROR:  SAR$ Open Input error – File not SDF, read status word:  status
```

The symbolic input file is not in Standard Data Format (SDF). See the *SYSLIB Programming Reference Manual* for a description of the status bits.

```
*FATAL ERROR:  SAR$ Print error – Invalid Attribute Type or Value, write status
word: status
```

Indicates an internal error or the image read contained an invalid attribute type or an invalid value for the attribute type. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

```
*FATAL ERROR:  SAR$ Read ACD or Shift-Code Format error, read status word:
status
```

The image read from the symbolic input element contains incorrectly formatted ACD or incorrectly formatted embedded shift-coded Kanji. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

```
*FATAL ERROR:  SAR$ Read error – Invalid Control Record, read status word:
status
```

An invalid SDF control record was encountered while reading the symbolic input element. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

```
*FATAL ERROR:  SAR$ Read error–SDF Image Exceeds Maximum Length, read status
word: status
```

The SDF data record or control record read from the symbolic input element exceeds the maximum allowed length of an SDF record. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

```
*FATAL ERROR:  SAR$ Write error–Invalid Attribute Type or Value, write status
word: status
```

Indicates an internal error or the image read contained an invalid attribute type or an invalid value for the attribute type. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

```
*FATAL ERROR:  Symbolic input element type does not match procedure type
specified
```

Indicates that, with the T option specified, the type or subtype of the element specified in the *si* field of the PDP control statement does not match the type or subtype specified by the procedure-type option of the PDP control statement. These must match for the appropriate procedure table to be updated and for the procedure table entries to remain when the file is packed (@PACK).

```
*FATAL ERROR:  SYSLIB Get Scratch File Routine error,return status:  status
```

Indicates an error occurred while obtaining a processor scratch file in which to place the "corrected" symbolic. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

```
*FATAL ERROR:  SYSLIB Identification Line ID$ routine error, return status:
status
```

Indicates an error occurred during formation or output of the PDP processor identification line. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

```
*FATAL ERROR:  SYSLIB Postprocessor routine POSTPR$ error
```

Indicates an error occurred during restoration of the Input/Output (I/O) files to their original assign state. See the *SYSLIB Programming Reference Manual* for additional information on this error.

```
*FATAL ERROR:  SYSLIB PREPRM$ Preprocessor routine error
```

Indicates an error occurred during interpretation of the processor call line or during the assign of the I/O files on the initial entrance to PDP. The message preceding this message provides additional information about this error.

```
*FATAL ERROR:  SYSLIB REPRM$ routine error
```

Indicates an error occurred during interpretation of the processor call line or during the assign of the I/O files on the reusable entrance to PDP. The message preceding this message provides additional information about this error.

```
*FATAL ERROR:  SYSLIB SIR$ Correction Image Sequence or partial-line-change
error
```

SIR$ detected a line-change statement out of sequence or a partial-line-change syntax error. See the *SYSLIB Programming Reference Manual* for additional information about this error.

`*FATAL ERROR:  SYSLIB SIR$ I/O error, return status:  status`

Indicates an I/O error occurred while processing the symbolic input sith SIR$. See the *SYSLIB Programming Reference Manual* for a description of the status bits.

`*FATAL ERROR:  SYSLIB SIR$ Invalid SDF label error`

Indicates SIR$ detected an invalid SDF label record. See the *SYSLIB Programming Reference Manual* for additional information about this error.

`*ERROR:  Attributed character data not allowed in procedure symbolic entry point name`

Attributed character data was encountered in the procedure symbolic entry point name field. The format of procedure table entries do not allow for attributed character data.

`*ERROR:  COBOL procedure symbolic entry point name greater than thirty characters`

COBOL procedure symbolic entry point names are limited by the size of two procedure table entries to a maximum of thirty characters.

`*ERROR:  COBOL procedure symbolic entry point name must contain at least one letter`

COBOL requires that at least one character of the procedure symbolic entry point name be an alphabetic character (in other words, 'A',...,'Z' or 'a',...,'z'). See the *ASCII COBOL Programming Reference Manual* for additional information.

`*ERROR:  Directives with $ prefix not recognized in ASM – line image ignored`

The directive encountered on this line image began with the dollar sign character ($). This form of directive is not acceptable input to ASM.

`*ERROR:  Illegal character encountered in source–lang procedure symbolic entry point name`

A character not allowed in this source language PDP procedure symbolic entry point name was encountered. *Source–lang* specifies the source language type. See Section 2 for a description of the characters allowed in the procedure symbolic entry point name.

`*ERROR:  Illegal first character for COBOL procedure symbolic entry point name`

COBOL does not allow the minus character (i.e., '–') to be the first character of a COBOL PDP procedure symbolic entry point name. See the *ASCII COBOL Programming Reference Manual* for additional information.

