UNITED STATES DEPARTMENT OF THE INTERIOR
GEOLOGICAL SURVEY

WOLF: Automatic Typing Program

By

Gerald Ian Evenden

U.S. GEOLOGICAL SURVEY
OPEN-FILE REPORT 82-379

Preface

In order to create a complete document for programmers involved with program modification and maintenance as well as a documentation for general users, it is necessary to include the technical description and details of implimentation. However, the user documentation, along with appendix A, can be extracted and distributed as a complete manual.

Abstract

A FORTRAN IV program for the Hewlett-Packard 1000 series computer provides for automatic typing operations and can, when employed with manufacturer's text editor, provide a system to greatly facilitate preparation of reports, letters and other text. The input text and imbedded control data can perform nearly all of the functions of a typist. A few of the features available are centering, titles, footnotes, indentation, page numbering (including Roman numerals), automatic paragraphing, and two forms of tab operations. This documentation contains both user and technical description of the program.

Introduction

The program WOLF (Word Oriented Line Formatter) transforms free form source text and special commands into formalized typewritten pages. A nearly exact analogy of the function of this program is to that of a typist. Although the source text must be in computer readable form and a typing operation is normally required to initially create this text, the typing is far less demanding than direct typewriter operation. In fact, those who generally prepare typewritten rather than handwritten drafts for typists could, just as easily, enter the text on the host computer system directly.

The principle advantage of WOLF is not in simple one-step typing operations (draft to final copy), but in multiple draft applications. For example, in the preparation of reports there are often at least 3 drafts produced: 1) an initial draft reviewed and corrected by the author 2) a second draft reviewed by critics and 3) the final draft. When using this system the only part of the job consuming considerable time and labor is the conversion of the source text to computer readable form. Subsequent corrections are readily and quickly inserted in the source text and the next copy is produced quickly and, just as important, accurately. Usage is, of course, not limited to reports. "Personalized" form letters are an excellent example where a general text body is quickly changed to reflect specific details and then typed. Even filling out frequently employed forms or even creating new forms can be expedited. The principle thing to remember is that the WOLF system is quite flexible and so are its potential applications.

Usage of WOLF is as simple or complex as the job requires. For example, to prepare the body of a letter the beginner is urged to just enter the text into a file and process that file with WOLF. To start a paragraph the only thing the user needs to know is that the paragraph must start at the beginning of an input line and a blank must preceed the capital letter of the first word. To add the heading and closing of the letter the user only needs to learn the usage of four WOLF command words.

In the following description of the use of WOLF the basic concepts and modes of operation are covered first. These are followed by description of editing characters and command words which control WOLF functions. Finally, a control summary and an error message summary are included.

Basic Principles

Program WOLF is basically a processor of "words". Most of the words it processes are the same as normal language words. However, there are other words recognized by WOLF so that a specific definition of a word needs to be made. A word in this system is defined as any continous string or group of characters separated by a delimiting condition which may be a blank or space, a tab character, or the end of the input line of text. Whether the blank or tab is used as a word separator is dependent upon which of two modes, "fill" or "no-fill" (discussed in more detail later), the program is processing as the input text. In fill mode the blank is a separator and the tab editing character (the > character described in more detail in the Editing Characters section) assume printable character status, while in the no-fill mode blanks are treated as printable characters and the tab editing character is a word separator. For example, in the fill mode the list

```
cow         and/or
WOLF?       second>third>fourth
$12.34      Thus,
end.        on-line!
```

contains eight words even though one might be considered a number, and others a connection of words. Note that the punctuation is also part of the word. In no-fill mode the same list contains six words since the second line has three words (because of the tab editing character) and the others have only one. Understanding the word concept is critical in later usage of certain editing functions.

The fill mode also has an important function other than defining word separations. In fill mode the input is considered a continuous stream of words to "fill" the output printed line. When an input word is encountered which won't fit in the current line, the current line is printed and the left over word is placed first in the next line. This process is repeated until the text is completed or a control condition occurs. To demonstrate this operation consider the following listing of a source text file and the resultant fill mode of output:

Source text...

```
This is an example
of how  fragmented
the
source text     can be  in fill     mode.
The resultant    text, however,
   does not reflect
                this    chaos.
We could go to the extreme and

put
one
word
```

per
line.

WOLF processed resultant text...

> This is an example of how fragmented the source text can be in fill mode. The resultant text, however, does not reflect this chaos. We could go to the extreme and put one word per line.

Since it is often desirable to have the last character of each line fall on the right margin, the "justify" function is also included in WOLF. For this operation the program simply determines the number of spaces left over at the end of the line and distributes them between the words. The following shows the effect of justifying the previous text example:

> This is an example of how fragmented the source text can be in fill mode. The resultant text, however, does not reflect this chaos. We could go to the extreme and put one word per line.

It is generally recommended for reports but will give away the machine processing when used for letters. Both fill and justify are the initial modes when executing WOLF.

In the no-fill mode, each source line produces one output line. Its primary purpose is in producing tables, listings or other text where automatic filling of output lines is not desired (i.e. poetry). If, for example, the previous source text was processed with no-fill mode, the identical text would be returned.

Obviously, typing is not always constant transformation of the source text. There must be provisions to adjust margins, control pagination, underscore words, etc. In standard human typing these controls are often verbally given, specified by margin notes in the source text and/or a wide variety of annotations and markings in the text itself. In an automatic system the control must be formalized into a structure or syntax understandable by both user and machine. In the WOLF system there are more than forty controls or commands which affect every aspect of text preparation. It should be noted, however, that the average usage of the program will employ a small percentage of these commands and their usage will become readily automatic. These controls are subdivided into two types: editing characters and command words.

Editing Characters

Editing characters are employed to perform special operations, or operations normally performed by control keys on the typewriter such as backspace and tab. These latter functions exist as characters in the computer system but are difficult to handle when using a text editor on the source text. To overcome this problem these functions are indicated by printable characters recognized by the WOLF program as control, but recognized as simply another printable character by other programs. The selected characters employed for editing control are used relatively infrequently in most text, but should the need arise, the user can readily change or, in some cases, suppress their meaning.

Underscore editing character

Since underscoring of words is a common part of typing operations, the underscore character, _, is included as an editing character. The action of the single underscore editing character only affects the current word and is similar to a toggle switch: turn on when off, and off when on. When two underscore editing characters occur consecutively all following words are to be underscored until a second double underscore editing character pair is encountered. The following example illustrates its use:

_WOLF	->	WOLF
_r_a_d_a_r	->	radar
__now is the time?	->	now is the time?
__no. 52.75__ is	->	no. 52.75 is

As can be seen from some of the above examples punctuation and special symbols are not underscored. How to underscore these characters is shown in the examples of literal editing.

Tab editing characters

Two forms of tab editing characters are provided: normal tab, >, and the tab with "fill string" or fill-tab, ^. Except for the condition discussed in the following paragraph, each of the tabs is only performed in the no-fill mode of operation since their effect in fill mode would normally be hard to predict. Each time either tab character is encountered the next word is positioned at the next tab position. Remember, tabs in no-fill are usually word separators. In the case of the tab with fill string, a set of up to four characters, period and blank initially, will be inserted from the end of the last word to the next tab position. Setting tab stops and altering the fill string will be discussed later in the command word section. In the following

examples the tab positions are assumed to be set at 10 and 20:

```
name>phone>office      ->    name      phone     office
>no.>no.               ->              no.       no.
Smith>165>201          ->    Smith     165       201
Jones^140^Bldg B       ->    Jones. . .140. . . .Bldg B
James^^207             ->    James. . . . . . . .207
|^|^|                  ->    |_________|_________|
```

In the last example line the fill string had been changed to _.

