LYRIX™ System Administrator's Guide

The Santa Cruz Operation, Inc.

The Santa Cruz Operation, Inc.
500 Chestnut Street
P.O. Box 1900
Santa Cruz, California 95061
(408) 425-7222 · TWX: 910-598-4510 SCO SACZ

Release: 6/15/84/3.10

Table of Contents

INTRODUCTION

The Lyrix word processing system features an integrated, user-definable environment. By editing certain text files, the Lyrix administrator can create a tailored interface made up of customized menus, editing command keystrokes, special printer and terminal effects, and system messages.

Menus can provide rapid access to virtually any word processing function, UNIX OS facility, application program, or other menus.

All command keystrokes used in the word processing editor are completely user assignable, making single command keystrokes possible for a wide variety of terminals with programmable function keys.

System messages can be rewritten as Lyrix is modified to suit individual requirements. They can be written in other languages to make Lyrix a truly universal and flexible system.

This guide explains how to control these variables to design a word processing system that suits your needs.

USING THIS GUIDE

Each chapter of this Guide explains a different concept or area of Lyrix Administration. The chapter contents are summarized here:

* *Chapter 1* describes the organization of the Lyrix modules that control the operation and appearance of the software.

* *Chapter 2* introduces the Command File, the module that contains most of the user-definable variables.

* *Chapter 3* discusses certain system attributes that control details of the editing environment.

* *Chapter 4* explains how editing command keystrokes are assigned.

* *Chapter 5* describes menus and how to design and alter them.

* *Chapter 6* discusses system messages and how to change them.

* *Chapter 7* shows how default rulers are changed.

* *Chapter 8* explains how printer effects are enabled.

* *Chapter 9* describes the Tcap module, and how it is altered to take advantage of different terminal and printer capabilities.

* *Chapter 10* introduces the Fprint module, which controls printing.

* *Chapter 11* discusses the Viewprint and Mail Merge screens.

* *Appendix A* contains a complete standard Command File.

* *Appendix B* contains complete standard Viewprint and Mail Merge screen files.

* *Appendix C* contains a table of ASCII decimal character equivalents.

UNDERSTANDING THIS GUIDE

To simplify use of this guide, several conventions are used:

* *Italics* are used to highlight command names, and wherever new concepts are introduced.

* Examples are used throughout to clarify user action and system response.

* Major sections are summarized to encapsulate main points.

* Control sequences are represented as <CTL>. For instance, control-C is represented by <CTL>C.

CHAPTER 1
Lyrix Modules

The Lyrix system contains two parts, the Word-Processing Editor, and the Menu System. You can alter both parts by editing certain text files, called *modules*, that govern specific areas. The owner should be *bin*, and file size will vary because of differences among the particular machines that the files are installed on.

The directories */usr/bin* and */usr/lib/wp* should have general read-execute permission. This can be verified by looking at the listing produced by the *ls –al* command.

When limited space is available in */usr* on your system, you can create a new file called */usr/lib/wp/Redirect*. This file contains a valid pathname of the directory that will contain the Lyrix files which would otherwise be found in */usr/lib/wp*.

Error messages displayed or printed by Lyrix are usually self-explanatory and often relate to syntax errors within the system Command File. Error messages relating to the Command File will generally notify you of an error in a particular section such as:

```
Error in section #SYSTEM
```

Errors in certain Command File sections can be fatal, causing Lyrix to fail. Other less critical errors can simply produce a badly formatted menu or an unexpected command result.

> ■ **NOTE:** Often, improperly formatted menus are caused by Lyrix's inability to correctly identify the terminal being used. This can usually be traced to a problem with interpretation of the UNIX OS /etc/termcap database, such as the user supplying an incorrect TERM environment variable on login.

Lyrix is provided with a function that stamps your program with a serial number which must be obtained from the Santa Cruz Operation or other authorized Lyrix supplier. Internal validation determines whether the stamped number is valid and whether it is a demonstration or full-featured serial number. If the

numer is invalid, Lyrix will not work. Refer to the Installation Notes provided with Lyrix to assign the correct serial number to your package.

CHAPTER 2
The Command File

Most user-variable Lyrix parameters are contained in the *Command File* module. The Command File is composed of sections that define how Lyrix appears to the user—all menus, the system messages, the keystrokes necessary to initiate editing commands, and certain general aspects of the editing environment. Other Lyrix modules control the printing system (see *Fprint*, Chapter 10), and printer and terminal special effects (see *Tcap*, Chapter 9).

This chapter provides an overview of the Command File and describes syntax that is common to all sections of the Command File. Subsequent chapters discuss each section of the Command File in detail.

Designing Your Own Word Processor

The Command File is an ordinary text file and can be edited using Lyrix or any other standard text editor. User interfaces consisting of customized menus, messages, and commands can be developed by simply editing the standard Command File. This allows different command interfaces for different terminal types, by enabling function keys, for instance. Menus using a foreign language can be easily developed.

The Command File can be either in a central location, serving multiple users, or each user can have a personal Command File. Multiple Command Files can be developed, each one addressing different levels of user experience. You can use this powerful facility to personalize Lyrix on an individual basis.

A group of users in an office, for instance, could all share a Command File that provides access to the functions most often used by the group by each keeping a copy of the special version in their home directories. A word processing pool, for example, might want a set of menus specifically designed for word processing applications, while a marketing group might want menus designed for database application access and simple letter generation.

■ **NOTE:** The standard Command File distributed with Lyrix is used in this guide as the basis for all explanations of Command File syntax. Modified sections are used to illustrate customized Command Files.

Always make a copy of the Command File before making any editing changes. Then, edit the copy, saving the original as a backup.

Command File Syntax

The Command File is organized into units called *sections*; each section is a collection of parameters that covers a particular application of the software.

For instance, the #COMMANDS section contains definitions of keystrokes that initiate Lyrix editing commands. Two others, #USER and #MESSAGES, contain all system messages. Sections can be separated by blank lines and remarks to enhance readability.

Each component of a Command File section is called an *entry*. Each entry defines some specific keystroke, message, or system detail. When literal strings are used in an entry, it must be surrounded by single right quotation marks (').

■ **NOTE:** You can instruct Lyrix to read single quotes literally by including a backslash (\) before the quote.

Section headers always start at the leftmost column (column one), are unique to the current Command File, and begin with the crosshatch symbol (#). Except for these three requirements, headers can contain any combination of spaces and upper and lowercase alphanumeric characters.

Header names for mandatory sections cannot be changed (mandatory sections: #SYSTEM, #MESSAGES, #COMMANDS, #USER, #RULERS, #EFFECTS, #SYSMENU, and #EDITMENU). The default #COMMANDS section is not terminal specific. Additional #COMMANDS sections can be terminal specific and must list the terminal name. Other Command File section headers can be changed as desired.

■ **NOTE:** For the sake of consistency, standard Command File sections are identified by upper-case names which describe the section contents.

Each Command File section is concluded with two right-hand parentheses on a separate line [))]. As with all entries, these symbols must appear beginning in the first column.

There are three types of Command File sections:

1) Those sections that are necessary to the operation of Lyrix (noted as mandatory sections above).
2) Additional #COMMANDS sections that define command keystrokes for particular terminal types.
3) The sections that define menus.

The header and double right parentheses conventions make each Command File section easily accessible by Lyrix's *Text Merge* feature. Command File sections can be easily copied from other files using Lyrix.

For example, to use Lyrix to merge a section into a Command File:

- Position the cursor below where the copied section is to be placed.
- Type the *Merge Insert* Command (in the standard release it is <ESC> mi).
- Type the source Command File name, followed by a comma and the section header for the section required, press <Return>.

For details, see the *Lyrix User's Guide, Copying Text Into Another File.*

■ **NOTE:** This technique is primarily designed for merging conventional text files. For this reason, merged sections are copied *without* the section headers and right parentheses delimiters; these can be added quickly with the editor.

Each section entry is limited to a single line of text of up to 252 characters.

Standard Command File Sections

The standard Command File contains 26 sections, most of which define menus. Of these, eight are mandatory for any Command File:

#SYSTEM	General parameter definitions
#MESSAGES	Text for Lyrix controlled system messages
#COMMANDS	Keystroke definition table
#RULERS	Default ruler settings
#EFFECTS	Printer effects defined for system
#SYSMENU	Main Menu definition
#EDITMENU	Document escape menu (escape to menu from within the editor)
#USER	Messages for user-definable menu entries

The other nine sections define menus for the standard release:

#COMLOCATE	Keystroke cross reference table
#DOCPREP	Word Processing menu
#PRINTING	Printing system menu
#SYSCOMMS	UNIX OS access menu
#CHECKER	File checking menu
#FILEMANAGE	File management menu
#MAIL	Communications menu
#HELP	Help menu
#COMSUM	Brief command definitions menu

In addition to these sections, the Standard Command file includes nine special #COMMANDS sections. Each defines command keystrokes for a specific terminal or computer.

In the next chapters of this guide, each of these sections is fully described along with selected examples of customized sections.

Summary

- Most user-variable Lyrix parameters are found in the *Command File* module, which defines all menus, system messages, and keystrokes necessary to initiate editing commands.
- Other Lyrix modules include Fprint and Tcap, which control the printing system and special printer and terminal effects, respectively.
- The Command File is an ordinary text file and can be edited using the Lyrix or any other text editor.
- Multiple Command Files can be used, each one addressing differing user experience levels and needs.
- The Command File is organized into units called *sections*, each one a collection of parameters that covers a particular area of the software.
- Sections can be separated by blank lines and comments.
- Each menu is defined in an individual section.
- Each part of a section is called an *entry*.
- Each line of a section must begin in column one.
- Although a section entry is limited to a single line, lines can be of up to 252 characters.
- The first entry in any section is the section header, which must be unique to the current Command File, and begin with the crosshatch symbol (#).
- Each Command File section is concluded with two right parentheses starting in column one on a line by themselves.
- Command File sections can be copied from other files using Lyrix's *Text Merge* facility.
- Twenty-two sections constitute the standard Command File. Eight of these are required for all Command Files.

CHAPTER 3
System Details

The #SYSTEM section of the Command File includes general information necessary to the functioning of Lyrix. The standard #SYSTEM section looks like this:

```
#SYSTEM
BACKUP='^.bak'
NAME='COMPANY NAME'
2SPACE='.!?:;',                          3SPACE=''
RULER='.#CIJLTRH'
WDEL=';:,.!? ',                          HLEN=5
DOTS='HE:HM:FO:FM:PL:PA:PN:SN:PM:SP:JY:JN:RE:ME:ST:SB'
KEYIN='Y'
MODE='',                              PAGE='66'
PNUM='#',                              STOP=' *&()<>'
DECTAB='.'
STATUS='S'
))
```

Each entry defines variables that correspond to a particular area of the software. Entries can be included in free format, in any order you choose. If you include more than one entry per line, be sure to put a comma between them, as in this line:

```
WDEL=';:,.!? ',                          HLEN=5
```

Be careful not to put a comma at the end of a line as this will produce an error message and will keep you from being able to use Lyrix.

Entries not being used do not have to be included at all. For example, the MODE entry is defined as a null string above only to note its existence (this entry could have been eliminated all together).

Each entry is described below:

BACKUP

This entry defines the name of the file in which Lyrix stores an automatic backup of any file being edited:

```
BACKUP='^.bak'
```

The caret (^) character places the name of the file to be backed up before the .bak suffix. Thus, the backup of a file named test would be called test.bak. The backup file is stored in the current directory, unless a complete pathname is specified, for instance,

```
BACKUP='/usr/uup/backup'
```

The tilde (~) character cuases Lyrix to place the backup file in the user's home directory:

```
BACKUP='~/filename'
```

Backups for systems supporting more than one user should be kept in different locations in order to avoid unintentional overwrites of another user's backup file. Only one user should ever access a backup file.

In such cases, Lyrix replaces the tilde with the user's environment variable, *$HOME*. The ^ symbol can be used in conjunction with the tilde.

Note that if there is no entry for BACKUP, you will be unable to edit with Lyrix.

2SPACE and 3SPACE

These entries define characters that require special spacing during paragraph formatting by Lyrix. The characters listed in 2SPACE and 3SPACE will have two or three spaces, respectively, added after them when Lyrix is formatting paragraphs. This adjustment is applied only if the characters are followed by at least one space in the file; Lyrix automatically adjusts the number of spaces to two or three as appropriate.

This entry is used in the standard release:

2SPACE='.!?:;', 3SPACE=''

DECTAB

Defines the decimal point character that Lyrix is to recognize when doing decimal tab alignment. In the standard release, this is defined as a period:

DECTAB='.'

If there is no entry for DECTAB, the numbers will align themselves at the space at the end of the numbers instead of at a specified character.

RULER

RULER contains a list of valid ruler characters that must appear in the correct order as noted in this table:

Position:	Function:	Standard Setting:
1	Blank position	.
2	Decimal tab	#
3	Center	C
4	Indent	I
5	Justify right margin	J
6	Left margin	L
7	Tab stop	T
8	Right margin, no justification	R
9	Paragraph hang	H

The characters can be changed as long as they maintain the order above. The standard entry is included as:

RULER='.#CIJLTRH'

If no ruler entry is provided, you will not be provided with rulers when you edit a file. Also your terminal will make a tone upon entering a file to edit it.

WDEL

WDEL contains the characters designated to be word delimiters during the preparation and formatting of text with Lyrix.

This information is used during the execution of certain functions and processes in the editor such as word-wrap, reformat, tab and so on.

In the standard distribution, WDEL is set to include the space character and all punctuation characters in normal use. However, if you use Lyrix with non-English character sets, WDEL will need to be expanded.

The standard entry is:

```
WDEL=':;,.!? ',
```

If no WDEL entry is provided, Lyrix reads the end of the word as representing the end of the line.

HLEN

This entry is used to control the Lyrix hyphenation system. The number entered here indicates to Lyrix the number of consecutive unfilled spaces permitted in a line before an attempt to hyphenate occurs.

Trial and error indicates that a suitable value for English is 5 spaces. Non-English languages will probably require different settings of this variable.

Thus, the standard entry is:

```
HLEN=5
```

DOTS

This entry defines the two letter codes that name Print-time Commands. The standard entry is:

```
DOTS='HE:HM:FO:FM:PL:PA:PN:SN:PM:SP:JY:JN:RE:ME:ST:SB'
```

These commands define certain parameters at print-time:

Print-Time Command	Tells Lyrix To ...
.HE**n**	The next n lines are printed as a header at the top of each page.
.HM**n**	Start printing text n lines after the end of the header.
.FO**n**	The next n lines are printed as a footer on each page.
.FM**n**	Stop printing text n lines before the beginning of the footer.
.PL**n**	Set the page length to n lines.
.PA	Start a new page here.
.PN**n**	Set the page number to n.
.SN**n**	Sends specified code to printer at print-time.
.PM**n**	Start a new page here if fewer than n lines remain.
.SP**n**	Insert n–1 blank lines between each printed line.
.JY	When the global reformat command is given, all text beneath the .JY print-time command is reformatted according to the rulers in effect.
.JN	Tells Lyrix not to reformat subsequent text.
.RE **comment**	Denotes comment line which is not printed at print time.
.ME **filename**	Merges indicated file at print time.
.ST**n**	Sends indicated code to printer for top of page controls.
.SB**n**	Sends indicated code to printer for bottom page controls.

You can rename any print-time command with a unique two letter combination if you choose. The only condition is that the commands are position dependent as are the RULER characters; to replace *HE* with *HD*, for instance, type *HD* in the first position.

No entry in DOTS means print-time commands placed within the text of a document cannot be understood by Lyrix.

KEYIN

KEYIN reads your keyboard input and saves it until Lyrix is ready to use it. This facilitates type-ahead when your system is slow. The default KEYIN entry is:

```
KEYIN='Y'
```

KEYIN is not necessary for systems running XENIX*.

If you use the XENIX system, the KEYIN entry will be ignored by the computer. For any other UNIX OS or system based on the UNIX OS, you may wish to leave KEYIN set to "Y". However, as this creates an extra process and may slow down your system, you may opt to set keyin to "N".

MODE

This entry specifies modes that are automatically activated when the user calls up the document editor.

