the engineering and scientific software series

PDL/74

Program Design Language

Reference Guide

(Processor Version 3)

Caine, Farber &
Gordon, Inc.

Warren Point
International Limited

RESTRICTED RIGHTS LEGEND

Comments or questions relating to this manual or to the subject software are welcomed and should be addressed to:

In North America:

Caine, Farber & Gordon, Inc.
1010 East Union Street
Pasadena, CA 91106
USA

Tel: (818) 449-3070
Telex: 295316 CFG UR

In the Rest of the World:

Warren Point International Limited
Babbage Road, Stevenage
Hertfordshire SG1 2EQ
ENGLAND

Tel: Stevenage (0438) 316311
Telex: 826255 DBDS G

Order Number: 9003-2(4)

25 September 1983

CONTENTS

APPENDICES

1. INTRODUCTION

PDL (Program Design Language) is a Licensed Program of Caine, Farber & Gordon, Inc (CFG). It is intended as an aid in the systematic and reliable design of software. PDL is not a code generator and executable programs are not automatically generated from PDL designs.

The input to the PDL processor consists of designs in a source language which may be termed "structured English". The output consists of a design document, much of which is automatically formatted. The components of a design document are:

1. Cover page: Identifies the document.
2. Table of Contents: Automatically generated from the input.
3. Design Body: Contains the actual design information.
4. Segment Reference Trees: Illustrates, by indented listings, the relationships among the procedural portions of the design.
5. Data Index: Lists the data items declared in the design and shows where each is used.
6. Flow Segment Index: Lists the "procedures" of the design and shows where each is used.

The design body is composed of a number of segments. There are four different segment types:

- Text Segments: These contain general descriptive text.
- Data Segments: These declare the names of data items which are to be collected in the Data Index.
- Flow Segments: Each of these represents a "procedure" of the design. Flow segments are automatically formatted based on the use of keywords. The standard keywords are:

```
IF      ELSEIF     ELSE      ENDIF
DO      UNDO       CYCLE     ENDDO
RETURN
```

  Keywords are highlighted in the design document if the printer is capable of processing the selected highlighting mode. References to other flow segments are automatically detected and indicated.

- External Segments: These are used to define "procedures" which are considered to be defined elsewhere than in the design document.

A number of installation options are available for tailoring the default behavior of PDL to the particular customer requirements. During installation, new keywords may be defined. Procedures for making these changes are presented in the appropriate installation guide.

2. GENERAL INFORMATION

This chapter discusses various general aspects of the PDL processor. It includes information on the form of a design and on the syntax of PDL commands.

2.1 FORMAT OF A DESIGN

The PDL processor accepts as input a series of source lines and produces a design document. A sample design appears at the back of this manual following the colored separator page.

The design document is composed of several major sections:

1. Front matter

2. Design body

3. Reports

4. Final page

which are now briefly described.

2.1.1 Front Matter

This is the first part of the design document. It begins with a title page which identifies the particular design. Primary information for this page comes from the title command (see Section 8.1) and the date command (see Section 8.2).

The title page is followed by the table of contents for the design. The table of contents lists all of the sections and subsections which make up the design along with their corresponding page numbers.

2.1.2 Design Body

The design body presents the actual data definitions, procedure definitions, and textual information of the design. This section is composed of various kinds of segments which may be structured into groups (see Chapter 3). The segment types are:

- Text Segments: which represent arbitrary commentary (see Chapter 4).

- Data Segments: which allow explicit definition of data items (see Section 5.3).

- Flow Segments: which represent the procedural flow of the design (see Chapter 6).
- External Segments: which allow declaration of procedures which are assumed to be defined somewhere outside of this design document (see Chapter 7).

2.1.3 Reports

The processor can be instructed to produce several reports (see Chapter 9) which provide information about the content and internal structure of the design. These reports are particularly useful in understanding a design. The possible reports are:

- Reference Trees: which shows all of the flow segments arranged in the form of a calling tree. There will be several trees if there are several flow roots in the design. Reference trees are further described in Section 9.1.
- Data Index: which lists each data item in alphabetic order and shows the points in the design where each is referenced. The data index is further described in Section 9.2.
- Flow Segment Index: which lists each flow segment in alphabetic order and shows the points in the design where each is referenced. The flow segment index is further described in Section 9.3.

2.1.4 Final Page

This is the last page of the design document. Besides confirming that the design was completely processed, this page displays a number of statistics about the processing.

2.2 OVERALL OPERATION

PDL processes a design in two passes. During the first pass, the source is read and written onto a scratch file in encoded form, page breaks are determined, and a dictionary of data item and segment names is constructed. During the second pass, the encoded source is read, references to data items and segments are detected, and the design document is formatted.

2.3 INPUT FORMAT

Input to the PDL processor consists of a sequence of source lines. The maximum length of a line is 80 characters.

Except for certain versions of the PDL processor (see Appendix B), all input must be in upper case and no control characters (e.g, tabs) may be used.

2.4 COMMAND LINES

If the first character of a line is a "%", the line is known as a command line. Command lines contain commands which direct various types of processing or provide various information to the processor.

The "%" must be immediately followed by a command name which extends to the first blank. After skipping any white space, the remainder, if any, of the line is considered to be the command argument. Thus, for example.

```
%TITLE    THIS IS A SAMPLE
```

is a command line with a command name of "TITLE" and a command argument of "THIS IS A SAMPLE".

2.5 DESIGN BODY CONVENTIONS

As outlined in Section 2.1.2, the design body is composed of a number of segments. There are no restrictions on the ordering of segments. The only restriction on the number of segments is that imposed by the amount of memory available to the PDL processor while processing a design.

2.5.1 Segment Delimiting

A segment is introduced by one of the segment commands described elsewhere in this manual. These commands are:

%TEXT or %T	start a text segment (Chapter 4)
%DATA or %D	start a data segment (Section 5.3)
%SEGMENT or %S	start a flow segment (Chapter 6)
%EXTERNAL or %E	start an external segment (Chapter 7)

A segment is terminated by the next occurrence of command or the end of the design source.

2.5.2 Display of Segments

Except for the versions of the PDL processor described in Appendix B, the maximum size of a segment is limited to that which can fit on one page of listing. This implies a limit of about 44 lines of listed output for the body of a segment.

Each segment will be enclosed in a box composed of characters specific to the type of segment. The various characters are:

#	text segment
D	data segment
*	flow segment
X	external segment

If the body of a segment is empty, the box will contain a generated notice that the segment was intentionally left blank.

2.5.3 Comment Characters

The description of data segments (Section 5.3) and of flow segments (Chapter 6) will refer to syntactic constructs known as comment characters which are used as delimiters in certain contexts (e.g., to separate a procedure name from its "arguments").

