Data General

Customer Documentation

FrameMaker®
Reference Manual

AViiON™
PRODUCT LINE

FrameMaker® Reference Manual

069-100332-00

For the latest enhancements, cautions, documentation changes, and other information on this product, please see the Release Notice (085-series) supplied with the software.

Ordering No. 069-100332
Rev. 00, December 1989

FrameMaker® Reference Manual
069-100332-00
069-100336-00 (Japan only)

Effective with: FrameMaker®, Revision 1.3
X Window System, X11 Version

Table of Contents

Introduction

This document describes all of the FrameMaker® program's commands and concepts in a reference manual style. We assume you have installed FrameMaker following the instructions in your FrameMaker Software Release Notice and have gone through the *FrameMaker User's Manual.* We also assume you know how to log in to your workstation and have a general understanding of how to use your operating system and your window system.

How This Manual Is Organized

In Chapter 2, you'll find instructions for starting and quitting FrameMaker and a description of important FrameMaker concepts. Chapter 3 includes a detailed description of each FrameMaker command that isn't specific to your workstation. In Chapter 4, you'll find a description of each tool icon in the Tools window. Chapter 5 covers the `fmbook` program, which allows you create documents such as tables of contents and indexes. Chapters 6 and 7 cover the Maker Markup Language (MML) and Maker Interchange Format (MIF) respectively.

Scope

This manual applies to FrameMaker version 1.3 only.

Make Backups Regularly

We have done extensive testing to ensure that you can use FrameMaker without any trouble. However, all computer systems are subject to failure due to hardware problems, power failures, and obscure bugs in operating systems or applications software.

It is good practice to save your documents periodically as you are working and to make regular tape backups of all of your files. If a problem arises, you could lose all the changes you made to a document since you last saved it. You should save a document whenever you have changed it enough that recreating the changes would require a significant amount of work. We suggest you save your work at least every 20 minutes.

Request for User Comments

Please call or write us if you experience any problems, if you have suggestions for improving the software or this documentation, or if you have questions. We are very interested in your input. We can be reached by contacting your local Data General sales representative.

FrameMaker Basics

This chapter covers the basic concepts and techniques of FrameMaker.

Conventions Used in This Manual

This manual uses certain format and style conventions. Please familiarize yourself with these conventions before continuing.

This section also shows how general key names used in this manual relate to keys and controls on your workstation, so read this section first.

- What you type is shown in

  ```
  bold text like this.
  ```

  You should type everything shown in bold text exactly as it appears in the documentation.

- What the system displays for you is shown in

  ```
  typewriter font like this.
  ```

  For example, in the line:

  ```
  tutorial% maker
  ```

  The system displays `tutorial%` (it is the system prompt). You type **`maker`** and press the Enter key.

  > **Note:** Each workstation has its own system prompt, so your system prompt probably isn't `tutorial%`. The prompt, however, probably does end with either a percent sign (%), dollar sign ($), or a number sign (#).

- In paragraphs, we show UNIX commands, such as *lp* and *enable*, in italics. In addition, we use italics to show emphasis or to refer to manual and section titles.

- When we indicate a particular key sequence, follow these instructions:

  If the key sequence looks like

  Ctrl-s

  with a hyphen separating the keys, press *and hold down* the first key, and then press the second key. In this example, you would press and hold down the Ctrl key and then press the unshifted S key.

If, on the other hand, the key sequence looks like

Esc e m

with no hyphen separating the keys, press and release the keys one after the other. In this example, you would press and release the Esc (Escape) key, the unshifted E key, and the unshifted M key in succession.

- When we tell you to press the Alt key, press either the key immediately to the left of the spacebar or the key immediately to the right of the spacebar. For example, if we tell you to

  Press Alt-d

  you should press and hold down either key and then press the unshifted D key.

Starting FrameMaker

In these instructions, we assume that you have the X Window system running on your workstation and that you have logged in (if you need help, see the manual that came with your workstation or your system administrator). We also assume that the FrameMaker directory is on your path (see your FrameMaker Software Release Notice for details).

The steps you follow next vary slightly depending on whether you installed a licensed version or a demonstration version of FrameMaker.

If you installed a licensed version of FrameMaker:

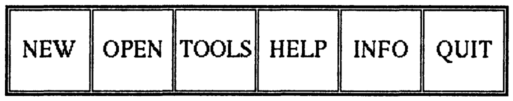
1. Move the mouse pointer within a shell window (for example, an xterm window) and type one of the following commands to start FrameMaker:

To start:	**Type:**
U.S. FrameMaker	`maker`
International FrameMaker (using the language for which your site is configured)	`imaker`
International FrameMaker (using UK English as the default language)	`imakerenglish`
International FrameMaker (using French as the default language)	`imakerfrench`
International FrameMaker (using German as the default language)	`imakergerman`

2. When the arrow pointer changes to an icon representing a window, move it to the position on the screen where you want the FrameMaker Message window to appear and click the left mouse button.

   FrameMaker opens the Message window and displays messages in the window while it loads. Keep the Message window visible while working, so you see any messages FrameMaker displays there.

The main FrameMaker window appears in the upper-right of the screen.



Main FrameMaker Window

3. To run through a self-guided demonstration, point on OPEN in the main FrameMaker window and click the left mouse button.

   The Open dialog appears.

4. Type `Demo.doc` and click OK. For a demonstration of the French language version of FrameMaker, type `Demo.doc.french` and click OK. For a demonstration of the German language version of FrameMaker, type `Demo.doc.german` and click OK.

When you use FrameMaker, you do so with either a reserved or a floating license. A reserved license is reserved solely for your use and is always available to you. Floating licenses are available to users on a first-come, first-served basis. If you are using a floating license and do not intend to use FrameMaker for a while, but do not want to quit FrameMaker, we recommend that you give up your license; your license then becomes available for another user. For more information about licensing, see the License command in Chapter 3.

If you installed a demonstration version of FrameMaker:

1. Type one of the following commands in the shell window to start the demonstration version of FrameMaker or FrameWriter.

To start:	**Type:**
U.S. FrameMaker	`demomaker`
International FrameMaker (using the language for which your site is configured)	`demoimaker`
International FrameMaker (using UK English as the default language)	`demoimakerenglish`
International FrameMaker (using French as the default language)	`demoimakerfrench`
International FrameMaker (using German as the default language)	`demoimakergerman`

   Messages appear in your console window during the 15 to 45 seconds the program takes to load.

   The Info dialog box appears, providing version and copyright information.

2. Read the information in the dialog box, point on the OK button, and click the left mouse button.

   A dialog box appears, suggesting you open the demonstration document `Demo.doc.usenglish`. If you're using the French language version of

FrameMaker, the demonstration document is named `Demo.doc.french`. If you're using the German language version of FrameMaker, the demonstration document is `Demo.doc.german`.

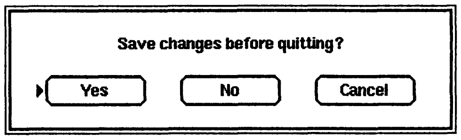
3. To open the demonstration document, click OK. To open any of the other demonstration documents described above, delete the contents of the edit box, type the document name, and click OK. To skip the demonstration document and begin using other FrameMaker features, click Cancel.

   The main FrameMaker window appears in the upper-right corner of the screen.

Quitting FrameMaker

1. Point on QUIT in the main FrameMaker window and click any mouse button.

   If there is any unsaved work, the Quit dialog box appears:



Save changes before quitting?

2. Click Yes to save your work before quitting (you see the Save dialog for each document with unsaved changes). Click No to quit without saving. Click Cancel to continue using FrameMaker instead of quitting.

Getting Help

You can get Help on using FrameMaker by clicking HELP in the main FrameMaker window. The first page of Help appears, showing an index of topics about which you can get more information. By pointing and clicking with the mouse, you can browse through the topics. See the Help command in Chapter 3 for more information or, better yet, click HELP and try it out—the first topic in the index is *Using Help*.

FrameMaker Concepts

Types of FrameMaker Documents

FrameMaker can be used both as a *document-oriented system*, suitable for creating reports or technical documentation such as this reference manual, and also as a *page-oriented system*, suitable for creating more complex page layouts such as those used in newsletters, flyers, and marketing literature. The next few sections of this chapter describe some of the basic concepts for using FrameMaker in those two different ways.

Overview of Creating a Typical Report

When using FrameMaker to create a report, technical document, or any other type of document whose format is consistent from page to page, all basic layout concerns and pagination are handled automatically by FrameMaker after you make a few basic formatting decisions.