The characters specified are capital letters denoting *Insert mode* (I), *Hyphenation mode* (H), or *Auto tab mode* (T). For example,

```
MODE='I'
```

causes the user to be automatically placed in Insert mode whenever the document editor is entered. The user can turn off any preset mode after entering the file.

The standard entry is defined as a null string:

```
MODE=''
```

As mentioned earlier, MODE is not used in the standard release, and could be omitted. It is shown here to note its existence.

PAGE

This entry specifies the default page length. If not specified in the Command File, 66 lines is used as the default page length. The standard release includes this entry to note its existence:

```
PAGE='66'
```

*XENIX is a trademark of Microsoft Corporation

PNUM

This entry defines the character used to represent the current page number in header and footer definitions. At print-time, this character is replaced with the current page number.

For example, to use the crosshatch character (#) for the page number, set PNUM as follows:

```
PNUM='#'
```

If there is no PNUM entry, Fprint will not number pages at print-time.

STATUS

The STATUS entry enables the status line during file edit. The standard entry is:

```
STATUS='S'
```

this enables the complete status line. The column counter on the status line (which indicates the cursor location) is disabled when the entry is included as:

```
STATUS='X'
```

This may be preferable in certain instances, as the system requires some time to update the counter each time the cursor is moved.

STOP

This entry defines characters that are not allowed in file names specified by the user. The standard entry is:

```
STOP=' *&()<>'
```

When any of the characters named in this entry are subsequently used in a file name specified by the user, the *Not a good file name!* message appears and the user is reprompted for a valid file name.

Summary

- The #SYSTEM section contains various details relating to the operation of Lyrix.
- Each entry defines variables that correspond to a particular area of the software.
- Entries can be arranged in free format, and those not required or not being used can be omitted entirely.
- The BACKUP entry specifies the name and location of the backup file in which Lyrix saves a copy of the file being edited.
- The 2SPACE and 3SPACE entries define characters that are to be followed by two or three spaces, respectively, when Lyrix formats text.
- The RULER entry defines valid ruler characters. If changed, the entries must preserve the order as included in the standard release.
- The WDEL entry defines characters used as word delimiters.
- The HLEN entry tells Lyrix at what point to attempt to hyphenate words when hyphenation mode is in effect.
- The DOTS entry defines two letter codes used for print-time commands. If changed, the entries must preserve the order as included in the standard release.
- The MODE entry specifies modes that are automatically activated when the user begins editing a document.
- The PAGE entry defines the default page length in number of lines.
- The PNUM entry establishes the character used to specify the current page number in headers and footers.
- The STOP entry specifies characters that are not allowed in file names.
- The DECTAB entry defines the character used for decimal alignment in text.

CHAPTER 4
Commands

Lyrix editing command keystrokes are completely assignable, making Lyrix compatible with a wide variety of terminals, including those with function keys. Using the function keys found on many terminals, a single keystroke can invoke any command.

> ■ **NOTE:** Always make a copy of the Command File before making any editing changes. Then edit the copy, saving the original as a backup.

Characteristics of specific terminal types can be included in any Command File. A single Command File can serve a group of users by specifying keystroke definitions for several terminals. Or, sites with many diverse terminals can be easily accommodated with personal command files.

The keystroke sequences that invoke editing commands are assigned in the #COMMANDS section. Keystrokes necessary to invoke commands can be changed by simply altering the entries in this section.

Commands are represented by codes (F01–F75); the correspondence of codes to editing commands is contained in the #COMLOCATE section:

#COMLOCATE

F01=Delete char.	F02=Destructive Bsp.	F03=Delete word	F04=Delete right
F05=Delete left	F06=Delete line	F07=Delete blanks	F08=Insert space
F09=Insert line	F10=Insert blank lines	F11=Return	F12=Left
F13=Right	F14=Up	F15=Down	F16=Tab
F17=Previous word	F18=Next word	F19=Line up	F20=Line down
F21=Go left	F22=Go right	F23=Go up	F24=Go down
F25=Top of screen	F26=Bottom of screen	F27=Scroll down	F28=Scroll up
F29=Top of file	F30=Bottom of file	F31=Lower case	F32=Upper case
F33=Center line	F34=Not in use	F35=Show effect	F36=Line split
F37=Do again	F38=Format paragraph	F39=Help	F40=Redraw screen
F41=Find NEXT	F42=Find pattern	F43=Search/repl.	F44=Format Document
F45=Stop command	F46=Go To Page	F47=Recall Ruler	F48=Store ruler
F49=Use Ruler	F50=Use mode	F51=Stop mode	F52=CP mark (block)
F53=CP blank	F54=CP leave	F55=CP Remove	F56=Not In Use
F57=CP Overlay	F58=CP Insert	F59=CP Elbow	F60=Menu escape
F61=Exit & save	F62=Quit (no save)	F63=Write no exit	F64=Not In Use
F65=Merge Insert	F66=Restore text	F67=Insert on	F68=Insert off
F69=Effect ON	F70=Effect OFF	F71=CP mark (ser)	F72=Save CP text
F73=Merge overlay	F74=Save to file	F75=Spell	

■ **NOTE: #COMLOCATE is only used for administrator reference, and as such, is not really a Command File section. Note the absence of the double right parentheses; this is done so that when #COMLOCATE is included just before #COMMANDS, they can be copied to other locations as a set. Technically, #COMLOCATE is a series of remarks between sections.**

The #COMMANDS Section

As an example of how keystroke sequences are assigned to the command codes, here is the first line of the standard #COMMANDS section:

```
F01=3,   F02=127,   F03=23,   F04=$-'DR'
```

All Lyrix keystroke sequences in the standard release are case independent, meaning that the user can type either an upper or lower case letter to invoke the same command. The standard release uses *F* in defining keystroke sequences, as above. Sometimes, it may be desirable to have a lower case letter invoke one command, and the same letter in upper case invoke another. For example, <ESC>a might invoke a particular command, and <ESC>A another one.

In these situations, the code is specified with a lower case *f*. The general syntax is:

fnn=<keystroke sequence>	where keystrokes typed are read literally as upper or lower case
Fnn=<keystroke sequence>	where keystrokes are case independent; either case can be typed by the user

In the above example, "nn" represents specific command code numbers as listed in the #COMMANDS section.

In the standard release the keystroke sequences are defined to be case independent because it makes Lyrix commands easier to use, as users don't have to bother with case. However, the second convention allows more command syntax possibilities, thus potentially improving mnemonics and allowing customization of function keys.

The keystroke sequence definition can be given in its ASCII decimal equivalent. These ASCII codes are found in the table in Appendix C. The codes are necessary for defining keystrokes that cannot be included literally, such as control characters.

Other codes can be used to specify particular command keystrokes:

Short Forms	Keystroke
$	Escape
T	Tab
U	Cursor Up
D	Cursor Down
L	Cursor Left
R	Cursor Right
-	Delete

These short forms can be used in combination with literal strings in single quote strings to make up complete entries. The first line of the standard #COMMANDS section uses both. Each entry in the first line is explained below.

By consulting #COMLOCATE, we find that *F01* corresponds to the *Delete character* command. The numeral *3* is the decimal numeric form of the ASCII character value for <CTL>C.

Note that the code explained above preserves Command File syntax by including a comma after each entry. The next entry, *F02=127*, translates to the *Erase previous character* command, the Rub or Delete key.

The entry, *F04=$-'DR'*, is the definition for the *Delete right* command. This entry uses the $ short form to specify the Escape key.

The next part of this entry can be quoted literally, and so is included in quotes as Command File syntax demands. The dash (–) is used to join definitions when more than one keystroke is used. The entire translation of this entry, then, is:

Delete right = <ESC> *DR* (or *dr*, as the upper-case F convention was used).

Furthermore, the same command can be assigned to different keystrokes in a single #COMMANDS section. For example, codes F12–F15 specify cursor movements using codes from the above table (L,R,U,D). Additionally, F12–F15 are also defined on the last entry line of #COMMANDS as 8,12,11, and 10, respectively. These codes are defined in the ASCII table in Appendix C as <CTL>H, <CTL>L, <CTL>K, and <CTL>J.

This preserves cursor motion conventions as used in other UNIX OS software and provides an alternative when arrow keys are unusable.

To assign a command to a function key, consult the manual for the terminal being used to obtain the ASCII code sequence generated by the function key or keypad key; this is then used in the definition of the Lyrix command. Find out what string is produced when each function key is pressed. For example, function key #1 on the Altos II™ terminal produces this string when pressed:

^[[L (^[is the control code for escape)

To assign the Insert line command to this key, for example, include this entry:

F09=$–'[L'

Function keys can also be combined with other keys to produce commands. For example, if a function key generates:

esc ?p

then the *Mark top left block* command can be defined as:

F52=$–'?pB'

and the *Mark top left serial* command as:

F71=$–'?pS'

This assigns the function key to mean "mark top left," and "B" or "S" must be typed afterward to indicate block or serial cutting.

Entries in the #COMMANDS section can be included in *free format*, that is, without regard to any particular order; however, the standard #COMMANDS section presents definitions in order according to function number, and is structured in tabular format.

#COMMANDS sections for specific terminals can be built using the syntax described above. The header must include the /etc/termcap name of the terminal which can be found in the user's environment. For example, the header of #COMMANDS section for the Altos II terminal would read:

#COMMANDS–TWO

TWO is the termcap code for the Altos II, and the dash tells Lyrix that this is a special terminal definition. Lyrix obtains the terminal type being used and matches it with this header to find the proper section. It uses the standard section (#COMMANDS) by default when no match is found.

Following simple rules of syntax, commands can be assigned to almost any set of keystrokes by simply editing the #COMMANDS section, and by developing other #COMMANDS sections for other terminal types as needed.

Summary

- By assigning keystrokes to commands in the #COMMANDS section, customized keystroke commands are possible.
- Separate Command Files can be developed to suit different needs. A central Command File can serve a group of users, while local ones can serve individuals.
- Command names are listed with their corresponding codes in the chart in #COMLOCATE.
- Keystrokes are defined by codes described in this chapter and listed in more detail in Appendix C. Literal strings can also be used in defining codes.
- Literal strings are entries surrounded by single quotation marks.
- Entries in the #COMMAND section can be entered in free format, but must be separated by commas if on the same line.
- #COMMANDS sections can be developed for specific terminals to allow full use of terminal capability.

CHAPTER 5
Menus

Sections defining menus, like all Command File sections, are easily edited. The #SYSMENU section, introduced previously, defines the Main Menu that is the gateway to the entire menu system. By modifying this menu, and creating others, menu systems can be developed to tailor Lyrix to individual requirements.

General Rules of Menu Definition

All Command File sections begin with the section header entry. As explained in Chapter 2, section headers must start with the crosshatch character in column one (technically, of the menu sections only the mandatory #SYSMENU and #EDITMENU sections must begin with the crosshatch; however, it is recommended that this convention be preserved in all sections for consistency). Refer to Chapter 2 for more about section headers.

As with all Command File sections, the last line of a menu section must contain two righthand parentheses starting in column one.

The second line of a menu section specifies the menu title as it appears on the screen. As with all literal strings in the Command File, this menu name is delimited with single quotation marks.

Between the second and last line, menu sections are comprised of a series of entries, each of which defines an option (or remark) of the menu. Each entry is made up of two parts, the first part is the *option* as it appears on the menu, the second is the *action* that Lyrix is to take when the option is selected.

The two parts are separated by an equal sign (=). The form for menu section entries is:

```
'Menu Option'=Action
```

As an example, consider the first menu option entry in #SYSMENU:

```
'1 - Word Processing Menu'=M('#DOCPREP')
```

The first part of this entry,

```
'1 – Word Processing Menu'
```

is the option as it appears on the menu. The option is surrounded by single quotation marks, as are all literal strings in the Command File.

Lyrix recognizes the first character of the Menu Option as the trigger that causes it to carry out the option. Here, the trigger is the number *1*; when the user presses *1* at this menu, Lyrix carries out the action that follows the equal sign.

Lyrix automatically converts lower-case characters received as triggers into upper-case, so that users need not concern themselves with shift keys. Lower-case option characters are not permitted within menus, but all upper case alphabetic characters, and other printable characters may be used.

The second part of the menu option entry is the *action definition*, which in the above example is:

```
M('#DOCPREP')
```

The first character following the equal sign, in this example, *M*, defines the *type* of action to be taken. These types of actions can be specified:

Action	Meaning
M	Jump to a menu
>	Comment line
X	Change directory
^	Go back to previous menu
?	List files or a directory
C	Create a file
E	Edit a file
D	Duplicate a file (*cp*)
R	Rename a file (*mv*)
K	Kill a file (*rm*)
*	Do this UNIX OS command line

The *M* action causes Lyrix to display the menu indicated in parentheses. Here, the #DOCPREP menu is specified:

```
M('#DOCPREP')
```

Information necessary for a particular action is included in parentheses and is delimited by quotation marks. If more than one piece of information is necessary, each is separated by commas and single quotes.

Some actions do not require any information. For instance, consider this entry from the #SYSMENU section:

```
'D – Change Directory'=X()
```

Here, the *X* defines the action, change directory. The option, as used in the above example, causes Lyrix to prompt for a directory name. Since no extra information is required, the parentheses are empty.

In the remaining part of this chapter, the ten menu action definitions are described, using examples from standard Command File menus.

M Action: Jump to a menu

This action provides the means to link menus together, with options on one menu invoking other menus.

Examples of menu jumping from #SYSMENU are highlighted here:

```
#SYSMENU
'*** Main Office System Menu ***'
'1 – Word Processing Menu'=M('#DOCPREP')
' '=>()
'2 – File Management Menu'=M('#FILEMANAGE')
' '=>()
'3 – Mail System Menu'=M('#MAIL')
' '=>()
'4 – Additional System Usage Menu'=M('#SYSCOMMS')
' '=>()
'D – Change Directory'=X()
'P – Printing System Menu'=M('#PRINTING')
'L – List Files'=?('*')
'? – HELP'=M('#HELP')
' '=>()
'* – Leave Lyrix'=*()
3=C()
5=E()
18=*('mail','','')
24=*('^','11','P')
2=*()
))
```

In menu action definitions, the section header name of the menu being jumped to is included between parentheses and single quotes.

In cases where Lyrix is unable to find or read the menu specified between parentheses, this message from the #MESSAGES section is displayed:

> Selected menu not available, or incorrectly entered in Command File.

If you receive this message, make sure the menu header is correctly stated and entries specified for the menu are correctly punctuated.

This message, like all Lyrix messages can be rewritten; see Chapter 7 for details.

> Action: Comment line

The Comment Action is used to enhance the visual appearance of menus by insertion of either blank or text lines.

For example, #SYSMENU uses the Comment Action several times to cause blank lines between sections of the menu:

```
#SYSMENU
'*** Main Office System Menu ***'
'1 – Word Processing Menu'=M('#DOCPREP')
' '=>()
'2 – File Management Menu'=M('#FILEMANAGE')
' '=>()
'3 – Mail and Communications Menu'=M('#MAIL')
' '=>()
'4 – Additional System Usage Menu'=M('#SYSCOMMS')
' '=>()
'D – Change Directory'=X()
'P – Printing System Menu'=M('#PRINTING')
'L – List Files'=?('*')
'? – HELP'=M('#HELP')
' '=>()
'* – Leave Lyrix'=*()
3=C()
5=E()
18=*('mail','','')
24=*('^','11','P')
2=*()
))
```

Note that a single blank space is required between the quotation marks of the menu option part of the definition when specifying blank lines, as Lyrix does not allow a null string as a menu line.

The Comment Action can also be used to add text lines to menus. For example, #FILEMANAGE uses the Remark Action in this way:

```
#FILEMANAGE
'*** File Management Menu ***'
' ----- Files ----- '=>()
'1 – Copy a file '=D()
'2 – Erase a file '=K()
'3 – Rename a file'=R()
'4 – File Checking Menu'=M('#CHECKER')
' '=>()
' ----- Directories ----- '=>()
'5 – Open a new directory'=*('mkdir ^','7','P')
'6 – Remove an empty directory'=*('rmdir ^','7','P')
'7 – Change directory'=X()
'8 – List directory'=*('ls *','','P')
' '=>()
'P – Printing System Menu'=M('#PRINTING')
'? – HELP'=M('#HELP')
' '=>()
'Press <Escape> to go back a menu'=>()
27=^()
3=C()
5=E()
18=*('mail','','')
24=*('^','11','P')
2=*()
))
```

Two of the comment lines, underlined above for emphasis, visually improve the menu:

```
' ----- Files -------'=>()
' ----- Directories ----- '=>()
```

Note that the parentheses in the action part of the definition are always left blank when using the Comment action.

