HEWLETT
PACKARD

# Programmatic Access to HPWORD Documents

Manual Part No.
32119-90001 E0986

# Notice

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

# Printing History

First Edition - October 1983
Second Edition - July 1985
Third Edition - September 1986

The text in this book was formatted using TDP/3000 and printed on an HP2688A laser printer.

# Preface

The HPWORD Intrinsic Set gives the applications programmer comprehensive and detailed access to the textual and structural components of HPWORD documents. By using the HPWORD Intrinsics in SPL or PASCAL/3000 programs, you can transcribe the contents of HPWORD documents into other word-processing systems or for display on devices not supported by HPWORD. Conversely, you can translate documents created in other text-processing systems into HPWORD documents. In either case, the precise access to the structural features of HPWORD allows you to control the layout and print enhancements of the document, and to retain the originator's format or to impose your own.

This manual provides you with all the information you need to implement the HPWORD Intrinsic Set in your programs. It is assumed that you have a sound knowledge of the HPWORD product, which is described in the manual *Using HPWORD* (Part No: 32120-90035) (for HPWORD Version A.05.00).

This manual is organized as follows:

Chapter 1 *Introduction* presents an overview of the capabilities of the HPWORD Intrinsics, and of their hardware and software requirements.

Chapter 2 *Using the Intrinsics* introduces a model of the underlying concepts of HPWORD document structure as seen by the programmer. The HPWORD intrinsics are listed with a brief description of the function of each, and the way in which the intrinsics are used within the model is discussed.

Chapter 3 *Intrinsic Descriptions* forms the main body of the manual and comprises a formal definition of each intrinsic and its parameters. Wherever necessary, the formal description is followed by a text discussion which clarifies important points on the working of the intrinsic.

Chapter 4 *Error Messages* describes the generation of error and warning messages by the intrinsics and provides a complete list of those messages.

The three appendixes contain a list of the HPWORD default values and example programs illustrating the use of the intrinsics in both SPL and PASCAL/3000.

# Table of Contents

## 1 Introduction

## 2 Using the Intrinsics

## 3 Intrinsic Descriptions

## 4 Error Messages

## Appendix A Document Default Values

## Appendix B SPL Programming Example

## Appendix C PASCAL Programming Example

## Index

# Chapter 1

# Introduction

# Introduction

## Objectives of the HPWORD Intrinsics Set

The HPWORD Intrinsics were developed to meet the following objectives:

- Programmatic conversion of the text and structural characteristics of a document created in another word-processing convention to HPWORD.
- Programmatic conversion of the text and structural characteristics of an HPWORD document to another word-processing system.
- Programmatic merging of text, data or graphics into documents which can then be edited or printed with HPWORD.
- Programmatic editing of HPWORD documents, without the use of HPWORD supported terminals or printers.

## Features of the HPWORD Intrinsics Set

To meet the above objectives, the HPWORD Intrinsic set provides the facilities to carry out the following tasks programmatically:

- The creation of new HPWORD document files and the deletion of existing ones.
- Positioning to any specified page and paragraph within an HPWORD document.
- Reading an HPWORD document sequentially, line by line.
- Writing to an HPWORD document sequentially, line by line.
- Deletion of existing whole paragraphs in HPWORD documents.
- Creation of empty paragraphs within HPWORD documents, for subsequent line-by-line writing.
- Setting and deletion of user page breaks at paragraph markers within HPWORD documents.
- Adjusting of text, hyphenation and pagination of HPWORD documents.

# The Environment

## Hardware

The HPWORD Intrinsics will run on any HP3000 system with a minimum configuration. You do not need an HPWORD word-processing terminal on the system to gain programmatic access to HPWORD documents.

## Software

The intrinsic set runs under MPE-V or later version.

If you want to use the intrinsics to access an HPWORD document with any form of write facility, HPWORD (Product No: 32120) does not have to be already installed on the system. However, if HPWord is installed, then it must be version A.02.00 or later. If you are using version A.01.02 of the intrinsics, then again, it is not necessary for HPWord to be installed, but if it is, it must be version A.05.00 or later.

The supported languages that can call the HPWORD Intrinsics are SPL/3000 (Product No: 32100) and PASCAL/3000 (Product No: 32106). Either or both these subsystems must be installed.

When converting a document from another word-processing system, the document must be resident as an MPE file on the HP3000. When you use the intrinsics to create an HPWORD document, it is created as an MPE file on the HP3000.

Only documents created in HPWORD A.01.00 or later version can be accessed by the intrinsics.

# Chapter 2

# Using the Intrinsics

# Using the Intrinsics

## Concepts

This section reinforces your knowledge of the structure of HPWORD and describes in general terms the way the intrinsics are used.

## The Document Model

An HPWORD document exists as an MPE file and is made up of the following components:

- The Document Information object. There is one per document. It contains general information about the document.
- The Print Format object. There is one per document. It contains the print specification for the document.
- The Document Format object. There is one per document. It contains the document format characteristics which are applied throughout the document, except where they are overridden locally by the paragraph format.
- The Paragraph object. This comprises two parts:
  - The paragraph format. There is one per paragraph. This contains the formatting characteristics for the paragraph. Within the paragraph to which it relates, it overrides the document format. Its effect is local to its paragraph
  - The paragraph contents. There is one per paragraph, and it may define the contents as one of several different types, such as header, text or figure space.

Usually there will be multiple paragraph objects in a document; one for each paragraph in the document.

An HPWORD document comprises one each of Document Information, Print Format and Document Format objects, together with a sequentially ordered set of Paragraph objects.

Every document has a default header paragraph and and a default footer paragraph which cannot be changed, although its effect may be modified by the subsequent insertion of a user-defined header or footer respectively. The default header and footer paragraphs are the first paragraphs in the document, conforming to HPWORD's rule that headers and footers precede the text to which they apply. If you create a new document using the HPWORD intrinsics, the intrinsics will establish these two paragraphs for you. Their format is shown in Appendix A *Document Default Values.*

An "empty" document within HPWORD actually contains a null text paragraph, which follows the default header and footer paragraphs. If an "empty" document is opened by a program using the HPWORD intrinsics, the null paragraph is deleted for the convenience of the programmer. When an empty document is closed by the intrinsics, a null paragraph is inserted for HPWORD, which cannot recover a truly empty document.

# List of Intrinsics

Below is a list of the intrinsics together with a brief description of the function that each performs.

| | |
|---|---|
| WOPEN | Opens a document |
| WCLOSE | Closes a document |
| WMARKPARAS | Sets up direct access positions |
| WMOVEINDOC | Moves to a new document position |
| WERRORINFO | Gets error information |
| WGETDOCINFO | Gets general document information |
| WGETPRINTINFO | Gets current printing parameters |
| WSETPRINTINFO | Sets new printing parameters |
| WGETDOCFORMAT | Gets global document format information |
| WSETDOCFORMAT | Sets new global document format |
| WDELETEPARA | Deletes current paragraph |
| WGETALLPARAF | Positions to & gets complete paragraph format |
| WSETALLPARAF | Sets a new paragraph format with complete control of format |
| WGETPARAFORMAT | Positions to & gets most of next paragraph's format |
| WSETPARAFORMAT | Sets a new paragraph format |
| WREADFIGSPACE | Gets figure space parameters |
| WWRITEFIGSPACE | Sets figure space parameters |
| WREADLINE | Reads a line of text from a paragraph |
| WWRITELINE | Writes a line of text to a new paragraph |
| WSETPAGEBREAK | Sets a user page break |
| WDELETEPAGEBREAK | Deletes a user page break |
| WVERSION | Gets the version number of the HPWORD intrinsics |

Note

The WGETPARAFORMAT and WSETPARAFORMAT intrinsics required new parameters to support HPWORD A.05.00, and to prevent code changes, two new intrinsics were created to supersede them: WGETALLPARAF and WSETALLPARAF. All four intrinsics are supported. In discussions of the Document Model, read WGETPARAFORMAT to also mean WGETALLPARAF, and WSETPARAFORMAT to also mean WSETALLPARAF.

## Model Constraints

The Document Information object is set up at create time, when the user has some control over its contents. It cannot be altered directly, but is automatically updated at each WOPEN. It can be read at any time using WGETDOCINFO.

The Print Format object may be accessed independently of the other objects, and read at any time using WGETPRINTINFO. If you have specified write access you can modify this object using WSETPRINTINFO.

The Document Format object may be accessed and read independently of the other objects, at any time. If you have write access to the document, it may be altered using WSETDOCFORMAT.

The Paragraph object is accessed by positioning to it and then either reading or writing data from/to it, or deleting it. Paragraphs may be created by inserting a new one after the current document position. A paragraph may not be written to and read from at the same time. Whichever access (read or write) to the paragraph is chosen, the paragraph format must be accessed first, then the paragraph data. Note that the default header and footer paragraphs may not be modified in any way although they may be read or superseded by user-defined header and footer paragraphs.

If a paragraph has a pagebreak in it, then that information will be returned when the line preceding the pagebreak is accessed. You can specify a Pending User pagebreak to follow any line that you write. The paragraph you are writing will then be terminated, complying with HPWORD's rule that a Pending User pagebreak can be inserted only at a Paragraph Marker. You can add or delete User pagebreaks at the end of an existing paragraph without accessing the rest of the data in the paragraph. You can specify that the document is to be paginated when you close it with WCLOSE.

The text of paragraphs inserted into the document will not be adjusted or hyphenated until the document is closed.

# Working within the Model

This section discusses the way in which the HPWORD intrinsics can be used to work in, and manipulate, the model.

# The Current Document Position

One of the most important concepts involved in the use of the intrinsic set is that of the Current Document Position. This is, in effect, a logical paragraph pointer which may be moved from paragraph to paragraph within a document. It points at the first part (paragraph format) of the paragraph object and does not move within the paragraph during subsequent operations. It can point only to paragraphs in a document. The Current Document Position neither affects, nor is affected by, your access to the other document objects.

It is impossible to institute any read or write operations until you have established an appropriate Current Document Position.

Whenever you open an HPWORD document with WOPEN, the Current Document Position is set at the first paragraph in the document. This is always the default header which you cannot modify; nor can you insert a new paragraph adjacent to it. The next paragraph will always be the default footer and this may not be modified. You can position to and read from these two paragraphs.

Once established the Current Document Position may be moved by use of any of the following intrinsics:

- WMOVEINDOC. This establishes a new Current Document Position at the specified paragraph.

  Note that WMOVEINDOC has two modes of operation. You can request a relative move, forward or backward through the document by a specific number of paragraphs, or you can specify a direct move to a paragraph previously defined as an edit position. The intrinsic WMARKPARAS allows you to define chosen paragraphs as edit positions at any time between WOPEN and the first transfer of data.

- WDELETEPARA. This intrinsic deletes the paragraph at the Current Document Position and moves that current position to the following paragraph. If the deleted paragraph was the last in the document, the Current Document Position is established at the preceding paragraph.

- WSETPARAFORMAT. This intrinsic inserts or appends a new paragraph object following the paragraph at the Current Document Position. The Current Document Position is re-established at the new paragraph object. This is necessary if you wish to write to the document, since you can neither overwrite nor append lines to an existing paragraph.

- WGETPARAFORMAT. A call to this intrinsic will have one of three effects dependent on previous activity and Current Document Position.

  - Since you must read the paragraph format before you can read or delete a paragraph, the first call to WGETPARAFORMAT at any Current Document Position returns the format parameters of the paragraph object at that position. The Current Document Position does not change.

  - Generally, a second call to WGETPARAFORMAT at any Current Document Position will produce a move to the next paragraph object, where the a new Current Document Position will be established. The paragraph format parameters at the new position will be returned. However there is an exception;

  - If a second call to WGETPARAFORMAT is made when the Current Document Position is at the last paragraph of a document, an error message is generated, since you cannot position beyond the end of the document.

  In each of these three cases, if the new Current Document Position has been established by a call to WGETPARAFORMAT, that call counts as the first at the new position.

## Text Lines and Intrinsics

"Text line" has two distinct meanings in the context of the HPWORD intrinsics.

- When you use WWRITELINE to write to an HPWORD document, the lines written to the intrinsic are nominal, in the sense that they will be the length of a line read from a source other than the HPWORD document. Commonly, the length of such a line will differ from that of the document or paragraph format established in the HPWORD document. The line retains its input length until WCLOSE is called. When WCLOSE is called, WCLOSE invokes HPWORD to "adjust" the document. This process is document-wide but its effect in any individual paragraph is the same as that produced by the ADJUST key, or by leaving the paragraph, after entering or editing text in an HPWORD document. That is, the line endings and hyphenation are rearranged to fit within the established document and paragraph formats.
- When an HPWORD document is being read by WREADLINE "lines" of text are returned. If no calls to WWRITELINE have been made since the document was last opened, the lines are the adjusted lines that an HPWORD user would see. However, if lines have been written using WWRITELINE, since the document was opened, and WREADLINE is called to read the same lines, those lines will be returned unadjusted, exactly as they were input.

## Pagination

HPWORD features automatic pagination and generates automatic (system) pagebreaks according to the rules described under *Pagination* in the book *Using HPWORD.* System pagebreaks may lie within, or at the end of, a paragraph. If an HPWORD document has been edited and not subsequently paginated, system pagebreaks may occur as the sole contents of an otherwise null paragraph object.

In addition to system pagebreaks, the user may specify user pagebreaks, which must be placed at the end of paragraphs. User pagebreaks are classified as pending until the document is paginated. They then become active.

Both system and user pagebreaks are associated with positions within the document and are not linked directly to the paragraphs within which they lie, or to which they are appended. When a paragraph is deleted, the pagebreaks are retained until the document is paginated. When paginated the system pagebreaks are adjusted; user pagebreaks remain until they are deleted by the operator.

If a document is to be printed interactively with HPWORD, it must be paginated at WCLOSE to activate the header and footer paragraphs.

# Operating on the Model

## The Main Steps

Generally the process of working on an HPWORD document will involve the following steps:

- Open the document (WOPEN).
- Establish edit positions (WMARKPARAS).
- Access the document format object (WGETDOCFORMAT, WSETDOCFORMAT).
- Move explicitly to a new Current Document Position (WMOVEINDOC, WGETPARAFORMAT).
- Access (append, insert, delete, read) a paragraph object and paragraph format (WGETPARAFORMAT, WSETPARAFORMAT, WDELETEPARA).
- Access the paragraph content (read or write, locate and establish pagebreaks) on a text line or figure space basis (WREADLINE, WWRITELINE, WREADFIGSPACE, WWRITEFIGSPACE, WSETPAGEBREAK).
- Remove a system or user pagebreak and establish a pending user pagebreak, on a paragraph basis (WSETPAGEBREAK, WDELETEPAGEBREAK).
- Close the document; paginate, if required, or purge the document (WCLOSE).

Information applicable globally to the document may be accessed at any time, without affecting the Current Document Position. Use the following intrinsics:

| | |
|---|---|
| WGETDOCINFO | to access the Document Information object |
| WGETPRINTINFO<br>WSETPRINTINFO | to access the Document Print object |
| WGETDOCFORMAT<br>WSETDOCFORMAT | to access the Document Format Object |

# Programming

Before you start to program with the HPWORD Intrinsics, you should have a thorough understanding and knowledge of the contents of the *Using HPWORD* and some user experience of HPWORD.

The intrinsic declarations for the HPWORD intrinsic set are contained in the intrinsic file, INTRDEC.PUB.SYS, prepared by the program BUILDINT.PUB.SYS, and the source file INTRDECS.PUB.SYS.

An MPE message catalog file, INTRCAT.PUB.SYS, contains text messages describing all intrinsic error and status conditions. These messages are accessed by the intrinsic WERRORINFO, which acts on the STATUS parameter described below.

The HPWORD intrinsics may be called from either SPL or PASCAL/3000 application programs. If a version of HPWord is already installed on the same system as the program calling the intrinsics, then the version of HPWord must be A.02.00 or later if write access is required.

You must declare the intrinsics in your application program before you can call them. The simplest way of declaring the intrinsics is as follows:

```
....................................
....................................
....................................
INTRINSIC (INTRDEC.PUB.SYS) WOPEN,......,WCLOSE
....................................
....................................
....................................
```

The intrinsics reside in the System Library (SL.PUB.SYS), and references to them are satisfied automatically at run-time, in the same way as the other MPE intrinsics.