For example, when you create a new document using the New command, you can use a template containing a preset page layout and paragraph formats or you can create a custom document, specifying the page size, margins, number of columns, and column spacing. FrameMaker then displays a new document window with the appropriate column layout already set up for you. As you enter new text, FrameMaker adds pages to the document as needed. FrameMaker automatically flows your text from column to column and from page to page as you edit it. Techniques for entering and editing text are described later in this chapter. At any time, you can use the Number of Columns command to reformat any or all pages using different column and margin settings.

FrameMaker has a rich set of paragraph formatting capabilities including automatic numbering, automatic bulleting, variable margins, four alignment styles, four kinds of tab stops with optional leaders, line spacing specified in points, optional automatic hyphenation, default paragraph fonts, and a variety of pagination (widow and orphan) control settings. You can give each paragraph a name, called the paragraph *tag*, that allows you to automatically reformat all paragraphs having the same tag, leaving other paragraphs unchanged. You control FrameMaker's paragraph formatting capabilities with the Paragraphs, Catalog, and Tabs commands described in Chapter 3.

FrameMaker offers a wide variety of fonts. The actual fonts available vary depending on the type of printer you use. Standard fonts include Times, Helvetica, Courier, and Symbol font families in sizes ranging from 7 to 24 points. Stylistic variations include *italic*, **bold**, ***bold italic***, <u>underline</u>, and ~~strike-through~~. Kerning and micropositioning are also available. You control these features using the Fonts command, which is described in Chapter 3.

You can insert graphics into the text so that they move with the text if the preceding text is edited. To do so, draw, import, or paste the graphics into *anchored frames*, which are described later in this chapter and under the Anchored Frame command in Chapter 3. There are eight anchoring settings, allowing you to use anchored frames for in-line illustrations, margin notes, quasi-footnotes, and top- or bottom-of-column illustrations.

You can also place graphics directly on a page (unanchored), so that their position is not affected by the editing of surrounding text. Instructions for creating both anchored and unanchored graphics are provided later in this chapter and in Chapter 4.

You can set up headers and footers using both the master page (see below) and the Headers & Footers command (see Chapter 3). You can use headers and footers to number pages automatically using arabic or roman numerals, or letters.

Finally, you can use the `fmbook` program to create tables of contents, lists of figures, indexes, and other types of lists automatically from one or more document files.

Overview of Creating a Typical Newsletter

All of the techniques described in the previous section also apply to creating newsletters and similar types of documents. If the newsletter has a simple layout, such as three columns per page with no items spanning the columns, then creating a newsletter is not very different from creating a report. The only exception is that reports are typically formatted using one or two columns, while newsletters typically use two or three columns.

Newsletters and marketing literature, on the other hand, often use more complex formats, where the layout of each page is unique. While each page usually has an underlying number of columns, items often overlap the columns, text may flow around freeform graphics, and columns may be subdivided to include photos or illustrations. You can use FrameMaker to create all of those layouts. In fact, FrameMaker supports totally arbitrary placement of text and graphics on a page.

The main technique associated with custom page layout concerns the placement of formatted text. FrameMaker uses a construct called a *TextRect* to control the placement of text on a page. A TextRect is a rectangle containing formatted text. FrameMaker allows you to manipulate TextRects as if they were objects on the page: You can move them to any position on the page and resize them. When a TextRect is resized, the text it contains is automatically reformatted to conform to the TextRect's new width and height. TextRects can also be linked together, so that text automatically flows from one TextRect to another as you edit it.

See the following command descriptions in Chapter 3 for information on setting up custom page layouts using TextRects: Connect TextRects, Disconnect Head, Disconnect Tail, Split TextRect, Auto Connect, and Feathering. Also see *Graphics Editing Overview* later in this chapter.

The FrameMaker Object Model

A FrameMaker document is composed of pages containing *objects*.

You use objects to create graphics and text. These objects include lines, arcs, squares, rectangles, circles, ovals, polygons, polylines, isolated TextLines, freehand lines, arrows, and TextRects containing formatted text. We call those things objects because FrameMaker allows you to manipulate them as if they were physical entities, much like shapes you paste onto a page to form a collage. In fact, the collage model is a good way of describing how text and graphics are created using FrameMaker.

For example, you use the Rectangle tool to draw a rectangle on a FrameMaker page. The result is very similar to placing a flat rectangular object on a sheet of paper. The similarity is that FrameMaker allows you to move the rectangle, stretch it, and place other objects on the page so that they overlap the rectangle. You can even change the front/back order of objects on a FrameMaker page, just as you can in a collage. Similarly, drawing a line on a FrameMaker page with the Line tool is like placing a piece of straight wire onto a sheet of paper. FrameMaker allows you to move the line (wire) around, grab an end and reorient it, and even change its length.

Text Objects

FrameMaker uses two types of objects for placing text on a page: TextLines and TextRects.

A TextLine is like a newspaper headline cut out and pasted onto a collage. It is always a single line of text, so inserting new characters causes the TextLine to get longer. TextLines are useful for things like picture captions. While they can contain a mixture of fonts, they are not used to create paragraph-oriented text. TextLines are created with the TextLine tool, described in Chapter 4.

TextRects are rectangular objects that you can fill with paragraphs of text. In most cases you can think of a TextRect as a *column*. Unlike rectangles created with the Rectangle tool (which are rectangles in the standard graphical sense of the word), TextRects normally do not have a visible border or fill pattern. Instead they define an area in which text is formatted. The Borders command on the Guides menu allows you to see the border of a TextRect as a dotted line. The single column of text on this page is actually in a TextRect. A two-column FrameMaker document has two TextRects on each page, one for each column.

The fact that TextRects are objects is especially useful in creating complex page layouts. You can move a TextRect anywhere on a page in the same way that you move lines and other graphic objects. You can draw any number of TextRects on a page using the TextRect tool (see Chapter 4, *Tools Window Icons*). TextRects can overlap other objects, and you can change their size. When you resize a TextRect, the text within it is reformatted to conform to the new TextRect width.

You can even give TextRects a visible border and fill pattern (including None, to create a transparent TextRect).

You can connect TextRects; think of this connection as an invisible pipeline between them. When you connect two TextRects, editing in the first TextRect may cause text to "spill over" from the first into the second TextRect. Similarly, if you make the first TextRect smaller, text that no longer fits in the first TextRect flows into the second one. You can set up chains of linked TextRects to create literally any kind of page and document layout, from simple one-column layouts to complex magazine or newspaper-style layouts.

A chain of linked TextRects is called a *text flow*. Your document can contain many independent text flows, and you can link a TextRect on one page to any other TextRect on any other page. Linked text flows allow newspaper-style layouts, in which a story on page 3 might be continued on page 18.

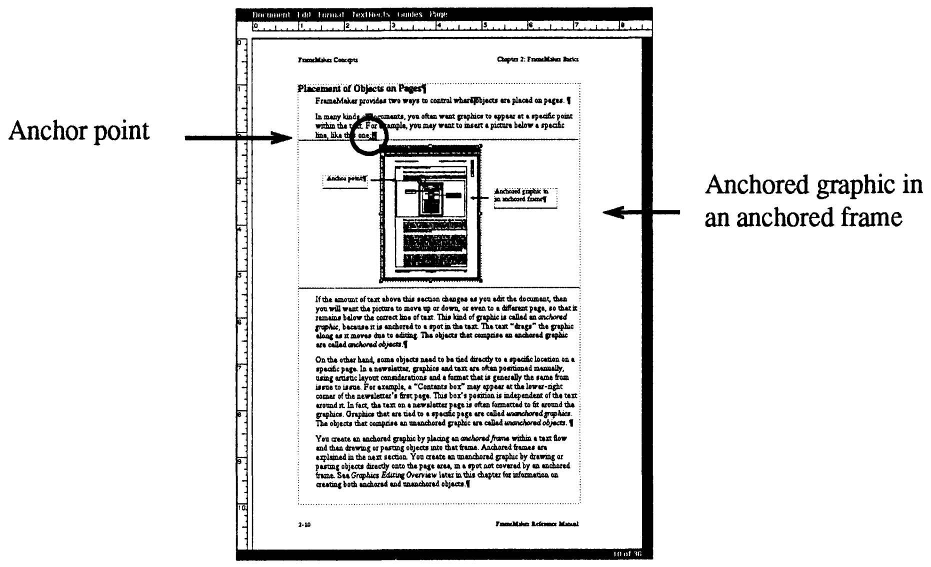
You can set up TextRects to generate a new page automatically whenever they become full (using the Auto Connect command). The number and location of TextRects on the new page are determined by settings established when the document is created, but you can change them at any time using the Number of Columns command. The master page shows where TextRects are placed on new pages.

