

MANU/3000 FORMATTING SYSTEM

Documentation



Information Networks Division

Location Code: 66-????

Project Number: ????-????

July 30, 1984

CAUTION

MANU/3000 is an *unsupported* application; it is not a product. It was developed internally to meet a specific need. You can use the MANU/3000 Formatting System and the HP2680 laser printer to produce camera-ready documents. This document is a demonstration of the results you can expect.

MANU/3000 uses TDP and the HP2680 laser printer. The latest version (NOV0183) was developed with:

TDP version HP36578A.03.00

MPE version HP32033C.A1.02

It has been used successfully on other versions of TDP and MPE, but may not produce identical results on all systems.

Again, MANU/3000 is *unsupported* and is intended for *internal use only*.

* HP Confidential *

Copyright © 1984 HEWLETT-PACKARD COMPANY

HP 3000 Computer Systems

**MANU/3000
FORMATTING SYSTEM**

Reference Manual



19420 HOMESTEAD ROAD, CUPERTINO, CA. 95014

Part No. 30000-?????
E1083

Printed in U.S.A. 10/83

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.

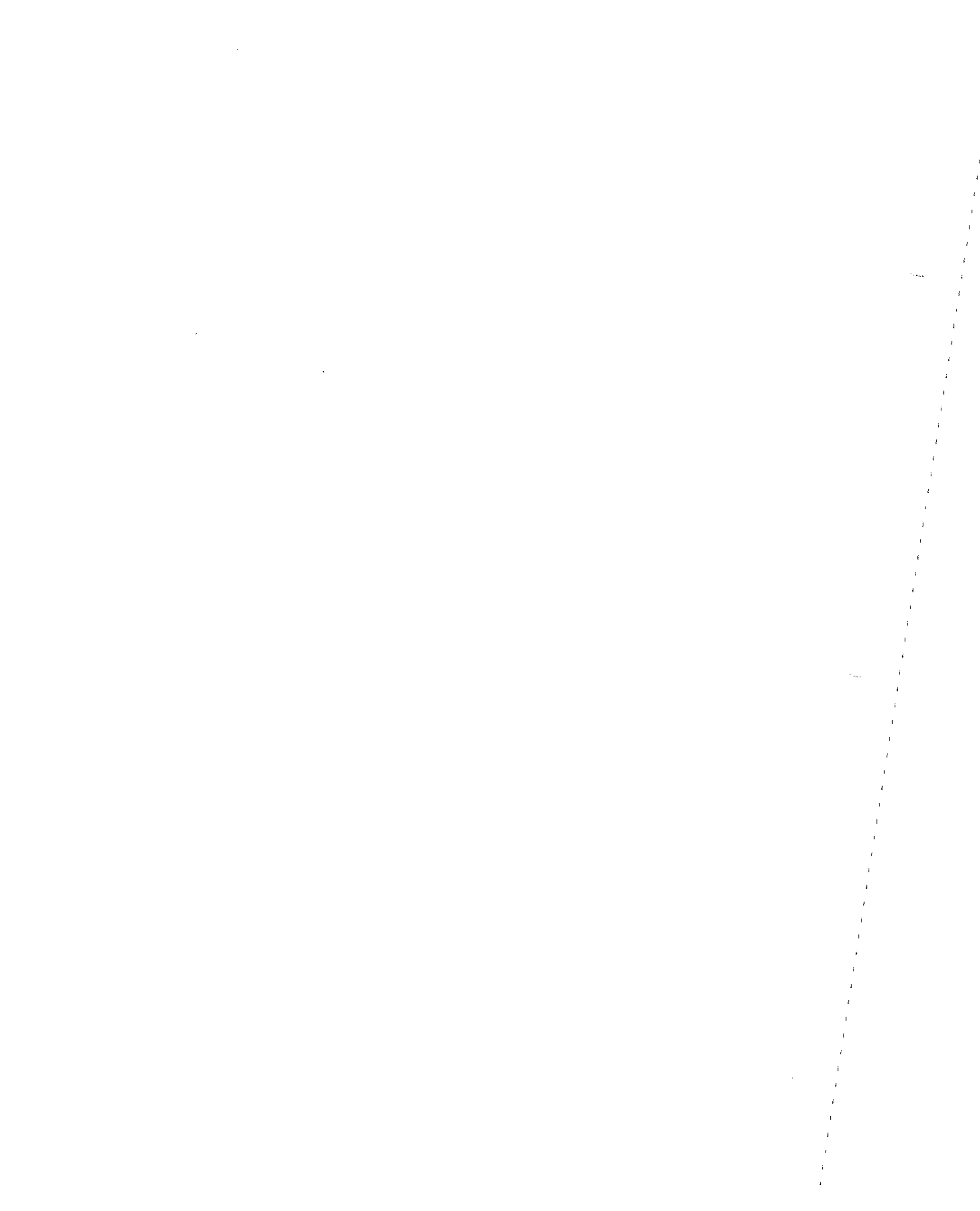
This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Hewlett-Packard Company.

PRINTING HISTORY

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The dates on the title page change only when a new edition or a new update is published. No information is incorporated into a reprinting unless it appears as a prior update; the edition does not change when an update is incorporated.

The software code printed alongside the date indicates the version level of the software product at the time the manual or update was issued. Many product updates and fixes do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one to one correspondence between product updates and manual updates.

First Edition.....	Mar 1983.....	?????v. uu. ff
Second Edition.....	May 1983.....	?????v. uu. ff
Third Edition.....	Oct 1983.....	?????v. uu. ff



LIST OF EFFECTIVE PAGES

The List of Effective Pages gives the date of the most recent version of each page in the manual. To verify that your manual contains the most current information, check the dates printed at the bottom of each page with those listed below. The date on the bottom of each page reflects the edition or subsequent update in which that page was printed.

Effective Pages	Date
all.....	Oct 1983



PREFACE

This manual describes the MANU/3000 Formatting System which allows you to produce manuals or other similar documents on the HP2680 laser printer using TDP/3000's formatter (SCRIBE). The MANU/3000 Formatting System is designed to make your life easier by performing typical and/or difficult TDP/HP2680 formatting for you.

MANU/3000 allows you to produce either 8.5 x 11 inch documents or 9 x 8.5 inch documents. Typically, reference manuals are 8.5 x 11 inches, while user's guides are 9 x 8.5 inches. You can easily test a section of your manual in either format simply by changing one parameter.

The MANU account contains the files that do the main formatting work. This account is set up such that everyone has read access to all the files. You should work in an account other than MANU; you will still be able to access the MANU files.

If you are using TDP/3000's editor, you can use the use files in the USE group. If you are using HPSLATE as your editor, you can insert the macro definitions and include statements manually. The IN and TABLE groups of the MANU account contain the include files.

In addition to this manual, you need to obtain a copy of the latest *TDP/3000 Reference Manual* (36578-90001) and refer to chapter 4, which documents the TDP formatter commands. The TDP manual is provided with the TDP product itself and will usually be on your system; "final" a copy or print a copy from a spoolfile tape of the formatted manual.

Chapter 3 of the *TDP/3000 Reference Manual* covers the TDP editor commands and will be useful if you are using TDP as your editor. If you are using HPSLATE as your editor, obtain a copy of the *HPSLATE/3000 Reference Guide* (36576-90001).



CONVENTIONS USED IN THIS MANUAL

NOTATION	DESCRIPTION
nonitalics	Words in syntax statements which are not in italics must be entered exactly as shown. Punctuation characters other than brackets, braces and ellipses must also be entered exactly as shown. For example: EXIT;
<i>italics</i>	Words in syntax statements which are in italics denote a parameter which must be replaced by a user-supplied variable. For example: CLOSE <i>filename</i>
[]	An element inside brackets in a syntax statement is optional. Several elements stacked inside brackets means the user may select any one or none of these elements. For example: $\left[\begin{array}{c} A \\ B \end{array} \right]$ User <i>may</i> select A or B or neither.
{ }	When several elements are stacked within braces in a syntax statement, the user must select one of those elements. For example: $\left\{ \begin{array}{c} A \\ B \\ C \end{array} \right\}$ User <i>must</i> select A or B or C.
...	A horizontal ellipsis in a syntax statement indicates that a previous element may be repeated. For example: [, <i>itemname</i>]...;
	In addition, vertical and horizontal ellipses may be used in examples to indicate that portions of the example have been omitted.
█	A shaded delimiter preceding a parameter in a syntax statement indicates that the delimiter <i>must</i> be supplied whenever (a) that parameter is included or (b) that parameter is omitted and any <i>other</i> parameter which follows is included. For example: <i>itema</i> [█ <i>itemb</i>][, <i>itemc</i>]

means that the following are allowed:

```
itema  
itema, itemb  
itema, itemb, itemc  
itema, , itemc
```

CONVENTIONS (continued)

Δ When necessary for clarity, the symbol Δ may be used in a syntax statement to indicate a required blank or an exact number of blanks. For example:

```
SET[(modifier)] $\Delta$ (variable);
```

underlining When necessary for clarity in an example, user input may be underlined. For example:


```
NEW NAME? ALPHA
```

In addition, brackets, braces or ellipses appearing in syntax or format statements which must be entered as shown will be underlined. For example:

```
LET var[[subscript]] = value
```

shading Shading represents inverse video on the terminal's screen. In addition, it is used to emphasize key portions of an example.



The symbol  may be used to indicate a key on the terminal's keyboard. For example, **RETURN** indicates the carriage return key.

CONTROL*char* Control characters are indicated by **CONTROL** followed by the character. For example, **CONTROL**Y means the user presses the control key and the character Y simultaneously.

CONTENTS

Section 1

INSTALLATION

Storing MANU.....	1-2
Restoring MANU.....	1-2
Printing This Manual.....	1-4

Section 2

GETTING STARTED

Include Files.....	2-2
Use Files.....	2-5
Creating and Printing Files.....	2-7
Single Files.....	2-7
The Core File.....	2-9
Batch Printing.....	2-11
Creating Documents.....	2-13

Section 3

MORE ABOUT FORMATTING

Using Multiple Fonts.....	3-2
Fonts and Fontids.....	3-2
Font Pairs.....	3-3
Conversion Fonts.....	3-4
Underlining.....	3-5
Using the Blank Character.....	3-6
Changing Logical Pages.....	3-7
Using Macros.....	3-8
Lengths and Margins.....	3-9
Output Margins.....	3-9
Input Margins.....	3-9
Understanding Proportional Fonts.....	3-11
Lining-Up Text.....	3-11
Understanding EMs.....	3-12
Using Multiple Columns.....	3-13
Rows.....	3-13
Formatting Modes.....	3-14
Indented Columns.....	3-15
Margins.....	3-15
Estimating Column Widths.....	3-16
EMs with Monospaced Fonts.....	3-16
EMs with RWIDTH.....	3-18

CONTENTS (continued)

Adjusting the Margins.....	3-19
Using INLFT and INRHT.....	3-19
Using INDENT.....	3-21
Spaces with RWIDTH.....	3-21
Spaces with Monospaced Fonts.....	3-23

Section 4

SPECIFIC TASKS

Creating Syntax.....	4-2
Generating Tables.....	4-5
Creating a Table.....	4-5
Editing a Table.....	4-10
Including HPDRAW Figures.....	4-15
Using FSCREEN.TOOLS.MANU.....	4-17
The RUN Command.....	4-17
Translating Terminal Displays.....	4-18
The Begin and End Lines.....	4-18
Running FSCREEN.....	4-19
A Simple Example.....	4-20
Advanced Capabilities.....	4-22
Identifying Terminal Enhancements.....	4-22
Using Terminal Enhancements.....	4-23
Drawings.....	4-24
Syntax Diagrams.....	4-26
Converting VPLUS/3000 Forms.....	4-27
Generating a Table of Contents.....	4-30
Generating an Index.....	4-33
Using INDEX.....	4-33
Using INDEXER.TOOLS.MANU.....	4-33
Customized Formatting.....	4-37

Section 5

FORMATTING FILES

APPENDX.....	5-3
BACKCOV.....	5-5
BACKMAT.....	5-6
BLOCK.....	5-7
BOX.....	5-8
BULLETS.....	5-9
CAUTION.....	5-11
CON.....	5-12
CONVEN.....	5-14
COPYRITE.....	5-15

CONTENTS (continued)

COVER.....	5-16
COVERUPD.....	5-17
DISCUSS.....	5-18
EASYREF.....	5-19
EFFPAGES.....	5-20
EXAMPLE.....	5-22
FIGTAB.....	5-23
FORMAT.....	5-24
FRONTMAT.....	5-25
HISTORY.....	5-26
IMAGE.....	5-28
INDX.....	5-29
LABCOV.....	5-30
LEVEL1.....	5-32
LEVEL2.....	5-33
LEVEL3.....	5-34
LEVEL4.....	5-35
MANSTART.....	5-36
MEMO.....	5-38
NOTE.....	5-40
ONESTART.....	5-41
PAGEHEAD.....	5-43
PARMS.....	5-44
PREFACE.....	5-45
RCS.....	5-46
SECTION.....	5-48
SUBPARMS.....	5-50
SYNTAX.....	5-52
TAB2COL.....	5-53
TAB3COL.....	5-55
TAB4COL.....	5-57
TEXTREF.....	5-60
TITLE.....	5-61
WARNING.....	5-63

Appendix A LASER PRINTER TERMINOLOGY

Appendix B FUNCTION KEYS FOR TDP'S EDITOR

Softkeys Example.....	B-2
-----------------------	-----

CONTENTS (continued)

Appendix C FONTS AND LOGICAL PAGES

Logical Pages.....	C-1
Font Identifications.....	C-2

Appendix D GUIDELINES FOR FONT USAGE

Appendix E MACROS AND NUMBERING ARGUMENTS

Macros.....	E-2
Numbering Arguments.....	E-3

Appendix F QUICK REFERENCE

FIGURES AND TABLES

LIST OF FIGURES

Figure 4-1. Column Widths Example.....	4-7
Figure 4-2. Two Methods for Including an HPDRAW Figure.....	4-16
Figure 4-3. Control Keys for an HP262X Terminal.....	4-19
Figure 4-4. Control Keys for an HP264X Terminal.....	4-19
Figure 4-5. Error Message Example.....	4-20
Figure 4-6. The Edited Error Message.....	4-21
Figure 4-7. Sample Terminal Test Pattern.....	4-23

LIST OF TABLES

Table 2-1. MANU Formatting Functions.....	2-14
Table 4-1. Display Enhancements.....	4-24
Table C-1. Logical Pages in the MANU Environments.....	C-1
Table C-2. Font Identifications.....	C-2
Table C-3. Conversions for the LINEDRAW Font.....	C-3
Table C-4. Conversions for the PARENS Font.....	C-4
Table C-5. Conversions for the BOXDRAW Font.....	C-4
Table C-6. Conversions for the KEYS Font.....	C-5
Table C-7. Conversions for the MATH Font.....	C-6
Table D-1. Font Usage Guidelines.....	D-1
Table F-1. Use Files and Corresponding Include Files.....	F-2
Table F-2. Use Files and Corresponding Include Files for Tables.....	F-3



INSTALLATION

SECTION

1

This section describes how to obtain and install the MANU/3000 Formatting System. It also describes how to print copies of this manual.

The same instructions are in the file `README.ASAP.MANU`, provided with the MANU system.

You should first check whether you already have the latest MANU Formatting System on your computer. To do this, issue the command:

```
:LISTF @.VERSION.MANU
```

If the MANU system does not exist on your computer, you will get an error.

If the MANU system does exist on your computer, this command will report the date of the installed version in the form *mmddyy* (for example, `MAR1883` for March 18th, 1983). The *latest* MANU Formatting System will report the date:

```
NOV0183
```

An earlier date indicates an earlier version of the MANU system.

STORING MANU

To obtain the MANU/3000 Formatting System, you must first find an HP 3000 computer with the latest version of MANU. Next, log on to that computer as OPERATOR.SYS, mount a tape on the tape drive, and make a store tape of the MANU account by issuing the following commands:

```
:FILE SYSLIST;DEV=LP
:FILE MANUTAPE;DEV=TAPE
:STORE @.@.MANU;*MANUTAPE;SHOW
```

Pick up the listing at the line printer and check that all files from the MANU account were successfully stored to your tape.

RESTORING MANU

To install the MANU system from a store tape, follow these steps:

1. If the MANU account does *not* already exist on your system, get your system manager to create the account as follows:

```
:NEWACCT MANU,MGR;ACCESS=(R,X:ANY;W,A,L:AC); &
:CAP=AM,AL,GL,SF,IA,BA,ND,PH,DS;FILES=20000
```

If the account *does* already exist, log on as MGR.MANU and remove any password from the user MGR:

```
:ALTUSER MGR;PASS=RETURN
```

2. Issue the following commands to install the MANU system:

```
:FILE MANUTAPE;DEV=TAPE
:RESTORE *MANUTAPE;MANUJOB.PUB.MANU;SHOW
```

Mount the tape and reply to the tape request. After the file MANUJOB has been restored, issue the command:

```
:STREAM MANUJOB.PUB.MANU
```

The job will ask for the tape to be mounted (put the tape back on-line) and will install the new MANU system. Pick up the job listing (under MGR.MANU) and look it over to make sure all the files were restored.

3. Put a password on MGR.MANU:

```
:ALTUSER MGR;PASS=password
```

This is important! The files of the MANU account are strongly interdependent! If you modify one file, you will get unexpected results when you use other (seemingly unrelated) files. Always follow these rules:

- Do *not* let anyone modify the files in the MANU account nor add additional files to the account. If files are changed, another user on your computer will be unable to use the MANU Formatting System because it will not work as documented. In addition, other people who obtain the MANU system from your computer will then have a version which will not work as documented.
- Do *not* allow users of the MANU system to log on to the MANU account. They should be working in a different group and account.

PRINTING THIS MANUAL

After installing the MANU system, you will probably want to print out one or more copies of the documentation. This manual was written using MANU/3000 and hence is an example of the results you can expect.

To obtain a copy of this manual, "final" the file CORE.DOC.MANU to *HP2680. Because the manual is quite large (and because "fancy" HP2680 formatting takes time and patience), we strongly recommend doing this from a job.

If you like, you may copy the file DOCJOB.DOC.MANU into your own group and account; then modify the first line (the JOB command) to refer to *your* group and account and keep the file. Do *not* change the JOB command to MGR.MANU, as you will have to include the password -- which others can then read. If you want more than one copy of the MANU documentation, also change the FINAL command in the job to include the number of copies. Then stream the job, as follows:

```
:STREAM DOCJOB
```

The job may take several hours to "final", depending on the load on your system.

NOTE

This manual is not intended to teach you TDP. You should already be familiar with TDP's formatter; you should understand the basic formatter commands and be able to use them correctly. You do not, however, need to know the TDP commands that only apply when using environment files and the laser printer; this manual will teach you those you need to know.

In addition, this manual will not teach you TDP's editor, HPSLATE, nor any other text editor or word processor. You should already be familiar with the editor you are going to use. (It does not matter which product you use as your editor, as long as the file can eventually be output through TDP's formatter; refer to section 2 (Use Files) for an explanation of the one advantage to using TDP's editor.)

GETTING STARTED

SECTION

2

This section explains how the MANU/3000 Formatting System helps you produce manuals and other similar documents on the HP2680 laser printer using TDP/3000's formatter (SCRIBE). With the MANU/3000 Formatting System, you can produce either 8.5 x 11 inch manuscripts or 9 x 8.5 inch manuscripts. These manuscripts might be:

- reference manuals
- user's guides
- documents or reports
- internal documents
- memos

The MANU/3000 Formatting System is designed to make your life easier by performing typical and/or difficult TDP/HP2680 formatting for you.

INCLUDE FILES

An **include file** is a file which is inserted, at output time, into the file you are currently formatting and printing. It is called an include file because you specify where you want the file to be inserted by using the `\IN` (or *include*) formatting command. Refer to the `\IN` command in section 4 of the *TDP/3000 Reference Manual*, 36578-90001.

The MANU formatting system provides you with a set of include files in the IN group of the MANU account. These files do most of the typical and/or difficult TDP/HP2680 formatting for you. For example, to start a section of a manual, an include file is provided which invokes the correct logical page containing the section banner form, positions the title of the section and section number correctly on the logical page, puts an entry in the table of contents file, and so on. In addition, the MANU formatting system includes a set of specialized include files in the TABLE group of the MANU account. These files also take care of the details of formatting for you; in this case to produce two, three, and four column tables, complete with vertical and horizontal lines.

Include files, just like the files you produce yourself, contain text and/or TDP formatting commands. Most of the formatting commands found in an include file are the same ones you are familiar with from TDP's formatter (for example, `\NEW` to cause a page break). In addition, the `\IF` formatting command allows include files to be used by many people doing different kinds of formatting. For example, the `\IF` command can be used to check the value of a TDP macro to see whether 8.5 x 11 or 9 x 8.5 format is desired; based on the value of the macro, the `\IF` command can control which environment file is used.

To make use of the MANU include files, you simply need to be using TDP's formatter to print your manuscript. It does not matter what product you use as your editor, as long as the manuscript can eventually be output through TDP's formatter. Obviously, TDP's editor is most commonly used. HPSLATE is also quite commonly used since the interface between HPSLATE and TDP's formatter is provided with the product.

When you wish to insert an include file, you will typically have to pass some information to that file in the form of macros. You will usually insert some lines in your document which look like this:

```
\**** comment ****  
\m1="macro"  
\m2="macro"  
.  
.  
.  
\mn="macro"  
\in includefile.in.manu
```

The first line is a comment line (at least one asterisk must follow the backslash); we strongly recommend the use of comments. The next *n* lines are macro definitions, used to pass information to the include file. The last line invokes the include file with a `\IN` command. All the lines must begin with the backslash in column one. Uppercase or lowercase may be used and will produce identical results, *except* within the definitions of the macros themselves (the text enclosed within quotes).

NOTE

You should always work in your own group and account (*not* in the MANU account); you can access the MANU include files by fully qualifying the file names (*file.IN.MANU*).

For example, to produce the first level 1 head in this section of this manual, the input (or editor) file contains the following:

```

.
.
.
The MANU/3000 Formatting System is designed to make your life
easier by performing typical and/or difficult TDP/HP2680
formatting for you.
**** LEVEL1 ****
\m1="Include Files"
\m2="INCLUDE FILES"
\in level1.in.manu
An ^ftinclude file^s is a file which is inserted, at output
time, into the file you are currently formatting and printing.
.
.
.

```

In the above example, a comment line indicates that we are starting a level 1 head. By looking up LEVEL1 in section 5 (Formatting Files), we find that the first macro is the entry for the table of contents; hence we define m1 to be the level 1 head in *initial* caps. The second macro is the level 1 head itself; hence we supply it in *all* caps. Finally, \in level1.in.manu invokes the include file which causes a page break and prints the level 1 head. (We can include level1x.in.manu instead of level1.in.manu if we do not wish to force a new page.)

Note that we did *not* include in our document any commands to do page breaks, font changes, or spacing; these are all performed by the include file. In this case, when the document was "finaled" (formatted and printed through TDP's formatter), LEVEL1 caused a page break, printed the level 1 head in the correct font, specified an entry in the table of contents, and printed two blank lines after the level 1 head.

Some include files come in pairs; you should never use one without also using the other. The second include file turns off or resets formatting commands which were issued in the first include file. For example, if you use the NOTE include file to produce a note, it is important not to forget to insert the ENDNOTE include file after the text of the note (see NOTE in section 5):

```

**** NOTE ****
\in note.in.manu
This is an example of a note. You must use two
include files (NOTE and ENDNOTE) to produce the correct
results.
\in endnote.in.manu

```

Getting Started

As a general rule, you should not insert another include file within a paired set of include files. For example, you should not use BULLETS within the paired set of include files NOTE/ENDNOTE. One useful exception to this rule is that the paired sets of include files NOTE/ENDNOTE, CAUTION/ENDNOTE, WARNING/ENDNOTE, and BULLETS/ENDBULL can be used successfully within the paired set of include files PARMs and ENDPARMS in order to produce notes, cautions, warnings, or bullets within the explanation of a parameter. In addition, the paired set of include files SUBPARMS/ENDSUBP is intended for use within PARMs/ENDPARMS in order to produce subparameters within the explanation of a parameter.

You will need to refer to section 5 (Formatting Files) of this manual in order to determine the proper "syntax" (macros, include file pairs, etc.) necessary for the include files. In addition, the discussions given in section 5 will tell you exactly what is performed for you (page breaks, spacing, etc.) by the include files. Refer to appendix F (Quick Reference) for a functional list of all the MANU include files.

USE FILES

A use file is a file containing a series of TDP/3000 editor commands. It is called a use file because it is invoked by issuing the USE command in TDP's editor. Refer to USE in section 3 of the *TDP/3000 Reference Manual*, 36578-90001.

The MANU formatting system supplies you with a set of use files in the USE group of the MANU account. By using these files, you do not have to worry about the proper "syntax" needed for the include files; the use files prompt you for information and insert the macro definitions and include statements for you. *You must, however, be in TDP's editor in order to execute the use files!*

Most of the editor commands found in a use file are the same ones you are familiar with if you are using TDP's editor (for example, ADD). In addition, there are some special TDP editor commands which allow prompting for input and conditional branching within a use file. This allows use files to operate much like programs -- asking you questions and acting according to the answers you supply.

From TDP's editor, use files can be executed in response to the editor's slash (/) prompt. You may *not* be in TDP's screen or add mode. Simply type `USE usefile.USE.MANU`. You should be logged on in your own group and account (*not* the MANU account).

For example:

```
/use level1.use.manu
Place at specific line number (Y) or end of workfile (N): n
Enter level 1 heading (in Initial Caps): Include Files
Do you want this level 1 head to start on a new page? (Y/N) y
235 //
```

As in the above example, most of the MANU use files will ask whether to add material at a specific line number or at the end of the workfile. If your workfile is empty, either choice is fine. If your workfile contains text, you may wish to specify a specific line number. The use file will then prompt you for the line number. For example:

Line number to insert level 1 head after: 98

You may specify a non-existent line number or an existent line number in your workfile to add material after. After prompting for the line number, the use file continues in its usual fashion.

Many of the MANU use files place you into add mode. In the above example, we were placed into add mode to add text following the level 1 head; we decided not to add text and exited add mode by typing two slashes (//). Because some use files add material (such as a second include statement for a pair of include files) after you exit add mode, you should *always* terminate add mode by typing two slashes -- *not* by typing `(CONTROL)Y`.

It is reassuring to list your workfile and see what the use file did. Continuing with the same example, we can list a portion of our workfile to see the lines which were added by the LEVEL1 use file:

```
/l 231/last
231 \***** LEVEL1 *****
232 \m1="Include Files"
233 \M2="INCLUDE FILES"
234 \in level1.in.manu
```

Getting Started

All of the use files supplied with the MANU system start by adding a comment line similar to the one the LEVEL1 use file added on line 231 above. Next, the LEVEL1 use file prompted us for the level 1 head in initial caps (to be used for the table of contents) and added a line defining m1 to be equal to the string we entered. It also added a line defining m2 to be an upshifted copy of m1 (to be used for the level 1 head itself). Finally, the use file added an include statement to invoke the LEVEL1 include file. If we had answered no to forcing a new page, an include statement for `level1x.in.manu` rather than `level1.in.manu` would have been added instead.