The user's application must be "PREPARED" with Process Handling capability.

## The Data Transfer Process

Data is passed between your application program and the intrinsics called by it, by means of the declared formal parameters. Your application program must set up an area of storage called the Document File Control Block (DFCB), which should be 2000 words long, for each open document. This is used internally by every intrinsic, and is passed to each intrinsic when it is called. The application program must not attempt to access the DFCB under any circumstances. Any such attempt will result in any subsequent intrinsic call returning a request for the document to be WCLOSEd.

## Status, Errors and Warnings

All status, error and informational communication is via the STATUS parameter which is passed back by all the intrinsics. Condition codes are not used and the condition code value is undefined after exiting an intrinsic. The intrinsic WERRORINFO is used to interrogate the STATUS parameter, and to interpret it by reference to the message catalog file, INTRCAT.PUB.SYS, using the primary status and intrinsic identity as indices. WERRORINFO then returns a buffer holding the text message and a length of text count.

A complete description of the STATUS parameter, and of the error and warning messages which it generates, is in Chapter 4 *Error Messages*.

## Recoverability

As soon as a document has been opened successfully for any kind of write access, it will be marked as being dirty. This will ensure that should the system or application crash, the document file will be recovered automatically by the intrinsics or HPWORD next time it is accessed. When the document is WCLOSEd the dirty flag is cleared. The maximum amount of data that could be lost due to a system or application crash while a document is open will be limited to the paragraph resident in the DFCB at the time of the crash.

# Chapter 3

# Intrinsic Descriptions

# Intrinsic Descriptions

This chapter contains detailed descriptions of all the HPWORD intrinsics in the sequence of four functional groups:

## Catalog of Intrinsics

Use of intrinsics marked by "*" will result in the document file being modified.

# Layout and Conventions

Each intrinsic description includes the following information:

- The intrinsic name, a summary of its function and the number of the intrinsic. (The number is significant for error diagnosis only; see Chapter 4 *Error Messages*)

- The intrinsic call descriptions in the format shown here for the WCLOSE intrinsic:

```
           LA     IA       LV         LV
WCLOSE  (DFCB, status, paginate, disposition);
```

  Required parameters are shown underlined; optional parameters are not. Capitals indicate that you should type the word exactly as it appears, whereas lower case indicates that the word is a generic term and should be replaced with the correct value. Above each parameter one of the following abbreviations indicates the type of the parameter:

| | |
|---|---|
| BA | Byte array |
| DA | Double array |
| I | Integer |
| IA | Integer array |
| IV | Integer by value |
| L | Logical |
| LA | Logical array |
| LV | Logical by value |
| R | Real |

- Parameters. Each parameter is described. Each parameter name is shown in the same typeface as in the intrinsic call. Its type and status, required or optional, is emphasized in the accompanying description.

- Text Discussion. The use of the intrinsic and its parameters, and any special considerations are discussed.

# WOPEN

## Intrinsic Number 11

Opens an HPWORD format document file.

```
OV       LA      IA      BA       LV        LV           LV
WOPEN  (DFCB, status, doc'name, domain, access'type, doc'format,
           BA          LV           BA           BA
        cat'name, doc'language, doc'creator, doc'author
            BA          IV
        doc'subject, doc'size);
```

## Parameters

DFCB

*Logical Array (Required)*
Document File Control Block.
A document file control block (2000 words long) is unique to each document open in HPWORD Intrinsics.

status

*Integer Array (Required)*
3 word intrinsic status and error information.

doc'name

*Byte array (Required)*
Contains a string of ASCII characters interpreted as a formal MPE file designator. The string must begin with a letter, contain alphanumeric characters, slashes, or periods and terminate with any non-alphanumeric character except a slash or a period. The string may name a User-predefined file in which case it must be preceded by an asterisk. (Max array size = 40 bytes)

domain

*Logical by value (Optional)*
File domain
0 = New
1 = Old
*default:* New.

# WOPEN continued

`access'type`

*Logical by value (Optional)*
Type of access to be allowed on this occasion of opening `doc'name`:
0 = Read
1 = Read/write
2 = Expanded Read Mode
This places the appropriate number of spaces where there is tabbing or indentation. See WREADLINE.
*default*: Read.

`doc'format`

*Logical by value (Optional)*
The document data structure type
1 = HPWORD
*default*: HPWORD.

`cat'name`

*Byte array (Optional)*
Contains the formal MPE file designator of the catalog file to be opened by WOPEN for subsequent access by the WERRORINFO intrinsic. (The supplied catalog is INTRCAT.PUBSYS)
(Max array size = 40 bytes)
*default*: No catalog file opened.

`doc'language`

*Logical by value (Optional)*
Document language (if available on system)

```
 1 = american'english
 2 = british'english
 3 = canadian'french
 4 = german
 5 = french
 6 = swedish
 7 = finnish
 8 = spanish
 9 = italian
10 = dutch
11 = norwegian
12 = danish
13 = south'african
```

*default*: `american'english`.

`doc'creator`

*Byte array (Optional)*
An ASCII string of thirty (non-control) characters that gives the name of the operator creating the document.
*default*: user's logon "user,group.account".

# WOPEN continued

`doc'author`

*Byte array (Optional)*
An ASCII string of thirty (non-control) characters that gives the name of the author of the document.
*default*: ASCII spaces.

`doc'subject`

*Byte array (Optional)*
An ASCII string of sixty (non-control) characters that describe the document's subject matter.
*default*: ASCII spaces.

`doc'size`

*Integer by value (Optional)*
Indicates the size of a new document.
80: *approx* 80 pages worth
600: *up to* 600 pages worth
See discussion on next page.
*default*: ~ 80 pages worth.

## Text Discussion

WOPEN opens an HPWORD format document file for subsequent access by other HPWORD intrinsics and establishes the Document File Control Block (DFCB).

A version of HPWord is used to perform hyphenation and text adjustment. If HPWORD is already installed on the system, then it must be version A.02.00 or later if read/write access is required. If the internal structure of a document is found to be incorrect, then in read/write mode the WOPEN intrinsic will attempt to call HPWORD to recover the document. Access to the document will be denied if the mode is read only or if the recovery process does not succeed. The HPWORD global RIN will be used to synchronize document read only access by different processes if HPWORD is installed on the system.

A template file is not specified to create a new document. WOPEN will set up a new document with the HPWORD default values for the document objects and with the HPWORD default header and footer paragraphs (see Appendix A *Document Default Values*). The document format of a document may be overwritten in read/write access by use of the WSETDOCFORMAT intrinsic.

doc'size. By default the Intrinsics will follow the same convention as HPWORD and create a document with filesize and extent attributes sufficient for approximately 80 pages of text. The actual capacity is difficult to specify as it depends on such factors as the number of blank lines in the text. HPWORD handles the expansion of a document that reaches its limit by creating a document of greater capacity and copying the text into it. The Intrinsics cannot do this.

If a write Intrinsic reaches that limit, it will return an error. So, if a very large document is to be created and filled using the Intrinsics, the larger doc'size should be specified. If an existing document is to be modified using the Intrinsics, and is likely to require expansion, it should be copied into a larger document created by calling WOPEN with the greater doc'size. Note that the default doc'size is sufficient for most documents and that the larger document incurs considerable overhead.

WOPEN accesses the HPWORD configuration file LANGCONF.WP.HPOFFICE to determine which languages are available on the system. A document can be created only in an available language. Read/write access will be denied to an existing document in an unconfigured document.

Note that the last 5 parameters doc'language.......doc'size have no significance unless the document is being created rather than edited.

## File Access Matrix

| | | ACCESS TYPE | |
|---|---|---|---|
| | | EXPANDED READ or READ ONLY | READ/WRITE |
| FILE DOMAIN | NEW | INVALID | READ/WRITE |
| | OLD | READ | READ/WRITE |
| ACCESS SCOPE | | SHARED | EXCLUSIVE |
| DOCUMENT RECOVERY ATTEMPTED | | NO | YES, WHEN REQUIRED |

# WCLOSE

## Intrinsic Number 12

Closes an HPWORD format document file

```
 OV       LA     IA        LV         LV
WCLOSE (DFCB, status, paginate, disposition);
```

## Parameters

DFCB

*Logical array (Required)*
Document File Control Block.

status

*Integer array (Required)*
3 word intrinsic status and error information.

paginate

*Logical by value (Optional)*
Specifies whether or not the document is to be paginated before being closed
0 = Do not paginate
1 = Paginate
*default*: Do not paginate.

disposition

*Logical by value (Optional)*
File disposition after being closed
0 = Save Permanent
1 = Purge
*default*: Save Permanent.

# WCLOSE continued

## Text Discussion

WCLOSE terminates access to an HPWORD format document file and closes it. This intrinsic must be called to ensure that all text is written back to the document file.

If any writing has taken place to the document, or the document format has been altered, then WCLOSE will automatically call HPWORD to adjust and hyphenate the whole document according to the paragraph and document format selections in force. Pagination may be requested if the document was opened in read/write mode, in which case HPWORD will be called to perform the operation.

If WCLOSE is not called then document integrity cannot be guaranteed even if the document was opened for read only access, or for read/write access and no modification has occurred.

Note that WCLOSE closes the message catalog (opened by WOPEN, cat'name).

## Options Available at WCLOSE Time

| | ACCESS TYPE | |
|---|---|---|
| | EXPANDED READ or READ ONLY | READ/WRITE |
| PAGINATION | NOT AVAILABLE | ON REQUEST |
| DISPOSITION | SAVE ONLY | SAVE OR PURGE |
| ADJUSTMENT | NO ADJUST | AUTOMATIC, IF DOCUMENT MODIFIED |

# WMARKPARAS

## Intrinsic Number 13

Establishes a set of positions within the document for subsequent direct access.

```
                LA      IA          IV           DA
WMARKPARAS  (DFCB, status, num'positions, positions);
```

## Parameters

DFCB

*Logical array (Required)*
Document File Control Block.

status

*Integer array (Required)*
3 word intrinsic status and error information.

num'positions

*Integer by value (Required)*
The number of document access positions specified in the positions parameter. A maximum of 25 positions can be specified.

positions

*Double array (Required)*
Defines a set of document edit positions to be used for direct access within the document file by calling the WMOVEINDOC intrinsic. Each position is defined by a double word element in the array as follows:

First word: Page number; a value of 0 specifies the last page.

Second word: Paragraph number; a value of 0 specifies the last paragraph on the given page.

Page and paragraph numbers start at 1, not 0.

# WMARKPARAS continued

## Text Discussion

WMARKPARAS is used to set up a series of edit positions within a document for subsequent direct access with the WMOVEINDOC intrinsic.

A position is defined by a page and paragraph number pair. Each position then marks a paragraph and will not change unless the paragraph is deleted. If the paragraph is deleted then the position moves to the following paragraph. However, if the deleted paragraph was at the bottom of the document the position stays at the new bottom paragraph. When all paragraphs have been deleted the document edit positions are all lost and do not reappear with the addition of more paragraphs. Since the positions refer to actual paragraphs rather than remaining as (page,para) indexes, the addition and deletion of pagebreaks or paragraphs has no effect on them.

If a paragraph continues over from one page and onto the next, then for the purpose of defining edit positions it is considered to be both the last paragraph on the previous page and the first paragraph on the next page. If a number given for a page or paragraph exceeds the number of pages in the document or the number of paragraphs on a page, an error condition arises and the edit position is not defined. When the page number is given as 0 then it is taken to mean the last page. When the paragraph number is given as 0, then it taken to refer to the last paragraph on the given page. The default header and footer are not included in the count as the first two paragraphs on the first page. In other words the designation "page 1, paragraph 1," will mark the first user-inserted paragraph on the first page of the document. That paragraph may be any one of the five types described under WGETPARAFORMAT.

WMARKPARAS may be called at any time after WOPEN until an intrinsic in the document data transfer group has been called. The set of positions defined by this intrinsic may be referenced at any point through WMOVEINDOC. If there are N positions defined then they are labeled from 1 to N, in the order in which they were specified. WMARKPARAS does not require the positions to be ordered in increasing page and para number.

As well as the user-defined edit positions, WMOVEINDOC provides two more edit positions. These are defined as:

- Position -1, the first paragraph in the document which will always be the default header.
- Position 0, the last paragraph in the document which will be either the last text paragraph or, if there are no text paragraphs, the default footer.

# WMOVEINDOC

## Intrinsic Number 14

Moves to a specified position within an HPWORD document file.

```
                  LA      IA        LV           IV
WMOVEINDOC (DFCB, status, move'mode, move'number);
```

## Parameters

DFCB

*Logical array (Required)*
Document File Control Block.

status

*Integer array (Required)*
3 word intrinsic status and error information.

move'mode

*Logical by value (Required)*
Defines the type of move to be made:

1 = **forward** through the document
2 = **backward** through the document
3 = **position** to a direct access edit position

move'number

*Integer by value (Required)*
Dependent on selection in move'mode, the number of paragraphs to move forward/backward, or the direct access edit position.

| move'mode | move'number Range | |
|---|---|---|
| forward/backward | 0 .. positive integer | |
| position | -1 | (first paragraph) |
| | 0 | (last paragraph) |
| | 1..N | an edit position defined by WMARKPARAS |

## Text Discussion

WMOVEINDOC establishes a new Current Document Position. Two methods are available:

1 Move Direct to Edit Position. After WOPEN, up to 25 edit positions may be specified by page and paragraph number using WMARKPARAS. Thereafter, those positions may be referred to by number in WMOVEINDOC and so used to provide a direct move facility.

2 Move Relative to Current Document Position. The move is specified as a number of paragraphs forward or backward, relative to the Current Document Position. This move may be made whether or not edit positions have been established by WMARKPARAS.

position. In addition to the edit positions specified by the user, direct access is provided to both the top and the bottom of the document by specifying move'number -1 or 0 respectively. The top of the document is always the default header paragraph.

forward/backward. If the relative move would take the current pointer outside of the document, the pointer is set to the default header or to the last paragraph in the document as appropriate and a warning is returned.

Note that a relative move of zero paragraphs can be specified. This allows the Current Document Position to be reset to the current paragraph, for example after a series of WREADLINEs.

WMOVEINDOC may be called at any time after WOPEN except for the period between requesting a new paragraph (WSETPARAFORMAT) and writing to that paragraph (WWRITELINE or WWRITEFIGSPACE). A call to WMOVEINDOC may terminate a series of WWRITELINEs.

There is no requirement that the programmer move sequentially through either the document or the edit positions.

# WERRORINFO

## Intrinsic Identity 21

Returns a text message associated with an intrinsic status return.

```
              LA     IA         IA          BA          I
WERRORINFO (DFCB, status, intr'status, message, msg'length);
```

## Parameters

DFCB
*Logical array (Required)*
Document File Control Block.

status
*Integer array (Required)*
3 word intrinsic status and error information.

intr'status
*Integer array (Required)*
The 3 word status array returned from the last call to an HPWORD Intrinsic

message
*Byte array (Required)*
Message string associated with intr'status. This consists of an ASCII string in the format discussed below. (72 bytes)

msg'length
*Integer (Required)*
Returns the length in bytes of the string in message
Range: 8 to 72 bytes.

## Text Discussion

Whenever a call to an HPWORD intrinsic is completed, the intrinsic returns a status (see Chapter 4 *Error Messages*). WERRORINFO takes this status and retrieves an associated ASCII message string from the Intrinsic catalog file specified in the WOPEN intrinsic.

Note that WCLOSE closes the message catalog and so WERRORINFO will be unable to access it if called after a WCLOSE or a failed WOPEN.

The text string returned in message will be up to 72 characters long and in the following fixed format:

Bytes 0 to 7 The first eight characters of the intrinsic name.

Bytes 8 to 10 The Primary Status. The HPWORD Intrinsics error or warning number, negative for an error and positive for a warning.

Bytes 11 to 71 The error message.

The Primary Status values and their messages are listed in Chapter 4 *Error Messages*.

## Examples

In the following examples, "_" denotes a space character.

(1) A call to WMOVEINDOC requests a forward move beyond the end of the document. The current pointer is positioned at the end of the document and a warning is returned.