Now you can understand how FrameMaker sets up standard multipage documents: When you start a new document, you tell FrameMaker what margins and how many columns to use. FrameMaker uses that information to automatically place TextRects on the first page of the new document, one TextRect for each requested column. If there is more than one column, the TextRects on the page are linked together. Also, the TextRects are set up with Auto Connect enabled. As you type the first page, FrameMaker keeps track of how much text is added. When there are too many lines to fit in the TextRects on the first page, FrameMaker generates a second page, and automatically links the last TextRect on page 1 to the first TextRect on page 2. Additional pages are added and linked as editing continues. This reference manual, for example, has one TextRect on each page, and each TextRect is linked to the one on the next page.

Placement of Objects on Pages

FrameMaker provides two ways to control where objects are placed on pages.

In many kinds of documents, you often want graphics to appear at a specific point within the text. For example, you may want to insert a picture below a specific line, like this one:



If the amount of text above this section changes as you edit the document, then you will want the picture to move up or down, or even to a different page, so that it remains below the correct line of text. This kind of graphic is called an *anchored graphic*, because it is anchored to a spot in the text. The text "drags" the graphic along as it moves due to editing. The objects that comprise an anchored graphic are called *anchored objects.*

On the other hand, some objects need to be tied directly to a specific location on a specific page. In a newsletter, graphics and text are often positioned manually, using artistic layout considerations and a format that is generally the same from issue to issue. For example, a "Contents box" may appear at the lower-right corner of the newsletter's first page. This box's position is independent of the text around it. In fact, the text on a newsletter page is often formatted to fit around the graphics. Graphics that are tied to a specific page are called *unanchored graphics.* The objects that comprise an unanchored graphic are called *unanchored objects.*

You create an anchored graphic by placing an *anchored frame* within a text flow and then drawing or pasting objects into that frame. Anchored frames are explained in the next section. You create an unanchored graphic by drawing or pasting objects directly onto the page area, in a spot not covered by an anchored frame. See *Graphics Editing Overview* later in this chapter for information on creating both anchored and unanchored objects.

Frames

In addition to the types of objects described in the previous sections, there is also a special kind of object called a *frame*. A frame is a rectangular object that can *contain* other objects (text and graphics), in the same way that a real-world picture frame contains a picture. That is, when you move a frame, everything in it moves too.

This property of frames allows you to anchor graphics to a position within a document's text flow, as explained in the previous section. To do so, place an anchored frame at a point in the text (using the Anchored Frame command) and then place objects within the anchored frame. FrameMaker moves the frame automatically whenever the text it is anchored to moves. Since moving a frame also moves its contents, the graphics in the anchored frame moves as needed to remain at the appropriate point in the surrounding text.

Unlike rectangles created with the Rectangle tool (which are rectangles in the standard graphical sense of the word), and unlike real-world picture frames, FrameMaker frames normally do not have a visible border or fill pattern. They define an area that contains other objects. The Borders command from the Guides menu allows you to see the border of a frame as a dashed line.

You can also use frames to *crop* objects. When you make a frame smaller, objects and parts of objects beyond the frame's border are not visible. This is again similar to a real-world picture frame, where any part of the picture that is beyond the frame's opening cannot be seen. You can "uncrop" objects within a frame by making the frame larger or by moving the objects so that they are totally within the frame's border.

Because frames can crop objects, it is sometimes useful to place an *unanchored* frame directly on a page. Specifically, if you want to place a graphic directly on a page (such as an imported bitmap image), and if that graphic needs to be cropped (perhaps because you want only part of the image to be visible), you would use the Frame tool to draw a frame directly onto the page and then draw or import the graphic into the frame. You would then crop the image by stretching the frame's borders as needed to obscure the appropriate portions of the graphic.

Master Page

When FrameMaker displays or prints a document page, it begins by showing all items from the master page. Then it shows the items that are unique to a specific page from the document, placing them as a layer over the master page items. Putting items on the master page is one way to create headers and footers (also see the Headers & Footers command in Chapter 3).

If you want items to appear on every page, you put those items on the master page. If you later add or modify items on the master page, their appearance on all other pages is instantly updated. Putting such items on the master page saves time and document storage space.

The master page can contain all types of objects that can be placed on main pages in a document, including text, drawn graphics, frames, and imported images.

When you display the master page, it shows three kinds of items:

- Automatic headers and footers set up using the Headers & Footers command.
- TextRects that will be placed on each new page added to the document. The layout of these TextRects is defined when a new document is created using the New command; you can change the layout at any time after a document is created using the Number of Columns command.
- Objects and frames that have been drawn on the master page. You can edit these items while viewing the master page.

You cannot edit the automatic headers, footers, and TextRects on the master page; they appear for reference only. To edit them, use the Number of Columns or Headers & Footers command.

Tagged Paragraphs and the Catalog

FrameMaker has two features that interact to give you powerful formatting control: paragraph tags and the Paragraph Catalog. You can use these features to enforce totally consistent formatting and to allow custom formatting where needed.

Paragraph tags: Each paragraph in a FrameMaker document can be given a name, called the paragraph *tag*. Effective use of tags has two requirements:

- Each paragraph must be assigned a name that describes its function or type.
- All paragraphs that have the same type must be assigned the same name.

For example, in this document all paragraphs that begin a section have the tag Section. Similarly, regular paragraphs like the current one are named Body, and each paragraph in a bulleted list is called Bullet.

If you tag all document paragraphs with meaningful names, it is easy to make dramatic cross-document reformatting changes. You can apply FrameMaker's formatting and font commands *in one step* to all paragraphs that have the same tag, leaving all the other paragraphs unchanged. For example, it would take only a few seconds to change the font size or line spacing for all Body paragraphs.

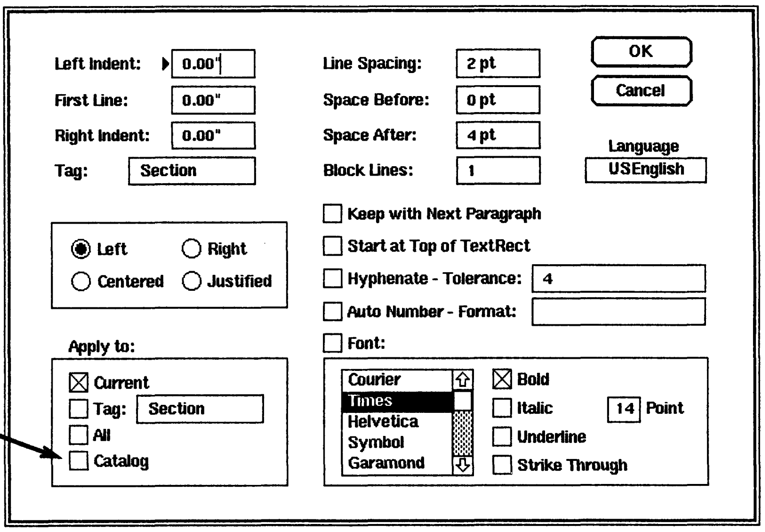
Each paragraph in a FrameMaker document contains its own unique formatting information (you can see a paragraph's formatting information by selecting the paragraph and using the Paragraphs command). This feature makes it *possible* for two paragraphs that have the same tag to have different formats.

In general, you will want paragraphs with the same tag to have the same format, but there are times when you will not want that to be the case. For example, you may decide to reduce the line spacing of a Body paragraph here and there to solve awkward page break problems.

Paragraph Catalog: You use the Paragraph Catalog to speed formatted text entry and to allow mass formatting updates. These are the steps for storing a paragraph format in the Catalog:

- Put the insertion point in a paragraph.
- Use the Paragraphs command to display the Paragraphs dialog box:

Turn on the Catalog setting


Left Indent: 0.00"
First Line: 0.00"
Right Indent: 0.00"
Tag: Section
Line Spacing: 2 pt
Space Before: 0 pt
Space After: 4 pt
Block Lines: 1
Language: USEnglish
[✓ Left] Right Centered Justified
Keep with Next Paragraph
Start at Top of TextRect
Hyphenate - Tolerance: 4
Auto Number - Format:
Font:
Apply to: [✓ Current] Tag: Section All Catalog
Courier [✓ Times] Helvetica Symbol Garamond
[✓ Bold] Italic Underline Strike Through
14 Point

- Fill in the desired settings.
- Turn on the Catalog check box in the Apply To area.
- Click OK.