If you make a mistake while executing a use file, you have three choices:

- Continue as if nothing had happened and edit the errors later, after the use file has terminated.
- If the use file has placed you in add mode, use TDP's `//m range` editor command in response to a line number prompt.
- Abort the use file with **CONTROL**Y. Note that you may have to "clean up" your workfile since any lines the use file has already added to your file will be there.

If you dislike having to type `USE usefile.USE.MANU`, you might be able to create a UDC which loads a softkey to prompt for the name of the MANU use file; you can then specify the use file name without fully qualifying it (for example, `level1` instead of `level1.use.manu`). Refer to appendix B (Function Keys for TDP's Editor) for an example of how this is done.

WARNING

Due to an interesting "feature" currently in TDP (36578A.03.00), we strongly advise against setting up a UDC of file equations for the MANU use files. Although it would be nice to be able to type `USE *usefile` instead of `USE usefile.USE.MANU`, the file equations are quite dangerous; they can cause you to unintentionally overwrite a file! [This happens because certain TDP functions "allow" (instead of "disallow") file equations when they open a file.]

Refer to section 5 (Formatting Files) for the "syntax" of the lines added to your document by each use file and for a description of the end result when your document is output and formatted through TDP's formatter. In general, the use file dialogues themselves are not documented (other than the examples shown in this section) since most of them are fairly straightforward. However, certain tasks, such as generating an index or a table, do require more detailed instructions. The use files for these tasks are documented as part of the discussion in section 4 (Specific Tasks).

CREATING AND PRINTING FILES

Before a file can be printed, the HP2680 laser printer needs to know certain information: the environment file being used, the font identifications, and so on (refer to appendix A for Laser Printer Terminology). In addition, in order for the MANU Formatting System include files to operate correctly, certain additional information needs to be defined: what the margins are set at, how page numbering is to be handled, whether this is an 8.5 x 11 or 9 x 8.5 inch document, etc. The ONESTART and MANSTART files provide all of this important information.

If your document is fairly small, you will probably keep it all in one file. In that case, you will simply use ONESTART at the beginning of your file (see Single Files, below).

If your document is a larger manuscript, you will want to use multiple files -- with each major portion or section in a separate file. Since you will probably want to print individual pieces of your manuscript for review purposes, you will still use ONESTART at the start of each of your files (see Single Files, below). Later, you will use MANSTART to build a file which "ties" all the pieces of your document together and prints your entire manuscript (see The Core File, later in this section).

Single Files

Whenever you create a file, you have to provide certain information to TDP and the HP2680 laser printer right at the *beginning* of the file. This is true whether you are creating a fairly small document which will be kept entirely in one file or whether you are creating one portion (or section) of a larger document which will be kept in multiple files. In order to "final" any single file, this information must exist right at the start of the file.

For example, the file containing this section of this manual starts with the following lines:

```
\**** ONESTART ****
\m1="manusec2.doc.manu"
\m2="yes"
\if main in onestart.in.manu
```

The description of ONESTART in section 5 helps us to understand the above example:

- 1) m1 simply documents the name of the file this document is kept in. When this file is "final", a "fold-back" page is printed containing the name of the file plus the date and time the copy was "final". It is suggested that you paginate your copies correctly (re-folding if necessary) and that you fold back this initial page to display the name, date, and time before the first page of each copy.
- 2) m2 informs TDP that this is an 8.5 x 11 inch document; hence, TDP knows to use the MANU90 environment file. If, instead, m2 is set to "no", the MANU0 environment file is used to produce a 9 x 8.5 inch document.
- 3) Finally, `\if main in onestart.in.manu` invokes the ONESTART include file *whenever this file is the main file*. By `main` we mean that this file, during this printing, is not being inserted as an include file into another file. (If this file were *not* the main file, the information TDP and the HP2680 needs would have already been provided, the `\if main` statement in this file would be false, and the ONESTART include file would not be invoked.)

Getting Started

ONESTART is all you need to be able to "final" your document to the HP2680 laser printer; if you do not specify otherwise, text will be printed in the roman font using logical page 0 (blank) of the MANU0 or MANU90 environment files and will be formatted with right justification. If you wish to change some of these things, you may do so within your document by using the include files provided with the MANU Formatting System or by using any of TDP's backslash or intraline (caret) formatting commands.

In addition to the `\if main in onestart.in.manu` statement, you may want to include two additional `\if main` statements right at the start of each file. These are:

```
\if main in numbered.in.manu
\if main in noblank.in.manu
```

The NUMBERED include file causes level heads to be numbered hierarchically (called subsection numbering). The NOBLANK include file inhibits the printing of extra pages which are otherwise printed to preserve left/right pagination. See ONESTART in section 5 for more information on these two include files.

If you are using TDP's editor, you can execute the ONESTART use file, as follows:

```
/use onestart.use.manu
Is workfile empty (Y/N)? y
Enter name of this file: manusec2.doc.manu
Is this an 8 1/2 by 11 manual? y
Do you wish to number subsections (e.g. 3.5.1)? n
Do you want blank pages for left/right pagination? y
/1 all
  1  \**** ONESTART ****
  2  \m1="manusec2.doc.manu"
  3  \m2="yes"
  4  \if main in onestart.in.manu
/
```

Most of the use files will ask whether to add material at a specific line number or at the end of the workfile. Since ONESTART must appear at the *start* of your file, the ONESTART use file instead asks if the workfile is empty. If it is, the macros and include statements will be added to your workfile starting with line number 1. If it is not, the ONESTART use file will then ask you for a line number which will be the first line in your file; specify a line number *smaller* than the first existing line number in your workfile (for example, specify .1 if the first existing line number is 1). This ensures that the ONESTART information is always at the start of your document.

To print a formatted copy of your document, you must output your file through TDP's formatter to the HP2680. If you are using TDP's editor, use the FINAL command to output to *HP2680. If you are using HPSLATE, press the **OUTPUT** function key, then press the **Tdp** function key and output to HP2680.

Typically, you will use a separate file for each section of a large manual or document. If one particular section of your manual is exceptionally large, you may want to also divide that one section into multiple files. In that case, use ONESTART only at the start of the *first* file for that section. At the *end* of the first file, use an include statement for the second file for that section. If you need a third or fourth file for that section, add additional include statements in the appropriate order at the end of your first file. This allows you to "final" that section of your manual simply by "finaling" the first file. (Note, however, that include files must be in TDP editor format; hence, if you are using HPSLATE, you will have to first convert your additional files for a particular section before including them at the end of your first file.)

When you "final" a file containing one section or appendix of a larger document, the section or appendix number will be a question mark (?). When you include this file in your core file (see The Core File, below), the sections will be numbered 1,2,3,... and the appendixes A,B,C...

The Core File

If you are creating a large document, you will probably want to divide that document into multiple portions or sections. You should keep each major portion or section in a separate file, using ONESTART at the start of each file and "finaling" each file separately for review purposes (see Single Files, above). When you are ready to print your entire document, you should build the core file. The core file is a file which includes MANSTART, followed by include statements for each portion or section of your document in the sequence you want them to appear.

NOTE

If you are using HPSLATE/3000 as your editor, you will have to use option 2 of HPSLATE's `convert` function to convert your files to EDIT/3000 files before including them in your core file. This is because the TDP formatter can only accept files in TDP editor format as include files. (The core file itself may be either an HPSLATE file or a TDP editor file.)

MANSTART provides all the information needed to "final" your entire document to the HP2680. This is the same information provided by ONESTART when you "final" just one portion or section, except that the section numbers are declared to start at 1 and the appendix numbers at A (rather than question marks). In addition, MANSTART names the table of contents file to be a file named CONTENTS and sets the page numbering to 1, centered. (Refer to Generating a Table of Contents in section 4 for how to finalize your table of contents file.)

The core file for this manual, for example, is called `core.doc.manu` and looks like this (in its entirety):

```

**** MANSTART ****
m1="yes"
\in manstart.in.manu
\in manufrnt.doc.manu
\in manucon.doc.manu
\in manusec1.doc.manu
\in manusec2.doc.manu
\in manusec3.doc.manu
\in manusec4.doc.manu
\in manusec5.doc.manu
\in manuappa.doc.manu
\in manuappb.doc.manu
\in manuappc.doc.manu
\in manuappd.doc.manu
\in manuappe.doc.manu
\in manuappf.doc.manu
\in manuback.doc.manu

```

The description of MANSTART in section 5 helps us to understand this core file:

- 1) `m1` informs TDP that this is an 8.5 x 11 inch manual; hence, TDP knows to use the MANU90

Getting Started

environment file. If, instead, `m1` is set to "no", the MANUO environment file is used to produce a 9 x 8.5 inch document.

- 2) `\in manstart.in.manu` invokes the MANSTART include file.
- 3) The rest of the include statements specify the names of the files containing the different portions or sections of this manual -- in the order you wish them to appear.

In addition to the `\in manstart.in.manu` statement, you may want to include two additional include statements at the start of your file. These are:

```
\in numbered.in.manu
\in noblank.in.manu
```

The NUMBERED include file causes level heads to be numbered hierarchically (called subsection numbering). The NOBLANK include file inhibits the printing of extra pages which are otherwise printed to preserve left/right pagination. See MANSTART in section 5 for more information on these two include files.

If you are using TDP's editor, you can use the MANSTART use file to create your core file, as follows:

```
/use manstart.use.manu
Is workfile empty (Y/N)? y
Is this an 8 1/2 by 11 manual? y
Do you wish to number subsections (e.g. 3.5.1)? n
Do you want blank pages for left/right pagination? y
You will now be placed into add mode to include the files
for each section of the manual. Use // to terminate.
 4  \in manufrnt.doc.manu
 5
   .
   .
   .
```

The MANSTART use file, like ONESTART, asks if the workfile is empty. If it is, the macros and include statements are added to your workfile starting with line number 1. If it is not (that is, if your workfile already contains include statements for the sections of your manual and you simply want to add the MANSTART information at the start of the file), the MANSTART use file will then ask you for a line number which will be the first line in your file; specify a line number *smaller* than the first existing line number in your workfile (for example, specify .1 if the first existing line number is 1). This ensures that the MANSTART information is always at the start of your core file.

To print a formatted copy of your entire manual or document, output your core file through TDP's formatter to the HP2680. If you are using TDP's editor, use the FINAL command to output to *HP2680. If you are using HPSLATE, press the **OUTPUT** function key, then press the **top** function key and output to HP2680.

If you have divided one large section of your manual into multiple files, you should use ONESTART only at the start of the *first* file and use include statements for the additional files at the *end* of the first file. (Include files must be in TDP editor format, hence you will have to convert your files if you are using HPSLATE.) In such a case, only include the name of the *first* file in your core file. For example, suppose one section of your manual (say section 2) is quite large. You might create it in three different files, say:

```
sect2a
sect2b
sect2c
```


You should use `ONESTART` at the start of `sect2a`. At the end of `sect2a`, you should put two include statements:

```
\in sect2b
\in sect2c
```

To "final" section 2, you would simply "final" `sect2a`. In your core file, you would put an include statement for `sect2a` only (not for `sect2b` nor for `sect2c`) in order to include all of section 2.

Batch Printing

When you "final" either a single file or the core file to the HP2680, your terminal will be unavailable until the formatting is complete. In the case of a very short document, this will usually not be a problem. Larger documents, however, will take a considerable amount of time. To solve this problem, create a stream file in TDP's editor. For example:

```
/A
1   :JOB WENDY.KING/XYZ,DOC
2   :RUN TDP.PUB.SYS
3   FINAL FROM FCOPYMNL TO *HP2680,COPIES=2
4   EXIT
5   :EOJ
6   //
/
```

In the above example, the first line is a `:JOB` command; you must include any passwords for your username, account, or group (XYZ on the KING account, above). Next, the job runs TDP/3000, issues the `FINAL` command, and `EXITs` (note that you do not supply a colon for the TDP editor commands). Finally, we have a `:EOJ` command to terminate the job.

In order to execute this job, we first need to keep it. We can then stream it by using the `MPE STREAM` command from within TDP's editor or in response to MPE's colon prompt. This causes the job to execute and allows us to continue working at our terminal:

```
/K MNLJOB
/STREAM MNLJOB
#J106
/
```

Once you have created a stream file such as in the above example, you can text it in, modify the file name and/or the number of copies, keep it, and stream it again whenever necessary.

Note that if you are using `HPSLATE` as your editor rather than TDP, you will first have to convert your `HPSLATE` file to a TDP editor file before "finaling" it from a job. This is because `HPSLATE` cannot be run in batch mode. You can check on the status of your job by using MPE's `SHOWJOB` command from within TDP's editor or in response to MPE's colon prompt; when your job number (#J106 in the above example) no longer appears, you know that your job has completed. At that point, TDP's formatter has finished formatting your document and stored it in a spoolfile. How soon your document actually prints, however, depends on how busy the systems are and whether the spoolfile needs to travel over DS lines or not.

Getting Started

You can check on the status of your spoolfile by using MPE's SHOWOUT command with the ;JOB= option. If you use it from within TDP's editor, you must type a colon before the command (this is because TDP will otherwise interpret the ;JOB= option as another command). For example:

```
/:SHOWOUT SP;JOB=J106  
NO SUCH FILE(S)  
OUTFENCE = 6
```

When the message NO SUCH FILE(S) appears, as in the above example, your spoolfile has either left your system (if it needs to travel over DS lines to reach the laser printer) or printed. (Note that you will have to remember the job number which was assigned to your job when you initially streamed it.)

The job itself generates a listing which is sent to device class LP (system line printer). If you cannot find your HP2680 laser printer listing, it may mean that the job has terminated unsuccessfully; in this case, the job listing will be helpful since it supplies information which will (hopefully) help you understand what went wrong. In addition, you can check your ERRORLOG file which is automatically created by TDP.

Refer to the *3000 MPE Commands Reference Manual* (30000-90009) for more information on the STREAM, SHOWJOB, and SHOWOUT commands and for more information on batch jobs.

You may also want to learn about SPOOK, a utility which allows you to obtain information about and operate on spoolfiles. Refer to the *MPE System Utilities Reference Manual* (30000-90044).

CREATING DOCUMENTS

No matter what kind of document you are creating, you must begin each file with `ONESTART` or `MANSTART` (see *Creating and Printing Files*, earlier in this section); these provide information to TDP and the HP2680 which determines how your document will look when it is "finalized". By default, for example, text will be printed in the roman font. To change the "finalized" result at various points within your document, you can use the MANU Formatting System include files. In addition, you can use any of the TDP backslash or intraline (caret) formatting commands (see section 4 of the *TDP/3000 Reference Manual*, 36578-90001). For example, you can invoke a font other than roman by using the TDP formatting command `^fid` (see *Using Multiple Fonts* in section 3).

The MANU Formatting System provides you with a set of files to do certain typical and/or difficult TDP/HP2680 formatting. For example, if you are writing a manual or other document with multiple sections, you will want to invoke the logical page containing the section banner, position the section title within the banner, and reset the page numbering whenever you start a new section; an include file is provided to do this (and more!) for you. If you are writing simple documents or reports (without multiple sections), you will not make use of the section or appendix include files; however, you will still find many of the other include files useful -- such as level heads, notes, and bullets.

Section 5 describes all of the MANU formatting files in alphabetical order by use file name. It is suggested that you familiarize yourself with those formatting files you think you will be using by reading about them in section 5. Read the descriptions given under "discussion" carefully, as these will explain the end result when you "final" your document. For example, the discussion of the NOTE use file explains that two blank lines are printed before the note logo and after the text of the note; thus you know not to leave blank lines before and after the note since this is all being done for you.

In addition, if you are using `HPSLATE` rather than TDP's editor, you will have to fully understand the specified "syntax" -- since you will have to add these lines yourself, rather than having the use file add them for you.

To help you read about the formatting files in a meaningful sequence (rather than alphabetically), the table which follows provides groups of formatting functions and corresponding use file names. Look up in section 5 those use file names which correspond to the formatting functions you think you will need most often.

Table 2-1. MANU Formatting Functions

Formatting Functions	Use Files (see section 5)
Starting any file Finaling multiple files	ONESTART MANSTART
Sections Appendixes	SECTION APPENDX
Level 1 heads Level 2 heads Level 3 heads Level 4 heads	LEVEL1 LEVEL2 LEVEL3 LEVEL4
Notes Cautions Warnings	NOTE CAUTION WARNING
Image mode "Block" mode Format mode	IMAGE BLOCK FORMAT
Bullets and numbered items	BULLETS
Boxes and bold boxes	BOX
2-column tables 3-column tables 4-column tables	TAB2COL TAB3COL TAB4COL
Page heads (reference items) Syntax Parameters Discussion Example Text reference An entire reference item	PAGEHEAD SYNTAX PARMS DISCUSS EXAMPLE TEXTREF EASYREF
Subparameters	SUBPARMS
Cover Cover update page Title page Copyright page Printing history List of effective pages Preface Conventions The entire front matter	COVER COVERUPD TITLE COPYRITE HISTORY EFFPAGES PREFACE CONVEN FRONTMAT

Table 2-1. MANU Formatting Functions (continued)

Formatting Functions	Use Files (see section 5)
Internal cover	LABCOV
Table of contents Figures and Tables	CON FIGTAB
Reader comment sheet/Business reply mail Back cover The entire back matter	RCS BACKCOV BACKMAT
Index	INDX
Memos	MEMO

MORE ABOUT FORMATTING

SECTION

3

The MANU formatting files perform many of the routine formatting tasks for you. By simply using the MANU files, you can easily produce documents containing section banners, level heads, notes, bullets, boxes, and so on. There will be times, however, when you will want to do something which is not automatically performed by a MANU formatting file. For example, you may want to put a particular word or phrase in **bold (romanb)** the first time you define it. This section provides the information you will need about using TDP with the MANU environment files.

USING MULTIPLE FONTS

The MANU Formatting System provides you with a set of fonts compiled into the MANU0 and MANU90 environment files. The ONESTART and MANSTART include files define the font identifications (called fontids) which allow these fonts to be used. The other MANU include files use the fontids when automatically switching fonts. For example, the LEVEL1 include file automatically switches to the helvb14 font to print the level 1 head; the SYNTAX include file invokes the delite font for your syntax; the NOTE include file switches to the achtung font to print the note logo (see appendix C for a listing of the MANU fonts). In addition, you may change from one font to another by using certain TDP formatting commands.

Fonts and Fontids

In the MANU0 and MANU90 environment files, roman is the default font (also called the base or 0th font); hence, if you have not specified otherwise, text is automatically printed in the roman font (used for the main text of documents). To invoke another font, use the `^fid` intraline formatting command, where `id` is the fontid character (refer to `^f` in section 4 of the *TDP/3000 Reference Manual*, 36578-90001). The fontid characters of the MANU Formatting System are defined as follows:

r	roman
s	romani
t	romanb
d	delite
e	delitei
f	deliteb
g	deliteg
i	delitegi
v	helvb10
w	helvb12
x	helvb14
y	helvb20
z	helvb24
l	linedraw
b	boxdraw
m	math
p	parens
a	achtung
k	keys
h	hplogo

The font represented by the fontid will continue until:

- 1) a `^s` command is encountered, which returns you to the roman font, or
- 2) another `^fid` command is encountered, which invokes the specified font.

For example:

This is roman; **this is roman bold**; this is roman
once again.

When "finaled", the above lines will appear as:

This is roman; **this is roman bold**; this is roman once again.

Note that ^s always returns you to the roman font. If you do not wish to return to roman, use the ^fid command to invoke another font, as follows:

This is roman; **this is roman bold** and *this is
roman italics*^s.

When "finaled", this example appears as:

This is roman; **this is roman bold** and *this is roman italics*.

It is important to understand that the intraline commands do *not* appear in the "finaled" copy of your document -- although they *do* occupy space in your input file. In the above example, for instance, the ^ft, ^fs, and ^s characters "disappear" when the document is "finaled".

Font Pairs

Certain fonts in the MANU0 and MANU90 environment files are *paired*. For example, ^fd really invokes the font pair delite/delitei, rather than just the delite font. This means that delitei is the *alternate font* when using delite. Four font pairs exist in the MANU environment files:

r	roman/romani
d	delite/delitei
g	deliteg/delitegi
q	delite/roman

A font which is an alternate font in a font pair can be invoked with TDP's ^a command whenever you have previously invoked the corresponding main font. The alternate font will then continue until a ^n is encountered, which returns you to the main font of the pair. (Refer to ^a in section 4 of the *TDP/3000 Reference Manual*, 36578-90001.)

NOTE

A ^n *must* be used following a ^a command! Simply using ^s to return to roman, or ^fid to invoke a new font, will not work. (If you wish to do either of these things right after using an alternate font, use ^n immediately followed by ^s or ^fid.)

More About Formatting

The roman/romani font pair is especially useful when switching to italics within the text of your document, as follows:

```
This is roman, this is roman italics, and this is
roman once again.
```

When "finald", the above lines will appear as:

```
This is roman, this is roman italics, and this is roman once again.
```

Note that the exact same result can be obtained as follows:

```
This is roman, this is roman italics, and this is
roman once again.
```

The delite/delitei font pair is useful when doing syntax. For example:

```
\**** SYNTAX ****
\m1="30"
\in syntax.in.manu
^maLET avar[(asubscript)] = avaluenmb
\in endbox.in.manu
```

The SYNTAX include file invokes the font pair delite/delitei, hence we can use ^a and ^n to switch in and out of the alternate font (delitei) for the lowercase, user-supplied parameters. Note that we could use ^fe and ^fd (not ^s) instead, but ^a and ^n are easier in this case.

The EXAMPLE include file invokes the font pair delite/roman. We can then use ^a to switch to roman for comment lines next to the example and ^n to switch back to delite for the example itself. In such a case, you will want to try to "line-up" the ^a commands. For example:

```
\**** EXAMPLE ****
\in example.in.manu
NAME? WENDY KING           ^aEnter name^n
NEW USER? YES             ^aAnswer YES if new user^n
.
.
.
\in endexamp.in.manu
```

Conversion Fonts

In order to use the linedraw, boxdraw, math, parens, achtung, keys, and hplogo fonts, you need to know which symbols or logos are represented by which characters. Because of this, these fonts are often called conversion fonts. Some examples are:

- ^fkB^s prints the **BREAK** logo;
- ^fb[]^s causes to print;
- ^fmA^s will print the symbol √.

Refer to appendix C (Fonts and Logical Pages) for the conversions of these fonts.

Underlining

Two of TDP's intraline underlining commands (^u and ^w) are terminated by ^s (see ^u and ^w in section 4 of the *TDP/3000 Reference Manual*, 36578-90001). This means you will *always* return to the *roman* font after using these commands to do underlining. Hence, you must be sure to reinvoke your font with the ^fid command if you desire a font other than roman. This happens quite often when, in an example, you wish to underline the user input. For example:

```

NAME? WENDY KING
NEW USER? ^_YES
.
.
.

```

In the above example, we had to reinvoke the delite font after using ^u and ^s to start and stop underlining. Note that we did not have to do this when we used TDP's ^_ command to underline for just one word (refer to ^_ in section 4 of the *TDP/3000 Reference Manual*, 36578-90001).

If you use the EXAMPLE include file provided with the MANU Formatting System, you are automatically in the font pair delite/roman. You should reinvoke this font pair (fontid q) if you use TDP's intraline commands to underline. For example:

```

\**** EXAMPLE ****
\in example.in.manu
NAME? WENDY KING
NEW USER? ^_YES
.
.
.
\in endexamp.in.manu

```

^aEnter name^n
^aAnswer YES if new user^n

Except for the ^u, ^s, and ^fq commands, the above example is the same as one shown earlier (see Font Pairs). Note that the ^a commands no longer "line-up" in the input file. This is due to the ^u, ^s, and ^fq commands which we inserted *after* "lining-up" the comment lines. You should always *insert* intraline commands after your text is correctly "lined-up"; although no longer "lined-up" in your input file, the appearance of the "finalized" document will be correct.

USING THE BLANK CHARACTER

The ONESTART and MANSTART include files define the grave accent character (`) to be a necessary blank. This means that whenever a grave accent character appears in your text, it will be replaced by a blank (or space) character when your file is "finalized". In addition, if the grave accent character appears between two words (as in Mr. `Hancock), the phrase will be treated as one unit. This can be very useful when in format mode with right justification since:

- 1) the phrase will not be broken across lines when the paragraph is formatted, and
- 2) the phrase will not be separated by additional blanks when the paragraph is right justified.

We suggest you use grave accents whenever you have a phrase in the delite font in the middle of a paragraph. For example:

```
Occasionally, you may wish to invoke a logical page
for a specific application. To do this, use TDP's
\LD\LAYOUT\NEW^an^n command, where ^fen^s is
the logical page number of the logical page you wish to
.
.
.
```

The grave accents prevent the phrase \LAYOUT NEW n from being separated by additional blanks and from being broken across two lines when the paragraph is formatted and right justified. (See Changing Logical Pages in this section for the "finalized" version of this paragraph).

If you need to print a grave accent, you will have to first assign a different character as the necessary blank character. To do this, use TDP's \BLANK command. For example:

```
\BLANK "$"
Grave accents will now print (````), while dollar
signs will be translated into blanks ($$$$).
\BLANK " "

```

When the above paragraph is "finalized", we get:

Grave accents will now print (````), while dollar signs will be translated into blanks ().

Always be sure to reset the blank character to the grave accent, as we did in the above example.

CHANGING LOGICAL PAGES

The MANU Formatting System provides you with a set of logical pages compiled into the MANU0 and MANU90 environment files. The most common use of these logical pages is to generate section and appendix banners, front matter boxes, and so on, which exist as forms associated with the logical pages. Appendix C contains a list of the logical pages available in the MANU environment files.

Invoking the correct logical page is done automatically by the MANU include files. For example, the COPYRITE include file invokes the logical page which has the form of the copyright box associated with it (logical page 7).

Occasionally, you may wish to invoke a logical page for a specific application. To do this, use TDP's `\LAYOUT NEW n` command, where *n* is the logical page number of the logical page you wish to invoke (see appendix C for the logical page numbers). This will force a physical page eject (a new page), activate the new logical page, and automatically deactivate the new logical page at the next pagebreak (whether natural or forced). Thus, the new logical page prints once and control returns to the previously activated logical page.

Refer to `\LAYOUT` in section 4 of the *TDP/3000 Reference Manual*, 36578-90001.

NOTE

The TDP commands `\ACTIVATE` and `\DEACTIVATE` may also be used. However, we suggest that they be avoided since they erratically require dummy pages to be printed in order to operate correctly.

USING MACROS

The TDP formatter allows you to represent a string by a two-letter code, called a macro. Macros can be useful in a number of situations. For example, suppose we are writing a manual for a new product called WIDGET/3000. Since the product's name will most likely change, it would be a good idea to use a macro for the name of the product. At the start of each file (following ONESTART), we would put the following line:

```
\me="WIDGET/3000"
```

Here we have defined the macro `me` to be equal to the string `WIDGET/3000`. We can now use `^me` anywhere in our document to refer to the name of our product. For example:

```
Your  goes beep when it moves, bop when it stops, and whir when it stands still; we never knew quite what it was and we hope you never will.
```

When the above document is "finalized", `^me` is replaced by the string `WIDGET/3000`. Once the product's name has been decided upon, we simply modify the string in the `\m` command at the start of our file and the new name will appear everywhere in place of `^me` the next time we "final".