The status returned is: 34 0 14

where 34 is the HPWORD Intrinsics warning number for this condition, and 14 is the intrinsic identity for WMOVEINDOC.

WERRORINFO returns the message string:

```
"WMOVE___34END OF DOCUMENT REACHED BEFORE MOVE COMPLETED"
```

(2) A call to WWRITELINE specifies a tab at the 29th position in the line of text but there is a character at the 29th position rather than the ASCII space required for a tab.

The status returned is: -25 29 37

where -25 is the HPWORD Intrinsics error number and 37 is the Intrinsic Identity for WWRITELINE.

WERRORINFO returns the message string:

```
"WWRITEL_-25THE FORMAT CHARACTER AT POS 29 WAS NOT UNDER A SPACE"
```

## Error Handling

If WERRORINFO cannot interpret the supplied intr'status the intrinsic will return an error status in status and return an error message in message that includes intr'status.

(3) WERRORINFO is passed an invalid intrinsic identity of 100.

The status that WERRORINFO returns is: -22 0 21

and the message string is:

```
WERROR_-22THE INTRINSIC ID IN THIS STATUS IS UNKNOWN  STATUS= 0 0 100
```

If WERRORINFO cannot access the message catalog, the intrinsic will return an error report in status and return intr'status in message.

(4) A call to WOPEN results in a file system error 50 because the account referenced in the user supplied file designator does not exist.

The status returned by WOPEN is: -48 50 11

where 48 is the HPWORD Intrinsics error number for "a file system error has been detected" and 11 is the intrinsic identity for WOPEN. Note that because WOPEN failed to open a document, it did not open the message catalog either, so a call to WERRORINFO would not function fully.

# WGETDOCINFO

## Intrinsic Identity 22

Provides access to HPWORD document information.

```
               LA     IA       BA           I
WGETDOCINFO  (DFCB, status, doc'name, doc'language,
                    L              !            BA
              doc'paginated, doc'page'cnt, doc'subject
                  BA          BA           BA
              doc'author, doc'creator, doc'revisor,
                    BA               BA
              doc'create'date, doc'revise'date,
                     BA
              doc'printed'date);
```

## Parameters

DFCB

*Logical array (Required)*
Document File Control Block.

status

*Integer array (Required)*
3 word intrinsic status and error information.

doc'name

*Byte array (Optional)*
Returns the MPE filename (less any lockword) of the HPWORD document file. (Max array size of 28 bytes)

doc'language

*Integer (Optional)*
Returns the document language. Values as defined in WOPEN.

doc'paginated

*Logical (Optional)*
Returns whether the document is correctly paginated; that is, whether the document has been paginated since the last modification.

0 = not paginated
1 = paginated

# WGETDOCINFO continued

`doc'page'cnt`

*Integer (Optional)*
Returns the number of pages in the document when last paginated.
Maximum value: < 1000

`doc'subject`

*Byte array (Optional)*
Returns a 60 byte ASCII string that describes the document. Supplied by the document's creator.

`doc'author`

*Byte array (Optional)*
Returns a 30 byte ASCII string given at document creation time as the name of the author of the document.

`doc'creator`

*Byte array (Optional)*
Returns a 30 byte ASCII string specifying the creator of the document.

`doc'revisor`

*Byte array (Optional)*
Returns a 30 byte ASCII string that indicates the operator who last revised the document.

`doc'create'date`

*Byte array (Optional)*
Returns the date and time that the document was created. A 16 byte ASCII string in the format :

`MM/DD/YYHH:MN TT` where:

```
MM = month   [01 to 12]
DD = day     [01 to 31]
YY = year    [00 to 99]
HH = hour    [00 to 12]
MN = minutes [00 to 59]
TT = time    [AM , PM]
```

`doc'revise'date`

*Byte array (Optional)*
Returns the date and time that the document was last revised. If a new document is being created, then it is the same as the `doc'create'date`. The format is the same as that for `doc'create'date`.

| | |
|---|---|
| doc'printed'date | *Byte array (Optional)*<br>Returns the date and time that an existing document was last printed. The format is the same as that of doc'create'date. |

## Text Discussion

This Intrinsic returns general information describing an HPWORD document. It may be called at any time after the document is opened and before it is closed.

Note that WGETDOCINFO returns whether the document is correctly paginated, that is, has not been modified since last paginated.

# WGETPRINTINFO

## Intrinsic Identity 23

Provides the capability to retrieve the current print specification associated with an HPWORD format document.

```
    OV√           LA     IA           I               L
WGETPRINTINFO  (DFCB, status, print'num'copies, print'final,
                       I                 I
                print'first'page, print'last'page,
                     L              L
                print'single, print'envelope,
                        BA                BA
                print'vardocname, print'environment,
                        I                  BA
                print'form'option, print'forms'msg);
```

## Parameters

DFCB

*Logical array (Required)*
Document File Control Block.

status

*Integer array (Required)*
3 word intrinsic status and error information.

print'num'copies

*Integer (Optional)*
Number of document print copies Maximum value of 9999.

print'final

*Logical (Optional)*
Specifies Draft or final print copy
0 = draft copy
1 = final copy

print'first'page
print'last'page

*Integer (Optional)*
The numbers of the first and last document pages which are to be printed. Maximum value of 9999 and 9999.

print'single

*Logical (Optional)*
Specifies single-sided or back-to-back page printing of the document
0 = back-to-back
1 = single-sided

print'envelope

*Logical (Optional)*
If the first paragraph of the document is specified as an envelope, then:
0 = do not print it
1 = print with envelope

print'vardocname

*Byte array (Optional)*
The MPE file designator of the file to be used for the substitution of data values for user defined variables in the document. This is meaningful only if there are user defined variables in the document. A 40 byte ASCII string in the format of doc'name in WOPEN.

print'environment

*Byte array (Optional)*
The MPE file designator of an HP2601, HP2602, HP2603 HP2687, HP2688, HP2680 or HP2934 printer environment file to be used at print time. A 40 byte ASCII string in the format of doc'name in WOPEN.

print'form'option

*Integer by value (Optional)*
The print forms selected:
0 = Plain bond for all pages
1 = Letterhead for first page
2 = Letterhead for all pages
3 = Plain bond for first page
4 = Plain bond and letterhead alternately

print'forms'msg

*Byte array (Optional)*
Text forms message. HPWORD sends this message at print time to the attender of an HP2601, HP2602, HP2603, HP2686, HP2687, or HP2934 printer or the HP3000 system operator (for system printers) A 40 byte ASCII string.

## Text Discussion

WGETPRINTINFO may be called at any time to return the print specification currently stored in the document. HPWORD interprets print information stored in the document as a record of how the document was last printed and uses it to produce the print menu when a print request is made. The intrinsics treat this information slightly differently as a print specification associated with the document describing how it is to be printed.

# WSETPRINTINFO

## Intrinsic Identity 24

Provides the capability to setup a new print specification to be associated with an HPWORD format document.

```
    OV           LA    IA            IV             LV
WSETPRINTINFO  (DFCB, status, print'num'copies, print'final,
                        IV                IV
                print'first'page, print'last'page,
                      LV           LV
                print'single, print'envelope,
                        BA                BA
                print'vardocname, print'environment,
                       IV                  BA
                print'form'option, print'forms'msg);
```

## Parameters

| | |
|---|---|
| `DFCB` | *Logical array (Required)*<br>Document File Control Block. |
| `status` | *Integer array (Required)*<br>3 word intrinsic status and error information. |
| `print'num'copies` | *Integer by value (Optional)*<br>Number of document print copies. Maximum value of 9999.<br>*default*: 1 copy. |
| `print'final` | *Logical by value (Optional)*<br>Specifies Draft or final print copy<br>0 = draft copy<br>1 = final copy<br>*default*: final copy. |
| `print'first'page`<br>`print'last'page` | *Integer by value (Optional)*<br>The page numbers of the first and last pages within the document to be printed. Maximum value of 9999 and 9999.<br>*default*: 1 and 1. |

# WSETPRINTINFO continued

`print'single`

*Logical by value (Optional)*
Specifies single-sided or back-to-back page printing of the document
  0 = back-to-back
  1 = single-sided
*default*: single-sided.

`print'envelope`

*Logical by value (Optional)*
If the first paragraph of the document is specified as an envelope, then:
  0 = do not print it
  1 = print with envelope
*default*: do not print it.

`print'vardocname`

*Byte array (Optional)*
Specifies the MPE file designator of the file to be used for the substitution of data values for user defined variables in the document. This is only meaningful if user defined variables exist in the document A 40 byte ASCII string in the format of `doc'name` in WOPEN.
*default*: ASCII spaces.

`print'environment`

*Byte array (Optional)*
The MPE file designator of an HP2601, HP2602, HP2603, HP2687, HP2688, HP2680 or HP2934 printer environment file to be used at print time. A 40 byte ASCII string in the format of `doc'name` in WOPEN.
*default*: ASCII spaces.

`print'form'option`

*Integer by value (Optional)*
The print forms selected:
  0 = Plain bond for all pages
  1 = Lettered for first pages
  2 = Lettered for all pages
  3 = Plain for first page
  4 = Plain and lettered alternately
*default*: 0.

`print'forms'msg`

*Byte array (Optional)*
Text forms message. HPWORD sends this message at print time to the attender of an HP2601, HP2602, HP2603, HP2686, HP2687, HP2934 printer or the HP3000 system operator (for system printers) A 40 byte ASCII string.
*default*: ASCII spaces.

# WSETPRINTINFO continued

## Text Discussion

This intrinsic may be called only if the document has previously been opened for write access. It establishes a new print specification for the document. This intrinsic does not have a functional equivalent in HPWORD unless a document is actually printed via the print menu screen. WSETPRINTINFO may be called at any point between the WOPEN and WCLOSE calls. A document created by WOPEN will have the default values specified above.

Note that by using this intrinsic it is possible to set up or change an existing print specification although no printing may have taken place.

Note also that you have two ways to print an HPWORD document:

- Print the document from an application program. In this case the document is read, and the text to be printed is prepared, by the application program. The program then prints the document directly on the target printer.
- Print the document interactively with HPWORD, by running HPWORD.PUB.SYS.

# WGETDOCFORMAT

## Intrinsic Identity 25

Returns the current document format parameters of an HPWORD document file.

```
            OV       LA      IA            R                  R
WGETDOCFORMAT (DFCB, status, doc'page'width, doc'page'height,
                     I             I                 I
               doc'pitch, doc'line'density, doc'para'spacing,
                      I               I              I
               doc'para'indent, doc'head'size, doc'foot'size,
                       R                R
               doc'inside'margin, doc'outside'margin,
                       I                  I
               doc'line'spacing, doc'starting'page,
                      L              L           L
               doc'auto'page, doc'hyphen, position'opt,
                  LA
               doc'tabs) ;
```

## Parameters

DFCB

*Logical array (Required)*
Document File Control Block.

status

*Integer array (Required)*
3 word intrinsic status and error information.

doc'page'width
doc'page'height

*Real (Optional)*
The width and height of the document page. These may be returned in inches or centimeters, within the range:

3 to 32 inches OR
7.62 to 81.3 centimeters

dependent on the units chosen. See *Text Discussion.*

doc'pitch

*Integer (Optional)*
The number of characters that will be printed horizontally per inch. For proportional character sets a nominal character width is taken.

0 = prop
10 = 10 pitch
12 = 12 pitch

# WGETDOCFORMAT continued

doc'line'density

*Integer (Optional)*
The number of lines to be printed per vertical inch.
6 or 8

doc'para'spacing

*Integer (Optional)*
The number of empty lines that precedes each paragraph, in the range:
0 to 19 lines

doc'para'indent

*Integer (Optional)*
The number of spaces by which the first line of each paragraph is indented.

doc'head'size
doc'foot'size

*Integer (Optional)*
Heading size is the number of lines from the top of the page to the first line of text of the first paragraph on that page. Footing size is the number of lines from the last possible text line on the page, to the bottom of the page. Default Heading and Footing Size is always 5 lines.

doc'inside'margin
doc'outside'margin

*Real (Optional)*
The inside and outside page margins in inches or centimeters will be returned in the same convention as the document page width and height values.

doc'line'spacing

*Integer (Optional)*
Vertical line spacing in multiples of half lines (see also doc'line'density, above).

doc'starting'page

*Integer (Optional)*
The page number of the first page in the document in the range 1 to 8500.

doc'auto'page

*Logical (Optional)*
Returns whether automatic page numbering has been specified for the document.
0 = off
1 = on

doc'hyphen

*Logical (Optional)*
Returns whether automatic hyphenation has been specified for the document.
0 = off
1 = on

# WGETDOCFORMAT continued

position'opt

*Logical (Optional)*
The paragraph position format selected; flush left or justified.
0 = flush left
1 = justified

doc'tabs

*Logical array (Optional)*
Definitions of any tabs which have been set up to apply to the whole document. Each tab setting is stored as a 2 byte entry in the array where:
Byte 1 : tab type (left,right,center,decimal)
Byte 2 : Horizontal tab column position
The first word contains the number of entries (tab settings) in the array.

tab type :- 1 = center
2 = right
3 = left
4 = decimal

tab column :- 1 to 155

## Text Discussion

This intrinsic returns the overall format specifications for an HPWORD document. There are no restrictions on when it may be called.

You can opt for the following parameters to be returned in either imperial or metric units:

```
doc'page'width

doc'page'height

doc'inside'margin

doc'outside'margin
```

The units are selected by passing a positive or a negative value in each parameter.

Initialize to: parameter value > 0 for inches

parameter value < 0 for centimeters

The format data returned applies globally to the whole document but may be overridden by formats specified for individual paragraphs.

# WSETDOCFORMAT

## Intrinsic Identity 26

Provides the capability to alter the document format parameters for an HPWORD document file.

```
    O✓          LA     IA          R               R
WSETDOCFORMAT (DFCB, status, doc'page'width, doc'page'height,
                 IV          IV              IV
               doc'pitch, doc'line'density, doc'para'spacing,
                    IV            IV            IV
               doc'para'indent, doc'head'size, doc'foot'size,
                      R                R
               doc'inside'margin, doc'outside'margin,
                     IV                IV
               doc'line'spacing, doc'starting'page,
                   LV            LV          LV
               doc'auto'page, doc'hyphen, position'opt,
                  LA
               doc'tabs);
```

## Parameters

DFCB

*Logical array (Required)*
Document File Control Block.

status

*Integer array (Required)*
3 word intrinsic status and error information

doc'page'width
doc'page'height

*Real (Optional)*
The width and height of the document page. These may be specified in inches or centimeters, within the range:

3 to 32 inches OR
7.62 to 81.3 centimeters

The values must be supplied as positive numbers for inches and negative numbers for centimeters. Note the value supplied will be truncated to the first three significant figures. This is also the case with the margin values.

Also, the combination of page width and margin widths must not cause the resulting line length to exceed 155 character spaces or an error will be caused.
*defaults*: 8.5 & 11 inches respectively.
(21.5 and 27.9 centimeters )

# WSETDOCFORMAT continued

`doc'pitch`

*Integer by value (Optional)*
The number of characters to be printed horizontally per inch. For proportional character sets a nominal character width is taken.
0 = prop
10 = 10 pitch
12 = 12 pitch
*default*: 10 pitch.

`doc'line'density`

*Integer by value (Optional)*
The number of lines to be printed per vertical inch.
6 or 8
*default*: 6.

`doc'para'spacing`

*Integer by value (Optional)*
The number of empty lines to precede each paragraph, in the range:
0 to 19 lines
*default*: 1 line.

`doc'para'indent`

*Integer by value (Optional)*
The number of spaces by which the first line of each paragraph is to be indented. The value must not be so large as to leave less than 10 character spaces on the first line of the paragraph or an error will occur.
*default*: 0.

`doc'head'size`
`doc'foot'size`

*Integer by value (Optional)*
Heading Size is defined as the number of lines between the top of the page and the first text line of the page. Similarly the Footing Size is the number of lines between the last text line of the page and the bottom of the page.
Range: 0 to 25 lines
*default*: 5 (Applies to default header and footer only - See Page 2-2).