```
*ERROR:  Illegal last character for COBOL procedure symbolic entry point name
```

COBOL does not allow the minus character (i.e., '–') to be the last character of a COBOL PDP procedure symbolic entry point name. See the *ASCII COBOL Programming Reference Manual* for additional information.

```
*ERROR:  Label field on a DEF directive is illegal – line image ignored
```

PDP does not allow a MASM label field on a PDP DEF directive line image.

```
*ERROR:  MASM $DEF (or DEF) directive is not allowed – line image ignored
```

PDP does not recognize the MASM form of this directive. This also indicates a MASM source line precedes a PDP DEF or MASM $PROC declaration.

```
*ERROR:  MASM procedure static nesting level exceeds 63 – procedure declaration
ignored
```

The number of MASM procedure declarations encountered in sequence without an intervening MASM end declaration is greater than 63. MASM does not allow more than 63 statically nested procedures. See the *MASM Programming Reference Manual* for additional information.

```
*ERROR:  Misplaced procedure declaration or source line encountered before
procedure declaration – line image ignored
```

PDP encountered a line image, beginning in column two, that is not a comment, preceding a valid PDP procedure declaration.

```
*ERROR:  Missing function end declaration and missing procedure or procedure end
declaration
```

The end of the symbolic input element was encountered before a line image containing a MASM function $END declaration. Also, this indicates that the line image (i.e., definitions or procedure text) where this function was declared is incomplete.

```
*ERROR:  Missing MASM directive field or source line encountered before
declaration – line image ignored
```

A line image with only a MASM label field was encountered before a PDP DEF or MASM $PROC declaration. MASM source lines must only appear within the procedure text.

```
*ERROR:  Missing PROC directive or unrecognizable characters in the symbolic
entry point name field – line image ignored
```

PDP encountered characters in the symbolic entry point name field of this line image but no recognizable characters followed this field. This may also indicate that a source line precedes the PDP procedure declaration.

```
*ERROR:  Missing procedure end declaration on nested procedures – procedures
ignored
```

The end of the symbolic input element was encountered before the PDP end declarations matching the PDP nested procedure declarations were encountered.

```
*ERROR:  Missing symbolic entry point name field on PDP procedure declaration
```

PDP encountered a PLUS PDP procedure declaration directive, but no characters followed it specifying the symbolic entry point name.

```
*ERROR:  Misspelled PDP directive or unrecognizable characters in PDP directive
field – line image ignored
```

PDP encountered the special character in column one of the line image indicating a PLUS PDP procedure declaration, but the characters that followed the special character were not recognizable to PDP.

```
*ERROR:  Misspelled/unrecognizable directive in PDP directive field or
unrecognizable charactes in the symbolic entry point name field – line image
ignored
```

PDP encountered a symbolic entry point name field and a PDP directive field, but the PDP directive field contained characters not recognized by PDP. This may also indicate a source line precedes the PDP procedure declaration.

*ERROR: Nested PDP procedures not allowed in the *source–lang* source language – procedure declaration ignored

A PDP procedure declaration was encountered following a previous PDP procedure declaration without an intervening PDP end declaration. *Source–lang* specifies the source language that does not allow nested PDP procedures.