A macro is defined with the `\mid` command and used (or "called-up") with the `^mid` intraline command. The `id` must be a single character or digit. Refer to M(Macro) in section 4 of the *TDP/3000 Reference Manual* (36578-90001).

The MANU/3000 Formatting System currently makes use of many macros. When defining your own macros, you should be careful not to use those reserved for use by the Formatting System. The easiest way to remember which macros you may use is to remember that you may use `e` through `n` as the `id`. That is:

```
me mf mg mh mi mj mk ml mm mn
```

are all macros you may define within your files. (Appendix E documents the macros used by the MANU Formatting System.)

When you use a macro for the name of a product, you will probably define it *once* (at the start of your file) and use it repeatedly throughout your document. Note, however, that you can also *redefine* a macro simply by repeating the `\mid` command with the same `id`, but with a different definition, at any point in your file; subsequent `^mid` commands will "call-up" the new definition of the macro.

Macros are very useful when you are entering text which will be printed in image mode and which is too long to fit on one editor line. Remember that a line in your input file does *not* correspond to what will fit on a line of your "finalized" document. In addition, if you have many intraline commands (font changes, underlining, etc.), your line is "really" a lot shorter. If you are using HPSLATE as your editor, you cannot expand the length of your input lines (it is limited to 80 characters); even if you are using TDP's editor, you may not *wish* to expand your line length past 80 since you will then be unable to use screen mode for your entire file. This problem often arises when documenting syntax; see Creating Syntax in section 4, for an example.

LENGTHS AND MARGINS

It is important to understand that there are two types of margins:

- 1) the **formatter or output margins**, which determine the width of your "finalized" document, and
- 2) the **editor or input margins**, which determine the maximum length of the lines in your editor file (also called your input file or workfile).

Output Margins

Because the output margins are the ones we are most often concerned with, the phrases **left margin** and **right margin** will be used to refer to the left and right *output margins*.

The MANU Formatting System sets the formatter margins for you by using TDP's `\LFT` and `\RHT` formatting commands. When you "final" an 8 1/2 x 11 inch document to the HP2680, the left and right margins are set at 1 and 56, respectively; when you "final" a 9 x 8 1/2 inch document to the HP2680, they are set at 2 and 57, respectively.

The `\LFT` and `\RHT` formatting commands can appear at two places in your file: (1) at the beginning of your file, and (2) following a `\LAYOUT` command. If you use the MANU Formatting System, you should never have to use these two commands; the `ONESTART` and `MANSTART` include files set the margins for you. There may be times, however, when you wish to *temporarily* move the left or right margin in toward the center of the page. To do this, use TDP's `\INLFT` and `\INRHT` commands (see section 4 of the *TDP/3000 Reference Manual*, 36578-90001).

`\INLFT n` moves the left margin *n* positions in from its original value; `\INRHT n` moves the right margin *n* positions in from its original value. Note that *n* is a *relative* amount; the margin moves *by* that amount from its *original* position -- not from its current position. For example, if the right margin is currently set at 56, `\INRHT 2` moves it in to position 54; a subsequent `\INRHT 5` command moves the right margin to position 51 (*not* 49). To reset the margins to their original values, use `\INLFT 0` and `\INRHT 0`.

For more information on `\INLFT` and `\INRHT`, see *Adjusting the Margins* later in this section.

Input Margins

The length of the lines in your editor file depends on the editor you are using. For example, `HPSLATE/3000` files have lines 80 bytes long, while `TDP/3000` files can have lines of any length.

The important thing to understand is that the margins and length of the lines in your input file have little bearing on the output margins you will see when your document is "finalized" (see *Output Margins*, above).

If you are using TDP's editor, there are three editor commands relating to your editor margins. They are entered in response to the TDP editor slash (/) prompt:

- `SET LEFT=n` refers to the left margin of your workfile. We can assume the left margin of your workfile is always set to 1.
- `SET LENGTH=n` refers to the physical width of your workfile; that is, the maximum number of characters that will fit on one line in your workfile.

More About Formatting

- SET RIGHT=*n* refers to the right margin of your workfile. If you exceed this value, the line will be truncated. If you are in add mode and a line is truncated, you will be warned; use the *//m linenumber* command in response to the next line number prompt to "clean up" what text has been left on the line, then begin the next line with the text which was truncated.

You can change the line length of your workfile by up to 22 characters at a time. Use the command VERIFY RIGHT, LENGTH to see the current values, then use the SET commands explained above to change right and length by a maximum of 22 at a time. Right must always be less than or equal to length, so set right before length when reducing a file and vice versa when increasing. When the file is kept, it will have the new line length. If you need to change the line length by more than 22 characters, do it in 22 character steps -- keeping and texting after each step.

Remember that the commands affect your workfile only, not your "finald" file. Also, the new values remain in effect *only* for the duration of the current session; the default values are restored every time you enter TDP's editor.

Refer to SET and VERIFY in section 3 of the *TDP/3000 Reference Manual* (36578-90001).

UNDERSTANDING PROPORTIONAL FONTS

Some fonts in the MANU environment files are **monospaced fonts**, while others are **proportional fonts**. A monospaced font is one in which all characters and symbols are exactly the same width; a proportional font is one in which the widths of the characters and symbols vary. In the MANU environment files, roman, romani, romanb, helv10, helv12, helv14, helv20, and helv24 are proportional fonts; delite, delitei, deliteg, deliteb, linedraw, boxdraw, parens, and math are monospaced fonts.

Lining-Up Text

It can be difficult to "line-up" text when using a proportional font. For example, suppose the following text appears in your workfile (or input file):

```
\image
ADD      Adds an association entry
CHANGE   Changes attributes of entity within a relationship
CREATE   Creates an entry for a new entity
DELETE   Deletes an association entry
```

When "finaled", the result would be:

```
ADD      Adds an association entry
CHANGE   Changes attributes of entity within a relationship
CREATE   Creates an entry for a new entity
DELETE   Deletes an association entry
```

These strange, undesired results are because we are using a proportional font (roman). Hence, although there are a total of eight characters and spaces before each of the sentences in our input file, the amount of space they take up *when output in the roman font* varies. One easy way to avoid this is to put all the characters (the words *and spaces*) before the sentences in the delite font, as follows:

```
\image
^f^dADD      ^sAdds an association entry
^f^dCHANGE   ^sChanges attributes of entity within a relationship
^f^dCREATE   ^sCreates an entry for a new entity
^f^dDELETE   ^sDeletes an association entry
```

Because delite is a monospaced font, every character (including a space) is exactly the same width. Thus the eight characters/spaces before each sentence take up an equal amount of space when the document is "finaled" -- and the sentences "line-up" correctly:

```
ADD      Adds an association entry
CHANGE   Changes attributes of entity within a relationship
CREATE   Creates an entry for a new entity
DELETE   Deletes an association entry
```

It is appropriate to use delite rather than roman for the commands in this example. There may be times, however, when it is not appropriate to use delite as we did here. In these cases you will want to make use of TDP's column-related commands. In addition, whenever you want multiple columns with some of the entries in *format* mode (rather than image), it will be necessary to use TDP's column commands no matter what fonts you are using. Refer to Using Multiple Columns later in this section.

Formatting Modes

You may change formatting modes for all or part of a particular entry in a row, and for all entries following it until the formatting mode is changed again. It is important to understand that formatting modes are *not* in any way "tied" to the columns. In the example which follows, we continually respecify the formatting modes for the particular entries of each row such that all entries for the first two columns are in image mode and all entries for the third column are in format mode. The command `\NEXT;FORMAT` precedes each entry for the third column, which is then followed by a `\NEXT;IMAGE` command:

```
\column (15,15,26)
```

```
\image
```

```
^fdREORDER CATEGORY
```

```
\next
```

```
PARENT CATEGORY
```

```
CHILD CATEGORY
```

```
NEW POSITION^s
```

```
\next; format
```

Reorders a child category within a parent category's entry list. Prompts for the parent's category, the child's category, and the new position.

```
\next; image
```

```
.  
.
.
```

```
\column 1
```

Similarly, because the `PARMS` and `SUBPARMS` include files use column mode, you may change formatting modes for all or part of the explanation of a parameter or a subparameter. In the following example, center mode is used to create "headings" for the columns containing subparameters and their corresponding explanations:

```
\**** PARMS ****
```

```
\m1="Parameters"
```

```
\in parms.in.manu
```

```
^feedit-mask^s
```

```
^ma
```

```
.  
.
.
```

For data elements, the edit-mask may consist of any of the following characters:

```
\**** SUBPARMS ****
```

```
\in subparms.in.manu
```

```
\center 1
```

Character

```
^ma; center 1
```

Effect

```
^mb
```

```
.  
.
.
```

Notice the macros `^ma` and `^mb`. Since `PARMS` and `SUBPARMS` use column mode, they also use the `\NEXT` command between entries, as well as the commands to change formatting modes. However, you

don't see any of these commands because macros are used instead. The macros are defined by the PARMs and SUBPARMs include files. For example, PARMs defines `^ma` as the command `\NEXT;FORMAT`. When "finald", TDP's formatter replaces the macros with the required commands.

Indented Columns

When you use multiple columns in TDP, the first column always starts at the left margin (`\LFT`). If the widths of the columns add up to less than 56 "ems", there will be space between the end of the last column and the right margin (`\RHT`).

If you wish to have your first column indented (not at the left margin), you must either use a "fake" column or use an `\INLFT` command for each entry of that column (see *Adjusting the Margins*, which follows). A "fake" column is a column which you specify, but which does not contain any text; use `\NEXT` to skip each entry for that column.

In the following example, the first column is a "fake" column which is 10 "ems" wide; extra `\NEXT` commands are included which skip the entry for that column in each row. Also notice that the last column ends 10 "ems" in from the right margin because the widths of the columns add up to 46 (10+10+26) -- 10 less than 56. The result is two columns of text which are centered between the margins -- with equal amounts of white space on either side:

```
\column (10,10,26)
\next
^fdADD^s
\next
Adds an association between unlike entitles.
\next;\next
^fdCHANGE^s
\next
Changes the attributes of a child within a relationship or the description
of the relationship which was established with the ^fdRELATE^s command.
\next;\next
^fdCREATE^s
.
.
.
\col 1
```

The `TAB2COL`, `TAB3COL`, and `TAB4COL` include files have a built-in "fake" first column, which starts as 1 "em" of white space between the table and the left margin. You can use this "fake" column to indent your table simply by increasing the width of the first column. If you want to center your table, just leave an equal amount of white space on the right side of your table, which you do by making the total of the widths of all the columns equal to 56 minus the width of the first "fake" column (See *Generating Tables*, section 4).

Margins

In the examples so far, there is no space or margin left between the columns -- except whatever space is left following image mode entries in particular rows. Often, this is not a problem. If, however, you have two entries next to each other in format mode, this could be confusing! To *control* the size of the margin between columns, use TDP's `\CMARGIN` command. For example:

More About Formatting

```
\column (14,13,25); cmargin 2
```

In this case, a margin of two columns (two roman "ems") will be left between the first and second columns, as well as between the second and third columns.

You must be sure to turn off the margin with `\CMARGIN 0`, otherwise your margin will be used again the next time multiple columns are encountered. Refer to section 4 of the *TDP/3000 Reference Manual*, 36578-90001.

Estimating Column Widths

Although you can choose the desired width of your columns by looking at a line of 56 roman "ems", it is quite another story to figure out whether the information you wish to print in a particular column will fit. If you are using format mode for a particular entry, the information will of course *always* fit -- no matter what font you are using. If, however, you are in image mode for an entry, the information may *not* fit.

EMs with Monospaced Fonts

If you are going to print text in image mode *in a monospaced font*, it is possible to determine how many roman "ems" that text is equivalent to. Every character in the delite, delitei, deliteb, deliteg, delitegi, linedraw, and parens fonts (but *not* boxdraw or math) is exactly 15 laser printer dots wide; a roman m character is 22 dots wide. Hence, to calculate an equivalent number of "ems" for text printed in one or more of these fonts, use the following formula:

$$\text{number of "ems"} = n \times 15 / 22$$

where n is the number of characters (including spaces) you wish to print. For example, suppose you wish to produce the following text within a column:

```
[ADD] subcommand
 [A]  subcommand
```

Your input file would contain a column command, followed by these lines:

```
.
.
.
\image 2
^fp^fdADD^fpW^-^fe subcommand^+      (16 characters)
^fpA^fd A ^fpS^s                      (5 characters)
\next
.
.
.
```

The longer of the two lines consists of 16 characters. Note that you count the characters which represent symbols in the parens font as well as the words and spaces in the delite font. You do *not* count the intraline (caret) commands. The number of roman "ems" needed to print this line is:

$$16 \times 15 / 22 = 10.9$$

or, approximately, 11 "ems". Hence, it would be wise to use a column which is at *least* 11 "ems" wide -- preferably 13 or 14 if no margin is specified.

Many of the MANU Formatting System include files use the \COLUMN command to produce multiple column output. For example, the PARMS include file uses two columns to produce the parameters portion for a reference item. The first of these two columns is 16 "ems" wide and the entries printed in that column are always in image mode; it is intended for the parameters you wish to explain. The parameters are typically printed using one or more of the delite fonts (and possibly the parens font). Because all these fonts have characters 15 dots wide, we can calculate the maximum number of characters which can be printed in any combination of these fonts within that column. A roman "em" is 22 dots wide; hence the maximum number of delite, delitei, deliteb, deliteg, delitegi, and/or parens characters is:

$$16 \times 22 / 15 = 23.5$$

Because no margin is specified in the PARMS include file, you should probably use a maximum of 21 characters in the delite fonts (and possibly parens font) per parameter line. Remember that spaces count, but intraline (caret) commands do *not*. For example:

```

\**** PARMS ****
\m1="Prompts"
\in parms.in.manu
.
.
^mb
^fdRESTRICT CLASS TO A FILE (N/Y) ?^s (19 characters)
FILE (N/Y) ?^s (12 characters)
^ma
Enter a ^fdY^s if the scope of the class is to be
restricted to one file.
^mb
^fdSTORAGE LENGTH (^alen^n)^s (20 characters)
^ma
The system calculates the storage length in bytes and displays it as
^felen^s. Enter a new value, otherwise ^felen^s is used.
.
.
\in endparms.in.manu

```

We want to restrict each parameter line to a maximum of 21 characters. Since the prompt RESTRICT CLASS TO A FILE (N/Y) ? is 32 characters long, we had to use two lines: the first line is 19 characters long, the second is 12. The next prompt, STORAGE LENGTH (*len*), is 20 characters long; hence we can put it on one line.

The SUBPARMS include file also uses multiple columns. As part of a parameter explanation, SUBPARMS allows you to produce subparameters and their corresponding explanations by setting up three columns. The first of these columns is the same width as the first column set up by PARMS; it is *automatically* skipped for you by SUBPARMS. Entries for the second column are always in image mode; these are the subparameters you wish to explain. Entries for the third column are always in format mode; these are the explanations of the subparameters. Because subparameters are typically printed using one or more of the delite fonts (and possibly the parens font), we can calculate the maximum number of characters which can be printed in these fonts within the SUBPARMS second column: the column is 11 "ems" wide, a roman "em" is 22 dots wide, and the characters are each 15 dots wide, hence:

$$11 \times 22 / 15 = 16.13$$

More About Formatting

Because no margin is specified in the SUBPARMS include file, you should probably use a maximum of 14 characters in the delite fonts (and possibly parens font) per subparameter line. Remember that spaces count, but intraline (caret) commands do *not*.

EMs with RWIDTH

If you are printing an entry for a column in image mode in the *roman font*, you need to know the size (number of dots) of *every roman character* in order to calculate the equivalent number of roman "ems". The program called RWIDTH in the TOOLS group of the MANU account will calculate this for you. For example:

```
/RUN RWIDTH.TOOLS.MANU
```

```
Char Width program. "" = blank. CR = done.
```

```
Enter text: INput/OUTput````
```

```
STORAGE LENGTH (len)````
```

```
28.66 mm    25 spaces    9 ems    15 Chars
```

```
Enter text: RETURN
```

```
END OF PROGRAM
```

```
TDP/3000
```

The RWIDTH program can be run from within TDP or in response to MPE's colon prompt. In the above example, we ran it from within TDP and entered the string `INput/OUTput`````. The three grave accent marks indicate three blank characters immediately following the string. The RWIDTH program tells us that this string (including the three trailing blank characters), *when printed in roman*, will be 28.66 millimeters long; it is equivalent in length to 25 roman spaces; it is also equivalent to 9 roman "ems"; and it is actually 15 characters (including the three spaces). The important piece of information for us is that it is equivalent in length to 9 roman "ems". Hence, the string `INput/OUTput` will fit in a column 9 or more "ems" wide -- with at least three spaces following the text.

In order to determine what column width to specify for *image mode* entries in *roman*, you will have to enter each line of text to the RWIDTH program. We suggest you follow each line of text with a consistent number of grave accent marks (for example, two) to indicate the minimum number of spaces you want following the entries in that column. Take the *largest* "em" value returned and use that for the width of your column. If you have used a consistent number of grave accent marks, you can use a `\COLUMN` command without using `\CMARGIN`; each line of roman text in an entry for that column will have at least that many spaces following it.

Note that when using the RWIDTH program to calculate roman characters in terms of "ems", you must be careful to enter text in upper or lower case -- as it will eventually appear in your document. Also, do not enter a colon (:) as part of the string; RWIDTH will interpret a colon as the end-of-data.

More About Formatting

When using multiple columns, you may use the `\INLFT` and `\INRHT` commands to change the left or right margins of all or part of a particular entry. For example:

```
\COLUMN (20,36)
This text will appear in the first column, which is 20 "ems" wide.
\next
This text will appear in the second column, which begins at the 21st "em".
\SPACE 1; INLFT 5
This text will appear in the second column also, but will begin at the
26th "em" (indented by five "ems").
\SPACE 1; INLFT 0
.
.
.
\COLUMN 1
```

Similarly, because the `PARMS` and `SUBPARMS` include files use column mode, you may use `\INLFT` and `\INRHT` to change the left and right margins for particular entries. In the following example, we use `\INLFT 5` and `\INLFT 0` to change the left margin of each entry in the subparameters column:

```
\**** PARMS ****
\m1="Parameters"
\in parms.in.manu
^feedit-mask^s
^ma
A character string enclosed in quotes, made up of insertion and
place-holding characters, as follows:
\**** SUBPARMS ****
\in subparms.in.manu
INLFT 5
^fd^s
^ma; INLFT 0
A character from the source data element replaces each caret.
^mb; INLFT 5
^fdZ^s
^ma; INLFT 0
Leading zeros are suppressed.
^mb; INLFT 5
^fdDR^s
^ma; INLFT 0
When used as the last characters of the edit mask, a negative
data element is displayed with a trailing ^fdDR^s.
\in endsubp.in.manu
.
.
.
\in endparms.in.manu
```

Since the paired sets of include files NOTE/ENDNOTE, CAUTION/ENDNOTE, WARNING/ENDNOTE, and BULLETS/ENDBULL use \INLFT and \INRHT, you may use these paired sets of include files to produce notes, cautions, warnings, or bullets within the explanation of a parameter or subparameter. The ENDNOTE and ENDBULL include files reset the left and right margins to 0. The following example shows the use of NOTE within the explanation of a parameter:

```

\**** PARMS ****
\m1="Prompts"
\in parms.in.manu
.
.
^fdGROUP^s
^ma
Enter the name of an existing HP Inform/3000 group.
\**** NOTE ****
\in note.in.manu
A pound sign indicates the element will not be displayed on the menu.
\in endnote.in.manu
.
.
\in endparms.in.manu

```

Using INDENT

Most of TDP's column-related commands are expressed in terms of "ems". One exception to this is the \INDENT command, which is expressed in terms of *spaces*. Since roman is the default font of the MANU environment files, the values specified with the \INDENT command represent the number of *roman spaces* you want printed before the first and successive lines of your paragraphs. (Refer to \INDENT in section 4 of the *TDP/3000 Reference Manual*, 36578-90001.)

Often, you will not use an \INDENT command. TDP's formatter automatically uses the number of spaces before your first input line for where the first line of the paragraph should start; it uses the number of spaces before your second input line for where the second and all successive lines of the paragraph should start. In this way, paragraphs can be *either* "over-hanging right" or "over-hanging left" by typing in the first line indented to the right or extended to left, respectively.

If you are using a proportional default font, however, and if it is critical exactly how the second and successive lines of your paragraph "line-up" with the first line, you will have to use \INDENT. This is because a space, just like *any* character in a proportional font, is its own special size (which is not necessarily the same as any other character); hence, although text may appear properly "lined-up" in your input file, the results when the file is "finaled" are not so nice.

Spaces with RWIDTH

Because the default font in the MANU environment files is a proportional font (roman), this "lining-up" problem occurs no matter what font you are printing in. Let's first look at what happens when printing in roman. Suppose your input file contains the following:

More About Formatting

indirect link A link between two files, each containing elements needed by HP Inform/3000 for a report, which is formed by linking through one or more other files which do not contain elements needed for the report.

In this case, it is critical that the second and successive lines of the paragraph "line-up" under the word A of the first line. When "finalized", however, the result is:

indirect link A link between two files, each containing elements needed by HP Inform/3000 for a report, which is formed by linking through one or more other files which do not contain elements needed for the report.

To solve this problem, you could use a `\COLUMN` command (see Using Multiple Columns, earlier) or an `\INDENT` command. To use `\INDENT`, you first need to know the number of roman spaces which are equivalent to your "over-hanging left" string. To figure this out you would have to know the size (number of dots) of every roman character. The program `RWIDTH` in the `TOOLS` group of the `MANU` account will do this calculation for you. For example:

```
/RUN RWIDTH.TOOLS.MANU
```

```
Char Width program. "`" = blank. CR = done.  
Enter text: indirect link````
```

```
indirect link````  
23.82 mm 21 spaces 8 ems 16 Chars
```

```
Enter text: RETURN  
END OF PROGRAM  
TDP/3000  
/
```

In the above example, we ran `RWIDTH` from within TDP's editor and entered the string `indirect link`````. The three grave accent marks indicate three blank characters immediately following the string. The `RWIDTH` program tells us that this string (including the three trailing blanks), when printed in roman, will be 23.82 millimeters long; it is equivalent in length to 21 roman spaces or to 8 roman "ems"; and it is actually 16 characters long. The important piece of information for the `\INDENT` command is that it is equivalent in length to 21 roman spaces. (Remember that when using the `RWIDTH` program to calculate roman characters in terms of roman spaces, you must be careful to enter the text in upper or lower case -- exactly as it will eventually appear in your document).

In this case, then, we use `\INDENT 0,21` as follows:

```
\indent 0,21  
indirect`link````A link between two files, each containing  
elements needed by HP Inform/3000 for a report, which is  
formed by linking through one or more other files which do not  
contain elements needed for the report.  
\indent off
```

The grave accents are used to prevent TDP from inserting extra blank characters when right-justifying. The result, when "finalized", is:

indirect link A link between two files, each containing elements needed by HP Inform/3000 for a report, which is formed by linking through one or more other files which do not contain elements needed for the report.

Note that the "lining-up" may at times not be perfect since the number of spaces `RWIDTH` calculates is rounded to the nearest whole number.

We suggest that you use 0 for the first value of your `\INDENT` command and use `\INLFT` to move the left margin over by the amount you want. Remember to use `\INLFT 0` and `\INDENT OFF` following your text.

Spaces with Monospaced Fonts

Suppose that, in the above example, we want to print the words *indirect link* in *delite* and *delitei*, rather than roman. The `RWIDTH` program cannot be used since it assumes the text you enter will be printed in roman. *Delite* and *delitei* are monospaced fonts, with each character 15 dots wide; a roman *space* is approximately 8 dots wide. Hence, we can calculate the equivalent number of roman spaces for text printed in *delite* or *delitei* by using the following formula:

$$\text{number of spaces} = n \times 15 / 8$$

where n is the number of characters (including spaces) you wish to print. Since every character in the *delite*, *delitei*, *deliteb*, *deliteg*, *delitegi*, *linedraw*, and *parens* fonts is 15 dots wide, this formula can be used for text printed in all of these fonts.

Suppose you want to print the string *indirect link* followed by three spaces. Counting the space between the two words and the three trailing blanks, but *not* counting the intraline commands, this string is 16 characters long. The equivalent number of roman spaces is:

$$16 \times 15 / 8 = 30$$

Hence, we use `\INDENT 0,30` to print the string in *delite* and *delitei*, as follows:

```
\indent 0,30
```

```
^feindirect^fdlink^^^sA link between two files, each containing
elements needed by HP Inform/3000 for a report, which is
formed by linking through one or more other files which do not
contain elements needed for the report.
```

```
\indent off
```

The result, when "finaled", is:

indirect link A link between two files, each containing elements needed by HP Inform/3000 for a report, which is formed by linking through one or more other files which do not contain elements needed for the report.

SPECIFIC TASKS

SECTION

4

Producing a reference manual, or similar document, requires a number of specific formatting tasks which are very complex. The MANU formatting system includes a variety of special tools which make these tasks easier for the writer. This section gives instructions for the following specific tasks:

- Creating Syntax.
- Generating Tables.
- Including HPDRAW Figures.
- Using FSCREEN.
- Generating a Table of Contents.
- Generating an Index.
- Creating your own include and use files.

CREATING SYNTAX

To create syntax statements, you will be using the delite and delitei fonts. If you need large braces, brackets, parentheses, or shaded delimiters to indicate positional parameters (refer to the conventions page) you will also have to use the parens font. Finally, you may need to use TDP's ^+ and ^- commands to do half-line spacing (see section 4 of the *TDP/3000 Reference Manual*, 36578-90001).

The symbols of the parens font allow you to build large brackets, braces, and parentheses. You use alphabetic characters in your text file which correspond to the parens font symbols (see Table C-4 in appendix C). The following shows you how to use the alphabetic characters as "place-holders" for the parens font symbols:

large brackets:

Q W = []	Q W = []	Q W = []	Q W = []	Q W = []	...
A S = []	: : = []	: : = []	: : = []	: : = []	
	A S = []	: : = []	: : = []	: : = []	
		A S = []	: : = []	: : = []	
			A S = []	: : = []	
				A S = []	

large braces:

() = { }	R T = { }	R T = { }	R T = { }	R T = { }	...
	6 5 = { }	G F = { }	. . = { }	. . = { }	
	F G = { }	T R = { }	6 5 = { }	G F = { }	
		F G = { }	. . = { }	T R = { }	
			F G = { }	. . = { }	
				F G = { }	

large parentheses:

R T = ()	R T = ()	R T = ()	R T = ()	R T = ()	...
	. . = ()	. . = ()	. . = ()	. . = ()	
	F G = ()	. . = ()	. . = ()	. . = ()	
		F G = ()	. . = ()	. . = ()	
			F G = ()	. . = ()	
				F G = ()	

Note that increasing the size of the brackets and parentheses is fairly straightforward; you simply use the colon (:) to extend brackets and the period (.) to extend parentheses. Different size braces, however, are a little more complicated:

- braces which are 2 lines high are created using the / and \ characters only;
- braces which are 3,5,... lines high are created using R and T at the tops, F and G at the bottoms, and 6 and 5 in the centers;
- braces which are 4,6,... lines high are created using R and T at the tops, F and G at the bottoms, and G together with T and F together with R to form the centers.