Either tab editing character may be used in fill mode when text has been indented to the left of the left margin and the first character of the word following a tab would be output to a column left of the left margin. In this case, the tab editing character will cause a tabbing operation to the left margin column. This feature is especially useful in paragraph or section numbering as shown in the following source text examples:

```
1.^Now is the time for all good men to come to
the aid of desperate programmers.

S.2>This section deals with tabs.

Sub. section 2>This shows how the tab is ignored when
  the next word is to the right of the left margin.
```

Assuming that indentation was 8 columns to the left of the left margin (these operations will be discussed later) the resultant text would appear as:

```
1. . . . Now is the time for all good men to come to the
         aid of desperate programmers.

S.2      This section deals with tabs.

Sub. section 2 This shows how the tab is ignored when
         the next word is to the right of the left margin.
```

Backspace editing character

The backspace editing character, <, is employed to print a second character over the previous character, such as accents and creating special mathematical symbols. It is not employed to rub out or correct an error as is typical in manual typing. The only restriction in its usage is that the user cannot backspace beyond the beginning of a word. The following are examples of its usage:

```
Saute<'          ->   Sauté
Di<'as           ->   Días
=</               ->   ≠
1000<<<///       ->   1ØØØ
```

Literal editing character

The literal editing character, *, causes the next following character in the text to be treated as a printable character and thus part of a word. Its principle usage is to allow blanks and other editing characters to become part of the output word. When two consecutive literal editing characters occur, all characters, except the underscore character, are considered part of the input word until the second pair of literal editing characters or the end of the input line occurs. The following are examples of literal editing:

```
2* PM                    ->  2 PM
*<<*_                    ->  <
_12.<*_34                ->  12.34
**12 AM**                ->  12 AM
joined*_word             ->  joined_word
*<*<bracketed*>*>        ->  <<bracketed>>
12.5*^-10                ->  12.5^-10
**L E T T E R  S P**     ->  L E T T E R  S P
**_WOLF**                ->  WOLF
symbol *  is             ->  symbol   is
**__A T** odds           ->  A T odds
```

Most of the above usage is obvious except for the two time examples. In some cases the user will want to be sure certain word groups are treated as one word so that there will be no possibility of their being split onto two output lines. 2 PM looks better if it is on one line than if the 2 and the PM are split.

The literal editing character cannot be included in text. If inclusion of the the literal editing character is required, the user must change it by use of the \CL command word discussed later.

Hyphen editing character

A distracting and undesirable appearance of fill mode text can be caused by words which will not fit on the current line because of their length and, consequently, create an excessive number of blank fill characters in the line or an excessive indentation of the right margin. In normal typing these words are hyphenated by the typist. To allow this system to hyphenate long words the user may insert one or more "ghost" hyphens, =, between word syllables. For example, the following text shows several words containing the ghost hyphens:

```
\MR18,52
Hy=phen=ation should be used spar=ingly.
However, extreme blank pad=ding, "rivers"
in the text, or other dis=trac=tions
in the over=all ap=pear=ance of the text
```

```
make hy=phen=ation extemely useful.
Note that shorter text lines will re=quire
greater use of the hyphen.
```

The result of processing the above text is:

> Hyphenation should be used sparingly. However, extreme blank padding, "rivers" in the text, or other distractions in the overall appearance of the text make hyphenation extemely useful. Note that shorter text lines will require greater use of the hyphen.

As can be seen, the ghost hyphen is deleted from the text when not needed. The ghost hyphen editing character has effect in fill mode only and is simply another printable character in no-fill mode and literal strings.

In practice, the use of the ghost hyphen in text with lines of 65 or more characters is generally not needed. In any case, it is usually desirable to prepare the initial draft without ghost hyphens and then insert them, if required, prior to a secondary processing.

Line shift editing characters

Two editing characters { and } allow respective up and down half line shifting of the following characters. This shifting remains in effect unil another line shift editing character or the end of the word. Typical applications are in superscripts, subscripts and mathematical typing as shown in the following example:

`latitude 27{o 30'`	->	latitude 27^{o} 30'
`The acid H}2{SO}4 is`	->	The acid H_2SO_4 is
`1.5x10{-5`	->	$1.5x10^{-5}$
`footnote{_3/ type`	->	footnote$^{3/}$ type
`s={d-a<<<___<<}}T`	->	$s=\frac{d-a}{T}$

These editing characters can, of course, be used only with devices capable of forward and reverse half line spacing.

Command Words

Command words are used to control general aspects of WOLF operations such as mode selection, margin control, indentation, etc. In this section the command words are grouped into generally related operations. As required, further details of WOLF operation are included to help explain the usage of the particular command words.

Each command word consists of a command prefix character, \, followed by a one or two letter function identifier (ie. \JR, \FS, \S). Although the following discussion of the commands employs upper case letters, lower case letters are equivalent: \JR = \jr = \Jr.

Many commands require additional control information to be supplied by the user. This information is transmitted to the program by means of numeric or character string values, called arguments, immediately following the function identifier. Numeric arguments, denoted in the text by the lower case letter n (or n1,n2,... for multiple arguments), may be a signed (+ or -) number. If the sign is omitted they are assumed positive. Character string arguments are denoted by the lower case s (or s1,s2,... for multiple arguments) and must consist of a character string enclosed by the literal editing character * (note: the character string may also be ended by the end of the input line). Each argument of multiple argument command words must be separated from the previous argument by a comma. Arguments may be omitted and in such cases the action taken is described. Where arguments are omitted prior to secondary arguments commas must be employed (ie. \MR,60) to show the absence of the previous argument). Arguments must agree with the type specified (numeric or character string) but are ignored when employed where no argument is expected. The user may use this latter feature to include comments in the source text which do not affect program operation nor appear in the output text (ie. \FS*start of main section*).

Placement of the command word in the input text depends upon the basic mode in effect. In fill mode command words may appear anywhere in the source text In no-fill mode they may only appear at the beginning of the input lines. It is generally recommended that command words occur on separate lines because their usage becomes more obvious and readable in the source text. Remember that command words are words and a blank character or end of line condition is required before and after every command word.

Break operation
\B

The break command word, \B, is not often directly employed by the user. However because the operation it performs is frequently a part of other WOLF command words special attention needs to be given to it. When later discussion employs the word "break" it is referring to this operation.

In fill mode it is often required to stop the current "filling" of an output line with input words and to start a new output line and print it prior to continued operation. When justify mode is in effect the current output line is not justified when the break is employed. Use of break in no-fill mode is meaningless since command words can only occur at the beginning of the input line where, in a sense, the break has already occured (no-fill mode can be considered as fill mode with a break at the end of each input line).

Mode selection
\FS \FR \JS \JR

To change the fill, no-fill and justify modes of WOLF processing four command words are provided. \FS, \FR, \JS and \JR initiate fill, no-fill, justify and stop justify modes respectively. Since fill mode is required for justify operation \JS also initiates fill mode if it is not already in effect. All of these command words cause a break.

Horizontal control
\MR \I \T \SF

When executing WOLF the left margin is initially set to column 1 and the right margin is set to maximum print column value of the execution parameter (column 72 if not specified). During execution these margins can be modified by \MRn1,n2 where n1 is the left margin column and n2 is the right margin column number. The only restrictions are that the left margin must be greater than 0 and less than the right margin and the right margin cannot be greater than the maximum number of columns specified at execution.

The margin command word causes a break and both left and right margin arguments are optional (ie. set left margin by \MR5, right margin by \MR,68 and both margins by \MR5,68) and when omitted the omitted margin remains at its previous value. If both margins are omitted the only action is the break.

Indentation, \In, causes a break and the start of the next line to be shifted n spaces from the current left margin. If n is negative the shift is to the left of the left margin and, consequently, the left margin must have been previously set to a number greater than or equal to the positive value of n. Also, a positive n should not cause an indentation beyond the right margin. If n is not specified or 0 no operation is performed except the break.