^ Action: Return to Last Menu

This Action makes Lyrix return to the previous menu. The inclusion of this action definition permits the user to return back up the chain of menus following

exactly the route taken on the way down.

The second benefit of this Action is that, when combined with appropriate Jump to Menu options, all menus can be reached at any level and by many different menu branches.

Actions That Involve Files

Seven menu actions provide access and information about files. Using these actions you can manipulate files, change directories, and obtain listings of directories.

X Action: Change Directory

This action can be included at any menu to change directories without leaving Lyrix.

An example taken from #SYSMENU was introduced earlier in the chapter:

```
'D – Change Directory' = X()
```

The user is prompted by Lyrix for the pathname of the requested directory.

? Action: List Directory

This action utilizes the UNIX OS utility *ls* to examine the contents of the current directory. The data is fed back from the *ls* command automatically into Lyrix, where it is presented neatly in screen-sized pages, with pauses in between.

The action must be defined as in the #SYSMENU section:

```
'L – List Files' = ?('*')
```

C Action: Create a File

This action causes Lyrix to prepare for file creation. Here is an example of this Action from the #DOCPREP menu:

```
'1 – Create a new file'=C()
```

When the parentheses are left blank, as above, the user is prompted for a filename, then Lyrix verifies that a file or directory of the same name does not already exist. If a file of the same name does exist, Lyrix displays this message:

```
Not a good file name!
```

and the user is prompted for another file name.

If the parentheses are not left blank, Lyrix will attempt to create a file of the same name as that listed in the parentheses. If it cannot, it will give you the *Not a good file name!* error message and will prompt you for a different filename.

> ■ **NOTE:** Messages shown in this guide are taken from the standard sections, #MESSAGES and #USER. These messages can be changed as desired; refer to Chapter 6.

E Action: Edit a File

This Action tells Lyrix to edit a file. Syntax and operation are virtually identical to the C Action described above. Here is an example from the #DOCPREP menu:

```
'2 – Edit an existing file' = E()
```

When the parentheses are left empty, as above, the user is prompted for a filename, then Lyrix verifies that the file exists. If the file does not exist in the current directory or the specified path, Lyrix displays this message:

```
Not a good file name!
```

Again, as with the C Action, when a file name is specified in the parentheses, Lyrix bypasses the prompting routine and enters the file immediately (assuming it exists).

The following is a useful way to use this Action, as users find themselves editing certain files often. A good example is:

```
'2 – Edit the Command File' = E('.wprc')
```

This option provides rapid access to the Command File named in parentheses.

D Action: Duplicate a File

This action provides an easy means of making copies of files. During the specification of the file name, file name verification and error handling are parallel to the previous options described above.

This example is from the #FILEMANAGE section:

```
'1 – Copy a file ' = D()
```

The user is prompted for the names of both the original and the new file.

> ■ **NOTE:** This Action only copies files for which the user has read permission. Also, the copy is made only if the user has write permission in the specified directory.

R Action: Rename a File

This option allows the user to rename a file without leaving Lyrix using the *mv* command.

The user is asked for the original name of the file. Then, after this has been accepted by Lyrix, the new name is requested.

Permissions and the names for the original file and the new name are checked for existence by Lyrix to provide file protection.

This example is from the #FILEMANAGE section:

```
'3 – Rename a file' = R()
```

K Action: Kill a File

This Action allows the user to remove a file without leaving Lyrix.

A file name is requested and its existence verified. As with other options, the user is notified of an unusable filename.

After verification the user is asked to confirm the erase action:

```
Enter "*" to confirm or <Return> to abandon:
```

■ **NOTE:** This Action only erases files for which the user has write permission.

* Action: External Function

This Action provides access to the UNIX OS command line, and as such, is one of the key ingredients to Lyrix customization.

By including information in parentheses, any UNIX OS call can be executed. The general form is:

```
=*('UNIX Command Line'','#USER Message Number','Pause/Return to Menu')
```

These three qualifiers must appear in the above order only, separated by commas and included between single quotation marks. Spaces between quotes, commas, and parentheses are not allowed.

Consider this example:

```
'1 – Where am I'=*('pwd','','P')
```

In this example, the second qualifier is not being used, so it is specified as a null string. The third qualifier tells Lyrix either to pause before returning the user to the menu (P), or return directly without pause (R). When a pause is specified as above, this message is displayed before returning the user to the menu:

```
Press <RETURN> to continue:
```

This preserves on screen any information provided by the call.

■ **NOTE:** The system defaults to the P qualifier when none is specified, thus the above example could be written as,

```
'1 – Where am I'=*('pwd','','')
```

Sometimes, no useful information is provided by a system call, so there is no need to tell the system to pause before returning the user to the menu. For example,

```
'2 – Remove back-up file'=#('rm backup','','R')
```

This tells Lyrix to perform the specified command, in this case remove a file, and return (R) the user to the menu directly.

The second qualifier is used when some user input is required to carry out the specified system call. It specifies the number of a message in the #USER section. This message is then displayed (the standard #USER section is listed in the Command File in Appendix A).

For example,

```
'3 – Rename back-up file'=#('mv backup ^','12','R')
```

The caret (^) following the UNIX OS command tells Lyrix to prompt the user for input. The '12' tells Lyrix to print the 12th message in the #USER section, which describes to the user the type of input that is necessary.

You may use more than one system call at a time. If you do, separate the individual UNIX OS commands with a semicolon. However, only one #USER message may be displayed.

The messages in #USER can be changed. If you change their order, however, be careful to renumber the menu options. For instance, in the above example, if the #USER message number 12 were moved to a different position in #USER, its new number should be placed between the second set of quotes.

Also, whenever the caret symbol is used, it is good practice to enclose it in double quotation marks, to avoid possible misinterpretation by the UNIX OS shell. For instance, the space between an argument consisting of two words could be interpreted as two separate arguments. Thus, the last example would be more properly written as,

```
'3 – Rename back-up file'=#('mv backup "^"','12','R')
```

A *double* caret (^^) indicates that the Lyrix *Point and Pick* facility may be used for file name selection. (Point and Pick allows the user to select a file name visually from a table of names, and is fully described in the Lyrix *User's Guide*).

For example,

```
'3 – Look at a file before printing'=#('fprint -t "^^" | more -d','12','')
```

This menu option tells Lyrix to expect user input and to prompt for it by displaying message number 12 in #USER, which is *Enter name of file:*.

Additionally, since the double caret was used instead of a single one, Point and Pick is available for specification of the file name when the user selects this option.

The * Action can also be used to provide the option of leaving Lyrix, as in this example from #SYSMENU:

```
'* – Leave Lyrix'=*()
```

Should the user's shell have been arranged to run Lyrix immediately on login and to log out immediately after leaving Lyrix, then this Action could be used to log the user off the computer the computer upon leaving Lyrix, as in this example:

```
'* – Log off of the computer'=*()
```

Lyrix assumes that you want to terminate when this action is used without information between the parentheses, as in the last two examples.

Hidden Options

Lyrix provides a useful way of making itself accessible to a variety of users simultaneously. Often, more advanced users are constricted by menu hierarchies. They usually know exactly what they want from Lyrix and would prefer to accomplish it immediately, without benefit of menus.

For such users, ***hidden options*** are available on menus. Hidden options are not displayed as menu options and are invoked using non-printing characters involving control sequences. Hidden options are specified in menu sections in ASCII decimal form. For example, here is a hidden option from #SYSMENU:

```
3=C()
```

The numeral 3 is a code for <CTL> C, which is what the user types to select the option, in this case, Create a File. A table of characters and their equivalent

ASCII codes is in Appendix C.

You can designate words to appear on the screen when you call a hidden option by enclosing the phrase in single quotes as follows:

```
3-'Create a File option chosen'=C()
```

In the above example, the message *Create a File option chosen* will appear at the bottom of the Lyrix menu when the option is called.

Using Hidden Options, direct access to Lyrix facilities can be provided without disturbing the menu environment important to many users.

Providing Help

Two menus exist in the standard Command File to provide help: #HELP and #COMSUM. The standard #HELP section looks like this:

```
'*** Help System Menu ***'
'1 – Command Summary – Quick Reference'=#('more -d /usr/lib/wp/wp.help',"","")
'2 – Brief Command Summaries Menu'=M('#COMSUM')
' '=>()
' ----- On-line Primer options -----'=>()
' '=>()
'3 – Using the Word Processor'=#('more -d /usr/lib/wp/prim1',"","")
'4 – Using Menus'=#('more -d /usr/lib/wp/prim2',"","")
'5 – Using Files'=#('more -d /usr/lib/wp/prim3',"","")
'6 – Using Commands'=#('more -d /usr/lib/wp/prim4',"","")
' '=>()
'Press <Escape> to go back a menu'=>()
' '=>()
27=^()
```

This menu section shows how help is presented to the user by using the UNIX OS *more* command to display help files at the user's terminal. The –d option of more results in a more explicit prompt to the user. #COMSUM is a sub-menu that displays help on more specific topics.

Note that these two lines of #HELP are related:

```
'Press <Escape> to go back a menu'=>()
27=^()
```

The former is a comment that tells the user that an escape jumps back a menu. The latter tells Lyrix to execute the action.

As Lyrix is modified to suit individual needs, the Help system can be adjusted accordingly.

Document Escapes

The #EDITMENU section defines the menu seen when the user gives the *Escape to Menu* command from within the word processor. The standard section looks like this:

```
'*** Document Menu ***'
'1 - Comprehensive Help Menu'=M('#HELP')
'2 - UNIX Command Line '=*('^','11','P')
'3 - Check mailbox'=*('mail','','')
' '=>()
'Press <Escape> to get back to the document'=>()
' '=>()
27=*()
```

■ **NOTE:** Although this menu section can be edited to provide almost any Lyrix or UNIX OS facility, it should be used with caution. Certain activities should NOT be attempted from the #EDITMENU. For instance, a second file should NEVER be opened using Lyrix while escaped from the original file.

As such, the *C* (Create file) and *E* (Edit file) Actions are not accepted as entries in the #EDITMENU section.

Summary

- The #SYSMENU defines the Main Menu that is the gateway to the entire menu system.
- The first line of a menu section is the header that identifies the section; the second line is the menu title as it appears to the user.
- Menu section entries define an option (or a comment) of the menu. Each menu entry is made up of two parts, the *option*, and the *action* that Lyrix takes when the option is selected.
- The first character of the option is the trigger that causes Lyrix to carry out the action.
- Lyrix automatically converts lower case characters to upper case (when used as menu selectors), so that users need not concern themselves with the shift key.
- Here are twelve distinct actions that Lyrix can carry out:

  1 The *M Action* causes Lyrix to jump to the menu specified.

  2 The *> Action* causes Lyrix to display the specified remark (which may be a blank line).

  3 The *X Action* causes Lyrix to change directories.

  4 The *? Action* causes Lyrix to list the current or specified directory.

  5 The *C Action* causes Lyrix to create a file.

  6 The *E Action* causes Lyrix to edit a file.

  7–9 The *D*, *R*, and *K* options cause Lyrix to duplicate (*cp*), rename (*mv*), and remove (*rm*) a file, respectively.

  10 The *^ Action* causes Lyrix to return to the previous menu.

  11 The ** Action* causes Lyrix to carry out an operating system call.

  12 The ** Action* causes Lyrix to terminate when the parentheses are left blank.

- The form for the * *Action* is:
  =*('UNIX Command Line','#USER message number','Pause/Return to menu')
- The second qualifier in the above definition example tells Lyrix to print the indicated message from the #USER section. When not used, this qualifier must be included as two adjacent single quotes.
- When a message is specified as above, a caret (^) must be included following the first qualifier to let Lyrix know that user input is expected.
- A double caret (^^) is used to tell Lyrix to allow the user to select a file using Point and Pick, rather than the routine prompt.
- When either a single or double caret is used, it should be surrounded by double quotes to prevent misinterpretation by the UNIX OS shell.
- The third qualifier tells Lyrix to pause before returning the user to the menu (P), or to return directly to the menu (R).
- Hidden options can be included in any menu to provide quick access for more advanced users.
- System help is provided by using menus to display help files at the user's terminal; the main help menu is labelled #HELP. #COMSUM is a sub-menu that provides help on specific topics.
- The #EDITMENU section defines the menu seen when the user gives the *Escape to menu* command. Other files must not be edited while escaped from the original file.
- The number of options on a menu is limited by screen display size.

CHAPTER 6
Messages

All system messages (with the exception of those used in menus, see Chapter 5), are loaded from the #MESSAGES section:

```
#MESSAGES
'[?] Select an option'
'Reserved for future use'
'Selected menu not available, or incorrectly entered in command file.'
'Press <RETURN> to continue:'
'Unable to run system command.'
'Not a good file name!'
'  selected.'
'No write permission! '
'Please enter the file name or "*" to return to menu '
'Copying file to '
'Unable to open file for edit.'
'Cannot create new backup file.'
'Cannot write to file '
'*** File copy ***'
'*** File re-name ***'
'*** File erase ***'
'Requested task is now complete.'
'Operation FAILED.'
'Original file'
'New file'
'Enter "*" to confirm or <RETURN> to abandon:'
'Merge file section not found.'
'Enter name of file:'
'Enter string to search for:'
'Option ? '
'Sorry but that search failed.'
'Enter "a"=all files, "l"=statistics, "s"=short'
'Unable to list selected directory'
'Enter "*" to exit or <RETURN> for more '
'*** List files ***'
```

'*** External function ***'
'Busy saving the document.....'
'*** Edit a file ***'
'*** Create a file ***'
'"–" to hyphenate or <RETURN> to skip: '
'Enter "*" to replace or <RETURN> to skip '
'Enter new characters or <RETURN> to delete '
'Enter "*" for global or <RETURN> for interactive '
'Invalid command entered'
'Enter "*" to confirm quit or <RETURN> to continue '
'Busy executing command'
'Enter <RETURN> for next or "*" to quit '
'Enter ruler number (0-9)'
'Enter (B)lank, (L)eave, (R)emove or (S)ave'
'Enter (O)verlay, (E)lbow or (I)nsert'
'Enter 1 (tab), 2 (hyphen), or 3 (insert):'
'Reserved for future use'
'Enter <RETURN> for next page,"*" <RETURN> to quit'
'INSERT'
'HYPHEN'
'TAB'
'Current directory : '
'Getting your directory listing'
'Directory, enter "*" to change directory'
'No write permission, enter "*" for read only '
'Sorry – no read permission on this file'
'Move cursor to table of names or enter a file/directory name'
'*** Change directory ***'
'Enter a directory name then press <RETURN>'
'Sorry – cannot change to requested directory'
'Enter page number, <RETURN> :'
'File exists – enter "*" to overwrite or <RETURN> to abandon '

```
'Press "*" for more '
'Reserved for future use'
'Enter (B)lock or (S)erial'
'Enter (D)ocument or (P)aragraph'
'Enter (I)nsert or (O)verlay'
'Enter ruler number (0-9) or "." for current'
))
```

You can change these messages as desired; they must remain in the same order as above.

CHAPTER 7
Rulers

The #RULERS section specifies ten standard rulers, numbered 0 through 9, which are available to the user to control formatting. You can access the standard rulers with the *Recall ruler* and *Use ruler* commands.