For example, suppose we wish to create the following syntax:

$$:FCOPY FROM \left[\begin{array}{l} \{filename\} \\ * \\ \langle empty \rangle \end{array} \right]; TO \left[\begin{array}{l} \{(dfile,kfile)\} \\ filename \\ * \\ \langle empty \rangle \end{array} \right] [;optionlist]$$

Normally we would not yet have the "finalized" version shown here. Instead, we would have written out on a piece of paper *exactly* how we want the "finalized" syntax to look. (It is very important to do this if the syntax is complicated!)

Since "lining-up" is very important, we first enter the syntax *without* any font changes -- using the alphabetic characters as "place-holders" for the actual brackets, braces, or parentheses. We will need room to insert *many* font changes later, so we use macros whenever necessary to keep the lines short.

In addition, if some of the syntax needs half-line spacing to "line-up" properly (such as the syntax following the TO in the above example), we have to choose a line to type it in on. For now, in other words, we do not worry about the half-line spacing. Later, we will use ^+ and ^- to get it properly positioned when we "final".

Hence, to create the above syntax, we first enter the following lines (if you are using TDP's editor, screen mode is strongly recommended).

```

\me=" Q R(dfile,kfile)TW"
\mf=";TO;Gfilename F[;optionlist]"
\mg=" * F"
\mh=" A F<empty> GS"
:FCOPY FROM Q RfilenameTW^me
          =B* S:^mf
          A F<empty> GS^mg
          ^mh
    
```

Note how the alphabetic characters are used as "place-holders" and how everything is properly aligned (including the syntax within the macros). Also, note that we have temporarily typed the block of syntax following the TO on lines which are really a half-line off from where that block of syntax belongs.

At this point, it is very important to check that everything is "lined-up" properly. Later, it will be hard to tell!

Next, we *insert* the intraline commands to do the font changes and half-line spacing. If we are using the TDP editor, we first screen the appropriate lines. Then, keeping our terminal's "insert char" function on, we move from left to right on each line and *insert* the necessary commands. The resulting lines no longer look "lined-up":

```

\me=" * Q R^fd("adfile^n,^kfile^n)^fpTW^="
\mf="^fd;TO;^fp:^fd=^fpG^filename ^fpF:^fd[;^aoptionlist^n]"
\mg="^+ : T^fd* ^fpR:^="
\mh=" * ^fpA F^fe<empty> ^fpGS^="
^fd ^fpQ R^fefilename^fpTW^me
^fd:FCOPY FROM^fp:^fd=^fpG^fd* ^fp5:^mf
          ^fpA F^fe<empty> ^fpGS^mg
          ^mh
    
```

Specific Tasks

We recommend that you check the font changes and half-line spacing by mentally "processing" each line in the same order that TDP's formatter will process the lines. That is, start with the first line of syntax *after* the macro definitions and, going from left to right, check whether you have all the necessary intraline commands; when you come to a "call" for a macro (that is, a `^mid` command), start reading the corresponding macro definition from left to right, checking whether you have all the necessary intraline commands there; at the end of the macro definition, return to where you left off (where the macro was "called") and continue this process.

Assuming everything was properly aligned *before* we inserted the caret commands, the alignment will still be correct when our document is "finalized" -- and the end result should be beautiful!

When you use the SYNTAX and ENDBOX include files to create the syntax portion for a reference item, you are automatically in the font pair delite/delitei (fontid d) at the beginning of each line. Hence, you may need to do a few less font changes than you would otherwise need. However, you need to remember the `^ma` and `^mb` surrounding each line. Refer to SYNTAX in section 5.

For example, suppose we want to create the same syntax shown earlier, but this time using the SYNTAX include file. We would do it in the same manner just explained (first typing in the lines with characters as "place-holders" and later inserting the commands to do font changes and half-line spacing). The resulting lines in our file would look like this:

```
\**** SYNTAX ****
\m1="12"
\in syntax.in.manu
\me="^+  Q R^fd(^adfile^N,^akfile^N)^fpTW^- "
\mf="^fd;TO^+^fp:^-^fd=^+^fpG^ffilename      ^fpF:^-^fd[;^aoptionlist^N]"
\mg="^+  : T^fd*                ^fpR:^-"
\mh="^+  ^fpA F^fe<empty>        ^fpGS^-^s"
^ma          ^fpQ R^ffilename^fpTW^me^mb
^ma:FCOPY FROM^fp:^fd=^fp6^fd*      ^fp5:^mf^mb
^ma          ^fpA F^fe<empty> ^fpGS^mg^mb
^ma          ^mh^mb
\in endbox.in.manu
```

Note that the SYNTAX include file automatically places you in image mode and the ENDBOX include file automatically reinvokes format mode.

Another method of creating syntax, where all the font commands are inserted for you, involves drawing the syntax diagram on your terminal screen and using the program FSCREEN to translate the displayed drawing into a TDP file (see Using FSCREEN.TOOLS.MANU in this section).

GENERATING TABLES

To generate a table, you use a specialized set of include files in the TABLE group (rather than the IN group) of the MANU account. We strongly recommend using the use files TAB2COL, TAB3COL, and TAB4COL since the "syntax" is somewhat lengthy (refer to TAB2COL, TAB3COL, and TAB4COL in section 5). These three use files are in the USE group of the MANU account. After the use file has completed, you may need to modify the commands that were inserted by the use file in order to adjust the appearance of your table.

This section discusses the following tasks:

- using the table use files;
- changing the widths of the columns;
- changing the formatting mode of a particular entry;
- changing the length of the vertical lines of a particular row;
- adding or deleting a horizontal line between rows;
- continuing a table over a page break.

NOTE

You should have read and understood Using Multiple Columns in section 3 of this manual before proceeding.

Creating a Table

Before you create a table, make the following decisions:

- Decide how many columns you will need for your table; you have a choice of 2, 3, or 4 columns. You will use the corresponding use file TAB2COL, TAB3COL, or TAB4COL.
- Decide how wide to make the columns. You may want to refer to the "em" and line scale in appendix C to help you estimate the number of "ems". Refer to Using Multiple Columns in section 3 for an explanation of "ems".
- Decide what the columns headings and the title of your table will be.

After you have made these decisions, run TDP/3000 and use the appropriate table use file. All three use files provide (similar) detailed instructions. Read these instructions carefully to understand how to answer the prompts.

You should start with an empty workfile -- otherwise you can quickly run out of room. After the table is completely edited, you can join it to your document file.

Specific Tasks

The following is a summary of how Table 2-1 (at the end of section 2) was created. Look at that table before reading this summary. The session is reproduced here so that you can study the use file instructions at your leisure. In addition, further explanations are inserted where appropriate.

```
/use tab2col.use.manu
```

```
Place at specific line (Y) or end of workfile (N): n
```

```
Enter the title of your table: Table 2-1. MANU Formatting Functions
```

You will see a column command displayed on the screen. Modify the 'a' and 'b' to be the widths you want for your columns of text.

Note--the FIRST column is a 'fake' column which prints a blank space of 1 'em'. If you want to increase the amount of blank space on the LEFT side of your table, simply modify the width of the FIRST column.

Also, if you want a certain number of 'ems' of blank space between your table and the RIGHT margin, the total of ALL the columns must be equal to 56 minus that number.

Make sure that the total of all the columns is 56 or less.

$$(1+2+a+2+b+1 \leq 56)$$

```
1.02 \m2="\column (1,2,a,2,b,1)"
```

```
Changes: di23
```

```
1.02 \m2="\column (1,2,23,2,b,1)"
```

```
Changes: di20
```

```
1.02 \m2="\column (1,2,23,2,20,1)"
```

```
Changes: r4
```

```
1.02 \m2="\column (4,2,23,2,20,1)"
```

```
Changes: RETURN
```

In this case, we chose 23 for the first column of text and 20 for the second. Notice that we also changed the first "fake" column to 4. The total of all the columns add up to 52, leaving 4 "ems" on the right side of the table. The result is a table which is centered on the page -- with 4 "ems" of white space on each side. The other columns in the column command (2, 2, and 1) are for the boxdraw font commands which draw vertical lines between the columns and for the sides of the table.

This is a picture of the columns we specified with the column widths:

56 "ems" total						
4	2	23	2	20	1	4
\next;format 1st entry, 1st row		\next;format 2nd entry, 1st row				
\next;format 1st entry, 2nd row		\next;format 2nd entry, 2nd row				
.		.				
.		.				
.		.				
\next;format 1st entry, last row		\next;format 2nd entry, last row				

Figure 4-1. Column Widths Example

Returning to the usefile session, we see the next screen of instructions:

At the START of each row of text you will be asked to guess a number between 1 and 20. This number becomes the 'N' of the commands `\in verN` and `\in endverN`, which determine the length of the vertical lines enclosing the entries of each row.

If you guess wrong, don't worry!

All the text of each entry will be printed, even if some rows have vertical lines that are too short or too long.

After you have a final of the new table, you can edit the length of the vertical lines for each row simply by changing the number 'N' of the `\in` commands to match the number of lines of the longest entry.

At the END of each row of text you will be asked if you want the row to have a horizontal (MID) line below it, or if you want it to be without (SANS) a line. These commands can also be edited later (if you change your mind...).

After we have thoroughly absorbed these instructions, we can start entering the text of our table. We will enter the entries from left to right for each row. In the case of a two-column table, this means we enter the first entry of the first row, the second entry of the first row, the first entry of the second row, the second entry of the second row, and so on (refer to Figure 4-1, above).

Guess the number of lines of this row? (1 to 20) 1

Add first entry. Enter '///' when done:

Specific Tasks

1.1 ^ftFormatting Functions^s

1.11 //

Add second entry. Enter '//' when done:

1.13 ^ftUse Files (see section 5)^s

1.14 //

More rows? (Y/N) y

Do you want a horizontal line below this row? (Y/N) y

Guess the number of lines of this row? (1 to 20) 2

Add first entry. Enter '//' when done:

1.24 Starting any file

1.25 Finaling multiple files

1.26 //

Add second entry. Enter '//' when done:

1.28 ONESTART

1.29 MANSTART

1.3 //

More rows? (Y/N) y

.
. .
.

Notice that the entries of the first row are actually the headings of the columns and that we included the font commands to print the headings in romanb. We continue to enter more entries until we complete all the rows of our table. When we are finished, we answer no to the More rows? (Y/N) prompt and we see the final screen of instructions:

More rows? (Y/N) n

TAB2COL.USE.MANU is finished. You can now final (and edit) your new table. Remember, your file must start with ONESTART!

***** STEPS FOR EDITING *****

- 1st) If you want any entries in image or center mode, change the \next;format command on the line before the text of those entries. Then final your table and correct any truncations; final again...
- 2nd) If the vertical lines for a particular row are too long or too short, change the 'N' of the \in commands for that row. The 'N' should be equal to the number of lines of text of the longest entry in that row. Then final...

3rd) When your entries look correct, you are ready to put in page breaks. Decide which row should be the last one on a page, then change the `\in MID[SANS]2col` command following the END OF ROW comment to an `\in END2col` command. Add a copy of the commands from the start of your table (`\m1`, `\m2`, and `\in TAB2col`). If you have column headings, copy the first row as well. Final to test your page breaks...

/k tabsec2.doc.manu

The following is a partial listing of our newly-created table:

```

\**** TAB2COL ****
\m1="Table ^#s1. MANU Formatting Functions"
\m2="\column (4,2,23,2,20,1)"
\in TAB2col.table.manu
\* Copy the above 4 lines for each page break *
\*
\* START OF ROW
\*
\in ver1.table.manu
\next;format
^ftFormatting Functions^s
\in ver1.table.manu
\next;format
^ftUse Files (see section 5)^s
\in endver1.table.manu
\*
\* END OF ROW
\*
\in MID2col.table.manu
\*
\* START OF ROW
\*
\in ver2.table.manu
\next;format
Starting any file
Finaling multiple files
\in ver2.table.manu
\next;format
ONESTART
MANSTART
\in endver2.table.manu
\*
\* END OF ROW
\*
\in MID2col.table.manu

```

Notice the table title and the column command.

The first row of the table -- with the column headings in romanb.

Notice that the number we guessed is entered in triplicate -- as the 'N' for each row.

Specific Tasks

```
\in MID2col.table.manu
/*
/* START OF ROW
/*
\in ver1.table.manu
\next;format
Memos
\in ver1.table.manu
\next;format
MEMO
\in endver1.table.manu
/*
/* END OF ROW
/*
\in END2col.table.manu
```

The last row of the table -- the 17th.

Editing a Table

Following the STEPS FOR EDITING given in the use file instructions, we now add ONESTART to the start of our file so that we will be able to test our new table. Before "finaling", however, we first correct the formatting modes for any entries we do not want in format mode. In this example, we want the column headings centered, so we change the \next;format commands for the first row to \next:center. We want the entries in all the other rows to be in image mode, so we change all the other \next;format commands to \next:image. Our workfile now looks like this:

```
**** ONESTART ****
\m1="tabsec2.doc.manu"
\m2="yes"
\if main in onestart.in.manu
**** TAB2COL ****
\m1="Table ^#s1. MANU Formatting Functions"
\m2="\column (4,2,23,2,20,1)"
\in TAB2col.table.manu
/* Copy the above 4 lines for each page break *
/*
/* START OF ROW
/*
\in ver1.table.manu
\next:center
^ftFormatting Functions^s
\in ver1.table.manu
\next:center
^ftUse Files (see section 5)^s
\in endver1.table.manu
/*
/* END OF ROW
/*
\in MID2col.table.manu
/*
/* START OF ROW
/*
\in ver2.table.manu
\next:image
```



```

Starting any file
Finaling multiple files
\in ver2.table.manu
\next, image
ONESTART
MANSTART
\in endver2.table.manu
/*
/* END OF ROW
/*
\in MID2col.table.manu
.
.
\in MID2col.table.manu
/*
/* START OF ROW
/*
\in ver1.table.manu
\next, image
Memos
\in ver1.table.manu
\next, image
MEMO
\in endver1.table.manu
/*
/* END OF ROW
/*
\in END2col.table.manu

```

After "finaling" this version of our table, we are ready for the second step in editing, correcting the length of the vertical lines for each row of our table. We decide to put more white space in the column headings. To accomplish this, we change the `ver1` and `endver1` include commands for the first row to `ver3` and `endver3` and add a blank line above each heading. We do *not* need to add blank lines after the headings since `ver3` and `endver3` will force the third line.

At this point, if the vertical lines for a particular row were too long or too short, we would change the `vern` and `endvern` include commands for that row; the n should be equal to the number of lines of formatted text of the longest entry of that row. We examine the "finalized" version of the table; we can see, *for each row*, which entry is the longest in the row. We then count the number of lines of that entry; all n 's of that row should match that number. In this example, our "guesses" for the vertical lines of each row were not really guesses; we knew exactly how many lines to specify because the entries are in image mode. In the case of format mode entries, however, it is not so easy to guess correctly the first time.

After "finaling" once more, we are ready to put in the page breaks, the last step in editing a table. With the vertical lines (n) for each row correct, we can now see just how many rows of the table will fit on one page. We decide that the 12th row should be the last row on the first page of the table, so we change the `\in MID2col` command which follows that row to an `\in END2col` command. Next, we copy the commands at the start of our table as well as the first row (the column headings) to immediately after that `\in END2col` command. Finally, we edit the title of the second page of the table to indicate that the table is continued.

Specific Tasks

After all these changes, our workfile looks like this:

```
\**** ONESTART ****
\m1="tabsec2.doc.manu"
\m2="yes"
\if main in onestart.in.manu
\**** TAB2COL ****
\m1="Table 4-1. MANU Formatting Functions"
\m2="\column (4,2,23,2,20,1)"
\in TAB2col.table.manu
/* Copy the above 4 lines for each page break *
/*
/* START OF ROW
/*
\in ver3.table.manu
\next:center
an added blank line
^ftFormatting Functions^s
\in ver3.table.manu
\next:center
an added blank line
^ftUse Files (see section 5)^s
\in endver3.table.manu
/*
/* END OF ROW
/*
\in MID2col.table.manu
/*
/* START OF ROW
/*
\in ver2.table.manu
\next:image
Starting any file
Finaling multiple files
\in ver2.table.manu
\next:image
ONESTART
MANSTART
\in endver2.table.manu
/*
/* END OF ROW
/*
\in MID2col.table.manu
.
.
.
\in MID2col.table.manu
/*
/* START OF ROW
/*
\in ver9.table.manu
\next:image
Cover
Cover update page
```

The first row (the column headings) with all the editing changes complete.

The 12th row of our table, after editing, is the last row on the first page of Table 2-1.

```

Title page
Copyright page
Printing history
List of effective pages
Preface
Conventions
The entire front matter
\in ver9.table.manu
\next;image
COVER
COVERUPD
TITLE
COPYRITE
HISTORY
EFFPAGES
PREFACE
CONVEN
FRONTMAT
\in endver9.table.manu
/*
/* END OF ROW
/*
\in END2col.table.manu
\new
**** TAB2COL ****
\m1="Table 4-1. MANU Formatting Functions (continued)"
\m2="\column (4,2,23,2,20,1)"
\in TAB2col.table.manu
/* Copy the above 4 lines for each page break *
/*
/* START OF ROW
/*
\in ver3.table.manu
\next:center

^ftFormatting Functions^s
\in ver3.table.manu
\next:center

^ftUse Files (see section 5)^s
\in endver3.table.manu
/*
/* END OF ROW
/*
\in MID2col.table.manu
/*
/* START OF ROW
/*
\in ver1.table.manu
\next;image
Internal cover
\in ver1.table.manu
\next;image
LABCOV

```

The second page of Table 2-1 starts here.

Notice that we copied both the TAB2COL command lines *and* the edited first row from the first page of our table.

Specific Tasks

```
\in endver1.table.manu
/*
/* END OF ROW
/*
\in MID2col.table.manu
.
.
\in MID2col.table.manu
/*
/* START OF ROW
/*
\in ver1.table.manu
\next;image
Memos
\in ver1.table.manu
\next;image
MEMO
\in endver1.table.manu
/*
/* END OF ROW
/*
\in END2col.table.manu
```

This is the last row of the second page
of Table 2-1 -- the 18th row.

Notice that we inserted a `\NEW` command before the second page of our table. The table include files do *not* force page breaks or add blank lines before the table.

We "final" our table one last time to make sure everything is correct. Once it's gorgeous, we delete `ONESTART` and join this file to the file containing the second section of our manual (`manusec2`).

INCLUDING HPDRAW FIGURES

HPDRAW/3000 is an easy-to-use, menu-driven office product which allows you to create complex drawings using lines, arcs, circles, and figures. These can be printed as part of a document by means of the TDP command:

```
\ILLUST filename:figurename n
```

To prepare an HPDRAW drawing for TDP/HP2680 printing, you need to do the following in HPDRAW:

- Create a drawing.
- Go to the Save Menu.
- Save the drawing as a drawing file *and* as a figure in a figure file (**Save Fig**).

Additional comments:

- We strongly suggest that you give the figure the same name as the drawing file.
- Remember the *figurefilename* and the *figurename*.
- All the figures for a particular document can be saved in the *same* figure file.
- *Only* the drawing file can be edited or plotted.

Some points to consider *while creating a drawing* in HPDRAW for later inclusion in a TDP file:

The Create Drawing Menu of HPDRAW gives you a choice of a Horizontal or a Vertical orientation for each new drawing. Optimally the orientation of the drawing should match that of the environment you intend to use for your document. It's not necessary, but it may give better results appearance-wise. This is because TDP uses its own algorithm to scale the drawing to fit into the number of lines specified in the '*n*' of your \ILLUST command. If scaling the drawing to fit would result in a strange-looking drawing, TDP's formatter prints the drawing at a size deemed appropriate, and outputs blank lines for the remainder of the lines specified. However, combining a horizontal drawing with a "vertical" page (MANU90) may still result in distorted letters and figures within your drawing. Even when the drawing orientation matches that of the environment, it may take several "finals" to find a number of lines '*n*' that result in a pleasing size.

The HP2680 Laser Printer line resolution is very good for text. For drawings, the resolution is acceptable for review copies or internal documents. However, if you intend to send the document to a printer, *and* if you want high-quality figures, be sure to have saved your drawing files. When your document is ready for the printer, have the drawings done on a plotter, and then pasted in (possibly photo-reduced) for the final reproduction copy (see figure 4-2). This also avoids any distortion due to the scaling factor.

If you decide that you do want to use plotted drawings for your final "repro", when you create your HPDRAW drawing, use an 8 point or larger size font to ensure "readable plots". You may also want to decide on pens. The TDP/HPDRAW interface does not provide bold lines, but plotted drawings can have a variety of line weights, depending on the pens available. A simple example: use pen 1 for bold lines (such as box outlines or large elements), and pen 2 for text and small elements. Since you have more options with the plotter, you will want to decide on how to take advantage of them *before* creating your drawing.

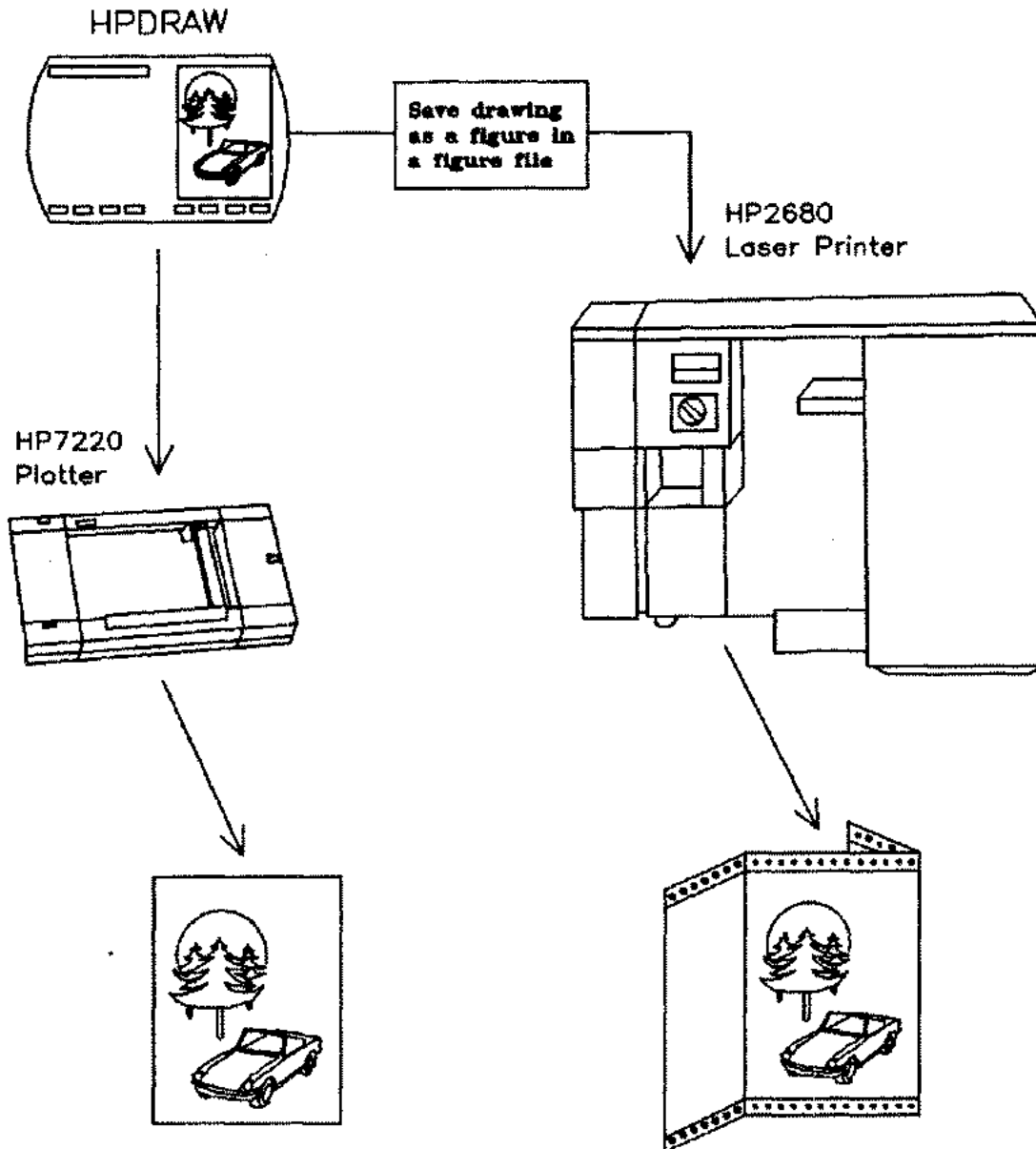


Figure 4-2. Two Methods for Including an HPDRAW Figure.

Drawings are saved in vector format, but the HP2680 only accepts files in raster format. When an HPDRAW file is "finalized" in a TDP text file, TDP's formatter calls a program to do a vector to raster conversion. This conversion is very time-consuming. The `\NAME RASTER` command can save time, if, after all your drawings are *finished* (no more changes), you issue the command `\NAME RASTER rast000` before the first `\ILLUST` command of your document. (Alternately, you can name each raster file separately, then substitute the *rasterfilename* in place of the *filename:figure* of the `\ILLUST` command.) The converted raster file for each drawing will be saved and used instead of reconverting the drawing for each "final". Result -- faster "finals".

Refer to the *HPDRAW/3000 Reference Guide* (32108-90001). Also refer to `\ILLUST` and `\NAME` in Section 4 of the *TDP/3000 Reference Manual* (36578-90001).

USING FSCREEN.TOOLS.MANU

FSCREEN is a program in the TOOLS group of the MANU account. It allows you to include in your document an image copied from your terminal screen or from a VPLUS/3000 form.

Your terminal can display a variety of information, ranging from the output of a program to complex drawings created using the terminal's line drawing set. It would be tedious to recopy all that information into a TDP file and difficult to "line-up" the drawings with all the necessary intraline font commands. Instead, you can use FSCREEN to translate whatever you have displayed on your terminal into a TDP file with the appropriate commands added for you.

HP's menu-driven products use VPLUS/3000 form files to store the screens that you see displayed on your terminal when using those products. Rather than trying to copy each screen (form) while using the product, you can use FSCREEN to access the form file directly. FSCREEN displays the VPLUS/3000 form on your terminal screen for you to edit as desired, then converts the finished version into a TDP file, ready to be printed.

FSCREEN can also add a border to the screen image or copy a 262X terminal's function key labels. If you later want to edit the FSCREEN-generated TDP file, you do not even need to look at the text file. You simply use FSCREEN to reverse the translation process and redisplay your original screen, ready for editing.

You include an FSCREEN-generated file in your document by referencing the file with an `\IN` command; the entire screen image is inserted into your document at that point.

CAUTION

FSCREEN should not be used at baud rates greater than 4800.

The RUN Command

You issue the RUN command for FSCREEN in response to MPE's colon prompt. Unfortunately, you cannot issue the FSCREEN RUN command from within TDP. You will need to specify a parameter on the RUN command to indicate what action you want FSCREEN to take.

```

RUN FSCREEN.TOOLS.MANU;INFO={
    "T[o]=output-file"
    "F[rom]=input-file"
    "V[plus][=form-file]"
    "?"
}

```

Two of the parameters, "To" and "From", are used to translate information displayed on your terminal screen to or from a TDP file. "Vplus" is used when converting VPLUS/3000 forms into TDP files, and will be discussed later. The parameter "?" prints a summary explanation of FSCREEN.