# WSETDOCFORMAT continued

doc'inside'margin<br>
doc'outside'margin

*Real (Optional)*
The inside and outside page margins in inches or centimeters specified in the same convention as the document page width and height values. The two margins must leave a line length of at least 1 inch or 2.54 centimeters.
*default*: 1 inch / 2.54 centimeters.

doc'line'spacing

*Integer by value (Optional)*
Vertical line spacing in multiples of half lines (see also doc'line'density, above).
1 to 7 half lines
*default*: 2 half lines.

doc'starting'page

*Integer by value (Optional)*
The page number of the first page in the document in the range
1 to 8500
*default*: 1.

doc'auto'page

*Logical by value (Optional)*
Determines whether automatic page numbering is specified for the document.
0 = off
1 = on
*default*: 1.

doc'hyphen

*Logical by value (Optional)*
Determines whether automatic hyphenation is specified for the document.
0 = off
1 = on
*default*: 1.

position'opt

*Logical by value (Optional)*
The paragraph position format selected; flush left or justified.
0 = flush left
1 = justified
*default*: 0.

# WSETDOCFORMAT continued

doc'tabs

*Logical array (Optional)*
Definitions of any tabs which are to apply to the whole document. Each tab setting is stored as a 2 byte entry in the array where:
Byte 1 : tab type (left,right,center,decimal)
Byte 2 : Horizontal tab column position
The first word contains the number of entries (tab settings) in the array.
tab type :- 1 = center
2 = right
3 = left
4 = decimal

tab column :- 1 to 155
*default*: none

## Text Discussion

This intrinsic sets up the overall format specifications for a document. It may be called at any time without affecting the other intrinsic operations.

Note that although it is possible to specify auto-hyphenation, the intrinsics do not perform hyphenation on a line-by-line basis as lines of text are added to a document. Hyphenation (and text adjustment) is performed at document close time by WCLOSE.

In order to determine the number of characters that may fit in a given length of line the following values should be used for the character widths:

10 pitch = 0.102 of an inch OR 0.26 centimeters

12 pitch = 0.085 of an inch OR 0.216 centimeters

prop sp. = 0.085 of an inch OR 0.216 centimeters

The default values the parameters take are those that are already present in the document format. After creating a new document these values are as specified above. Thus if a parameter is not specified then no action is taken on that value.

The values specified in WSETDOCFORMAT, or existing there by default, are imposed on any newly created paragraph unless other values are specified in WSETPARAFORMAT. The values are not imposed on existing paragraphs until WCLOSE, then the whole document is readjusted.

# WDELETEPARA

## Intrinsic Identity 31

Provides the capability to delete a paragraph within a document.

```
                 LA     IA
WDELETEPARA    (DFCB, status);
```

## Parameters

DFCB

*Logical array (Required)*
Document File Control Block.

status

*Integer array (Required)*
3 word intrinsic status and error information.

## Text Discussion

WDELETEPARA deletes the whole paragraph object (paragraph format and contents) at the Current Document Position. When the paragraph has been deleted, the Current Document Position is reset to the following paragraph. Repeated calls to WDELETEPARA delete paragraphs sequentially. Note that WDELETEPARA will also delete a paragraph that has been set up by WSETPARAFORMAT but to which no contents have yet been written by WWRITELINE. If the end of the document is reached, further calls to WDELETEPARA have the effect of deleting the last paragraph and moving the Current Document Position to the new last paragraph; that is, the Current Document Position moves backward through the document.

If this intrinsic is used to delete a paragraph that has been set up but has had no text added to it, the current paragraph position is advanced to the next paragraph. If the deleted paragraph would have been the last one in the document then the Current Document Position remains at the existing last paragraph.

The default header and footer paragraphs cannot be deleted with WDELETEPARA. Any attempt to do so fails, and the current document position is unchanged.

# WGETALLPARAF

## Intrinsic Identity 40

Returns the paragraph formatting and descriptor parameters for the next paragraph in an HPWORD document file. It also establishes the read access path within the paragraph.

```
    OV
                  LA      IA         I       L      LA
WGETALLPARAF  (DFCB, status, para'type, EOD, para'tabs,
                      I                 I
               para'left'margin, para'right'margin
                     I                I
               para'top'space, para'line'spacing,
                     I             I              I
               para'pos'opt, para'hyph'opt, para'indent,
                   I            I         I       I
               para'flags, para'lineno, spare1, spare2);
```

## Parameters

DFCB

*Logical array (Required)*
Document File Control Block.

status

*Integer array (Required)*
3 word intrinsic status and error information

para'type

*Integer (Required)*
The paragraph type of the next paragraph
0 = text paragraph
1 = envelope paragraph
2 = figure space
3 = footer
4 = header

EOD

*Logical (Required)*
End of document indicator returned if this is the last paragraph in the document.
0 = not the end
1 = current paragraph is the last

para'tabs

*Logical array (Optional)*
Definitions of the tabs which are to apply to this paragraph.
Format as WSETDOCFORMAT doc'tabs.

# WGETALLPARAF continued

`para'left'margin`
`para'right'margin`

*Integer (Optional)*
Left and right horizontal margin size for this paragraph measured as the number of columns from the inside and outside document margins, respectively. The range of values is such that the resulting line length must not be less than 10 characters wide.

`para'top'space`

*Integer (Optional)*
The number of empty spacing lines that precede this paragraph, if the `para'flags` parameter indicates that the paragraph is to be positioned by top space.
Range: 0 to 19

`para'line'spacing`

*Integer (Optional)*
Vertical line spacing for lines in this paragraph in multiples of half lines.
Range: 0 to 7 half lines

`para'pos'opt`

*Integer (Optional)*
The paragraph text format selected for this paragraph; flush left, flush right, centered, justified.
1 = center
2 = flush right
3 = flush left
4 = justified

`para'hyph'opt`

*Integer (Optional)*
Determines whether automatic hyphenation has been specified for this paragraph.
0 = hyphenation off
1 = hyphenation on

`para'indent`

*Integer (Optional)*
Determines whether the paragraph first line indent is zero or as currently specified in the document format.
0 = no first line indent
N = number of spaces in the indent

`para'lineno` *Integer (Optional)*

The line number that the paragraph would be positioned on if the `para'flags` parameter specified that it was Fixed Position. The value is in half lines. A value of 1 indicates the first half-line of the physical page, not the first half-line below the heading area. If `para'flags` indicates that the paragraph is not Fixed Position, then `para'lineno` may be 0.

Range: 1 to 199 half-lines

`para'flags` *Integer (Optional)*

Bit 0 1 = Fixed Position paragraph
0 = paragraph positioned by Top Space

Bits 1 to 14 Unused

Bit 15 1 = Note (non-printing) paragraph
0 = printing paragraph

A paragraph that is a Heading, Footing, Envelope or Figure will not be a Note.

`spare1`
`spare2` *Integer (Optional)*

These parameters are not used in release A.01.00 of the intrinsics. The value returned is 0.

# WGETALLPARAF continued

## Text Discussion

If the current document position is set to the beginning of the document by WOPEN or has just been set to any other position by WMOVEINDOC or WDELETEPARA, WGETALLPARAF will return the local paragraph format specification of the paragraph at the current position. Otherwise it will return the paragraph format of the next sequential paragraph and reset the current document position to that paragraph. Consequently it is possible to position to a paragraph using WMOVEINDOC and read sequentially from that point with subsequent calls to WGETALLPARAF or WGETPARAFORMAT advancing the current document position paragraph by paragraph.

The paragraph format returned is local to the paragraph and overrides the global document format specification. It must however be specified as a subset of the global format.

WGETALLPARAF returns the paragraph type of the paragraph thus allowing the application program to decide how to process it. For instance, WREADLINE would be called to read a text paragraph line by line, while WREADFIGSPACE would be called to return the parameters of a figure space paragraph.

Note that it would not be legal to call WWRITELINE immediately after WGETALLPARAF as a new paragraph must first be established with WSETPARAFORMAT.

From Version A.05.00 of HPWORD onwards, a paragraph may either be positioned by its top space, relative to the paragraph before, or it may be positioned at a fixed line number. A paragraph has three pieces of information associated with it: the last top space value that the user selected, the last line number that the user selected, and which of these two is currently active. WGETALLPARAF returns all three values in parameters para'top'space , para'lineno and para'flags.

If the WGETALLPARAF intrinsic is called for a document that precedes HPWORD A.05.00, the paragraph will always be positioned by the top space, and so para'lineno will be zero.

# WSETALLPARAF

## Intrinsic Identity 41

Provides the capability to set up paragraph formatting and descriptor parameters for the next sequential paragraph in an HPWORD document file. It also establishes the write access path to the paragraph for subsequent write operations.

```
    ø✓            LA      IA       IV           LV
WSETALLPARAF  (DFCB, status, para'type, default'format,
                   LA           IV                IV
               para'tabs, para'left'margin, para'right'margin
                      IV             IV
               para'top'space, para'line'spacing,
                      IV           IV            LV
               para'pos'opt, para'hyph'opt, para'indent,
                   IV           IV          I         I
               para'flags, para'lineno, spare1, spare2);
```

## Parameters

`DFCB` — *Logical array (Required)*
Document File Control Block.

`status` — *Integer array (Required)*
3 word intrinsic status and error information.

`para'type` — *Integer by value (Required)*
The paragraph type of the next paragraph
- 0 = text paragraph
- 1 = envelope paragraph
- 2 = figure space
- 3 = footer
- 4 = header

`default'format` — *Logical by value (Required)*
Specifies parameter default rules. Either taken from the last call to WSETALLPARAF or from global Document format
- 0 = document format
- 1 = previous paragraph

# WSETALLPARAF continued

`para'tabs`

*Logical array (Optional)*
Definitions of the tabs which are to apply to this paragraph.
(see *WSETDOCFORMAT* for construct).
*default*: Document format OR the last paragraph accessed, dependent on the value of `default'format`.

`para'left'margin`
`para'right'margin`

*Integer by value (Optional)*
Left and right horizontal margin size for this paragraph measured as the number of columns from the inside and outside document margins respectively. The range of values is such that the resulting line length must not be less than 10 characters.
*default*: 0.

`para'top'space`

*Integer by value (Optional)*
The number of empty spacing lines to precede this paragraph when it is positioned by top space.
Range: 0 to 19
*default*: `doc'format` value.

`para'line'spacing`

*Integer by value (Optional)*
Vertical line spacing for lines in this paragraph in multiples of half lines
Range: 0 to 7 half lines
*default*: `doc'format` value OR value in previously accessed paragraph dependent on the value of `default'format`.

`para'pos'opt`

*Integer by value (Optional)*
The paragraph text format selected for this paragraph: flush left, flush right, centered, justified
1 = center
2 = flush right
3 = flush left
4 = justified
*default*: `doc'format`.

`para'hyph'opt`

*Integer by value (Optional)*
Determines whether automatic hyphenation has been specified for this paragraph.
0 = hyphenation off
1 = hyphenation on
*default:* `doc'format` OR previously accessed paragraph dependent on the value of `default'format`.

`para'indent`

*Logical by value (Optional)*
Specifies whether the paragraph first line indent should be zero or as currently specified by the document format.
0 = no first line indent
1 = as document format
*default:* as document format.

`para'lineno`

*Integer (Optional)*

The line number that the paragraph is to be positioned on if it is a Fixed Position paragraph. The value is in half-lines. A value of 1 indicates the first half-line on the physical page and not the first half-line below the Heading Area.
Range: 1 to 199 half-lines
*default:* 0 half lines. This is a legal value provided `para'flags` specifies that the paragraph is positioned by top space.

`para'flags`

*Integer by value (Optional)*

Bit 0 1 = Fixed Position paragraph
0 = paragraph positioned by top space

Bit 15 1 = Note (non-printing) paragraph
0 = printing paragraph

A paragraph that is a Heading, Footing, Envelope or Figure may not be a Note.
*default:* a printing paragraph, positioned by top space.

`spare1`
`spare2`

*Integer (Optional)*
These parameters take no effect in Version A.01.00 of the intrinsics.

## Text Discussion

WSETALLPARAF sets up the local paragraph format specification for a new paragraph. The document must be opened with write access to allow this call. If the current position within the document is set to the last paragraph then WSETALLPARAF will add a new paragraph to the end of the document. If the current position is set to some other paragraph within the document then WSETALLPARAF will insert a new paragraph after the one at the current position. In both cases the paragraph is set up for subsequent write access operations. The paragraph format created for the new paragraph overrides the global document format. Format parameters which are not specified in a call assume the format values from either the global document format or the last call to WSETALLPARAF or WSETPARAFORMAT.

Since you cannot insert a paragraph before the default footer, a call to WSETALLPARAF from a Current Document Position at the default header will result in an error report.

Note that it is not possible to create a paragraph containing nothing. Once WSETALLPARAF has been called there are only two options.

- To place something into the paragraph using WWRITELINE or WWRITEFIGSPACE
- To delete the paragraph using WDELETEPARA. If WCLOSE is called, the new paragraph will effectively be deleted.

# WGETPARAFORMAT

## Intrinsic Identity 32

Returns the paragraph formatting and descriptor parameters for the next paragraph in an HPWORD document file. It also establishes the read access path within the paragraph.

The number of descriptor values in a paragraph increased in HPWORD A.05.00. This intrinsic can be used with documents created with HPWORD A.05.00 or later, but it does not have enough parameters to return these extra values. If they are of interest, then the intrinsic WGETALLPARAF should be used. Either WGETALLPARAF or WGETPARAFORMAT can be used on documents created before A.05.00.

```
   o√            LA     IA          I      L       LA
WGETPARAFORMAT  (DFCB, status, para'type, EOD, para'tabs,
                        I                  I
                 para'left'margin, para'right'margin
                        I                I
                 para'top'space, para'line'spacing,
                        I             I             I
                 para'pos'opt, para'hyph'opt, para'indent);
```

## Parameters

DFCB

*Logical array (Required)*
Document File Control Block.

status

*Integer array (Required)*
3 word intrinsic status and error information

para'type

*Integer (Required)*
The paragraph type of the next paragraph
0 = text paragraph
1 = envelope paragraph
2 = figure space
3 = footer
4 = header

EOD

*Logical (Required)*
End of document indicator returned if this is the last paragraph in the document.
0 = not the end
1 = current paragraph is the last

`para'tabs`

*Logical array (Optional)*
Definitions of the tabs which are to apply to this paragraph. Format as WSETDOCFORMAT `doc'tabs`.

`para'left'margin`
`para'right'margin`

*Integer (Optional)*
Left and right horizontal margin size for this paragraph measured as the number of columns from the inside and outside document margins, respectively. The range of values is such that the resulting line length must not be less than 10 characters wide.

`para'top'space`

*Integer (Optional)*
The number of empty spacing lines that precede this paragraph. If the document is of HPWORD A.05.00 or later, and the paragraph is Fixed Position, then this intrinsic ignores that characteristic. It returns a top space value which bears no relation to the line number.

Range: 0 to 19

`para'line'spacing`

*Integer (Optional)*
Vertical line spacing for lines in this paragraph in multiples of half lines.

Range: 0 to 7 half lines

`para'pos'opt`

*Integer (Optional)*
The paragraph text format selected for this paragraph; flush left, flush right, centered, justified.

1 = center
2 = flush right
3 = flush left
4 = justified

`para'hyph'opt`

*Integer (Optional)*
Determines whether automatic hyphenation has been specified for this paragraph.

0 = hyphenation off
1 = hyphenation on

`para'indent`

*Integer (Optional)*
Determines whether the paragraph first line indent is zero or as currently specified in the document format.

0 = no first line indent
N = number of spaces in the indent

# WGETPARAFORMAT continued

## Text Discussion

If the current document position is set to the beginning of the document by WOPEN or has just been set to any other position by WMOVEINDOC or WDELETEPARA, WGETPARAFORMAT will return the local paragraph format specification of the paragraph at the current position. Otherwise it will return the paragraph format of the next sequential paragraph and reset the current document position to that paragraph. Consequently it is possible to position to a paragraph using WMOVEINDOC and read sequentially from that point with subsequent calls to WGETPARAFORMAT advancing the current document position paragraph by paragraph.

The paragraph format returned is local to the paragraph and overrides the global document format specification. It must however be specified as a subset of the global format.

WGETPARAFORMAT returns the paragraph type of the paragraph thus allowing the application program to decide how to process it. For instance, WREADLINE would be called to read a text paragraph line by line, while WREADFIGSPACE would be called to return the parameters of a figure space paragraph.