The two comment characters are the dot (".") and the left parenthesis ("(").

3. GROUPS

A design may be broken into various sections by use of commands of the form

```
+------------------------------------------------------------+
|                                                            |
| %GROUP   text                                              |
|                                                            |
+------------------------------------------------------------+
```

or

```
+------------------------------------------------------------+
|                                                            |
| %G   text                                                  |
|                                                            |
+------------------------------------------------------------+
```

where text is any sequence of characters to be used as the title for the group. In the design document, each group will be prefaced with a page containing the title of the group centered and boxed. The title will also appear as a subtitle on each design page within the group and will be placed in the table of contents for the design.

A group is terminated by the next "GROUP" or "G" command or by the end of the design.

Examples of group declarations are

```
%GROUP   PASS ONE PROCESSING

%G   INPUT EDITING PHASE
```

4. TEXT SEGMENTS

Text segments are used to place blocks of commentary into a design. They are frequently used to supply such material as introductory information, table layouts, and record layouts.

A text segment is introduced by the command

```
+------------------------------------------------------------+
|                                                            |
| %TEXT   text                                               |
|                                                            |
+------------------------------------------------------------+
```

or

```
+------------------------------------------------------------+
|                                                            |
| %T   text                                                  |
|                                                            |
+------------------------------------------------------------+
```

where text is any sequence of characters to be used as the title of the segment. The title will be displayed at the top of the segment page and will be entered in the table of contents.

The lines comprising the body of a text segment are simply input and printed as is. No automatic formatting will be performed. White space on input lines is kept and blank input lines will result in blank output lines.

Examples of commands to introduce a text segment are:

```
%TEXT   INTRODUCTION TO POSITION MONITORING MODULE

%T   OTHER DOCUMENTS RELATING TO THIS SUBSYSTEM
```

5. DATA ITEM DECLARATION

PDL allows the designer to declare certain items known as *data items*. References to these items within flow segments will be collected and may be displayed in the data item index (see Section 9.2).

5.1 DATA ITEMS

Within data segments (see Section 5.3) and flow segments (see Chapter 6), tokens consisting of letters, digits, and certain special characters are considered to be *potential data items*. A potential data item will be considered to be an actual data item if it is defined as such in an implicit (see Section 5.2) or explicit (see Section 5.3) data declaration.

Lines which begin (possibly after some leading white space) with a comment string (see Section 2.5.3) will not be examined for potential data items.

The special characters which may be part of a potential data item are "$", "#", "@", and "_". Thus, in the line

```
X = A$1+BB*CC
```

the potential data items are

```
X   A$1   BB   CC
```

5.2 IMPLICIT DATA ITEM DECLARATION

When a potential data item is encountered in a flow segment, it will be declared as an implicit data item if

1. it contains a *data character*;

2. it is longer than one character; and

3. it is not declared elsewhere in the design as an explicit data item.

Initially, the underscore ("_") is the data character. The data character may be redefined by the command

```
+------------------------------------------------------------+
|                                                            |
| %DATACHR  char                                             |
|                                                            |
+------------------------------------------------------------+
```

where *char* is a non-blank, non-alphanumeric character. If *char* is absent, there will be no data character and, thus, no implicit data definition. Any use of this command should precede the first segment.

For example, the "-" may be defined as the data character by

```
%DATACHR -
```

5.3 EXPLICIT DATA ITEM DECLARATION (DATA SEGMENTS)

Data items are explicitly defined in *data segments*. A data segment is introduced by the command

```
+---------------------------------------------------------------+
!                                                               !
! %DATA   text                                                  !
!                                                               !
+---------------------------------------------------------------+
```

or

```
+---------------------------------------------------------------+
!                                                               !
! %D   text                                                     !
!                                                               !
+---------------------------------------------------------------+
```

where *text* is a sequence of characters to be used as the title of the data segment. The title will be displayed at the top of the segment page and will be entered in the table of contents.

```
********
* NOTE *
********
```

The "DATA" or "D" commands do *not*, themselves, declare data items -- they *introduce* segments *in which* data items are declared.

Examples of these commands are:

```
%DATA   FORMATS FOR MASTER FILE RECORDS

%D   MISCELLANEOUS DATA DEFINITIONS
```

The actual data definitions occur in the *body* of a data segment. Such a body is composed of one or more *statements*. Each statement consists of one or more lines with all but the last line of a statement having a '/' as the last non-blank character.

Before a statement is printed, leading and trailing blanks are removed, each sequence of imbedded blanks is replaced by a single blank, and each '/' used as a continuation character is replaced by a single blank. Statements which are too wide to fit in the segment box will be automatically continued when printed. Blank statements are ignored.

The first potential data item in each statement of the body is declared to be an actual data item. Anything following the data item is taken as commentary. Thus, in the line

```
    CTYPE IS THE TYPE OF THE COMMAND
```

"CTYPE" will be declared to be a data item.

Statements beginning with a comment string (see Section 2.5.3) are considered to be comments and are not scanned for declarations.

6. FLOW SEGMENTS

A *flow segment* presents, in a program-like form, the procedural flow of a portion of a design. Generally, each flow segment represents a *procedure* in the program. A flow segment is introduced by the command

```
+------------------------------------------------------------+
|                                                            |
| %SEGMENT   text                                            |
|                                                            |
+------------------------------------------------------------+
```

or

```
+------------------------------------------------------------+
|                                                            |
| %S   text                                                  |
|                                                            |
+------------------------------------------------------------+
```

where *text* is a sequence of characters which represents the name of the segment. The name will appear at the top of the segment page. That portion of the name up to the first comment character (see Section 2.5.3) will be placed in the table of contents and will be saved in a dictionary for indexing purposes. In saving the name in the dictionary, leading and trailing blanks are removed and each sequence of imbedded blanks is collapsed into a single blank. Some examples are:

Command	Saved Name
%SEGMENT SYSTEM START	"SYSTEM START"
%S INSTALL IN DATA BASE (NAME, TYPE)	"INSTALL IN DATA BASE"
%SEGMENT SEARCH DICTIONARY (NAME)	"SEARCH DICTIONARY"
%S CREATE FILE (NEWFILE)	"CREATE FILE"

6.1 FLOW SEGMENT BODY

The body of a flow segment is composed of one or more *statements*. Each statement consists of one or more lines with all but the last line of a statement having a '/' as the last non-blank character.

The PDL processor will format and indent each statement on output and may supply various forms of visual enhancement to the printed line.

Before a statement is printed, leading and trailing blanks are removed, each sequence of imbedded blanks is replaced by a single blank, and each '/' used as a continuation character is replaced by a single blank. Statements which are too wide to fit in the segment box will be automatically continued when printed. Blank statements are ignored.