Translating Terminal Displays

The "To" parameter is used to copy the information displayed on your terminal screen to a TDP file. You must supply the name of the file you want to write to. (No file equations are allowed.) The file is automatically created by FSCREEN. If you specify a file that already exists, you are prompted Purge old file (N/Y)?.

The "From" parameter is used to copy the FSCREEN-generated TDP file back to the terminal screen for editing. Again, you supply the name of the file when you issue the RUN command. FSCREEN then displays the original image on your terminal screen, translating any TDP intraline font commands into the appropriate display enhancements.

The Begin and End Lines

You use special comment lines to mark the beginning and ending of the information to be copied from your terminal screen. These are described below.

```

\*>>ΔBEGIN [-B[order]
             . [-F[unctionkeylabels]
             . [-S[yntax]
             .
screencontents
             .
             .
\*>>ΔEND
    
```

The BEGIN line is placed on an otherwise blank line above the information to be copied; the END line is similarly placed below the end of the information. Both lines *must* be entered in all caps and *must* start in the first column of the terminal screen. FSCREEN will copy everything between the BEGIN and the END lines, up to a maximum of 119 lines. (Your terminal may not have this much memory available.) The FSCREEN-generated file has a maximum line length of 168 characters. The options on the BEGIN line are as follows:

- B[order] The intraline font commands needed to box the information (using the linedraw font) are added to each line when the screen contents are copied. The border is not displayed on the terminal screen.
- F[unctionkeylabels] The programmatically-defined function key labels are copied along with the screen contents. These labels are displayed when you use the "From" parameter to redisplay information copied from a terminal screen. Also, the border option is assumed when you specify this option.
- S[yntax] FSCREEN translates the terminal's escape and control sequences for inverse video and for the line drawing set into the TDP intraline font commands for delitei and for parens, respectively (see Advanced Capabilities later in this section).

Running FSCREEN

You use FSCREEN to copy information displayed on your terminal screen. This information may be already on your screen, such as a program example or listing you want to copy, or you may be entering information to be copied. Note, however, that if you do not want whatever is currently on your screen, you need to clear the screen *before* you enter the information to be copied. In this way, you make sure nothing extraneous will be included. You can also exclude information, such as the RUN FSCREEN command, by carefully positioning the BEGIN and END lines to only include the information you want copied. You will need to put your terminal into LOCAL mode to edit the information displayed on your terminal screen. To do that, you need to identify the control keys for the type of terminal you are using--HP264X or HP262X.



Figure 4-3. MODES Keys for an HP262X Terminal

The function key labels shown above are displayed on an HP262x terminal when you press the **MODES** key. The appropriate function key for each terminal control key is indicated by the corresponding screen label. When the terminal is in REMOTE mode, the **REMOTE** label has an asterisk *. Press the function key that corresponds to the **REMOTE** label. This will cause the asterisk to disappear; your terminal is now in LOCAL mode.

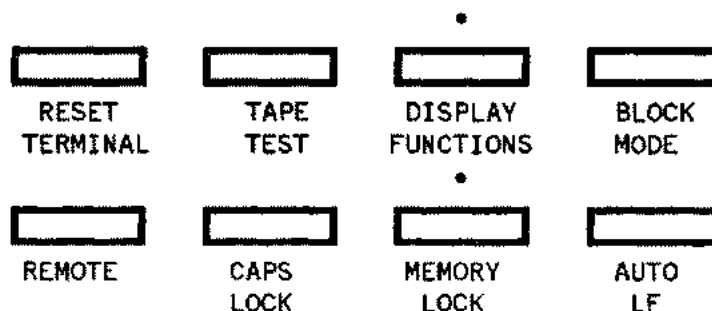


Figure 4-4. Control Keys for an HP264X Terminal

On the HP264x terminals, the terminal control keys are on the keyboard. The terminal is in LOCAL mode when the REMOTE key is in the up position; you will need to press the REMOTE key, then release it.

Now that you're in LOCAL mode, you can use the terminal's editing and cursor keys when editing information on the terminal screen. You may want to add enhancements, such as underlining for user input or inverse video for highlighting. (See Advanced Capabilities, later in this section, for a discussion on how to add and modify your terminal's display enhancements.) When you have finished editing or entering information on the screen, put your terminal back into REMOTE mode, and issue the RUN command with the "To=output-file" parameter. The RUN command can be either above the BEGIN line or below the END line. FSCREEN will copy everything on the terminal screen between the BEGIN and the END lines to the *output-file*.

The FSCREEN-generated file is now ready to be "finalized", either alone or as part of your document. You do not need to add any TDP commands, not even ONESTART, since FSCREEN automatically adds the appropriate TDP commands to define the environment file. When you want to use the

Specific Tasks

FSCREEN-generated file in your document, simply use an `\IN filename` command embedded in your document where *filename* is that of the FSCREEN-generated TDP file. The file can be printed as part of a document using either the 8.5 x 11 inch or the 9 x 8.5 inch format.

Do *not* join the FSCREEN-generated file to your document file. This is because FSCREEN builds files with a line length of greater than 80 characters; you would not be able to use the resulting file with TDP's screen mode or with HPSLATE.

You will find it is convenient to keep the FSCREEN-generated file as a separate file because you can edit your file using FSCREEN's "From" parameter. Actually, if you edit the FSCREEN-generated text file using any method other than FSCREEN, you are taking a risk. The file may not print correctly afterwards and FSCREEN may not be able to redisplay the file.

A Simple Example

To understand how FSCREEN can be used, assume that you want to include an error message in your document. You have already run the program, and the message is now displayed on your terminal screen:

```
Enter name of source file: wscntd1
**** STRING INDEX EXCEEDS CURRENT LENGTH (PASCERR 651)
***      STACK DISPLAY      ***

                S=002457    DL=177644    Z=012236
            Q=002463 P=013225 LCST= S223  STAT=U,1,1,L,0,0,CCE  X=000006

END OF PROGRAM
:
```

Figure 4-5. Error Message Example

In order to prepare the information for FSCREEN to copy, you first put your terminal into LOCAL mode. Next, you position the cursor on the line above the information to be copied and insert a line containing the BEGIN line, observing exactly the syntax shown earlier. You can use the terminal's editing keys to insert or delete lines and characters. When the text on the screen looks exactly as you want it to look, add the END line just after the last line you want copied.

You now have the following displayed on your terminal:

```

Enter name of source file: wscntd1

\*>> BEGIN
**** STRING INDEX EXCEEDS CURRENT LENGTH (PASCERR 651)
      ***      STACK DISPLAY      ***

                S=002457   DL=177644   Z=012236
      Q=002463 P=013225 LCST= S223 STAT=U,1,1,L,0,0,CCE   X=000006
\*>> END

END OF PROGRAM
:

```

Figure 4-6. The Edited Error Message

In this example, you ensured that the program prompt and the END OF PROGRAM statement will not be copied by placing the BEGIN and END lines appropriately. Remember that the BEGIN and END lines must start in the first column of your terminal screen and be entered in all caps. Move the cursor below the END line and put your terminal back into REMOTE mode. Issue the RUN command with the parameter "To=error651". FSCREEN will create a TDP file (if necessary), copy the information between the BEGIN and the END commands to the file, and add the commands necessary to allow you to "final" the file alone or as part of a document.

Here is a listing of the FSCREEN-generated file for the error message example:

```

HP36578A.03.00 TDP/3000 EDITOR (C) Hewlett Packard Co. 1980
      MON, SEP 5, 1983, 8:37 AM (DAY #248)
/t_error651
** Texting unnumbered **

/1 all
1  \*>> DATE: MON, SEP 5, 1983, 8:06 AM
2  \if not main skip 10
3  \environment manu90.env.manu
4  \if *hp2680 lft 1;rht 56;skip 1
5  \lft 1;rht 82 Notice that the environment, the margins,
6  \fontid d "delite" and a subset of the fonts are defined.
7  \fontid e "delitei"
8  \fontid g "deliteg"
9  \fontid l "linedraw"
10 \fontid p "parens"

```

Specific Tasks

```
11  \ma="^fd  ";mb=" "  
12  \skip 2  
13  \column 1  
14  \informat 0  
15  \m0="^fd^^ch^fg"  
16  \image 6                This is the error message, copied  
17  \if not main need 6    directly from the terminal screen.  
18  \*>> BEGIN  
19  ^fd**** STRING INDEX EXCEEDS CURRENT LENGTH (PASCERR 651)  
20      ***      STACK DISPLAY      ***  
21  
22  
23      S=002457      DL=177644      Z=012236  
24      Q=002463 P=013225 LCST= S223 STAT=U,1,1,L,0,0,CC CE X=0  
00006  
25  ^s  
26  \*>> END FSCREEN A.02.01  
/
```

Notice all the TDP commands added by FSCREEN. The first 17 lines define the format to be used when printing the file to the HP2680. The \IF NOT MAIN commands (lines 2 and 17) allow the file to be "finalized" alone or as part of a document which starts with ONESTART (or MANSTART). The FSCREEN-generated file will always print using MANU90 when "finalized" alone; if you include it in your document, however, it will print using MANU90 or MANU0, depending on what environment file your document uses. In either case, the information copied from the terminal is printed in image mode, correctly "lined-up" and in the font delite.

If you now want to edit the error message, you issue the RUN command with the parameter "From=error651". FSCREEN reads the file specified (error651), translates the TDP commands as appropriate, and displays the error message on your terminal screen. The only TDP commands displayed are comment lines. Put your terminal into LOCAL mode and make your editing changes. When you have finished, run FSCREEN again, using the "To" parameter to copy the results to a TDP file. You will be prompted Purge old file (N/Y)? as described above.

Advanced Capabilities

Many HP terminals have display enhancement options installed -- including inverse video, underlining and several character sets. If your terminal does, you can access these features by entering the specific terminal escape or control key sequences which correspond to the desired display enhancement. FSCREEN can translate some of these corresponding terminal escape and control sequences into the appropriate TDP intraline font commands. Once you know which key sequences to use, you can use the display enhancements in many ways, including highlighting parts of an example, underlining user-input, and creating drawings or syntax diagrams with the terminal line drawing set.

Identifying Terminal Enhancements

The easiest way to find out what display enhancements have already been installed in your terminal is to do a terminal test. Press the test key in the terminal control group of your terminal. For a HP264X terminal, the test key is on the keyboard. For an HP262X, the function key for terminal test is available in the MODES set (see Figures 4-3 and 4-4, shown earlier).

Specific Tasks

Inverse video and underlining do not need to be designated to be used. You merely turn them on and off with their assigned escape sequences on a character by character basis. For example, if you want inverse video, press the escape sequence `(ESCAPE)&dJ` with the cursor on the first character you wish to highlight; then press `(ESCAPE)&d@` with the cursor on the character immediately after the last character you want highlighted.

Your terminal may also support half/full-bright and blinking enhancements. These extra enhancements are not recognized by FSCREEN.

Table 4-1. Display Enhancements

Keys	Function	Font
<code>(ESCAPE)B</code>	Designates the Line Drawing Set as the Terminal's Alternate Set	--
<code>(CONTROL)N</code>	Changes Alphabetic Characters to Linedraw Symbols	linedraw ^f1 or parens ^fp
<code>(CONTROL)O</code>	Changes Linedraw Symbols back to Alphabetic Characters	delite ^fd
<code>(ESCAPE)&dJ</code>	Turns on Inverse Video	deliteg ^fg or delitei ^fe
<code>(ESCAPE)&d@</code>	Turns off Enhancements (Inverse video and Underlining)	delite ^fd
<code>(ESCAPE)&dD</code>	Underlining	underline ^u/^s

NOTE

Many types of terminals have these and other special escape sequences programmed into the function keys for easy use. Refer to the Terminal Reference Manual for your particular terminal.

Drawings

A common method of creating drawings is to use the special characters of the alphabet ('|', '->', '<<', and others) to emulate lines, corners and arrows. The drawing is in image mode and a monospaced font, such as delite. For example:

```

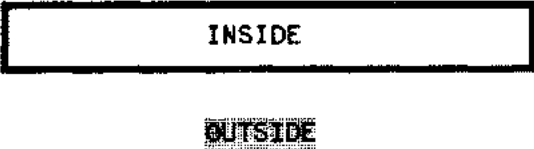
\image
^fd  +---+---+---+---+
      | 0 | 0 | 1 | <Text> |
      +---+---+---+---+
      | 1 | 0 | 0 | <Text> |
      +---+---+---+---+^s
\format

```

The advantage of this method is that it is easy to create and edit the text file, since the drawing looks essentially the same in the text file as it does in the print. Unfortunately, the printed results may be clear, but their appearance is not equal to the time and care lavished on them.

Another method is to use the linedraw font to create drawings with actual lines, arrows, and corners. The problem is that your text file does not look anything like your drawing, since the intraline font commands greatly complicate your text file, and make editing your drawing a difficult task. (See Creating Syntax in this section.)

You can avoid problems with "lining-up" your drawing by using FSCREEN to copy the drawing directly from the terminal display. Like the first method, you see exactly what you wish to draw on the terminal screen--only in this case you use the terminal's line drawing set. For example:



This looks correct; you can visualize easily; editing and "lining-up" are obvious since lines look like lines. The actual escape and control sequences do not show on your screen; all you see is the effect they have on the characters. FSCREEN will find these hidden escape and control sequences and change them into the TDP font commands for delite, linedraw, and deliteg (refer to Table 4-1). However, if you could see the terminal escape and control sequences, your drawing would become:

```

(ESCAPE) ) B
(CTRL)N->Q; ::::::::::::::::::::::::::::::::::::::::::::
(CTRL)N->:   (CTRL)O->INSIDE (CTRL)N->:
(CTRL)N->A; ::::::::::::::::::::::::::::::::::::::::::::

          OUTSIDE
          ^      ^
(ESCAPE)&dJ (ESCAPE)&d@

```

Notice the arrows. These indicate exactly where the escape and control sequences were entered to produce the drawing. With the effect of the hidden characters removed, you can see the *alphabetic* characters which were entered. These characters correspond to the symbols of the terminal's line drawing set.

When FSCREEN copies the drawing, these alphabetic characters will be copied *exactly* as you entered them. The terminal's set of line drawing symbols is not as extensive as that available with the MANU linedraw font; compare the terminal's line drawing set with the linedraw font (refer to Table C-2 in appendix C). This could cause a problem. Only the UPPER case characters of the MANU linedraw font correspond to the symbols you see on your terminal screen. If your printed version does not look like the original drawing, it could be that you entered a character in lower case. All the lower case characters

Specific Tasks

(and five of the special characters) are unique to the linedraw font and can not be displayed on the terminal screen. In the example above, if you were to use a lower case q instead of an upper case Q, you would still see a bold box corner on the terminal, but the printed version would have an arrow, since that is the corresponding symbol in the linedraw font. Hence, use upper-case characters when you are drawing; only use lower case deliberately when you want a specific linedraw symbol which is not part of the terminal's line drawing set, and thus will not be displayed.

Syntax Diagrams

Just as you can use FSCREEN to copy drawings, you can use it to copy syntax diagrams drawn on your terminal screen. But wait... To create syntax diagrams, you need the fonts delite, delitei, and parens, as discussed earlier in Creating Syntax. You do not have these fonts on your terminal. However, you can still use FSCREEN to simplify the task of creating syntax.

How? By drawing the syntax diagram on your terminal screen using the symbols of the line drawing set to represent the parens symbols (braces, brackets, etc), and inverse video to represent delitei (italics) for parameters. If you compare the alphabetic characters of the parens font with those of the terminal's line drawing set (see appendix C), you will notice that they are very similar. This was done deliberately so a drawing on your terminal screen will look as much like the printed syntax diagram as possible. A few of the "place-holding" characters of the parens font do not "match" the symbols of the line drawing set. However, although they look odd in the drawing, they print correctly in the syntax diagram. After the drawing of the syntax diagram is complete, you specify the -Syntax option on the BEGIN line and issue the RUN command. FSCREEN will copy everything you have drawn on your terminal screen, with one *minor* difference. When FSCREEN translates your drawing into a TDP file, the TDP intraline font commands for the parens font are substituted for the hidden characters for the line drawing set. Similarly, inverse video is translated to the font commands for delitei. Hence, when you "final" the FSCREEN-generated file, you will see a syntax diagram, not a drawing.

As an example, let's try using this method on the same syntax diagram we created earlier (see Creating Syntax):

```
:FCOPY FROM [ { filename } ] ; TO [ { (dfile,kfile) } ] [ ;optionlist ]
           [ * ]
           [ <empty> ]
           [ { filename } ]
           [ * ]
           [ <empty> ]
```

First, we clear the screen and put our terminal into LOCAL mode. Then we draw the syntax diagram on the terminal screen, using the alphabetic characters of the parens font, *not the line drawing set*, as "place-holders" for the actual braces and brackets. Using CONTROLN, we convert the "place-holders" into line drawing symbols. Then we add inverse video to the parameters. At this point, we see the following on our terminal screen:

```
:FCOPY FROM [ { filename } ] ; TO [ { (dfile,kfile) } ] [ ;optionlist ]
           [ * ]
           [ <empty> ]
           [ { filename } ]
           [ * ]
           [ <empty> ]
```

Notice how easy it is to visualize the syntax drawing. Now we need to add the ^+ and ^- commands for half-line spacing. FSCREEN copies these commands directly into the TDP file, without translation, so you must use the "insert char" function on your terminal when entering these commands. Unfortunately, this will change the alignment of your drawing. If you want the syntax diagram to look "lined-up", you can use a null character (^0) to correct the alignment of the rest of the drawing. This null character is

not a TDP command. It is a feature included in FSCREEN, and is only to be used to "line-up" a drawing of a syntax diagram. FSCREEN ignores the null character when it copies your drawing. This is the finished drawing, ready to be copied by FSCREEN:

```
:FCOPY FROM [ [filename] ] ;TO [ (dfile,kfile) ] [ [optionlist] ]
          [ * ] [filename]
          [ <empty> ] [ * ] [ <empty> ]
```

Issue the RUN command with the To parameter, and FSCREEN will copy your drawing to a TDP file, with intraline font commands inserted where appropriate. FSCREEN even includes the macros for SYNTAX (^ma and ^mb) when it copies the drawing. You can now use an \IN command to include the finished syntax diagram in a document (see SYNTAX in section 5). The resulting lines in our file would look like this:

```
\**** SYNTAX ****
\m1="12"
\in syntax.in.manu
\in syntaxdiagramfilename
\in endbox.in.manu
```

Converting VPLUS/3000 Forms

Using FSCREEN, you can display a VPLUS/3000 (FORMSPEC) form on your terminal, make editing changes if you wish (such as filling in the screen fields with examples of user input), and then have FSCREEN convert the displayed form into a TDP file to be printed as part of your document. Before you use FSCREEN, you will need to get a copy of the VPLUS/3000 form file for your product. You will also need to know the form name for each screen displayed by your product.

Once you have the form file, and you know the form names, you issue the RUN command with the *Vplus=formfile* parameter. FSCREEN will display the Form Selection menu shown below. When you want to perform one of the operations, you press the function key which corresponds to the screen label for that operation. For example, to convert a form to a TDP file, you enter the information necessary in the fields, and press the function key which corresponds to the **Write Form** screen label. If you want to edit the form, press the function key for **Display Form**. Unfortunately, the FSCREEN screen labels are not displayed while a form is displayed, so you will have to remember which function key performs which function.

```

FSCREEN A.02.03 Form Selection
Enter form information and/or select a function key.

Form file
Form
Window

Copy function key labels?

Output file

Note

Display  Format  Format  Write  Cancel  Refresh  Help  Exit
Form    Off **  On **  Form *  **      *      FSCREEN
*These function keys can also*/only** be used while form is displayed on screen.
    
```

Fields

- Form file** Name of an existing FORMSPEC file containing the screen forms to be copied by FSCREEN.
- Form** The specific form to be copied.
- Window** Special protected field defined in FORMSPEC which can be entered by the writer.
- Copy function key labels** For HP262X terminals only -- enter any character in the field, and the function key labels will be copied.
- Output file** Specify a name for a TDP file to be created by FSCREEN. If a file already exists with the same name, it will be purged *without* asking you, so be very careful when you choose a filename.
- Note** Used to specify a note to be added, as a comment line, to the TDP file which contains the copied form.

FSCREEN also provides an on-line help screen:

FSCREEN A.02.03	Help
<p>To generate a TDP file from a form, enter the form file, form name and name of the output file in the form selection screen. If an old file exists with the same name, the old file is purged. You can also enter the contents of the window and a note to be included in the file.</p>	
<p>Press Display Form * to display the selected form on the screen. You only need to do this if you want to enter or change field values.</p>	
<p>Press ENTER ** if you want to perform field editing.</p>	
<p>Press Format Off ** if you want to overwrite protected fields, then press Format On **</p>	
<p>Press Write Form * to write the form to the file.</p>	
<p>The keys marked by * may be used when the form selection menu is displayed and when your form is displayed. Keys marked by ** may only be used when your form is displayed. For more information, see the MANU/3000 Reference Manual.</p>	
	<p>End Help</p>

Notice that when you use the **Format Off** function key you will be able to enter text into protected fields.

Once you have converted a VPLUS/3000 form to a TDP file, you can print it as part of your document by embedding an `\IN output-file` command.

GENERATING A TABLE OF CONTENTS

The include files SECTION, APPENDX, PAGEHEAD, and LEVEL1 through LEVEL3 use the TDP command \CONTENTS to specify entries to be automatically included in a table of contents. When you "final" a file with ONESTART, the entries and their respective page numbers are stored in a *temporary* file which is printed at the end of your document. This file contains the entries specified by the include files and their associated page numbers in the following format:

- The SECTION and APPENDX entries are printed as two lines, in roman bold, with the title in all caps. There are two blank lines before the entry and a single blank line following.
- The PAGEHEAD entries are printed in all caps.
- The LEVEL1 entries are printed in initial caps.
- The LEVEL2 entries are printed in initial caps and indented 4 spaces from the left margin.
- The LEVEL3 entries are printed in initial caps and indented 8 spaces.
- If subsection numbering is in effect the entries are preceded by their respective subsection numbers, and the LEVEL2 and LEVEL3 entries are not indented. (See MANSTART in section 5 for information on subsection numbering.)

When you "final" your core file, using MANSTART, a *permanent* file named CONTENTS is created for you in your own group and account.

NOTE

Sometimes you may get error messages that some of your contents entries have been truncated by x number of characters, even when you see plenty of space remaining between the end of the entry and the right margin of the contents printed at the end of your document. This is due to a bug in TDP (36578A.03.00); it is not a problem with the MANU Formatting System.

After you have the permanent CONTENTS file, you can use the use file CON to add the include file commands which print a banner box and a footing for each page of your table of contents. The following is a reproduction of the session which produced the table of contents for this manual:

/use con.use.manu

WARNING Before using this use file, you should:

- (1) Make ALL the changes you're going to make to your document files (except for the table of contents), create a CORE file containing include statements for all your files (except the contents), and pick up a FINALED copy of this CORE file from the HP2680. YOU WILL THEN HAVE A FILE NAMED CONTENTS (OR SOMETHING ELSE IF YOU RENAMED IT) IN YOUR GROUP AND ACCOUNT.
- (2) Look at the output generated by the CONTENTS file and decide where the pagebreaks should be (remember that the pages need room for the banner box). YOU SHOULD WRITE DOWN THE LINE NUMBERS OF THE LINES AFTER WHICH YOU WANT THE PAGES TO BREAK -- BECAUSE THIS USE FILE WILL ASK YOU FOR THEM. If a CONTENTS page is going to start with the entries for a new section, the line number you should write down is the line number of the \SPACE 2 command.
- (3) Make sure your work file is either empty or has stuff in it you don't care about. THIS IS BECAUSE THIS USE FILE WILL ASK YOU WHETHER YOUR WORK FILE CAN BE DELETED AND WILL TERMINATE IF YOU ANSWER NO.

Have you done all of the above things? (Y/N) y

Is it okay to delete your work file? (Y/N) y

What is the name of the file containing your contents ? contents.doc.manu

** Texting unnumbered **

8 lines deleted

Enter the roman numeral page number of the first contents page (usually roman numeral xi), using the lowercase characters x, i, v, etc. : xi

Enter date for first contents page (MON YY): OCT 83

Is there another contents page? (Y/N) y

Enter the line number of the line AFTER WHICH you want a pagebreak: 51

51 EMe with RWIDTH^E.3-18

Does this line contain a \SPACE 2 command? (Y/N) n

Enter the roman numeral page number of this contents page: xii

Enter date for this contents page (MON YY): OCT 83

Is this contents page a right-hand page (Y) or a left-hand page (N) ? n

Is there another contents page? (Y/N) y

Enter the line number of the line AFTER WHICH you want a pagebreak: 96

96 COPYRITE^E.5-15

Does this line contain a \SPACE 2 command? (Y/N) n

Enter the roman numeral page number of this contents page: xiii

Enter date for this contents page (MON YY): OCT 83

Is this contents page a right-hand page (Y)

Specific Tasks

or a left-hand page (N) ? y

Is there another contents page? (Y/N) y

Enter the line number of the line AFTER WHICH you want
a pagebreak: 140

140 \SPACE 2

Does this line contain a \SPACE 2 command? (Y/N) y

1 lines deleted

Enter the roman numeral page number of this contents page: xiv

Enter date for this contents page (MON YY): OCT 83

Is this contents page a right-hand page (Y)

or a left-hand page (N) ? n

Is there another contents page? (Y/N) n

Your contents file is now in your work file and is ready
for testing. Be sure to:

- (1) Use ONESTART to put the onestart information at the start of the file.
- (2) FINAL it to the HP2680. If the pages need changing, modify the file by moving appropriate groups of macro definitions and include file statements; note that you will have to put the \SPACE 2 command back in if you move a page break which was just before the entries for a new section. Keep FINALing it until you get the pages working nicely.
- (3) Keep the final' contents to a file named something other than CONTENTS. Do NOT keep it to CONTENTS !
- (4) Modify your CORE file to include an \IN command for your final' contents file. Then FINAL your CORE file to print your final manual with the table of contents in the right spot.

Good luck...

/k manucon.doc.manu

Keeping unnumbered

/

Notice that the finished table of contents for this manual is named manucon. With the command \IN manucon.doc.manu added to the file CORE.DOC.MANU the entire manual can be "finalled" and the table of contents prints correctly as part of the front matter (see The Core File in section 2).

The file manucon also contains the list of figures and tables and their page numbers. We used FIGTAB (see section 5) to add the banner box and footing.

GENERATING AN INDEX

The index for a document is created after a document is complete and all the page numbers are set. You will need to go through your document carefully and decide which items to include in the index, which occurrences of each item, and whether each item is a main entry or a subentry. Then, create a file of index entries by manually entering the entries and page numbers for each item or by using the TDP command \INDEX; TDP will generate a file with the entries and their corresponding page numbers for each "final" of your text file. Once you have a file of index entries, you can use the program INDEXER in the TOOLS group of the MANU account to sort and combine the entries to produce the finished index. If you already have a sorted index file, you can use the use file INDX (see section 5).

Using INDEX

If you want TDP to generate the index file, the TDP command \INDEX "string" must be embedded within your text file for *each* item you want in the index. The \INDEX command should be as close as possible to the item, but not within formatted text, since it may cause a paragraph break.