The default rulers in the standard Command File are:

Ruler 0

```
L.....T.....T.....T.....T.....T.....T.....T.....T.....T...R.
```

Ruler 1

```
L.....T.....T.....T.....T.....T.....T.....T.....T.....T...J.
```

Ruler 2

```
.....L.....T.....T.....T.....T.....T.....T.....T.....T...R.
```

Ruler 3

```
.....L.....T.....T.....T.....T.....T.....T.....T.....T...J.
```

Ruler 4

```
.....L.....T.....T.....T.....T.....T.....T.....T.....T...R.
```

Ruler 5

```
.....L.....T.....T.....T.....T.....T.....T.....T.....T...J.
```

Ruler 6

```
.....L.....T.....T.....T.....T.....T.....T.....T...R.
```

Ruler 7

```
.....L.....T.....T.....T.....T.....T.....T.....T...J.
```

Ruler 8

```
T.....L.....T.....T.....T.....T.....T.....T.....T.....T...J.
```

Ruler 9

```
T.....T.....T.....T.....T.....T.....T.....T.....T.....T...T.
```

■ **NOTE: The last ruler, Ruler 9, is 252 characters in length—too long to be fully displayed here. This ruler as it is displayed in the .wprc file may wrap around your screen, therefore appearing to take up two or more lines in the #RULER section.**

The #RULERS section can be changed so that rulers for special formatting requirements can be accessible to the user. The rulers as shown above are position dependent; the first one is always Ruler 0, the second position is always Ruler 1, and so on.

Note that if you replace any ruler characters in #RULERS with other characters, you must also specify them in the RULER entry in the #SYSTEM section, (see Chapter 3).

Note also that Ruler 0 is the default ruler when a file is entered.

CHAPTER 8
Printer Effects

The Lyrix module *Tcap* contains codes that control how printer effects are displayed on the particular terminals being used. Tcap also includes the codes needed to enable different printers to correctly carry out printer effects. Each Tcap entry specifies codes for a particular type of terminal or printer. Tcap is fully described in Chapter 9.

The #EFFECTS section of the Command File defines the print effect prompt that the user sees when the *Start effect* command is given. The standard #EFFECTS section looks like this:

```
#EFFECTS
A=Bold
B=Double Strike
C=Underline
D=Underline text only
E=Bold & Underline
F=Superscript
G=Subscript
H=Condensed(DMPs)
I=Compressed (DMP200)
J=Elongation(DMPs)
K=Correspondence 10 pitch
L=12 pitch
M=Italics (DMP110)
))
```

Each line of the #EFFECTS section can contain only one entry; note that quotes are not allowed as delimiters in this section.

Up to 26 effects can be included, using the capital letters *A* through *Z* as triggers. Letters *A* through *E* must be reserved for the effects noted above. Otherwise, entries can be arranged in free format, (one per line) appearing on the user's screen in the order in which they are included in #EFFECTS.

If a particular effect is not listed in #EFFECTS, it cannot be used, even if it is listed in Tcap.

Summary

- The Lyrix Tcap module contains codes that control how print effects are displayed at the terminal and on the printer. Each Tcap entry specifies codes for a particular terminal or printer. Tcap is fully described in Chapter 9.
- The #EFFECTS section of the Command file defines the print effect prompt that the user sees when the *Start effect* command is given.
- Each line of #EFFECTS can contain only one entry.
- Quotes are not needed as delimiters in this section.
- Up to 26 effects can be included, using the capital letters *A* through *Z*.
- Entries can be arranged in free format, appearing on the user's screen in the order in which the effects are included in #EFFECTS.
- Entries A–E must be reserved for the standard definitions.

CHAPTER 9
The Tcap File

The *Tcap file*, resident in /usr/lib/wp, includes printer effect codes used by each different kind of printer and terminal in the system. To learn more about your printer or terminal's special capabilities, refer to the documentation provided with them.

Lyrix expects to find the file */usr/lib/wp/Tcap* or as directed by a Redirect file. If it does not, or if there are errors in this file, the message

cannot read Tcap file

appears, and Lyrix terminates. If special effects are not needed, the Tcap file should be present but empty.

A separate section is reserved for each different type of printer and terminal. Each Tcap section defines the codes necessary to produce the effects on the specific terminal or printer. The actual effects being used in the system are named in the #EFFECTS section of the Command File, see Chapter 8.

If a code is represented in the #EFFECTS section, but not defined in Tcap for the terminal in use, the standard terminal standout mode (as defined by the termcap entries *so* and *se*) is used to represent the effect on the video terminal.

For the printer, if no sequences are given for the effects A-E then Fprint will produce the effects in a standard way internally in the software. If no other controlling entries are present and text needs to be bolded, the text will instead be triple struck. Any other sequence needed by the printer to produce an effect that is not present in Tcap (i.e., effects F-Z) is ignored, leaving text on the line unaffected.

Defining Printer Effect Codes

Each section of the Tcap file is indicated by the name of the printer or terminal for which effect codes are being described. The name is preceded by the pound

sign (#), as is the convention for sections of the Command File. Thus an effect definition section header of Tcap might appear as:

```
#diablo
```

or

```
#vt100
```

These labels correspond to those used in both the #COMMANDS section of the Command File (e.g. #COMMANDS–vt100) and in the termcap database. The #diablo label, in this example, defines a printer, and #vt100 defines a terminal.

Each defined effect takes the form:

```
<code>=<Sequence to start effect>,<sequence to end effect>
```

where *<code>* is a capital letter A–Z.

A sample definition might look like:

```
A=27-93-65,27-93-91
```

In this example, the ASCII sequence 27–93–65 starts print effect A and 27–93–91 stops it. Using the ASCII table in Appendix C, we find that 27–93–65 corresponds to ^[ ] A, and 27–93–91 translates to ^[ ] [.

Syntax Rules for code definitions in Tcap correspond to those used in the Command File. Note that literal strings may not be used for defining effects strings A–Z (see Chapter 8). Literal strings may, however, be used in defining MAP strings (see below). For example,

```
#test
MAP='i','e'-8-''
))
```

works just as well as its corresponding ASCII sequence

```
#test
MAP=124,101-8-96
))
```

■ **NOTE:** *Standout mode* is used for a terminal when a particular printer effect is not defined in Tcap, or if the terminal is not defined in Tcap at all. This assumes that standout mode for the terminal is defined in /etc/termcap.

Using Optional Switches

Two sets of switches, one for printer definition, one for terminal definition, can also be included within a Tcap section. These switches provide extra control over peripheral devices.

Six terminal options are available; when used, each should appear on a separate line.

ns Do not use terminal standout mode. This option is a must if the standout mode places a phantom blank space (called an *attribute byte*) before the string to be highlighted.

nt Do not use terminal insert line/delete line capabilities. This can be useful where scrolling is faster if performed by Lyrix than by insert/delete line at the terminal.

nu Cursor motion sequences may send null characters to screen. This option invokes special routines in Lyrix version-3 that bypass this problem.

rw Place terminal in *RAW* mode rather than *CBREAK* mode. This is allied with the *nu* option, though some terminals which need *nu* do not need to be in raw mode. If this option is present, the *Terminate command* feature of the editor is lost.

co Use full screen width for printing. The default in Lyrix version-3 is terminal width minus one since many terminals scroll if a character is printed in the rightmost column, particularly on the last line. This option should only be used if the terminal does not scroll when a character is written in the lower rightmost column position.

Five printer options can be used; again, when used, each should appear on a separate line, followed by the necessary code(s).

PITCH This entry is used if the printer itself does not have the capability to bold text by means of a code sequence. This is equivalent to the *PITCH1* and *PITCH2* locations in implementations of Lyrix version-2. The syntax is: PITCH=<sequence to set character pitch small>,<reset sequence>

FF This option is used to cause the blank lines at page ends to generate a form-feed. This is sometimes required when using cut-sheet feeders. The syntax is: FF=<sequence to effect a form-feed> e.g., FF=12

INIT The code entered here is sent to the printer just prior to printing the document and can be used for a number of purposes such as printer reset. The syntax is: INIT=<sequence to initialize the printer>

DEINIT The code defined here is sent to the printer after the document printing has been completed. The syntax is: DEINIT=<sequence to reset the printer>

MAP Two sequences can be specified for each MAP location and there may be up to 24 different MAP's for each printer entry in Tcap.

Whenever the from-sequence is located in the document the to-sequence is substituted and sent to the printer. This can be used to drive special character sequences across to the printer from the document such as might be required for graphics or other controls. This switch is convenient for languages other than English, where a letter may consist of two characters. The syntax is: MAP=<mapped-from sequence>,<mapped-to sequence>

For example, the following entry shows how to map a single character (the pipe "¦" symbol) to an "e" with a back accent (" ` ") over it:

```
#diablo
MAP=124,101-8-96
))
```

According to the ASCII table, 124 represents the pipe symbol, 101 is "e", 8 is backspace, and 96 is a back accent.

If you have a "smart" printer with an alternate character set, the MAP sequence after the comma could also be defined as as an escape sequence to turn on the alternate character set, print a character, and turn off the character set.

ALLOW Allows mapped characters to be processed as alphabetic when they are hyphenated.

Example Tcap entry

This is a sample entry for the Radio Shack Dot Matrix Printer, defined by the printer label "DMP". This printer has capabilities such as superscripting and elongated type.

```
#DMP
A=27-31,27-32
C=15,14
E=27-31-15, 27-32-14
F=27-30, 27-28
G=27-28, 27-30
H=27-20, 27-19
I=27-23, 27-19
J=27-14, 27-15
K=27-18, 27-19
L=27-23, 27-19
M=27-66, 27-19
INIT=20-27-19-27-21-14
DEINIT=19-27-22
))
```

Summary

This section described how the Tcap file works in conjunction with the Command File to define printer effects for the particular terminal(s) and printer(s) being used.

- Tcap is resident in /usr/lib/wp; if Lyrix cannot find it here, if no Redirect file exists, if the user does not have read permission, or if there are errors in the file, Lyrix terminates with the message

  cannot read Tcap file.

- A separate section is used for each different type of printer or terminal.
- Each Tcap section describes the codes necessary to produce each effect on the particular terminal or printer specified.
- The actual effects being used in the system are named in the #EFFECTS section of the Command File.
- Standard terminal standout mode (termcap entries *so* and *se*) is used to represent effects on a terminal not defined in Tcap, or for any effects not defined for a terminal.
- For the printer, if no sequences are given for effects A-E, then Fprint produces the effects in a standard way internally in the software.
- Any other sequence needed by the printer to produce an effect not included in Tcap is ignored, leaving text unaffected.
- Syntax rules for code definitions in Tcap correspond to those used in the Command File.
- Two sets of optional switches (one for terminals, one for printers) can be included in Tcap sections.

CHAPTER 10
The Fprint Module

The Fprint filter module controls print time variables, such as print effects (bold, etc.), paging, and other printer dependent factors.

fprint is a binary executable file normally resident in the directory */usr/bin* or */bin*; however, this is system specific and may vary.

Along with full user-definable printer effects, options include the following:

–i	— Interactive page-by-page print
–m	— Set left hand margin
–a	— Print alternate pages for easy collation
–S	— Set first page to print
–E	— Set last page to print
–f	— Set form length
–c	— Indicates number of copies to print
–p	— Specifies printing device
–t	— Previews file at the terminal

Each of these features is described below.

Fprint can be run as a stand-alone program to facilitate easy option testing, prior to inclusion in the menu system. Note that if you use the Fprint command at the UNIX OS command line, it must be spelled with a lower-case "f".

Interactive Page-By-Page Print

This option allows the Fprint module to drive a sheet-by-sheet printer.

Fprint halts just before beginning to print each page. This message is printed on the screen

Enter <RETURN> for next page, "*" <RETURN> to quit

and the following two options are offered:

Press <Return> to print next page	activates Fprint and the next page is sent to the printer.
Press "*" <Return> to abandon printing	terminates printing and execution of Fprint.

The latter option will normally be used in conjunction with the standard output re-direct as spooling overrides the interactive process.

Syntax Example:

```
fprint -i filename > (printer device name)
```

Menu Option Example:

```
'P – Print interactively'=*('fprint -i "^^"','12','R')
```

Set Left Hand Margin

This option is intended to provide an offset in the output stream suitable for driving cut-sheet-feeders (although it also works with other printers).

Syntax Example:

```
fprint -mnn filename          where "nn" is the offset.
```

Menu Option Example:

```
'P – Print with margin'=*('fprint -m15 "^^" ','12','R')
```

Print Alternate Pages for Collation

This Fprint option causes printing of alternate pages. This can be used for easy collation of printed work for photocopying.

The option is often used in conjunction with the *set first page to print* option. An example of this is given below:

Syntax Example:

```
fprint –a filename
```

Menu Option Examples:

```
'P – Print odd pages'=#('fprint –a "^^" ','12','R')
'P – Print even pages'=#('fprint –a –S2 "^^" ','12','R')
```

Set First Page to Print

Here, Fprint can be told to begin printing at a page number other than "1." This can be used together with the *Set Last Page to Print* option to print only a certain range of pages.

The option is also useful with the *Print Alternate Page* option to cause printing of the even pages as is shown in the examples in the description of that option above.

Syntax Example:

```
fprint –Snn filename          where "nn" is the last page to start
```

Menu Option Example:

```
'P – Print even pages'=#('fprint –a –S2 "^" ','12','R')
```

Set Last Page to Print

This option is complementary to the *Set First Page* option and specifies the last page that Fprint is to send.

Syntax Example:

```
fprint –Enn filename          where "nn" is the page to end
```

This option and the previous one can be combined to print a range of pages. For example,

```
fprint –S4 –E6 advent
```

will print the file "advent" starting with page 4 and ending with page 6. Note that the number assigned to the "E" argument must be greater than that assigned to "S".

Number of Copies

You can request more than one copy of the file to print using the –c option followed by the number of copies requested. The syntax is:

```
fprint –cnn filename          where nn is the number of copies
```

This command prints the file in its entirety the specified number of times.

Form Length

You can specify a form length different from the default (66 lines) with the –f option, followed by the number of lines on the form. The syntax is:

```
fprint –fnn filename          where nn is the form length in lines
```

Reset Page Number

Fprint begins numbering each file with 1 rather than sequentially when multiple files are being printed. The syntax is:

```
fprint –r filenames
```

Reset Headers and Footers

Prints headers and footers specified for each separate file, rather than carrying over headers and footers from the first file when multiple files are being printed. If no headers or footers have been defined for the print run, this flag is not necessary. The syntax is:

```
fprint –h filenames
```

Specifying Printer Device

This option specifies the printing device in order for Tcap to be accessed. If the –p flag is not specified, the standard print effects A–E will be supported. The syntax is:

```
fprint –p printername filename
```

Where *name* is the name of the printing device, as specified in the Tcap file.

Previewing at the Terminal

Files can be viewed at the terminal by typing:

```
fprint filename
```

The output defaults to standard (the terminal). However, any print effects are stripped out. To preview a file with print effects, include the –t option:

```
fprint –t filename
```

The print effects are displayed according to the conventions established for the particular terminal being used as defined in the user's environment.

You can cause Fprint to print multiple files by simply including all the file names separated by spaces.

From a UNIX OS command line, all shell expansion characters, such as the asterisk (*) may be used.

Is your file is more than 23 lines, stop the output using <CTL>s and <CTL>q, or pipe the output through a paging program such as *more*.

Summary

This section described how the Fprint module works to print files.

- You can use Fprint to control sheet by sheet printers as well as continuous form printers.
- Fprint can print any specified pages of a file instead of the entire file.
- Using other flags, you can specify number of copies of the file to print, form length, and page numbering.
- Headers and footers can be carried over when printing multiple files, or respecified for each file.
- Print effects can be previewed at the terminal.

CHAPTER 11
The Viewprint and Mail Merge Screens

You can access the Viewprint screen from the Printing System menu or from the UNIX OS command line. Viewprint provides special options for the user at print-time, such as printing only a portion of a file or changing form length. Viewprint invokes the *Fprint* program.

You can also access Mail Merge from the Printing System menu or the command line. It is a screen-driven function that allows the user to merge text files with files containing variables in order to create form letters. It invokes the *Rmerge* program.

For more information about what Viewprint and Mail Merge do, refer to the *Lyrix User's Guide*.

Filling in the Viewprint and Mail Merge screens creates an Fprint or Rmerge command line, respectively. In this chapter, we will discuss the files which contain and define the Viewprint and Mail Merge screens, their structure and contents, and how to edit and recompile them to conform to your system's needs.

The file containing the Viewprint screen is called *uprint*. The Mail Merge screen is in *vmerge*. These are the uncompiled editable files. You will find them in /usr/lib/wp.