This automatic formatting means that there is no need for the designer to do any special formatting of the flow segment input lines. In fact, each statement is normally just typed flush left on the input line and layout is left to PDL.

With the exception of the special statements discussed in Section 6.5, the contents of a statement may be anything desired. Some examples are:

```
COUNT = COUNT +1

INCREMENT COUNT

BUMP COUNT TO REFLECT/
THE RECORD JUST PROCESSED
```

Note that, since '/' is the continuation character, the last two lines of this example are equivalent to:

```
BUMP COUNT TO REFLECT THE RECORD JUST PROCESSED
```

6.2 REFERENCE RECOGNITION

Each statement in a flow segment, except for a statement which begins with a comment character (see Section 2.5.3), will be scanned to see if it is the name of a flow segment. If a statement begins with a keyword (see Section 6.5), the scan begins following the keyword. The scanning stops at the first comment character.

```
********
* NOTE *
********
```

To be recognized as the name of a flow segment, the entire reference must be contained on the first source line of a statement.

In any match, leading and trailing blanks are removed, and each sequence of imbedded blanks is replaced by a single blank.

6.3 LABELS

It is occasionally desirable to place labels in a design. They are convenient for denoting statement sequences in DO CASE constructs (see Section 6.5.3.7) and in supplying names for DO constructs for use with UNDO and CYCLE statements (see Section 6.5.3.3 and Section 6.5.3.4).

If a colon (":") is encountered before the first blank in a statement, that statement is considered to be a label. The statement will be printed left adjusted on the output. Anything following the colon on the same line will be treated as commentary. Some examples of labels are:

```
MAIN_SEARCH_LOOP:
END_OF_FILE:
+,-,*:
+:
"OTHER":
```

6.4 DATA ITEMS IN COMMENT LINES

Normally, comment lines (those beginning with a left parenthesis or a period) are not scanned for the occurrence of implicit or explicit data items. Scanning of these lines may be requested by the command

```
+-------------------------------------------------------------+
|                                                             |
| %CDATA                                                      |
|                                                             |
+-------------------------------------------------------------+
```

When the processor is installed, the CDATA option may be established as the default. If this is done, scanning of comment lines may be prevented by the command

```
+-------------------------------------------------------------+
|                                                             |
| %NOCDATA                                                    |
|                                                             |
+-------------------------------------------------------------+
```

6.5 SPECIAL STATEMENTS

The so-called special statements comprise the flow-of-control statements in the PDL procedural language. This section describes each of the special statements.

6.5.1 Keywords

Each special statement begins with a keyword followed by a blank or the end of the input line. The keywords, grouped generally as used, are:

```
IF        ELSEIF       ELSE       ENDIF
DO        UNDO         CYCLE      ENDDO      ENDO
RETURN
```

The particular keyword which starts a statement determines the indentation level for that and subsequent statements. A word is considered to be a keyword only when it is the first word of a statement.

The keywords discussed above are those defined in the the distributed version of the PDL processor. New project-wide keywords may be added by modifying the processor as described in the appropriate installation guide.

****
* NOTE *
****

> The PDL processor uses the keywords only to control the formatting of the design output. The processor will not detect misused or mismatched keywords.

6.5.1.1 Keyword Enhancement. The form in which a keyword is printed depends on the particular version of the PDL processor. For most versions, keywords are enhanced by being underscored.

6.5.2 The IF Construct

The IF construct consists of the keywords IF, ELSEIF, ELSE, and ENDIF. In its simplest form, it can be written as:

```
IF condition
    sequence
ENDIF
```

which implies that the statements comprising "sequence" are only to be executed if "condition" is true.

The basic form can be expanded by adding an alternate as in:

```
IF condition
    sequence-1
ELSE
    sequence-2
ENDIF
```

which implies that "sequence-1" is to be executed if "condition" is true and that "sequence-2" is to be executed if "condition" is false.

Multiple IF constructs can be nested as in:

```
        IF condition-1
            sequence-1
        ELSE
            IF condition-2
                sequence-2
            ELSE
                sequence-3
            ENDIF
        ENDIF
```

Since nested IF constructs are quite common, an alternate form can be used as in:

```
        IF condition-1
            sequence-1
        ELSEIF condition-2
            sequence-2
        ELSE
            sequence-3
        ENDIF
```

Thus, the general form of the IF construct is

1. an IF
2. zero or more ELSEIF's
3. zero or one ELSE
4. an ENDIF

6.5.3 The DO Construct

The DO construct consists of the keywords DO, ENDDO, CYCLE, and UNDO. The word "ENDO" is considered an alternate spelling of "ENDDO". The DO construct is used to produce flow figures of the general form:

```
        DO iteration or selection criteria
            statement
            statement
            ...
        ENDDO
```

The iteration or selection criteria may be arbitrarily chosen. The remainder of this section presents examples of the more commonly used criteria.

6.5.3.1 The DO WHILE Construct. This form of the DO construct implies iteration as long as some given condition remains true. No iteration at all would be performed if the condition is initially false. It can be written as:

```
DO WHILE condition
    statement
    statement
    ...
ENDDO
```

Some examples of possible conditions are:

```
DO WHILE THERE IS SOURCE INPUT REMAINING
DO WHILE THE CURRENT CHARACTER IS A SPACE
DO WHILE THERE IS ROOM IN THE TABLE
```

6.5.3.2 The DO UNTIL Construct. This construct implies iteration until some condition becomes true and, further, implies that at least one iteration will always be performed. It can be written as:

```
DO UNTIL condition
    statement
    statement
    ...
ENDDO
```

Some examples are:

```
DO UNTIL TABLE IS FULL
DO UNTIL LAST RECORD IS READ
DO UNTIL SOURCE IS DEPLETED
```

6.5.3.3 The UNDO Statement. The UNDO statement is used to indicate that control should pass to the statement immediately following the ENDDO of the current DO construct, thus causing premature exit from the loop. It might be used in the following context:

```
DO WHILE SOURCE INPUT REMAINS
    PROCESS NEXT SOURCE LINE
    IF DYNAMIC MEMORY IS FULL
        UNDO
    ENDIF
ENDDO
```

An alternate form of the UNDO statement is:

```
UNDO IF condition
```

which can make the design more concise as in:

```
DO WHILE SOURCE INPUT REMAINS
    PROCESS NEXT SOURCE LINE
    UNDO IF DYNAMIC MEMORY IS EXHAUSTED
ENDDO
```

When DO constructs are nested, it may sometimes be necessary to indicate a premature exit from an outer loop. This is most easily shown by labelling the outer DO and writing

```
UNDO label
```

6.5.3.4 The CYCLE Statement. The CYCLE statement indicates premature transfer of control to the loop test or selection point -- that is, to a point just before the ENDDO statement which terminates the loop. The most commonly used forms of the statement are:

```
CYCLE
CYCLE IF condition
CYCLE label
```

6.5.3.5 The DO FOREVER Construct. The simplest form of loop is written as:

```
DO FOREVER
    statement
    statement
    ...
ENDDO
```

which implies continuous repetition until something (either in the loop or an outside event) causes an exit. Thus the body of the loop should usually contain an UNDO or a RETURN statement.