Tab stops, employed by the tab and fill-tab editing characters are set with the \T command word. The WOLF system provides for up to 33 tab stops with initial settings of columns 6, 11, 16, 21, ... etc. (every fifth column). When the user enters new tab stops each tab entry in the list must be greater than the previous tab stop entry and the last tab entry terminates the list. This also means that a null or absent value also terminates the list (ie. if \T5,15,,25 is entered the last tab stop is 15). If the user attempts to tab beyond the last established

tab stop an error condition will occur.

The \SFs command word allows the user to set the fill string employed by the fill-tab editing character. The string s may contain from 1 to 4 characters which will be repeated until the next tab stop. If more than 4 characters are in the string only the first 4 will be employed. If the string s is not specified the fill-tab editing character will act the same as the simple tab. The initial setting of s is ". " (period-blank).

Centering control
\C \CS \CE

Centering of text can be used to prepare title pages and chapter or section headings. Three command words are provided for centering operations. \C indicates that the following text on the same input line as the command word or the next non-blank input line should be centered between the current left and right margins. \CS indicates that all following input lines are to be centered until the \CE command word is encountered. When centering is initiated (\C or \CS) the mode is forced to no-fill and all no-fill rules apply to the source text. At the end of centering operation the modes previously in effect are restored. Both \C and \CS cause a break.

Paragraph control
\P \AP \AR \PP

WOLF can perform "automatic paragraphing" (break, added line spacing and indentation) if the source text line in fill mode begins with one or more blanks followed by an upper case letter. Selection of automatic paragraphing (initial condition of program) is made by the \AP command word and can be halted by the \AR command word. In certain cases paragraphing operations need to be forced by the use of the \P command word (ie. the paragraph starts in the middle of the source text line, does not start with an upper case letter or automatic paragraphing is turned off).

The \PPn1,n2 command word allows the user to control the details of the paragraphing operation. The optional argument n1 specifies the number of added blank lines to preceed the new paragraph (initially set to 1) and the second optional parameter, n2, specifies the indentation of the first word of the new paragraph line (initially set to 5). Note that the same rules of indentation of the \I command word apply to n2. Omission of either argument does not cause the current associated value to be changed.

Changing editing characters
\CL \CC \CU \CT \CB \CF \CH \CK

Command words \CLs, \CCs, \CUs, \CTs, \CBs, \CFs, \CHs and \CKs1,s2 allow changing the respective literal editing character, command word

prefix, underscore, tab, backspace, fill-tab, hyphen and line shift editing characters. The character string s (if more than one character is present only the first is used) replaces the current editing character. In the case of \CK s1 is the line up-shift editing character and s2 is the line down-shift editing character. The arguments for \CU, \CT, \CB, \CF, \CH and \CK may be omitted and in such cases the corresponding editing function is eliminated. The literal editing and command prefix characters cannot be omitted. In all cases the new editing character must not be a blank, sign (+ or -) character, numeric digit or an editing character currently in use. As an example of usage, \CL*$* and \CU will cause the literal editing character to become $ and the underscoring operation is eliminated.

In practice the only required usage is the \CL command word when the literal editing character is to be employed in the output text. All other editing characters can be incorporated by prefixing them with the literal editing character.

Page operations

Before proceeding it is necessary to discuss the details of WOLF operation with respect to pagination. All text output is in terms of pages and WOLF always performs the following sequence of operations for each page:

1 Pause if output to teletype and wait for user response.
2 Increment page number.
3 Print page number if page number line less than first text line.
4 Print title group A.
5 Print title group B.
6 Print title group C.
7 Print main text body, including figures.
8 Print footnotes.
9 Print page number if page number is not already printed.
10 Eject page.

Any one of the above operations except 10 can be omitted but the order cannot be altered (ie. title B cannot be printed before title A). This operating sequence is not started until a line of main text is ready for output, a page command word (\PG) or a \TO command word condition is encountered. The \PG and \TO commands are discussed in more detail later.

Operations 3 through 9 have associated line numbers defining the starting position of the output on the page (note: line 1 is top of page). In each of these cases, however, if the line number of the next output line is greater than the specified line number of the operation, the operation starts on the next line number. For example, if the page number line is 4, the title A line number is 6 (titles B and C are not used) and the first line of main text is specified as 6, the following output lines will be employed: page number on line 4, title A on lines 6, 7 and 8 and the text starts on line 9. Everything is "pushed down" based on the effect of the previous operations except the end of main

text and footnote lines which are always cut off at the specified last line.

Vertical control
\S \SP \CP \PG

The \Sn command word causes a break and a space of n blank lines to be output. In fill mode it is the only way, besides paragraphing, that blank lines can be introduced into the output text. If the value of n is less than 1 or omitted 1 blank line is assumed. Spacing will not continue from one page to the next. That is, if 5 lines are left on the current page and an \S10 command word is given, spacing is only made to the bottom of the current page and the residual 5 spacing lines are ignored.

The user may change the number of blank lines between output lines (what is normally called "spacing") by the \SPn command word where n is the number of blank lines desired minus 1. For example, a "double spacing" request is \SP2, and "single spacing" is \SP1 or \SP. If n is less than 1 or omitted 1 is assumed. The initial value of n is 1.

For controlling the vertical size and position of output on a page the \CPn1,n2,n3 command word is employed. n1 specifies the first line for text and n2 the last line. n3 specifies the first line on which footnotes may start so that some normal text will appear on a page prior to footnote output. To provide one inch top and bottom margins for standard 66 line pages (11 inch stock) the initial value of n1 is set to 6 and n2 is set to 6 less than the maximum lines per page specified at execution time. The initial value of n3 is line 10. n1 and n2 are similar to the left and right margin values and the same rules apply (ie. n1<n2, etc.). When any argument value is absent the corresponding control remains unaltered.

The \PGn command work provides for page ejection. If the value of n is 0 or omitted a break is performed, applicable figures, footnotes and page number are output and a new page is started. If the value of n is greater than 0 the previous operations are only performed if there are less than n lines remaining on the current page. Major divisions of a report frequently employ \PG or \PG0 to start the section on a new page. Use of \PGn, where n is positive, is primarily at the head of minor divisions where the user wants to ensure that the heading with one or more lines of text body will occur on the same output page. If n is greater than the number of text lines per page then the command is identical to the \PG command.

A third form of the \PG command is where n is less than 0. This case is the same as the positive n except the break function is not performed. The priciple usage of the negative argument \PG command is where the user does not want a shortened line (in fill mode) caused by the break function to occur at the bottom of the page if the page ejection is made. Note that one or more words preceeding the command may be transfered to the next page along with the following text. The negative argument form is identical with the positive argument form in

no-fill mode.

Page numbering control
\PE \PS \PO \PN \AN \RN \NC

The WOLF system provides automatic page numbering services. The internal page number is incremented at the start of each page and is printed at either the top or the bottom of the page, depending upon user selection. Printing of the page number may be stopped by the \PE command word or re-enabled by the \PS command word (initial condition). The page number is kept current even when not being printed. Since printing of page number 1 is often suppressed the WOLF system will not print page 1 unless the \PO command word is given.

Changing of the current page number can be made at any time with the \PNn command word where n is the new page number. This is particularly useful when segementing the preparation of large reports into independent jobs. Initially the page number is set to 0. Remember that the page number is incremented when the program starts to print a new page. Consequently, if a \PG command word is followed by \PNn the page number printed will be n+1 since the output of the new page is not started until text is to be printed or a \TO command word (see miscellaneous commands) is given. If a \PNn is specified in the middle of page output the result will depend upon whether the page number was printed at the top of the page or the bottom. If printed at the top it will obviously contain the previous page value and the next page will contain n+1. If printed at the bottom the new page number will be n. Because of the confusion that can exist it is recomended that \PNn be employed after a \PG command word where the user will be sure of the operational sequence.