Structure of the vprint and vmerge Files

The structure of vprint and vmerge is the same. Each is divided into three sections, #DEFINE, #SCREEN, and #FIELDS. Each section is aligned against the left edge of the screen, begins with a crosshatch (#) character, and ends with two right parentheses.

The #DEFINE section lists the Fprint or Rmerge flags available for the screen in use. #DEFINE sets the field parameter characters and also defines the escape sequences which allow the user to ask for help, to quit the Viewprint or Mail Merge screen, or to accept the current screen.

The #SCREEN section defines what the Viewprint or Mail Merge screens actually look like. #SCREEN assigns each field a lower-case alphabetic character which is defined in the third section, #FIELDS.

#FIELDS section assigns certain code characters to the alphabetic letters designated in the #SCREEN section. These characters specify the type of information the user must enter in a particular field when filling out the Viewprint or Mail Merge screens.

You will find complete copies of the standard vprint and vmerge files in Appendix B. You may wish to look at these before continuing to read this chapter in order to become familiar with their organization and content.

The Standard #DEFINE Section

This is the standard #DEFINE section of vprint:

```
#DEFINE
-p @b -c@h -S@f -E@g -f@e -m@d @i @j @k @a @c &
FIELDS = [_]
HELP=$-'h'
GO=$-'a'
QUIT=$-'q'
))
```

This is the standard vmerge #DEFINE section:

```
#DEFINE
-b @c -n@k -f'@g'-r'@h' @d @j @e @f @a < @b @i @l &
FIELDS = [_]
HELP=$-'h'
GO=$-'a'
QUIT=$-'q'
))
```

The first line of the #DEFINE section represents an actual command line with represented fields providing the necessary arguments. The characters preceded by the "at" sign (@) in the first line refer to specific fields in the #SCREEN section which are in turn mapped to Fprint or Rmerge flags in the #FIELDS

section. The #SCREEN and #FIELDS sections are discussed later in the chapter.

The second line (FIELDS = [_]) defines the #SCREEN field parameter characters, in this case, square brackets, and the character that fills the space between the two parameters, in this case, the underbar character.

The third, fourth, and fifth lines of the section define the escape sequences available to the user filling out the screen. The user can ask for help (HELP=$-'h'), accept and execute the screen, (GO=$-'a'), or quit the screen (QUIT=$-'q').

A brief summary of the Fprint and Rmerge flags follows:

Fprint flags

For more detailed explanations of the Fprint flags and the Fprint module, refer to Chapter 10.

-p printer name	Accesses Tcap file for print effects (see Chapter 9).
-c nnnn	Specifies the number of copies to printed. Preset value is one copy.
-Snnnn -Ennnn	Specifies pages to start and end printing of the document. Preset value on the screen is is to start at 1 and end at 6000. If file is less than 6000 pages, stops at end of file.
-f nnn	Sets form length. Preset screen value is 66 lines, and default if omitted at command line is also 66.
-m nnn	Sets margin offset. This is added to any offset automatically set by your printer. Preset value on the screen is 11.
-a	Prints alternate pages of a document (to print even-numbered pages only, combine with the -S flag to start printing at page 2). Preset value on the screen is to not print alternate pages
-i	Interactive printing option. Will prompt after each page is printed for permission to continue. Non-interactive printing is the default setting.
-r	Resets page numbers for documents sent in batches. Not to reset page numbers is the default setting.

-h	Reset headers and footers for documents sent in batches. Not to reset headers and footers is the default setting.
-t	Sets print effects for terminal environment (highlight, reverse video, etc.) This is not included on the standard vprint screen.

■ **NOTE:** If you do not give either the -p or -t flags Fprint strips out special printer/terminal effects.

Rmerge flags

-L'<var>'	All fields left justify. This is a default value.
-R'<var>'	All fields right justify.
-S'<var>'	Insert one space on both sides of field.
-N'<var>'	Ignore L, R, and S options within text file.
-T'<var>'	Truncates end of oversized field. This is a default value.
-E'<var>'	Extends end of oversized field.
-D'<var>'	Deletes line in which field has been left blank.
-I'<var>'	Does not delete line if field has been left blank.

■ **NOTE:** If you wish, you may specify individual field names after each flag. When the upper-case flags are used at the command line, they override any previous use of the flags in the Rmerge text or variable files.

-b	Suppresses banner that states version number. This is always included when using the Rmerge screen.
-p	When printing multiple documents, if there are no page break commands within documents, will not put page breaks in. Default is to insert page breaks in output, following the processing of each full record of data to merge.
-n nn	Maximum number of records to be merged. Default is 6000. The maximum is 32767 (a UNIX OS limitation).

-d 'string'	Redefine variable declaration character. Default is ".V". This flag is not included in the standard #DEFINE screen but is available.
-f	Define field delimiter. Preset value is a new line. Default value is a space.
-r	Define record delimiter. Preset value is an asterisk on a line by itself. Default value is a space.

You may change any part of the #DEFINE section. For instance, to change the current field parameters defined in line 2 to curly brackets ({}), edit line 2 so it looks like this:

```
FIELDS={_}
```

To change the character that fills the space within the field to a period (.) edit #DEFINE as follows:

```
FIELDS={.}
```

These two changes produce a field that looks like this:

```
{ ..... }
```

Note that the exact length of a particular field is assigned in the #SCREEN section, discussed below. Also, you must change the field parameters in the #SCREEN section to correspond to any changes made to the parameters in #DEFINE.

To change the command keystrokes to call up the help screen or accept or reject the current screen, edit the appropriate line in #DEFINE. To change the keystrokes to call up the help screen from "<ESC> h" to "<ESC> ?", for example, edit the third line so it looks like this:

```
HELP=$_'?'
```

Compiling the vprint and vmerge Files

After making any changes to the vprint or vmerge files, you must recompile the file. The command

vpc –p

compiles the vprint file. The command

vpc –m

compiles the vmerge file.

The compilation process produces a *vprint.vp* or *vmerge.vp* file consisting of the screen as the user sees it and a listing of compiler output.

If the compilation is unsuccessful, vpc creates diagnostic message files, called *vprint.er* or *vmerge.er* along with the messages *Sorry, the compilation did not work* and *See the diagnostic file for details.*

The Standard #SCREEN Section

The standard #SCREEN section of the vprint file looks like this:

```
#SCREEN

                    -------- LYRIX PRINT SET-UP SCREEN --------

Enter file to print: [aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa]

Select printer type:?  [bbbbb]        Interactive?  (y/n):  [c]

Print from page:  [fffff]       to page:     [ggggg]

Number of copies to print: [hhhhh]        Page length:  [eeeee]

Set left margin to:  [ddd]                Print alternate pages?  (y/n)  [i]

-----------------------------------------------------------------

Accept  <ESC> a           Quit  <ESC> q          Help  <ESC> h
```

The standard vmerge #SCREEN section looks like this:

```
-------- LYRIX MAIL MERGE SCREEN --------
Enter name of text file      [aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa]
Enter name of merge file     [bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb]
Print or save in file?       [i]     Output filename:  [iiiiiiiiiiiiiiiiiiiiiiiiii]
Stop page breaks? (y/n)      [c]     Maximum copies to print      [kkkkkk]
Justify Field:               [d]     Ignore commands in text? (y/n)    [j]
Extend Field:                [e]     Missing Variable: delete/ignore   [f]
End of field symbol:  [ggggg]        End of Record symbol:        [hhhhh]
------------------------------------------------------------
Accept <ESC> a          Quit <ESC> q          Help <ESC> h
```

The #SCREEN section depicts essentially the same screen the user sees, except that the fields are filled in with lower-case alphabetic characters, called *field labels*. Each of the field labels refers to a set of conditions defined in the #FIELDS section discussed below. The conditions specify what type of input must be entered in the field when the user fills out the Viewprint or Mail Merge screen.

The #SCREEN section can be edited to include new options or exclude standard ones. To change the #SCREEN section, simply delete the entries you do not want or edit in options you wish to add to the screen. Assign the new entries a field label not already used in a #SCREEN field.

For example, to change the text in the first line of the Mail Merge screen from "Enter name of text file" to "Please type in name of text file", edit the appropriate line of the #SCREEN section so it reads as desired:

```
Please type in name of text file      [aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa]
```

To shorten the amount of space provided for field "a", edit the field so it is the

desired length:

```
Please type in name of text file        [aaaaaaaaaaaaaaaa]
```

To add a new option, edit in the appropriate text, and add a field of the appropriate length, assigning it a field label:

```
Change page length to 50 lines? (y/n)  [z]
```

Changes to the #SCREEN section affect only the way the Viewprint or Mail Merge screens look. To actually activate a new screen option or eliminate an old one, or to change a preset value, you must edit the #FIELDS section.

The Standard #FIELDS Section

This is the #FIELDS section for vprint:

```
#FIELDS
a=i()
a=p(Enter the name of the file to print)
b=a(DMP dmp DWP dwp DRAFT draft)
b=p(DMP – Dot Matrix Printer, DWP – Daisy Wheel Printer, DRAFT – Non-Radio Shack Printer
b=&(DMP)
b=m(DMP 'DMP')
b=m(dmp 'DMP')
b=m(DWP 'DWP')
b=m(dwp 'DWP')
b=m(DRAFT 'dumbprinter')
b=m(draft 'dumbprinter')
c=a(Y y N n)
c=p(Enter Y for interactive printing; default is spooled)
c=&(N)
c=m(Y '-i')
c=m(y '-i')
c=m(N '')
c=m(n '')
)
```

```
j=@c
j=m(y "| wp.filter >/dev/lp')
j=m(Y "| wp.filter >/dev/lp')
j=m(N "| wp.filter | lpr -F x &')
j=m(n "| wp.filter | lpr -F x &')
h=n(>0 <32767)
h=p(Enter page to start printing)
h=&(1)
f=n(>0 <32767)
f=p(Enter page to start printing)
f=&(1)
g=n(>0 <32767 >=f)
g=p(Enter page to stop printing)
g=&(6000)
e=n(>0)
e=p(Enter page length. Default is 66 lines)
e=&(66)
d=n(>=0 <245)
d=p(Enter left margin offset up to 245; Radio Shack DMP has one inch by default)
d=&(0)
i=a(y Y n N)
i=p(Enter Y to print alternate pages only)
i=m(y '-a')
i=m(Y '-a')
i=m(n '')
i=m(N '')
i=&(N)
)
```

This is the #FIELDS section for vmerge:

```
#FIELDS
a=i
a=&(_)
a=p(Enter the name of your text file – the "blank form" to fill in)
b=i
b=p(Enter name of merge file – [information to be merged into textfile])
c=a(y n Y N)
c=m(n '')
c=m(y '–p')
c=m(N '')
c=m(Y '–p')
c=p(Enter Y to stop page breaks between merge copies)
c=&(N)
d=a(I L r R s S i I)
d=m(I '–L')
d=m(L '–L')
d=m(r '–R')
d=m(R '–R')
d=m(s '–S')
d=m(S '–S')
d=m(i '')
d=m(I '')
d=&(I)
d=p(L - justify left, R - justify right, S - space on both sides, I - ignore)
j=a(y Y n N)
j=&(N)
j=p(Enter Y to ignore justification & spacing commands in file)
j=m(y '–N')
j=m(Y '–N')
j=m(n '')
j=m(N '')
```

```
e=a(t T e E n N)
e=m(t '-T')
e=m(T '-T')
e=m(e '-E')
e=m(E '-E')
e=p(E – extend field, T – truncate value when value is longer than field)
e=&(E)
k=n(>0 <32767)
k=p(Enter maximum number of copies to create)
k=&(6000)
f=a(d D i I n N)
f=p(If no add file entry matches the variable:  D to delete line, I to ignore)
f=m(d '-D')
F=m(D'-D')
f=m(i '')
f=m(I '')
f=&(I)
g=a()
g=p(Enter symbol for end of field. Default is a new line)
g=&(\n)
h=a()
h=p(Enter symbol for end of record. Default is an asterisk [*] alone on a line)
h=&(*\n)
i=a(p P f F)
i=m(p '| fprint -p DMP | wp.filter | lpr -F x ')
i=m(P '| fprint -p DMP | wp.filter | lpr -F x ')
i=m(f '>')
i=m(F '>')
i=&(P)
i=p(Enter P to print, F to save copy in a file)
l=o
l=+
l=p(Enter name of file in which to save copy)
))
```

The syntax of each line of the #FIELDS section is:

field label=special code character (listed below)

The field label in #FIELDS corresponds directly to the same letter assigned to an individual field in the #SCREEN section.

The code character represents a specific condition that must be met by the user's entry in the #SCREEN field represented by the field label. The codes are defined as follows:

r	A file for which the user has read permission
o	A file for which the user has write permission
/	A valid accessible directory
d	Can be opened as a new directory
a()	Alphabetic character (choice, range, or type) choice a=a(a b c d) range a=a(a–z) type a=a() input data must not be a digit
n()	Numeric character (choice, range, or type) choice a=n(1 2 3 4) range a=n(1–9) range a=n(>0 <10 >=b) input data must be greater than 0, less than 10 and greater than the value of field 'b' type a=n() input data must be digit or one of ', . –'
m()	map string 1 to string 2 a=m(p "| lpr2") map the response 'p' to the command "pipe to lpr2"
p()	prompt string as it appears to the user at the top of the screen p(Enter filename here)
+	discretionary input (field does not have to be filled in, otherwise will require an entry).
&()	default value

Field 'a' of the vprint #SCREEN section

```
Enter file to print: [aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa]
```

is defined as follows in the #FIELDS section:

```
a=i()
a=p(Enter the name of the file to print)
```

Explanation:

a=i() — The user response to field "a" must be a file name for which the user has read permission.

a=p(Enter...) — The user is prompted *"Enter the name of the file to print"*

Field c of the vprint #SCREEN section

```
Draft or letter quality printer? [c]
```

is defined in the vprint #FIELDS section as:

```
c=a(d D l L)
c=p(Enter D for draft or L for letter quality printer)
c=&(D)
c=m(l "| lpr2 ")
c=m(L "| lpr2 ")
c=m(d "| lpr ")
c=m(D "| lpr ")
```

Explanation:

c=a(d D I L)	— The user must answer with one of the alphabetic characters "d", "D", "I", or "L".
c=p(Enter ...)	— The user is prompted "Enter D for draft or L for letter quality printer"
c=&(D)	— The default is "D" for draft quality printer.
c=m(I 'I lpr2')	— The user's response is mapped to the command to pipe the file to the letter or draft quality printer.

To remove an option from the #SCREEN section, delete the option, and then delete the corresponding set of field definitions in the #FIELDS section. For instance, if you want to remove the *Ignore commands in text* option from the vmerge screen, you would delete that option and its corresponding field from the #SCREEN section, and then delete all references to the letter assigned to that option, "j" in #FIELDS. Compile the file with the *vpc* command.

To add an option to the #SCREEN section, edit in the option, assigning it a unique lower-case letter. Edit the #FIELDS section to add an appropriate series of definitions for the new option. Compile the file with the *vpc* command.

To change any of the default options, you must change the #FIELDS section. For instance, to change the default printer type in the vprint file, you must refer to the #FIELDS section defining fields **b** and **c** and edit it so it looks like this:

```
b=a()
b=p(Enter the name of the file to print)
b=&(your printer name)
c=a(d D I L)
c=p(Enter D for draft or L for letter quality printer)
c=&(D)
c=m(I 'I your letter quality printer spooler program')
c=m(L 'I your letter quality printer spooler program')
c=m(d 'I your draft quality printer name spooler program')
c=m(D 'I your draft quality printer name spooler program')
```

You must then compile the file with the *vpc* command.

Additional Viewprint and Mail Merge Files

The contents of the additional Viewprint and Mail Merge files *vp.help*, *vp.message*, and *vpc.message* follow. You will find vp.help, vp.message and vpc.message in */usr/lib/wp*.