6.5.3.6 The DO FOR Construct. This construct is used for selecting items from some list or sequence. Its general form is:

```
DO FOR selector
    statement
    statement
    ...
ENDDO
```

The actual selector can be chosen to be as meaningful as possible to the designer and reader. Examples are:

```
DO FOR EACH TABLE ENTRY
DO FOR EACH ELEMENT IN THE POSITIONS ARRAY
DO FOR ALL NODES IN THE TREE
DO FOR ALL "INTERESTING" ENTRIES IN THE DICTIONARY
```

6.5.3.7 The DO CASE Construct. The DO CASE construct is used to select one of a group of actions according to a given selection criterion. The meaning of the construct is clearest if each action is given a label as in:

```
DO CASE selector
label-1:
    sequence-1
label-2:
    sequence-2
  ...
label-n:
    sequence-n
ENDDO
```

Examples of DO CASE statements are:

```
DO CASE OF COMMAND NAME
DO CASE SWITCH SETTING
DO CASE ERROR MESSAGE NUMBER
```

6.5.3.8 Other Possible DO Constructs. The various DO constructs described above are only examples of the most common ones. The designer is perfectly free to invent some other form to fit the needs of a particular design. For example,

```
DO IN PARALLEL
```

might introduce a collection of labelled sequences which are to be executed in parallel, with synchronization automatically assured before proceeding past the ENDDO statement. As another example,

```
DO WITH INTERRUPTS DISABLED
```

might be used to introduce a sequence in which interrupts are not allowed.

6.5.4 The RETURN Statement

Normally, a flow segment will "return" to its "caller" when control reaches the end of the segment. However, the RETURN statement can be used to indicate premature exit from a flow segment. As with UNDO and CYCLE, the form

```
RETURN IF condition
```

is often useful. Some examples of RETURN statements are:

```
RETURN
RETURN IF END HAS BEEN REACHED
RETURN SYMBOL'S VALUE
RETURN "ILLEGAL REFERENCE"
```

7. EXTERNAL SEGMENTS

It is frequently desirable to collect references to flow segments which are not part of the current design document. These might, for example, represent operating system services or utility operations which are defined in other design documents. Such segments are known as external flow segments and are declared in external segments.

An external segment is introduced by the command

```
+------------------------------------------------------------+
!                                                            !
! %EXTERNAL   text                                           !
!                                                            !
+------------------------------------------------------------+
```

or

```
+------------------------------------------------------------+
!                                                            !
! %E   text                                                  !
!                                                            !
+------------------------------------------------------------+
```

where text is a sequence of characters to be used as the title of the external segment. The title will be displayed at the top of the external segment page and will be entered in the table of contents.

```
********
* NOTE *
********
```

The "External" or "E" commands do not, themselves, declare external flow segments -- they introduce segments within which external flow segments will be declared.

Examples of these commands are:

```
%EXTERNAL   BASIC UTILITIES
%EXTERNAL   I/O SUPPORT SERVICES
%E   GENERAL STORAGE MANAGEMENT FUNCTIONS
```

The body of an external segment consists of statements, each of which represents the name of a flow segment not defined elsewhere in the design document. Each statement consists of one or more lines with all but the last line of a statement having a '/' as the last non-blank character. Each statement is scanned as if it were the argument of a "%SEGMENT" command (see Chapter 6) and the resulting name is entered into the dictionary as the name of a flow segment. If a statement begins with a comment character (see

Section 2.5.3), no scanning is performed.

Before a statement is printed, leading and trailing blanks are removed, each sequence of imbedded blanks is replaced by a single blank, and each '/' used as a continuation character is replaced by a single blank. Statements which are too wide to fit in the segment box will be automatically continued when printed. Blank statements are ignored.

An example of the input form of an external segment is:

```
%EXTERNAL   LOW-LEVEL I/O OPERATIONS
OPEN FILE (FILE-NAME)
CLOSE FILE (FILE-ID)

READ (FILE-ID, INTO, MAX-BYTES)
WRITE (FILE-ID, FROM, COUNT)
```

8. LISTING CONTROL COMMANDS

This chapter describes a number of commands which are used to control various aspects of the listing of the design document.

8.1 SPECIFYING DESIGN TITLES

The title of the design may be specified by commands of the form

```
+------------------------------------------------------------+
|                                                            |
| %TITLE   text                                              |
|                                                            |
+------------------------------------------------------------+
```

where text is any sequence of characters. Several "TITLE" commands may be used in a single design. The text of these commands will be placed, centered and boxed, double spaced, with leading and trailing blanks removed, on the cover page of the design document. In addition, the text of the first "TITLE" command will be capitalized and placed at the top of each design page.

Some examples are:

```
%TITLE   FORTRAN COMPILER: PASS 3
%TITLE   TREE TRANSFORMATION PHASE
```

The "TITLE" commands should appear before the first segment.

8.2 SPECIFYING THE LISTING DATE

Normally, the date on which the current PDL run was started is the date displayed on the design title page and at the top of the other pages of the design. The date may be changed by the command

```
+------------------------------------------------------------+
|                                                            |
| %DATE   string                                             |
|                                                            |
+------------------------------------------------------------+
```

where the first nine characters of string will be used as the date. No checking is performed on this substitute date and it will be used as is in place of the system date.

Some examples are:

```
%DATE   6 May 81
%DATE   6.5.81
%DATE   5/6/81
%DATE   81/06/05
```

If used, the "DATE" command should appear before the first segment.

8.3 CONTROL OF UNDERSCORING

Normally, the PDL processor will underscore each keyword which begins a statement in a flow segment. If underscoring is not desired, it may be suppressed by the command:

```
+------------------------------------------------------------+
¦                                                            ¦
¦ %NOUSCORE                                                  ¦
¦                                                            ¦
+------------------------------------------------------------+
```

When the processor is installed, the NOUSCORE option may be established as the default. If this is done, the command

```
+------------------------------------------------------------+
¦                                                            ¦
¦ %USCORE                                                    ¦
¦                                                            ¦
+------------------------------------------------------------+
```

will cause underscoring to be performed. If used, these commands should precede the first segment.

****
* NOTE *
****

Underscoring of keywords may not be supported in all versions of the PDL processor due to lack of operating system or device support.