Note that it would not be legal to call WWRITELINE immediately after WGETPARAFORMAT as a new paragraph must first be established with WSETPARAFORMAT.

From Version A.05.00 of HPWORD onwards, a paragraph may be positioned by line number or top space. This intrinsic assumes the paragraph is positioned by top space (see WGETALLPARAF).

# WSETPARAFORMAT

## Intrinsic Identity 33

Provides the capability to set up paragraph formatting and descriptor parameters for the next sequential paragraph in an HPWORD document file. It also establishes the write access path to the paragraph for subsequent write operations.

The number of descriptor values in a paragraph increased in HPWORD A.05.00. This intrinsic can be used to create documents on a site with HPWORD installed as a separate product if it is version A.05.00 or later, and the extra descriptors will take default values of a printing paragraph positioned by top space. To control the values of these extra descriptors, use the WSETALLPARAF intrinsic. (Having version A.01.02 of the intrinsics installed without a separate version of HPWORD is equivalent to having HPWORD version A.05.04.)

```
     OV           LA     IA        IV           LV
WSETPARAFORMAT  (DFCB, status, para'type, default'format,
                     LA           IV                IV
                 para'tabs, para'left'margin, para'right'margin
                       IV              IV
                 para'top'space, para'line'spacing,
                       IV            IV            LV
                 para'pos'opt, para'hyph'opt, para'indent);
```

## Parameters

DFCB

*Logical array (Required)*
Document File Control Block.

status

*Integer array (Required)*
3 word intrinsic status and error information.

para'type

*Integer by value (Required)*
The paragraph type of the next paragraph
0 = text paragraph
1 = envelope paragraph
2 = figure space
3 = footer
4 = header

default'format

*Logical by value (Required)*
Specifies parameter default rules. Either taken from the last call to WSETPARAFORMAT or from global Document format
0 = document format
1 = previous paragraph

# WSETPARAFORMAT continued

para'tabs

*Logical array (Optional)*
Definitions of the tabs which are to apply to this paragraph.
(see WSETDOCFORMAT for construct).
*default*: Document format OR the last paragraph accessed, dependent on the value of default'format.

para'left'margin
para'right'margin

*Integer by value (Optional)*
Left and right horizontal margin size for this paragraph measured as the number of columns from the inside and outside document margins respectively. The range of values is such that the resulting line length must not be less than 10 characters.
*default*: 0.

para'top'space

*Integer by value (Optional)*
The paragraph is to be positioned by top space and this is the value to be used.
Range: 0 to 19
*default*: doc'format value.

para'line'spacing

*Integer by value (Optional)*
Vertical line spacing for lines in this paragraph in multiples of half lines
Range: 0 to 7 half lines
*default*: doc'format value OR value in previously accessed paragraph dependent on the value of default'format.

para'pos'opt

*Integer by value (Optional)*
The paragraph text format selected for this paragraph; flush left, flush right, centered, justified
1 = center
2 = flush right
3 = flush left
4 = justified
*default*: doc'format.

| | |
|---|---|
| para'hyph'opt | *Integer by value (Optional)*<br>Determines whether automatic hyphenation has been specified for this paragraph.<br>0 = hyphenation off<br>1 = hyphenation on<br>*default:* doc'format OR previously accessed paragraph dependent on the value of default'format. |
| para'indent | *Logical by value (Optional)*<br>Specifies whether the paragraph first line indent should be zero or as currently specified by the document format.<br>0 = no first line indent<br>1 = as document format<br>*default:* as document format. |

## Text Discussion

WSETPARAFORMAT sets up the local paragraph format specification for a new paragraph. The document must be opened with write access to allow this call. If the current position within the document is set to the last paragraph then WSETPARAFORMAT will add a new paragraph to the end of the document. If the current position is set to some other paragraph within the document then WSETPARAFORMAT will insert a new paragraph after the one at the current position. In both cases the paragraph is set up for subsequent write access operations. The paragraph format created for the new paragraph overrides the global document format. Format parameters which are not specified in a call assume the format values from either the global document format or the last call to WSETPARAFORMAT or WSETALLPARAF.

Since you cannot insert a paragraph before the default footer, a call to WSETPARAFORMAT from a Current Document Position at the default header will result in an error report.

Note that it is not possible to create a paragraph containing nothing. Once WSETPARAFORMAT has been called there are only two options.

- To place something into the paragraph using WWRITELINE or WWRITEFIGSPACE
- To delete the paragraph using WDELETEPARA. If WCLOSE is called, the new paragraph will effectively be deleted.

# WREADFIGSPACE

## Intrinsic Identity 34

Returns Figure Space parameters from a paragraph containing a figure space in an HPWORD document.

```
                 LA      IA        I           I
WREADFIGSPACE  (DFCB, status, fig'size, fig'placement
                      L           BA            BA
                fig'overlay, figure'file, figure'name,
                      L           L           L
                fig'rotate, bind'line, page'break);
```

## Parameters

DFCB
*Logical array (Required)*
Document File Control Block.

status
*Integer array (Required)*
3 word intrinsic status and error information.

fig'size
*Integer (Required)*
The vertical size of the figure space in lines.

fig'placement
*Integer (Optional)*
The horizontal placement of the figure within the figure space area:
1 = centered
2 = right flush
3 = left flush

fig'overlay
*Integer (Optional)*
Specifies whether the figure overlays the text in the following paragraph or is stand-alone:
0 = stand alone
1 = overlays

figure'file
*Byte array (Optional)*
The MPE file designator for the HP Figure File or Raster file containing the figure to be drawn. A 35 byte ASCII string to the format specified for doc'name in WOPEN.

# WREADFIGSPACE continued

figure'name

*Byte array (Optional)*
The name of a figure. This is specified if the filename specified in figure'file is an HP Figure File rather than a Raster File. A 16 byte alphanumeric string.

fig'rotate

*Logical (Optional)*
Specifies whether or not the figure should be rotated through 90 degrees anti-clockwise for printing.

0 = do not rotate
1 = rotate

bind'line

*Logical (Optional)*
Indicates whether the figure is bound to the line of text following.

0 = not bound
1 = bound

page'break

*Logical (Optional)*
Returns whether a page break occurs after this figure:

0 = no page break
1 = user active page break
2 = user pending page break
3 = system active page break

## Text Discussion

WREADFIGSPACE should be called when a previous call to WGETPARAFORMAT has indicated that the next paragraph in the document contains a figure space. WREADFIGSPACE will then return the figure space parameters.

# WWRITEFIGSPACE

## Intrinsic Identity 35

Provides the capability to set up figure space parameters in a paragraph in an HPWORD document that has been designated a Figure space paragraph.

```
     O✓          LA      IA        IV           IV
WWRITEFIGSPACE  (DFCB, status, fig'size, fig'placement,
                       LV          BA             BA
                 fig'overlay, figure'file, figure'name,
                       LV          LV           LV
                 fig'rotate, bind'line, page'break);
```

## Parameters

`DFCB`
*Logical array (Required)*
Document File Control Block.

`status`
*Integer array (Required)*
3 word intrinsic status and error information.

`fig'size`
*Integer by value (Required)*
The vertical size of the figure space in lines
Minimum value: 1,
Maximum value: computed from:

```
doc'line'density * doc'page'height
    - doc'head'size
    - doc'foot'size
```

where the values are as specified in WSETDOCFORMAT or as returned in WGETDOCFORMAT.

`fig'placement`
*Integer by value (Optional)*
The horizontal placement of the figure within the figure space area:

1 = centered
2 = right flush
3 = left flush

*default*: centered.

# WWRITEFIGSPACE continued

`fig'overlay`

*Logical by value (Optional)*
Specifies whether the figure overlays the text of the following paragraph or stands alone:
0 = stand alone
1 = overlays
*default*: stand alone.

`figure'file`

*Byte array (Optional)*
The MPE file designator for the HP Figure File or Raster file containing the figure to be drawn.
A 35 byte ASCII string to the format specified for `doc'name` in WOPEN.
*default*: ASCII spaces.

`figure'name`

*Byte array (Optional)*
The name of a figure. This is specified if the filename specified in `figure'file` is an HP Figure File rather than a Raster File. A 16 byte alphanumeric string.
*default*: ASCII spaces.

`fig'rotate`

*Logical by value (Optional)*
Specifies whether the figure should be printed rotated 90 degrees anti-clockwise or not.
0 = do not rotate
1 = rotate
*default*: do not rotate.

`bind'line`

*Logical by value (Optional)*
Indicates whether the figure is bound to the line of text following.
0 = not bound
1 = bound
*default*: not bound.

`page'break`

*Logical by value (Optional)*
Indicates whether a page break occurs after this figure:
0 = no page break
1 = a user pagebreak occurs after this figure
*default*: no page break.

# WWRITEFIGSPACE continued

## Text Discussion

After WSETPARAFORMAT has been called to set up the next paragraph as a figure space paragraph, this intrinsic is called to specify the details of the figure. Any attempt to set up a figure space specification for a non figure space paragraph will be disallowed.

The HP Figure file must have been produced by either the DSG/3000, HPDraw or HPEasyChart packages. The intrinsic does not check for the existence of the Figure file; this information is not relevant until the document is printed from within HPWORD.

# WREADLINE

## Intrinsic Number 36

WREADLINE reads a line from an HPWORD format document file.

```
   OV        LA      IA     L        I          BA
WREADLINE   (DFCB, status, EOP, line'length, text'line,
              LA          DA           L        L
            formats, enhancements, newline, bind'line,
                    L             L
            line'hyphenated, page'break);
```

## Parameters

DFCB

*Logical array (Required)*
Document File Control Block.

status

*Integer array (Required)*
3 word intrinsic status and error information

EOP

*Logical (Required)*
End of Paragraph indicator which is returned on the last line in the paragraph.
0 = not the last line
1 = last line

line'length

*Integer (Required)*
This parameter will supply the number of ASCII characters that are in the text'line array. If an HPWORD document is being read then this value will be less than 160 characters. But if any lines of text have been written to a paragraph which were greater than this value then they will be returned in the same format and length as they were supplied until the document is closed and adjusted. At this time, and only at this time, will the lines of text be split up and adjusted to fit between the paragraph margins.

# WREADLINE continued

`text'line`

*Byte array (Required)*
Current line of text comprising printable ASCII characters and spaces. A tab is replaced by a single ASCII space, unless the Expanded Read Mode has been selected in WOPEN, when the correct number of spaces replaces the tab character. There is no formatting information included in the text string.

A required space is replaced by an ASCII space, and its presence indicated by a flag in the `formats` parameter.

The first line of a paragraph may have a paragraph indent. In Expanded Read Mode, the correct number of spaces will be inserted at the start of the line. In other modes they will not. This number is available from WGETPARAFORMAT in parameter `para'indent`.

In Expanded Read Mode, spaces will be inserted at the start of the `text'line` for the `para'left'margin`. In other modes they will not. Spaces are never returned at the end of the `text'line` to denote the `para'right'margin`.

The array should be at least 250 bytes.

`formats`

*Logical array (Optional)*
This is not valid if Expanded Read Mode is selected. Specifies where tab, indent and required space characters should occur in `text'line`. ASCII blank characters actually occupy the positions in `text'line`.
The array is organized as a parallel array, one element to each character in `text'line`. The following mutually exclusive bit settings indicate which format character should be substituted into `text'line` in place of the ASCII space character :

Bit 0 Tab character
Bit 1 Indent character
Bit 2 Required space character

Bit 0 is the left most bit in the logical word.

This array should contain at least 250 words.

# WREADLINE continued

`enhancements`

*Double array (Optional)*
Specifies which enhancements apply to `text'line`. The array is organized as a parallel array, one element (double word) to each character in `text'line`. In expanded read mode the array is expanded so that it is still parallel with the text line. The following mutually exclusive bit settings in each element indicate which enhancements apply to the corresponding character in `text'line`:

Bit 0 Bold
1 Italic
2 Underline
3 Double underline
4 Discretionary hyphen
5 Superscript
6 Subscript
7 System variable
8 User variable

Bit 0 is taken to be the left most bit in the double word. The array size should be at least 250 elements (250 doublewords) long.
*default*: none.

`newline`

*Logical (Optional)*
This text line ends with a return marker
1 = there is a return marker
0 = there is not one present

`bind'line`

*Logical (Optional)*
Indicates whether this line is bound to the next line.
1 = the line is bound to the next.
0 = the line is not bound

`line'hyphenated`

*Logical (Optional)*
This line ends with a hyphenated word (only the first part of the word up to the hyphenation point should be supplied in the array `text'line`.
1 = the last word is hyphenated
0 = the last word is complete

page'break

*Logical (Optional)*
A page break occurs at the end of this line. Note user page breaks can not occur unless this is the last line in the paragraph.
0 = no page break
1 = user active page break
2 = user pending page break
3 = system active page break

## Text Discussion

WREADLINE is called to read text from a paragraph in an HPWORD document on a line-by-line basis. It is the only means by which text may be read from a document. WREADLINE may be called after WGETPARAFORMAT has returned the paragraph format for the next paragraph and established the current document position at that paragraph. WREADLINE is used to read line-by-line through the paragraph, provided the paragraph contains text, and returns bare ASCII text and the set of formatting and enhancement attributes associated with that text. An end of paragraph indication is returned when the last line in the current paragraph is read.

Note that when WGETPARAFORMAT is called, a read access path is established; WREADLINE and WWRITELINE calls cannot be mixed within a paragraph.

It is possible for this intrinsic to return a text-line length of 0. A user editing a document might have deleted all the text in a paragraph containing a page break.

## Example of Unexpanded Read Mode

In the following example of formats and enhancements arrays the zeros have been replaced by periods to make the bit settings more visible.

Actual line as printed by HPWORD :

This is **the 1$^{st}$** Example of a   *user variable*   var1

In this line, there are three required spaces between "a" and "user", and a tab character between "variable" and "var1".

Arrays returned by the WREADLINE intrinsic (where spaces are shown as underlines):

```
TEXT'LINE :

This_is_the_1st_Example_of_a___user_variable__ var1
---------------------------------------------------
                                                   FORMATS
..............................................1....  tab char
...................................................  indent char
............................111....................  reqd space
---------------------------------------------------
                                                   ENHANCEMENTS
........1111111....................................  bold
...............................1111111111111.......  italic
................1111111............................  underline
...................................................  dbl undrline
.......................................1...........  disc hyphen
.............11....................................  superscript
...................................................  subscript
...................................................  system var
...............................................1111  user var
```

Note the discretionary hyphen under the "i" of "variable" indicates that one was placed betweeu the "r" and the "i" although it does not show in the printing of the text line.

## Example of Expanded Read Mode

In the following example of formats and enhancements arrays the zeros have been replaced by periods to make the bit settings more visible.

Actual line as printed by HPWORD :

This is **the 1$^{st}$** *Example* of a   *user variable*   var1

In this line, there are three required spaces between "a" and "user", and a tab character between "variable" and "var1".

Arrays returned by the WREADLINE intrinsic (where spaces are shown as underlines):

```
TEXT'LINE :

This_is_the_1st_Example_of_a___user_variable____________var1

                                                            FORMATS
............................................................  tab char
............................................................  indent char
............................111.............................  reqd space

                                                            ENHANCEMENTS
........1111111.............................................  bold
...............................1111111111111................  italic
................1111111.....................................  underline
............................................................  dbl undrlne
.......................................1....................  disc hyphen
.............11.............................................  superscript
............................................................  subscript
............................................................  system var
........................................................1111  user var
```

Note the discretionary hyphen under the "i" of "variable" indicates that one was placed between the "r" and the "i" although it does not show in the printing of the text line.

# WWRITELINE

## Intrinsic Identity 37

Writes a text line to an HPWORD format document file.

```
   OV         LA      IA          IV           BA
WWRITELINE (DFCB, status, line'length, text'line,
              LA           DA          LV         LV
           formats, enhancements, newline, bind'line,
                  LV           LV
           line'hyphenated, page'break);
```

## Parameters

`DFCB`

*Logical array (Required)*
Document File Control Block.

`status`

*Integer array (Required)*
3 word intrinsic status and error information.