Vp.help Screen

If the user types the sequence to call up the help screen while trying to complete the Viewprint or Mail Merge screen, he receives the vp.help screen. This screen is editable.

```
                    *** HELP SCREEN ***
CORRECTING THE SCREEN
Erase previous character                              RUB
Reset field to default                                  #
Reset all fields to defaults                        <ESC> #

MOVING AROUND THE SCREEN
Next field                                            TAB
Field below                    RETURN OR CURSOR DOWN
Field above                                     CURSOR UP
Back a char                                     CURSOR LEFT
Forward a char                                  CURSOR RIGHT

CURSOR LEFT     will wrap to previous field
when cursor is on first space of the field

CURSOR RIGHT will wrap to next field when
cursor position is last space of the field
```

The vp.help screen is in */usr/lib/wp*.

Vp.message File

If the user makes a mistake filling out the vprint or Rmerge screens, he receives a message from the vp.message file, in /usr/lib/wp. You may edit the messages in this file. The messages are position dependent.

The vp.message file follows:

```
There must be a reply in this field.
Please fill in all fields where a reply is required.
Invalid replies – move to the bad fields for more details
Only digits – , and . allowed
File does not exist
No read permission for this file
File already exists
Directory does not exist
No access to this directory
Directory already exists
No write permission in this directory
Cannot execute this file
Invalid choice
Screen fields exceed terminal width
No termcap entry
Can't access termcap
Entry contains an invalid character
Invalid number
Numeric characters not allowed in this field
Press <RETURN> to continue
Press <RETURN> for next page or "*" to quit
```

Vpc.message File

The vpc.message file contains the messages output by the compiler, vpc. These messages are editable but are position dependent. The contents of this file follow:

```
Ghost fields may only be used with map type
Range badly specified
Missing end quote for character in sequence
First character cannot be a hyphen
Bad character in sequence
This line does not contain a valid sequence name
Sorry, the compilation did not work
See the diagnostic file for details
Unable to open report file
More than one sequence name specified
No field character for field
No command line in #DEFINE
No #DEFINE section
Instruction file has missing sections
Your sequences are not unique
Not enough space in screen definition for this type
Default should not be declared before type is specified
Default value is too long for this field
Incomplete field line
String too long for field
Missing space
Bad field definition syntax
Missing bracket
Bad syntax : map fields require two character strings
Bad mix of types for a numeric field
Bad format for the second field of the range
Current field cannot appear in a numeric range
Syntax error in field definition
```

```
Your compilation was successful
Bad character in field
Can't open a temporary file
No #SCREEN section
Too many screen lines
Missing end of section marker
Screen field is not defined in #FIELDS section
Memory allocation error
Screen fields cannot be ghosts
No #FIELDS section
This should be an '=' character
Bad field type
This ghost field has already been defined
This field is not defined in the screen section
Ghost field id is not in the command line
This id is not one from the screen
```

■ **NOTE:** The first compiler message refers to ghost fields. These are fields that are listed in the first line of #DEFINE but which do not actually appear in #SCREEN. They are instead mapped in the #FIELDS section to other fields which are listed in #SCREEN.

An example of a ghost field in use is the hypothetical #SCREEN option,

```
Run in background? (y/n) [q]
```

Using the ghost field "z", this would be defined in #FIELDS as:

```
q=a(y Y n N)
z=@q
z=m('Y' '&')
z=m('y' '&')
z=m('N' ' ')
z=m('n' ' ')
```

The response to field "q" is mapped to field "z", which in turn maps the user response "Y" or "y" to the command to run a process in the background.

Field "z" is a ghost field which is not actually mentioned in #SCREEN. It is, however, listed in #DEFINE, along with the other field characters.

Summary

- Viewprint accesses the Fprint program.
- Mail Merge accesses the Rmerge program.
- The vprint and vmerge files are divided into three sections, #DEFINE, #SCREEN, and #FIELDS. Each section begins with the hatchmark symbol (#) and ends with two right parentheses.
- The #DEFINE section lists all flags for a particular screen, establishes the field parameter character, and defines the keystroke sequences that allow the user to ask for help, quit, or accept the current screen.
- You can change any part of the #DEFINE section.
- After any change to any of the three sections, you must compile the file with the vpc command. The files containing compiler output are called vprint.vp and vmerge.vp.
- The #SCREEN depicts the vprint or vmerge screen as the user sees it and assigns a lower-case letter to each field.
- You can add or remove options from #SCREEN. New options must be assigned a unique lower-case letter. Fields deleted from #SCREEN must be deleted from #FIELDS. Fields added to #SCREEN must be added to #FIELDS.
- The #FIELDS section assigns definition types to the fields assigned in #SCREEN, setting specific conditions that must be met by the user's entry in the #SCREEN field.
- You may edit the #FIELDS section to include new options, delete old ones, or to change preset values, such as page length and printer type.
- Additional Viewprint and Mail Merge files include the Viewprint help file, vp.help, the Viewprint message file, vp.message, and the vpc compiler message file, vpc.message.
- A ghost field is a field which appears in the command line portion of #DEFINE and is defined in #FIELDS, but which does not actually appear in #SCREEN itself. Instead, it is mapped to another field which does appear in #SCREEN.

APPENDIX A
Lyrix Standard Command File

Following is the Radio Shack (TM) Customized Lyrix Command File. #COMMANDS sections for several different terminal types have been included, and Radio Shack printer effects are reflected in #EFFECTS.

```
**************************
*                        *
*     LYRIX for RADIO SHACK     *
*     CUSTOM COMMAND-FILE       *
*       Vs 3.12 — 10/15/84      *
*                        *
*     Copyright (c) 1984 by     *
*   The Santa Cruz Operation, Inc.  *
*                        *
**************************
```

```
---------- < System Attributes > ----------
#SYSTEM
BACKUP='^.bak'
NAME='COMPANY NAME'
2SPACE='.!?:;',                    3SPACE=''
RULER='.#CIJLTRH'
WDEL=';:,.!? ',                    HLEN=5
DOTS='HE:HM:FO:FM:PL:PA:PN:SN:PM:SP:JY:JN:RE:ME:ST:SB'
KEYIN='Y'
MODE='',                   PAGE='66'
PNUM='#',                  STOP=' *&()<>'
DECTAB='.'
STATUS='S'
))
```

```
---------- < Program Message Table > ----------
#MESSAGES
'[?]  Select an option'
'Reserved for future use'
'Selected menu not available or incorrectly entered in command file.'
'Press <RETURN> to continue:'
'Unable to run system command.'
'Not a good file name!'
'  selected.'
'No write permission! '
'Please enter the file name or "*" to return to menu '
'Copying file to  '
'Unable to open file for edit.'
'Cannot create new backup file.'
'Cannot write to file '
'*** File copy ***'
'*** File re-name ***'
'*** File erase ***'
'Requested task is now complete.'
'Operation FAILED.'
'Original file'
'New file'
'Enter "*" to confirm or <RETURN> to abandon:'
'Merge file section not found.'
'Enter name of file :'
'Enter pattern to search for:'
'Option ? '
'Sorry but that search failed.'
'Enter "a"=all files, "l"=statistics, "s"=short'
'Unable to list selected directory'
'Enter "*" to exit or <RETURN> for more '
'*** List files ***'
'*** External function ***'
'Busy saving the document.....'
'*** Edit a file ***'
'*** Create a file ***'
'"–" to hyphenate or <RETURN> to skip: '
```

```
'Enter " " (space) to replace, <RETURN> to skip, or "*" to quit'
'Enter new characters or <RETURN> to delete '
'Enter "*" for global or <RETURN> for interactive '
'Invalid command entered'
'Enter "*" to confirm quit or <RETURN> to continue '
'Busy executing command'
'Enter <RETURN> for next or "*" to quit '
'Enter ruler number (0–9)'
'Enter (B)lank, (L)eave, (R)emove or (S)ave'
'Enter (O)verlay, (E)lbow or (I)nsert'
'Enter 1 (tab), 2 (hyphen), or 3 (insert) :'
'Reserved for future use'
'Enter <RETURN> for next page,"*" <RETURN> to quit'
'INSERT'
'HYPHEN'
'TAB'
'Current directory: '
'Getting your directory listing'
'Directory, enter "*" to change directory'
'No write permission, enter "*" for read only '
'Sorry — no read permission on this file'
'Move cursor to table of names or enter a file/directory name'
'*** Change directory ***'
'Enter a directory name then press <RETURN>'
'Sorry — cannot change to requested directory'
'Enter page number, <RETURN> :'
'Press "*" for more '
'File exists — enter "*" to overwrite or <RETURN> to abandon '
'Reserved for future use'
'Enter (B)lock or (S)erial'
'Enter (D)ocument or (P)aragraph'
'Enter (I)nsert or (O)verlay'
'Enter ruler number (0–9) or "." for current'
))
```

---------- < Function Table Reference List > ----------

#COMLOCATE

F01=Delete char.	F02=Destructive Bsp.	F03=Delete word	F04=Delete right
F05=Delete left	F06=Delete line	F07=Delete blanks	F08=Insert space
F09=Insert line	F10=Insert blank lines	F11=Return	F12=Left
F13=Right	F14=Up	F15=Down	F16=Tab
F17=Previous word	F18=Next word	F19=Line up	F20=Line down
F21=Go left	F22=Go right	F23=Go up	F24=Go down
F25=Top of screen	F26=Bottom of screen	F27=Scroll down	F28=Scroll up
F29=Top of file	F30=Bottom of file	F31=Lower case	F32=Upper case
F33=Center line	F34=Not in use	F35=Show effect	F36=Line split
F37=Do again	F38=Format paragraph	F39=Help	F40=Redraw screen
F41=Find NEXT	F42=Find pattern	F43=Search/repl.	F44=Format Document
F45=Stop command	F46=Go To Page	F47=Recall Ruler	F48=Store ruler
F49=Use Ruler	F50=Use mode	F51=Stop mode	F52=CP mark (block)
F53=CP blank	F54=CP leave	F55=CP Remove	F56=Not In Use
F57=CP Overlay	F58=CP Insert	F59=CP Elbow	F60=Menu escape
F61=Exit & save	F62=Quit (no save)	F63=Write no exit	F64=Not In Use
F65=Merge Insert	F66=Restore text	F67=Insert on	F68=Insert off
F69=Effect ON	F70=Effect OFF	F71=CP mark (ser)	F72=Save CP text
F73=Merge overlay	F74=Save to file	F75=Spell	

---------- < Commands sections > ----------

```
#COMMANDS
F01=3,          F02=127,        F03=23,         F04=$-'DR'
F05=$-'DL',     F06=24,         F07=$-'DB',     F08=5
F09=15,         F10=$-'AL',     F11=13,         F12=L
F13=R,          F14=U,          F15=D,          F16=T
F17=16,         F18=14,         F19=30,         F20=22
F21=$-L,        F22=$-R,        F23=$-U,        F24=$-D
F25=20,         F26=2,          F27=4,          F28=21
F29=$-'T',      F30=$-'B',      F31=$-'KL',     F32=$-'KU'
F33=$-'C',      F34=0,          F35=$-'@',      F36=$-'L'
F37=1,          F38=6-'P',      F39=$-'H',      F40=$-'V'
F41=$-'N',      F42=$-'F',      F43=$-'G',      F44=6-'D'
F45=28,         F46=$-'P',      F47=$-'R',      F48=$-'S'
F49=$-'U',      F50=$-'{',      F51=$-'}',      F52=$-'(B'
F53=$-')B',     F54=$-')L',     F55=$-')R',     F56=0
F57=$-'*O',     F58=$-'*I',     F59=$-'*E',     F60=$-'!'
F61=$-'E',      F62=$-'Q',      F63=$-'W',      F64=0
F65=$-'MI',     F66=18,         F67=$-'I',      F68=$-'O'
F69=$-'<',      F70=$-'>',      F71=$-'(S',     F72=$-')S'
F73=$-'MO',     F74=$-'X',      F75=$-'$'
F12=8,          F13=12,         F14=11,         F15=10
))
```

The following #COMMANDS sections are specific to certain terminals.
Any available function keys on these terminals have been utilized.
If you do not intend to use the terminals these commands are designed for, you can remove the associated sections.