8.4 SOURCE LISTING CONTROL

Normally, the source input to the PDL processor is not listed. A listing may be requested by the command

```
+------------------------------------------------------------+
|                                                            |
| %SOURCE                                                    |
|                                                            |
+------------------------------------------------------------+
```

The source listing may be suppressed by the command

```
+------------------------------------------------------------+
|                                                            |
| %NOSOURCE                                                  |
|                                                            |
+------------------------------------------------------------+
```

This command suppresses source listing starting with the next source line. When NOSOURCE is in effect, only source lines in error will be listed.

8.5 CONTROLLING SOURCE LINE NUMBER PRINTING

The left margin of segment output pages contains the number of the source line which was used to form the corresponding output line. If more than one source line was used to form an output line, the number of the first of these lines will be displayed.

Display of these numbers may be prevented by the command

```
+------------------------------------------------------------+
|                                                            |
| %NOLNO                                                     |
|                                                            |
+------------------------------------------------------------+
```

When the processor is installed, the NOLNO option may be established as the default. If this is done, source line number printing may be requested by the command

```
+------------------------------------------------------------+
|                                                            |
| %LNO                                                       |
|                                                            |
+------------------------------------------------------------+
```

9. PROCESSOR REPORTS

Several types of reports can be printed which provide information about the content and structure of the design. The designer may choose the specific reports to be included.

9.1 SEGMENT REFERENCE TREES

This report shows the nesting of flow segment references. A separate tree is printed for each *root segment*, which is a flow segment that is not referenced by any flow segment but which, itself, references at least one flow segment.

When a segment is referenced recursively, its name is prefixed by an asterisk and the recursion is not further traced.

The presence or absence of this report is controlled by the command

```
+------------------------------------------------------------+
|                                                            |
| %TREE                                                      |
|                                                            |
+------------------------------------------------------------+
```

which specifies that the report is to be printed, and by

```
+------------------------------------------------------------+
|                                                            |
| %NOTREE                                                    |
|                                                            |
+------------------------------------------------------------+
```

which specifies that the report is not to be printed.

A special abbreviated form of the trees can be selected by the command

```
+------------------------------------------------------------+
|                                                            |
| %STREE                                                     |
|                                                            |
+------------------------------------------------------------+
```

In these so-called *short trees*, only the first occurrence of each subtree is printed. For subsequent occurrences, only the name of the first segment in the subtree will be printed, prefixed with a minus sign ("-").

If any of these commands are used, they should appear before the first segment. The default setting is "NOTREE".

9.2 DATA ITEM INDEX

The data item index shows each data item which was implicitly or explicitly declared in the design and the locations in the design where each is referenced. The data item index is requested by

```
%DINDEX
```

and is inhibited by

```
%NODINDEX
```

If either of these commands is used, it should appear before the first segment. The default setting is "NODINDEX".

9.3 FLOW SEGMENT INDEX

The flow segment index lists all flow segments (both internal and external) in the design. For each, it shows the location of its definition and the segment names and locations of all references to it.

The flow segment index is requested by

```
%SINDEX
```

and is inhibited by

```
%NOSINDEX
```

If either of these commands is used, it should appear prior to the first segment. The default setting is "SINDEX".

```
**************
*            *
* APPENDICES *
*            *
**************
```

A. LIST OF COMMANDS

D	start a data segment
DATA	start a data segment
DATACHR	define the data character
DATE	define date for printing purposes
DINDEX	print a data item index
E	start an external segment
EXTERNAL	start an external segment
G	start a group
GROUP	start a group
LCASE	force keywords to lower case
LNO	print source line numbers
NODINDEX	do not print data index
NOLNO	do not print source line numbers
NOSINDEX	do not print a flow segment index
NOSOURCE	do not print the source
NOTREE	do not print reference trees
NOUSCORE	do not underscore keywords
S	start a flow segment
SCASE	do not adjust keyword case
SEGMENT	start a flow segment
SINDEX	print flow segment index
SOURCE	print the source
STREE	print short reference trees

T	start a text segment
TEXT	start a text segment
TITLE	specify design titles
TREE	print reference trees
UCASE	print keywords in upper case
USCORE	underscore keywords

B. SPECIAL FEATURES FOR CERTAIN PDL VERSIONS

This chapter discusses features of the PDL processor which are available only in the following versions:

- 5331-PD1, PDL for DEC PDP-11 under RSX-11M
- 5332-PD1, PDL for DEC VAX-11 under VMS

B.1 CASE INSENSITIVITY OF COMMANDS AND KEYWORDS

PDL commands (such as %TITLE and %S) and flow segment keywords (such as IF and DO) will be recognized whether they are entered in upper-case, in lower-case, or even in a mixture of the two cases.

B.2 CASE INSENSITIVITY OF DICTIONARY ENTRIES

The PDL processor places several different kinds of items into an internal dictionary. These items are:

- flow segment names;
- external segment names; and
- data item names.

References to an entry will be detected regardless of whether the text is typed in upper-case, in lower-case, or in a mixture of the two cases.

In the body of the design, the text for an item will be printed in the same case(s) in which it appeared in the source. However, the table of contents and the indexes will be printed in upper-case.

B.3 TABS IN THE INPUT

Tab characters (octal 011) appearing in the input will be expanded to one or more blanks so as to position the input cursor to the next "tab stop". Tab stops are permanently set every eight columns so as to appear in columns 9, 17, 25,.... These inserted blanks will be treated just as if they had been entered in place of the tab character. Thus, they will be retained in text segments and replaced by a single blank in all other segments.

B.4 CONTROLLING KEYWORD DISPLAY CASE

Keywords will be printed in upper case, regardless of how they are entered, if the command

```
+------------------------------------------------------------+
|                                                            |
| %UCASE                                                     |
|                                                            |
+------------------------------------------------------------+
```

is used. They will be printed in lower case, regardless of how they are entered, if the command

```
+------------------------------------------------------------+
|                                                            |
| %LCASE                                                     |
|                                                            |
+------------------------------------------------------------+
```

is used. They will be printed in the same case(s) in which they are entered if the command

```
+------------------------------------------------------------+
|                                                            |
| %SCASE                                                     |
|                                                            |
+------------------------------------------------------------+
```

is used. The default action is SCASE.

Note that these case control commands are independent of the %USCORE and %NOUSCORE commands (Section 8.3). For example,

```
%UCASE
%USCORE
```

will force all keywords to upper case and will underscore them.