Two additional command words \RN and \AN allow selection of Roman and Arabic numeral presentation of the page number (Arabic is the initial mode). Roman numerals are frequently employed on pre-introduction sections of a report. Typically, if such page numbering is employed in a one step execution of WOLF the command sequence would be: \RN at the beginning of the report followed by the title page, preface, etc. and then prior to the main section of the report the \PG \AN \PN0 command word sequence would be given.

To control the position and page number annotation the \NCs1,s2,n1,n2 command word is used. s1 and s2 are strings of up to 10 characters each which will preceed and follow, respectively, the page number. For example, \NC*Page * or \NC*- *,* -* will cause all subsequent page numbers to appear as "Page 10" or "- 10 -". If s1 and/or s2 is omitted the respective previous string remains in effect. If any numeric value is in s1 and/or s2 (this is the only place a numeric value is allowed in a string argument) the respective content of the string is deleted. If more than 10 characters are in either string only the first 10 are used. Initially s1 and s2 are empty.

Arguments n1 and n2 of the \NC command control the column position of the first character of the total number string and output line number

of the page number. If n1 is omitted the previous position remains in effect and if n1 is less than 1 (initial condition) the page number string is centered between column 1 and the maximum column number specified at execution (it is not centered between margins). If the line number, n2, is less than the first text line, the page number is printed at the top of the page; otherwise it is printed after the last output text line. Initially n2 is specified as the maximum number of lines - 4. Omission of n2 leaves the previous value in effect.

Titles and footnotes
\TA \TB \TC \FT \EN

Three title line groups, appearing at the top of each page, and footnotes are controlled with the \TAn, \TBn, \TCn and \FT command words. Creating titles and footnotes is similar to starting a new output file in that text entry may contain command words and editing characters as shown in this example:

```
\TA \MR50 \FR
Cape Cod Windmill Co.
Murky Pond, MA 02541
(617) 999-9999
\EN
  ........
```

The most important factor in creating titles or footnotes is that control established in the main text does not apply to these entries. Similarly, the mode and/or margin command words in the above example do not alter the mode and/or margins of the main text nor footnotes. In addition, all commands generally relating to basic page control, figures and other title/footnote commands are not allowed in the title/footnote text. The main thing to remember is that main or body text, titles and footnotes are three entirely independent processing entities, each initialized in the identical manner.

Title entries are separated into three groups (A, B and C) each allowing up to 5 lines (including blank lines) of text. To enter text in each of these groups the \TAn, \TBn and \TCn command words must preceed the text and where n is a positive number or omitted. Title text must be followed by the \EN command word to return to main text processing. The positive value of n, if specified, denotes the first output line on which the title is to be printed. All title groups are under the same general control such that command words like \MR or \SP used in \TA will affect \TB and \TC processing.

If the argument n is specified, but less than 1, two control functions are involved and title text does not follow. When n is zero the title lines for the specified group are deleted and when n is less than zero the printing status is toggled (ie. print to no-print, no-print to print) and the title text is not altered.

The \FT command word starts the entry of text into the footnote section at the bottom of the page. Footnote input is terminated by the

\EN command word (same as titles). Up to 40 lines excluding blank lines generated by \S, multiple spacing or paragraphing may be placed in the footnote section (the blank lines are regenerated on output). To ensure that a footnote reference in the main text starts on the same page as the reference, the footnote entry should immediately follow the referencing text word as shown in this example:

```
source text reference 1/
\FT 1/ See last chapter for more details. \EN
cannot be applied.
```

The WOLF system keeps track of footnote requirements so that if the footnote cannot be placed on the same page the line with the reference word in it is not printed on the current page and a new page is started. When footnote output cannot be completed on the current page it is continued on the next page.

When a footnote is entered prior to the completion of output of the previous footnote it is appended to the previous footnote. Again, assurance is made that the appended footnote will occur on the same page as the footnoted word of main text.

Footnotes are placed at the bottom of the page using no more lines than are allowed or required. In addition, a blank line followed by a line of dashes and a second blank line preceed the footnote text.

Input file control
\FI \EF

When the \FIs command word is encountered subsequent source text is obtained from the file specified by the contents of s. The new input is incorporated into the input stream as if it was part of the old file. When the text from the new file is exhausted the system resumes processing from the old file. Input files may be nested (ie. the new file contains an \FI reference to another file) up to 5 levels deep. The argument string s must be specified and the details of its contents are dependent upon the file naming conventions of the host computer system.

This feature may be used to put together parts of a long report, prepared and corrected independently, into a single "final edition" run such as in the following example:

```
\FI*TITLEP* \PG
\FI*PREFAC* \PG
\FI*CHAP1* \PG
\FI*CHAP2* \PG
\FI*APPNDX* \PG
```

Another application is in form letter preparation where the constant portion of letter text resides in a file which has references to secondary files containing the date, addressee's name and address and if

necessary certain particular details. The advantage here is that each time the form letter is typed only the detail files need to be altered.

The \EF command word signifies the end of the current input file. It is not required but can be used to delete ending sections of a source file without actually deleting the text.

Miscellaneous functions
\TO \NP \FG

The \TO command word has effect only when WOLF is at the top of the output page. It causes the system to process the page heading functions (steps 1 through 6 previously discussed in the page operations section). For example, if a new section is started and a new title is to be specified for the first page followed by a second title line on subsequent pages, the following source text is required:

```
\PG*start of section A*
\TB \C Section A.
\EN \TO*forces current titles to be printed*
\TB \CS Section A.
(Con't)
\CE
\EN*end of title for remaining pages.*
   ..........
```

When WOLF is writing output to a teletype device it automatically pauses at the top of each page to allow the user to change paper and to notify the program when to continue. However, if continuous form paper material is being used, this operation can be bypassed with the \NP command word.

For certain reports figures are included in the text. Figures require a contiguous sequence of blank lines where the \S command word cannot be used since it is usually not known where the resultant operation is going to occur in the output text. If it occurs near the bottom of the page insufficient space could result. The \FGn command however causes n blank lines to be inserted in the text either immediately (if there's room) or on subsequent pages. If there are additional figure requests before the current request is satisfied they are added to a figure list of up to 17 entries. When there is a list of entries each is satisfied, in order of entry, as soon as possible and deleted from the list.

Control summary

This section is provided as a quick reference to WOLF program control for users already familiar with its basic operation.

Editing characters

* literal editing character causes the next character to be treated as printable character even though it is a blank or other editing character. When ** occurs all following characters, except the underscore and literal editing characters, until a second ** or end of the input line, are treated as printable characters.

_ underscore editing character causes the underscoring of the remaining letters of the word or until a second underscore editing character is encountered. Two underscore editing characters (__) initiates underscoring of all words until a second pair of underscore editing characters occurs.

< backspace editing character causes backspacing prior to the printing of the next text character.

> tab editing character (no-fill mode only) causes horizontal spacing to the next tab position. See \T command word for initial tab settings and how to change them.

^ fill-tab editing character (no-fill mode only) performs like > except that instead of spaces a string of characters is repeated.

= ghost hyphen editing character (fill mode only) allows a word to be split across two output lines. The = character is replaced with a normal hyphen (-) if the preceeding segment will fit on the current line and the remaining characters of the word are placed on the next output line. If hyphenation is not performed the ghost hyphen editing character is omitted.

{ up-shift editing character shifts the output line up 1/2 line.

} down-shift editing character shifts the output line down 1/2 line.

Command words

In the following alphabetically ordered list of WOLF commands, optional arguments are enclosed in square brackets [] and n denotes numeric arguments and s denotes character string arguments. Values enclosed in parentheses after n or s are minimal assignments followed by initial system assignments when applicable.

\AN (initial)
Set output of page numbers to Arabic numeral form.

\AP (initial)
Perform automatic paragraphing. Automatic paragraphing (fill mode only) occurs when one or more blank characters start a line followed by an upper case letter.