Radio Shack TRS-80 (R) Model 16 console :

```
#COMMANDS-trs16
F01=3,          f02=$-'K',      F03=23,         f04=$-'dr'
f05=$-'dl',     F06=24,         f07=$-'db',     F08=5
F09=15,         f10=$-'al',     F11=13,         f12=L
f13=R,          f14=U           f15=D,          F16=T
F17=16,         F18=14,         F19=30,         F20=22
f21=$-L,        F22=$-R,        f23=$-U,        f24=$-D
F25=20,         F26=26,         F27=4,          F28=21
F29=$-'T',      f30=$-'b',      f31=$-'kl',     F32=$-'ku'
F33=$-'c',      F34=0,          F35=$-'@',      F36=$-'L'
F37=25,         F38=6-'P',      F39=$-'H',      F40=$-'V'
F41=$-'N',      F42=$-'F',      F43=$-'G',      F44=6-'D'
F45=28,         F46=$-'P',      F47=$-'R',      F48=$-'S'
F49=$-'U',      F50=$-'{',      F51=$-'}',      F52=$-'(B'
F53=$-')B',     F54=$-')L',     F55=$-')R',     F56=0
F57=$-'*O',     F58=$-'*I',     F59=$-'*E',     F60=$-'!'
F61=$-'E',      F62=$-'Q',      F63=$-'W',      F64=0
F65=$-'MI',     F66=18,         F67=$-'I',      F68=$-'O'
F69=$-'<',      F70=$-'>',      F71=$-'(S',     F72=$-')S'
F73=$-'MO',     F74=$-'X',      F75=$-'$'       F15=10
F12=8,          F13=12,         F14=11,
F38=$-'JP',     F39=1,          F61=2,          F44=$-'JD'
))
```

```
#COMMANDS-adds25
F01=3,          F02=127,        F03=23,         F04=$-'DR'
F05=$-'DL',     F06=24,         F07=$-'DB',     F08=5
F09=15,         F10=$-'AL',     F11=13,         F12=L
F13=R,          F14=U,          F15=D,          F16=T
F17=16,         F18=14,         F19=30,         F20=22
F21=$-L,        F22=$-R,        F23=$-U,        F24=$-D
F25=20,         f26=$-'b',      F27=4,          F28=21
F29=$-'T',      f30=$-'B',      f31=$-'kl',     f32=$-'ku'
F33=$-'C',      F34=0,          F35=$-'@',      F36=$-'L'
F37=25,         F38=$-'JP',     F39=$-'H',      F40=$-'V'
F41=$-'N',      F42=$-'F',      F43=$-'G',      F44=$-'JD'
F45=28,         F46=$-'P',      F47=$-'R',      F48=$-'S'
F49=$-'U',      F50=$-'{',      F51=$-'}',      F52=$-'(B'
F53=$-')B',     F54=$-')L',     F55=$-')R',     F56=0
F57=$-'*O',     F58=$-'*I',     F59=$-'*E',     F60=$-'I'
F61=$-'E',      F62=$-'Q',      F63=$-'W',      F64=0
F65=$-'MI',     F66=18,         F67=$-'I',      F68=$-'O'
F69=$-'<',      F70=$-'>',      F71=$-'(S',     F72=$-')S'
F73=$-'MO',     F74=$-'X',      F75=$-'$'       F13=12
F12=8,          F13=6,          F14=26,         F15=10
F39=1,          F61=2,          f02=$-'K',      F14=11
))
```

```
#COMMANDS-adm5
F01=3,          F02=127,        F03=23,         F04=$-'DR'
F05=$-'DL',     F06=24,         F07=$-'DB',     F08=5
F09=15,         F10=$-'AL',     F11=13,         F12=L
F13=R,          F14=U,          F15=D,          F16=T
F17=16,         F18=14,         F19=30,         F20=22
F21=$-L,        F22=$-R,        F23=$-U,        F24=$-D
F25=20,         f26=$-'b',      F27=4,          F28=21
F29=$-'T',      f30=$-'B',      f31=$-'kl',     f32=$-'ku'
F33=$-'C',      F34=0,          F35=$-'@',      F36=$-'L'
F37=25,         F38=6-'P',      F39=$-'H',      F40=$-'V'
F41=$-'N',      F42=$-'F',      F43=$-'G',      F44=6-'D'
F45=28,         F46=$-'P',      F47=$-'R',      F48=$-'S'
F49=$-'U',      F50=$-'{',      F51=$-'}',      F52=$-'(B'
F53=$-')B',     F54=$-')L',     F55=$-')R',     F56=0
F57=$-'*O',     F58=$-'*I',     F59=$-'*E',     F60=$-'I'
F61=$-'E',      F62=$-'Q',      F63=$-'W',      F64=0
F65=$-'MI',     F66=18,         F67=$-'I',      F68=$-'O'
F69=$-'<',      F70=$-'>',      F71=$-'(S',     F72=$-')S'
F73=$-'MO',     F74=$-'X',      F75=$-'$'       F44=$-'JD'
F12=8,          F13=12          F14=11          F15=10
F39=1,          F61=2,          f02=$-'K',      F38=$-'JP'
))
```

```
#COMMANDS-tvi910
F01=3,          F02=127,        F03=23,         F04=$-'DR'
F05=$-'DL',     F06=24,         F07=$-'DB',     F08=5
F09=15,         F10=$-'AL',     F11=13,         F12=L
F13=R,          F14=U,          F15=D,          F16=T
F17=16,         F18=14,         F19=30,         F20=22
F21=$-L,        F22=$-R,        F23=$-U,        F24=$-D
F25=20,         f26=$-'b',      F27=4,          F28=21
F29=$-'T',      f30=$-'B',      f31=$-'kl',     f32=$-'ku'
F33=$-'C',      F34=0,          F35=$-'@',      F36=$-'L'
F37=25,         F38=6-'P',      F39=$-'H',      F40=$-'V'
F41=$-'N',      F42=$-'F',      F43=$-'G',      F44=6-D
F45=28,         F46=$-'P',      F47=$-'R',      F48=$-'S'
F49=$-'U',      F50=$-'{',      F51=$-'}',      F52=$-'(B'
F53=$-')B',     F54=$-')L',     F55=$-')R',     F56=0
F57=$-'*O',     F58=$-'*I',     F59=$-'*E',     F60=$-'I'
F61=$-'E',      F62=$-'Q',      F63=$-'W',      F64=0
F65=$-'MI',     F66=18,         F67=$-'I',      F68=$-'O'
F69=$-'<',      F70=$-'>',      F71=$-'(S',     F72=$-')S'
F73=$-'MO',     F74=$-'X',      F75=$-'$'       F38=$-'JP'
F12=8           F13=12          F14=11          F15=10
F39=1,          F61=2,          f02=$-'K',      F44=$'JD'
))
```

```
#COMMANDS-dt100
F01=3,          F02=127,        F03=23,         F04=$-'DR'
F05=$-'DL',     F06=24,         F07=$-'DB',     F08=5
F09=15,         F10=$-'AL',     F11=13,         F12=L
F13=R,          F14=U,          F15=D,          F16=T
F13=$-'OC',     F14=$-'OA',     F15=$-'OB',     F12=$-'OD'
F13=$-'[C',     F14=$-'[A',     F15=$-'[B',     F16=$-'[D',
F17=16,         F18=14,         F19=30,         F20=22
F21=$-L,        F22=$-R,        F23=$-U,        F24=$-D
F25=20,         F26=2,          F27=4,          F28=21
F29=$-'T',      F30=$-'B',      f31=$-'kl',     f32=$-'ku'
F33=$-'C',      F34=0,          F35=$-'@',      F36=$-'L'
F37=1,          F38=6-'P',      F39=$-'H',      F40=$-'V'
F41=$-'N',      F42=$-'F',      F43=$-'G',      F44=6-D
F45=28,         F46=$-'P',      F47=$-'R',      F48=$-'S'
F49=$-'U',      F50=$-'{',      F51=$-'}',      F52=$-'(B'
F53=$-')B',     F54=$-')L',     F55=$-')R',     F56=0
F57=$-'*O',     F58=$-'*I',     F59=$-'*E',     F60=$-'!'
F61=$-'E',      F62=$-'Q',      F63=$-'W',      F64=0
F65=$-'MI',     F66=18,         F67=$-'I',      f68=$-'o'
F69=$-'<',      F70=$-'>',      F71=$-'(S',     F72=$-')S'
F73=$-'MO',     F74=$-'X',      F75=$-'$'       F02=$-'K'
F12=8           F13=12          F14=11          F15=10
F37=25,         F38=$-'JP',     F44=$-'JD'


f54=$-'Oul',    f01=$-'Ow',     f03=$-'Ox',     f52=$-'Otb'
f71=$-'Ots',    f55=$-'Our',    f53=$-'Oub',    f72=$-'Ous'
f61=$-'Op',     f09=$-'OQ',     f69=$-'Oq',     f70=$-'Or'
f67=$-'OR',     f68=$-'OS',     f47=$-'Om',     f33=$-'Os'
f39=$-'OM',     f38=$-'On',     f08=$-'OP',     f57$-'Ovo'
f58=$-'Ovi',    f59=$-'Ove'
f48=$-'Ol',     f06=$-'Oy'
))
```

Digital Equipment Corporation VT100 (TM):
#COMMANDS-vt100

```
F01=3,          F02=127,        F03=23,         F04=$-'DR'
F05=$-'DL',     F06=24,         F07=$-'DB',     F08=5
F09=15,         F10=$-'AL',     F11=13,         F12=L
F13=R,          F14=U,          F15=D,          F16=T
F13=$-'OC',     F14=$-'OA',     F15=$-'OB',     F16=$-'OD'
F13=$-'[C',     F14=$-'[A',     F15=$-'[B',     F16=$-'[D'
F17=16,         F18=14,         F19=30,         F20=22
F21=$-L,        F22=$-R,        F23=$-U,        F24=$-D
F25=20,         F26=2,          F27=4,          F28=21
F29=$-'T',      F30=$-'B',      F31=$-'KL',     F32=$-'KU'
F33=$-'C',      F34=0,          F35=$-'@',      F36=$-'L'
F37=1,          F38=6-'P',      F39=$-'H',      F40=$-'V'
F41=$-'N',      F42=$-'F',      F43=$-'G',      F44=6-'D'
F45=28,         F46=$-'P',      F47=$-'R',      F48=$-'S'
F49=$-'U',      F50=$-'{',      F51=$-'}',      F52=$-'(B'
F53=$-')B',     F54=$-')L',     F55=$-')R',     F56=0
F57=$-'*O',     F58=$-'*I',     F59=$-'*E',     F60=$-'!'
F61=$-'E',      F62=$-'Q',      F63=$-'W',      F64=0
F65=$-'MI',     F66=18,         F67=$-'I',      f68=$-'o'
F69=$-'<',      F70=$-'>',      F71=$-'(S',     F72=$-')S'
F73=$-'MO',     F74=$-'X',      F75=$-'$'
F12=8,          F13=12,         F14=11,         F15=10

f54=$-'Oul',    f01=$-'Ow',     f03=$-'Ox',     f52=$-'Otb'
f71=$-'Ots',    f55=$-'Our',    f53=$-'Oub',    f72=$-'Ous'
f61=$-'Op',     f09=$-'OQ',     f69=$-'Oq',     f70=$-'Or'
f67=$-'OR',     f68=$-'OS',     f47=$-'Om',     f33=$-'Os'
f39=$-'OM',     f38=$-'On',     f08=$-'OP',     f57=$-'Ovo'
f58=$-'Ovi',    f59=$-'Ove'
f48=$-'Ol',     f06=$-'Oy'
))
```

```
Visual Technology Inc. VISUAL 55 (TM)
#COMMANDS-vi55
F01=3,          F02=127,        F03=23,         F04=$-'DR'
F13=R,          F14=U,          F15=D,          F16=T
F17=16,         F18=14,         F19=30,         F20=22
F21=$-L,        F22=$-R,        F23=$-U,        F24=$-D
F25=20,         f26=$-'b',      F27=4,          F28=25
F29=$-'T',      f30=$-'B',      F31=$-'KL',     F32=$-'KU'
F33=$-'C',      F34=0,          F35=$-'@',      F36=$-'L'
F37=1,          F38=$-'JP',     F39=$-'H',      F40=$-'V'
F41=$-'N',      F42=$-'F',      F43=$-'G',      F44=$-'JD'
F45=28,         F46=$-'P',      f47=$-'r',      F48=$-'S'
F49=$-'U',      F50=$-'{',      F51=$-'}',      F52=$-'(B'
F53=$-')B',     F54=$-')L',     F55=$-')R',     F56=0
F57=$-'*O',     F58=$-'*I',     F59=$-'*E',     F60=$-'!'
F61=$-'E',      f62=$-'q',      F63=$-'W',      F64=0
F65=$-'MI',     F66=18,         F67=$-'I',      F68=$-'O'
F69=$-'<',      F70=$-'>',      F71=$-'(S',     F72=$-')S'
F73=$-'MO',     F74=$-'X',      F75=$-'$'
F12=8,          F13=12,         F14=11,         F15=10

f08=$-'?w',     f09=$-'?x',     f67=$-'?y',     f68=$-'?m'
f01=$-'?t',     f03=$-'?u',     f06=$-'?v',     f47=$-'?l'
f52=$-'?qb',    f71=$-'?qs',    f53=$-'?rb',    f54=$-'?rl'
f55=$-'?rr',    f72=$-'?rs',    f57=$-'?so',    f58=$-'?sl',    f59=$-'?se',
```

```
f61=$-'?p',     f38=$-'?np',    f44=$-'?nd',    f48=$-'?M'

f46=$-'P',      f04=2-'!',      f05=2-'A'
f42=$-'Q',      f41=2-'"',      f43=2-'B'
f63=$-'R',      f62=2-'#',      f36=2-'C'
f33=$-' ',      f32=2-'$',      f31=2-'D'
f60=$-'!',      f39=2-'%',      f40=2-'E'
f69=$-'"',      f70=2-'&',      f35=2-'F'
f65=$-'#i',     f73=$-'#o',     f50=2-'\'',     f51=2-'G'
f75=$-'$',      f49=2-'('
f25=$-'%',      f29=2-')',      f28=2-'I',      f17=2-'9'
f26=$-'&',      f30=2-'*',      f27=2-'J',      f18=2-':'
f08=$-'.',      f01=2-'+',      f02=2-'K'
f52=$-'(b',     f71=$-'(s',     f66=2-',',      f37=2-'L'
))
```

```
Visual Technology Inc. VISUAL 200 (TM)
#COMMANDS-vi200
F01=3,          F02=127,        F03=23,         f04=$-'dr'
f05=$-'dl',     F06=24,         f07=$-'db',     F08=5
F09=15,         f10=$-'al',     F11=13,         f12=L
f13=R,          f14=U,          f15=D,          F16=T
F17=16,         F18=14,         F19=30,         F20=22
f21=$-L,        f22=$-R,        f23=$-U,        f24=$-D
F25=20,         F26=$-'b',      F27=4,          F28=21
F29=$-'T',      f30=$-'b',      F31=$-'KL',     F32=$-'KU'
f33=$-'c',      F34=0,          F35=$-'@',      F36=$-'L'
F37=1,          F38=6-'P',      F39=$-'H',      F40=$-'V'
F41=$-'N',      F42=$-'F',      F43=$-'G',      F44=6-'D'
F45=28,         F46=$-'P',      F47=$-'R',      F48=$-'S'
F49=$-'U',      F50=$-'{',      F51=$-'}',      F52=$-'(B'
F53=$-')B',     F54=$-')L',     F55=$-')R',     F56=0
F57=$-'*O',     F58=$-'*I',     F59=$-'*E',     F60=$-'!'
F61=$-'E',      F62=$-'Q',      F63=$-'W',      F64=0
F65=$-'MI',     F66=18,         F67=$-'I',      F68=$-'O'
F69=$-'<',      F70=$-'>',      F71=$-'(S',     F72=$-')S'
F73=$-'MO',     F74=$-'X',      F75=$-'$'
F12=8,          F13=12,         F14=11,         F15=10

f46=$-'P',      f62=$-'Q',      f47=$-'R',      f48=$-' '
f60=$-'!',      f30=$-'"',      f63=$-'#',      f47=$-'$'
f43=$-'%',      f41=$-'&',      f42=$-'\'',     f52=$-'(b'
f71=$-'(s',     f53=$-')b',     f54=$-')l',     f55=$-'(r'
f72=$-')s',     f57=$-'*o',     f54=$-'*i',     f59=$-'*e'

f08=$-'?w',     f09=$-'?x',     f66=$-'?y'
f67=$-'?t',     f68=$-'?u',     f33=$-'?v'
f01=$-'?q',     f03=$-'?r',     f06=$-'?s'
f69=$-'?p',     f70=$-'?l',     f35=$-'?n'
f61=$-'?m',     f38=$-'?Mp',    f39=$-'?Md'
))
```

```
Altos Computer Systems ALTOS II (TM)
#COMMANDS-TWO
F01=3,          F02=127,        F03=23,         F04=$-'DR'
F05=$-'DL',     F06=24,         F07=$-'DB',     F08=5
F09=15,         F10=$-'AL',     F11=13,         F12=L
F13=R,          F14=U,          F15=D,          F16=T
F17=16,         F18=14,         F19=30,         F20=22
F21=$-L,        F22=$-R,        F23=$-U,        F24=$-D
F25=20,         F26=2,          F27=4,          F28=21
F29=$-'T',      F30=$-'B',      F31=$-'KL',     F32=$-'KU'
F33=$-'C',      F34=0,          F35=$-'@',      F36=$-'L'
F37=26,         F38=6-'P',      F39=$-'H',      F40=$-'V'
F41=$-'N',      F42=$-'F',      F43=$-'G',      F44=6-'D'
F45=28,         F46=$-'P',      F47=$-'R',      F48=$-'S'
F49=$-'U',      F50=$-'{',      F51=$-'}',      F52=$-'(B'
F53=$-')B',     F54=$-')L',     F55=$-')R',     F56=0
F57=$-'*O',     F58=$-'*I',     F59=$-'*E',     F60=$-'I'
F61=$-'E',      F62=$-'Q',      F63=$-'W',      F64=0
F65=$-'MI',     F66=18,         F67=$-'I',      f68=$-'o'
F69=$-'<',      F70=$-'>',      F71=$-'(S',     F72=$-')S'
F73=$-'MO',     F74=$-'X',      F75=$-'$'
F12=8,          F13=12,         F14=11,         F15=10

f61=$-'Op',     f52=$-'Oq',     f55=$-'Or',     f58=$-'Os'
f69=$-'Ot',     f70=$-'Ou',     f35=$-'Ov',     f01=$-'Ow'
f03=$-'Ox',     f06=$-'Oy',     f38=$-'On',     f47=$-'Om'
f48=$-'Ol',     f65=$-'OM',     f08=$-'OP',     f09=$-'OQ'
f67=$-'OR',     f68=$-'OS',

f09=$-'[L',     f06=$-'[M',     f08=$-'[O',     f01=$-'[P'
f28=$-'[T',     f27=$-'[S',     f25=$-'[F',     f39=1-'P'-13

f63=1-'@'-13,   f18=1-'A'-13,   f25=1-'B'-13,   f29=1-'C'-13
f46=1-'D'-13,   f42=1-'E'-13,   f43=1-'F'-13,   f52=1-'G'-13-'b'
f71=1-'G'-13-'s', f57=1-'H'-13-'o', f58=1-'H'-13-'i'
))
```

```
---------- < Messages for User Defined Prompts > ----------
#USER
'*** AUXILLIARY MESSAGES FOR USE WITHIN LYRIX SYSTEM CALLS ***'
'Enter name and location of person to talk to:'
'Enter [year] or [month year] like 1982 or 5 1982 :'
'Enter name of person:'
'Enter name of document to check for spelling:'
'Enter which document to look at:'
'Enter subject to look for:'
'Enter name of sub-directory'
'Enter name of person, <Return>; then type message <Return> <CTL> d'
'Enter names and numbers on same line then CTL D'
'Enter the name of the file to find:'
'Enter Unix command-line:'
'Enter name of file:'
'Enter manual entry required:'
'Enter password:'
))

---------- < Default Margin Ruler Definitions > ----------

#RULERS

L.....T.....T.....T.....T.....T.....T.....T.....T.....T... R.

L.....T.....T.....T.....T.....T.....T.....T.....T.....T... J.

.....L.....T.....T.....T.....T.....T.....T.....T.....T... R.

.....L.....T.....T.....T.....T.....T.....T.....T.....T... J.

.....L.....T.....T.....T.....T.....T.....T.....T.....T... R.

.....L.....T.....T.....T.....T.....T.....T.....T.....T... J.

.....L.....T.....T.....T.....T.....T.....T.....T.....T... R.

.....L.....T.....T.....T.....T.....T.....T.....T.....T... J.

T.....L.....T.....T.....T.....T.....T.....T.....T.....T... J.

T.....T.....T.....T.....T.....T.....T.....T.....T.....T... T.

))
```

```
#EFFECTS
A=Bold
B=Double Strike
C=Underline
D=Underline text only
E=Bold & Underline
F=Superscript
G=Subscript
H=Condensed(DMPs)
I=Compressed(DMP200)
J=Elongation(DMPs)
K=Correspondance 10 pitch
L=12 pitch
M=Italics(DMP110)
))
```

```
---------- < Primary Menu > ----------
#SYSMENU
'*** Main Office System Menu ***'
'1 – Word Processing Menu'=M('#DOCPREP')
' '=>()
'2 – File Management Menu'=M('#FILEMANAGE')
' '=>()
'3 – Mail System Menu'=M('#MAIL')
' '=>()
'4 – Additional System Usage Menu'=M('#SYSCOMMS')
' '=>()
'D – Change Directory'=X()
'P – Printing System Menu'=M('#PRINTING')
'L – List Files'=?('*')
'? – HELP'=M('#HELP')
' '=>()
'* – Leave Lyrix'=*()
3=C()
5=E()
18=*('mail','','')
24=*('^','11','P')
2=*()
))
```

```
#DOCPREP
'*** Word Processing Menu ***'
'1 - Create a new file'=C()
'2 - Edit an existing file'=E()
'3 - Look at file before printing'=*('fprint -t "^^" | more -d','12',')
'4 - File Checking Menu'=M('#CHECKER')
' '=>()
' '=>()
'D - Change Directory'=X()
'P - Printing System Menu'=M('#PRINTING')
'L - List Files'=?('*')
'? - HELP'=M('#HELP')
' '=>()
'Press <Escape> to go back a menu'=>()
' '=>()
27=^()
3=C()
5=E()
18=*('mail','','')
24=*('^','11','P')
2=*()
))
```

```
#MAIL
'*** Mail System Menu ***'
'1 – Check mailbox'=*('mail','','P')
'2 – Send electronic mail'=*('mail ^','8','P')
' '=>()
' '=>()
'D – Change Directory'=X()
'P – Printing System Menu'=M('#PRINTING')
'L – List Files'=?('*')
'? – HELP'=M('#HELP')
' '=>()
'Press <Escape> to go back a menu'=>()
27=^()
3=C()
5=E()
18=*('mail','','')
24=*('^','11','P')
2=*()
))

#HELP
'*** Help System Menu ***'
'1 – Command Summary -Quick Reference'=*('more –d /usr/lib/wp/wp.help','','')
'2 – Brief Command Summaries Menu'=M('#COMSUM')
' '=>()
' ---------- On-line Primer options ---------- '=>()
' '=>()
'3 – Using the Word Processor'=*('more –d /usr/lib/wp/prim1','','')
'4 – Using Menus'=*('more –d /usr/lib/wp/prim2','','')
'5 – Using Files'=*('more –d /usr/lib/wp/prim3','','')
'6 – Using Commands'=*('more –d /usr/lib/wp/prim4','','')
' '=>()
'Press <Escape> to go back a menu'=>()
' '=>()
27=^()
))
```

```
#COMSUM
'*** Brief Command Definitions Menu ***'
'A – Often Used Commands'=*('cat /usr/lib/wp/help/often','','')
'B – Cursor Movement'=*('cat /usr/lib/wp/help/cursor','','')
'C – Scrolling Text'=*('cat /usr/lib/wp/help/scroll','','')
'D – Storing Documents'=*('cat /usr/lib/wp/help/exit','','')
'E – Deleting Text'=*('cat /usr/lib/wp/help/delete','','')
'F – Inserting Text'=*('cat /usr/lib/wp/help/insert','','')
'G – Altering Text'=*('cat /usr/lib/wp/help/alter','','')
'H – Emphasising Text'=*('cat /usr/lib/wp/help/emphasis','','')
'I – Using Rulers'=*('cat /usr/lib/wp/help/rulers','','')
'J – Marking Text'=*('cat /usr/lib/wp/help/CP','','')
'K – Moving Text'=*('cat /usr/lib/wp/help/CP2','','')
'L – Locating and Replacing Text'=*('cat /usr/lib/wp/help/SR','','')
'M – Using the Printing System'=*('cat /usr/lib/wp/help/PT','','')
'N – Using Modes'=*('cat /usr/lib/wp/help/modes','','')
'Press <Escape> to go back a menu'=>()
27=^()
))
```

```
#PRINTING
'*** Printing & Mail Merge Menu ***'
'1 – Look at file before printing'=*('fprint –t "^^ ¦ more','12',")
''=>()
' ---------- Interactive Printing ---------- '.>()
'2 – Dot Matrix Printer (DMP110/200/2100)'=*('spooler –off;fprint –i –p DMP "^^"¦wp.filter> /dev/lp;spooler
–on', '12', ''
'3 – Daisy Wheel Printer (DWIIB/DWP210/410)'=*('spooler –off;fprint –i –p DWP "^^" ¦ wp.filter>
/dev/lp;spooler –on', '12', ''
' ---------- Spooled Printing ---------- '.>()
'4 – Dot Matrix Printer (DMP110/200/2100) =*('fprint –p DMP "^^"¦ wp.filter ¦ lpr –F x','12','12')
'5 – Daisy Wheel Printer (DWIIB/DWP210/410) =*('fprint –p DWP "^,^" ¦ wp.filter ¦ lpr –F x','12','12')

'6 – Draft Quality Non-Radio Shack Printer =*('fprint –p dumbprinter "^,^" ¦ lpr','12','R'
' '=>()
'V – Use Viewprint Screen'=*('spooler –off;viewprint; spooler –on','','','','R')
'M – Use Mail Merge Screen'=*('viewmerge','','','','')
'D – Change Directory'=X()
'L – List Files'=?('*')
'? – HELP'=M('#HELP')
' '=>()
'Press <Escape> to go back a menu'=>()
27=^()
3=C()
5=E()
18=*('mail','','','','')
24=*('^','11','P')
2=*()
))
```

```
#FILEMANAGE
'*** File Management Menu ***'
' ----- Files ----- '=>()
'1 - Copy a file '=D()
'2 - Erase a file '=K()
'3 - Rename a file'=R()
'4 - File Checking Menu'=M('#CHECKER')
' '=>()
' ----- Directories ----- '=>()
'5 - Open a new directory'=*('mkdir ^','7','P')
'6 - Remove an empty directory'=*('rmdir ^','7','P')
'7 - Change directory'=X()
'8 - List directory'=*('ls *','','P')
' '=>()
'P - Printing System Menu'=M('#PRINTING')
'? - HELP'=M('#HELP')
' '=>()
'Press <Escape> to go back a menu'=>()
27=^()
3=C()
5=E()
18=*('mail','','')
24=*('^','11','P')
2=*()
))
```

```
#CHECKER
'*** File Checking Menu ***'
'1 – Count lines, words, characters in a file'=*('wc ^','5','P')
'2 – Detailed information about a file'=*('ls –l "^"','5','P')
'3 – Find a file in a sub-directory'=*('find . –name "^" –print','10','P')
' '=>()
'D – Change Directory'=X()
'P – Printing System Menu'=M('#PRINTING')
'L – List Files'=?('*')
'? – HELP'=M('#HELP')
' '=>()
'Press <Escape> to go back a menu'=>()
27=^()
3=C()
5=E()
18=*('mail','','')
24=*('^','11','P')
2=*()
))
```

```
#SYSCOMMS
'*** Additional System Usage Menu ***'
' ----- Status -----'=>()
'1 – Who is also on the computer'=*('who','','')
'2 – Date and time'=*('date','','')
'3 – Calender'=*('cal ^| more –d','2','')
' '=>()
' '=>()
'D – Change Directory'=X()
'P – Printing System Menu'=M('#PRINTING')
'L – List Files'=?('*')
'? – HELP'=M('#HELP')
' '=>()
'Press <Escape> to go back a menu'=>()
27=^()
3=C()
5=E()
18=*('mail','','')
24=*('^','11','P')
2=*()
))

#EDITMENU
'*** Document Menu ***'
'1 – Comprehensive Help Menu'=M('#HELP')
'2 – UNIX Command Line '=*('^','11','P')
'3 – Check mailbox'=*('mail','','')
' '=>()
'Press <Escape> to get back to the document'=>()
' '=>()
27=*()
))
```