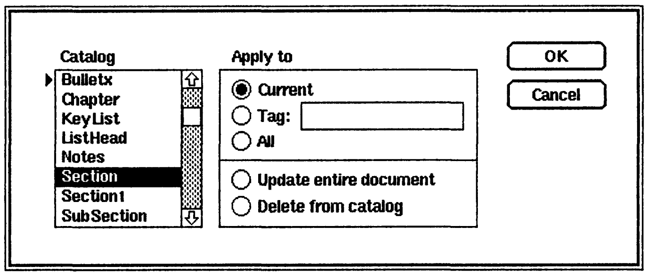
FrameMaker then stores all of the settings that appear in the dialog box, plus the tab settings that appear in the top ruler, in a Catalog entry whose name matches the paragraph's tag. In this case the Catalog entry would be called "Section."

Once you have stored a paragraph format in the Catalog, you can apply it to paragraphs as you create them or any time later.

For example, to start a new section, you might follow these steps:

- Put the insertion point at the end of an existing paragraph.
- Press the Enter key to begin a new paragraph.

- Choose the Catalog command to display the Catalog dialog box:



Apply to: [✓ Current] Tag: All

Update entire document
Delete from catalog

- Select Section in the Catalog scroll list and click OK or press the Enter key.

This procedure is the same as using the Paragraphs command and filling in all the values that appear in the dialog box using the settings that appear in the Paragraphs dialog box above.

You can also use the Catalog to store paragraph formatting information that you want to use in more than one document. You can *export* a document Catalog using the Maker Interchange Format (MIF) setting in the Save dialog box. After you create a Catalog MIF file, you can *import* that file into an existing document. This causes the Catalog entries from the MIF file to be merged with the existing Catalog. You can then click the Update Entire Document button in the Catalog dialog box to reformat the document using the imported Catalog. For more information on saving a document in MIF format, see the Save command in Chapter 3.

User Interface Basics

This section covers fundamental techniques for using FrameMaker. Later sections in this manual assume an understanding of the techniques presented here.

Noun-Verb Model

You carry out most actions in FrameMaker using what we call the noun-verb model. This means you first select text or an object and then choose what you want to do with what you've selected. For example, to delete text, you select the text (noun) by highlighting it and then press (verb) the Delete key.

Using the Mouse

The three mouse buttons are used for a variety of purposes, outlined below. The main use for the buttons is simple: Use the left mouse button to select objects and commands, use the middle button to select an insertion point in text or in a range of text, and use the right mouse button to display menus.

Use the left mouse button to:

- Select objects and frames for moving, stretching, and editing.
- Select commands from menu bar menus.
- Click on items in dialog boxes, the main FrameMaker window, and other FrameMaker windows (such as the Tools window and the Search window).

Use the middle mouse button to:

- Set an insertion point in text so you can insert new characters.
- Highlight a block of text for editing. After highlighting the block, you can:
  - Delete the block by pressing the Delete key.
  - Cut the block for later pasting using the Cut command from the Edit menu.
  - Change the block's font using the Fonts command from the Format menu.
- Deselect everything by clicking where there is no object.

Use the right mouse button to:

- Display the Maker menu (a menu containing commonly used commands): Point in the content area of a document window and press and hold down the right mouse button.

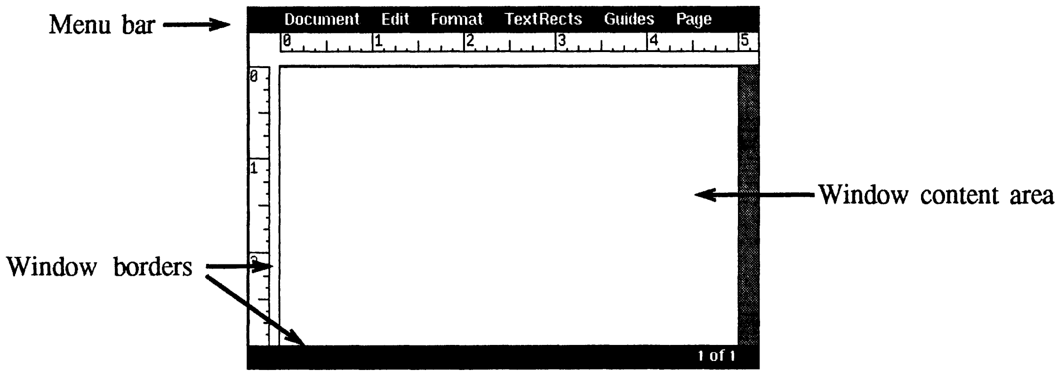
You can use several keyboard modifiers to change the meaning of the mouse buttons. See Appendix A, *Keyboard Commands,* for a complete list of these keyboard modifiers.

Using Windows and Menus

A FrameMaker document window is shown below. (On your system, the window may look slightly different. See your window system documentation for a description of system-specific window commands.)

You use different areas of the document window to perform different functions. With the arrow pointer in the content area of the document window, press and hold down the right mouse button to display the Maker menu, containing commonly used commands such as Undo, Cut, and Paste.

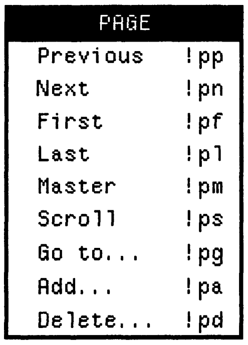
The menu bar contains six pop-up menus: Document, Edit, Format, TextRects, Guides, and Page. The commands on each FrameMaker menu are shown at the start of Chapter 3.



A FrameMaker Document Window

To use a command from a menu, follow these steps:

- Point on a word in the menu bar. For example, to use a command in the Page menu, point on the word *Page*.
- Press and hold down any mouse button (the menu pops up):



PAGE	
Previous	!pp
Next	!pn
First	!pf
Last	!pl
Master	!pm
Scroll	!ps
Go to...	!pg
Add...	!pa
Delete...	!pd

- Move the arrow pointer up or down while keeping the mouse button pressed, until the command you want is highlighted.
- Release the mouse button to select the command.

If you display a menu and want to put it away without using one of its commands, move the arrow pointer outside the menu so that no command is highlighted and then release the mouse button.

Keyboard Equivalents

You can also invoke all menu commands from the keyboard. There are three levels of keyboard commands:

- First, there are mnemonic three- or four-character commands, which appear on the menus (the exclamation point (!) represents the Esc key). For example, the keyboard equivalent for Next in the Page menu is `!pn`. This sequence means: Press and release the Esc key, then press the unshifted P key, and finally press the unshifted N key.

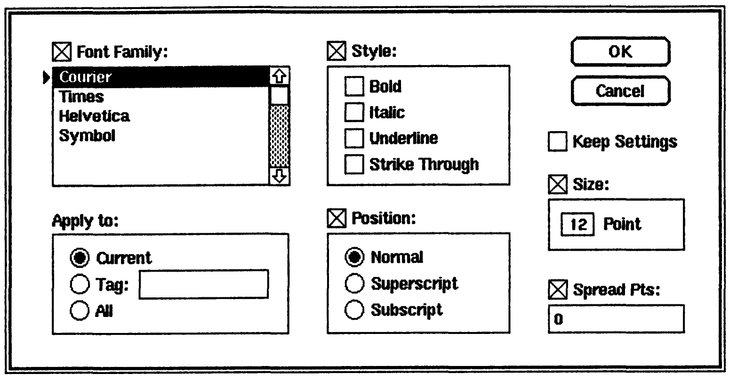
  The three characters used for the mnemonic commands follow a standard pattern. The first character is the Esc key, the second character is the starting letter in the menu's name, and the third character is the starting letter in the command's name.

  > **Caution:** Case is significant in keyboard commands. For example, the keyboard equivalent for the Markers command is Esc e m. That's lowercase *e* and lowercase *m*; uppercase *E* and *M* will not work. If a keyboard command does not seem to work, make sure the Caps Lock is not on.

- It can become tedious to repeatedly type three characters for the more commonly used commands, so shorter key sequences are also provided. While these sequences may be more difficult to remember, they can significantly improve your productivity once you have mastered them. For information on built-in keyboard sequences, see Appendix A, *Keyboard Commands*.
- Finally, you can record any sequence of keys and assign this sequence to any single key or sequence of keys (called a *macro*). Use macros to complete your most common typing and editing tasks with a minimum of keystrokes. See the Record Keys command and the Keyboard command in Chapter 3.