\AR
Halt automatic paragraphing.

\B
Break processing of current output line and start new output line with following text. Last output line is not right justified.

\C
Perform break function (\B) and center between margins the text when it occurs a) on current input line when non-blank characters are present or b) first following non-blank input line. No-fill mode is temporarily in effect.

\CB[s(,<)]
Change backspace editing character to s. If s omitted, operation is suspended.

\CCs(,\)
Change command prefix character to s.

\CE
Halt centering operation initiated by \CS. Fill and justify modes in effect prior to \CS are restored.

\CF[s(,^)]
Change fill-tab editing character to s. If omitted, operation is suspended.

\CH[s(,=)]
Change ghost hyphen editing character to s. If omitted, the operation is suspended.

\CK[s1(,{)][,s2(,})]
Change respective up-shift and down-shift editing characters to s1 and s2. If respective values are omitted the operation is suspended.

\CLs(,*)
Change literal editing character to s.

\CP[n1(1,6)][,n2(,maxlines-6][,n3(,10)]
Set first (n1) and last (n2) text lines. n3 is first line number on which footnote output may begin.

\CS
Perform break function (\B) and center all following non-blank input lines. No-fill mode is in effect.

\CT[s(,>)]
Change tab editing character to s. If omitted, operation is suspended.

\CU[s(,_)]
Change underscore editing character to s. If omitted, operation is suspended.

\EF
End of current input file.

\EN
End of title or footnote entry.

\FGn
Space, when available, n lines for figure entry.

\FIs
Start processing input from file defined by s. Input files may be nested five deep.

\FR
Perform break operation (\B) and start no-fill and no-justify mode of operation.

\FS (initial)
Start fill mode of operation.

\FT
Start footnote entry.

\In
Perform break function (\B) and indent start of next line n spaces from left margin.

\JR
Perform break operation and stop justify mode of operation.

\JS (initial)
Perform break operation and start justify mode of operation. Fill mode is also initiated if necessary.

\MR[n1(1,1)][,n2(1,max cols)]
Perform break operation (\B) and set left margin (n1) and/or right margin (n2). n1≤n2.

\NC[s1(,)][,s2(,)][,n1(0,0)][,n2(0,maxlines-4)]
Set left (s1) and right (s2) bracket strings of page numbers. Any numeric value in s1 or s2 deletes the string. n1 specifies column of first page character and if ≤ 0 then centers page number. n2 specifies line number to print page number.

\NP
Do not pause at top of page.

\P
Perform break (\B) and start next line as a paragraph.

\PCn
Identical to \PG command. Maintained for historical purposes.

\PE
Stop printing of page numbers.

\PGn(0)
If n<0 eject page only if less than |n| text lines remain on current page.
If n absent or equal to 0 perform break and eject page.
If n>0 perform break and eject page if less than n text lines remain on current page.

\PNn(1,0)
Set current page number to n.

\PO
Print page number one (normally suppressed).

\PP[n1(0,5)][,n2(0,1)]
Set paragraph parameters. n1 is number of indent spaces. n2 is number of blank lines to preceed paragraph.

\PS (initial)
Start printing of page numbers.

\RN
Set output of page numbers to Roman numeral form.

\S[n(1)]
Perform break (\B) and space n blank lines. Blank lines are not spaced across page boundaries.

\SF[s(,.)]
Set tab fill characters to string s.

\SP[n(1,1)]
Set number (n-1) of blank lines ("spacing") to be inserted between output lines.

\T[n1(1,6)][,n2(1,11)][,n3(1,16)].....
Set tab stops to n1, n2, n3, ... n1<n2<n3<..... .

\TA[n(,1)]
\TB[n(,1)]
\TC[n(,1)]
Start output or control of respective A, B or C title lines.
If n<0 change printing status of title.
If n=0 suspend printing of title and delete.
If >0 n is line number of first line of title and following text will be output as title. If n absent previous n setting employed and title text follows.

\TO
If at top of page force printing of titles.

Error messages

Whenever error conditions are encountered in the source text WOLF outputs an error message and suspends operation. The form of the error message is:

WOLF>error number/input file data/output file data

The error numbers and their meanings are listed below. The input file data consists of the current input file name and line number of the source text containing the error. Output file data consists of the output file name and the system page number (not user page number) and next output line number.

The first 17 error numbers are reserved for host computer system errors and the user must refer to appendix A for further information. The name in parentheses following the error numbers listed identifies the program module generating the error and is meant for program maintenance personnel. If "pathologic" errors (program checks which should not occur) are encountered see personel involved with program maintenance.

-18 (IINIT) null control line.

-19 (IINIT) improper number of lines per page.

-20 (IOPNI) null or otherwise invalid name specified for new input file in the \FI command word.

-21 (IGETC) end of file. For system use. Not seen by user.

-22 (IWORD) input word too large to be handled by current version of WOLF.

-23 (IWORD) attempted to backspace beyond beginning of word.

-24 (ILINE) too many words in one line for current version of WOLF to process.

-25 (ILINE) input line cannot fit between margins. Too many characters in one word (fill mode) or line too long (no-fill mode).

-26 (ILINE) attempted to indent paragraph to the left of left margin.

-27 (ICMDA) invalid margin values. Out of range.

-28 (IPARS) pathologic error.

-29 (IINIT) improper text offset values in RUN parameter list.

-30 (IINIT) improper maximum columns specified in RUN parameter list.

-31 (IFIGC) overflow of figure request list.

-32 (IFTHC) overflow of number of lines allowed in titles or footnotes.

-33 (IPAGP) Roman number too large (>3999).

-34 (IPAGP) error in positioning line number. Check \NC n1 parameter.

-35 (IPARC) exceeded the number of command arguments allowed.

-36 (IPARC) parsing pathologic error.

-37 (IPARC) pathologic error.

-38 (IPARC) invalid command.

-39 (IPARC) number argument too large.

-40 (ICMDA) alphanumeric argument occurred when numeric argument required.

-41 (ICMDA) non-ascending tab stop positions.

-42 (ICMDA) numeric argument occurred when alphanumeric argument required.

-43 (ICMDA) null character invalid for literal editing character or command prefix character.

-44 (ICMDA) specified character not allowed as editing character or command prefix.

-45 (ICMDA) pathologic error. Invalid function number.

-46 (ICMDA) command word not allowed in title or footnote text.

-47 (IFILE) invalid or null field specified for \FI argument.

-48 (IFILE) input files nested too deep.

-49 (IFTHC) too many lines in title (>5).

-50 (IFTHC) too many lines in footnotes.

-51 (ITIFT) footnote or title request while processing other title or footnote.

-52 (ITIFT) end footnote/title (\EN) command word while not in footnote or title processing mode.

-53 (IFTLP) pathologic error in footnote printing.

Technical Description

General

Program WOLF is written for the Hewlett-Packard 2100 series computer with RTE-II or III operating system. It requires 11 k-words of user memory, approximately 18 k-words of system disc for working storage in addition to normal system overhead requirements. It should be noted that current output capability is limited to terminals or devices capable of properly processing ASCII backspace and formfeed control characters. In addition to normal RTE (HP, 1977D), FMP (HP, 1977A) and relocatable library (HP, 1977C) subroutine calls it employs the Decimal String Arithmetic package (HP, 1977B). In the following description WOLF design philosophy, general flow description and overlay-linking factors will be covered. A magnetic tape supplied with this documentation contains copies of all WOLF source code, and other data is described in appendix E.

Design philosophy

The WOLF program is a high modular set of subprograms which are nearly all coded in FORTRAN IV and where many have no machine dependent coding nor inclusion of non-standard FORTRAN features except as noted below. In a few cases, most notably I/O, machine dependent code was necessary. Obviously, this provides for relative ease in transportation of the program to other computer system environments and generally facilitates modifications. A reasonable effort was made to "comment" the code and, along with this description, an experienced programmer should be able to modify the system after some study.