`line'length`

*Integer by value (Required)*
The number of text characters supplied in `text'line`. You are allowed to supply up to 250 characters. These are stored in the paragraph record as a single line until HPWORD adjusts the paragraph. If necessary the line is then split into multiple lines to fit between the margins.

`text'line`

*Byte array (Required)*
Current line of text comprising printable ASCII characters and spaces. Tabs, indents and required spaces are replaced by ASCII spaces. There is no formatting information included in the text string. The array should not exceed 250 bytes.

| | |
|---|---|
| formats | *Logical array (Optional)*<br>Specifies where tab, indent and required space characters should occur in text'line. ASCII blank characters actually occupy the positions in text'line.<br>The array is organized as a parallel array, one element to each character in text'line. The following mutually exclusive bit settings indicate which format character should be substituted into text'line in place of the ASCII space character :<br>Bit 0 Tab character<br>Bit 1 Indent character<br>Bit 2 Required space character<br>Bit 0 is the left most bit in the logical word. This array should contain at most 250 words.<br>*default*: none. |
| enhancements | *Double array (Optional)*<br>Specifies which enhancements apply to text'line. The array is organized as a parallel array, one element (double word) to each character in text'line. The following bit settings in each element indicate which enhancements apply to the corresponding character in text'line:<br>Bit 0 Bold<br>1 Italic<br>2 Underline<br>3 Double underline<br>4 Discretionary hyphen<br>5 Superscript<br>6 Subscript<br>7 System variable<br>8 User variable<br>Bit 0 is taken to be the left most bit in the double word. The array size should be at most 250 elements (250 doublewords) long.<br>*default*: none. |
| newline | *Logical by value (Optional)*<br>This text line ends with a return marker<br>1 = there is a return marker<br>0 = there is not one present<br>*default*: 0. |

# WWRITELINE continued

`bind'line`

*Logical by value (Optional)*
Indicates whether this line is bound to the next line.
1 = the line is bound to the next
0 = the line is not bound
*default:* 0.

`line'hyphenated`

*Logical by value (Optional)*
This line ends with a hyphenated word (only the first part of the word up to the hyphenation point should be supplied in the array `text'line`).
1 = the last word is hyphenated
0 = the last word is complete
*default:* 0.

`page'break`

*Logical by value (Optional)*
A page break occurs after this line. Note only user page breaks can be set and these can only occur in the last line of a paragraph. Thus the paragraph will be terminated if this parameter is set.
0 = no page break
1 = a user page break is to be placed at the end of the line
*default:* 0.

## Text Discussion

WWRITELINE is used to write lines of text within a paragraph. WSETPARAFORMAT must have been previously called to set up a new paragraph format and establish the current position at the next paragraph. WWRITELINE is then used to perform line by line writes through the paragraph. To move to the next paragraph WSETPARAFORMAT must be called again after which more calls to WWRITELINE may be made if desired for further text entry. WWRITELINE and WREADLINE calls may not be mixed within the same paragraph. It is important to remember that actual access to a paragraph may only be either read or write but not both.

The length of the line that may be written to the paragraph must be less than 250 characters. It is important to remember to include a space between the last word on a line written by WWRITELINE and the first word of the next line to be written. If this is not done then the two words will be treated as one long one. When the document is WCLOSEd the entire document will be adjusted and any hyphenation taken care of.

It should be noted that the current paragraph will be closed by either supplying the page'break parameter with a value of 1 (a User pagebreak) or calling one of the following intrinsics :

WSETPARAFORMAT

WGETPARAFORMAT

WSETALLPARAF

WGETALLPARAF

WDELETEPARA

WMOVEINDOC

WCLOSE

Also, it should be noted that when page break is specified it is entered as a user pending page break. When the document is WCLOSEd and pagination is requested, it is converted to an actual page break.

# WSETPAGEBREAK

## Intrinsic Identity 38

Places a User page break at the end of the current paragraph.

```
                LA     IA
WSETPAGEBREAK  (DFCB, status);
```

## Parameters

DFCB

*Logical array (Required)*
Document File Control Block.

status

*Integer array (Required)*
3 word intrinsic status and error information.

## Text Discussion

WSETPAGEBREAK provides the calling application with some pagination control. It is used to establish a new user pagebreak after the paragraph currently positioned to.

Note that a user page break may also be specified while writing to a paragraph (via WWRITEFIGSPACE or WWRITELINE) with the following result:

When this intrinsic is called the paragraph will be terminated thus requiring one of the following intrinsics to be called to set the paragraph pointer to a new paragraph :

WSETPARAFORMAT

WGETPARAFORMAT

WSETALLPARAF

WGETALLPARAF

WMOVEINDOC

# WDELETEPAGEBREAK

## Intrinsic Identity 39

Deletes a user pagebreak

```
                     LA     IA
WDELETEPAGEBREAK   (DFCB, status);
```

## Parameters

DFCB — *Logical array (Required)*
Document File Control Block

status — *Integer array (Required)*
3 word intrinsic error and status information.

## Text Discussion

WDELETEPAGEBREAK is used to delete the page break that follows the paragraph currently positioned to. It should be noted that only a user specified page break can be deleted.

As with WSETPAGEBREAK calling this intrinsic will terminate access to the current paragraph requiring the paragraph pointer to be set to a new paragraph in the document.

# WVERSION

## Intrinsic Identity 20

Returns the version number of the HPWORD Intrinsic Set in use.

```
              IA       BA
WVERSION  (status, version);
```

## Parameters

status

*Integer array (Required)*
3 word intrinsic error and status information.

version

*Byte array (Required)*
Returns the version number of the HPWORD Intrinsic Set in use. (16 bytes)

## Text Discussion

WVERSION returns the version number of the HPWORD Intrinsics intrinsic code in use. It may be called at any time. No document need be open when WVERSION is called.

The intrinsics version number is defined as follows:

```
HP32119A.vv.ff
```

where vv = version number
 ff = fix number

with 2 trailing blanks following.

# Chapter 4

# Error Messages

# Error Messages

## Meanings and Actions

Errors are reported through the 3 word array STATUS in the following manner :

Word 1 - primary status indicating the error/warning number. The values supplied in this word correspond to the error message numbers in the message catalog sets.

| | |
|---|---|
| Zero | implies that the intrinsic call was successful. |
| Negative | values indicate that an error has occurred. |
| Positive | values are to warn the user that although the intrinsic call was completed satisfactorily some problem or unusual condition was encountered. |

Word 2 - secondary status supplying more information such as the number of the file system error or HPWORD INTRINSICS internal error that occurred.

Word 3 - the identity number of the intrinsic that has returned the status.

This status may be passed to WERRORINFO which will return a message string taken from a message catalog specified at WOPEN time (see *WERRORINFO*, description and examples).

For the purpose of identification, the important words in the status array are the primary status which gives the error number and the intrinsic ID which identifies the intrinsic called. The Intrinsic identities are as follow:

STATUS WORD THREE: INTRINSIC ID

| | |
|---|---|
| WOPEN | 11 |
| WCLOSE | 12 |
| WMARKPARAS | 13 |
| WMOVEINDOC | 14 |
| WERRORINFO | 21 |
| WGETDOCINFO | 22 |
| WGETPRINTINFO | 23 |
| WSETPRINTINFO | 24 |
| WGETDOCFORMAT | 25 |
| WSETDOCFORMAT | 26 |
| WDELETEPARA | 31 |
| WGETPARAFORMAT | 32 |
| WSETPARAFORMAT | 33 |
| WREADFIGSPACE | 34 |
| WWRITEFIGSPACE | 35 |
| WREADLINE | 36 |
| WWRITELINE | 37 |
| WSETPAGEBREAK | 38 |
| WDELETEPAGEBREAK | 39 |
| WGETALLPARAF | 40 |
| WSETALLPARAF | 41 |

The messages in the catalog are arranged into sets by the ID number of the intrinsic that returns them. The number of the message within the set is given by the modulus of the Primary Status. If the sign is positive then the message is a warning. If the primary status is negative then there is an error condition. Some warnings and errors are common to the intrinsics and are to be found in Set 1, they have a primary status in the range from 1 to 18 inclusive. A Primary Status of zero (indicating successful completion) is mapped onto message number 19.

In the following lists of error messages, warnings are marked "w".

## Key

FS - File System error. The parameter value is the errorcode returned by FCHECK.

WPERR - HPWORD error. The parameter value is an HPWORD error number, consult Appendix F *Error and Warning Messages* in the *HPWORD Administration* manual.

IT - Intrinsic error. An internal inconsistency has been detected by the intrinsics. The parameter value is secondary information.

Please report these errors to Hewlett-Packard with the following documentation on a tape:

- Source listing of the application program
- Application program file
- The MPE files being accessed when the error occurred.
- Description of the program's design and steps taken to reproduce the problem.

## SET 1 Common Messages

These messages are common to all the intrinsics and are numbered from 1 to 19.

```
  1 NO DFCB PARAMETER SUPPLIED
  2 DFCB SUPPLIED IS CORRUPTED, DOCUMENT SHOULD BE WCLOSEd
  3 FILE SYSTEM ERROR (FSxx) OCCURRED
  4 INTERNAL ERROR(ITxx) OCCURRED IN HPWORD INTRINSICS
  5 FATAL ERROR HAS OCCURRED, DOCUMENT SHOULD BE WCLOSEd
  6 INVALID POINT AT WHICH TO CALL THIS INTRINSIC (ITxx)
  7 REQUIRED PARAMETER  !xx MISSING
  8 INVALID DOCUMENT FORMAT SUPPLIED TO THE INTRINSIC
  9 A DOCUMENT MUST BE OPEN TO CALL THIS INTRINSIC
 10 PARAMETER !'s START ADDRESS IS OUT OF BOUNDS
 11 PARAMETER ! NOT LARGE ENOUGH, BOUNDS VIOLATION MAY OCCUR
 12 PARAMETER ! MAY CAUSE A BOUNDS VIOLATION
w19 INTRINSIC CALL COMPLETED WITHOUT ERROR
```

Each of the following sets contains messages local to a single intrinsic. Within each set the messages are numbered from 20 upwards.

## SET 11 WOPEN

```
  20 DOCUMENT OF THIS NAME IS ALREADY OPEN
  22 THE DFCB SUPPLIED BELONGS TO AN OPEN DOCUMENT
  23 THE DOCUMENT NAME IS A REQUIRED PARAMETER
  24 CREATION OF A DOCUMENT REQUIRES READ'WRITE ACCESS
  26 UNKNOWN VALUE (!) SPECIFIED FOR DOMAIN
  28 UNKNOWN VALUE (!) SPECIFIED FOR ACCESS TYPE
  31 ACCESS FAILED TO HPWORD's LANGUAGE CONFIGURATION FILE (FS!)
* 32 UNABLE TO OPEN THE MESSAGE CATALOG
  33 R/W ACCESS DENIED, LOC LANGUAGE (!) NOT CONFIGURED ON SYSTEM
  34 UNKNOWN VALUE (!) SPECIFIED FOR DOCUMENT LANGUAGE
  36 UNKNOWN VALUE (!) SPECIFIED FOR DOCUMENT SIZE
  38 INCORRECT VERSION OF HPWORD INSTALLED ON SYSTEM
  40 ACCESS TO THE HPWORD CONFIGURATION FILE FAILED (FS!)

  41 MANIPULATION OF HPWORD GLOBAL RIN FAILED (FS!)
  43  A FILE OF THAT NAME ALREADY EXISTS
  44 DOCUMENT IS EMPTY, INCORRECT, AND CANNOT BE RECOVERED
  46 MADE INSUFFICIENT ROOM FOR THE SIZE OF DOCUMENT
  48 UNABLE TO OPEN THE SPECIFIED DOCUMENT FILE (FS!)
  50 ACCESS TO THE FILE FAILED, SUBSEQUENT TO OPENING IT (FS!)
  52 FILE SPECIFIED IS NOT AN HPWORD DOCUMENT
  54 ACCESS DENIED, DOCUMENT IS MARKED AS QUEUED FOR PRINTING
  56 DOCUMENT IS MARKED AS BEING ACCESSED BY OTHER PROCESS

  58 DOCUMENT LANGUAGE IS INCONSISTENT WITH LANGUAGE SPECIFIED
  59 VERSION OF DOCUMENT INCOMPATIBLE WITH VERSION OF INTRINSICS
  60 DOCUMENT NEEDS RECOVERY, READ/WRITE ACCESS REQUIRED
  62 UNABLE TO PREPARE DOCUMENT FOR RECOVERY BY HPWORD (FS!)
  66 DOCUMENT IS USELESS, COMPLETELY EMPTY, SUGGEST PURGE IT.
  68 DOCUMENT RECOVERY WAS REQUIRED, ATTEMPTED BUT FAILED
  70 INCONSISTENCY IN DOCUMENT, WOPEN FAILED
  72 WOPEN FAILED TO RECLOSE DOCUMENT SUBSEQUENT TO AN ERROR
  74 UNABLE TO CREATE DOCUMENT (FS!)
  76 INITIALIZATION OF CREATED DOCUMENT FAILED (FS!)
  78 ACCESS TO THE HPWORD MESSAGE CATALOG FAILED (!FS)
  80 DOCUMENT REQUIRES RECOVERY, UNABLE TO,INVOKE HPWORD (FS !)
  82 DOCUMENT RECOVERY WAS ATTEMPTED BUT FAILED (WPERR!)
  84 DOCUMENT NEEDS RECOVERY & PH CAPABILITY NEEDED ON PROGRAM
  86 DOCUMENT REQUIRES RECOVERY, FAILED TO INVOKE HPWORD (IT!)
```

## SET 12 WCLOSE

```
w 20 UNKNOWN VALUE (!) SPECIFIED FOR PAGINATION PARAMETER
w 22 UNKNOWN VALUE (!) SPECIFIED FOR DISPOSITION PARAMETER
w 24 DOCUMENT CLOSED, THE DFCB WAS CORRUPT
  25 DOCUMENT COULD NOT BE CLOSED (FS!) & THE DFCB WAS CORRUPT
w 26 DOCUMENT CLOSED, BUT COULD NOT REPAGINATE IN READ/ONLY ACCESS
w 28 DOCUMENT CLOSED, BUT COULD NOT PURGE IN READ/ONLY ACCESS
w 29 DOCUMENT CLOSED, UNABLE TO CLOSE THE MESSAGE CATALOG (FS!)
  30 DOCUMENT WOULD NOT CLOSE (FS!)
w 31 DOCUMENT MAY REQUIRE RECOVERY, MAY BE INCONSISTENT (IT!)
w 32 DOCUMENT CLOSED BUT UNABLE TO PURGE IT (FS!)
w 42 DOCUMENT WAS NOT OPEN TO BE CLOSED (FS!)
  80 DOCUMENT UNADJUSTED, UNABLE TO INVOKE HPWORD (FS!)
  82 HPWORD LOCATED AN INCONSISTENCY IN THE DOCUMENT (WPERR!)
  84 PH CAPABILITY REQUIRED TO INVOKE HPWORD TO ADJUST DOCUMENT
  86 DOCUMENT UNADJUSTED, COULD NOT INVOKE HPWORD (IT!)
```

## SET 13 WMARKPARAS

```
  20  THERE IS NO DOCUMENT OPEN TO SET EDIT POSITIONS WITHIN
  21  THE DOCUMENT IS EMPTY APART FROM THE TWO DEFAULT PARAGRAPHS
  22  BETWEEN 1 AND ! EDIT POSITIONS MAY BE SPECIFIED
  24  DOCUMENT SIZE (! PAGES) TOO GREAT FOR THIS INTRINSIC
  26  THE DOCUMENT ONLY CONTAINS ! PAGES
  28  A NEGATIVE PARAGRAPH NUMBER WAS SPECIFIED FOR POSITION !
  30  A NEGATIVE PAGE NUMBER WAS SPECIFIED FOR POSITION !
  32  NO PARAGRAPH COULD BE FOUND AT THIS POSITION !
```