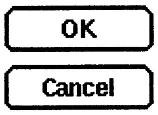
Using Dialog Boxes

A dialog box is a window that appears in the center of the screen when you use certain commands. Commands that need additional information use dialog boxes to gather that information. For example, the Fonts command from the Format menu displays this dialog box containing all of the types of items that a dialog box can contain:



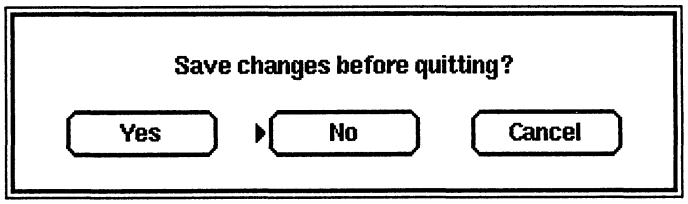
Buttons

Buttons are rounded rectangles containing a word, such as these OK and Cancel buttons:



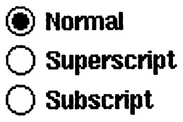
Clicking a button carries out an action (the specific action varies depending on the button). To *click* a button, point on the button and click the left mouse button.

You can also carry out the action of a button by making the button active and then pressing Enter. A button is active when the small triangle points to it. You move the small triangle using the Tab or Shift-Tab keys. For example, in the following dialog box, the No button is active—pressing Enter is the same as clicking No:



Radio Buttons

Radio buttons are small circles arranged in groups, such as the set of circles in the Position area in the Fonts dialog box.

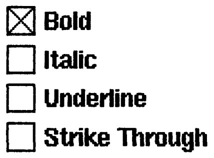


You use radio buttons to choose only one setting from a group of alternatives. For example, you can choose either Normal, Superscript, or Subscript, but not a combination of these.

One of the radio buttons in the group always contains a black dot, indicating that the alternative to its right has been chosen; the other buttons are empty. Clicking a radio button selects the alternative to the right of the button and deselects all other alternatives in the group.

Check Boxes

A check box is a small square to the left of a setting.

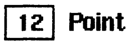


You use check boxes to choose whether or not an individual setting is turned on. For example, the font can be Bold or not. Unlike radio buttons, you can choose more than one setting from a list of check boxes. For example, you could turn on both Bold and Italic.

An *x* in a check box indicates that the setting to its right has been selected. Clicking in a check box alternately puts a check in the box or removes the check.

Scroll Boxes

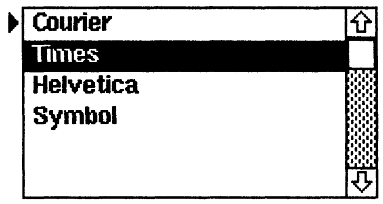
A scroll box is a small rectangle that contains one value from a small set of values, such as the box to the left of the word *Point* in the Fonts dialog box.



To scroll forward through the available values, click the scroll box until the desired value appears. To scroll backward, press and hold down the Shift key while clicking the mouse button.

Scroll Lists

A scroll list is a window containing a large list of items and scroll arrows for scrolling through the list. The Font Families list in the Fonts dialog box is an example of a scroll list.



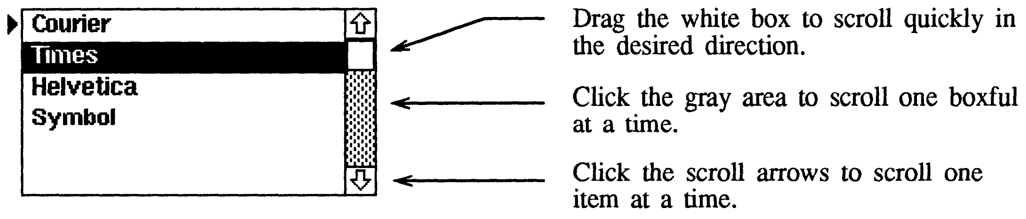
You can choose only one item at a time from a scroll list. For example, you can choose Helvetica, Times, or Courier but not both Helvetica and Times. (Because you can add font families to FrameMaker, the list can include many more than four items.)

To select an item from the list, click on it; the item becomes highlighted. If the scroll list is active (the small triangle points to it), you can also search for and select an item by typing the characters that uniquely identify the item. For example, to select Courier from the Font Families list, you could type `c`. Since Courier is the only item in the list that begins with a *c,* FrameMaker selects it.

If a scroll list contains more than one item that begins with the letter you type, FrameMaker selects the first matching item. If you continue to type letters, FrameMaker searches for and selects an item that has all the letters you type. For example, if your Catalog contains the tag names Step and Step1, you could type `step1` to select the tag Step1. This search feature in scroll lists is useful when creating keyboard macros that use the Catalog.

You can scroll through a list in several ways:

- To scroll one item at a time, click the up or down scroll arrows to move in the desired direction.
- To scroll one "boxful" of items at a time, click in the gray area between the up and down arrows. Click above the white square in the gray area to scroll up; click below the white square to scroll down.
- To quickly scroll to a relative position in the list, point in the white square in the gray area, press and hold down any mouse button, and drag the white square in the direction you want to scroll.


Courier
Times
Helvetica
Symbol
Drag the white box to scroll quickly in the desired direction.
Click the gray area to scroll one boxful at a time.
Click the scroll arrows to scroll one item at a time.

Edit Boxes

An edit box is a rectangle containing text you can edit.

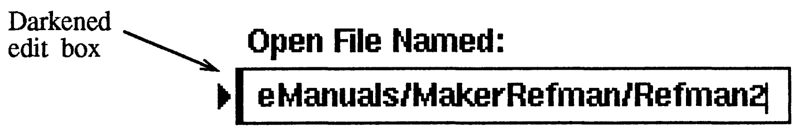
You use edit boxes to type text or numbers, such as page header strings, top margin settings, or first page numbers.

To edit text in an edit box, click in the box with any mouse button. An insertion point (I) appears in the text where you clicked. You can edit the text just as you edit text in a document; most of FrameMaker's keyboard commands for deleting and selecting text and moving the insertion point are available in edit boxes. For example:

- Press Ctrl-u to delete all the characters to the left of the insertion point.
- Press Ctrl-a to move the insertion point to the beginning of the line.
- Press Esc h p to select all the text in the edit box.
- Press Esc e u to undo the last change you made in the edit box.

For a complete list of FrameMaker's keyboard commands, see Appendix A, *Keyboard Commands*.

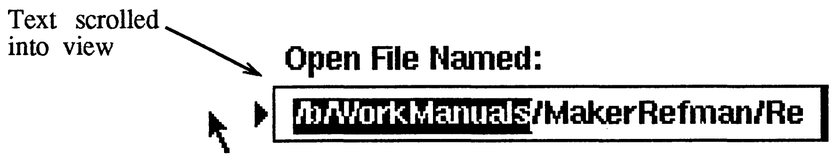
The text you type in an edit box can be longer than the edit box. As you type the extra characters, the first characters you typed scroll off the left edge of the edit box, but they are still stored within the edit box. The left edge of the edit box is darkened to indicate that there are extra characters to the left.



If you move the insertion point to the beginning of an edit box containing more text than can be displayed, the right side of the edit box is darkened to indicate that there are extra characters to the right.

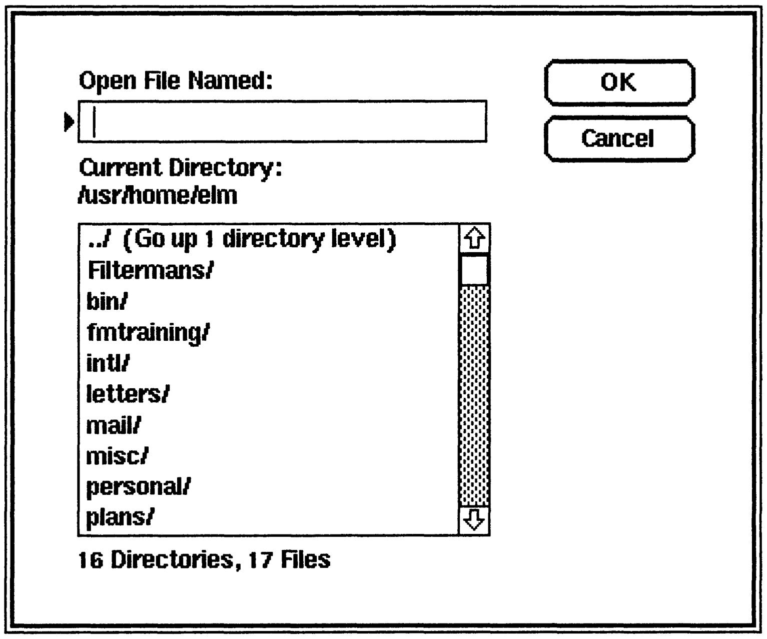
To see text that is beyond the edge of an edit box, use the keyboard commands to move the insertion point in the desired direction. For example, if there are extra characters to the left, press the Left Arrow key until the text beyond the left edge of the edit box becomes visible. (You could also press Ctrl-a to move to the beginning of the text.)