The only known non-standard FORTRAN exception was the short form of the arithmetic IF statement. For example,

```
IF (I) 10,20
```

is identical to the proper form:

```
IF (I) 10,20,20
```

Although COMMON was used for storage of most working variables, many of the subprograms have all data passed as arguments including many variables employed only in the routine itself. To minimize argument lists these local variables are often stored in arrays with "PCA" suffixes and "loaded" into local arrays equivalenced with local simple variable names. Including COMMON declarations in the modules would have improved efficiency but since the final contents of COMMON was not known until the final stages, continued updating of the debugged modules would have, to say the least, impeded the development. In most cases, the subprograms are serially reusable and, consequently, can be freely employed in overlay segments.

It is recognized that any system of such general usage interest as WOLF will never be complete and even at this date certain changes and improvements are planned. Despite shortcomings, it was deemed more

important to make it available for wider distribution than try to hold off until a "perfectly polished" version was coded. Hopefully, the design philosophy employed will allow WOLF to easily grow and change.

Basic Control Flow

Because it is not practical to describe all details of a program as complex as WOLF the following description is an overall view of the program. For specific details, the reader can refer to the source code listings in appendix B. When refering to specific statements in the program listings, reference is always made to the FORTRAN statement number plus (if necessary) an offset to the statement discussed. For example, "statement 500+4" means the fourth line following the FORTRAN statement number 500.

It should also be noted that because of the overlaying method used (described later) several modules referred to in the main segment have the first letter changed from I to J. For example, IPAGC in MAINC is referred to as JPAGC.

The main program module, WOLF, merely calls IINIT for initialization, MAINC for basic operation and IINIT again for closing up the program. Note that the initial development of WOLF preceeded availability of FORTRAN BLOCK DATA segment so that all common initialization was performed by assignment statements in IINIT. It should be also noted that the HP system effectively clears memory for all variables so that a zero initial value can be assumed if not otherwise specified. IINIT also retrieves the remainder of the program execution line, opens initial input file, output file and sets optional run time variables. Comments at the beginning of IINIT also gives one line descriptions of most COMMON variables.

Routine MAINC provides basic control of WOLF operation. For source text uninterupted by command words, MAINC principally loops in the segment from statements 190 to 390, calling IPARS, ILINO and either IPAGC or IFTHC in sequence. When commands are encountered by IPARS, flow interupts and proceeds to statement 400 where IPARC interprets the command for function and arguments. MAINC then either processes some of the simpler commands locally, statements 410 through 6091, or calls additional command processing routines at statements 700 to 734. After successful processing of a command, control goes back to IPARS. When errors are encountered statements 800 on are associated with message monitoring, cleanup and return to WOLF.

Routine IPARS is responsible for retrieving, character by character, the input text data stream by the calls to routine IGETC and determining, by means of a decision table, the action to be performed by the system. The decision table is a two dimensional array (although coded as one dimensional) with the columns representing input character type and the rows representing phases of processing. Associated with each particular array position are two factors: 1) the operation to be performed on the current character and 2) the next phase or row to be employed when the next character is processed. Arrays IACTON and IPHASA represent these respective decision tables in IPARS and their DATA

statements show the operations and phases. The comments in IPARS also define the meaning of each row and column. These data statements represent the syntax of control of WOLF operation.

The basic sequence of IPARS functions can be summarized as follows: 1) Secure next input character and its type code (see comments in IGETC) from IGETC at statement 500 (Note: IGETC ignores and discards all non-printable ASCII characters), 2) decode next action or sequence of actions to be performed upon current phase and character type and 3) perform these actions as defined in code contained in statements 5012 through 519 and return for next character at step 1. The resultant operation (ignoring command word traps and other control) is to build a word by stacking printable characters with calls to IWORD. When the end of word is sensed the stacked characters are transferred to a word stack by calls to ILINE. ILINE keeps track of the total number of print positions required by the input word stream and will notify IPARS (IER return from ILINE >0) when the line is full (fill mode) which in turn causes an exit from IPARS with indication that a line is ready for printing. Note that when the full line condition exists there is a pending word in the character stack which must be placed in the new line upon reentry into IPARS. Of course, in no-fill mode the completion of the line is determined by the end-of-line condition returned by IGETC.

Returning to MAINC we can see that when IPARS returns with completed line and that after checking spacing and centering options this stack of words is now processed by ILINO at statement 290+1. ILINO takes the words in the word stack, performs required centering or justification operations and transfers them to the final output line, in "packed" form, ready for printing. MAINC then calls either IPAGC (normal text) or IFTHC (footnotes or titles) with line spacing instructions and with the final output line at statements 300 through 370.

Unfortunately, IPAGC is a complex routine for its size, but it basically controls all page related actions of WOLF. Its actions are best summarized by the page operations description of the users manual section. The page numbering routines, IPAGP, and output routine, IOUTP, are fairly self explanatory. Title and footnote operation will be covered later.

Returning to IPARS we will trace the command word control operation of the system. IPARS traps on the proper occurrance of the command word prefix character and returns to MAINC. In MAINC control is transferred to IPARC at label 400+2 to interpret the following characters of the command word. IPARC also employes a decision table for interpreting the input character stream. After IPARC determines the command word function number by calling ICMDH and stacks the command word arguments along with their type (null, integer or alpha) and values, control is returned to MAINC. Most of the remaining control action is fairly straight forward tracing of the path created by the command word number.

Footnotes and titles are handled in nearly the identical method as normal text and use the same control areas of COMMON, consequently reducing the amount of program and data storage memory requirements.

Because of this shared COMMON area, the basic text control variables (COMMON variables ITRANS through LFILLT) are swapped out to temporary system disc when these changes of text processing are encountered. It is this swapping which explains the mutual independence of footnotes, titles and main text control discussed in the users manual section.

The major difference in footnote-title processing is that MAINC calls IFTHC instead of IPAGC when the output line is ready for printing. For titles, IFTHC then simply adds the spacing or lines to temporary disc storage to be recalled and printed when IPAGC calls ITTLP at the beginning of the output page. Footnote lines are also stored on disc but because of the appending problem of footnotes a "first in, first out" type of stack system is kept to define their location. IPAGC calls IFTLP to first inform IPAGC of the number of footnote lines required and then later to print the lines at the bottom of the page.

Although several WOLF modules are not discussed here, their usage is primarily in support of the aforementioned routines. The reader should have no difficulty in understanding their usages and function.

A peculiarity in the code should also be mentioned. Because of the HP2100's lack of an integer subtraction machine function (resulting in added instructions to compliment before addition) and the failure of the FORTRAN compiler to compensate by proper handling of negative constants (always treated as positive values) nearly all negative constants are initialized by data statements so that addition can be directly performed. For example, the normal statement

```
NN-1
```

will be coded as:

```
DATA N1/-1/
 ........
N=N+N1
```

Overlaying and Linking

To further conserve memory WOLF is divided into 5 overlay segments. Because of problems associated with utilizing Hewlett-Packard's method of overlaying and the resultant necessity of considerable non-standard code in the FORTRAN programs and lack of calling parameter list capability, a system of linking was devised. This system works as follows. First the names of the modules in the overlay segments referred to in the main segment are changed (in this case, substituting first letter J for I). The overlay map module, $$WLF, resolves these names with a jump to routine $.OVL and defines for each entry the overlay segment name and offset to the desired module address in the overlay segment. $.OVL then performs overlaying, if necessary, and properly structures the call to the actual routine desired by referring to the type 5 directory programs (WLF.0, WLF.1,... etc.) at the head of each overlay segment. Although this system was developed for WOLF it is

quite flexible and is useful for other overlay applications. The final load map for this system is listed in appendix C.

References

HP, 1977A, Batch spool monitor reference manual: Hewlett-Packard Co., pp. 3-1 - 3-56.

