the tool for software designers

PDL/81™

Ada® Design Language

Reference Guide

(Version 2.0)

Caine, Farber & Gordon, Inc.

Warren Point International Ltd.

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: (800) 424-3070 or
(818) 449-3070

Fax: (818) 440-1742

In the Rest of the World:

Warren Point International Ltd.
Babbage Road
Stevenage, Herts SG1 2EQ
England

Tel: 0438 316311

Fax: 0227 86521

Form Number: 9102-52

1 August 1988
1 December 1991

Contents

Appendices

1. Introduction

PDL/81 is a software tool intended as an aid to designing and documenting a program or system of programs. The tool consists of a processor and a data base which is used to tailor the processor to the particular requirements of the document being produced. As distributed, the data base includes definitions for such document styles as:

- program designs;
- manuals and reports;
- memoranda; and
- business letters.

This manual describes the particular data base components which relate to formatting program designs which are intended to be implemented in the Ada programming language. Other manuals (see Section 1.4) describe the other data base components and the methods for modifying the data base.

The original version of PDL, known as PDL/74, was first developed in 1973. It was intended exclusively for processing program design documents and displaying these documents in a predetermined style. Over the years since the first release, the large PDL user community has provided numerous suggestions for changes and improvements. Most of these suggestions came from the desire to improve the text handling capabilities of PDL/74 and the desire to have significantly more control over the detailed format of the resulting document. PDL/81 addresses these desires directly while still presenting an interface to the designer which is easy to use.

1.1 Designing for Ada

While the standard PDL/81 *design* style is intended for use with almost any desired implementation language, the *ada* style is intended for use when design or implementation requirements call for use of the Ada language. It has been found to satisfy the needs of a design language based on the Ada programming language while maintaining the readability of a PDL/81 design.

The PDL/81 *ada* style supports those features of the Ada language that are suitable to the high-level design of software. It is entirely possible that a design which was started in PDL/81 may, at some stage, be moved to a rigorous, but less readable, Ada design language or rapid prototyping system prior to the start of implementation.

1.2 Features and Capabilities of PDL/81

The development of PDL/81 was guided by three primary goals:

- to develop a system which extended the capabilities of PDL/74 in the area of program design;
- to develop a system which could process designs written in the PDL/74 language with little or no modifications of those designs being required; and
- to develop a system which was flexible enough to process both program designs in the style of PDL/74 and conventional documents such as reports and manuals.

The result is a tool which integrates the capabilities commonly associated with a program design language processor and those of a text processing system.

This integration is accomplished by providing an extensive set of primitive formatting operations and a definitional language which allows a format designer to compose abstract constructs from these primitive operations. As an example, a document style for Ada program designs might contain such concepts as “specification segment” and “task body segment” while a style for manuals might contain such concepts as “chapter”, “enumerated list”, and “paragraph”.

The end user of PDL/81 uses these abstract concepts without any need to understand the underlying implementation or format design methods. Writing and processing program designs is simple and straightforward while, at the same time, the local project manager has significant control over the detailed layout and appearance of the resulting design document.

The primitive operations of the Format Design Language allow the format designer a very high degree of flexibility in creating document styles. Among the available capabilities are:

- Complete control over page layout including sheet dimensions and top, bottom, left, and right margins;
- Simple measurements of the cyclomatic complexity of a design;
- Tracking of requirements sections throughout a design;
- Checking that procedure definitions and invocations are consistent;
- Arbitrary running text at top and bottom of each page including security banners with document classification and sheet count;
- Definition of primary and secondary keywords for use in program designs;
- Definition of layout and characteristics of all program design segment types and the ability to create new types of segments;
- Ability to include input from alternate files;
- Automatic generation of table of contents and other such tables (e.g., table of figures, table of tables);

- Automatic generation of document indexes in various forms.

1.3 Document Styles and the PDL/81 Data Base

The document styles which are available at an installation reside in the PDL/81 "data base". The form of the data base depends on the particular host operating system. The particular style to be used in a PDL/81 run is specified as an option when PDL/81 is invoked.

1.4 Related Publications

Other publications relating to the use of PDL/81 are:

- *PDL/81 Introduction and Invocation Guide* – a guide to invoking PDL/81 under various operating environments
- *PDL/81 Design Language Reference Guide* – a guide to using PDL/81 for producing general-purpose software design documents *PDL/81 Document Language Reference Guide* – a guide to using PDL/81 for producing various documents such as manuals and reports
- *PDL/81 Format Designers Guide* – A guide to developing new types of PDL/81 design and document styles
- *PDL/81 Installation Guide* – a guide to installing PDL/81 under the various supported operating systems.

1.5 A Note to the Reader

This manual describes the distributed *ada* document style which is intended to be the standard style for formatting program designs to be implemented in the Ada programming language. Two sample program designs are presented in Section C.1 and Section C.2.

If you don't like the results of this style, you may desire to modify the data base. Simple modifications can generally be accomplished after an examination of various data base entries. Extensive modifications, and the development of entirely new design styles, will require reference to the *PDL/81 Format Designers Guide*.

The *ada* style should be considered as an example of the kind of design tool which may be defined with PDL/81. For any particular project, it may be desirable to tailor a specific definitions file by removing many of the options which are described here.

2. General Information

This chapter discusses various aspects of the PDL/81 *ada* design style which are of general interest. It includes information on the form of a design, overall operation of the processor, and the syntax of PDL/81 commands.

This chapter does not discuss how to invoke the PDL/81 processor under the various supported operating systems. Invocation is discussed in the *PDL/81 Introduction and Invocation Guide.*

2.1 Format of a Design

The PDL/81 *ada* style accepts as input a series of source lines and produces a design document. The document can be formatted for printing on different types of output devices and different paper sizes.

Two sample designs appear as Section C.1 and Section C.2. 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 10.1) and the *date* command (see Section 10.2).

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 *packages* (see Chapter 3). The segment types are:

- *Text Segments*: which represent arbitrary commentary (see Chapter 4).
- *Specification Segments*: which allow definition of procedures, functions, tasks, task entries, records, and data which are assumed to be global to a package (see Chapter 7).
- *Data Segments*: which allow definition of data items that are assumed to be local to a package (see Section 6.3).
- *Procedure Segments*: which define procedures which are not defined in specification segments and supply the bodies of all procedures (see Chapter 8).
- *Function Segments*: which define functions which are not defined in specification segments and supply the bodies of all functions (see Chapter 8).
- *Task Body Segments*: which supply the bodies of all tasks (see Chapter 8). Task entry points are defined in specification segments.

Procedure segments, function segments, and task body segments are collectively referred to as *flow segments*.

2.1.3 Reports

The processor can be instructed to produce several reports (see Chapter 12) 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 procedure, function, and task references arranged in the form of a calling tree. There will be several trees if there are several flow roots in the design. Recursive references will be indicated. Reference trees are further described in Section 12.1.
- *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 12.2.
- *Flow Segment Index*: which lists each procedure, function, task, and entry point in alphabetic order and shows the points in the design where each is referenced. The flow segment index is further described in Section 12.3.
- *Overly Complex Segment Index*: which lists each segment which has a cyclomatic complexity value greater than the selected maximum (see Section 11.1 for a discussion of complexity measurement and Section 12.4 for a discussion of the report.
- *Requirements Index*: which lists each declared requirement number and the segments that address that requirement (see Section 11.2 for a discussion of requirements tracking and Section 12.5 for a discussion of the report.

- *Calls-in-Context Index*: which shows each procedure, function, or entry pint definition along with each line that calls it (see Section 11.3 for a discussion of calls-in-context and Section 12.6 for a discussion of the report.

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 Invocation of PDL/81

The manner of invoking PDL/81 depends on the particular operating system being used. Invocation is discussed in the *PDL/81 Introduction and Invocation Guide*.

2.3 Overall Operation

PDL/81 processes a design in two passes. During the first pass, the source is read, page breaks are determined, and a dictionary of data item and segment names is constructed. During the second pass, the source is reread, references to data items and segments are detected, and the design document is formatted. During both passes, progress is noted by displaying the current page number and processing phase on a file (which will usually be the controlling terminal).

2.4 Input Format

Input to PDL/81 consists of a sequence of source lines. Each line is terminated by a newline character. This section describes the interpretation of various special characters and character sequences within source lines. The only ASCII control codes allowed on an input line are “tab” and “newline”.

2.4.1 Tab Expansion on Input

ASCII tab characters are allowed on input lines. Each tab will be replaced by enough blanks to position the immediately following character to the next input tab stop. Input tab stops are set at columns 1, 9, 17, ...

2.4.2 Continuation of Input Lines

Any input line may be continued in one of two ways:

- The sequence “\<newline>” results in deletion of both characters, thus causing the following line to be considered part of the current line. The character “\” is known as the *escape character* and has additional uses as described in Section 2.4.3.
- The sequence “/<newline>” will be replaced by a single blank, thus causing the current and following lines to be treated as a single line with their contents separated by a blank. The character “/” is known as the *continue character*. It has special significance only when it immediately precedes a newline character – in any other context, it is just another character.

2.4.3 Special Characters

The character sequence “#{” is used to introduce a text function as described in Chapter 9. The sequence should not appear in any other context, as the results will be unexpected. If it is necessary to use the sequence for some other purpose, the “#” should be protected by an escape character as in “\#{”.

The special sequence “*” will be replaced by the so-called *bullet* character (bullet) in the printed output. The form of this special character depends greatly on the output device being used.

The escape character followed by a space is known as the *unpaddable space*. It will be replaced by a single space in the printed output, but will not be considered to mark a word break during processing.

2.5 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 PDL/81.

When a command line is encountered, white space (blanks and tabs) following the “%” is skipped. If a newline is encountered, the command line is ignored. If an asterisk (“*”) is encountered, the line is considered to be a *comment command*, the rest of the line is skipped, and the whole line is ignored.

If anything else is encountered, it is assumed to start a *command name* which extends to the first blank, tab, or newline. 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”.

The case (upper, lower, mixed) of a command name is immaterial. Thus, for example, “Title”, “TITLE”, “title”, or even “tiTlE” all represent the same command name.

2.6 Including Alternate Source Files

At any point in the design source, input may be switched to another source file by the command

```
%Include  file
```

where *file* is the name of the file to be included. Files included with an %Include command may contain %Include commands.

2.7 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 PDL/81 while processing a design.

2.7.1 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)
%TextF or %TF	start a formatted text segment (Chapter 4)
%SPEC	start a specification segment (Chapter 7)
%Data or %D	start a data segment (Section 6.3)
%Function or %F	start a function segment (Chapter 8)
%Procedure or %P	start a procedure segment (Chapter 8)
%Taskbody or %Task	start a task body segment (Chapter 8)

A segment is terminated by the next occurrence of a segment command, a %Package command (see Chapter 3), or the end of the design source.

2.7.2 Display of Segments

Each segment can occupy one or more pages. However, experience has shown that designs are generally much more readable and understandable if each segment is limited to a single page.

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
F	function segment
P	procedure segment
T	task body segment

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

2.7.3 Comment Strings

The descriptions of many of the segments refer to syntactic constructs known as *comment strings* which are used as delimiters in certain contexts (e.g., to separate a procedure name from its “arguments”).

Initially, there are two comment strings defined – the Ada comment delimiter “--” the left parenthesis (“(”). Replacement or additional comment strings may be defined by the command

```
%CString   [string]
```

where *string* is one or two non-blank printing characters other than letters or digits. No two comment strings may begin with the same first character. If *string* is absent, all comment strings will be deleted.

3. Packages

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

```
%PACKAGE  text
```

where *text* is any sequence of characters to be used as the title for the package. For compatibility with previous versions of the *ada* style, the commands

```
%GROUP   text
```

or

```
%G   text
```

may be used instead of %PACKAGE.

In the design document, each package will be prefaced with a page containing the title of the package centered and boxed. The title will also appear as a subtitle on each design page within the package and will be placed in the table of contents for the design.

Examples of package declarations are

```
%Package  Pass One Processing
%Package  Input Editing Phase
```

A package is terminated by the next “Package” command or by the end of the design.

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. There are two types of text segments – *unformatted* and *formatted*. Only commands specific to text segments are described in this chapter. See Chapter 9 and Chapter 5 for a discussion of other functions and commands which are useful in text segments.