You can also select text that is beyond the edge of an edit box using either keyboard commands or the mouse. To select with the mouse, point on where you want to start selecting, press and hold down the middle mouse button, and drag the mouse in the desired direction. The text beyond the edge of the edit box scrolls into view as you select it.



The File Browser

The file browser, which consists of an edit box and a scroll list, allows you to see the contents of any directory you specify and to select a filename or directory without typing its pathname. In other words, it incorporates the properties of the UNIX commands *cd* and *ls*. You will find the file browser in the Capture, Import, Keyboard, New, Open, and Save dialog boxes. The Open dialog box is shown below:


Open File Named:

Current Directory:
/usr/home/elm

../ (Go up 1 directory level)
Filtermans/
bin/
fmtraining/
intl/
letters/
mail/
misc/
personal/
plans/

16 Directories, 17 Files

When a dialog box with a file browser appears, the scroll list displays all or part of a directory's contents, depending on what you specify. See the Capture, Import, Keyboard, New, Open, and Save commands in Chapter 3 for an explanation of what the scroll list initially displays when you select these commands.

If you want a file or document in *another* directory or need to look at the contents of another directory, you can:

- Click on a directory name in the scroll list and click OK. When you click on the directory name, it appears in the Open File Named edit box.

  The first entry in the scroll list (`../`) represents the parent directory (the one above the current directory); other directories in the scroll list contain a slash (/) after the name.

  You can also double-click a directory name in the scroll list, which is the same as selecting the filename and clicking OK.

- Type the directory's unique initial letters. (When selecting the directory in this way, make sure that the black triangle is pointing to the scroll list, not the edit box.) When the directory name appears in the edit box, click OK.

- Click on a directory similar to the one you want, edit the name that appears in the Open File Named edit box, and click OK.

- Type a pathname in the edit box and click OK. The pathname can be absolute (such as `/usr/home/emily`) or relative (such as `../plans/current`).

  You can use the directory abbreviations ~ (for the home directory) and .. (for the parent directory) as well as environment variables in the pathname.

As you change directories, FrameMaker updates the Current Directory pathname (above the scroll list) as well as the line that displays the number of directories and files in the current directory (below the scroll list).

If you want a file or document in the *current* directory, you can:

- Click on a filename in the scroll list and click OK. When you click on the filename, it appears in the Open File Named edit box.

  You can also double-click a filename in the scroll list, which is the same as selecting the filename and clicking OK.
- Type the filename's unique initial letters. (When selecting the filename in this way, make sure that the black triangle is pointing to the scroll list, not the edit box.) When the filename appears in the edit box, click OK.
- Click on a filename similar to the one you want, edit the name that appears in the Open File Named edit box, and click OK.
- Type the filename in the edit box and click OK.

You can also use the wildcard symbols ? (for a single character) and * (for any number of characters) to display some of a directory's contents. For example, if you type `*.doc` in the Open File Named edit box and click OK, a list of all filenames ending with `.doc` (in the current directory) appears in the scroll list. (Directory names are not affected.) If you then change the current directory, the scroll list will continue to display only some of the directory's contents. To display the entire contents of the directory again, type * in the Open File Named edit box and click OK.

Dialog Box Keyboard Commands

You can use keyboard commands to fill in a dialog box. These commands are useful not only for interactive work but also for use in keyboard macros (see the Record Keys command in Chapter 3):

Key	Meaning
Anywhere:	
`Tab`	Move to next item
`Shift-Tab`	Move to previous item
`Alt-Tab`	Move to first edit box
`Enter`	If no button is active, click OK If a button is active, click in the active button
`Ctrl-c`	Click in the Cancel button
In Check Box:	
`Space`	Mouse click (toggle)
`0`	Off
`1`	On
In Radio Button:	
`Any key`	Mouse click (turn on)
In Scroll Box:	
`Space` and `Up Arrow`	Mouse click (scroll to next setting)
`Down Arrow`	Scroll to previous setting
`Alt-Up Arrow`	First setting
`Alt-Down Arrow`	Last setting
In Scroll List:	
`Up` and `Down Arrows`	Click in the up and down scroll arrows (scroll list up or down)
`Alt-Up Arrow`	Select first item in list
`Alt-Down Arrow`	Select last item in list
`Shift-space`	Select item currently in first line of list
`Unshifted key(s)`	Search forward and select item starting with the typed letter(s)
`Shifted key(s)`	Search backward and select item starting with the typed letter(s)
In Edit Box:	
You can select and delete text and move the insertion point with keyboard commands, which are listed in Appendix A.	

Text Editing Overview

This section describes basic text entry and editing techniques.

Typing New Text

You can type text into TextRect and TextLine objects.

To add text, move the arrow pointer's tip into text, to the place where text is to be added, and click the middle mouse button. This action places the *insertion point* (⊥) between characters in the text at the point where you clicked the mouse button. Now, type text using the keyboard. The new text appears at the insertion point, and the insertion point advances to show where the next typed character will appear.

You cannot put the insertion point beyond the end of a line or below the last line in a TextRect. If clicking near text does not place an insertion point where expected, make sure the arrow pointer is within the TextRect or TextLine (turn on borders if needed).

The font used for inserted typing is called the *insert font*. FrameMaker resets the insert font whenever you explicitly reposition the insertion point using the mouse or keyboard commands (but not while you type). FrameMaker sets the insert font to match the font of the closest visible character on the line containing the insertion point. If you put the insertion point directly between two visible characters, the font of the character to the right is used. If the line contains no visible characters, the font of the end-of-line character is used. (You can see the end-of-line character if display of text symbols is enabled. See the Text Symbols command in Chapter 3.)

FrameMaker keeps track of line length as you type and automatically starts a new line when the current line overflows. The only time you need to press the Enter key while typing is when you want to end one paragraph and begin another. The new paragraph automatically has the same formatting characteristics as the previous paragraph. Each paragraph can have unique formatting characteristics. See the Paragraphs command in Chapter 3.

To end one line and move the insertion point to the start of a new line, without ending the paragraph, press Alt-Enter. This key sequence is useful for creating tables in which all lines in the table comprise one paragraph. When you change the tab settings for this type of paragraph, all lines in the table change immediately.

Correcting Typos

To delete a character you just typed, press either the Delete key or the Backspace key. If no text is selected, these keys delete the character to the left of the insertion point. When there is selected text, these keys delete the selected text and move the insertion point to the place where the deleted text began (see *Deleting a Block of Text* later in this chapter).

To delete the character to the right of the insertion point, press Ctrl-d.

To delete the word to the right of the insertion point, press Alt-d.

See Appendix A, *Keyboard Commands,* for information about other keyboard commands for text editing.

Highlighting a Block of Text

Highlight a block of text to modify all characters in that block as a unit. For example, you would first highlight a block of text when you want to delete, move, duplicate, or format the entire block.

To highlight a block of text, follow these steps:

- Move the arrow pointer to just before the first character or just after the last character of the block.
- Press and hold down the middle mouse button.
- Move the arrow pointer to the other end of the block without releasing the mouse button. As the arrow pointer moves, all text between the starting point and the arrow pointer is highlighted.
- When the desired text is highlighted, release the mouse button.

Text from one text flow may be highlighted across TextRect boundaries using the above technique as long as both the starting point and endpoint of the highlighting can be seen on the screen at the same time (see the Scroll command in Chapter 3).

There is a shortcut for highlighting a word:

- Move the arrow pointer's tip to anywhere within the word.
- Click the middle mouse button twice in rapid succession (this is often called double-clicking). The entire word will be highlighted (but not the space around it).

Extend text selection: You can highlight larger areas of text as follows:

- Move the arrow pointer to just before the first character or just after the last character of the block.
- Click the middle mouse button to put the insertion point at the arrow pointer's location.
- Move the arrow pointer to the other end of the block. You may use commands from the Page menu to move to another page.
- Press and hold down the Shift key and click the middle mouse button.

All text between the first insertion point and the Shift-click location is highlighted. If the highlighted region stops at the wrong character, Shift-click at a different location. Each time it is used, Shift-click resets the highlighting to the range between the insertion point and the click location. Use this Shift-click technique to select any block of text, from a few characters to an entire multipage document.

To deselect a block of text, click the middle mouse button anywhere in the document window or highlight a different block of text.

There are several keyboard commands for selecting text:

Key Sequence	Meaning
`!hc`	Select a character to the right of the insertion point
`!hw`	Select current word, then words to the right
`!hl`	Select current line, then lines below
`!hs`	Select current sentence, then next
`!hp`	Select current paragraph, then next
`!hf`	Shift selecting forward one character
`!hb`	Shift selecting backward one character
`!h0`	Clear selecting

Deleting a Block of Text

To remove a block of text, follow these steps:

- Select the block (see *Highlighting a Block of Text* earlier in this chapter).
- Press the Delete or Backspace key or use the Cut command.