## SET 14 WMOVEINDOC

```
  22  UNKNOWN VALUE (!) SPECIFIED FOR MOVE MODE
  24  EDIT POSITION SPECIFIED OUTSIDE OF LEGAL RANGE -1 TO !
  26  EDIT POSITIONS WERE NEVER DEFINED
  28  EDIT POSITIONS WERE LOST WHEN DOCUMENT BECAME EMPTY
  30  NUMBER OF PARAGRAPHS SPECIFIED IS ! AND IS ILLEGAL
  32  DFCB SUPPLIED DOES NOT SPECIFY AN OPEN DOCUMENT
w 34  END OF DOCUMENT REACHED BEFORE MOVE COMPLETED
w 36  TOP OF DOCUMENT REACHED BEFORE MOVE COMPLETED
  38  THE DFCB IS INCONSISTENT (IT!)
```

## SET 21 WERRORINFO

The following error messages are returned by WERRORINFO. The intrinsic status passed to WERRORINFO for interpretation is included in the message.

```
22 THE INTRINSIC ID IN THIS STATUS IS UNKNOWN             STATUS=!
24 NO MESSAGE FOUND FOR THIS ERROR NUMBER                 STATUS=!
26 NO SET NUMBER IN THE MESSAGE FILE FOR THIS INTRINSIC  STATUS=!
30 WERROR UNABLE TO INTERPRET THIS STATUS                 STATUS=!
```

When WERRORINFO is unable to access the message catalog, it returns one of the following Primary Status values which are to be interpreted as:

```
34 THERE IS NOT AN OPEN MESSAGE FILE
36 A FILE SYSTEM ERROR OCCURRED.  WERRORINFO's secondary
   status gives the MPE Genmessage error number.
```

## SET 22 WGETDOCINFO

```
  22  DOCUMENT SIZE (! PAGES) TOO GREAT FOR THIS INTRINSIC
  30  THE SUPPLIED DFCB DOES NOT REFER TO AN OPEN DOCUMENT
w 32  UNABLE TO DETERMINE STATUS OF PAGINATION
```

## SET 23 WGETPRINTINFO

------- No error messages ----------

## SET 24 WSETPRINTINFO

```
20  NEGATIVE VALUE (!) SPECIFIED FOR NUMBER OF COPIES
22  UNKNOWN VALUE (!) SPECIFIED FOR OPTION FINAL OR DRAFT
24  UNKNOWN VALUE (!) SPECIFIED FOR OPTION SINGLE-SIDED
26  FIRST PAGE NUMBER (!) MUST BE IN RANGE 1 TO 9999
28  LAST PAGE NUMBER (!) MUST BE IN RANGE 1 TO 9999
30  LAST PAGE SPECIFIED (!) IS LESS THAN FIRST PAGE
32  UNKNOWN VALUE (!) SPECIFIED FOR ENVELOPE OPTION
34  UNKNOWN VALUE (!) SPECIFIED FOR FORMS OPTION
36  READ/WRITE ACCESS IS REQUIRED TO SET THE PRINT INFORMATION
```

## SET 25 WGETDOCFORMAT

```
20  THE INTRINSIC COULD NOT ACCESS THE DATA (FS ! )
21  AN INTERNAL FILE READ ERROR OCCURRED (FS ! )
22  THE DOCUMENT FORMAT RECORD COULD NOT BE READ (FS ! )
```

## SET 26 WSETDOCFORMAT

```
20 THE DOCUMENT POINTER COULD NOT BE RESET (FS ! )
21 THE DOCUMENT FORMAT RECORD COULD NOT BE ACCESSED (FS ! )
22 PAGE WIDTH MUST BE BETWEEN  7.62 & 81.3 CMS
23 PAGE HEIGHT MUST BE BETWEEN  7.62 & 81.3CMS
24 THE RESULTING LINE LENGTH IS LESS THAN 1 INCH
25 INVALID PITCH PARAMETER ( ! ) SPECIFIED
26 INVALID LINE DENSITY PARAMETER ( ! ) SPECIFIED
27 INVALID PARA SPACING PARAMETER ( ! ) SPECIFIED
28 INVALID HEAD AREA PARAMETER ( ! ) SPECIFIED
29 INVALID FOOT AREA PARAMETER ( ! ) SPECIFIED
30 INVALID LINE SPACING PARAMETER ( ! ) SPECIFIED
31 INVALID STARTING PAGE PARAMETER ( ! ) SPECIFIED
32 INVALID AUTO PAGE NUMBERING PARAMETER ( ! ) SPECIFIED
33 INVALID AUTO HYPHENATION PARAMETER ( ! ) SPECIFIED
34 INVALID DOCUMENT JUSTIFICATION PARAMETER ( ! ) SPECIFIED
35 NUMBER OF TABS ( ! ) SPECIFIED FOR THE DOCUMENT IS INVALID
36 THE TAB TYPE ( ! ) WAS NOT UNDERSTOOD
37 THE TAB COLUMN ( ! ) IS NOT A VALID POSITION
38 MULTIPLE TABS WERE SPECIFIED AT POSITION ( ! )
39 THE DOCUMENT FORMAT RECORD COULD NOT BE UPDATED (FS ! )
40 DOCUMENT POINTER WAS NOT RESET AFTER FORMAT UPDATE (FS ! )
41 THE INDENT VALUE ( ! ) IS INVALID
42 THE INDENT VALUE IS TOO LARGE FOR THE LENGTH OF LINE
43 THE DEFAULT INDENT VALUE IS TOO LARGE FOR THE LINE LENGTH
44 YOU MUST HAVE READ/WRITE ACCESS TO ALTER THE DOC FORMAT
45 PAGE WIDTH MUST BE BETWEEN 3 & 32 INCHES
46 PAGE HEIGHT MUST BE BETWEEN 3 & 32 INCHES
47 INVALID SIZE SPECIFIED ( IN CMS ) FOR THE INSIDE MARGIN
48 INVALID SIZE SPECIFIED ( IN INS ) FOR THE INSIDE MARGIN
49 INVALID SIZE SPECIFIED ( IN CMS ) FOR THE OUTSIDE MARGIN
50 INVALID SIZE SPECIFIED ( IN INS ) FOR THE OUTSIDE MARGIN
51 MORE THAN 155 CHARACTERS HAVE BEEN SPECIFIED ( ! )
52 THE DOCUMENT FORMAT RECORD COULD NOT BE ACCESSED (FS ! )
53 THE NEW DOCUMENT FORMAT RECORD COULD NOT BE ADDED(FS ! )
54 THE DEFAULT FOOTER COULD NOT BE ACCESSED (FS ! )
55 THE DEFAULT FOOTER COULD NOT BE UPDATED (FS ! )
```

## SET 31 WDELETEPARA

```
20 YOU MUST HAVE READ/WRITE ACCESS TO DELETE A PARAGRAPH
21 YOU CAN NOT DELETE THE DEFAULT HEADER
22 YOU CAN NOT DELETE THE DEFAULT FOOTER
23 INTERNAL ERROR (IT ! ) OCCURRED WHEN RESETTING THE POINTER
24 INTERNAL ERROR (IT ! ) OCCURRED WHEN ACCESSING NEXT PARA
25 INTERNAL ERROR (IT ! ) OCCURRED WHILE DELETING PARA
26 INTERNAL ACCESS ERROR (IT ! ) OCCURRED WHILE DELETING PARA
27 INTERNAL ERROR (IT ! ) OCCURRED ON FINAL POINTER RESET
28 INTERNAL ERROR (IT ! ) WHEN ACCESSING THE CURRENT PARA
29 INTERNAL ERROR (IT ! ) WHEN ACCESSING PREVIOUS PARA
```

## SET 40 WGETALLPARAF

```
  20 THERE ARE NO MORE PARAGRAPHS IN THE DOCUMENT
w 21 WARNING - THE PARAGRAPH TYPE IS INVALID.
```

## SET 41 WSETALLPARAF

```
  20 NO PARAGRAPH TYPE HAS BEEN SPECIFIED.
  21 NO DEFAULT POSITION WAS SPECIFIED.
  22 YOU CANNOT CALL THIS INTRINSIC WITH READ ONLY ACCESS.
  23 YOU CAN'T SET A PARAGRAPH BEFORE THE DEFAULT FOOTER.
  24 THE INTRINSIC CAN'T BE CALLED IMMEDIATELY AFTER ITSELF.
  25 AN ENVELOPE ALREADY EXISTS IN THE DOCUMENT.
  26 ENVELOPE CAN ONLY BE PLACED AT THE BEGINNING OF THE DOCUMENT
  27 THE PARAGRAPH TYPE SUPPLIED ( ! ) IS NOT A VALID TYPE.
  28 THE DEFAULT FORMAT SUPPLIED ( ! ) IS NOT VALID.
  29 YOU HAVE SPECIFIED TOO MANY TABS.
  30 INVALID TAB TYPE AT POSITION  ( ! ) IN THE TAB ARRAY
  31 INVALID TAB COLUMN AT POSITION ( ! ) IN THE TAB ARRAY
  32 TWO TABS HAVE BEEN SPECIFIED AT POSITION ( ! )
  33 ( ! ) IS NOT A VALID VALUE FOR THE LEFT MARGIN.
  34 ( ! ) IS NOT A VALID VALUE FOR THE RIGHT MARGIN.
  35 MARGINS ARE TOO WIDE - TOO LITTLE ROOM LEFT ON THE LINE.
  36 THE VALUE OF ( ! ) FOR THE TOP SPACE IS NOT VALID.
  37 VALUE OF ( ! ) FOR THE PARA LINE SPACING IS INVALID.
  38 VALUE ( ! ) FOR THE PARA POSITIONING OPTION IS INVALID
  39 VALUE OF ( ! ) FOR THE HYPHENATION OPTION IS INVALID.
  40 THE PARA INDENT PARAMETER SHOULD BE "0" OR "1"
  41 THERE IS NO ROOM TO ADD ANOTHER PARAGRAPH.
  42 LAST SEGMENT OF TEXT JUST WRITTEN COULD NOT BE SAVED (IT!)
  43 FAILED (IT!) WHILE TRYING TO ACCESS THE END OF THE PARA
  44 A FIGURE SPACE CAN NOT TAKE THE JUSTIFIED POSITION OPTION
  45 INVALID VALUE ( ! ) SPECIFIED FOR LINE NUMBER
  46 FIXED POSITION OPTION REQUESTED BUT NO LINE NUMBER GIVEN
  47 ONLY PARAGRAPHS OF TYPE TEXT MAY BE NON-PRINTING
```

## SET 32 WGETPARAFORMAT

```
  20 THERE ARE NO MORE PARAGRAPHS IN THE DOCUMENT
w 21 WARNING - THE PARAGRAPH TYPE IS INVALID.
```

## SET 33 WSETPARAFORMAT

```
20 NO PARAGRAPH TYPE HAS BEEN SPECIFIED.
21 NO DEFAULT POSITION WAS SPECIFIED.
22 YOU CANNOT CALL THIS INTRINSIC WITH READ ONLY ACCESS.
23 YOU CAN'T SET A PARAGRAPH BEFORE THE DEFAULT FOOTER.
24 THE INTRINSIC CAN'T BE CALLED IMMEDIATELY AFTER ITSELF.
25 AN ENVELOPE ALREADY EXISTS IN THE DOCUMENT.
26 ENVELOPE CAN ONLY BE PLACED AT THE BEGINNING OF THE DOCUMENT
27 THE PARAGRAPH TYPE SUPPLIED ( ! ) IS NOT A VALID TYPE.
28 THE DEFAULT FORMAT SUPPLIED ( ! ) IS NOT VALID.
29 YOU HAVE SPECIFIED TOO MANY TABS.
30 INVALID TAB TYPE AT POSITION  ( ! ) IN THE TAB ARRAY
31 INVALID TAB COLUMN AT POSITION ( ! ) IN THE TAB ARRAY
32 TWO TABS HAVE BEEN SPECIFIED AT POSITION ( ! )
33 ( ! ) IS NOT A VALID VALUE FOR THE LEFT MARGIN.
34 ( ! ) IS NOT A VALID VALUE FOR THE RIGHT MARGIN.
35 MARGINS ARE TOO WIDE - TOO LITTLE ROOM LEFT ON THE LINE.
36 THE VALUE OF ( ! ) FOR THE TOP SPACE IS NOT VALID.
37 VALUE OF ( ! ) FOR THE PARA LINE SPACING IS INVALID.
38 VALUE ( ! ) FOR THE PARA POSITIONING OPTION IS INVALID
39 VALUE OF ( ! ) FOR THE HYPHENATION OPTION IS INVALID.
40 THE PARA INDENT PARAMETER SHOULD BE "0" OR "1"
41 THERE IS NO ROOM TO ADD ANOTHER PARAGRAPH.
42 LAST SEGMENT OF TEXT JUST WRITTEN COULD NOT BE SAVED (IT!)
43 FAILED (IT!) WHILE TRYING TO ACCESS THE END OF THE PARA
44 A FIGURE SPACE CAN NOT TAKE THE JUSTIFIED POSITION OPTION
```

## SET 34 WREADFIGSPACE

```
20 THE CURRENT PARAGRAPH IS TEXTUAL AND NOT A FIGURE SPACE
22 THERE IS NO CURRENT PARAGRAPH
24 THE FIGURE SPACE HEIGHT (IN LINES) IS A REQUIRED PARAMETER
```

## SET 35 WWRITEFIGSPACE

```
20 OUTSTANDING REQUEST IS FOR A TEXT PARAGRAPH, NOT FOR A FIGURE
24 THE FIGURE SIZE (IN LINES) MUST BE SPECIFIED
26 FIGURE SIZE MUST BE IN RANGE 1 TO ! LINES
28 UNKNOWN VALUE (!) SUPPLIED FOR PLACEMENT OPTION
30 UNKNOWN VALUE (!) SUPPLIED FOR OVERLAY OPTION
32 UNKNOWN VALUE (!) SUPPLIED FOR BIND LINE OPTION
34 UNKNOWN VALUE (!) SUPPLIED FOR ROTATION OPTION
36 INVALID VALUE (!) SUPPLIED FOR PAGE BREAK OPTION
```

## SET 36 WREADLINE

```
  20 AN INTERNAL ERROR HAS OCCURRED THERE ARE NO MORE LINES
  21 INTERNAL ERROR (FS ! ) OCCURRED WHEN ACCESSING PARAGRAPH
w 22 WARNING - PENDING PAGEBREAK ON THIS LINE WAS NOT RECOGNIZED
```

## SET 37 WWRITELINE

```
  20 LENGTH OF THE TEXT SUPPLIED ( ! ) IS TOO LONG (MAX 250)
  21 THE LENGTH OF THE TEXT MUST BE GREATER THAN 0
  22 THE VALUE OF PARAMETER ! IS INCORRECT.
  23 AN INVALID TEXT CHARACTER  WAS DETECTED AT POSITION !
  24 MORE THAN 1 FORMAT CHARACTER WAS SPECIFIED AT POSITION !
  25 THE FORMAT CHARACTER AT POSITION ! WASN'T UNDER A SPACE.
  26 BOTH A USER AND A SYSTEM VARIABLE WAS SPECIFIED AT POS !
  27 A FORMAT CHARACTER CAN NOT BE PLACED IN A VARIABLE - POS !
  28 AN ASCII SPACE CAN NOT BE PLACED IN A VARIABLE - POS !
  29 ASCII SPACE AT POS ! WAS ENHANCED AS A VARIABLE
  30 MULTIPLE UNDERLINE CHOICE WAS SPECIFIED AT POS !
  31 THE CHAR AT POS ! CANNOT BE SUBSCRIPTED & SUPERSCRIPTED
```

## SET 38 WSETPAGEBREAK

```
  20 YOU CAN NOT MODIFY THE DEFAULT HEADER
  21 YOU CAN NOT MODIFY THE DEFAULT FOOTER
  22 YOU MUST HAVE READ/WRITE ACCESS TO SET A PAGE BREAK
  23 INTERNAL ERROR (FS ! ) OCCURRED WHEN ACCESSING PARAGRAPH
w 24 THERE IS ALREADY A USER PENDING PAGE BREAK PRESENT
w 25 A PENDING SYSTEM PAGE BREAK WAS REPLACED
w 26 THERE ALREADY EXISTS AN ACTIVE USER PAGE BREAK
w 27 AN ACTIVE SYSTEM PAGE BREAK WAS REPLACED
  28 INTERNAL ERROR (FS ! ) OCCURRED WHEN STORING PREVIOUS PARA
  29 INTERNAL ERROR (FS ! ) OCCURRED WHEN STORING CURRENT PARA
  30 INTRINSIC CAN NOT BE CALLED IMMEDIATELY AFTER WSETPARAFORMAT
```

## SET 39 WDELETEPAGEBREAK

```
  20 YOU MUST HAVE READ/WRITE ACCESS TO DELETE A PAGE BREAK
  21 YOU CAN NOT MODIFY THE DEFAULT HEADER
  22 YOU CAN NOT MODIFY THE DEFAULT FOOTER
  23 THERE CAN NOT BE A PAGE BREAK TO BE DELETED
  24 INTERNAL ERROR (FS ! ) OCCURRED WHEN ACCESSING PARAGRAPH
  25 A PENDING SYSTEM PAGE BREAK WAS FOUND AND DELETED
w 26 A SYSTEM PAGE BREAK WAS FOUND AND DELETED
w 27 PAGE BREAK DELETED BUT THE PARAGRAPH MIGHT BE INCONSISTENT
  28 AN INTERNAL ERROR(FS ! ) OCCURRED WHILE UPDATING THE PARA
w 29 THERE IS NO PAGE BREAK TO BE DELETED
```

# Appendix A

# Document Default Values

# Document Default Values

This appendix defines the values that HPWORD INTRINSICS will establish for the document objects when creating a new document file. They are also the default values taken by the intrinsics in the Document Attribute Group. These values are those used by HPWORD to build the TEMPLATE file.