4.1 Unformatted Text Segments

An unformatted 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 an unformatted text segment are simply input and printed as is. No automatic formatting will be performed except that lines which are too long to fit into the segment box will be split at word boundaries and printed on two or more lines. White space on input lines is kept and blank input lines will result in blank output lines.

Examples of commands to introduce an unformatted text segment are:

```
%Text   Introduction to Position Monitoring Module
%T      Other Documents Relating to this Subsystem
```

4.2 Formatted Text Segments

A formatted text segment is introduced by the command

```
%TextF   text
```

or

```
%TF   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 into the table of contents.

The lines which comprise the segment are considered to be running text. Words are collected, regardless of the input line boundaries, and are put into the output line until a word does not fit. The output line is then printed and a new output line is started. This action is known as “breaking” the line. A break is forced by a blank input line and by the commands described in Section 4.2.1 and Chapter 5. White space on input lines is kept and blank input lines result in blank output lines.

Examples of commands introducing formatted text segments are:

```
%TextF   Standards Used in this Design
%TF      Outline of Link-Level Protocols
```

4.2.1 Lists

Within formatted text segments, three types of lists may be automatically formatted:

bullet — each list entry is prefixed by the bullet (bullet) character.

numbered — each list entry is prefixed by an automatically generated number.

verb — each list entry is prefixed by an arbitrary word or phrase. (This list is an example of a verb list.)

The same general structure is used for generating each kind of list. This structure, presented in the form of a numbered list, is:

1. a list start command specifying the type of the list
2. one or more list entries
3. a %LE (list end) command to mark the end of the list

Lists may be nested.

Each list will be automatically preceded and followed by a blank line.

4.2.1.1 Bullet Lists

A bullet list is introduced by the command

```
%BL
```

The text for the list entries should follow, separated from each other by a single blank line. The list is closed by the command

```
%LE
```

which should not be preceded by a blank line.

4.2.1.2 Numbered Lists

A numbered list is introduced by the command

```
%NL
```

The text for the list entries should follow, separated from each other by a single blank line. The list is closed by the command

```
%LE
```

which should not be preceded by a blank line.

4.2.1.3 Verb Lists

A verb list is introduced by the command

```
%VL [indent]
```

where *indent* is a decimal integer which specifies the number of characters to indent the text of the list items. In the absence of *indent*, a value of 16 will be used.

Each item in a verb list is introduced by the command

```
%Verb text
```

where *text* is the word or phrase to be displayed at the left margin. The text of the list entry follows on succeeding lines.

The last entry in a verb list should be followed by the command

```
%LE
```

to close the list.

4.3 Switching Between Formatted and Unformatted Modes

The initial formatting mode (formatted or unformatted) for a text segment is determined by the command which introduces that text segment. Within a text segment, either formatting mode may be established at any point by the commands

```
%Fill
```

which establishes formatted mode and

```
%NoFill
```

which establishes unformatted mode.

5. General Formatting Commands

This chapter describes commands which relate to the general control of formatting within a segment. These commands are most often used in text segments (see Chapter 4) but may be used in any kind of segment.

5.1 Vertical Spacing Commands

Blank lines may be inserted into a segment by the command

```
%Space   number
```

where *number* is a decimal integer giving the number of blank lines to insert. If the given number of blank lines exceeds the number of available lines remaining on the page, a new page is started instead. For example,

```
%Space   3
```

would cause three blank lines to be inserted or would cause a page eject if there were not at least three lines remaining on the page.

The command

```
%Need   number
```

where *number* is a decimal integer, will cause a page eject if at least *number* lines do not remain on the current page. If at least that many lines remain, the command has no effect. Thus,

```
%Need  5
```

will cause a new page to be started if fewer than five lines remain on the current page.

The command

```
%Eject
```

will cause a new page to be started.

5.2 Heading Commands

The heading commands allow descriptive headings to be placed within a segment. When mentioned below, the terms “centered” and “flush left” are to be taken as being relative to the textual display area within the segment box.

The command

```
%MajorHeading   text
```

will skip two lines; print *text* centered, underscored, and capitalized; and skip two lines.

The command

```
%Heading   text
```

will skip two lines; print *text* flush left, underscored, and capitalized; and skip one line.

The command

```
%SubHeading   text
```

will skip one line; print *text* flush left and underscored; and skip one line.

6. Data Item Declaration

PDL/81 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 12.2).

6.1 Data Items

Within data segments (see Section 6.3), specification segments (Chapter 7), and flow segments (see Chapter 8), 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 6.2) or explicit (see Section 6.3) data declaration.

Lines which begin (possibly after some leading white space) with a comment string (see Section 2.7.3) will not be examined for potential data items.

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

```
x = a$1+bb*cc
```

the potential data items are

```
x    a$1    bb    cc
```

The case (upper, lower, mixed) of letters in names of potential data items is immaterial. Thus, for example, “test”, “Test”, “TEST”, and even “tEst” all represent the same item.

The set of these special characters may be modified by the command

```
%DSChar   char
```

where *char* is a non-blank, non-alphanumeric character to be added to the set of data item special characters. If *char* is absent, the set is made empty. All uses of the “DSChar” command should precede the first segment.

As an example, the special characters “%” and “!” can be added to the set by the commands

```
%DSChar   %
%DSChar   !
```

and the set can be defined to contain only the character “$” by

```
%DSChar
%DSChar   $
```

6.2 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 only data character. New data characters may be added to the set of data characters by the command

```
%DChar   char
```

where *char* is a non-blank, non-alphanumeric character to be added. If *char* is absent, the set is made empty. All uses of the “DChar” command should precede the first segment.

For example, the characters “-” and “$” can be added to the set of data characters by the commands

```
%DChar   -
%DChar   $
```

and the character “!” can be defined as the only data character by

```
%DChar
%DChar   !
```

For compatibility with older versions, the command

```
%DataChar  char
```

establishes the single character *char* as the only data character. The preferred method of changing data characters is to use the “DChar” described above.

6.3 Explicit Data Item Declaration (Data Segments)

Data items are explicitly defined in *data segments* and in *specification 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 that 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. Lines beginning with a comment string (see Section 2.7.3) are considered to be comments and are not scanned for declarations. White space on source lines is kept and blank input lines will result in blank output lines.

6.3.1 Normal Declaration Mode

In the normal mode of data item declaration, the first potential data item in each line of the body is declared to be an actual data item. Anything following the data item on the line is taken as commentary. Thus, in the line

```
CType is the type of the command
```

“CType” will be declared to be a data item.

6.3.2 Special Declaration Mode

In the special declaration mode, a potential data item is declared as an actual data item *only* if it contains a data character. If the data character is the *first* character of the potential data item, it is not included as part of the name of the actual data item.

At the start of each data segment, the normal declaration mode is in effect. The special declaration mode can be established for that segment by the command

```
%SDMode
```

and the normal declaration mode can be re-established by the command

```
%NoSDMode
```

As an example, consider the line

```
Items _c1, file_count, and _open_count are counters
```

In the special declaration mode, the actual data items will be

```
c1     file_count     open_count
```

As another example, the lines

```
*********************************//*******************
*       *        *                                   *
* _code * _count *            record_text            *
*       *        *                                   *
*********************************//*******************
```

will declare “code”, “count”, and “record_text” to be actual data items.

7. Specification Segments

A *specification* segment may be used to group definitions of procedures, functions, tasks, entry points, and data. Typically, these are thought of as *global* even though the PDL/81 processor does not make such a distinction for reference purposes.

A specification segment is introduced by the command

```
%Spec   text
```

where *text* is a sequence of characters to use as the title of the segment. The title will be used to label the segment page in the design and will appear in the table of contents. Some examples are

```
%Spec  Background Tasks
%Spec  File Manipulation Functions
```

7.1 Defining Procedures and Functions

A procedure is defined by a *procedure* statement which has the form

```
PROCEDURE name ( argument-list )
```

and a function is defined by a *function* statement which has the form

```
FUNCTION name ( argument-list ) RETURNS return-item
```

Note that these statements only define names and arguments – procedure bodies are defined in procedure segments (Chapter 8) and function bodies are defined in function segments (Chapter 8).

Some examples are

```
PROCEDURE open file (file name)
procedure terminate processing
Procedure send initial message(message text, \
       message class, message code)

Function normalize(value) returns normalized value
FUNCTION decrypt password (text, key) returns \
       decrypted string
```

7.2 Defining Tasks

Tasks are defined in a specification segment by use of the TASK construct which has the form

```
TASK task name IS
    first entry definition
    second entry definition
    ...
END TASK
```

The “IS” token is optional. If it is not given, the PDL/81 processor will supply it in the output listing. Each task entry definition has the form

```
entry name ( argument-list )
```

Some examples are

```
TASK sensor poll IS
END TASK

TASK status monitor
    monitor it (detector code, detector state)
END TASK

task time stamper
    time stamp(type, value)
    change time (new time)
end task
```

7.3 Defining Data

Data may be defined in a specification segment in the same way it is defined in a data segment (Section 6.3). Both normal and special declaration modes may be used. In addition, data records may be declared by a construct of the form

```
RECORD record-name
    first data item
    second data item
    ...
END RECORD
```

An example is

```
RECORD message
    type
    length
    text
END RECORD
```

8. Flow Segments

A *flow segment* presents, in a program-like form, the procedural flow of a portion of a design. In the *ada* design style, the three types of flow segments are:

- Procedure segments,
- Function segments, and
- Task body segments.

All of these have essentially the same form and differ mainly in the particular command that is used to introduce them.

8.1 Flow Segment Commands

A procedure segment is introduced by the command

```
%PROCEDURE   text
```

or

```
%P   text
```

A function segment is introduced by the command

```
%FUNCTION   text
```

or

```
%F   text
```

A task body segment is introduced by the command

```
%TASKBODY   text
```

or

```
%TASK   text
```

In each case, *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 string (see Section 2.7.3) 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
`%Procedure System Start`	System Start
`%P Install in Data Base (Name, Type)`	Install in Data Base
`%F Search Dictionary (Name) Returns entry`	Search Dictionary
`%Task Monitor Sensors`	Monitor Sensors

Note that this definition implies that there is no real difference between the three types of flow segment. In fact, the PDL/81 processor treats them very much alike. For consistency, however, argument lists and return items should be written as shown in the examples above.

8.2 Flow Segment Body

The body of a flow segment is composed of one or more lines. These lines are known as *statements* to emphasize the relation between a flow segment and a procedure in a programming language.

A statement may start anywhere on a line. PDL/81 will correctly format and indent each statement on output and may supply various forms of visual enhancement to the printed line. Leading blanks will be removed and each sequence of imbedded blanks will be replaced by a single blank. Blank input lines will be ignored. Statements which are too wide to fit in the segment box will be automatically continued when printed.

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/81.

With the exception of the *special statements* discussed in Section 8.6, 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 *continue character* (see Section 2.4.2), the last two lines of this example are equivalent to:

```
Bump Count to reflect the record just processed
```

8.3 Reference Recognition

Each statement in a flow segment, except for a statement which begins with a comment string (see Section 2.7.3), will be scanned to see if it is the name of a flow segment. If a statement begins with a *keyword* (see Section 8.6), the scan begins following the keyword and any subsequent *secondary keywords*. The scanning stops at the first comment string.

In any match, leading and trailing blanks are removed, each sequence of imbedded blanks is replaced by a single blank, and the case (upper, lower, mixed) of all letters is ignored.

Lines beginning with comment strings in data segments and flow segment are normally not scanned for data item definitions or references or for flow segment references. Scanning can be specified in this case by the command

```
%CData
```

and may be inhibited by the command

```
%NoCData
```

If used, these commands should appear before the first segment.

8.4 Labels

It is occasionally desirable to place *labels* in a design. If the first non-blank character in a statement is a "<" and a ">" is encountered before the next blank or the end of the statement, that statement is considered to be a *label*. Note that this loosely supports Ada labels of the form `<<identifier>>`. The statement will be printed flush with the left margin so that it will stand out. Anything following the label on the same line will be treated as commentary. Some examples of labels are:

```
<<MainSearchLoop>>
<<END_OF_FILE>>
<<ReBooot>>
```

8.5 Block Names

It is occasionally desirable to identify a particular block or loop for purposes of reference. If a colon (“:”) is encountered before the first blank in a statement, that statement is considered to be a block name. The statement will be printed flush with the left margin so that it will stand out. Anything following the name on the same line will be treated as commentary. Some examples of block names are:

```
Search:
inner:
ReCycler:
```

8.6 Special Statements

The so-called *special statements* comprise the flow-of-control statements in the PDL/81 procedural language. This section describes each of the special statements.

8.6.1 Keywords and Secondary Keywords

Each special statement begins with a *keyword* followed by a blank or the end of the input line. The keywords are

IF	ELSEIF	ELSE	END IF
WHILE	FOR	LOOP	END LOOP
BEGIN	END	CASE	END CASE
SELECT	END SELECT	ACCEPT	OR
EXIT	GOTO	RETURN	DELAY
TERMINATE	RAISE	ABORT	EXCEPTION

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.

There is also a so-called *secondary keyword* which is recognized as such only when immediately following a keyword or a secondary keyword. The secondary keyword is

```
NOT
```

The case (upper, lower, mixed) of keywords and secondary keywords is ignored.

The keywords and secondary keywords discussed above are those defined in the *ada* style as distributed. New project-wide keywords and secondary keywords may be added by modifying the style file.

8.6.1.1 Keyword Enhancement

The form in which a keyword or secondary keyword is printed depends on the use of various commands described in this section. These commands, if used, should appear before the first segment. Initially, keywords and secondary keywords are printed in *upper* case regardless of the case in which they are entered.

The command

```
%LCase
```

causes keywords and secondary keywords to be printed in lower case regardless of the case in which they are entered.

The command

```
%SCase
```

causes keywords and secondary keywords to be printed in the same case in which they were entered.

For compatibility with older versions, the command

```
%NoLCase
```

also specifies that keywords are to be printed in the same case in which they are entered. The preferred method of accomplishing this is to use the “SCase” command described above.

The command

```
%UCase
```

causes keywords and secondary keywords to be printed in upper case regardless of the case in which they were entered. This is the default setting.

The command

```
%UScore
```

causes each keyword and secondary keyword to be underscored when printed.

The command

```
%NoUScore
```

specifies that each keyword and secondary keyword is not to be underscored when printed. This is the default setting.

Flexibility in font selection is provided by the command

```
%KWFont   n
```

where *n* is a font number:

0 base font (no special treatment except for possible conversion to upper case or lower case under control of the %UCASE or %LCASE commands).

1 underscored (same effect as obtained by the %USCORE command).

2 bold face (only if supported by your installation on the selected device).

8.6.2 Special Statement Display

The PDL/81 processor tries to display special statements in a canonic form regardless of how they were input. For example,

```
WHILE some condition
```

will be displayed as

```
WHILE some condition LOOP
```

as will

```
while some condition loop
```

8.6.3 The IF Construct

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

```
IF condition THEN
    sequence
END IF
```

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 THEN
    sequence-1
ELSE
    sequence-2
END IF
```

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

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

```
IF condition-1 THEN
    sequence-1
ELSEIF condition-2 THEN
    sequence-2
ELSE
    sequence-3
END IF
```

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 END IF

The THEN in the IF and ELSEIF statements is optional and will be supplied by the processor if it is missing.

8.6.4 The LOOP Constructs

The LOOP construct consists of the keywords LOOP, WHILE, FOR, and END LOOP. END LOOP may also be written as ENDLOOP.

The basic loop construct has the form

```
LOOP
    statements...
END LOOP
```

and implies iteration until some condition causes exit from the loop.

The WHILE construct has the form

```
WHILE condition LOOP
    statements...
END LOOP
```

and implies iteration as long as the condition is true.

The FOR construct has the form

```
FOR selector LOOP
    statements...
END LOOP
```

and implies iteration while selecting items or values from some list or sequence. The actual selector can be chosen to be as meaningful as possible to the designer and reader. Examples are:

```
FOR each table entry LOOP
FOR each element in the Positions array LOOP
FOR all nodes in the tree LOOP
FOR all "interesting" entries in the dictionary LOOP
```

In the WHILE and FOR constructs, the LOOP keyword is optional and will be supplied by the processor if it is missing.

8.6.5 The EXIT Statement

The EXIT statement is used to indicate that control should pass to the statement immediately following the END LOOP of the current LOOP construct, thus causing premature exit from the loop. It might be used in the following context:

```
WHILE source input remains LOOP
    process next source line
    IF dynamic memory is full THEN
        EXIT
    END IF
END LOOP
```

An alternate form of the EXIT statement is

```
EXIT WHEN condition
```

which can make the design more concise as in:

```
WHILE source input remains LOOP
    process next source line
    EXIT WHEN dynamic memory is exhausted
END LOOP
```

When LOOP constructs are nested, it may sometimes be necessary to indicate a premature exit from an outer loop. This is most easily shown by naming (Section 8.5) the outer loop and writing using the full form of the statement which is

```
EXIT [loop-name] [WHEN condition]
```

8.6.6 The CASE Construct

The CASE construct consists of the CASE, WHEN, and END CASE keywords. END CASE may be written as ENDCASE. The CASE construct is used to select one of a group of actions according to a given selection criterion.

```
CASE selector IS
    WHEN choice-1 =>
        sequence-1
    WHEN choice-2 =>
        sequence-2
        ....
    WHEN choice-n =>
        sequence-n
END CASE
```

The IS keyword and the “=>” delimiter are optional and will be supplied by the processor if they are missing.

Examples of CASE statements are

```
CASE of command name IS
CASE switch setting IS
CASE error message number IS
```

8.6.7 The BEGIN Construct

The BEGIN construct consists of the BEGIN and END keywords and has the form

```
BEGIN
    statements...
END
```

8.6.8 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. Some examples of RETURN statements are:

```
RETURN
RETURN Symbol’s Value
RETURN "Illegal Reference"
```

8.6.9 The ACCEPT Construct

This construct has the form

```
ACCEPT entry definition DO
    statements...
END
```

The DO is optional and will be supplied by the processor if it is missing.

8.6.10 The SELECT Construct

The SELECT construct has the form

```
SELECT text
    [ WHEN condition => ]
        statements...
    [ OR [ WHEN condition => ]
        statements...
    ...
    [ ELSE ]
        statements...
END SELECT
```

The => delimiter is optional and will be supplied by the processor if it is missing.

8.6.11 The GOTO Statement

The GOTO statement has the form

```
GOTO label name
```

See Section 8.4 for a discussion of labels.

8.6.12 The EXCEPTION Construct

Exception handlers may be declared with the EXCEPTION construct. For ease of visual recognition, the processor draws a horizontal line across the segment at the start of the construct. Thus, an exception construct should only appear at the end of a segment. The construct has the form

```
EXCEPTION
    WHEN exception name =>
        statements...
    WHEN exception name =>
        statements...
    ...
```

The => delimiter is optional and will be supplied by the processor if it is missing.

8.6.13 Miscellaneous Statements

The keywords ABORT, DELAY, RAISE, and TERMINATE, support the Ada statements of the same form. Examples are

```
ABORT all pending tasks
DELAY ten minutes
RAISE overflow
TERMINATE
```

9. Text Functions

Text functions are used to insert special information into a design or to perform some kind of textual modification. They are most commonly used in text segments but may appear in any type of segment.

The general form of a text function invocation is

```
#{name[;argument;argument;...]}
```

where *name* is the name of the text function and the *arguments* depend upon the requirements of the particular function. If an argument to a text function contains any of

```
#{   {   }   ;
```

each must be preceded by an escape character ("\") as described in Section 2.4.3. The entire text function invocation must appear on a single (possibly continued) source line.

9.1 The DATE Text Function

The date on which the current run of PDL/81 was started may be obtained by

```
#{date}
```

which will be replaced by the date in the same form as it appears at the top of each page of the design. For example, the line

```
The current date is #{date}
```

will be printed as

```
The current date is 6 Sep 86
```

9.2 Underscoring of Text

Keywords and secondary keywords in flow segments may be automatically underscored by use of the "UScore" command as described in Section 8.6.1.1. Other text may be underscored by

```
#{us;text}
```

which causes each non-blank character in "text" to be underscored and by

```
#{uc;text}
```

which causes each character in "text" to be underscored. For example,

```
this #{us;is under}scored and #{uc;so is this}
```

will print as

```
this ulineis ulineunderscored and ulineso is this
```

If bold face output is supported at your installation on the selected device, the text function

```
#{bf;text}
```

will print *text* in bold face,

```
#{bfu;text}
```

will print *text* in bold face with non-blank characters underscored, and

```
#{bfuc;text}
```

will print *text* in bold face with all characters underscored.

9.3 Tags and References

A *tag* is a symbol which can be used to mark a particular point in a design and is declared by

```
%Tag   symbol
```

where *symbol* is the name of the tag. The output page number corresponding to the location of the tag will be associated with the tag and may be retrieved by the text function

```
#{ref;symbol}
```

For example, if the command

```
%Tag  test
```

appeared at a point which was to be printed on page five of the design document, the line

```
See page #{ref;test} for a description.
```

would print as

```
See page 5 for a description.
```

10. Listing Control Commands

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

10.1 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 unless a “Ptitle” (Section 10.1.1) command is used.

Some examples are:

```
%Title   Fortran Compiler: Pass 3
%Title   Tree Transformation Phase
```

The “Title” commands should appear before the first segment.

10.1.1 Defining a Page Head

The command

```
%PTitle   text
```

will cause the text to be used as the running page head for the design. If this is not used, the running head will be the text of the first %Title command.

10.2 Specifying the Listing Date

Normally, the date on which the current PDL/81 run was started is the date displayed on the design title page and at the top of the other pages of the design and is the date returned by the “date” text function (see Section 9.1). 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 91
%Date   6.5.91
%Date   5/6/91
%Date   91/06/05
```

If used, the “Date” command should appear before the first segment.

10.3 Specifying Security Banners

A security banner will be placed at the top and bottom of each output page by the command

```
%Security   classification
```

where *classification* is a word or phrase specifying the security classification of the design document. The command

```
%Project text
```

specifies “text” to be a project identification word or phrase to be included in the security banner. The “Project” command will be ignored in the absence of a “%Security” command. Both of these commands, if used, should appear before the first segment.

In addition to the security classification and optional project name, each security banner will contain a sequential sheet number for document control purposes. These numbers start at “one” for the title page and are incremented by one for each sheet printed. They are independent of the page numbers assigned by PDL/81 for reference purposes. The last page of the design will contain a count of the total number of sheets printed.

As an example, the security banners which appear on the sample design at the end of this manual were specified by

```
%Security   UNCLASSIFIED
%Project    PDL/81 SAMPLE DESIGN
```

10.3.1 Security Banner Style

The format of security banners may be changed to reflect various standards.

By default, banners will have the classification centered, the sheet number on the right, and the project identification on the left. This mode is known as *Security Style* 0.

An alternate security style, 1, is the same as zero except that the sheet number will appear on the left and the project identification will appear on the right on even numbered sheets. This is useful when printing duplexed designs.

The default security style may be changed by editing the style files. On a per-document basis, the command

```
%SecStyle   number
```

where the number is one of the security styles (0 or 1) will set the security style in a design.

10.4 Specifying "Special" Boxes

Experience has shown that designs are often printed on serial printers since such printers are available with compressed type fonts which allow a full-width design to be printed on 8-1/2 by 11 inch paper. These printers can be very slow, however, when printing designs because of the large amount of white space which may appear between the end of a statement and the right edge of the segment box. The command

```
%SBox
```

specifies that the right edge of all boxes is not to be printed. This usually results in faster printing during the draft stage.

The command

```
%NoSBox
```

specifies that the right edge of all boxes is to be printed (the default case).

If either of these commands is used, it should appear prior to the first segment.

10.5 Specifying Line Number Printing

The PDL/81 processor does not normally display source line numbers in the design document. This can be changed by the

```
%LNO
```

which causes source line numbers to appear to the right of the segment box. Not all lines will be numbered. The display of line numbers may be stopped by

```
%NOLNO
```

10.6 Specifying Change Bars

Change bars, which appear on the right of segment boxes, can be displayed by the command

```
%MC   char
```

where *char* is a single character to be used as the change bar. If *char* is absent, the display of change bars is stopped.

For example, bracketing a section of changed design with