Each time you "final" a file with embedded \INDEX commands, TDP's formatter stores the index entries and their respective page numbers in a permanent file named INDEX.

NOTE

Sometimes you may get error messages that some of your index entries have been truncated by x number of characters, even when you see plenty of space remaining between the end of the entry and the right margin when you print the file INDEX. This is due to a bug in TDP (36578A.03.00); it is not a problem with the MANU Formatting System.

Once the INDEX file is created, you will be asked if you want to Purge old INDEX? when you "final" any file containing \INDEX commands. If you do not want INDEX to be overwritten (purged), answer no, and you will be prompted for a filename. If the file doesn't exist, TDP will build one for you. Alternately, you can supply a filename before you "final" by embedding the TDP command \NAME INDEX *filename* before the first \INDEX command in your document. TDP will store the index entries in a file with the *filename* supplied.

At this point, you will have a TDP file containing all of your index entries (in the same order as they occurred in the document) and their corresponding page numbers. The file is neither sorted nor in alphabetic order.

Using INDEXER.TOOLS.MANU

The program called INDEXER in the TOOLS group of the MANU account will sort and format your TDP-generated or manually-generated file of index entries. INDEXER also adds the necessary INDX and ONESTART commands. It can be run from within TDP or in response to MPE's colon prompt. When you issue the command RUN INDEXER.TOOLS.MANU you will first be given instructions on how to set up an index file in a format that the program INDEXER will accept, as follows:

INDEXER2.P46, Jerehal's Debug
LEN = 72
ALL E MUST BE.

Specific Tasks

**** PROGRAM INDEXER ****

Written by someone

Do you want instructions? y/n yes

INDEXER takes a TDP-generated index file and creates a "real" index, ready to "final" to the HP2680. The MANU Formatting System commands for ONESTART and INDX will be added for you.

To make an index file, use TDP's \INDEX "string" command within your document -- like this:

```
\INDEX "entry"  
\INDEX "entry\subentry"
```

Put the \INDEX commands right before or after the paragraph containing the entry or subentry. When you "final" your document, TDP generates a file called INDEX with one entry for each \INDEX command, as follows:

```
entry^E.page  
entry\subentry^E.page
```

INDEXER sorts these entries into a "real" index, as follows:

```
entry1, page, page, page  
entry2  
subentry1, page  
subentry2, page, page
```

**** Type any character and press return ****

s
WARNING: You cannot have a "\", "//", or "^E." within an entry.
Also, you cannot have a "//" or "^E." within a subentry.

If you are not using TDP's \INDEX commands, any editor can be used to create the input file. However, the file must be converted to an unnumbered EDIT or TDP editor file before running this program.

NOTE: You may use "//" instead of "^E." in your file.

**** Type any character and press return ****

w
After alphabetizing, this program applies the following rules:

1. Duplicate lines are deleted from the file.
2. Lines identical except for pages are combined into one line with multiple pages separated by commas.
3. If an entry has only one subentry, the result is:
entry, subentry, page, page, page

4. If the same entry occurs with different subentries, an entry line (no page) is created followed by indented lines for the subentries.
5. If the entry and/or subentry and the accumulated pages will not fit on a single line, they are written on separate lines in the "real" index file.

***** Enter s to start or q to quit ***** q

Read the instructions carefully, and you should be able to use INDXER without any problems.

The following is an example of using INDXER. Imagine you are just finishing the *WIDGET/3000 Reference Manual*. You have thoroughly indexed every possible item; numerous \INDEX commands are embedded in the files of each section of your document. All the review changes have been entered, and the page numbers are set. Just before you "final" the entire document, you add the \NAME INDEX WINDEX command to your core file, immediately after the MANSTART include command. You now have a core file which looks something like this:

```
\**** MANSTART ****
\m1="yes"
\in manstart.in.manu
\name index windex
\in widgfrnt
\in widgsec1
.
.
.
\in widgback
```

After "finaling" this core file, your group and account will have a new file called *windex*. This is your "source" file which the INDXER program will ask you for. At this point, you might want to make a backup copy of your file, just in case--in this example, the copy is called *windex1*. Issue the command `RUN INDXER.TOOLS.MANU` and you will see the following:

**** INDXER.TOOLS.MANU ****

Written by someone

Do you want instructions?n

What is the name of the file containing your index? windex1

Specify a filename for the 'final' index(do not use INDEX!): widgindx

Loading sortfile
**Done

Sorting sortfile
**Done

Specific Tasks

```
Writing unformatted index file
**Done
```

```
Formatting index file - Pass 1
**Done
```

```
Formatting index file - Pass 2
**Done
```

```
Is this an 8 1/2 by 11 manual? y
Do you want blank pages for left/right pagination? y
Enter date for index pages (MON YY): OCT 83
```

Index file is now alphabetized and formatted.

END OF PROGRAM

Notice the last three questions. INDEXER adds both the ONESTART and INDX include file commands. Hence, your formatted index file (widgindx in this example) is now ready to "final". Check where the page breaks occur. You may have to add \NEW (or \NEED) commands and "final" again until you are satisfied with the page breaks. Then change the core file such that the index for your document will be printed before the back matter. Returning to the WIDGET/3000 example, the core file is changed as follows:

```
\**** MANSTART ****
\m1="yes"
\in manstart.in.manu
\in widgfrnt
\in widgsec1
.
.
.
\in widgindx
\in widgback
```

Notice that the \NAME INDEX WINDEX command is deleted from the core file, and an include command for the index is added. The WIDGET/3000 index is now completed.

If you follow the steps described in this example, your finished index, formatted and sorted, will be correctly printed as part of your document when you "final" your core file.

CUSTOMIZED FORMATTING

There may be times when you wish to create additional include files (and perhaps corresponding use files) to perform TDP/HP2680 formatting which is not currently performed by the MANU Formatting System. We recommend that you do this whenever you have formatting tasks which are repeated *often*; in all other cases, simply "code" the formatting commands directly into the file containing your document.

For example, you may have a specific TDP/HP2680 formatting task which occurs quite often within a manual or a series of manuals. In such a case, you will probably want to create your own include files to aid you in this repetitious task.

NOTE

You should *not* modify existing files or add additional files to the MANU account.

When you create your own include files, you should keep them in your *own* group and account; to include them in a document, insert a line containing `\IN filename`. If they are going to be used by numerous people, you may want to create a special account called OURMANU (for example) and put them in an IN group in that account. Be sure that the group and account is created such that everyone has read access to all the files. Also, put a password on the account so that the files are not modified. To include a file from this account in your document, insert a line containing `\IN filename.IN.OURMANU`.

If a formatting task is similar to, but not exactly the same as, a formatting task performed by the MANU Formatting System, obtain a copy of the appropriate MANU include file from the IN group of the MANU account and make the desired changes to your copy. Keep the changed file in your group and account with the same name it had in the MANU account (for example, LEVEL1). This allows you to use the MANU use files and simply modify the include statements afterward to "unqualify" the filenames. Or, if you are not using the MANU use files, you can simply follow the "syntax" specified in section 5 for the corresponding MANU file and "unqualify" the names of the include files. For example, `\in level1` instead of `\in level1.in.manu`.

We suggest that you store to tape any "non-MANU" include or use files which you use for a particular manual *together with* the files for that manual.

This section describes each of the use files in the USE group of the MANU account. They are covered alphabetically by filename for easy reference.

The following items are given for each use file:

- a one line description of the use file;
- the syntax of the lines added to your document by the use file (these are the lines you would add yourself if you are using HPSLATE as your editor);
- an explanation of the parameters shown in the syntax;
- a discussion of the end result when you "final" your document.

Use files can only be used with the TDP editor. If you are not familiar with a particular use file, the description given will tell you what will be performed for you by the MANU formatting system. For example, the discussion of the NOTE use file explains that two blank lines are printed before the note logo and after the text of the note; thus you know not to leave blank lines before and after the note since this is all being done for you. In addition, you will find the explanation of the lines added to your document by the use file helpful if you wish to change something after you have used a use file. For example, if you have used the PARMS use file and now need to add an additional parameter and corresponding explanation, the syntax and parameter explanations shown will help you do that (there is no need to delete the lines and reuse the use file).

If you are using HPSLATE as your editor rather than the TDP editor or if you are using TDP's screen mode, you will be inserting the macro definitions and include statements manually, relying on the syntax and parameter explanations shown. In most cases, to obtain information about an include file, simply refer to the use file of the same name. For example, if you wish to insert the include file LEVEL2.IN.MANU (for a level 2 head), look up LEVEL2; this will give you the information you need since LEVEL2.USE.MANU inserts the include file LEVEL2.IN.MANU in your document. There are some include files, however, which are not identical in name to a use file. For example, if you wish to insert the include file SECTION2.IN.MANU (for a two-line section title), look up SECTION; this will give you the information you need since SECTION.USE.MANU inserts *either* SECTION1.IN.MANU *or* SECTION2.IN.MANU, depending on whether your section title is one or two lines long.

In the case of a complicated syntax or if this is your first time using the MANU Formatting System, you may want to convert your HPSLATE file to an editor file, run TDP/3000, text in your file, and use the appropriate use file; after the use file has completed, run HPSLATE/3000 and convert your file back to an HPSLATE file. This will allow you to study the lines added to your document by the use file. As you become familiar with the syntax, you will be able to add the lines yourself rather than converting your files in order to use the use files.

Appendix F contains a useful summary of MANU functions, use files, and corresponding include files. (You may want to zerox the two tables in that appendix and pin them up by your work area for easy reference.)

NOTE

Typically, syntax is read horizontally. The syntax in this section, however, represents multiple lines in your file and hence is read vertically. Because of this, the usual meaning of multiple elements stacked inside brackets (see page ix) does not apply. In this case, when the syntax contains multiple lines inside large brackets, it means the *set of lines* contained within the brackets is optional; you may include all or none of the lines. See PREFACE for an example.

In addition, because the syntax is read vertically, vertical ellipses are used instead of horizontal ellipses to indicate that a previous element may be repeated. See CAUTION for an example.

Starts an appendix.

Syntax

```

\**** APPENDX ****
\m1={ "mon yy"
      "* HP Confidential *"
      "* Review Copy *" }
\m2="head"
\m3="title"
[ \m4="titlecntd" ]
\in appendixn.in.manu

```

Parameters

- mon yy* is the print date (month and year) of the pages in the appendix. The month is represented by three characters in all caps; the year is represented by two digits. For example, DEC 82 represents December, 1982. The print date of the pages in the appendix should be the date of the edition or subsequent update in which the pages were printed.
- * Review Copy * is used for review copies. This causes the string * Review Copy * rather than the print date to appear at the bottom of each page.
- * HP Confidential * is used for internal documents. This causes the string * HP Confidential * rather than the print date to appear at the bottom of each page.
- head* is the running head of the appendix. This is usually the same as or an abbreviated version of the title of the appendix, only in *initial caps* rather than all caps.
- title* is the title of the appendix in all caps. If the appendix title will fit on one line within the appendix banner, this is the entire title; if two lines are needed, this is the first half of the title.
- titlecntd* is the continuation of the title of the appendix. This is only included if you need a two line title.
- appendx*n* is either appendix1 or appendix2, where:
- appendx1 is used when the appendix title is one line long;
 - appendx2 is used when the appendix title is two lines long.

APPENDX

Discussion

When your document is "finaled", APPENDX prints a right-hand page containing the appendix banner, prints the appendix title in the banner, leaves three blank lines, defines the running head, resets the page numbering, defines the two-line footing as the print date and the section-page (which alternate from right to left for odd and even pages), and specifies an entry for the table of contents file. You can use `^#s` to obtain the appendix number followed by a dash, as in `table ^#s2` for the second table in the current appendix.

If you are preserving left/right pagination (see `noblank` under `MANSTART`), a blank page is printed before the start of the appendix whenever the previous section or appendix was an odd number of pages.

Prints the back cover. Also see BACKMAT.

Syntax

```
\**** BACKCOV ****  
\m1="partnum"  
\m2="editiondate"  
\m3="datecode"  
\in backcov.in.manu
```

Parameters

- partnum* is the part number of the manual in the form *nnnnn-nnnnn*, where *n* represents a digit.
- editiondate* is the print date (month and year) of the latest edition of the manual in the form *mm/yy*. For example, 08/82 represents August, 1982.
- datecode* is a 5 character code in the form *Emmyy*, where *myy* is the month and year of the print date of the latest edition. For example, E0882 represents August, 1982.

Discussion

When your document is "finalized", BACKCOV prints a left-hand page containing three lines of back cover information in the lower left-hand corner and the HP logo in the lower right-hand corner.

If you are preserving left/right pagination (see *noblank* under MANSTART), a page is printed before the back cover reminding you to include the HP Sales & Support lists.

BACKMAT

Prints back matter (reader comment sheet, business reply mailer, and back cover).

Discussion

This use file simply invokes the use file RCS followed by the use file BACKCOV; refer to those for details.

Switches to "block" mode.

Syntax

```
\**** BLOCK ****  
\in block.in.manu
```

Discussion

When your document is "finaled", **BLOCK** moves the left margin in by three "ems", initiates image mode (where text is printed in exactly the format that appears in the input file), prints a blank line if not currently in image mode, and prevents the block of text which follows from being split across a page boundary. A block of text is defined as text until a blank line or backslash command is encountered.

Note that you should always use **FORMAT** when you want to reinvoke format mode after having invoked "block" mode.

BOX

Prints a box with regular or bold lines.

Syntax

```
\**** BOX ****
\m1="inrht"
\m2="inlft"
\in {box
     boxb}.in.manu
^m $line$ ^mb
^m $line$ ^mb
.
.
.
\in {endbox
     endboxb}.in.manu
```

Parameters

- inrht* is a number indicating how many "ems" of blank space you want between the right side of the box and the right margin.
- inlft* is a number indicating how many "ems" of blank space you want between the left side of the box and the left margin.
- box* is used if you want the box to have regular lines.
- boxb* is used if you want the box to have bold lines.
- line* is a line of text to be printed within the box. These lines are in image mode.
- endbox* is used to close a box with regular lines.
- endboxb* is used to close a box with bold lines.

Discussion

When your document is "finald", BOX moves the left and right margins in according to the values given and prints the lines of text in image mode in the font pair delite/delitei (fontid d), surrounded by a box. After the box is printed, the margins are returned to their original values and format mode is reinvoked.

Prints bulleted or numbered items.

Syntax

```

\**** BULLETS ****
\m1="inlft"
\in bullets.in.manu
{
  ^mc
  ^md } text
  ^#n
  \indent 4,4
  textcntd
  \indent 0,4
  [ \ ]
  {
    ^mc
    ^md } text
    ^#n
    .
    .
    .
\in endbull.in.manu

```

Parameters

inlft

is a number indicating how many "ems" of blank space you want between the bullet (or number) and the left margin.

^mc

is used if you want the items marked by a round bullet (•).

^md

is used if you want the items marked by a square bullet (■).

^#n

is used if you want the items numbered. Use 9 for *n* if you want numbers (1,2,...) which are reset to one with each use of BULLETS.

If subsection numbering is *not* in effect (see ONESTART and MANSTART for information on subsection numbering), you may number all items within a certain subsection consecutively -- *regardless* of how many times BULLETS is used. To do this, use a number for *n* which is one greater than the level of the level head at which you want the item numbers reset:

- if *n* is 1, item numbers are not reset until the next section or appendix;
- if *n* is 2, item numbers are not reset until the next level 1 head;
- if *n* is 3, item numbers are not reset until the next level 2 head;
- if *n* is 4, item numbers are not reset until the next level 3 head.

For example, to number all items within a section of your manual consecutively, use ^#1.

BULLETS

text

is the text of an item. An item can be more than one line long, but is assumed to consist of at most one paragraph. Only the *first* line of each item is preceded by `^mc`, `^md`, or `^#n`. For example:

`^mc`This is the text of a bulleted item. Each bulleted item can be one or more lines long.

In the case of numbered items (`^#n`), *text* should begin with whatever characters you want following the numbers. In order to obtain proper indentation of numbered paragraphs, you should use a period or right parenthesis followed by a space. For example, `^#9)Δ --` as in:

`^#9)` This is the text of a numbered item. Numbered items can also be one or more lines long.

`\indent 4,4`

is used immediately after the first paragraph of an item which consists of multiple paragraphs.

`textcntd`

is the second (and following) paragraphs of an item which consists of multiple paragraphs.

`\indent 0,4`

is used immediately after the last paragraph of an item which consists of multiple paragraphs.

`\`

is used if you do not want a blank line between items. In order for BULLETS to work correctly, you must leave either a blank line or a line with a backslash in column one between items. (This is because the items are in format mode.)

Discussion

When your document is "finalized", BULLETS prints one blank line, moves the left margin in according to the value given, and prints the items. The items begin with a round bullet, a square bullet, or a number, depending on whether `^mc`, `^md`, or `^#n` has been used. Each item is formatted into a paragraph. After the items are printed, a blank line is printed and the left margin is returned to its original value.

You may successfully use BULLETS within PARMs in order to obtain bulleted or numbered items within the explanation of a parameter. You may also use BULLETS within PREFACE; however, if you need a second page you must reset `\m1="non yy"` before the line `\in prefcntd.in.manu` to get the correct footing on that page.

Prints a caution.

Syntax

```
\**** CAUTION ****  
\in caution.in.manu  
text  
text  
.  
.  
.  
\in endnote.in.manu
```

Parameters

text is the text of the caution. These lines are in format mode.

Discussion

When your document is "finalized", CAUTION prints two blank lines, moves the left and right margins in by eight "ems", prints the logo for a caution (centered), leaves one blank line, and prints the text of the caution (formatted). After the caution is printed, two blank lines are printed and the left and right margins are returned to their original values. CAUTION prevents the caution logo and the start of the text of the caution from being broken across page boundaries.

You may successfully use CAUTION within PARMs in order to produce a caution within the explanation of a parameter.

CON

Prints the Contents page(s).

Syntax

```
\**** CON ****
\m1="numeral"
\m2="mon yy"
\in conbegin.in.manu
firstpage
[ \m1="numeral"
  [\m2="mon yy"
  \in conleft.in.manu
  leftpage
  ]
[ \m1="numeral"
  [\m2="mon yy"
  \in conright.in.manu
  rightpage
  ]
.
.
.
\in endcon.in.manu
```

Parameters

- numeral* is a roman-numeral page number for a Contents page. The roman-numeral is represented by the lowercase characters x, v, i, etc. The first Contents page is usually roman-numeral xi. Note that the roman-numeral page number must be supplied for each Contents page.
- mon yy* is the print date (month and year) of a Contents page. The month is represented by three characters in all caps; the year is represented by two digits. For example, DEC 82 represents December, 1982. The print date (month and year) of a Contents page should be the date of the edition or subsequent update in which it was printed. The print date need be supplied only *once* if all the Contents pages have the same print date.
- firstpage* is the text of the first page of the Contents in image mode.
- `\in conleft.in.manu` is used if a left-hand Contents page is needed.
- leftpage* is the text of a left-hand Contents page in image mode.
- `\in conright.in.manu` is used if a right-hand Contents page is needed.
- rightpage* is the text of a right-hand Contents page in image mode.

Discussion

When your document is "finaled", CON prints the Contents page(s) starting on a right-hand page following the Conventions pages.

If you are preserving left/right pagination (see noblank under MANSTART), CON prints a blank page following the Contents pages whenever there are an odd number of pages in the Contents.

Refer to Generating a Table of Contents in section 4.

CONVEN

Prints the Conventions pages. Also see FRONTMAT.

Syntax

```
\**** CONVEN ****  
\m1="mon yy"  
\in conven.in.manu
```

Parameters

mon yy

is the print date (month and year) of the Conventions pages. The month is represented by three characters in all caps; the year is represented by two digits. For example, DEC 82 represents December, 1982. The print date (month and year) of the Conventions pages should be the date of the edition or subsequent update in which they were printed.

Discussion

When your document is "finaled", CONVEN prints the Conventions pages (2 pages) starting on a right-hand page following the Preface.

Prints the Copyright page. Also see FRONTMAT.

Syntax

```
\**** COPYRITE ****  
\m1="years"  
\in copyrite.in.manu
```

Parameters

years is a list of *all* years, separated by commas, in which editions or updates were printed. For example, 1977, 1979-1981, 1983

Discussion

When your document is "finald", COPYRITE prints a left-hand page containing the copyright statement for a software manual following the Title page.

COVER

Prints the cover. Also see FRONTMAT.

Syntax

```
\**** COVER ****  
\m1="computer"  
\m2="type"  
\m3="name"  
[\m4="namecntd"]  
\in covern.in.manu
```

Parameters

- computer* is the name in initial caps of the computer systems to which the manual pertains. For example, HP 3000 Computer Systems.
- type* is the type of manual in initial caps. For example, Reference Manual.
- name* is the title of the manual, not including the type of manual. In most cases, only the product name in all caps will be used. For example, PASCAL/3000. If the title (product name) will fit on one line, this is the entire title. If the title needs two lines, this is the first part of the title of the manual. For example, IMAGE/3000 where the entire title is IMAGE/3000 DATA BASE MANAGEMENT SYSTEM.
- namecntd* is the continuation of the title of the manual. This is only included if you want a two line title.
- covern* is either *cover1* or *cover2*, where:
- | | |
|---------------|---|
| <i>cover1</i> | is used when the title of the manual is one line long. |
| <i>cover2</i> | is used when the title of the manual is two lines long. |

Discussion

When your document is "finald", COVER prints a page containing the name of the computer systems and the HP logo along the top, followed by the complete title of the manual.

If you are preserving left/right pagination (see *noblank* under MANSTART), COVER prints a blank page following the cover page.

See LABCOV for an internal document cover.

Prints the cover page for an update package. Also see FRONTMAT.

Syntax

```
\**** COVERUPD ****  
\m1="manualtitle"  
\m2="partnum"  
\m3="edition"  
\m4="updatenum"  
\m5="update"  
\m6="datecode"  
\in coverupd.in.manu
```

Parameters

- manualtitle* is the complete title of the manual. For example, Pascal/3000 Reference Manual.
- partnum* is the part number of the manual in the form *nnnnn-nnnnn*, where *n* represents a digit.
- edition* is the date of the latest edition of the manual in the form *Month yyyy*, where *Month* is the month in initial caps and *yyyy* is the year represented by four digits. For example, December 1982.
- updatenum* is the update number of the latest update, represented by one digit (for example, 2). Updates are numbered 1,2,... for each set of updates following the latest edition.
- update* is the date of the latest update in the form *Month yyyy*, where *Month* is the month in initial caps and *yyyy* is the year represented by four digits. For example, January 1983.
- datecode* is a 5 character code in the form *Ummyy*, where *mmyy* is the month and year of the print date of the latest update. For example, U0183 represents an update printed in January, 1983.

Discussion

When your document is "finald", COVERUPD prints the manual update cover page, including the title of the manual, part number, latest edition date, update number, and update date.

If you are preserving left/right pagination (see noblank under MANSTART), COVERUPD prints a blank page following the update cover.

DISCUSS

Starts the discussion portion for an item in a reference section. Also see EASYREF.

Syntax

```
\**** DISCUSS ****  
\in discuss.in.manu
```

Discussion

When your document is "finaled", DISCUSS prints two blank lines, prints the head for the discussion portion, and prints one more blank line. The text of your discussion should follow. DISCUSS prevents the head for the discussion portion and the start of the text from being broken across page boundaries.

Prints a complete reference (page head, syntax, parameters, discussion, example, and text reference) for an item in a reference section.

Discussion

This use file simply invokes the use file PAGEHEAD followed by SYNTAX, optionally followed by PARMs, DISCUSS, EXAMPLE, and TEXTREF; refer to those for details.

EFFPAGES

Prints the List of Effective Pages page(s). Also see FRONTMAT.

Syntax

```
\**** EFFPAGES ****
\m1="mon yy"
\m2={ "yes"
      "no" }
\in effpages.in.manu
firsthalf
[ \next
  secondhalf
  [ \in effcntd.in.manu ]
  firsthalf
  [ \next
    secondhalf
    \in endeff.in.manu ] ] ]
```

Parameters

mon yy is the print date (month and year) of the List of Effective Pages page(s). The month is represented by three characters in all caps; the year is represented by two digits. For example, DEC 82 represents December, 1982. The print date (month and year) of the List of Effective Pages page(s) should be the date of the edition or subsequent update in which it was printed.

yes indicates that there are more than 10 entries in the List of Effective Pages.

no indicates that there are 10 entries or less in the List of Effective Pages.

firsthalf is the first half of the entries for a List of Effective Pages page. In the case of 10 entries or less, this is the *complete* list of entries. In the case of more than 10 entries, this is the first half (the left column) of the list on the page.

Each entry is in the following form:

page(s)^e.mon yyyy

where *mon yyyy* is the date of the most recent version of the specified pages. The month is represented by three characters in initial caps; the year is represented by four digits.

For example:

1-1 to 1-15^e.Dec 1982

In the case of a new edition, you should use the following as the only entry:

`all^e.mon yyyy`

- `\next` is used to start a second column on a List of Effective Pages page.
- `secondhalf` is the second half of the entries for a List of Effective Pages page. In the case of 10 entries or less, this is not included. In the case of more than 10 entries, this is the second half (the right column) of the list on the page.
- `\in effcntd.in.manu` is used if a second List of Effective Pages page is needed.

Discussion

When your document is "finalized", EFFPAGES prints the List of Effective Pages page(s) on a right-hand page following the Printing History. If there are 10 entries or less, the entries are centered in one column on a page. If there are more than 10 entries, two columns are used; two pages may be used if necessary.

If you are preserving left/right pagination (see `noblank` under `MANSTART`), EFFPAGES prints a blank page following the List of Effective Pages whenever only one page is used.

EXAMPLE

Prints the example portion for an item in a reference section. Also see EASYREF.

Syntax

```
\**** EXAMPLE ****  
\in example.in.manu  
line  
line  
.  
.  
.  
\in endexamp.in.manu
```

Parameters

line is a line of text of the example. These lines are in image mode.

Instead of these lines, you can use a single line consisting of an IN command for a file containing an example. (This file can be a file generated with FSCREEN.TOOLS.MANU -- see section 4.)

Discussion

When your document is "finaled", EXAMPLE prints two blank lines, prints the head for the example portion, moves the left margin in by three "ems", prints one blank line, and prints the example in image mode in the font pair delite/roman (fontid q). After the example is printed, the left margin is returned to its original value, a blank line is printed, and format mode is reinvoked. EXAMPLE prevents the head for the example portion and the start of the example from being broken across page boundaries.

NOTE

The font pair delite/roman allows you to switch to the roman font for comments next to the example. Surround each comment with ^a and ^n.

Prints the Figures and Tables page(s).

Syntax

```

\**** FIGTAB ****
\m1="numeral"
\m2="mon yy"
\in figtab.in.manu
firstpage
[ \m1="numeral"
  [ \m2="mon yy"
    \in figtcntd.in.manu
  ]
  secondpage
  \in endfigt.in.manu

```

Parameters

- numeral* is a roman-numeral page number for a Figures and Tables page. The roman-numeral is represented by the lowercase characters x, v, i, etc. (The first Figures and Tables page usually follows the last page of the Contents.)
- mon yy* is the print date (month and year) of a Figures and Tables page. The month is represented by three characters in all caps; the year is represented by two digits. For example, DEC 82 represents December, 1982. The print date (month and year) of the Figures and Tables page(s) should be the date of the edition or subsequent update in which it was printed.
- firstpage* is the text of the first page of the Figures and Tables page(s). These lines are in image mode.
- `\in figtcntd.in.manu` is used if a second Figures and Tables page is needed.
- secondpage* is the text of the second page of the Figures and Tables pages. These lines are in format mode.