```
*ERROR:  Nested procedure end encountered – nested procedure ignored
```

A PDP end declaration matching the PDP procedure declaration for a nested PDP procedure was encountered in a PDP procedure that does not allow nested PDP procedures.

```
*ERROR:  No PDP procedure declarations encountered in the symbolic input – No
symbolic entry points produced
```

The end of the symbolic input element was encountered and no PDP procedure declarations were encountered. Check the PDP procedure declaration format for the source language desired. The error messages preceding this message will provide additional information.

```
*ERROR:  PDP Procedure end declaration encountered before PDP procedure
declaration – line image ignored
```

A PDP end declaration was encountered before a valid PDP procedure declaration was encountered.

```
*ERROR:  PLUS PDP procedure static nesting level exceeds ten – procedure
declaration ignored
```

The number of PLUS PDP procedure declarations encountered in sequence without an intervening PLUS PDP end declaration is greater than 10. PLUS does not allow more than 10 physically nested PDP procedures. See the *UCS PLUS Programming Reference Manual Volume 1* and *Volume 2* for additional information.

```
*ERROR:  Procedure declaration encountered before function end – procedure
declaration ignored
```

A MASM $PROC declaration was encountered before the function's matching $END declaration was encountered. This indicates a procedure declaration nested within a function—which is ignored by PDP.

*ERROR: *source–lang* procedure symbolic entry point name greater than twelve characters

Procedure symbolic entry point names are limited, by the size of the procedure table entries, to a maximum of 12 characters, except for COBOL. *Source–lang* specifies the source language type.

```
*ERROR:  Source line encountered before procedure declaration – line image
ignored
```

A no-comment, unrecognizable line image was encountered before a valid PDP procedure declaration was encountered. Source language line images may only appear within the procedure text.

```
*ERROR:  Source line encountered between DEF directive and procedure declaration
– line image ignored
```

A line image, not containing one of the following MASM directives, was encountered following a PDP DEF declaration but preceding the subsequent MASM $PROC declaration: $EQU, $EQUF, $FORM, or $FUNC. PDP does not allow MASM source lines in the definitions section.

```
*ERROR:  Source line with sequence number encountered before procedure
declaration – line image ignored
```

A no-comment, unrecognizable line image, with a sequence number in the beginning of the image, was encountered before a valid PDP procedure declaration was encountered. Source language line images may only appear within the procedure text.

*WARNING: CLIST$ failure to register PDP control statement with executive, return status: *status*

This indicates that the Executive Request to register the @PDP statement with the EXEC failed. See the *ER Programming Reference Manual* and the *ECL End Use and Programming Reference Manual* for a description of the status bits.

```
*WARNING:  Correction images may not be applied when input is from the runstream
– V option ignored
```

Both the I and V options were specified in the PDP control statement. When the symbolic input is from the runstream, correction images to update the symbolic input may not also appear in the runstream. Therefore, listing base and PCF line numbers is meaningless and will be ignored.

```
*WARNING:  Fieldata output is not allowed – ASCII output assumed
```

The PDP control statement contains the P option. PDP converts all Fieldata-coded input to ASCII-coded output and is not designed to generate any Fieldata-coded output.

```
*WARNING:  Final procedure missing end declaration – END assumed
```

The end of the symbolic input element was encountered before the PDP end declaration matching the last PDP procedure declaration. PDP assumes a PDP end declaration and completes processing.

```
*WARNING:  Line change images may not be applied when input is from the
runstream – W option ignored
```

The PDP control statement contains both the I and W options. When the symbolic input is from the runstream, correction images to update the symbolic input may not also appear in the runstream. Therefore, listing line-change statements and change images is meaningless and will be ignored.

```
*WARNING:  Listing type option conflict – no listing assumed
```

Two or more conflicting listing options have been specified on the @PDP statement. The conflict is ignored and no listing is assumed.

```
*WARNING:  Microstring in externalized MASM label, no substitution performed –
externalized label ignored
```

PDP encountered a microstring in the symbol subfield of an externalized label. Because PDP does not do any microstring substitition, the actual symbol cannot be determined and therefore the label field is ignored.