```
%MC    |
```

and

```
%MC
```

will cause the character "|" to be used as a change bar for that section of the design.

11. Advanced Features

This chapter describes several advanced features of the *ada* style including:

- Cyclomatic complexity measurement and reporting;
- Automatic requirements tracking;
- Consistency checking;
- Flow figure enhancement;
- Maintaining design and code in the same file.

11.1 Complexity Analysis

This version supports a form of cyclomatic complexity measurement based on the work of McCabe (*A Complexity Measure*, McCabe, Thomas J., IEEE Transactions on Software Engineering, Vol SE-2, No 4, Dec 1976). It performs this measurement by assigning a complexity value to certain keywords and secondary keywords and summing these values for each flow segment – the higher the value, the more complex the segment.

The complexity for each flow segment is printed on the segment's output page and in the *Index of Flow Segments* (Section 12.3). If a segment's complexity exceeds some specified value (6, by default), a warning message is issued on the standard output and also appears on the segment's output page. An index to such overly complex segments is also printed (Section 12.4). Finally, various complexity statistics are given on the summary page at the end of the design document.

11.1.1 Complexity Measurement Commands

The command

```
%Complexity  [max]
```

specifies that complexity measurement is to be performed. If *max* is given, it should be an integer constant giving the maximum allowable complexity to use instead of the default value of 6.

The command

```
%NoComplexity
```

specifies that complexity measurement is not to be performed.

11.2 Automatic Requirements Tracking

Information about requirements are input by the command

```
%Req   r1;r2;...;rn
```

or

```
%R   r1;r2;...;rn
```

where each of the “rsub{i}” is a paragraph number of a requirement taken from the controlling requirements document. When used with DOD-STD 2167, this might be the *Software Requirements Specification*, DI-MCR-80025. The paragraph numbers must have the general form of section and subsection identifiers separated by decimal points. Such an identifier is a decimal integer optionally prefixed by an alphabetic string. Examples are

```
3.5.6   3.9.6.2   R4.2   4.6.R3.2
```

If a segment has associated requirements, the %R commands for the segment must immediately follow the segment command (e.g., %SPEC, %P, %T). When a segment which references requirements is printed, the associated requirements will be displayed following the segment box.

11.2.1 Requirements Index

Optionally, an index of all requirements and their associated segments may be printed. This is accomplished by using the command

```
%RIndex
```

If this action is established as the default during installation, it may be suppressed by the command

```
%NoRIndex
```

11.3 Consistency Checking

This release provides for consistency checking of segment references in a style that is in the spirit of PDL/81. This is done by optionally producing a report known as the *Calls-In-Context List* which shows each flow segment definition and a listing of each line that calls that segment. Those calls which appear to be inconsistent in number of arguments with the definition are flagged in the report. For the purpose of this report, an argument list is assumed to be enclosed in parentheses and arguments are separated by zero-level (with respect to parentheses and single and double quotation marks).

The report is requested by the command

```
%CiC
```

If this action is established as the default during installation, it may be suppressed by the command

```
%NoCiC
```

11.4 Flow Figure Enhancement

The command

```
%KwV
```

specifies that the beginning and end of each flow figure will be connected by a series of a predetermined character. This may be turned off by the command

```
%NoKwV
```

A default character and font may be established by editing the style and/or device definition files. The character and font may be chosen on a per-device basis so that advantage may be taken of any specialized device characteristics (e.g., a line-drawing character set).

The character and font may be set on a per-design basis by the command

```
%KwVC  char[;font-expr]
```

If the font is not specified, the base font will be used.

11.5 Design and Code in the Same File

This new feature of PDL/81 allows maintaining both the design and the code for a program in the same file. Code sequences, known as *code segments*, are introduced by the command

```
%Code  [file-name]
```

where *file-name* is the name of a file to receive this code when code selection is enabled. If the file name is not specified, the code will be written to the file name given in the last preceding %Code command that had a file name or, if none such exist, to the file specified by the last preceding

```
%CodeFile  [file-name]
```

command. If no file name is in effect, code is written to the standard output.

During normal runs of PDL/81, code segments are *not* output; rather, they are completely skipped. To cause code segment selection, invoke PDL/81 with the “GetCode” number register set as in

```
pdl81 -rGetCode file
```

As a final option, a normal design run of PDL/81 can be made with code segments being displayed in the output document. This is accomplished by invoking PDL/81 with the “ShowCode” number register set as in

```
pdl81 -rShowCode file
```

If this is done, code segments may *not* contain sequences which look like invocations of Format Design Language functions.

12. 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.

12.1 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. If no root segments are found, an arbitrary choice will be made and the resulting tree will be printed.

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 “STree”.

12.2 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 code “DI” in the report indicates an explicitly defined data item while the code “ID” indicates an implicitly defined item.

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 “DIndex”.

12.3 Flow Segment Index

The flow segment index lists all procedures, functions, tasks, and entry points in the design. For each, it shows the location of its definition and the segment names and locations of all references to it. The type of an item is indicated by a code:

P	Procedure
GP	Global Procedure
F	Function
GF	Global Function
SP	Specification Segment
TK	Task Body
EP	Task Entry Point

Global procedures, global functions, and task entry points will have appeared in specification segments. Global procedures and global functions may also appear later in the design as segments. In that case, the definition page given in the index will be the page for the flow segment and not for the specification segment.

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”.

12.4 Index of Overly Complex Segments

If complexity measurement is enabled (Section 11.1), this index will be printed if any segments excede the predefined maximum allowable complexity. By default, this value is 6.

The index will show the complexity, type, location, and name of each segment with too high a complexity.

12.5 Index to Requirements

If requirements tracking is enabled (Section 11.2), an index of requirements will be printed. This index will be sorted by requirement and will show the location and name of each segment that addresses that requirement.

12.6 Calls-in-Context List

When enabled (Section 11.3), this listing will show each procedure or function call together with its definition. Inconsistent usage will be flagged.

A. Error Messages

This Appendix lists error messages which may be issued during processing of a design. Error messages are displayed on the standard error file. If applicable, the message will be prefixed with the name of the current input file and the current line number within the file.

A.1 Non-Terminal Error Messages

The error messages described in this section do not cause termination of PDL/81 processing:

- COMMAND INVALID OUTSIDE OF SEGMENT – this command may only appear within a segment.
- COMMAND INVALID OUTSIDE OF TEXT SEGMENT – this command may only be used within a text segment.
- DUPLICATE DATA ITEM: <item> – the named item has been previously defined as a data item.
- DUPLICATE ENTRY POINT: <name> – the given name has previously been defined as the name of a flow segment or of an entry point.
- DUPLICATE GLOBAL NAME: <name> – the given name has already been declared as global in a Specification segment.
- DUPLICATE NAME: <name> – the given name, which appears as the argument of a “Procedure” or “Function” command, has been previously defined as the name of a flow segment.
- DUPLICATE TAG: <name> – the given name has been previously defined in another “Tag” command.
- ENDING KEYWORD WITH NO OPEN FLOW FIGURE – an ending keyword, such as ENDIF, was encountered but a flow figure is not open for it to close.
- FLOW FIGURE NOT CLOSED AT END OF SEGMENT – a flow figure is still open when the end of a segment was encountered.
- INVALID CHARACTER IN LINE – an input line contains an ASCII control character other than “tab” or “newline”.

- NAME MISSING – a name was not provided for a group or a segment.
- REQUIREMENTS MUST BE PART OF A SEGMENT – A “Req” or “R” command has been encountered but it is precedes the first segment or immediately follows a “Package” statement.
- SEGMENT TOO COMPLEX – the cyclomatic complexity of the segment is greater than the allowable maximum.
- TEXT OUTSIDE OF SEGMENT – a source line which was not a command appeared outside of a segment. A generated “Segment” command will be inserted.
- UNBALANCED BRACKETS – the number of unescaped left brackets is not the same as the number of unescaped right brackets within a call on a text function.
- UNDEFINED TAG: <name> – the given name was referenced in a “Ref” text function but did not occur also in a “Tag” command.
- UNKNOWN COMMAND – a command name on a command line is not one of those recognized by PDL/81.

A.2 Terminal Error Messages

The error messages described in this section cause immediate termination of PDL/81 processing:

- CAN’T OPEN TEMP FILE <file name> – the named temporary file cannot be opened. This usually means that disk space is not available for the file or that write access privileges are not available in the directory on which the file is to be written.
- CANNOT ALLOCATE DYNAMIC MEMORY FOR A BUFFER – Memory was needed for an input/output buffer, but insufficient memory was available.
- DYNAMIC MEMORY OVERFLOW (n) – all available dynamic memory is allocated and more is needed. The character “n” indicates the particular point in the processor where overflow was detected and is of interest only to PDL/81 processor maintenance personnel.
- MKTEMP: CANNOT GENERATE UNIQUE FILE NAME: <file name> – Names of PDL/81 temporary files are generated by the internal PDL/81 “mktemp” function. This function can generate up to 26 unique names for each invocation of PDL/81. Since names will be reused when possible, and since PDL/81 deletes temporary files after they are closed, this message usually means that a large number of temporaries were left around following a system crash. Examine the directory given in the message and delete the abandoned temporaries.
- SOURCE FILE NOT GIVEN – a source file was not specified when PDL/81 as invoked.
- UNABLE TO OPEN FILE <file name> – the named file cannot be opened for input. Possibly, it doesn’t exist.
- UNKNOWN DEVICE TYPE: <name> – the named device type was specified by an invocation option but no such device is supported.

- UNKNOWN INVOCATION OPTION: <option> – the invocation line contained an option which was not recognized by PDL/81.

A.3 Other Error Messages

The error messages described above are those which relate to processing designs using the *design* document style. Other messages may be issued but they relate to internal processing errors or system problems and should not appear when processing designs. A more complete list of such messages may be found in the {it PDL/81 Format Designers Guide}.

B. List of Commands

BL	start a “bullet” list
CIC	enable calls-in-context index printing
Code	start a Code segment
CodeFile	define a file to receive extracted code
Complexity	enable complexity measurement
CString	define comment strings
D	start a data segment
Data	start a data segment
Date	define date for printing purposes
DChar	define data characters
DIndex	print a data item index
DSChar	define data item special characters
Eject	begin a new page of output
External	start an external segment
F	start a function segment
Fill	switch to formatted mode in a text segment
Function	start a function segment
G	start a group
Group	start a group
Heading	print a second level heading
Include	include source from an alternate file
KWFont	specify font in which to print keywords
KWV	enable flow figure enhancement

KWVC	define character for flow figure enhancement
LCase	print keywords in lower case
Le	end a list
LNO	start display of source line numbers
MajorHeading	print a first level heading
MC	start or stop display of change bars
Need	assure enough lines remain on a page
NL	start a numbered list
NoCIC	disable calls-in-context index
NoComplexity	disable complexity measurement
NoDIndex	do not print data index
NoFill	switch to unformatted mode in a text segment
NoKWV	disable flow figure enhancement
NOLNO	stop display of source line numbers
NoRindex	disable requirements index
NoSBox	do not print special boxes
NoSIndex	do not print a flow segment index
NoTree	do not print reference trees
NoUScore	do not underscore keywords
P	start a procedure segment
Project	specify name of project for security banners
Procedure	start a procedure segment
PTitle	define running page title
R	specify requirements
Req	specify requirements
Rindex	enable requirements index
S	start a flow segment
SBox	use special segment boxes
SCase	print keywords in same case as entered
SecStyle	specify style of security banners
Security	specify security classification of design
SIndex	print flow segment index
Space	space a given number of blank lines
Spec	start a specification segment
STree	print short reference trees

SubHeading	print a third level heading
T	start an unformatted text segment
Tag	define a tag
Task	start a task body segment
TaskBody	start a task body segment
Text	start an unformatted text segment
TextF	start a formatted text segment
TF	start a formatted text segment
Title	specify design titles
Tree	print reference trees
UCase	print keywords in upper case
UScore	underscore keywords
Verb	put a verb in a verb list
VL	start a verb list

C. Sample PDL/81 Designs for Ada

This Appendix presents two short examples of PDL/81 designs for Ada. Each is followed by a listing of the input source which resulted in the design listings.

C.1 Illustration of Features

This sample illustrates some of the more commonly used features of the *ada* style. Note that security banners are used to illustrate the distributed format. Of course, the format can be easily changed to suit the requirements of a particular program.

C.1.1 Output of PDL/81 Processor

Beginning on the next page is the actual output of the PDL/81 processor when presented with the source shown in Section C.1.2.

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                 UNCLASSIFIED                         SHEET 1  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


                         CAINE, FARBER & GORDON, INC.
                            1010 EAST UNION STREET
                          PASADENA, CALIFORNIA 91106


                              *******************
                              *                 *
                              *    PDL/81 Ada   *
                              *                 *
                              *  Demonstration  *
                              *                 *
                              *    21 Jan 92    *
                              *                 *
                              * PDL/81 X2.0.911 *
                              *                 *
                              *     5500-PD8    *
                              *                 *
                              *******************
```


```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                 UNCLASSIFIED                         SHEET 1  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
TABLE OF CONTENTS
-----------------
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                  UNCLASSIFIED                        SHEET 3  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
CFG, INC.       PDL/81 ADA                                              PAGE  2
21 Jan 92
```

```
                                ****************
                                *              *
                                * Introduction *
                                *              *
                                ****************
```



```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                  UNCLASSIFIED                        SHEET 3  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
CFG, INC.       PDL/81 ADA                                             PAGE  3
21 Jan 92       Introduction
```

```
     Features of the Ada Design Style

     #############################################################################
     #                                                                           #
     #  1    The Ada design style differs from the regular PDL/81 design         #
     #  2    style in several important ways:                                    #
     #                                                                           #
     #  3         1. Ada keywords are used.                                      #
     #                                                                           #
     #  4         2. The constructs for Ada tasking and exceptions are added     #
     #  5            to the usual constructs for structured programs.            #
     #                                                                           #
     #  6         3. The general Flow Segment is replaced by Procedure           #
     #  7            Segments, Function Segments, and Task Body Segments.        #
     #                                                                           #
     #  8         4. The External Segment is replaced by the Specification       #
     #               Segment.                                                    #
     #                                                                           #
     #  9         5. Task entries are included in the flow segment               #
     #               references.                                                 #
     #                                                                           #
     #                                                                           #
     #############################################################################
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                 UNCLASSIFIED                       SHEET 5  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       PDL/81 ADA                                          PAGE  4
21 Jan 92       Introduction


     The Specification Segment


     ##############################################################################
     #                                                                            #
     #  1    The Specification Segment provides a way to define the              #
     #       following:                                                           #
     #                                                                            #
     #  2         1. Procedures                                                   #
     #                                                                            #
     #  3         2. Functions                                                    #
     #                                                                            #
     #  4         3. Tasks                                                        #
     #                                                                            #
     #  5         4. Task Entries                                                 #
     #                                                                            #
     #  6         5. Records                                                      #
     #                                                                            #
     #  7         6. Data                                                         #
     #                                                                            #
     #  8    If a procedure, function, or task is later defined as a             #
     #  9    segment, the reference number on the left will be to that           #
     # 10    segment.  Task entries must be defined in a Specification           #
     #       Segment.                                                             #
     #                                                                            #
     ##############################################################################
```


```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                 UNCLASSIFIED                       SHEET 5  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
CFG, INC.       PDL/81 ADA                                             PAGE  5
21 Jan 92       Introduction


     The Flow Segment Index


     ############################################################################
     #                                                                          #
     #  1     The Flow Segment Index uses many more types than in the           #
     #  2     standard design style.  What used to be just a Flow Segment can   #
     #  3     now be a Procedure, a Function, or a Task (which may have         #
     #  4     entries).  A procedure or function which appears in a             #
     #  5     Specification Segment is considered to be global.  The possible   #
     #        types are:                                                        #
     #                                                                          #
     #  7     P      procedure                                                  #
     #                                                                          #
     #  8     GP     global procedure                                           #
     #                                                                          #
     #  9     F      function                                                   #
     #                                                                          #
     # 10     GF     global function                                            #
     #                                                                          #
     # 11     SP     specification segment                                      #
     #                                                                          #
     # 12     TK     task body                                                  #
     #                                                                          #
     # 13     EP     task entry                                                 #
     #                                                                          #
     # 14     Global procedures, global functions, and task entries will have   #
     # 15     appeared in Specification Segments.  Global procedures and        #
     # 17     global functions may also apppear later as segments.  In that     #
     #        case, the definition page given in the index is the page for      #
     # 18     the segment, itself, and not the page for the Specification       #
     #        Segment.                                                          #
     #                                                                          #
     ############################################################################
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                  UNCLASSIFIED                       SHEET 7  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       PDL/81 ADA                                             PAGE  6
21 Jan 92




                                   ************
                                   *          *
                                   * Examples *
                                   *          *
                                   ************
```



```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                  UNCLASSIFIED                       SHEET 7  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                UNCLASSIFIED                       SHEET 8  ||
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       PDL/81 ADA                                           PAGE  7
21 Jan 92       Examples


     A Specification Segment

REF
PAGE SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
     S                                                                            S
   8 S         PROCEDURE first procedure (first arg, second arg) --defined later   S
     S         PROCEDURE second procedure --not defined in this design            S
     S                                                                            S
   9 S         FUNCTION first function () return word                             S
     S                                                                            S
     S         RECORD                                                             S
     S         |   aaa -- a data item                                             S
     S         |   bbb -- another data item                                       S
     S         END RECORD                                                         S
     S                                                                            S
     S         --other data not in a record                                       S
     S         ccc                                                                S
     S         ddd                                                                S
     S                                                                            S
  11 S         TASK first task IS                                                 S
     S         |   first entry --this is the only way to define entries           S
     S         |   second entry (an argument)                                     S
     S         END TASK                                                           S
     S                                                                            S
     S         TASK another task IS --not defined in this design                  S
     S         |   another entry                                                  S
     S         END TASK                                                           S
     S                                                                            S
     SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
```

```
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                UNCLASSIFIED                       SHEET 8  ||
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                 UNCLASSIFIED                         SHEET 9  ||
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       PDL/81 ADA                                          PAGE  8
21 Jan 92       Examples


     PROCEDURE First Procedure (arg one, arg two)

REF
PAGE PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
     P                                                                            P
   9 P        WHILE first function LOOP                                           P
     P        |   set ccc to arg two                                              P
     P        |   EXIT WHEN time is up                                            P
     P        END LOOP                                                            P
     P        CASE of some selection IS                                           P
     P        |   WHEN yellow =>                                                  P
   8 P        |       first procedure (aaa,ddd)                                   P
   7 P        |       second procedure                                            P
     P        |   WHEN red =>                                                     P
     P        |       RETURN                                                      P
     P        END CASE                                                            P
     P                                                                            P
     PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP


|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                 UNCLASSIFIED                         SHEET 9  ||
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                 UNCLASSIFIED                       SHEET 10  ||
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       PDL/81 ADA                                             PAGE  9
21 Jan 92       Examples


     FUNCTION First Function () RETURN word

REF
PAGE FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
     F                                                                           F
     F       IF something THEN                                                   F
     F       |   set something to nothing                                        F
     F       ELSEIF something else THEN                                          F
     F       |   set something to nothing else                                   F
     F       ELSE                                                                F
   8 F       |   first procedure (ccc, ddd)                                      F
     F       END IF                                                              F
     F                                                                           F
     FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