Discussion

When your document is "finaled", FIGTAB prints the Figures and Tables page(s) starting on a right-hand page following the Contents pages. Note that you will have to include headings for the list of Figures and for the list of Tables as part of the text of the page(s).

If you are preserving left/right pagination (see `noblank` under `MANSTART`), FIGTAB prints a blank page following the list of Figures and Tables whenever there is only one Figures and Tables page.

FORMAT

Switches to format mode.

Syntax

```
\**** FORMAT ****  
\in format.in.manu
```

Discussion

When your document is "finaled", **FORMAT** prints a blank line if not currently in format mode, sets the left and right margins to their original values, and initiates format mode with right justification.

Note that you should always use **FORMAT** to reinvoke format mode after having invoked image or "block" mode (see **IMAGE** and **BLOCK**).

FRONTMAT

Prints front matter (cover or cover update page, title page, copyright notice, printing history, list of effective pages, preface, and conventions).

Discussion

This use file simply invokes the use file COVER or COVERUPD, followed by TITLE, COPYRITE, HISTORY, EFFPAGES, PREFACE, and CONVEN; refer to those for details.

HISTORY

Prints the Printing History page. Also see FRONTMAT.

Syntax

```
\**** HISTORY ****  
\m1="mon yy"  
\in history.in.manu  
version^e.date  
^ma  
^e.softwarecode  
^mb  
version^e.date  
^ma  
^e.softwarecode  
.  
.  
.  
\in endhist.in.manu
```

Parameters

- mon yy* is the print date (month and year) of the Printing History page. The month is represented by three characters in all caps; the year is represented by two digits. For example, DEC 82 represents December, 1982. The print date (month and year) of the Printing History page should be the date of the edition or subsequent update in which it was printed.
- version* is the version of the manual in a Printing History entry. Some examples are:
- Third Edition
Update #2
Update #4, incorporated
- date* is the print date of the version of the manual in a Printing History entry. The date is in the form *mon yyyy*, where *mon* is the month represented by three characters in initial caps and *yyyy* is the year represented by four digits. For example, Dec 1982.
- softwarecode* is the software code indicating the software product level at the time the corresponding version of the manual in a Printing History entry was issued. The software code is in the form *nnnnna.nn.nn*, where *n* represents a digit and *a* represents an alphabetic character. For example, 32215B.03.00.

Discussion

When your document is "finalized", HISTORY prints a right-hand Printing History page containing an explanation and the printing history of the manual following the Copyright page.

If you are preserving left/right pagination (see noblank under MANSTART), HISTORY prints a blank page following the Printing History page.

IMAGE

Switches to image mode.

Syntax

```
\**** IMAGE ****  
\in image.in.manu
```

Discussion

When your document is "finalized", IMAGE moves the left margin in by three "ems", prints a blank line if not currently in image mode, and initiates image mode (where text is printed in exactly the format that appears in the input file).

Note that you should always use FORMAT when you want to reinvoke format mode after having invoked image mode.

Prints the Index page(s).

Syntax

```

\**** INDX ****
\m1="mon yy"
\in indx.in.manu
line
line
.
.
.
\inlft 0

```

Parameters

mon yy

is the print date (month and year) of the Index pages. The month is represented by three characters in all caps; the year is represented by two digits. For example, DEC 82 represents December, 1982. The print date (month and year) of the Index pages should be the date of the edition or subsequent update in which they were printed.

line

is a line to be printed in the Index. This will be an Index heading (a letter of the alphabet), an Index entry or subentry with associated page numbers, or a blank line. These lines are in image mode.

Discussion

When your document is "finalized", INDX prints the Index page(s) starting on a right-hand page.

If you are preserving left/right pagination (see `noblank` under `MANSTART`), INDX prints a blank page before the first Index page whenever the previous section or appendix is an odd number of pages.

The Index file can be generated with `INDXER.TOOLS.MANU` (refer to `Generating an Index` in section 4).

LABCOV

Prints an internal cover.

Syntax

```
\**** LABCOV ****  
\m1="name"  
\m2="type"  
\m3={ "yes"  
      "no" }  
\m4="locationcode"  
\m5="projectnum"  
\m6="division"  
\in labcov.in.manu  
  teammember  
  teammember  
  .  
  .  
  .  
[\illust filename:figure n]  
\in endlcov.in.manu
```

Parameters

<i>name</i>	is the name of the product in all caps. For example, PASCAL.
<i>type</i>	is the type of document in initial caps. For example, External Specifications.
<i>yes</i>	indicates that this is the final draft of the document.
<i>no</i>	indicates that this is a preliminary draft of the document.
<i>locationcode</i>	is the location code in the form <i>nn-nnnn</i> , where <i>n</i> represents a digit.
<i>projectnum</i>	is the project number in the form <i>nnnn-nnnn</i> , where <i>n</i> represents a digit.
<i>division</i>	is the name of the division in initial caps. For example, Information Networks Division.
<i>teammember</i>	is the name of a member of the project team. These lines are in center mode. For example, Wendy King.
<i>illust</i>	is a TDP <code>\ILLUSTRATION</code> command. This is used if you want to include an illustration on your cover.

Discussion

When your document is "finaled", LABCOV prints an internal cover, including the title of the document, whether it is a draft version, the location code, the project number, the print date, the division name, the name of the project team members, and a copyright line for the current year.

If you are preserving left/right pagination (see noblank under MANSTART), LABCOV prints a blank page following the internal cover page.

LEVEL1

Starts a level 1 subsection.

Syntax

```
\**** LEVEL1 ****  
\m1="contents"  
\m2="level1"  
\in {level1 } .in.manu  
    {level1x}
```

Parameters

- contents* is the level 1 head in *initial caps*. This is used to specify an entry for the table of contents file.
- level1* is the level 1 head in *all caps*.
- level1* is used if you want level 1 heads to always start on a new page.
- level1x* is used if you do *not* want level 1 heads to always start on a new page.

Discussion

When your document is "finalized", LEVEL1 either forces a new page or prints two blank lines (depending on whether *level1* or *level1x* has been used), prints the level 1 head, prints two blank lines, and specifies an entry for the table of contents file.

Starts a level 2 subsection.

Syntax

```
\**** LEVEL2 ****  
\m1="level2"  
\in level2.in.manu
```

Parameters

level2 is the level 2 head in initial caps.

Discussion

When your document is "finaled", LEVEL2 prints two blank lines, prints the level 2 head, prints another blank line, and specifies an entry for the table of contents file. LEVEL2 prevents the level 2 head and the start of the text from being broken across page boundaries.

LEVEL3

Starts a level 3 subsection.

Syntax

```
\**** LEVEL3 ****  
\m1="contents"  
[\m2="level3"]  
\in {level3  
    level3x}.in.manu
```

Parameters

- contents** is the level 3 head in *initial* caps. This is used to specify an entry for the table of contents file. It is also used for the level 3 head if you do not want run-in level 3 heads.
- level3** is the level 3 head in *all* caps. This is only included if you want run-in level 3 heads.
- level13** is used if you want level 3 heads to be run-in heads.
- level13x** is used if you do *not* want level 3 heads to be run-in.

Discussion

When your document is "finaled", LEVEL3 prints two blank lines and prints the level 3 head. If level13 has been used, the head is printed in all caps as a run-in head (followed by a period); if level13x has been used, it is printed in initial caps on a line by itself followed by one blank line. LEVEL3 also specifies the level 3 head as an entry for the table of contents file.

Starts a level 4 subsection.

Syntax

```
\**** LEVEL4 ****  
\m1="level4"  
\in level4.in.manu
```

Parameters

level4 is the level 4 head in initial caps.

Discussion

When your document is "finaled", LEVEL4 prints one blank line and prints the level 4 head as a run-in head (followed by a period). An entry is *not* specified for the table of contents file.

MANSTART

Prints your entire manual or document.

Syntax

```
\**** MANSTART ****
\m1={ "yes"
      "no" }
\in manstart.in.manu
[\in numbered.in.manu]
[\in noblank.in.manu]
\in filename
\in filename
.
.
.
```

Parameters

yes indicates that this is an 8.5 x 11 inch manual (typically a reference manual).

no indicates that this is a 9 x 8.5 inch manual (typically a user's guide).

\in numbered.in.manu is used if you want numbered subsections. When subsection numbering is in effect, the level heads are numbered hierarchically. For example, 2.3.1 would be printed along with the *first* level 2 head in the *third* level 1 subsection of the *second* section.

\in noblank.in.manu is used if you do not want extra pages printed to preserve left/right pagination. Otherwise, a page will be printed whenever necessary to preserve left/right pagination. For example, if the last page of a section is an odd page number, an extra page is printed before the first page of the next section.

These extra pages are useful in that they allow you to fold your document such that it paginates correctly. They also serve as a reminder to indicate blank pages on the pagination record; *however, the extra pages themselves should be removed before giving the reproduction package to the printer.*

filename is the name of a file containing a section of your manual or document. These files should be included in the sequence you wish them to appear. For example: the file containing the front matter, the file containing the first section, the file containing the second section, ... the file containing the last section, and the file containing the back matter.

Discussion

MANSTART builds your *core* file. Your core file is a file containing only the syntax of MANSTART. It should be built after you have completed all the sections of your manual or document.

When your core file is "finalized", MANSTART defines the environment file, font identifications, table of contents file, margins, macros, and blank character for standard formatting of your entire manual or document. Your files are printed with sections numbered 1,2,3,... and appendixes numbered A,B,C,...

More specifically, MANSTART defines the following items:

- 1) The environment file is defined as MANU90.ENV.MANU for an 8.5 x 11 inch document and as MANU0.ENV.MANU for a 9 x 8.5 inch document.
- 2) Font identifications are defined as shown in appendix A.
- 3) Left and right margins are set at 1 and 56 for "finaling" an 8.5 x 11 inch document to the laser printer, 2 and 57 for "finaling" a 9 x 8.5 inch document to the laser printer, 1 and 80 for "finaling" to a terminal, and 1 and 100 for "finaling" to a line printer.
- 4) Top and bottom margins are set at 6 and 6 for an 8.5 x 11 inch document and at 4 and 2 for a 9 x 8.5 inch document.
- 5) Page numbering is set to start at 1, centered. (This can be overwritten by SECTION or APPENDX to alternate.)
- 6) Right justification in format mode is turned on.
- 7) The table of contents file is declared to be a file named CONTENTS.
- 8) The blank character is defined to be the grave accent (`).
- 9) The section and appendix number base are set to start at 1 and A, respectively.
- 10) Macros are initialized. These include macros indicating whether this is an 8.5 x 11 inch or 9 x 8.5 inch document, whether subsection numbering is in effect, and whether extra pages to preserve left/right pagination should be printed.

CAUTION

Certain macros and automatic numbering arguments are reserved for use by the MANU formatting system (see appendix B). When defining your *own* macros and automatic numbering arguments, you should use:

me - mn
#6 - #8

MEMO

Prints a memo.

Syntax

```
\**** MEMO ****
\m1="division"
\m2="address"
\m3="phone"
\in memofrom.in.manu
fromname
fromname
.
.
\in memoto.in.manu
toname
toname
.
.
\in memosub.in.manu
[subject]
[in memocc.in.manu]
ccname
ccname
.
.
\in endmemo.in.manu
memo
```

Parameters

- division* is the name of the division in initial caps. For example, Information Networks Division.
- address* is the address of the division in initial caps. For example, 19420 Homestead Road, Cupertino, CA 95014.
- phone* is the phone number of the division. For example, (408)725-8111.
- fromname* is the name of a person sending the memo. These lines are in image mode. For example, Wendy King.
- toname* is the name of a person receiving the memo. These lines are in image mode. For example, Leslie Tobey.
- subject* is the description of the topic of the memo. These lines are in image mode.

As many lines as needed may be used; however, they should be kept short.
For example:

**Enhancements to
the MANU System**

- \in memocc.in.manu* is used if you wish to copy the memo to other people.
- ccname* is the name of a person you wish to copy the memo to. These lines are in image mode.
- memo* is the text of the memo. These lines are in format mode.

Discussion

When your document is "finaled", MEMO prints the memo. The first page looks like the standard memo form for an HP division -- only not in blue!

NOTE

Prints a note.

Syntax

```
\**** NOTE ****  
\in note.in.manu  
text  
text  
.  
.  
.  
\in endnote.in.manu
```

Parameters

text is the text of the note. These lines are in format mode.

Discussion

When your document is "finaled", NOTE prints two blank lines, moves the left and right margins in by eight "ems", prints the logo for a note (centered), leaves one blank line, and prints the text of the note (formatted). After the note is printed, two blank lines are printed and the left and right margins are returned to their original values. NOTE prevents the note logo and the start of the text of the note from being broken across page boundaries.

You may successfully use NOTE within PARMS in order to produce a note within the explanation of a parameter.

Starts one portion of your manual or document.

Syntax

```
\**** ONESTART ****  
\m1="filename"  
\m2={ "yes"  
      "no" }  
\if main in onestart.in.manu  
[\if main in numbered.in.manu]  
[\if main in noblank.in.manu]
```

Parameters

yes indicates that this is an 8.5 x 11 inch manual (typically a reference manual).

no indicates that this is a 9 x 8.5 inch manual (typically a user's guide).

filename is the name of the file. This name will appear on a front "fold-back" page printed by ONESTART.

\if main in numbered.in.manu is used if you want numbered subsections. When subsection numbering is in effect, the level heads are numbered hierarchically. For example, when you "final" one section of your manual in one file, 7.3.1 will be printed along with the *first* level 2 head in the *third* level 1 subsection of the *current* section.

\if main in noblank.in.manu is used if you do not want extra pages printed to preserve left/right pagination. Otherwise, a page will be printed whenever necessary to preserve left/right pagination. For example, if the file you are "finaling" contains the front matter of your manual, a blank page is printed between the printing history (page iii) and the list of effective pages (page v).

These extra pages are useful in that they allow you to fold your document such that it paginates correctly. They also serve as a reminder to indicate blank pages on the pagination record; *however, the extra pages themselves should be removed before giving the reproduction package to the printer.*

Discussion

You should put each major portion of your manual or document in a separate file. Typically, you would put each section, each appendix, the front matter, and the back matter of a manual in separate files. ONESTART should be used at the *start* of each of these files.

ONESTART

ONESTART allows you to "final" one of these files by defining the environment file, font identifications, margins, macros, and blank character for standard formatting of one portion of a manual or document. Specifically, ONESTART defines the same items as MANSTART except that page numbering is not set, the table of contents file is not named as a permanent file, and the section and appendix numbers are declared to be a question mark (?). In addition, ONESTART prints a "fold-back" page containing the name of the file and the date and time (supplied by the system) that this copy was "finalized".

When you include a file in your core file (see MANSTART for a description of the core file), the items defined by MANSTART override those specified by ONESTART. Hence, when you "final" your core file, the sections will be numbered 1,2,3... and the appendixes A,B,C... rather than question marks. This allows you to reorder the sections of your manual simply by changing the order in which they are included in your core file.

NOTE

If, for reasons of performance and/or convenience, you have divided a large section of your document into multiple files, use ONESTART only at the start of the *first* file. At the *end* of the first file, use an include statement for the second file. If you need a third or fourth file, add additional include statements in the appropriate order at the end of your first file. To "final" just this section of your document, simply "final" the first file. When you include this section in your core file, simply include the name of first file.

PAGEHEAD

Starts a subsection for an item in a reference section. Also see EASYREF.

Syntax

```
\**** PAGEHEAD ****  
\m1="pagehead"  
\in pagehead.in.manu  
text  
text  
.  
.  
.  
\in pagecntd.in.manu
```

Parameters

- pagehead* is the name of the item in the reference section, usually in all caps. For example, ADD.
- text* is a brief description of the item in the reference section. These lines are in format mode.

Discussion

When your document is "finalized", PAGEHEAD forces a new page, prints the name of the reference item at the running head position, prints the description of the item (formatted) at the start of the page, and specifies an entry for the table of contents file. In addition, PAGEHEAD redefines the running head such that the name of the reference item will continue to be used until it is changed by the next occurrence of PAGEHEAD, SECTION, or APPENDX.

PARMS

Prints the parameters portion for an item in a reference section. Also see EASYREF.

Syntax

```
\**** PARS ****  
\m1="keyword"  
\in parms.in.manu  
parameter  
^ma  
explanation  
^mb  
parameter  
^ma  
explanation  
.  
.  
.  
\in endparms.in.manu
```

Parameters

keyword is the head for the parameters portion in initial caps. Typically, this will be Parameters. Other examples might be Prompts or Subcommands.

parameter is the element you wish to explain. More than one line can be used if needed; the lines are in image mode. Typically, a *parameter* will be one of the following:

- 1) an element in delite lowercase italics;
- 2) an element in uppercase in delite nonitalics;

You must invoke the desired font for the *parameter*. If you are using delite or delitei, use a maximum of 21 characters per parameter line; if necessary, use multiple lines.

explanation is the description of a *parameter*. As many lines as needed may be used; the lines are in format mode.

Discussion

When your document is "finalized", PARMS prints two blank lines, prints the specified head for the parameters portion, prints one more blank line, and prints the elements and corresponding explanations using two columns. After the parameters portion is printed, a single column is reinstated. PARMS prevents the head for the parameters portion and the start of the parameter explanations which follow from being broken across page boundaries. See SUBPARMS if you need to produce subparameters for a parameter.

PREFACE

Prints the Preface page(s). Also see FRONTMAT.

Syntax

```
\**** PREFACE ****  
\m1="mon yy"  
\in preface.in.manu  
firstpage  
[ \in prefctd.in.manu ]  
secondpage  
\in endpref.in.manu
```

Parameters

- mon yy* is the print date (month and year) of the Preface page(s). The month is represented by three characters in all caps; the year is represented by two digits. For example, DEC 82 represents December, 1982. The print date (month and year) of the Preface page(s) should be the date of the edition or subsequent update in which it was printed.
- firstpage* is the text of the first page of the Preface. These lines are in format mode. If your preface is only one page long, this is the entire preface.
- `\in prefctd.in.manu` is used if a second Preface page is needed.
- secondpage* is the text of the second page of the Preface. These lines are in format mode.

Discussion

When your document is "finalized", PREFACE prints the Preface page(s) on a right-hand page following the List of Effective Pages. Two pages may be used if necessary.

If you are preserving left/right pagination (see noblank under MANSTART), PREFACE prints a blank page following the Preface whenever only one page is used.

RCS

Prints the Reader Comment Sheet page and the "Business Reply Mail" form. Also see BACKMAT.

Syntax

```
\**** RCS ****  
\m1="name"  
\m2="type"  
\m3="partnum"  
\m4="date"  
\m5="computer"  
\m6="permitnum"  
\m7="division"  
\m8="streetaddr"  
\m9="cityaddr"  
\in rcs.in.manu
```

Parameters

- name* is the title of the manual, not including the type of manual. In most cases, this will be the product name in all caps. For example, PASCAL/3000.
- type* is the type of manual in initial caps. For example, Reference Manual.
- partnum* is the part number of the manual in the form *nnnnn-nnnnn*, where *n* represents a digit.
- date* is the date of the latest edition or update of the manual in the form *Month yyyy*, where *Month* is the month in initial caps and *yyyy* is the year represented by four digits. For example, January 1983.
- computer* is the name in initial caps of the computer systems to which the manual pertains. For example, HP 3000 Computer Systems.
- permitnum* is the first class permit number plus the city and state in all caps. For example, 1070 CUPERTINO, CALIFORNIA.
- division* is the name of the division in initial caps. For example, Information Networks Division.
- streetaddr* is the street address in initial caps of the division producing the manual. For example, 19420 Homestead Road.
- cityaddr* is the city address in initial caps of the division producing the manual. For example, Cupertino, CA 95014.

Discussion

When your document is "finaled", RCS prints the Reader Comment Sheet (including the title of the manual, part number, and latest update or edition date) followed by the "Business Reply Mail" form.

If you are preserving left/right pagination (see noblank under MANSTART), RCS prints a blank page before the Reader Comment Sheet whenever the previous section, appendix, or index was an odd number of pages.

SECTION

Starts a section.

Syntax

```
\**** SECTION ****  
\m1={ "mon yy"  
      "* HP Confidential *"  
      "* Review Copy *" }  
\m2="head"  
\m3="title"  
[ \m4="titlecntd" ]  
\in sectionn.in.manu
```

Parameters

- mon yy* is the print date (month and year) of the pages in the section. The month is represented by three characters in all caps; the year is represented by two digits. For example, DEC 82 represents December, 1982. The print date of the pages in the section should be the date of the edition or subsequent update in which the pages were printed.
- * Review Copy * is used for review copies. This causes the string * Review Copy * rather than the print date to appear at the bottom of each page.
- * HP Confidential * is used for internal documents. This causes the string * HP Confidential * rather than the print date to appear at the bottom of each page.
- head* is the running head of the section. This is usually the same as or an abbreviated version of the title of the section, only in *initial caps* rather than all caps.
- title* is the title of the section in all caps. If the section title will fit on one line within the section banner, this is the entire title; if two lines are needed, this is the first half of the title.
- titlecntd* is the continuation of the title of the section. This is only included if you need a two line title.
- sectionn* is either *section1* or *section2*, where:
- section1* is used when the section title is one line long;
- section2* is used when the section title is two lines long.

Discussion

When your document is "finalized", **SECTION** prints a right-hand page containing the section banner, prints the section title in the banner, leaves three blank lines, defines the running head, resets the page numbering, defines the two-line footing as the print date and the section-page (which alternate from right to left for odd and even pages), and specifies an entry for the table of contents file. You can use `^#s` to obtain the section number followed by a dash, as in `table ^#s2` for the second table in the current section.

If you are preserving left/right pagination (see `noblank` under `MANSTART`), a blank page is printed before the start of the section whenever the previous section was an odd number of pages.

SUBPARMS

Prints subparameters and explanations for a parameter. Intended for use within the explanation of a parameter (refer to PARMS).

Syntax

```
\**** PARMS ****  
.  
.  
parameter  
^ma  
explanation  
\**** SUBPARMS ****  
\in subparms.in.manu  
subparameter  
^ma  
subexplanation  
^mb  
subparameter  
^ma  
subexplanation  
.  
.  
\in endsub.in.manu  
[  
^ma  
explanationcntd  
^mb  
parameter  
.  
.  
]  
\in endparms.in.manu
```

Parameters

parameter

is an element you wish to explain. More than one line can be used if needed; the lines are in image mode. Typically, a *parameter* will be one of the following:

- 1) an element in delite lowercase italics;
- 2) an element in uppercase in delite nonitalics;

You must invoke the desired font for the *parameter*. If you are using delite or delitei, use a maximum of 21 characters per parameter line; if necessary, use multiple lines.

SUBPARMS

explanation is the description of a *parameter*. As many lines as needed may be used; the lines are in format mode.

subparameter is a sub-element of a *parameter* which you wish to explain. More than one line can be used if needed; the lines are in image mode. Typically, a *subparameter* will be one of the following:

- 1) an element in delite lowercase italics;
- 2) an element in uppercase in delite nonitalics;

You must invoke the desired font for the *subparameter*. If you are using delite or delitei, use a maximum of 14 characters per subparameter line; if necessary, use multiple lines.

subexplanation is the description of a *subparameter*. As many lines as needed may be used; the lines are in format mode.

explanationcntd is an optional continuation of the description of a *parameter*. It follows the subparameters for that *parameter*. Note that a \wedge ma must be added after the `\in endsup.in.manu` and before the *explanationcntd*.

Discussion

When your document is "finalized", SUBPARMS prints the subparameter elements and corresponding explanations in columns as part of the explanation of a parameter. After the subparameters are printed, columns for the parameters are reinstated.

Note that SUBPARMS is *only* intended for use within the explanation of a parameter. Refer to PARMS for information on producing parameters.

SYNTAX

Prints the syntax portion for an item in a reference section. Also see EASYREF.

Syntax

```
\**** SYNTAX ****  
\m1="inrht"  
\in syntax.in.manu  
^maline^mb  
^maline^mb  
      .          or  \in filename  
      .  
      .  
\in endbox.in.manu
```

Parameters

- inrht* is a number indicating how many "ems" of blank space you want between the right side of the box and the right margin.
- line* is a line of syntax to be printed within the box. These lines are in image mode.
- \in filename* is the name of a file generated with FSCREEN.TOOLS.MANU using the `*>>BEGIN -S` option. See section 4.

Discussion

When your document is "finaled", SYNTAX prints two blank lines, prints the head for the syntax portion, prints one more blank line, moves the left margin in by three "ems", moves the right margin in according to the value given, and prints the lines of syntax in image mode in the font pair *delite/delitei* (fontid *d*), surrounded by a box. After the syntax is printed, the margins are returned to their original values and format mode is reinvoked. SYNTAX prevents the head for the syntax portion and the start of the syntax from being broken across page boundaries.

Prints a two-column table.

Syntax

```

\**** TAB2COL ****
\m1="tabletitle"
\m2="\column (z,2,a,2,b,1)"
\in TAB2col.table.manu
/* Copy the above 4 lines for each page break *
/*
/* START OF ROW
/*
\in vern.table.manu
\next;format
text
\in vern.table.manu
\next;format
text
\in endvern.table.manu
/*
/* END OF ROW
/*
\in {MID }2col.table.manu
      .
      .
      .
/*
/* START OF ROW
/*
\in vern.table.manu
\next;format
text
\in vern.table.manu
\next;format
text
\in endvern.table.manu
/*
/* END OF ROW
/*
\in END2col.table.manu

```

Parameters

tabletitle

is the one-line title of the table. This line is in center mode.

z

is the number of "ems" (1 or more) of blank space between the left side of your table and the left margin.

TAB2COL

<i>a</i>	is the width (number of "ems") of the first column of the two-column table.
<i>b</i>	is the width (number of "ems") of the second column of the two-column table.
<i>vern</i>	is <i>ver1</i> , <i>ver2</i> , ... or <i>ver20</i> for each row, where <i>n</i> is the length (1 to 20 lines) of the vertical lines enclosing the entries that row.
<code>\next;format</code>	is a TDP formatting command for the next entry of the table. The formatting modes used for the entries of the table can be changed by replacing <i>format</i> in these commands with <i>image</i> or <i>center</i> .
<i>text</i>	is the text of an entry of the table. An entry can be at most 20 lines of output.
<i>endvern</i>	is <i>endver1</i> , <i>endver2</i> , ... or <i>endver20</i> for each row, where <i>n</i> is the length (1 to 20 lines) of the vertical lines enclosing the entries of that row.
MID	is used if you want a horizontal line below a row of the table.
SANS	is used if you do not want a horizontal line below a row of the table.

Discussion

When your document is "finaled", TAB2COL prints the title of the table (centered), prints one blank line, and then prints the two-column table. Note that a new page is not forced and that no blank lines are printed before the title of the table.

Refer to Generating Tables in section 4.

Prints a three-column table.