HP, 1977B, Real-time executive III software system program and operating manual: Hewlett-Packard Co., pp. 3-1 - 3-42.

HP, 1977C, Decimal string arithmetic routines: Hewlett-Packard Co., pp. 2-3 - 2-10.

HP, 1977D, DOS/RTE relocatable library reference manual: Hewlett-Packard Co., p. 3-15.

Appendix A

This description relates to details of WOLF execution on an HP21 series mini-computer with RTE-II or III executive system.

To run WOLF:

```
RUN,WOLF,input file,output file[,n1(66)[,n2(0)[,n3(72)]]]
```

where the input and output file can be either logical unit numbers or FMP "namr" file names. Both file names must be specified. The same file name convention, when enclosed with literal editing characters, applies to the argument value of the \FI command word. Optional parameters n1, n2 and n3 (default values in parenthesis) control, respectively, the maximum lines per page, right shift and maximum number of columns or print positions in the output line. The right shift parameter allows adding a left margin to the output text by shifting all output n2 columns to the right.

The input file can be either the users terminal or the source text file. The former case is usually employed when the user desires to make temporary control changes as shown in this example:

```
:RUN,WOLF,52,52,,10
\SP2 \FI*LETTER::300* \EF
```

where the spacing is changed to double. Note that if the \EF is not used the program goes back to the terminal and expects more input. If the text was to be processed as is, then:

```
:RUN,WOLF,LETTER::300,52,,10
```

In both cases the output was to logical unit 52 and the fourth argument (third and fifth arguments are omitted) indicated the output was to be shifted to the right 10 spaces.

Output of WOLF can only be directed to disc, magnetic tape and the Anderson-Jacobson terminals if complete capability of program is required. Underscoring, backspacing and page control for line printer and HP2640 terminals are not supported in WOLF although these devices can be used to perform quick copies of most text.

Error message numbers -1 to -17 are those values returned by the FMP system. See Batch Spool Monitor manual for description.

Appendix B

Program module summary

This appendix contains a summary listing of all modules unique to the WOLF program. Unless otherwise noted all modules are written in FORTRAN IV.

Module name		Summary
Main program -	WOLF	
Subprogram -	MAINC	principle control section
	IPARS	input scanner
	IGETC	input character and translate
	IWORD	word stack control
	ILINE	line building
	ILINQ	restructure line into ASCII string
	IDSWP	data overlay controller
	IOUTP	output device control
	IFIGC	figure stack control
	IINIT	initiallization
	IOPEN	input-output file control
	IPAGC	page control
	IPAUS	top of page pause
	IPAGP	page numbering format and output
	ROMAN	roman numeral generator
	DECIM	arabic numeral generator
	IPARC	command scanner
	ICMDH	command lookup
	ITTLP	title line output
	IFTLP	footnote output
	IFTHC	figure-footnote input
	IALPX	ASCII command argument return
	IERRM	error message handler
	CFILE	decode file data
	CMOVE	special string transfer
	ICMDA	command control
	ITIFT	title-footnote control
	INUMX	numeric command argument return
	IFILE	input file stacker
	$.OVY	overlay control (Assembler)
Overlay directories -	$$WLF, WLF.0, WLF.1, WLF.2, WLF.3, WLF.4 (Assembler)	

Appendix C

LOADR Map

The following listing of the map produced by the RTE loader is provided as a check of the execution of the WOLFL transfer file contained in the program tape. Of course module address locations will vary depending upon system's first available user page.

```
COM     36002 43237
WOLF    43240 43275  22682-10993 REV. A 780502
$$WLF   43276 43354  22682-10993 REV. A 780502
MAINC   43355 44300  22682-10993 REV. A 780502
IDSWP   44301 44410  22682-10993 REV. A 780502
ILINO   44411 45254  22682-10993 REV. A 780502
IPARS   45255 47262  22682-10993 REV. A 780619
IOUTP   47263 47557  22682-10993 REV. A 780502
IWORD   47560 50711  22682-10993 REV. A 780502
ILINE   50712 52354  22682-10993 REV. A 780619
IGETC   52355 52704  22682-10993 REV. A 780502
IFIGC   52705 53150  22682-10993 REV. A 780502
$.OVL   53151 53240  22682-10993 REV. A 780502

READF   53241 53776  92002-16006 760702
REIO    53777 54101  92001-16005 741120
P.PAS   54102 54130  92002-16006 740801
CLRIO   54131 54137  750701  24998-16001
MOD     54140 54167    751101  24998-16001
R/W$    54170 54323  92002-16006  740801
RW$UB   54324 54575  92002-16006 750422
RWND$   54576 54706  92002-16006  740801
SGET    54707 54736
SPUT    54737 54771
SFILL   54772 55017


WLF.0   55020 55022  22682-10993 REV. A 780502
IINIT   55023 56613  22682-10993 REV. A 780502
IOPEN   56614 57462  22682-10993 REV. A 780502

OPEN    57463 57650  92002-16006 741205
CLOSE   57651 57757  92002-16006  740801
CREAT   57760 60235  92002-16006  741022
NAMR    60236 60532   750701  24998-16001
NAM..   60533 60627  92002-16006 740801
IAND    60630 60637   750701  24998-16001
IGET    60640 60646   750701  24998-16001
$OPEN   60647 61055  92002-16006   740801
RMPAR   61056 61113  770812  24998-16001


WLF.1   55020 55023  22682-10993 REV. A 780502
IPAGC   55024 55403  22682-10993 REV. A 780502
IPAUS   55404 55443  22682-10993 REV. A 780502
```

```
IPAGP   55444 56225   22682-10993 REV. A 780502
ROMAN   56226 56317   22682-10993 REV. A 780502
DECIM   56320 56460   22682-10993 REV. A 780502
IPARC   56461 57467   22682-10993 REV. A 780502
ICMDH   57470 60113   22682-10993 REV. A 780502
ITTLP   60114 60302   22682-10993 REV. A 780502
IFTLP   60303 61146   22682-10993 REV. A 780502

IAND    61147 61156    750701   24998-16001
SMOVE   61157 61221


WLF.2   55020 55023   22682-10993 REV. A 780502
IFTHC   55024 55337   22682-10993 REV. A 780502
IFILE   55340 55612   22682-10993 REV. A 780502
IOPEN   55613 56461   22682-10993 REV. A 780502
IALPX   56462 56551   22682-10993 REV. A 780502

OPEN    56552 56737   92002-16006 741205
CLOSE   56740 57046   92002-16006  740801
CREAT   57047 57324   92002-16006  741022
NAMR    57325 57621    750701   24998-16001
NAM..   57622 57716   92002-16006 740801
IAND    57717 57726    750701   24998-16001
IGET    57727 57735    750701   24998-16001
$OPEN   57736 60144   92002-16006   740801
RMPAR   60145 60202   770812   24998-16001


WLF.3   55020 55022   22682-10993 REV. A 780502
IERRM   55023 55433   22682-10993 REV. A 780502
CFILE   55434 55542   22682-10993 REV. A 780502
CMOVE   55543 55614   22682-10993 REV. A 780502
DECIM   55615 55755   22682-10993 REV. A 780502


WLF.4   55020 55023   22682-10993 REV. A 780502
ICMDA   55024 56775   22682-10993 REV. A 780502
ITIFT   56776 57326   22682-10993 REV. A 780502
INUMX   57327 57435   22682-10993 REV. A 780502
IALPX   57436 57525   22682-10993 REV. A 780502

SMOVE   57526 57570


11 PAGES REQUIRED
 \LOADR:WOLF    READY

 \LOADR:$END
```

Appendix D

Program HASHT

Since hash table lookup algorithm was employed in WOLF routine ICMDH it was necessary to write a small program, HASHT, to create the required table contents. Operation is fairly simple: a list of command identifiers and their function number preceeded by the number of entries in the list (see current list following program listing) is input on the first logical unit (lul) specified at run time by:

```
RUN,HASHT,lul,lu2,lu3
```

and where lu2 specifies logical unit of DATA statements and lu3 specifies collision monitoring logical unit number. The output on lu2 must be edited and /'s placed at the end of each array data statement and the file merged into ICMDH.