These commands put the deleted text onto the clipboard for later pasting.

Moving a Block of Text

To move a block of text within a document or between documents, follow these steps:

- Select the block (see *Highlighting a Block of Text* earlier in this chapter).
- Use the Cut command or press the Delete or Backspace key to place the text on the clipboard.
- Put the insertion point where the cut text should be inserted by clicking there with the middle mouse button.
- Use the Paste command to insert the cut text at the insertion point.

Duplicating a Block of Text

To copy a block of text from one place to another within a document or between documents, follow these steps:

- Select the block (see *Highlighting a Block of Text* earlier in this chapter).
- Use the Copy command to copy the text onto the clipboard.
- Put the insertion point where the copied text should be inserted by clicking there with the middle mouse button.
- Use the Paste command to insert the copied text at the insertion point.

Text quick-copy feature: There is a faster way to copy small blocks of text from one place to another when both places are visible on the screen:

- Put the insertion point where you want the copy to go.
- Press and hold down the Shift and Ctrl keys and, using the middle mouse button, select the text you want to copy.

The selected text is copied to the original insertion point, retaining all of its font information, including style, size, and kerning. This quick-copy feature is especially useful for entering special symbols, whose keyboard position may be awkward or is unknown. See the on-line document `HelpSymbols.doc`.

Changing the Font of a Block of Text

To change the font of a block of text in a TextRect or a TextLine, follow these steps:

- Select the block (see *Highlighting a Block of Text* earlier in this chapter).
- Use the Fonts command.

Changing the Formatting of Paragraphs

To change the formatting (such as alignment, line spacing, and margins) of a paragraph, put the insertion point anywhere in the paragraph and use the Paragraphs, Catalog, or Tabs commands; or change the margin and tab settings by manipulating the corresponding symbols in the ruler at the top of the document window.

Keyboard Commands for Text Editing

You can use the arrow keys to move the insertion point (this method is faster than using the mouse to move the insertion point a few characters to the left or right). Appendix A, *Keyboard Commands,* lists all keyboard commands for moving the insertion point, selecting text, formatting, and the like.

Graphics Editing Overview

This section describes basic graphic, layout, and editing techniques.

Drawing Objects and Frames

You draw objects and frames using tools in the Tools window. See *The FrameMaker Object Model* and *Frames* earlier in this chapter for basic information on objects, frames, and the creation of page images.

You can place objects directly on the page (*unanchored*) or tie them to a location in a text flow (*anchored*).

Create unanchored objects by drawing within the document window on the surface of a page. These objects are directly tied to the page, as in a standard page layout program.

Create anchored objects by drawing within anchored frames or pasting predrawn objects into anchored frames. Create anchored frames using the Anchored Frame command. An anchored frame has an *anchor symbol* (⊥) somewhere within a text flow. As this symbol's position moves due to editing and formatting changes, the anchored frame's position also moves. This document contains many examples of anchored frames. For example, all of the pictures of dialog boxes in Chapter 3 are in anchored frames.

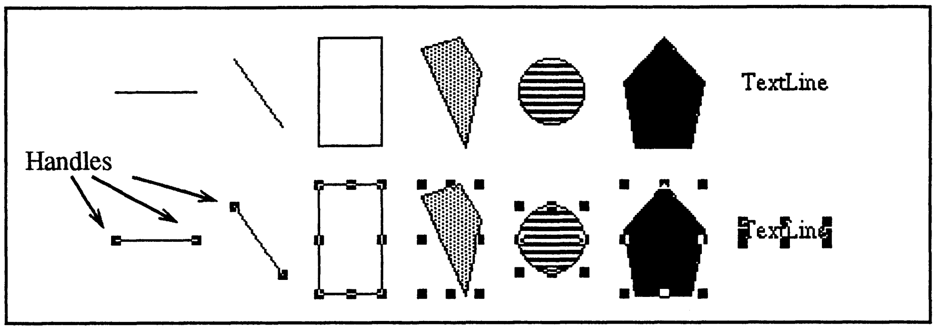
The basic steps for drawing an object or frame are as follows:

- Select the appropriate tool from the Tools window.
- Move the arrow pointer to the appropriate point in the document window.
- Stop moving the mouse for a moment, until the arrow pointer turns into a drawing pointer (⌖) or the TextLine I-beam pointer (⌶).
- Press and hold down the left mouse button at one corner of where you want to draw the object.
- Move the mouse without releasing the mouse button until the desired shape appears on the screen.
- Release the mouse button.

See Chapter 4, *Tools Window Icons,* for more detailed instructions.

Selecting Objects and Frames

To select a single object or frame: Move the arrow pointer so that its tip is near the edge of the object or frame and click the left mouse button. Handles appear to show that the object is selected:



Screen Appearance of Unselected (top) and Selected (bottom) Objects

Clicking anywhere in a closed, filled object or TextRect selects the object. But closed objects whose fill pattern is None are like empty wire frames, so clicking anywhere but on their edges does not select them.

If more than one object is under the arrow pointer when the left button is clicked, the object *on top* (in front) is selected. To select objects that are behind others, select the front objects and move them out of the way (see *Moving Objects and Frames within a Page or Frame,* later in this chapter) or send them to the back using the Back command.

To select more than one object or frame: Select each object one at a time with the left mouse button.

To select all objects in a rectangular area (selection border): Move the arrow pointer so that it is in one corner of the rectangular area and outside of all objects in the frame or page. Press and hold down the left mouse button. Move the arrow pointer to the diagonally opposite corner of the rectangular area; a *selection border* shows the area enclosed by the mouse movement. When the desired objects are completely enclosed by the border, release the mouse button. All objects within the selection border are selected.

Sometimes, when you try to draw a selection border, you select or move a single object instead (because you press the mouse button while the pointer is over an object). To avoid this problem, press and hold down the Shift key before pressing the left mouse button and then follow the steps above.

To deselect objects and frames: Click the middle mouse button anywhere in the document window or click with the left mouse button in an empty area of the window.

Sizing Objects and Frames

You can change the size of an object or frame using the mouse. In addition, you can scale objects using the Scale command.

Changing size with the mouse: To change the size of an object with the mouse, follow these steps:

- Select the appropriate object or frame by clicking near its border with the left mouse button.

  Two stretch handles appear on selected lines and arrows—one handle at each end. You use these handles to drag the associated end to a new location, changing the length and/or slope of the line or arrow.

  Eight stretch handles appear on other kinds of selected items—four corner handles and four side handles. Use a corner handle to drag that corner in or out while the opposite corner remains stationary. Use a side handle to drag the associated side in or out while the other sides remain stationary.
- Move the arrow pointer so that its tip is within one of the handles.
- Press and hold down the left mouse button.

  Dimension information appears in the status line at the bottom of the window, showing the current size of the item being stretched. If rulers are visible (see the Rulers command in Chapter 3), tick marks appear on the rulers, marking the location of the item's sides and center.
- Move the arrow pointer while holding the mouse button down to shrink or stretch the object or frame. As the item is stretched, the dimensions displayed in the status line and the tick marks on the ruler are updated.
- Release the mouse button when the item is the correct size.

All types of objects, except TextLines, can be resized with the mouse.

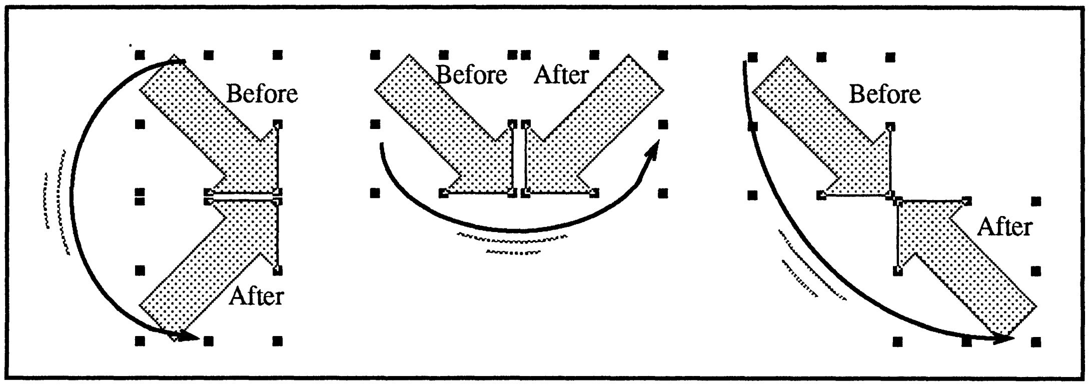
Resizing a TextRect causes the enclosed text to be reformatted.