Syntax

```

\**** TAB3COL ****
\m1="tabletitle"
\m2="\column (z,2,a,2,b,2,c,1)"
\in TAB3col.table.manu
/* Copy the above 4 lines for each page break *
/*
/* START OF ROW
/*
\in vern.table.manu
\next;format
text
\in vern.table.manu
\next;format
text
\in vern.table.manu
\next;format
text
\in endvern.table.manu
/*
/* END OF ROW
/*
\in {MID }3col.table.manu
.
.
.
/*
/* START OF ROW
/*
\in vern.table.manu
\next;format
text
\in vern.table.manu
\next;format
text
\in vern.table.manu
\next;format
text
\in endvern.table.manu
/*
/* END OF ROW
/*
\in END3col.table.manu

```

TAB3COL

Parameters

<i>tabletitle</i>	is the one-line title of the table. This line is in center mode.
<i>z</i>	is the number of "ems" (1 or more) of blank space between the left side of your table and the left margin.
<i>a</i>	is the width (number of "ems") of the first column of the three-column table.
<i>b</i>	is the width (number of "ems") of the second column of the three-column table.
<i>c</i>	is the width (number of "ems") of the third column of the three-column table.
<i>vern</i>	is <i>ver1</i> , <i>ver2</i> , ... or <i>ver20</i> for each row, where <i>n</i> is the length (1 to 20 lines) of the vertical lines enclosing the entries of that row.
<i>\next;format</i>	is a TDP formatting command for the next entry of the table. The formatting modes used for the entries of the table can be changed by replacing <i>format</i> in these commands with <i>image</i> or <i>center</i> .
<i>text</i>	is the text of an entry of the table. An entry can be at most 20 lines of output.
<i>endvern</i>	is <i>endver1</i> , <i>endver2</i> , ... or <i>endver20</i> for each row, where <i>n</i> is the length (1 to 20 lines) of the vertical lines enclosing the entries of that row.
MID	is used if you want a horizontal line below a row of the table.
SANS	is used if you do not want a horizontal line below a row of the table.

Discussion

When your document is "finaled", TAB3COL prints the title of the table (centered), prints one blank line, and then prints the three-column table. Note that a new page is not forced and that no blank lines are printed before the title of the table.

Refer to Generating Tables in section 4.

Prints a four-column table.

Syntax

```

\**** TAB4COL ****
\m1="tabletitle"
\m2="\column (z,2,a,2,b,2,c,2,d,1)"
\in TAB4col.table.manu
\* Copy the above 4 lines for each page break *
\*
\* START OF ROW
\*
\in vern.table.manu
\next;format
text
\in vern.table.manu
\next;format
text
\in vern.table.manu
\next;format
text
\in vern.table.manu
\next;format
text
\in endvern.table.manu
\*
\* END OF ROW
\*
\in {MID }4col.table.manu
.
.
.
\*
\* START OF ROW
\*
\in vern.table.manu
\next;format
text
\in vern.table.manu
\next;format
text
\in vern.table.manu
\next;format
text
\in vern.table.manu
\next;format
text
\in endvern.table.manu

```

TAB4COL

Syntax (continued)

```
/*  
/* END OF ROW  
/*  
\in END4col.table.manu
```

Parameters

<i>tabletitle</i>	is the one-line title of the table. This line is in center mode.
<i>z</i>	is the number of "ems" (1 or more) of blank space between the left side of your table and the left margin.
<i>a</i>	is the width (number of "ems") of the first column of the four-column table.
<i>b</i>	is the width (number of "ems") of the second column of the four-column table.
<i>c</i>	is the width (number of "ems") of the third column of the four-column table.
<i>d</i>	is the width (number of "ems") of the fourth columns of the four-column table.
<i>vern</i>	is <i>ver1</i> , <i>ver2</i> , ... or <i>ver20</i> for each row, where <i>n</i> is the length (1 to 20 lines) of the vertical lines enclosing the entries of that row.
<i>\next;format</i>	is a TDP formatting command for the next entry of the table. The formatting modes used for the entries of the table can be changed by replacing <i>format</i> in these commands with <i>image</i> or <i>center</i> .
<i>text</i>	is the text of an entry of the table. An entry can be at most 20 lines of output.
<i>endvern</i>	is <i>endver1</i> , <i>endver2</i> , ... or <i>endver20</i> for each row, where <i>n</i> is the length (1 to 20 lines) of the vertical lines enclosing the entries of that row.
MID	is used if you want a horizontal line below a row of the table.
SANS	is used if you do not want a horizontal line below a row of the table.

Discussion

When your document is "finalized", TAB4COL prints the title of the table (centered), prints one blank line, and then prints the four-column table. Note that a new page is not forced and that no blank lines are printed before the title of the table.

Refer to **Generating Tables** in section 4.

TEXTREF

Starts the text reference portion for an item in a reference section. Also see EASYREF.

Syntax

```
\**** TEXTREF ****  
\in textref.in.manu
```

Discussion

When your document is "finalized", TEXTREF prints two blank lines, prints the head for the text reference portion, and prints one more blank line. Your reference to a related discussion within the same or another manual should follow. It is strongly recommended that you not give a specific page number, but rather a section name or the name of a level head within a specified section. TEXTREF prevents the head for the text reference portion and the reference from being broken across page boundaries.

Prints a Title page. Also see FRONTMAT.

Syntax

```

\**** TITLE ****
\m1="partnum"
\m2="datecode"
\m3="editiondate"
\m4="updatedate"
\m5="computer"
\m6="address"
\m7="type"
\m8="name"
[\m9="namecntd"]
\in titlen.in.manu

```

Parameters

- partnum* is the part number of the manual in the form *nnnnn-nnnnn*, where *n* represents a digit.
- datecode* is a 5 character code representing the print date of the latest edition or update. If this is a new edition, *datecode* has the form *Emmyy*. If this is an update, *datecode* has the form *Ummyy*. In both cases, *mmyy* is the month and year of the print date, each represented by two digits. For example, U0183 represents an update printed in January, 1983.
- editiondate* is the print date (month and year) of the latest edition of the manual in the form *mm/yy*. For example, 08/82 represents August, 1982. If this is a new edition, *editiondate* is the print date of *this* edition; if this is an update, *editiondate* is the print date of the *latest* edition.
- updatedate* represents the print date of the update in the form Updated *mm/yy* if this is an update; *updatedate* is an empty string if this is a new edition. For example, Updated 01/83 indicates that this is an update printed in January, 1983.
- computer* is the name in initial caps of the computer systems to which the manual pertains. For example, HP 3000 Computer Systems.
- address* is the address in all caps of the division producing the manual. For example, 19420 HOMESTEAD ROAD, CUPERTINO, CA 95014.
- type* is the type of manual in initial caps. For example, Reference Manual.
- name* is the title of the manual, not including the type of manual. If the title will fit on one line of the title page, this is the entire title (for example, PASCAL/3000). If the title needs two lines, this is the first half of the title

TITLE

of the manual. For example, IMAGE/3000 where the entire title is IMAGE/3000 DATA BASE MANAGEMENT SYSTEM.

namecntd

is the continuation of the title of the manual. This is only included if you want a two line title.

titlen

is either *title1* or *title2*, where:

title1 is used if the title of the manual is one line long.

title2 is used if the title of the manual is two lines long.

Discussion

When your document is "finaled", TITLE prints a right-hand Title page for the manual following the cover of the manual.

WARNING

Prints a warning.

Syntax

```
\**** WARNING ****  
\in warning.in.manu  
text  
text  
.  
.  
.  
\in endnote.in.manu
```

Parameters

text is the text of the warning. These lines are in format mode.

Discussion

When your document is "finalized", **WARNING** prints two blank lines, moves the left and right margins in by eight "ems", prints the logo for a warning (centered), leaves one blank line, and prints the text of the warning (formatted) in the romanb font (fontid t). After the warning is printed, two blank lines are printed and the left and right margins are returned to their original values. **WARNING** prevents the warning logo and the start of the text of the warning from being broken across page boundaries.

You may successfully use **WARNING** within **PARMS** in order to produce a warning within the explanation of a parameter.

OCT 83
5-64

You should be familiar with the following terms:

- Physical Page** The physical page is the 8.5 x 11 inch paper that is used by the laser printer. The print on the physical page may be in one of four orientations: 0, 90, 180, or 270 degrees. The page you are reading is printed in 90 degree orientation by using the MANU90.ENV.MANU environment file. Pages printed using the MANU0.ENV.MANU environment file are in 0 degree orientation.
- Logical Page** The logical page is an area within the physical page that contains the printing and any forms that have been associated with that logical page. A logical page may be smaller than or equal to the size of the physical page. Up to 32 separate or overlapping logical pages can be specified on the physical page. In the MANU90 and MANU0 environment files, there are different logical pages containing the section and appendix banners, the copyright notice box, and other fixed material. In order to obtain a particular logical page containing a particular form, that page must be activated by TDP's formatter. The logical pages contained in MANU90 are listed in appendix C; the logical pages in MANU0 are similarly defined.
- Forms** Forms are artwork created through IDIFORM. A form is placed in a fixed location within a logical page during the creation of the environment file. When that particular logical page is activated through TDP formatting commands, the form is printed.
- Fonts** A font is a collection of letters, numbers, and/or symbols all in the same height and style. They are originally hand-generated using IDSCHAR. The fonts available with the MANU system are listed in appendix C. Characters are called by typing associated letters. In some special fonts, such as the *parens* font, a conversion chart must be used to determine the correspondence between symbols and letters (see appendix C). Embedded TDP formatting commands inform the HP2680 of the font in which the letters should be printed.
- Logos** A logo is a special kind of symbol. Logos can be combined with forms on logical pages (this was done to generate the section and appendix logical pages supplied with the MANU system).
- Environment File** An environment file is a special file created by IFS2680. It combines character fonts, logical pages, and forms into a single unit for use within a document on the laser printer. The environment files supplied by the MANU system are MANU90.ENV.MANU (for 8.5 x 11 inch documents) and MANU0.ENV.MANU (for 9 x 8.5 inch documents).

For more information, refer to the *IFS/3000 Reference Guide* (36580-90001).

FUNCTION KEYS FOR TDP'S EDITOR

APPENDIX

B

It is possible to create a UDC which loads your terminal's function keys with TDP editor commands and then runs TDP/3000. You can then press the function keys from within TDP's editor to perform specific functions.

This appendix describes an example of such a UDC file. The UDCs load and label four function keys on a 264x or 262x terminal, as follows:

F1 to prompt for a MANU use file and issue the USE command for that use file;

F2 to prompt for your own use file and issue the USE command for your use file;

F3 to "final" your work file to *HP2680;

F4 to prompt for a text file and "final" it to *HP2680.

The example described in this appendix is provided in the file SOFTKEYS in the PUB group of the MANU account.

NOTE

Because loading and labeling function keys is done differently on different terminals, you will have to experiment to see whether you can use these UDCs. They were designed for the 2645 and 2626. You may have to check your terminal's reference manual and do some experimenting to find out how to load and label function keys on your terminal; then modify your copy of the SOFTKEYS file accordingly.

SOFTKEYS EXAMPLE

SOFTKEYS.PUB.MANU contains four UDCs: TDP4X, TDP2X, LOAD4X, and LOAD2X. TDP4X and LOAD4X are intended for use on a 2645 terminal; TDP2X and LOAD2X are intended for a 2626 terminal.

TDP4X and TDP2X, when executed, simply issue a HELP command for LOAD4X and LOAD2X, respectively, and then run TDP/3000. LOAD4X and LOAD2X consist of a series of COMMENT commands which, when executed, load and label the function keys **F1**, **F2**, **F3**, and **F4**.

First copy SOFTKEYS.PUB.MANU into *your* group and account. Then use the SETCATALOG command to invoke your copy of this UDC file together with any other UDC files you want invoked. For example:

```
:SETCATALOG MYUDCS,SOFTKEYS;ACCOUNT
```

(Note that if you set your UDCs for your *account*, as in the above example, you need to use the command SETCATALOG;ACCOUNT when nobody else is logged on in your account in order to unset them.)

To run TDP on a 2645 terminal with your function keys labeled, type:

```
:TDP4X
```

Or, to run TDP on a 2626 terminal with your function keys labeled, type:

```
:TDP2X
```

Your function keys will be loaded and labeled; then you will be placed into TDP/3000. On a 2645 terminal, the function key labels will be memory-locked on your terminal and the top of your screen will look like this:

Use MANU	Use Own	Final Workfile	Final...
HP36578A.03.00 TDP/3000 EDITOR		(C) Hewlett Packard Co. 1980	
SUN, OCT 9, 1983, 8:08 AM (DAY #282)			
/			

On a 2626 terminal, you will also be placed into TDP/3000 and the bottom of your screen will look like this:

Use MANU	Use Own	Final Workfile	Final...	F5	F6	F7	F8
-------------	------------	-------------------	----------	----	----	----	----

You can now proceed as you normally would in TDP's editor. When you wish to perform one of the labeled functions, simply press the appropriate function key, as follows:

- 1) If you want to use a use file from the USE group of the MANU account, simply press **F1**. You will be prompted for the name of the MANU use file. For example:

use MANU: onestart

The use file will then execute and prompt you in its usual fashion.

- 2) If you want to use a use file in your *own* group and account, press **F2**. You will be prompted for the name of your own use file. For example:

use OWN: myuse

Your use file will then execute in its usual fashion.

- 3) To "final" your work file to *HP2680, press **F3**. Your work file is automatically "finalized" to the laser printer.

- 4) To "final" a text file to *HP2680, press **F4**. You will be prompted for the name of your "from" file. For example:

From: mydoc

The file you specify is automatically "finalized" to the laser printer.

Occasionally you will get an error such as INVALID COMMAND or KEEP OR DELETE WORK FILE when you press one of the function keys. If this happens, press **CONTROL Y** and try again.

On a 262x terminal, you may use TDP's screen mode as you usually would; when you return from screen mode, the function keys will still be loaded and labeled.

On a 264x terminal, the function key labels will occupy two lines of your screen while you are in screen mode (leaving less room on your screen for text) and will scroll off your screen when you return to line mode because they are no longer memory-locked. Because of this, we suggest that you first turn the memory-lock function off on a 264x terminal before using screen mode. When you enter screen mode, the function key labels will disappear (giving you the full screen to work in); when you return to line mode, you can label your function keys again by pressing **BREAK**, typing LOAD4X (*not* TDP4X, as this will run TDP once again), and then RESUME. Note that the function keys are still *loaded* when you return from screen mode, they are simply not *labeled*; if you can remember what they are, you can use them without re-labeling them.

On a 264x terminal, you should turn memory-lock off after exiting TDP.

The logical pages defined for the MANU environment files are listed below. Where appropriate, the various logical pages for the MANU90 environment have been demonstrated within this document; the logical pages of MANU0 are similarly defined. This appendix also documents the font identifications defined by the MANU90 and MANU0 environment files in the ENV group of the MANU account. A scale for calculating "ems" and lines for the standard logical page in the MANU90 environment is provided at the end of this appendix.


LOGICAL PAGES

Table C-1. Logical Pages in the MANU Environments

Page Number	Function
0	Blank.
1	Banner box for front and back matter.
2	Banner box for the start of a section.
5	Banner box for the start of an appendix.
7	Box for the copyright notice page.

FONT IDENTIFICATIONS

Table C-2. Font Identifications

Font	Id	Font Sample
roman,romani romani romanb	^fr ^fs ^ft	Aa Bb Cc Dd Ee 1 2 3 4 5 6 7 8 9 0 Aa Bb Cc Dd Ee 1 2 3 4 5 6 7 8 9 0 Aa Bb Cc Dd Ee 1 2 3 4 5 6 7 8 9 0
delite,delitei delitei deliteb	^fd ^fe ^ff	Aa Bb Cc Dd Ee 1 2 3 4 5 6 7 8 9 0 Aa Bb Cc Dd Ee 1 2 3 4 5 6 7 8 9 0 Aa Bb Cc Dd Ee 1 2 3 4 5 6 7 8 9 0
deliteg,delitegi delitegi	^fg ^fi	Aa Bb Cc Dd Ee 1 2 3 4 5 6 7 8 9 0 ^fg=␣ Aa Bb Cc Dd Ee 1 2 3 4 5 6 7 8 9 0 ^fi=␣
delite,roman	^fq	A B C D E ^aA B C D E ^nA B C D E
helvb10 helvb12 helvb14 helvb20 helvb24	^fv ^fw ^fx ^fy ^fz	Aa Bb Cc Dd Ee 1 2 3 4 5 6 7 8 9 0 Aa Bb Cc Dd Ee 1 2 3 4 5 6 7 8 9 0 Aa Bb Cc Dd Ee 1 2 3 4 5 6 7 8 9 0 Aa Bb Cc Dd Ee 1 2 3 4 5 Aa Bb Cc Dd Ee 1 2 3
linedraw *	^fl	␣ † # ◻ • ◻ # = ↑ ∥ † † † → ∞ † † † † †
parens *	^fp	() [] { } Δ □ ▣ ▤ ▥ ▦ ▧ ▨ ▩
boxdraw *	^fb	┌ ┐ └ ┘ ─ ┬ ┴ ┬ ┴ ─ []
keys *	^fk	^fkR=(RETURN) ^fkE=(ENTER) ^fk2=(/ 2)
math *	^fm	∇ ∈ Σ ψ ϕ \$ ⇒ ϕ Ψ ∃ ⊙ ⋈ ξ θ ↗ ↘ ⊖ β
achtung	^fa	N= NOTE C= CAUTION W= WARNING
hplogo	^fh	^fHP=  HEWLETT PACKARD

* Refer to the Conversion Tables on the following pages for these fonts.

**Table C-3. Conversions for the LINEDRAW Font
AND for the Terminal's Alternate Character Set (Linedrawing)**

^f1 "linedraw"														
†	‡	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
1	2	3	4	5	6	7	8	9	0	-	=			
†	‡	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
!	"	#	\$	%	&	'	()	^	\				
HP2680 Laser Printer ONLY *														
↑	→	⊥	√	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
q	w	e	r	t	y	u	i	o	p					
↓	←	⊥	⊥	:	⊥	⊥	⊥	/	□	•				
a	s	d	f	g	h	j	k	l	~					
⊥	→	⊥	⊥	:	⊥	⊥	\	/						
z	x	c	v	b	n	m	{	}						
⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
A	S	D	F	G	H	J	K	L	:	:]	+	*	
Z	X	C	V	B	N	M	,	.	/	<	>	?		
*Refer to Section 4--Using FSCREEN.TOOLS.MANU														

Table C-4. Conversions for the PARENS Font

Character	Symbol	Character	Symbol
Q or q	[W or w]
A or a	⌊	S or s	⌋
R or r	(T or t)
F or f	\	G or g	/
6	{	5	}
\	\	/	/
. or :		-	Δ
;	⋮	,	⋮

Table C-5. Conversions for the BOXDRAW Font

Character	Symbol	Character	Symbol
R	┌	.	
T	┐	,	—
F	└	:	⋮
G	┘	;	—
Q	┌	5	┌
W	┐	6	┐
A	└	7	└
S	┘	8	┘
[[/	+
]]	=	—

Table C-6. Conversions for the KEYS Font

Character	Symbol	Character	Symbol
R	RETURN	1	/1
E	ENTER	2	/2
B	BREAK	3	/3
C	CONTROL	4	/4
T	TAB	5	/5
S	SHIFT	6	/6
e	ESCAPE	7	/7
<	BACKSPACE	8	/8
A	AIDS	f	/
M	MODES	b	
U	USERKEYS		

Table C-7. Conversions for the MATH Font

Character	Symbol	Character	Symbol
A	∇	a	α
B	ε	b	β
C	⊂	c	ψ
D	∩	d	∅
E	∃	e	ε
F	€	f	θ
G	⊆	g	λ
H	ℵ	h	π
I	∩	i	ι
J	∩	j	θ
K	⊗	k	κ
L	∩	l	ω
M	∧	m	μ
N	≠	n	ν
O	•	o	ρ
P	⊗	p	π
Q	∅	q	γ
R	⊃	r	θ
S	∩	s	σ
T	→	t	τ
U	∪	u	ε
V	∩	v	Δ
W	∪	w	δ
X	×	x	×

Table C-7. Conversions for the MATH Font (continued)

Character	Symbol	Character	Symbol
Y	∅	y	∪
Z	■	z	ζ
1	¹	!	✓
2	²	"	
3	³	#	§
4	⁴	\$	∇
5	⁵	%	±
6	⁶	&	α
7	⁷	'	∩
8	⁸	(+
9	⁹)	≈
0	⁰	-	≡
=	∩		→
^	⇐	@	⊙
\	⇒	-	↓
-	←	˘	¶
[<	;	Λ
]	>	:	Ω
{	↑	+	Γ
}	†	*	Π
,	Ψ	<	∞
.	Φ	>	†
/	≡		
?	Σ		

GUIDELINES FOR FONT USAGE

APPENDIX

D

The default font in the MANU Formatting System is roman. Roman is a 10 point, proportionally spaced, serif font. If you do not specify otherwise, text is printed in this font.

Many of the MANU include files automatically switch fonts for you. In addition, there are many other times when you will want to change fonts. Some guidelines follow:

Table D-1. Font Usage Guidelines

Fontid	Font	Usage
<code>^fr</code>	roman	Use for the main text of documents. Use all caps for product names referred to within text.
<code>^fs</code>	romani	Use to emphasize a word or phrase within text. Also use when referring to the title of a manual, course, or book.
<code>^ft</code>	romanb	Use for a word or phrase the first time it is defined. Also use for figure titles, table titles, column headings, and similar applications.
<code>^fd</code>	delite	Use to represent any type of text the user will see on a terminal screen, see on a listing, or type in exactly as shown. This includes what the user must enter as shown in syntax statements or parameter explanations, examples of interactive programs, examples of listings, and so on. Commands, intrinsics, keywords, etc. which are referred to within text should also be in delite.
<code>^fe</code>	delitei	Use for parameters for which the user must supply a value. This includes user-supplied parameters in syntax statements, in parameter explanations, and within text. Do not use for a regular-size bracket or for parenthesis.
<code>^fg</code>	deliteg	Use for anything displayed on the terminal in inverse video. This includes function key labels and VPLUS/3000 screen fields. Also use to highlight portions of an example.
<code>^fi</code>	delitegi	Use in tutorials to highlight parameters for which the user must supply a value. Also use for field descriptions in VPLUS/3000 screen fields.
<code>^fp</code>	parens	Use to build large brackets, braces, and parentheses in syntax statements. (Use delite for a regular-size bracket or parenthesis in syntax.) Also use for shaded delimiters to indicate positional parameters and for required spaces in syntax statements (refer to the conventions page).

MACROS AND NUMBERING ARGUMENTS

APPENDIX

E

Certain macros and automatic numbering arguments are reserved for use by the MANU formatting system. This appendix describes how they are used by the MANU formatting system.

When defining your *own* macros and automatic numbering arguments, you should be careful not to use those listed in this appendix.

MACROS

The following macros are used by the MANU formatting system:

m0	Scratch pad macro used by include files.
m1 - m9	Parameters passed to include files.
ma - md	Parameters output by include files.
me - ms	Reserved for future use.
mt	Indicates format of manual (yes if 8.5 x 11, no if 9 x 8.5).
mu	Indicates whether to preserve left/right pagination (yes if extra pages are to be inserted, no if not).
mv	Indicates whether blank pages are needed in front matter.
mw	String with which to start number appendice; #0 is set to this macro when the appendices are started.
mx	Indicates whether the appendices have started printing yet (yes if the first appendix has been started, no if not).
my	Indicates whether subsection numbering is in effect (yes if in effect, no if not).
mz	Indicates whether in image mode or not (yes if in image mode, no if not).

NUMBERING ARGUMENTS

The following automatic numbering arguments are used by the MANU formatting system:

#0	Section/appendix number.
#1 - #4	Subsection number (such as 3.4.2).
#5	Reserved for future use.
#9	Bullet number.

QUICK REFERENCE

APPENDIX

F

This appendix contains a chart of all the MANU use files and corresponding include files, organized by function. Include files shown in brackets are optionally included by the use file. If a use file invokes other use files, it is shown in bold, and the use files it invokes are indented below it.

Table F-1. Use Files and Corresponding Include Files

Function	USE Files	Corresponding Include Files
Print entire manual Start a file Start a section Start an appendix Start a level 1 subsection Start a level 2 subsection Start a level 3 subsection Start a level 4 subsection	MANSTART ONESTART SECTION APPENDX LEVEL1 LEVEL2 LEVEL3 LEVEL4	MANSTART, [NUMBERED], [NOBLANK] ONESTART, [NUMBERED], [NOBLANK] SECTION1 or SECTION2 APPENDX1 or APPENDX2 LEVEL1 or LEVEL1X LEVEL2 LEVEL3 or LEVEL3X LEVEL4
Print a note Print a caution Print a warning Switch to image mode Switch to "block" mode Switch to format mode Print bulleted items Print a box Print a bold box	NOTE CAUTION WARNING IMAGE BLOCK FORMAT BULLETS BOX BOX	NOTE and ENDNOTE CAUTION and ENDNOTE WARNING and ENDNOTE IMAGE BLOCK FORMAT BULLETS and ENDBULL BOX and ENDBOX BOXB and ENDBOXB
Print a memo	MEMO	MEMOFROM, MEMOTO, MEMOSUB, MEMOCC, and ENDMEMO
Print a complete reference Start a reference Print syntax Print parameters Start a discussion Print an example Start a text reference Print subparameters	EASYREF PAGEHEAD SYNTAX PARMS DISCUSS EXAMPLE TEXTREF SUBPARMS	PAGEHEAD and PAGECNTD SYNTAX and ENDBOX PARMS and ENDPARMS DISCUSS EXAMPLE and ENDEXAMP TEXTREF SUBPARMS and ENDSUBP
Print front matter Print a cover [or] Print an update cover Print a title page Print a copyright page Print the printing history Print the effective pages Print the preface Print the conventions	FRONTMAT COVER [COVERUPD] TITLE COPYRITE HISTORY EFFPAGES PREFACE CONVEN	COVER1 or COVER2 [COVERUPD] TITLE1 or TITLE2 COPYRITE HISTORY and ENDHIST EFFPAGES, [EFFCNTD], and ENDEFF PREFACE, [PREFCNTD], and ENDPREF CONVEN
Print an internal cover Print a figures/tables page Print an index Print a table of contents	LABCOV FIGTAB INDX CON	LABCOV and ENDLCOV FIGTAB, [FIGTCNTD], and ENDFIGT INDX (and INDXER. TOOLS.MANU) CONBEGIN, [CONRIGHT], [CONLEFT], and ENDCON
Print back matter Print a comment sheet Print a back cover	BACKMAT RCS BACKCOV	RCS BACKCOV

Table F-2. Use Files and Corresponding Include Files for Tables

Function	USE Files	Corresponding INclude Files in the TABLE Group
Print a two-column table	TAB2COL	TAB2COL MID2COL or SANS2COL END2COL VER[1-20] ENDVER[1-20]
Print a three-column table	TAB3COL	TAB3COL MID3COL or SANS3COL END3COL VER[1-20] ENDVER[1-20]
Print a four-column table	TAB4COL	TAB4COL MID4COL or SANS4COL END4COL VER[1-20] ENDVER[1-20]

OCT 83
F-4

INDEX

NOTE

If this manual had an index, this is where it would go...