## Document Information Object

This contains general and summary information about the document. Values marked with an asterisk can be provided at WOPEN. All can be accessed with WGETDOCINFO.

| | | |
|---|---|---|
| DOC'LANGUAGE | american'english | * |
| DOC'FORMAT | HPWORD | * |
| DOC'PAGINATED | no | |
| DOC'PAGE'CNT | 1 | |
| DOC'SUBJECT | ASCII spaces | * |
| DOC'AUTHOR | ASCII spaces | * |
| DOC'CREATOR | user's logon id | * |
| DOC'REVISOR | user's logon id | |
| DOC'SUBJECT | ASCII spaces | * |
| DOC'SIZE | 80 (approx 80 pages) | * |
| DOC'CREATE'DATE | date of creation | |
| DOC'REVISE'DATE | date of last revision | |
| DOC'PRINTED'DATE | date last printed | |

## Print Format Object

This object contains the print specification for the document. Values can be set with WSETPRINTINFO and accessed with WGETPRINTINFO.

| | |
|---|---|
| PRINT'NUM'COPIES | 1 |
| PRINT'FINAL | final copy |
| PRINT'ENVELOPE | do not print |
| PRINT'FIRST'PAGE | 1 |
| PRINT'LAST'PAGE | 1 |
| PRINT'SINGLE | single'sided |
| PRINT'VARDOCNAME | ASCII spaces |
| PRINT'ENVIRONMENT | ASCII spaces |
| PRINT'FORM'OPTION | 0 |
| PRINT'FORMS'MSG | ASCII spaces |

# Document Format Object

This object contains the overall document format characteristics. They can be accessed with WGETDOCFORMAT and can be changed at any time with WSETDOCFORMAT.

| | |
|---|---|
| DOC'PAGE'WIDTH | 8.5 inches (21.6 cm) |
| DOC'PAGE'HEIGHT | 11 inches (27.9 cm) |
| DOC'PITCH | 10 pitch |
| DOC'LINE'DENSITY | 6 lines per inch |
| DOC'PARA'SPACING | 2 half lines |
| DOC'PARA'INDENT | 0 |
| DOC'HEAD'SIZE | 5 lines |
| DOC'FOOT'SIZE | 5 lines |
| DOC'INSIDE'MARGIN | 1 inch (2.54 cm) |
| DOC'OUTSIDE'MARGIN | 1 inch (2.54 cm) |
| DOC'LINE'SPACING | 2 half line |
| DOC'STARTING'PAGE | 1 |
| DOC'AUTO'PAGE | on |
| DOC'HYPHEN | on |
| POSITION'OPT | left flush |
| DOC'TABS | none |

# Default Header Paragraph

HPWORD INTRINSICS follows the convention of HPWORD and builds a default header paragraph as the first paragraph in the document. This header consists of five blank lines. It can be accessed with WGETPARAFORMAT but cannot be deleted.

# Default Footer Paragraph

HPWORD INTRINSICS follows the convention of HPWORD and places a default footer paragraph after the default header. This footer consists of five lines. The first four are blank. The last line contains the system variable "$pagenum", which is centered. This paragraph can be accessed with WGETPARAFORMAT but cannot be deleted.

# Appendix B

# SPL Programming Example

# SPL Programming Example

The following example program illustrates the basic use of the HPWORD Intrinsic set.

Once the source has been created (shown below), it should be compiled and Prep'ed as follows. Note that if the document is to be opened with write access then the program must be preped with PH capability.

```
:SPL sourcefilename,uslfilename

:PREP uslfilename,progfilename

:RUN progfilename


$CONTROL                                              USLINIT,MAP
 <<*****************************************************************>>
 <<                PROGRAM:  progfilename                            >>
 <<                                                                 >>
 <<     This program prompts the user for an HPWORD format         >>
 <<     filename, then displays the document information to        >>
 <<     the user and reprompts for another filename.               >>
 <<                                                                 >>
 <<     The program is terminated by the user replying with        >>
 <<     'E' to the prompt.                                          >>
 <<                                                                 >>
 <<     Note that all HPWORD Intrinsic calls are followed by>>
 <<     calls to WERRORINFO to illustrate error handling.           >>
 <<*****************************************************************>>

 BEGIN

  EQUATE Old = 1,
         Ok  = 0;

  LOGICAL done,
          doc'open,
          doc'paginated;

  INTEGER Len,
          msg'len,
          numchar,
          doc'language,
          doc'page'cnt;
```

```
ARRAY dfcb(0:1999);
ARRAY l'inbuf(0:15);
BYTE ARRAY b'inbuf(*)=l'inbui;
BYTE ARRAY b'mess'cat(0:15):="INTRCAT.PUB.SYS ";
BYTE ARRAY b'filename(0:31);
ARRAY l'err'message(0:35);
BYTE ARRAY b'err'message(*)=l'err'message;
INTEGER ARRAY status(0:2),
              error(0:2);

ARRAY l'doc'subject(0:29);
BYTE ARRAY b'doc'subject(*)=l'doc'subject;
ARRAY l'doc'author(0:14);
BYTE ARRAY b'doc'author(*)=l'doc'author;
ARRAY l'doc'creator(0:14);
BYTE ARRAY b'doc'creator(*)=l'doc'creator;
ARRAY l'doc'revisor(0:14);
BYTE ARRAY b'doc'revisor(*)=l'doc'revisor;
ARRAY l'doc'create'date(0:7);
BYTE ARRAY b'doc'create'date(*)=l'doc'create'date;
ARRAY l'doc'revise'date(0:7);
BYTE ARRAY b'doc'revise'date(*)=l'doc'revise'date;
ARRAY l'doc'printed'date(0:7);
BYTE ARRAY b'doc'printed'date(*)=l'doc'printed'date;
ARRAY l'doc'language(0:10);
BYTE ARRAY b'doc'language(*)=l'doc'language;
ARRAY l'doc'paginated(0:5);
BYTE ARRAY b'doc'paginated(*)=l'doc'paginated;
ARRAY l'doc'page'cnt(0:3);
BYTE ARRAY b'doc'page'cnt(*)=l'doc'page'cnt;

ARRAY l'file'prompt(0:29):=
  "Enter Word document filename: ";

ARRAY l'msg1 (0:11):="Document Language     : ";
ARRAY l'msg2 (0:11):="Pagination State      : ";
ARRAY l'msg3 (0:11):="Number of Pages       : ";
ARRAY l'msg4 (0:11):="Document Subject      : ";
ARRAY l'msg5 (0:11):="Author                : ";
ARRAY l'msg6 (0:11):="Creator               : ";
ARRAY l'msg7 (0:11):="Revisor               : ";
ARRAY l'msg8 (0:11):="Document Creation Date: ";
ARRAY l'msg9 (0:11):="Document Revision Date: ";
ARRAY l'msg10(0:11):="Document last printed : ";
ARRAY l'msg11(0:31):=
"**************************************************************
  ***";
```

```
INTRINSIC PRINT,READ,ASCII,TERMINATE;
INTRINSIC (INTRDEC.PUB.SYS)
                      WOPEN,WERRORINFO,WGETDOCINFO,WCLOSE;

<<-------------------------------------------------------->>
<<                  ENTRY  POINT                          >>
<<-------------------------------------------------------->>
done:=FALSE;

DO
BEGIN
  DO
  BEGIN
    << Prompt user for a Word document filename >>
    doc'open:=FALSE;
    PRINT(l'file'prompt,-30,%320);
    move b'inbuf:=32(" ");
    len:=READ(l'inbuf,-32);
    IF l'inbuf="E " OR l'inbuf="e " THEN done:=TRUE
    ELSE
    BEGIN
      << Open the Word document file >>
      move b'filename:=b'inbuf,(32);
      WOPEN(dfcb,status,b'filename,old,,,b'mess'cat);
      WERRORINFO(dfcb,error,status,b'err'message,msg'len);
      PRINT(l'err'message,-msg'len,%40);
      IF status=ok THEN doc'open:=TRUE;
    END;
  END UNTIL doc'open OR done;

  IF not done THEN
  BEGIN
    << Access and retrieve the document information >>
    WGETDOCINFO(dfcb,status,,doc'language,doc'paginated,
                doc'page'cnt,b'doc'subject,b'doc'author,
                b'doc'creator,b'doc'revisor,
                b'doc'create'date,b'doc'revise'date,
                b'doc'printed'date);
    WERRORINFO(dfcb,error,status,b'err'message,msg'len);
    PRINT(l'err'message,-msg'len,%40);
```

```
<< Format data returned for display >>
move b'doc'language:="Unimplemented Language";
CASE doc'language-1 OF
BEGIN
  move b'doc'language:="American English       ";
  << Additional languages to be added here >>
END;

IF doc'paginated THEN move b'doc'paginated:="Paginated    "
ELSE                  move b'doc'paginated:="Unpaginated ";

numchar:=ASCII(doc'page'cnt,10,b'doc'page'cnt);

<< Output the document information to the
user >> PRINT(l'msg11,32,%40);

PRINT(l'msg1,12,%320);
PRINT(l'doc'language,11,%40);

PRINT(l'msg2,12,%320);
PRINT(l'doc'paginated,6,%40);

PRINT(l'msg3,12,%320);
PRINT(l'doc'page'cnt,4,%40);

PRINT(l'msg4,12,%320);
PRINT(l'doc'subject,30,%40);

PRINT(l'msg5,12,%320);
PRINT(l'doc'author,15,%40);

PRINT(l'msg6,12,%320);
PRINT(l'doc'creator,15,%40);

PRINT(l'msg7,12,%320);
PRINT(l'doc'revisor,15,%40);

PRINT(l'msg8,12,%320);
PRINT(l'doc'create'date,8,%40);

PRINT(l'msg9,12,%320);
PRINT(l'doc'revise'date,8,%40);

PRINT(l'msg10,12,%320);
PRINT(l'doc'printed'date,8,%40);

PRINT(l'msg11,3
2,%40);
```

```
      << Close the Word document file >>
      WCLOSE(dfcb,status);
      WERRORINFO(dfcb,error,status,b'err'message,msg'len);
      PRINT(l'err'message,-msg'len,%40);
    END;

  END UNTIL done;

   TERMINATE;
END. << of program >>
```

# Appendix C

# PASCAL Programming Example

# PASCAL Programming Example

This program performs the same function as the SPL programming example in Appendix B.

```
$code_offsets On$
$tables On$


{Program      : Pascdemo

                This program prompts the user for an HPWORD
                format filename, then displays the document
                information to the user and reprompts for
                another filename.

                The program is terminated by the user
                replying with <return> to the prompt.

                Note that all HPWORD intrinsic calls are
                followed by  calls to "wErrorInfo" to
                illustrate error handling.
}



 program PascalDemo (input, output);

 const
   AMERICAN_ENGLISH      = 1;
   OK                    = 0;
   OLD                   = 1;
   PAGINATED             = 1;
   UN_PAGINATED          = 0;

 type
   small_integer         = -32768..32767;

 var
   author                : string[40];
   creator               : string[40];
   create_date           : string[40];
   done                  : boolean;
```

```
  {"dfcb" can defined as anything provided that at least 2000
  words are allocated for it in the outer block.
  It could, for example, be defined as "array [1..1000] of
  integer".}
  dfcb                    : array [1..2000] of small_integer;
  error                   : array [1..3] of small_integer;
  err_message             : string[255];
  filename                : string[40];
  language                : small_integer;
  language_string         : string[40];
  len                     : small_integer;
  msg_len                 : small_integer;
  numchar                 : small_integer;
  open                    : boolean;
  page_count              : small_integer;
  paging                  : small_integer;
  paginated_string        : string[40];
  printed_date            : string[40];
  reply                   : string[40];
  revisor                 : string[40];
  revise_date             : string[40];
  mess_cat                : string[16];
  status                  : array [1..3] of small_integer;
  subject                 : string[255];


$Splintr 'intrdec.pub.sys'$
   procedure wClose          ; intrinsic;
   procedure wErrorInfo      ; intrinsic;
   procedure wGetDocInfo     ; intrinsic;
   procedure wOpen           ; intrinsic;


 Begin

   done := false;
   Repeat Begin {until done}

     open := false;
     Repeat Begin {until done or open}

      {Prompt user for an HPWORD document filename.}
      Writeln;
       Prompt ('Enter word document name : <return> exits : ');
       Readln (reply);

       if (reply = '') then Begin
         done := true;
  End else begin
   {Open the HPWORD document file.}
    filename := reply;
    mess_cat := 'INTRCAT.PUB.SYS '
    wOpen (dfcb, status, filename, OLD,,,mess_cat);

    wErrorInfo (dfcb, error, status, err_message, msg_len);
    Setstrlen (err_message, msg_len);
    Writeln;
    Writeln (err_message);

    if (status[1] = OK) then Begin
      open := true;
    End;
  End;

End until open or done;


if not done then Begin
 {Access and retrieve the document information.
  The strings below have to be initialized with spaces to
  set their lengths. If we didn't do this, 'Writeln' would
  output nothing when asked to print a string.}

  subject := '                                        '
          + '           ';
  author := '               ';
  creator := '               ';
  revisor := '               ';
  create_date := '        ';
  revise_date := '        ';
  printed_date := '        ';

  wGetDocInfo (dfcb, status,       , language, paging,
            page_count, subject, author, creator, revisor,
            create_date, revise_date, printed_date);

  wErrorInfo (dfcb, error, status, err_message, msg_len);
  Setstrlen (err_message, msg_len);
  Writeln;
  Writeln (err_message);

 {Format the language and pagination data for
 display.}
 case language of
```

```
         AMERICAN_ENGLISH:
           Begin
             language_string := 'American English';
           End;

        {Additional languages to be added here }

         Otherwise
           Begin
             language_string := 'Unimplemented language';
           End;

         End {case};

         if (paging = PAGINATED) then Begin
           paginated_string := 'Paginated';
         End else Begin
           paginated_string := 'Unpaginated';
         End;

        {Output the document information.}
         Writeln;
         Writeln ('******************************************');
         Writeln ('Document language : ', language_string);
         Writeln ('Pagination state  : ', paginated_string);
         Writeln ('Number of pages   : ', page_count);
         Writeln ('Document subject  : ', subject);
         Writeln ('Author            : ', author);
         Writeln ('Creator           : ', creator);
         Writeln ('Revisor           : ', revisor);
         Writeln ('Creation date     : ', create_date);
         Writeln ('Revision date     : ', revise_date);
         Writeln ('Last printed on   : ', printed_date);
         Writeln ('******************************************');
         Writeln;


        {Close the HPWORD document file.}
         wClose (dfcb, status);

       wErrorInfo (dfcb, error, status, err_message, msg_len);
       Setstrlen (err_message, msg_len);
       Writeln (err_message);
     End {if not done};

 End until
done;

 End {of program PascalDemo}.
```

# Index

# Index