```
*WARNING:  No external labels encountered in the symbolic input – no procedure
symbolic entry points produced
```

PDP has scanned the entire MASM/ASM symbolic input and has not encountered any dictionary control characters appearing on the label fields attached to non-nested PROC or NAME directives. PDP will not create or update the Assembler procedure table.

```
*WARNING:  No procedure type option specified – MASM procedure type assumed
```

No procedure-type option was specified on the @PDP statement; therefore PDP assumes that the symbolic input contains MASM procedures.

```
*WARNING:  Not defined or unsupported option specified – option ignored
```

An A option was specified on the @PDP statement, which PDP does not recognize or support. See 1.2 for the format of the @PDP statement.

```
*WARNING:  S option not recognized – L option assumed
```

PDP does not have a “short” listing; therefore, the standard listing is assumed when the S option is specified on the PDP call statement.

```
*WARNING:  Symbolic output not allowed with T option – so field ignored
```

The *so* field was specified in the PDP control statement along with the T option. The T option implies that no symbolic output is generated; therefore the *so* field is meaningless and will be ignored.

```
*WARNING:  V option print is only meaningful with symbolic output – V option
ignored
```

Both the T and V options or the V option and no *so* field are specified in the PDP control statement. Correction images are applied only when the symbolic input is updated, creating symbolic output. Therefore, listing base and PCF line numbers is meaningless and will be ignored.

```
*NOTICE:  No symbolic entry point produced
```

This procedure that no procedure symbolic entry point information was saved for this PDP procedure declaration.

```
*NOTICE:  PDP may not be reused
```

This indicates that an error occurred that caused PDP to not be reusable. See 1.4 for additional information.

Appendix B
Restrictions

The following restrictions exist with PDP level 13R1:

- W Option Output

  Correction images containing Attributed Character Data (ACD) or embedded shift-coded Kanji characters will not appear correctly on the listing. SIR$ does not allow printing of images in character sets other than Fieldata or ASCII.

- COBOL Procedure Names

  PDP allows the dollar sign character ($) in COBOL procedure names, even though the COBOL compiler does not recognize this character. This restriction exists because DMS 100 COBOL procedure names are padded with the dollar sign ($) character.

- MASM Procedures

  Line levelers and microstrings are ignored by PDP. No microstring substitution or line leveling is performed by PDP. In addition, multiple symbols in MASM label fields are not handled by PDP. It is recommended that MASM procedure processing with PDP be replaced by MASM definition mode assemblies.

- T Option Output

  Procedure table entries deleted by T option processing will not be eliminated when the file is packed (@PACK). FURPUR only eliminates procedure table entries associated with a deleted procedure element. This may cause the procedure table to be full when most of the entries are deleted.

- Segmented Line Numbers

  Segmented line numbers printed in the listing reflect, at most, 10 digits of the fractional part of the line number. On a V option listing, no digits of the fractional part of a segmented line number are printed in the listing.

- FORTRAN Procedures

  FORTRAN procedures generated by PDP will not be recognized by the FORTRAN V compiler. The FORTRAN V compiler only recognizes Fieldata-coded FORTRAN procedures and PDP cannot generate Fieldata-coded procedures.

- Partial-Line Changes

  Partial-line changes will not work on line images containing Attributed Character Data (ACD) or embedded shift-coded Kanji data. Only whole-line changes may be applied to symbolic input containing ACD or embedded shift-coded Kanji data.

- Asterisk on COBOL/FORTRAN Procedure Names

  PDP allows the asterisk character to be appended to COBOL/FORTRAN procedure names even though the compilers do not recognize this character in procedure

names. PDP will remove the asterisk character from the procedure name before the name is entered into the appropriate procedure table and generate a warning message. In addition, this will specifically allow the COBOL procedure name field to contain 31 characters (including the asterisk) and the FORTRAN procedure name field to contain 13 characters (including the asterisk). This restriction exists because DPS 1100 erroneously appends the asterisk character to the COBOL procedure name it generates.