```
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                 UNCLASSIFIED                       SHEET 10  ||
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                 UNCLASSIFIED                       SHEET 11  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       PDL/81 ADA                                            PAGE  10
21 Jan 92       Examples


     PROCEDURE A Procedure with an Exception

REF
PAGE PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
     P                                                                           P
     P  1    FOR all time LOOP                                                   P
   7 P  2    |   second procedure                                                P
     P  3    END LOOP                                                            P
     P--4------------------------------------------------------------------------P
     P       EXCEPTION                                                           P
     P  5        WHEN overflow =>                                                P
     P  6            set to largest number                                       P
     P  7        WHEN underflow =>                                               P
     P  8            set to zero                                                 P
     P                                                                           P
     PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
```


```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                 UNCLASSIFIED                       SHEET 11  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
CFG, INC.       PDL/81 ADA                                             PAGE  11
21 Jan 92       Examples


     TASK BODY First Task

REF
PAGE TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
     T                                                                            T
     T       LOOP                                                                 T
     T       |   SELECT                                                           T
     T       |   WHEN true =>                                                     T
   7 T       |   |   ACCEPT first entry                                           T
     T       |   OR WHEN true =>                                                  T
   7 T       |   |   ACCEPT second entry (an in-out argument) DO                  T
     T       |   |       incement argument by 5                                   T
     T       |   |   END                                                          T
     T       |   OR                                                               T
     T       |       DELAY for half an hour                                       T
     T       |   ELSE                                                             T
     T       |       nobody is responding                                         T
     T       |   END SELECT                                                       T
     T       END LOOP                                                             T
     T                                                                            T
     TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                 UNCLASSIFIED                       SHEET 13  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       PDL/81 ADA                                            PAGE  12
21 Jan 92       Examples


     TASK BODY Second Task

REF
PAGE TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
     T                                                                           T
     T           LOOP                                                            T
   7 T           |   another entry                                               T
     T           |   SELECT                                                      T
   7 T           |   |   second entry (ddd)                                      T
     T           |   ELSE                                                        T
   8 T           |   |   first procedure (aaa,bbb)                               T
     T           |   END SELECT                                                  T
     T           END LOOP                                                        T
     T                                                                           T
     TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                 UNCLASSIFIED                       SHEET 13  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
***********************
*                     *
* INDEX TO DATA ITEMS *
*                     *
***********************
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                  UNCLASSIFIED                       SHEET 15  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       PDL/81 ADA                                         PAGE  13.001
21 Jan 92       INDEX TO DATA ITEMS


INDEX TO DATA ITEMS
-------------------


PAGE  LINE  TYPE    NAME AND REFERENCES
----  ----  ----    -------------------

   7     7   DI      aaa
                         8  first procedure
                                 7
                        12  Second Task
                                 6

   7     8   DI      bbb
                        12  Second Task
                                 6

   7    12   DI      ccc
                         9  first function
                                 6
                         8  first procedure
                                 2

   7    13   DI      ddd
                         9  first function
                                 6
                         8  first procedure
                                 7
                        12  Second Task
                                 4
```


```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                  UNCLASSIFIED                       SHEET 15  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO               UNCLASSIFIED                        SHEET 16  ||
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

CFG, INC.       PDL/81 ADA                                           PAGE  14
21 Jan 92
```

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

```
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                 UNCLASSIFIED                       SHEET 17  ||
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       PDL/81 ADA                                          PAGE  14.001
21 Jan 92       INDEX TO FLOW SEGMENTS


INDEX TO FLOW SEGMENTS
----------------------


PAGE  LINE  TYPE    NAME AND REFERENCES
----  ----  ----    -------------------

  10         P       A Procedure with an Exception

   7    21   EP      another entry
                        12  Second Task
                                2

   7    16   EP      first entry
                        11  First Task
                                4

   9         GF      first function
                         8  first procedure
                                1

   8         GP      first procedure
                         9  first function
                                6
                         8  first procedure
                                7
                        12  Second Task
                                6

  11         TK      First Task

   7    17   EP      second entry
                        11  First Task
                                6
                        12  Second Task
                                4

   7     2   GP      second procedure
                        10  A Procedure with an Exception
                                2
                         8  first procedure
                                8

  12         TK      Second Task
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                UNCLASSIFIED                        SHEET 18  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

CFG, INC.       PDL/81 ADA                                            PAGE  15
21 Jan 92

                            *************************
                            *                       *
                            * CALLS-IN-CONTEXT LIST *
                            *                       *
                            *************************
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                 UNCLASSIFIED                       SHEET 19  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       PDL/81 ADA                                         PAGE  15.001
21 Jan 92       CALLS-IN-CONTEXT LIST


CALLS-IN-CONTEXT LIST
---------------------


F PAGE  NAME / CALL
- ----  -----------

    10   A Procedure with an Exception

     7   another entry
    12   another entry

     7   first entry --this is the only way to define entries
    11   accept first entry

     7   function first function () return word
     9   First Function () return word
*    8   while first function loop

     7   procedure first procedure (first arg, second arg) --defined later
     8   First Procedure (arg one, arg two)
     8   first procedure (aaa,ddd)
     9   first procedure (ccc, ddd)
    12   first procedure (aaa,bbb)

     7   second entry (an argument)
    11   accept second entry (an in-out argument) do
    12   second entry (ddd)

     7   procedure second procedure --not defined in this design
     8   second procedure
    10   second procedure
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                UNCLASSIFIED                 SHEET 20 OF 20  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

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

```
STATISTICS
----------

5 flow segments.

2370 lines in definition file(s).
151 lines in source file(s).

668 dictionary entries allocated.
2539 string segments allocated;  2374 in use.

72192 bytes of dynamic memory allocated.
```


```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  ADA STYLE DEMO                UNCLASSIFIED                 SHEET 20 OF 20  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

C.1.2 Source Listing

The input lines which resulted in the design document of the preceding section are:

```
%*- -sada
%* @(#) $Header: ada1,v 2.0 88/03/27 16:09:16 shc Rel $
#{nr;_kwerr;0}
%security UNCLASSIFIED
%project ADA STYLE DEMO
%title PDL/81 Ada
%title Demonstration
%cic
%kwv |
%NoSbox
%NoTree
%g Introduction
%TF Features of the Ada Design Style
The Ada design style differs from the regular PDL/81 design style in several
important ways:
%nl
Ada keywords are used.