When adding commands the user must be careful not to exceed the dimension, ITMAX in HASHT, of the ICMDH arrays and if the dimensions must be changed, the new value must be a prime number. The hashing equation in HASHT and ICMDH obviously must be identical so that if changes are desired care must be taken to ensure proper operation of ICMDH. Monitoring of lookup collisions allows testing of the quality of the hashing equation. Although collisions will occur, they should be kept to a minimum.

```
FTN4
C
C  WOLF - AUTOMATIC TYPING SYSTEM,  1977
C
C  USDI, GEOLOGICAL SURVEY
C  GERALD I. EVENDEN
C
C
C  SUPPLIMENTARY PROGRAM TO GENERATE LOOKUP TABLE FOR
C  SUBROUTINE ICMDH.
C
      PROGRAM HASHT,3
C
      DIMENSION IP(5)
      EQUIVALENCE (IP(1),LU),(IP(2),LIST),(IP(3),MON)
C
      DIMENSION IC1T(100),IC2T(100),ICOMN(100)
C
      DATA IC1T,IC2T,ICOMN/300*-1/
C
C  INSERT TABLE SIZE
C  MUST BE PRIME NUMBER
      DATA ITMAX/59/
C
C
      CALL RMPAR(IP)
C
      READ(LU,9900) NENT
9900  FORMAT(I5)
      DO 200 I=1,NENT
        READ(LU,9901) IW,ICOM
9901    FORMAT(4X,A2,I5)
        CALL SGET(IW,1,IC1)
        CALL SGET(IW,2,IC2)
C
C  HASH ALGORITM
C  RANDOMIZE..
          IHT=MOD((IC1-64)*(IC2-31),ITMAX)
          IF (IHT.LE.0) IHT=1
          IHS=IHT
100       IH=MOD(IHS,ITMAX)+1
          FS=FS+1.
          IF (IC1T(IH)) 110,120,120
C
C  NO COLLISION ADD TO TABLE
110       IC1T(IH)=IC1
          IC2T(IH)=IC2
          ICOMN(IH)=ICOM
          GO TO 200
C
C  CHECK IF DOUBLE ENTRY
120       IF (IC1-IC1T(IH)) 140,130,140
130       IF (IC2-IC2T(IH)) 140,800,140
C
```

```
C  NO, RE-RANDOMIZE
140         CONTINUE
            K=IHS
            IHS=IAND(IHS+IHT,77777B)
            WRITE(MON,9905) IW,ICOM,K,IHS,IHT,IH
9905        FORMAT(11H COLLISION:,4X,A2,I5,4I6)
            GO TO 100
C
200      CONTINUE
       FS=FS/NENT
       WRITE(MON,9940) FS
9940   FORMAT("AVERAGE COLLISION RATE:",F10.3)
C
C  PRINT RESULTS
       WRITE(LIST,9909)
9909   FORMAT(16H      DATA IC1T/)
       WRITE(LIST,9910) (IC1T(I),I=1,ITMAX)
9910   FORMAT((6H     *,10(I5,1H,)))
       WRITE(LIST,9911)
9911   FORMAT(16H      DATA IC2T/)
       WRITE(LIST,9910) (IC2T(I),I=1,ITMAX)
       WRITE(LIST,9912)
9912   FORMAT(17H      DATA ICOMN/)
       WRITE(LIST,9910) (ICOMN(I),I=1,ITMAX)
C
       GO TO 900
C
800    WRITE(MON,9920) IW,ICOM
9920   FORMAT(13HDOUBLE ENTRY:,4X,A2,I5)
C
900    STOP
C
       END
       END$
```

The following is a listing of the input to lul which contains the current command words of the WOLF system.

```
46
 B      1
 P      2
 C      3
 CS     4
 CE     5
 FS     6
 FR     7
 JS     8
 JR     9
 PG    10
 PC    10          HISTORICAL
 S     12
 SP    13
 PP    14
 AP    15
 AR    16
 MR    17
 T     18
 CP    19
 CT    20
 CF    21
 CU    22
 CC    23
 CB    24
 CL    25
 PN    26
 PS    27
 PE    28
 NC    29
 RN    30
 AN    31
 I     32
 SF    33
 PO    34
 FI    40
 EF    41
 TA    42
 TB    43
 TC    44
 FT    45
 EN    46
 TO    35
 NP    36
 FG    37
 CH    38
 CK    39
```

Appendix E

WOLF Program Tape

The 9 track, 800 bpi tape supplied with this documentation contains source and binary files of all software unique to the WOLF program. In addition, FMGR transfer files are also recorded to facilitate copying the tape to disc and program loading. The following listing of the disc transfer file (first file on tape) also serves as a directory of tape contents. In all cases, Hewlett-Packard file naming conventions are employed.

```
:** TRANFER FILE FOR LOADING TAPE COPY OF WOLF SYSTEM TO DISC
:CN,1G,RW     MAKE SURE WE'RE AT BOT
:CN,1G,FF     SKIP THIS FILE
:ST,1G,&WOLF ::2G::-1,AS      MAIN SEGMENT
:ST,1G,&MAINC::2G::-1,AS      THE FOLLOWING ARE WOLF SUBPROGRAMS
:ST,1G,&IPARS::2G::-1
:ST,1G,&IGETC::2G::-1
:ST,1G,&IWORD::2G::-1
:ST,1G,&ILINE::2G::-1
:ST,1G,&ILINO::2G::-1
:ST,1G,&IDSWP::2G::-1
:ST,1G,&IOUTP::2G::-1
:ST,1G,&$.OVY::2G::-1
:ST,1G,&$$WLF::2G::-1
:ST,1G,&IFIGC::2G::-1
:ST,1G,&WLF.0::2G::-1
:ST,1G,&IINIT::2G::-1
:ST,1G,&IOPEN::2G::-1
:ST,1G,&WLF.1::2G::-1
:ST,1G,&IPAGC::2G::-1
:ST,1G,&IPAUS::2G::-1
:ST,1G,&IPAGP::2G::-1
:ST,1G,&ROMAN::2G::-1
:ST,1G,&DECIM::2G::-1
:ST,1G,&IPARC::2G::-1
:ST,1G,&ICMDH::2G::-1
:ST,1G,&ITTLP::2G::-1
:ST,1G,&IFTLP::2G::-1
:ST,1G,&WLF.2::2G::-1
:ST,1G,&IFTHC::2G::-1
:ST,1G,&IFILE::2G::-1
:ST,1G,&IALPX::2G::-1
:ST,1G,&WLF.3::2G::-1
:ST,1G,&IERRM::2G::-1
:ST,1G,&CFILE::2G::-1
:ST,1G,&CMOVE::2G::-1
:ST,1G,&WLF.4::2G::-1
:ST,1G,&ICMDA::2G::-1
:ST,1G,&ITIFT::2G::-1
:ST,1G,&INUMX::2G::-1
:ST,1G,%WOLFP::2G::-1,BR  RELOCATABLE READY FOR LOADR
:ST,1G,*WOLFL::2G::-1,AS   TRANSFER FILE TO LOAD WOLF PROGRAM
```

```
:**                          FROM INDIVIDUAL BINARIES
:ST,1G,"APPXA::2G::-1,AS     SOURCE TEXT FOR APPENDIX A
:ST,1G,&HASHT::2G::-1,AS     HASH TABLE PROGRAM
:ST,1G,HASHT ::2G::-1,AS     CURRENT COMMANDRNO. LIST FOR PROG. HASHT
:CN,1G,RW
::
```