INDEX

```
                PROGRAM DESIGN
                 GENERATED BY
              CFG PDL PROCESSOR
```

```
****************************************
*                                      *
* P32-01/PHASE 30:  ALLOCATION (22.90) *
*                                      *
*          FIRST LEVEL DESIGN          *
*                                      *
*             CHAPTER:  22             *
*                                      *
*             SECTION:  90             *
*                                      *
*               21 JUN 83              *
*                                      *
*               PDL 03.6B              *
*                                      *
****************************************
```

TABLE OF CONTENTS

```
************************
*                      *
* THE ALLOCATION PHASE *
*                      *
************************
```

```
     PERFORM ALLOCATION

REF
PAGE ************************************************************************************************
     *                                                                                              *
   4 *  1     INITIALIZE PHASE                                                                      *
   5 *  2     COMPRESS AND OUTPUT NAME TABLE                                                        *
   6 *  3     COMPUTE STRUCTURE SIZES                                                               *
   7 *  4     ASSIGN NORMAL LOCATIONS                                                               *
   8 *  5     ASSIGN "AT" LOCATIONS                                                                 *
  10 *  6     RELOCATE ATTRIBUTE TABLE                                                              *
  11 *  7     TERMINATE PHASE                                                                       *
     *                                                                                              *
     ************************************************************************************************
```

```
         INITIALIZE PHASE

    REF
    PAGE ********************************************************************************************************************
         *                                                                                                          *
17       *  1    OPEN NAME FILE                                                                                     *
18       *  2    SET CODE AND DATA LOCATION COUNTERS TO ZERO                                                        *
         *                                                                                                          *
         ********************************************************************************************************************
```

COMPRESS AND OUTPUT NAME TABLE

```
REF
PAGE ***************************************************************************************************************
     *                                                                                                             *
20   *  1    SET LNT TO ZERO TO KEEP TRACK OF FINAL NAME TABLE LOCATIONS                                           *
21   *  2    DO FOR EACH HASH TABLE SLOT                                                                           *
22   *  3        DO FOR EACH NAME TABLE ENTRY, NT, ON THE HASH TABLE CHAIN                                         *
23   *  4            IF NT IS NOT A KEYWORD AND HAS A LINK CHAIN                                                   *
24   *  5                INCREMENT LNT BY NEW LENGTH OF ENTRY                                                      *
25   *  6                DO FOR EACH ATTRIBUTE TABLE ENTRY ON THE LINK CHAIN                                       *
26   *  7                    SET NAME FIELD OF ATTRIBUTE TABLE ENTRY TO LNT                                        *
27   *  8                ENDDO                                                                                     *
28   *  9                WRITE COUNT AND NAME FIELDS OF NTE ONTO NAME FILE                                         *
29   * 10            ENDIF                                                                                         *
30   * 11        ENDDO                                                                                             *
31   * 12    ENDDO                                                                                                 *
     *                                                                                                             *
     ***************************************************************************************************************
```

COMPUTE STRUCTURE SIZES

```
    REF
    PAGE *****************************************************************************************
         *                                                                                       *
33       *  1     DO FOR EACH DICTIONARY ENTRY, D                                                *
34       *  2         IF D IS A STRUCTURE                                                        *
35       *  3             SET SIZE FIELD OF D TO ZERO                                            *
36       *  4         ELSEIF D IS A STRUCTURE MEMBER                                             *
37       *  5             LET DS BE THE DICTIONARY ENTRY FOR THE STRUCTURE POINTED TO BY D       *
38       *  6             SET THE ADDRESS FIELD OF D TO THE SIZE FIELD OF DS                     *
39       *  7             COMPUTE THE MEMBER SIZE, MS, OF D ..ITEM WIDTH (1 OR 2) TIMES DIMENSION (IF THE MEMBER IS /*
         *                      AN ARRAY)                                                        *
41       *  8             INCREMENT THE SIZE FIELD OF DS BY MS                                   *
42       *  9         ENDIF                                                                      *
43       * 10     ENDDO                                                                          *
         *                                                                                       *
         *****************************************************************************************
```

```
     ASSIGN NORMAL LOCATIONS

 REF
 PAGE *****************************************************************************************************************
      *                                                                                                              *
45    *  1    SET EXTERNAL ITEM COUNT TO ZERO                                                                       *
46    *  2    DO FOR EACH DICTIONARY ENTRY, D, WHICH HAS NEITHER THE MEMBER NOR THE AT ATTRIBUTES                   *
48    *  3        IF THE ITEM IS EXTERNAL                                                                           *
49    *  4            SET THE EXTERNAL NUMBER FIELD OF D TO THE EXTERNAL ITEM COUNT                                 *
50    *  5            INCREMENT THE EXTERNAL ITEM COUNT                                                             *
51    *  6            SET THE ADDRESS FIELD TO ZERO                                                                 *
52    *  7        ELSEIF IT IS A PROCEDURE                                                                          *
53    *  8            SET THE LC FIELD TO ZERO                                                                      *
54    *  9            SET THE STACK LC FIELD TO ZERO                                                                *
55    * 10            SET THE STACK SIZE FIELD TO ZERO                                                              *
56    * 11        ELSEIF IT IS A VARIABLE OR A STRUCTURE                                                            *
57    * 12            IF IT HAS THE "BASED" ATTRIBUTE                                                               *
58    * 13                SET ITS ADDRESS FIELD TO ZERO                                                             *
59    * 14            ELSEIF IT DOES NOT HAVE THE "MEMORY" ATTRIBUTE                                                *
60    * 15                COMPUTE AMOUNT OF SPACE TO ALLOCATE (D) ..RETURNS AMOUNT, N                               *
61    * 16                IF IT HAS THE "DATA" ATTRIBUTE                                                            *
62    * 17                    PUT THE CONTENTS OF THE CODE AREA LOCATION COUNTER INTO THE ADDRESS FIELD OF D        *
64    * 18                    INCREMENT THE CODE AREA LOCATION COUNTER BY N                                         *
65    * 19                ELSEIF IT HAS THE "AUTOMATIC" ATTRIBUTE                                                   *
66    * 20                    LET DP BE THE DICTIONARY ENTRY FOR THE CONTAINING PROCEDURE                           *
67    * 21                    PUT THE CONTENTS OF THE STACK LC FIELD OF DP INTO THE ADDRESS FIELD OF D              *
68    * 22                    INCREMENT THE STACK LC FIELD OF DP BY N                                               *
69    * 23                ELSE                                                                                      *
70    * 24                    PUT THE CONTENTS OF THE VARIABLE AREA LOCATION COUNTER INTO THE ADDRESS FIELD OF     /*
      *                               DP                                                                             *
72    * 25                    INCREMENT THE VARIABLE AREA LOCATION COUNTER BY N                                     *
73    * 26                ENDIF                                                                                     *
74    * 27            ENDIF                                                                                         *
75    * 28        ENDIF                                                                                             *
76    * 29    ENDDO                                                                                                 *
      *                                                                                                              *
      ****************************************************************************************************************
```