The constructs for Ada tasking and exceptions are added to the usual constructs
for structured programs.

The general Flow Segment is replaced by Procedure Segments, Function Segments,
and Task Body Segments.

The External Segment is replaced by the Specification Segment.

Task entries are included in the flow segment references.
%le
%TF The Specification Segment
The Specification Segment provides a way to define the following:
%nl
Procedures

Functions

Tasks

Task Entries

Records

Data
%le
If a procedure, function, or task is later defined as a segment, the reference
number on the left will be to that segment.
Task entries must be defined in a Specification Segment.
%TF The Flow Segment Index
The Flow Segment Index uses many more types than in the standard design style.
What used to be just a Flow Segment can now be a Procedure, a Function, or a
Task (which may have entries).
A procedure or function which appears in a Specification Segment is considered
to be global.
The possible types are:
%vl 7
%verb P
procedure
%verb GP
global procedure
%verb F
function
```

```
%verb GF
global function
%verb SP
specification segment
%verb TK
task body
%verb EP
task entry
%le
Global procedures, global functions, and task entries will have appeared in
Specification Segments.
Global procedures and global functions may also apppear later as segments.
In that case, the definition page given in the index is the page for the
segment, itself, and not the page for the Specification Segment.
%G Examples
%SPEC A Specification Segment
procedure first procedure (first arg, second arg) --defined later
procedure second procedure --not defined in this design

function first function () return word

record
aaa -- a data item
bbb -- another data item
end record

--other data not in a record
ccc
ddd

task first task
first entry --this is the only way to define entries
second entry (an argument)
end task

task another task --not defined in this design
another entry
end task
%P First Procedure (arg one, arg two)
while first function loop
set ccc to arg two
exit when time is up
end loop
case of some selection
when yellow
first procedure (aaa,ddd)
second procedure
when red =>
return
end case
%F First Function () return word
if something
set something to nothing
elseif something else
set something to nothing else
else
first procedure (ccc, ddd)
end if
%P A Procedure with an Exception
for all time
second procedure
end loop
exception
when overflow
set to largest number
when underflow
```

```
set to zero
%TASKBODY First Task
loop
select
when true
accept first entry
or when true
accept second entry (an in-out argument) do
incement argument by 5
end
or
delay for half an hour
else
nobody is responding
end select
end loop
%TASKBODY Second Task
loop
another entry
select
second entry (ddd)
else
first procedure (aaa,bbb)
end select
end loop
```

C.2 A Complete High-Level Design

This design presents a complete high-level design using the *ada* style.

C.2.1 Output of PDL/81 Processor

Beginning on the next page is the actual output of the PDL/81 processor when presented with the source shown in Section C.2.2.

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 1  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


                           CAINE, FARBER & GORDON, INC.
                              1010 EAST UNION STREET
                            PASADENA, CALIFORNIA 91106


                           ***************************
                           *                         *
                           * Aircraft Monitor System *
                           *                         *
                           *     A Sample Design     *
                           *                         *
                           *          Using          *
                           *                         *
                           *  Ada Tasking Facilities *
                           *                         *
                           *        21 Jan 92        *
                           *                         *
                           *     PDL/81 X2.0.911     *
                           *                         *
                           *        5500-PD8         *
                           *                         *
                           ***************************
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                        SHEET 2  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

CFG, INC.       AIRCRAFT MONITOR SYSTEM                                PAGE  1.001
21 Jan 92       TABLE OF CONTENTS
```

TABLE OF CONTENTS

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                        SHEET 2  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                        SHEET 3  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

CFG, INC.       AIRCRAFT MONITOR SYSTEM                                PAGE  2
21 Jan 92

                              ********************
                              *                  *
                              * Task Definitions *
                              *                  *
                              ********************
```


```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                        SHEET 3  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
CFG, INC.       AIRCRAFT MONITOR SYSTEM                               PAGE  3
21 Jan 92       Task Definitions
```

Task Connections

```
#############################################################################
#                                                                           #
#  1                                                                        #
#  2    command   <-------------------------------------- keyboard          #
#  3    executor                                              !             #
#  4     ! ! ! !                                  smoke       !             #
#  5     ! ! ! --------------------------------> detector---->!             #
#  6     ! ! !                                      !         !             #
#  7     ! ! !               ------> lights         !         !             #
#  8     ! ! !               !                      v         !             #
#  9     ! ! !             alarm                  smoke       !             #
# 10     ! ! -----------> handler <------<------ monitor      !             #
# 11     ! !                 !           !                    !             #
# 12     ! !                 !           !                    !             #
# 13     ! !                 !           !                    !             #
# 14     ! -----> display <---------- sensor----------------->!             #
# 15     !           !       !         poll <----    clock    !             #
# 16     !           !       !          !       !      !      v             #
# 17     !           !       !          !       !      ----> time           #
# 18     --> VDU  <--<--------          !      sensors      stamper         #
# 19        formatter                   v                     !             #
# 20          !                        dials                  !             #
# 21          !                                               v             #
# 22          v                                             recorder        #
# 23        screen                                                          #
# 24                                                                        #
#                                                                           #
#############################################################################
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                        SHEET 5  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       AIRCRAFT MONITOR SYSTEM                                 PAGE  4
21 Jan 92       Task Definitions


     Foreground Tasks

REF
PAGE SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
     S                                                                            S
  10 S          TASK command executor IS                                          S
     S          |   command keystroke (keystroke)                                 S
     S          END TASK                                                          S
     S                                                                            S
  12 S          TASK alarm handler IS                                             S
     S          |   acknowledge alarm (acknowledge code)                          S
     S          |   change alarm state (alarm number, state)                      S
     S          END TASK                                                          S
     S                                                                            S
  17 S          TASK display handler IS                                           S
     S          |   change display mode (display mode)                            S
     S          |   change display state (display state)                          S
     S          |   display sensor value (sensor number, value)                   S
     S          END TASK                                                          S
     S                                                                            S
  20 S          TASK VDU formatter IS                                             S
     S          |   clear VDU display                                             S
     S          |   display command on VDU (message)                              S
     S          |   display line on VDU (message)                                 S
     S          END TASK                                                          S
     S                                                                            S
     SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

     Requirements: 3.2.1, 3.8
```


```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                        SHEET 5  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                        SHEET 6  ||
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       AIRCRAFT MONITOR SYSTEM                                PAGE  5
21 Jan 92       Task Definitions


     Background Tasks

REF
PAGE SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
     S                                                                         S
  14 S        TASK sensor poll IS                                              S
     S        END TASK                                                         S
     S                                                                         S
  16 S        TASK smoke status monitor IS                                     S
     S        |   monitor smoke (detector number, state)                       S
     S        END TASK                                                         S
     S                                                                         S
  21 S        TASK time stamper IS                                             S
     S        |   time stamp (type, first value, second value)                 S
     S        |   change time (time)                                           S
     S        END TASK                                                         S
     S                                                                         S
     SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

     Requirements: 3.2.1, 3.8
```


```
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                        SHEET 6  ||
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                        SHEET 7  ||
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

CFG, INC.       AIRCRAFT MONITOR SYSTEM                                PAGE  6
21 Jan 92




                       ***********************************
                       *                                 *
                       * The Hardware Software Interface *
                       *                                 *
                       ***********************************




|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                        SHEET 7  ||
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
CFG, INC.       AIRCRAFT MONITOR SYSTEM                                PAGE  7
21 Jan 92       The Hardware Software Interface


     Hardware Operations


     #############################################################################
     #                                                                           #
     #  1     The aircraft monitor system has the following hardware             #
     #        interfaces:                                                        #
     #                                                                           #
     #  2     Sensors             Input from the sensors yields a value or no    #
     #                            response.                                      #
     #                                                                           #
     #  3     Smoke Detectors     An interrupt gives the detector number and     #
     #  4                         the new state.  Output to a smoke detector     #
     #                            puts it through a test cycle.                  #
     #                                                                           #
     #  5     Lights              Output to a light change it to green or        #
     #                            red.                                           #
     #                                                                           #
     #  6     Dials               Output to a dial change the value              #
     #                            displayed.                                     #
     #                                                                           #
     #  7     Keyboard            An interrupt gives the keystroke value.        #
     #                                                                           #
     #  8     VDU Screen          Output to the VDU changes the display.         #
     #                                                                           #
     #  9     Clock               An interrupt gives a new time.                 #
     #                                                                           #
     # 10     Recorder            Output to the recorder is saved.               #
     #                                                                           #
     # 11     The next page shows the interface between the hardware             #
     #        interrupts and the preceding tasks.                                #
     #                                                                           #
     #############################################################################
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 9  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       AIRCRAFT MONITOR SYSTEM                               PAGE  8
21 Jan 92       The Hardware Software Interface


     PROCEDURE Interrupt Actions

REF    (CX = 0)
PAGE PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
     P                                                                           P
     P  1    --keystroke interrupt                                               P
   4 P  2    command keystroke (keystroke)                                       P
   5 P  3    time stamp (key type, keystroke, null)                              P
     P  4    --                                                                  P
     P  5    --smoke interrupt                                                   P
   5 P  6    monitor smoke (number, state)                                       P
   5 P  7    time stamp (smoke type, state, number)                              P
     P  8    --                                                                  P
     P  9    --clock interrupt                                                   P
   5 P 10    change time (clock time)                                            P
     P                                                                           P
     PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP

     Requirements: 3.7, 3.14.2
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                        SHEET 10  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

CFG, INC.       AIRCRAFT MONITOR SYSTEM                                PAGE  9
21 Jan 92
```

```
***************
*             *
* Task Bodies *
*             *
***************
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 11  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       AIRCRAFT MONITOR SYSTEM                               PAGE  10
21 Jan 92       Task Bodies


     TASK BODY Command Executor

REF    (CX = 2)
PAGE TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
     T                                                                           T
     T          LOOP                                                             T
   4 T          |   ACCEPT command keystroke (keystroke)                         T
     T          |   determine command type and code from keystroke               T
     T          |   CASE of command type IS                                      T
     T          |   |   WHEN smoke test =>                                       T
     T          |   |       FOR all smoke detectors LOOP                         T
     T          |   |       |   output to detector (test command)                T
     T          |   |       END LOOP                                             T
     T          |   |   WHEN modes =>                                            T
   4 T          |   |       change display mode (command code)                   T
     T          |   |   WHEN acknowledgements =>                                 T
   4 T          |   |       acknowledge alarm (command code)                     T
     T          |   END CASE                                                     T
   4 T          |   display command on VDU (command name)                        T
     T          END LOOP                                                         T
     T                                                                           T
     TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT

     Requirements: 3.8, 3.12.5
```


```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 11  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
CFG, INC.       AIRCRAFT MONITOR SYSTEM                               PAGE  11
21 Jan 92       Task Bodies
```

```
     Commands

     ############################################################################
     #                                                                          #
     #  1    test commands                                                      #
     #  2            smoke test                                                 #
     #  3    acknowlegements                                                    #
     #  4            engine pressure warning                                    #
     #  5            engine temperature warning                                 #
     #  6            fuel level warning                                         #
     #  7    display modes                                                      #
     #  8            most recent readings                                       #
     #  9            sensor histories                                           #
     # 10            calculated values                                          #
     # 11            rates of change                                            #
     #                                                                          #
     ############################################################################