VT100 is a trademark of Digital Equipment Corporation.
VISUAL 55 and VISUAL 200 are trademarks of Visual Technology Inc.
Altos II is a trademark of Altos Computer Systems.

TRS–80 is a registered trademark of the Radio Shack Division of Tandy Corp. Radio Shack is a registered trademark of Tandy Corporation.

APPENDIX B
Viewprint and Mail Merge Files

This appendix contains the complete contents of the standard versions of the vprint and vmerge files. For more information about these files, refer to Chapter 11.

The vprint File

```
#DEFINE
-p @b -c@h -S@f -E@g -l@e -m@d @c @i @a @j
FIELDS = [_]
HELP=$-'h'
GO=$-'a'
QUIT=$-'q'
))
#SCREEN

            -------- LYRIX PRINT SET-UP SCREEN --------

Enter file to print: [aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa]

Select printer type: [bbbbb]        Interactive? (y/n) [cccccccccccccc]

Print from page: [ffffffffff]       to page:    [ggggg]

Number of copies to print: [hhhhh]      Page length: [eeeee]

Set left margin to: [ddd]               Print alternate pages? (y/n) [i]

 -----------------------------------------------------------

Accept <ESC> a          Quit <ESC> q           Help <ESC> h

))
```

```
#FIELDS
a=i()
a=p(Enter the name of the file to print)
b=a(DMP dmp DWP dwp DRAFT draft)
b=p(DMP – Dot Matrix Printer, DWP – Daisy Wheel Printer, DRAFT – Non-Radio Shack Printer)
b=&(DMP)
b=(DMP 'DMP')
b=(dmp 'DMP')
b=(DWP 'DWP')
b=(dwp 'DWP')
b=(DRAFT 'dumbprinter')
b=(draft 'dumprinter')
c=a(Y y N n)
c=p(Enter Y for interactive printing; default is spooled)
c=&(N)
c=m(Y '-i')
c=m(y '-i')
c=m(N '')
c=m(n '')
j=@c
j=m(y '| wp.filter >/dev/lp')
j=m(Y '| wp.filter >/dev/lp')
j=m(N '| wp.filter | lpr -F x &')
j=m(n '| wp.filter | lpr -F x &')
h=n(>0 <32767)
h=p(Enter number of copies to print)
h=&(1)
f=n(>0 <32767)
f=p(Enter page to start printing)
f=&(1)
g=n(>0 <32767 >f)
g=p(Enter page to stop printing)
g=&(6000)
```

```
e=n(>0)
e=p(Enter page length. Default is 66 lines)
e=&(66)
d=n(>=0 <245)
d=p(Enter left margin offset up to 245; Radio Shack DMP has one inch by default)
d=&(0)
i=a(y Y n N)
i=p(Enter Y to print alternate pages only)
i=m(y '-a')
i=m(Y '-a')
i=m(n '')
i=m(N '')
i=&(N)
))
```

The vmerge File

```
#DEFINE
-b @c -n@k -f'@g'-r'@h' @d @j @e @f @a < @b @i @l &
FIELDS = [_]
HELP=$-'h'
GO=$-'a'
QUIT=$-'q'
))

#SCREEN

          -------- LYRIX MAIL MERGE SET-UP SCREEN --------

Enter name of text file      [aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa]

Enter name of merge file     [bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb]

Print or save in file?       [i]     Output filename:  [IIIIIIIIIIIIIIIIIIIIIIIIII]

Stop page breaks? (y/n)      [c]     Maximum copies to print       [kkkkkk]

Justify Field:               [d]     Ignore commands in text? (y/n)     [j]

Extend Field:                [e]     Missing Variable: delete/ignore    [f]

End of field symbol:  [ggggg]        End of Record symbol:         [hhhhh]

 ------------------------------------------------------------

Accept <ESC> a          Quit <ESC> q          Help <ESC> h

))
```

```
#FIELDS
a=l
a=&(_)
a=p(Enter the name of your text file – the "blank form" to fill in)
b=l
b=p(Enter name of add file – [information to be merged into textfile])
c=a(y n Y N)
c=m(n '')
c=m(y '–p')
c=m(N '')
c=m(Y '–p')
c=p(Enter Y to stop page breaks between merge copies)
c=&(N)
d=a(l L r R s S i I)
d=m(l '–L')
d=m(L '–L')
d=m(r '–R')
d=m(R '–R')
d=m(s '–S')
d=m(S '–S')
d=m(i '')
d=m(I '')
d=&(I)
d=p(L – justify left, R – justify right, S – space on both sides, I – ignore)
j=a(y Y n N)
j=&(N)
j=p(Enter Y to ignore justification & spacing commands in file)
j=m(y '–N')
j=m(Y '–N')
j=m(n '')
j=m(N '')
e=a(t T e E n N)
e=m(t '–T')
e=m(T '–T')
e=m(e '–E')
e=m(E '–E')
e=p(E – extend field, T – truncate value when value is longer than field)
e=&(E)
```

```
k=n(>0 <32767)
k=p(Enter maximum number of copies to create)
k=&(6000)
f=a(d D i I n N)
f=p(If no add file entry matches the variable:  D to delete line, I to ignore)
f=m(d '-D')
f=m(D '-D')
f=m(i '')
f=m(I '')
f=&(I)
g=a()
g=p(Enter symbol for end of field.  Default is a new line)
g=&(\n)
h=a()
h=p(Enter symbol for end of record.  Default is an asterisk [*] alone on a line)
h=&(*\n)
i=a(p P f F)
i=m(p "¦uprint -p dumbprinter¦lpr2')
i=m(P "¦uprint -p dumbprinter¦lpr2')
i=m(f '>')
i=m(F '>')
i=&(P)
i=p(Enter P to print, F to save copy in a file)
f=o
f=+
f=p(Enter name of file in which to save copy)
))
```

APPENDIX C
Special Character Equivalents

This table is used when specifying keystroke sequences used in the #COMMANDS and all menu sections.

	Code		Code	+<Ctrl>		Code
space ...	32	@.....	64	0	'.....	96
!.....	33	A.....	65	1	a.....	97
".....	34	B.....	66	2	b.....	98
#.....	35	C.....	67	3	c.....	99
$.....	36	D.....	68	4	d.....	100
%.....	37	E.....	69	5	e.....	101
&.....	38	F.....	70	6	f.....	102
'.....	39	G.....	71	7	g.....	103
(.....	40	H.....	72	8	h.....	104
).....	41	I.....	73	9	i.....	105
*.....	42	J.....	74	10	j.....	106
+.....	43	K.....	75	11	k.....	107
,.....	44	L.....	76	12	l.....	108
-.....	45	M.....	77	13	m.....	109
.....	46	N.....	78	14	n.....	110
/.....	47	O.....	79	15	o.....	111
0.....	48	P.....	80	16	p.....	112
1.....	49	Q.....	81	17	q.....	113
2.....	50	R.....	82	18	r.....	114
3.....	51	S.....	83	19	s.....	115
4.....	52	T.....	84	20	t.....	116
5.....	53	U.....	85	21	u.....	117
6.....	54	V.....	86	22	v.....	118
7.....	55	W.....	87	23	w.....	119
8.....	56	X.....	88	24	x.....	120
9.....	57	Y.....	89	25	y.....	121
:.....	58	Z.....	90	26	z.....	122
;.....	59	[.....	91	27	{.....	123
<.....	60	\.....	92	28	|.....	124
=.....	61	].....	93	29	}.....	125
>.....	62	^.....	94	30	~.....	126
?.....	63	_.....	95	31	DEL.....	127

Index