ASSIGN "AT" LOCATIONS

```
    REF
    PAGE ***********************************************************************************************
          *                                                                                               *
 78       *  1     DO FOR EACH ENTRY ON THE AT/INIT FILE                                                  *
 79       *  2         IF RECORD IS "AI END"                                                              *
 80       *  3             RETURN                                                                         *
 81       *  4         ELSEIF RECORD IS "AI AT"                                                           *
 82       *  5             READ STATEMENT NUMBER (SN), DICTIONARY POINTER (DP), POINTER TO RESTRICTED REFERENCE  /*
          *                     (RD), SUBSCRIPT VALUES (RS1, RS2), AND THE ABSOLUTE VALUE (RV)            *
 85       *  6             IF RD IS ZERO                                                                  *
 86       *  7                 SET L TO RV                                                                *
 87       *  8             ELSE                                                                           *
 88       *  9                 IF ENTRY RD IS A MEMBER                                                    *
 89       * 10                     SET SD TO PARENT OF RD TO POINT TO STRUCTURE                           *
 90       * 11                     SET L TO ADDRESS OF SD + RS1* SIZE OF ONE ELEMENT OF SD + ADDRESS OF RD + RS2*   /*
          *                            SIZE OF ONE ELEMENT OF RD                                          *
 92       * 12                     SET RD TO SD SO RD NOW POINTS TO THE STRUCTURE                         *
 93       * 13                 ELSE                                                                       *
 94       * 14                     SET L TO ADDRESS OF RD + RS1* SIZE OF ONE ELEMENT OF RD                *
 95       * 15                 ENDIF                                                                      *
 96       * 16                 INCREMENT L BY RV                                                          *
 97       * 17             ENDIF                                                                          *
 98     9 * 18             ASSIGN ONE "AT" LOCATION                                                       *
 99       * 19         ENDIF                                                                              *
100       * 20     ENDDO                                                                                  *
          *                                                                                               *
          *************************************************************************************************
```

ASSIGN ONE "AT" LOCATION

```
     REF
     PAGE ***********************************************************************************************
          *                                                                                             *
102       *  1    DO FOREVER ..ONCE PER DICTIONARY ENTRY IN THE SEQUENTIAL GROUP IF ENTRY AT DP HAS THE "AT"   /*
          *          ATTRIBUTE                                                                          *
104       *  2       SET ADDRESS FIELD OF DP TO L                                                       *
105       *  3       IF RD IS ZERO                                                                      *
106       *  4          SET "ABSOLUTE" BIT IN DP                                                        *
107       *  5       ELSE                                                                               *
108       *  6          IF RD HAS THE "EXTERNAL" ATTRIBUTE                                              *
109       *  7             SET "EXTERNAL" ATTRIBUTE IN DP                                               *
110       *  8             SET EXTERNAL NUMBER FIELD IN DP TO EXTERNAL NUMBER FIELD IN RD               *
112       *  9          ELSEIF RD HAS THE "DATA" OR "MEMORY" OR "AUTOMATIC" ATTRIBUTES                  *
113       * 10             SET THESE ATTRIBUTES IN DP                                                   *
114       * 11          ENDIF                                                                           *
115       * 12       ENDIF                                                                              *
116       * 13       COMPUTE AMOUNT OF SPACE TO ALLOCATE (DP)                                           *
117       * 14       INCREMENT L BY THAT AMOUNT                                                         *
118       * 15       MOVE DP TO NEXT DICTIONARY ENTRY THAT DOES NOT HAVE THE "MEMBER" ATTRIBUTE         *
120       * 16       UNDO IF THAT ENTRY DOES NOT HAVE THE "SEQUENTIAL" ATTRIBUTE                        *
121       * 17    ENDDO                                                                                 *
          *                                                                                             *
          ***********************************************************************************************
```

RELOCATE ATTRIBUTE TABLE

```
     REF
     PAGE **************************************************************************************************************
          *                                                                                                            *
123       *  1    MOVE ATTRIBUTE TABLE TO HIGH END OF DYNAMIC AREA AND RECORD ITS NEW LOCATION                         *
125       *  2    ALLOCATE ENTRY POINT ADDRESS TABLE BELOW ATTRIBUTE TABLE                                             *
127       *  3    ALLOCATE CGL ADDRESS TABLE BELOW ENTRY POINT ADDRESS TABLE                                           *
129       *  4    ALLOCATE CGL PROC NR TABLE BELOW CGL ADDRESS TABLE                                                   *
131       *  5    INITIALIZE ENTRY POINT AND CGL TABLES TO ZERO                                                        *
          *                                                                                                            *
          **************************************************************************************************************
          TERMINATE PHASE

     REF
     PAGE ************************************************************************************************************
          *                                                                                                          *
133       *  1    OUTPUT END OF MODULE RECORD TO T25 FILE                                                            *
134       *  2    REWIND NAME, AT/INITIAL AND T25 FILES                                                              *
          *                                                                                                          *
          ************************************************************************************************************
```

```
**************************
*                        *
* INDEX TO FLOW SEGMENTS *
*                        *
**************************
```

INDEX TO FLOW SEGMENTS

```
***************************
*                         *
* SEGMENT REFERENCE TREES *
*                         *
***************************
```

```
PERFORM ALLOCATION
------------------


LN  DEF     SEGMENT
--  ---     -------

 1    3      PERFORM ALLOCATION
 2    4         INITIALIZE PHASE
 3    5         COMPRESS AND OUTPUT NAME TABLE
 4    6         COMPUTE STRUCTURE SIZES
 5    7         ASSIGN NORMAL LOCATIONS
 6    8         ASSIGN "AT" LOCATIONS
 7    9            ASSIGN ONE "AT" LOCATION
 8   10         RELOCATE ATTRIBUTE TABLE
 9   11         TERMINATE PHASE
```

```
                                   **************************
                                   *                        *
                                   * END OF DESIGN DOCUMENT *
                                   *                        *
                                   **************************




DL RUN TIME STATISTICS:
     135 INPUT CARDS PROCESSED
       9 FLOW SEGMENTS
   10000 WORDS OF CORE AVAILABLE,   2.2 % USED
```