     Requirements: 3.4.5, 3.4.6, 3.7.2, 3.9
```

```
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 13  ||
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       AIRCRAFT MONITOR SYSTEM                               PAGE  12
21 Jan 92       Task Bodies


     TASK BODY Alarm Handler

REF    (CX = 4)
PAGE TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
     T                                                                            T
     T          LOOP                                                              T
     T          |   SELECT whichever is ready                                     T
   4 T          |   |   ACCEPT acknowledge alarm (acknowledge code)               T
     T          |   |   FOR all alarm flags LOOP                                  T
     T          |   |   |   IF the flag belongs to the acknowledged class THEN    T
     T          |   |   |   |   reset current alarm flag                          T
     T          |   |   |   END IF                                                T
     T          |   |   END LOOP                                                  T
  13 T          |   |   change alarms                                             T
     T          |   OR                                                            T
   4 T          |   |   ACCEPT change alarm state (alarm number, state)           T
     T          |   |   CASE of state IS                                          T
     T          |   |   |   WHEN present =>                                       T
     T          |   |   |       output to lights (alarm number, red)              T
     T          |   |   |       set alarm flag (alarm number)                     T
  13 T          |   |   |       change alarms                                     T
     T          |   |   |   WHEN absent =>                                        T
     T          |   |   |       output to lights (alarm number, green)            T
     T          |   |   END CASE                                                  T
     T          |   END SELECT                                                    T
     T          END LOOP                                                          T
     T                                                                            T
     TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT

     Requirements: 3.2.2.3, 3.4, 3.7.8




|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 13  ||
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
CFG, INC.      AIRCRAFT MONITOR SYSTEM                          PAGE  13
21 Jan 92      Task Bodies


     PROCEDURE Change Alarms

REF    (CX = 2)
PAGE PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
     P                                                                          P
     P        IF there are any alarms set THEN                                  P
   4 P        |   change display state (suspend)                                P
   4 P        |   clear VDU display                                             P
     P        |   FOR every alarm LOOP                                          P
   4 P        |   |   display line on VDU (message for alarm)                   P
     P        |   END LOOP                                                      P
     P        ELSE                                                              P
   4 P        |   clear VDU display                                             P
   4 P        |   change display state (resume)                                 P
     P        END IF                                                            P
     P                                                                          P
     PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP

     Requirements: 3.7.8, 3.4
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 15  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

CFG, INC.       AIRCRAFT MONITOR SYSTEM                               PAGE  14
21 Jan 92       Task Bodies

     TASK BODY Sensor Poll

REF    (CX = 2)
PAGE TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
     T                                                                           T
     T        LOOP                                                               T
     T        |   DELAY until the start of the second                            T
     T        |   FOR each sensor LOOP                                           T
     T        |   |   read from sensor                                           T
     T        |   |   IF the sensor did not give a reading THEN                  T
     T        |   |   |   set reading to default value                           T
     T        |   |   END IF                                                     T
     T        |   |   convert reading to degrees                                 T
     T        |   |   output value to circular display (sensor number, degrees)   T
   4 T        |   |   display sensor value (sensor number, reading)              T
   5 T        |   |   time stamp (sensor type, sensor number, reading)           T
  15 T        |   |   monitor sensor status (sensor number, reading)             T
     T        |   END LOOP                                                       T
     T        END LOOP                                                           T
     T                                                                           T
     TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT

     Requirements: 3.2.2, 3.4.4
```

	AIRCRAFT MONITOR SYSTEM UNCLASSIFIED SHEET 15	

```
CFG, INC.       AIRCRAFT MONITOR SYSTEM                             PAGE  15
21 Jan 92       Task Bodies


     PROCEDURE Monitor Sensor Status (sensor number, reading)

REF    (CX = 1)
PAGE PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
     P                                                                           P
     P  1    determine when three in a row condition is met for this sensor      P
     P  2    IF the sensor changed its state THEN                                P
   4 P  3    |   change alarm state (alarm number for sensor, new state)         P
     P  4    END IF                                                              P
     P                                                                           P
     PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP

     Requirements: 3.8.6, 3.8.7, 3.12
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 17  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       AIRCRAFT MONITOR SYSTEM                               PAGE  16
21 Jan 92       Task Bodies


     TASK BODY Smoke Status Monitor

REF    (CX = 3)
PAGE TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
     T                                                                           T
     T          LOOP                                                             T
   5 T          |   ACCEPT monitor smoke (number, state)                         T
     T          |   IF the state was smoke THEN                                  T
     T          |   |   set smoke for that detector number                       T
     T          |   |   IF smoke state is not set THEN                           T
     T          |   |   |   set smoke state                                      T
   4 T          |   |   |   change alarm state (smoke alarm number, present)     T
     T          |   |   END IF                                                   T
     T          |   ELSE --nosmoke                                               T
     T          |   |   set nosmoke for that detector number                     T
     T          |   |   IF smoke state is set and nosmoke is set for all detectors  T
     T          |   |   |       THEN                                             T
     T          |   |   |   reset smoke state                                    T
   4 T          |   |   |   change alarm state (smoke alarm number, absent)      T
     T          |   |   END IF                                                   T
     T          |   END IF                                                       T
     T          END LOOP                                                         T
     T                                                                           T
     TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT

     Requirements: 3.8.6, 3.8.7, 3.12
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 17  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 18  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
CFG, INC.       AIRCRAFT MONITOR SYSTEM                               PAGE  17
21 Jan 92       Task Bodies
```

```
     TASK BODY Display Handler

REF    (CX = 2)
PAGE TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
     T                                                                           T
     T        LOOP                                                               T
     T        |   SELECT whichever is ready                                      T
   4 T        |   |   ACCEPT change display state (new state)                    T
     T        |   |   CASE of new state IS                                       T
     T        |   |   |   WHEN suspended =>                                      T
     T        |   |   |       set suspend                                        T
     T        |   |   |   WHEN resumed =>                                        T
     T        |   |   |       reset suspend                                      T
  18 T        |   |   |       rebuild display                                    T
     T        |   |   END CASE                                                   T
     T        |   OR                                                             T
   4 T        |   |   ACCEPT change display mode (mode)                          T
     T        |   |   save display mode                                          T
  18 T        |   |   rebuild display                                            T
     T        |   OR                                                             T
   4 T        |   |   ACCEPT display sensor value (sensor number, sensor reading) T
     T        |   |   enter reading in the sensor history buffer                 T
  19 T        |   |   display one line                                           T
     T        |   END SELECT                                                     T
     T        END LOOP                                                           T
     T                                                                           T
     TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT

     Requirements: 3.8.6, 3.8.7, 3.12
```


```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 18  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                        SHEET 19  ||
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       AIRCRAFT MONITOR SYSTEM                                PAGE  18
21 Jan 92       Task Bodies


     PROCEDURE Rebuild Display

REF    (CX = 1)
PAGE PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
     P                                                                            P
     P  1    choose a starting place in the sensor history buffer                 P
     P  2    WHILE the current sensor entry is not the latest LOOP                P
  19 P  3    |   display one line                                                 P
     P  4    END LOOP                                                             P
     P                                                                            P
     PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP

     Requirements: 3.8.1, 3.12.12
```


```
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                        SHEET 19  ||
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
CFG, INC.       AIRCRAFT MONITOR SYSTEM                               PAGE  19
21 Jan 92       Task Bodies


     PROCEDURE Display One Line

REF    (CX = 1)
PAGE PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
     P                                                                            P
     P          RETURN if suspend is set                                          P
     P          --build a line with the appropriate information                   P
     P          CASE of display mode IS                                           P
     P          |   WHEN most recent readings =>                                  P
     P          |   WHEN sensor histories =>                                      P
     P          |   WHEN rates of change =>                                       P
     P          |   WHEN calculated values =>                                     P
     P          END CASE                                                          P
   4 P          display line on VDU (line just constructed)                       P
     P                                                                            P
     PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP

     Requirements: 3.9.1
```

```
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 21  ||
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       AIRCRAFT MONITOR SYSTEM                               PAGE  20
21 Jan 92       Task Bodies


     TASK BODY VDU Formatter

REF    (CX = 1)
PAGE TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
     T                                                                          T
     T        LOOP                                                              T
     T        |   SELECT whichever is ready                                     T
   4 T        |   |   ACCEPT display line on VDU (message)                      T
     T        |   |   write message at the bottom of the screen                 T
     T        |   OR                                                            T
   4 T        |   |   ACCEPT clear VDU display                                  T
     T        |   |   clear the screen except for the command line              T
     T        |   OR                                                            T
   4 T        |   |   ACCEPT display command on VDU (message)                   T
     T        |   |   write message on the command line                         T
     T        |   END SELECT                                                    T
     T        END LOOP                                                          T
     T                                                                          T
     TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT

     Requirements: 3.9.1
```


```
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 21  ||
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
CFG, INC.       AIRCRAFT MONITOR SYSTEM                             PAGE  21
21 Jan 92       Task Bodies


     TASK BODY Time Stamper

REF    (CX = 1)
PAGE TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
     T                                                                           T
     T        LOOP                                                               T
     T        |   SELECT whichever is ready                                      T
   5 T        |   |   ACCEPT time stamp (type, first value, sencond value)       T
     T        |   |   build timestamp record                                     T
     T        |   |   put the current time in the record                         T
     T        |   |   output the record to the recorder                          T
     T        |   OR                                                             T
   5 T        |   |   ACCEPT change time (time)                                  T
     T        |   |   save the new time                                          T
     T        |   END SELECT                                                     T
     T        END LOOP                                                           T
     T                                                                           T
     TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT

     Requirements: 3.9.1, 3.9.2
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 23  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

CFG, INC.       AIRCRAFT MONITOR SYSTEM                               PAGE  22
21 Jan 92

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



```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 23  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
INDEX TO FLOW SEGMENTS
----------------------


PAGE  LINE  TYPE CX NAME AND REFERENCES
----  ----  ---- -- -------------------

   4     6   EP      acknowledge alarm
                        12  Alarm Handler
                               3
                        10  Command Executor
                               12

  12         TK   4  Alarm Handler

   4     7   EP      change alarm state
                        12  Alarm Handler
                               11
                        15  Monitor Sensor Status
                               3
                        16  Smoke Status Monitor
                               7  13

  13         P    2  Change Alarms
                        12  Alarm Handler
                               9  16

   4    11   EP      change display mode
                        10  Command Executor
                               10
                        17  Display Handler
                               12

   4    12   EP      change display state
                        13  Change Alarms
                               2  9
                        17  Display Handler
                               3

   5    10   EP      change time
                         8  Interrupt Actions
                               10
                        21  Time Stamper
                               8
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 25  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       AIRCRAFT MONITOR SYSTEM                               PAGE  22.002
21 Jan 92       INDEX TO FLOW SEGMENTS


INDEX TO FLOW SEGMENTS  (continued)
-----------------------------------


PAGE  LINE  TYPE CX NAME AND REFERENCES
----  ----  ---- -- -------------------

   4    17   EP      clear VDU display
                        13  Change Alarms
                               3  8
                        20  VDU Formatter
                               6

  10         TK   2  Command Executor

   4     2   EP      command keystroke
                        10  Command Executor
                               2
                         8  Interrupt Actions
                               2

   4    18   EP      display command on VDU
                        10  Command Executor
                               14
                        20  VDU Formatter
                               9

  17         TK   2  Display Handler

   4    19   EP      display line on VDU
                        13  Change Alarms
                               5
                        19  Display One Line
                               9
                        20  VDU Formatter
                               3

  19         P    1  Display One Line
                        17  Display Handler
                               18
                        18  Rebuild Display
                               3

   4    13   EP      display sensor value
```


```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 25  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 26  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       AIRCRAFT MONITOR SYSTEM                               PAGE  22.003
21 Jan 92       INDEX TO FLOW SEGMENTS


INDEX TO FLOW SEGMENTS  (continued)
-----------------------------------


PAGE  LINE  TYPE CX NAME AND REFERENCES
----  ----  ---- -- ------------------

                        17  Display Handler
                               16
                        14  Sensor Poll
                               10

   8         P       Interrupt Actions

  15         P    1  Monitor Sensor Status
                        14  Sensor Poll
                               12

   5     5   EP      monitor smoke
                         8  Interrupt Actions
                               6
                        16  Smoke Status Monitor
                               2

  18         P    1  Rebuild Display
                        17  Display Handler
                               9  14

  14         TK   2  Sensor Poll

  16         TK   3  Smoke Status Monitor

   5     9   EP      time stamp
                         8  Interrupt Actions
                               3  7
                        14  Sensor Poll
                               11
                        21  Time Stamper
                               3

  21         TK   1  Time Stamper

  20         TK   1  VDU Formatter
```

```
CFG, INC.       AIRCRAFT MONITOR SYSTEM                               PAGE  23
21 Jan 92
```

```
                       ************************************
                       *                                  *
                       * INDEX TO REQUIREMENTS REFERENCES *
                       *                                  *
                       ************************************
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 28  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
CFG, INC.        AIRCRAFT MONITOR SYSTEM                            PAGE  23.001
21 Jan 92        INDEX TO REQUIREMENTS REFERENCES
```