PROGRAM DESIGN LANGUAGE PROCESSOR

21 JUN 83

PDL 03.6B

```
%STREE

%TITLE  P32-01/PHASE 30:  ALLOCATION (22.90)

%TITLE  FIRST LEVEL DESIGN

%TITLE  CHAPTER:  22

%TITLE  SECTION:  90

%G  THE ALLOCATION PHASE

%S PERFORM ALLOCATION
INITIALIZE PHASE
COMPRESS AND OUTPUT NAME TABLE
COMPUTE STRUCTURE SIZES
ASSIGN NORMAL LOCATIONS
ASSIGN "AT" LOCATIONS
RELOCATE ATTRIBUTE TABLE
TERMINATE PHASE

%S INITIALIZE PHASE
OPEN NAME FILE
SET CODE AND DATA LOCATION COUNTERS TO ZERO

%S COMPRESS AND OUTPUT NAME TABLE
SET LNT TO ZERO TO KEEP TRACK OF FINAL NAME TABLE LOCATIONS
DO FOR EACH HASH TABLE SLOT
DO FOR EACH NAME TABLE ENTRY, NT, ON THE HASH TABLE CHAIN
IF NT IS NOT A KEYWORD AND HAS A LINK CHAIN
INCREMENT LNT BY NEW LENGTH OF ENTRY
DO FOR EACH ATTRIBUTE TABLE ENTRY ON THE LINK CHAIN
SET NAME FIELD OF ATTRIBUTE TABLE ENTRY TO LNT
ENDDO
WRITE COUNT AND NAME FIELDS OF NTE ONTO NAME FILE
ENDIF
ENDDO
ENDDO

%S COMPUTE STRUCTURE SIZES
DO FOR EACH DICTIONARY ENTRY, D
IF D IS A STRUCTURE
SET SIZE FIELD OF D TO ZERO
ELSEIF D IS A STRUCTURE MEMBER
LET DS BE THE DICTIONARY ENTRY FOR THE STRUCTURE POINTED TO BY D
SET THE ADDRESS FIELD OF D TO THE SIZE FIELD OF DS
COMPUTE THE MEMBER SIZE, MS, OF D ..ITEM WIDTH (1 OR 2) TIMES/
DIMENSION (IF THE MEMBER IS AN ARRAY)
INCREMENT THE SIZE FIELD OF DS BY MS
ENDIF
ENDDO

%S ASSIGN NORMAL LOCATIONS
SET EXTERNAL ITEM COUNT TO ZERO
DO FOR EACH DICTIONARY ENTRY, D, WHICH HAS NEITHER THE MEMBER NOR/
```

```
THE AT ATTRIBUTES
IF THE ITEM IS EXTERNAL
SET THE EXTERNAL NUMBER FIELD OF D TO THE EXTERNAL ITEM COUNT
INCREMENT THE EXTERNAL ITEM COUNT
SET THE ADDRESS FIELD TO ZERO
ELSEIF IT IS A PROCEDURE
SET THE LC FIELD TO ZERO
SET THE STACK LC FIELD TO ZERO
SET THE STACK SIZE FIELD TO ZERO
ELSEIF IT IS A VARIABLE OR A STRUCTURE
IF IT HAS THE "BASED" ATTRIBUTE
SET ITS ADDRESS FIELD TO ZERO
ELSEIF IT DOES NOT HAVE THE "MEMORY" ATTRIBUTE
COMPUTE AMOUNT OF SPACE TO ALLOCATE (D) ..RETURNS AMOUNT, N
IF IT HAS THE "DATA" ATTRIBUTE
PUT THE CONTENTS OF THE CODE AREA LOCATION COUNTER INTO THE/
ADDRESS FIELD OF D
INCREMENT THE CODE AREA LOCATION COUNTER BY N
ELSEIF IT HAS THE "AUTOMATIC" ATTRIBUTE
LET DP BE THE DICTIONARY ENTRY FOR THE CONTAINING PROCEDURE
PUT THE CONTENTS OF THE STACK LC FIELD OF DP INTO THE ADDRESS FIELD OF D
INCREMENT THE STACK LC FIELD OF DP BY N
ELSE
PUT THE CONTENTS OF THE VARIABLE AREA LOCATION COUNTER INTO/
THE ADDRESS FIELD OF DP
INCREMENT THE VARIABLE AREA LOCATION COUNTER BY N
ENDIF
ENDIF
ENDIF
ENDDO

%S ASSIGN "AT" LOCATIONS
DO FOR EACH ENTRY ON THE AT/INIT FILE
IF RECORD IS "AI END"
RETURN
ELSEIF RECORD IS "AI AT"
READ STATEMENT NUMBER (SN), DICTIONARY POINTER (DP), POINTER TO/
RESTRICTED REFERENCE (RD), SUBSCRIPT VALUES (RS1, RS2),/
AND THE ABSOLUTE VALUE (RV)
IF RD IS ZERO
SET L TO RV
ELSE
IF ENTRY RD IS A MEMBER
SET SD TO PARENT OF RD TO POINT TO STRUCTURE
SET L TO ADDRESS OF SD + RS1* SIZE OF ONE ELEMENT OF SD +/
ADDRESS OF RD + RS2* SIZE OF ONE ELEMENT OF RD
SET RD TO SD SO RD NOW POINTS TO THE STRUCTURE
ELSE
SET L TO ADDRESS OF RD + RS1* SIZE OF ONE ELEMENT OF RD
ENDIF
INCREMENT L BY RV
ENDIF
ASSIGN ONE "AT" LOCATION
ENDIF
ENDDO
```

```
%S ASSIGN ONE "AT" LOCATION
DO FOREVER ..ONCE PER DICTIONARY ENTRY IN THE SEQUENTIAL GROUP/
IF ENTRY AT DP HAS THE "AT" ATTRIBUTE
SET ADDRESS FIELD OF DP TO L
IF RD IS ZERO
SET "ABSOLUTE" BIT IN DP
ELSE
IF RD HAS THE "EXTERNAL" ATTRIBUTE
SET "EXTERNAL" ATTRIBUTE IN DP
SET EXTERNAL NUMBER FIELD IN DP TO EXTERNAL NUMBER/
FIELD IN RD
ELSEIF RD HAS THE "DATA" OR "MEMORY" OR "AUTOMATIC" ATTRIBUTES
SET THESE ATTRIBUTES IN DP
ENDIF
ENDIF
COMPUTE AMOUNT OF SPACE TO ALLOCATE (DP)
INCREMENT L BY THAT AMOUNT
MOVE DP TO NEXT DICTIONARY ENTRY THAT DOES NOT HAVE THE/
"MEMBER" ATTRIBUTE
UNDO IF THAT ENTRY DOES NOT HAVE THE "SEQUENTIAL" ATTRIBUTE
ENDDO

%S RELOCATE ATTRIBUTE TABLE
MOVE ATTRIBUTE TABLE TO HIGH END OF DYNAMIC AREA AND RECORD ITS/
NEW LOCATION
ALLOCATE ENTRY POINT ADDRESS TABLE BELOW/
ATTRIBUTE TABLE
ALLOCATE CGL ADDRESS TABLE BELOW/
ENTRY POINT ADDRESS TABLE
ALLOCATE CGL PROC NR TABLE BELOW/
CGL ADDRESS TABLE
INITIALIZE ENTRY POINT AND CGL TABLES TO ZERO

%S TERMINATE PHASE
OUTPUT END OF MODULE RECORD TO T25 FILE
REWIND NAME, AT/INITIAL AND T25 FILES

%END *GENERATED CARD*

NO ERRORS DETECTED
```