INDEX TO REQUIREMENTS REFERENCES

REQUIREMENT	PAGE	SEGMENT NAME
3.2.1	5	Background Tasks
	4	Foreground Tasks
3.2.2	14	Sensor Poll
3.2.2.3	12	Alarm Handler
3.4	12	Alarm Handler
	13	Change Alarms
3.4.4	14	Sensor Poll
3.4.5	11	Commands
3.4.6	11	Commands
3.7	8	Interrupt Actions
3.7.2	11	Commands
3.7.8	12	Alarm Handler
	13	Change Alarms
3.8	5	Background Tasks
	10	Command Executor
	4	Foreground Tasks
3.8.1	18	Rebuild Display
3.8.6	17	Display Handler
	15	Monitor Sensor Status
	16	Smoke Status Monitor
3.8.7	17	Display Handler
	15	Monitor Sensor Status
	16	Smoke Status Monitor
3.9	11	Commands
3.9.1	19	Display One Line
	21	Time Stamper
	20	VDU Formatter
3.9.2	21	Time Stamper
3.12	17	Display Handler
	15	Monitor Sensor Status
	16	Smoke Status Monitor
3.12.5	10	Command Executor
3.12.12	18	Rebuild Display
3.14.2	8	Interrupt Actions


```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 28  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
*************************
*                       *
* CALLS-IN-CONTEXT LIST *
*                       *
*************************
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 30  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       AIRCRAFT MONITOR SYSTEM                               PAGE  24.001
21 Jan 92       CALLS-IN-CONTEXT LIST


CALLS-IN-CONTEXT LIST
---------------------


F PAGE  NAME / CALL
- ----  -----------

     4   acknowledge alarm (acknowledge code)
    10   acknowledge alarm (command code)
    12   accept acknowledge alarm (acknowledge code)

     4   change alarm state (alarm number, state)
    12   accept change alarm state (alarm number, state)
    15   change alarm state (alarm number for sensor, new state)
    16   change alarm state (smoke alarm number, present)
    16   change alarm state (smoke alarm number, absent)

    13   Change Alarms
    12   change alarms
    12   change alarms

     4   change display mode (display mode)
    10   change display mode (command code)
    17   accept change display mode (mode)

     4   change display state (display state)
    13   change display state (suspend)
    13   change display state (resume)
    17   accept change display state (new state)

     5   change time (time)
     8   change time (clock time)
    21   accept change time (time)

     4   clear VDU display
    13   clear VDU display
    13   clear VDU display
    20   accept clear VDU display

     4   command keystroke (keystroke)
     8   command keystroke (keystroke)
    10   accept command keystroke (keystroke)

     4   display command on VDU (message)
    10   display command on VDU (command name)
    20   accept display command on VDU (message)


||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 30  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 31  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


CFG, INC.       AIRCRAFT MONITOR SYSTEM                               PAGE  24.002
21 Jan 92       CALLS-IN-CONTEXT LIST


CALLS-IN-CONTEXT LIST  (continued)
----------------------------------


F PAGE  NAME / CALL
- ----  -----------

     4   display line on VDU (message)
    13   display line on VDU (message for alarm)
    19   display line on VDU (line just constructed)
    20   accept display line on VDU (message)

    19   Display One Line
    17   display one line
    18   display one line

     4   display sensor value (sensor number, value)
    14   display sensor value (sensor number, reading)
    17   accept display sensor value (sensor number, sensor reading)

     8   Interrupt Actions

    15   Monitor Sensor Status (sensor number, reading)
    14   monitor sensor status (sensor number, reading)

     5   monitor smoke (detector number, state)
     8   monitor smoke (number, state)
    16   accept monitor smoke (number, state)

    18   Rebuild Display
    17   rebuild display
    17   rebuild display

     5   time stamp (type, first value, second value)
     8   time stamp (key type, keystroke, null)
     8   time stamp (smoke type, state, number)
    14   time stamp (sensor type, sensor number, reading)
    21   accept time stamp (type, first value, sencond value)
```


```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||  AIRCRAFT MONITOR SYSTEM        UNCLASSIFIED                       SHEET 31  ||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

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

```
STATISTICS
----------

Maximum complexity measure (CX) is 4.
0 flow segments had a complexity greater than 6.

12 flow segments.

2370 lines in definition file(s).
312 lines in source file(s).

686 dictionary entries allocated.
2573 string segments allocated;  2412 in use.

73216 bytes of dynamic memory allocated.
```

C.2.2 Source Listing

The input lines which resulted in the design document of the preceding section are:

```
%*- -sada
%* @(#) $Header: ada2,v 2.0 88/03/27 16:09:18 shc Rel $
%Security UNCLASSIFIED
%Project AIRCRAFT MONITOR SYSTEM
%Title Aircraft Monitor System
%Title A Sample Design
%Title Using
%Title Ada Tasking Facilities
%NoSBox
%NoTree
%NoDIndex
%cic
%rindex
%complexity
%kwv |
%G Task Definitions
%T Task Connections

command   <-------------------------------------- keyboard
executor                                              !
 ! ! ! !                                  smoke       !
 ! ! ! --------------------------------> detector---->!
 ! ! !                                      !         !
 ! ! !               ------> lights         !         !
 ! ! !               !                      v         !
 ! ! !             alarm                  smoke       !
 ! ! -----------> handler <------<------ monitor      !
 ! !                 !           !                    !
 ! !                 !           !                    !
 ! !                 !           !                    !
 ! -----> display <---------- sensor----------------->!
 !           !       !         poll <----    clock    !
 !           !       !          !       !      !      v
 !           !       !          !       !      ----> time
 --> VDU  <--<--------          !      sensors      stamper
    formatter                   v                     !
      !                        dials                  !
      !                                               v
      v                                             recorder
    screen

%SPEC Foreground Tasks
%req 3.2.1; 3.8
task command executor
    command keystroke (keystroke)
end task

task alarm handler
    acknowledge alarm (acknowledge code)
    change alarm state (alarm number, state)
end task

task display handler
    change display mode (display mode)
    change display state (display state)
    display sensor value (sensor number, value)
end task

task VDU formatter
    clear VDU display
```

```
    display command on VDU (message)
    display line on VDU (message)
end task
%SPEC Background Tasks
%req 3.2.1;3.8
task sensor poll
end task

task smoke status monitor
    monitor smoke (detector number, state)
end task

task time stamper
    time stamp (type, first value, second value)
    change time (time)
end task
%G The Hardware Software Interface
%TF Hardware Operations
The aircraft monitor system has the following hardware interfaces:
%VL 20
%VERB  Sensors
Input from the sensors yields a value or no response.
%VERB  Smoke Detectors
An interrupt gives the detector number and the new state.
Output to a smoke detector puts it through a test cycle.
%VERB Lights
Output to a light change it to green or red.
%VERB Dials
Output to a dial change the value displayed.
%VERB Keyboard
An interrupt gives the keystroke value.
%VERB VDU Screen
Output to the VDU changes the display.
%VERB Clock
An interrupt gives a new time.
%VERB Recorder
Output to the recorder is saved.
%LE
The next page shows the interface between the hardware
interrupts and the preceding tasks.
%P Interrupt Actions
%req 3.7;3.14.2
--keystroke interrupt
command keystroke (keystroke)
time stamp (key type, keystroke, null)
--
--smoke interrupt
monitor smoke (number, state)
time stamp (smoke type, state, number)
--
--clock interrupt
change time (clock time)
%G Task Bodies
%TASKBODY Command Executor
%req 3.8;3.12.5

loop
        accept command keystroke (keystroke)
        determine command type and code from keystroke
        case of command type
            when smoke test
                for all smoke detectors loop
                        output to detector (test command)
                end loop
            when modes
                change display mode (command code)
```

```
            when acknowledgements
                acknowledge alarm (command code)
        end case
        display command on VDU (command name)
end loop

%T Commands
%req 3.4.5;3.4.6;3.7.2;3.9
test commands
        smoke test
acknowlegements
        engine pressure warning
        engine temperature warning
        fuel level warning
display modes
        most recent readings
        sensor histories
        calculated values
        rates of change
%TASKBODY Alarm Handler
%req 3.2.2.3;3.4;3.7.8

loop
        select whichever is ready
                accept acknowledge alarm (acknowledge code)
                for all alarm flags loop
                        if the flag belongs to the acknowledged class
                                reset current alarm flag
                        endif
                end loop
                change alarms
            or
                accept change alarm state (alarm number, state)
                case of state
                    when present
                        output to lights (alarm number, red)
                        set alarm flag (alarm number)
                        change alarms
                    when absent
                        output to lights (alarm number, green)
                end case
        end select
end loop


%P Change Alarms
%req 3.7.8;3.4

if there are any alarms set
        change display state (suspend)
        clear VDU display
        for every alarm loop
                display line on VDU (message for alarm)
        end loop
else
        clear VDU display
        change display state (resume)
end if


%TASKBODY Sensor Poll
%req 3.2.2;3.4.4

loop
        delay until the start of the second
        for each sensor loop
```

```
                read from sensor
                if the sensor did not give a reading
                        set reading to default value
                end if
                convert reading to degrees
                output value to circular display (sensor number, degrees)
                display sensor value (sensor number, reading)
                time stamp (sensor type, sensor number, reading)
                monitor sensor status (sensor number, reading)
        end loop
end loop

%P Monitor Sensor Status (sensor number, reading)
%req 3.8.6;3.8.7;3.12

determine when three in a row condition is met for this sensor
if the sensor changed its state
        change alarm state (alarm number for sensor, new state)
end if

%TASKBODY Smoke Status Monitor
%req 3.8.6;3.8.7;3.12

loop
        accept monitor smoke (number, state)
        if the state was smoke
                set smoke for that detector number
                if smoke state is not set
                        set smoke state
                        change alarm state (smoke alarm number, present)
                end if
        else --nosmoke
                set nosmoke for that detector number
                if smoke state is set and nosmoke is set for all detectors
                        reset smoke state
                        change alarm state (smoke alarm number, absent)
                endif
        end if
end loop


%TASKBODY Display Handler
%req 3.8.6;3.8.7;3.12

loop
        select whichever is ready
                accept change display state (new state)
                case of new state
                    when suspended
                        set suspend
                    when resumed
                        reset suspend
                        rebuild display
                end case
            or
                accept change display mode (mode)
                save display mode
                rebuild display
            or
                accept display sensor value (sensor number, sensor reading)
                enter reading in the sensor history buffer
                display one line
        end select
end loop
```

```
%P Rebuild Display
%req 3.8.1;3.12.12

choose a starting place in the sensor history buffer
while the current sensor entry is not the latest loop
        display one line
end loop


%P Display One Line
%req 3.9.1

return if suspend is set
--build a line with the appropriate information
case of display mode
        when most recent readings
        when sensor histories
        when rates of change
        when calculated values
end case
display line on VDU (line just constructed)


%TASKBODY VDU Formatter
%req 3.9.1

loop
        select whichever is ready
                accept display line on VDU (message)
                write message at the bottom of the screen
            or
                accept clear VDU display
                clear the screen except for the command line
            or
                accept display command on VDU (message)
                write message on the command line
        end select
end loop


%TASKBODY Time Stamper
%req 3.9.1;3.9.2

loop
        select whichever is ready
                accept time stamp (type, first value, sencond value)
                build timestamp record
                put the current time in the record
                output the record to the recorder
            or
                accept change time (time)
                save the new time
        end select
end loop
```

Index