You can resize groups of objects as a unit. To do so, select all of the objects to be resized, group them using the Group command in the Tools window, and follow the instructions for resizing individual objects (grouping objects effectively turns them into one object). Use this feature to create small, complex shapes: Draw the shape in a larger size using as many objects as needed, group the objects, and shrink them down to the size you want.

Forcing an object to stretch: If you miss the handle by mistake while trying to stretch an object, you may deselect the object or move it instead. You can use the Undo command to recover from an accidental move. To *avoid* this problem, however, press and hold down the Ctrl key while pressing the left mouse button. This action tells FrameMaker that you are trying to stretch something. In this case, FrameMaker looks for a handle near the arrow pointer and uses that if it can. If it cannot find a nearby handle, it ignores the mouse press, leaving everything selected and performing no other action.

Flipping Objects with the Mouse

You can flip most objects by dragging their stretch handles. For example, drag the top-center handle down to flip an object about its horizontal axis. Drag the left-center handle to the right to flip an object about its vertical axis. Drag the top-left handle down and to the right to flip an object in both dimensions:



You cannot flip imported images, TextLines, and TextRects.

Moving Objects and Frames within a Page or Frame

You can move objects and frames using the mouse or the arrow keys. To move an object or frame with the mouse, follow these steps:

- Select the appropriate object or frame by clicking near its border with the left mouse button.

  Handles appear on the selected item.

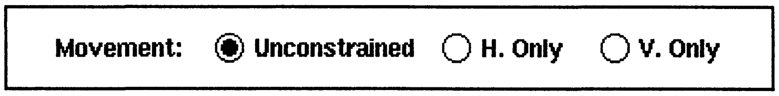
- Move the arrow pointer so that its tip is along the selected item's border but not near any of the item's handles.
- Press and hold down the left mouse button.

  If rulers are displayed, tick marks appear on the rulers, marking the location of the item's sides and center.

- Move the arrow pointer while holding the mouse button down to drag the object or frame to a new location. As the item is moved, the tick marks on the ruler are updated.
- Release the mouse button when the item's position is correct.

Forcing an object to move: If you miss the item's border by mistake while trying to move it, you may deselect it or stretch it instead. Inadvertently stretching a very small object can easily happen because its handles completely cover its border. You can use the Undo command to recover from accidentally stretching an object. To *avoid* this problem, however, press and hold down the Ctrl key while you press the right mouse button. This action tells FrameMaker that you are trying to move something. In this case, FrameMaker looks for an object that encloses the arrow pointer or for a border that is near the arrow pointer, and it uses that object if it can. If it cannot find a nearby object, it ignores the mouse press, leaving everything selected and performing no other action.

Use the Movement Constraint settings in the Tools window to affect how objects can be moved with the mouse. For example, to move an object horizontally without affecting its vertical position, click H. Only before moving the object. The Movement Constraint settings stay in effect until you change them.


Movement: [✓ Unconstrained] H. Only V. Only

Movement Constraint Settings in the Tools Window

Use the Snap command to affect how objects can be moved with the mouse. When you turn the snap feature on, if the first direction in which you move an object is to the left, then, as you move it, the left side of the object snaps to positions that correspond to the ruler divisions. You can snap the other sides of an object to ruler division positions by moving the object in the direction that corresponds to the appropriate side.

You can move all types of objects with the mouse. Moving a TextRect has no effect on the format of the text it contains or on its links with other TextRects.

To move an object or frame using the arrow keys, follow these instructions:

- Select a single object, group, or frame by clicking near its border with the left mouse button.

  Handles appear on the selected item.

- Press and hold down the Ctrl key and press one of the four arrow keys to move the item 1 point in the indicated direction.

  Also, you can press and hold down the Shift key and press an arrow key to move the item 1/2 pica (6 points) in the indicated direction.

You can move groups of objects as a unit. To do so, select all of the objects to be moved, group them (see the Group command in Chapter 3), and follow the instructions for moving individual objects (grouping objects effectively turns them into one object).

Moving Objects and Frames between Pages or Frames

You cannot move an object or group directly from one frame or page to another. To move an object or group to a different page, anchored frame, unanchored frame, or document, delete it from its original location using the Cut command, and then put it in its new location using the Paste command. To force an object to be pasted into a frame, select the frame or an object within the frame before using the Paste command. Similarly, to force an object to be pasted onto a page (and *not* within a frame on the page), select an object on the page before pasting.

Deleting Objects and Frames

To delete objects or frames, select them and use the Cut command. The Delete key is a shortcut for Cut.

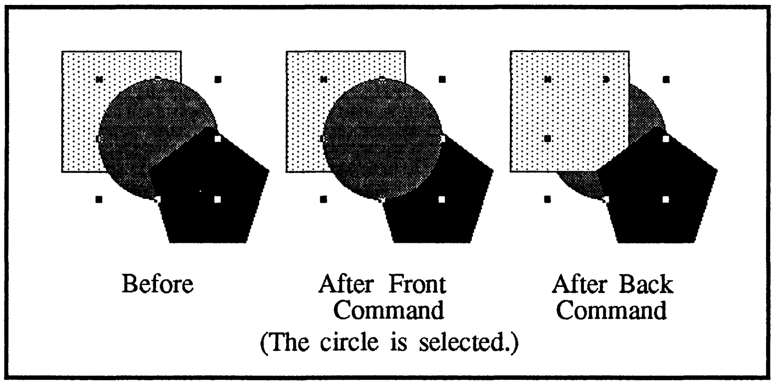
Duplicating Objects and Frames

To duplicate objects or frames, select them and use the Copy command. Copy stores a copy of the selected objects or frames onto the clipboard. Then use the Paste command to copy whatever is on the clipboard to a new location.

Quick-copy feature: To duplicate an object or group within one page or frame, use the quick-copy feature. First select the object you want to copy. Point on the object, and while holding down the Shift key, press the right mouse button. When you move the arrow pointer, you move a copy of the object. The quick-copy feature makes it easy to build pages that contain many of the same objects, such as tables made of TextRect cells.

Changing Object and Frame Overlap

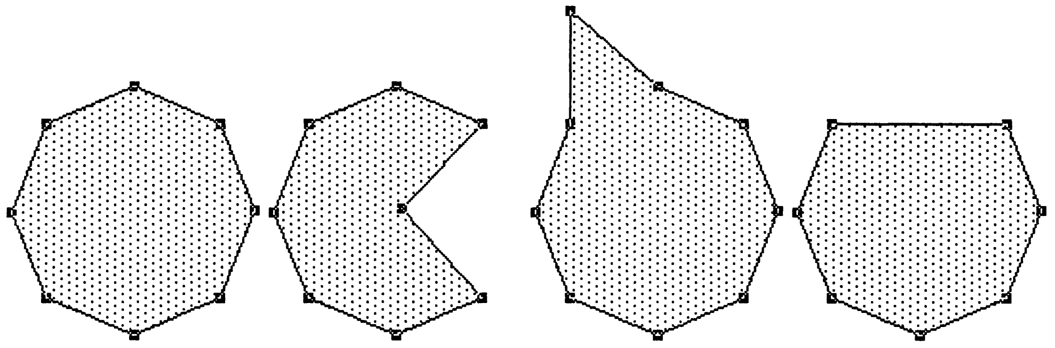
To change the overlap of objects or frames, select one and send it to the back or bring it to the front as needed using the Front and Back commands from the Tools window. These commands can also be used when more than one item is selected as well as on groups.



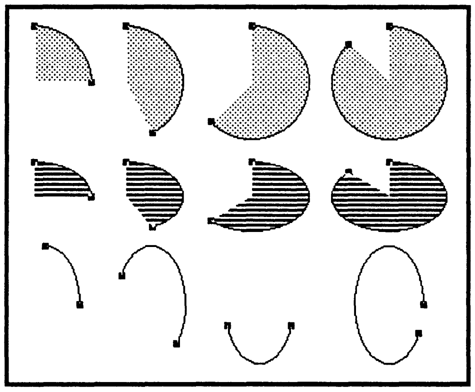
Reshaping Objects

You can reshape polygons, polylines, and freehand shapes with the Reshape command from the Tools window. You can also add points to or delete points from these objects:

- To add a point, select the object and use the Reshape command to turn on reshape handles. While holding down the Ctrl and Shift keys, click with the right mouse button somewhere along the object's border.
- To delete a point, select the object and use the Reshape command to turn on reshape handles. While holding down the Ctrl and Shift keys, click with the left mouse button on the point you want to remove.

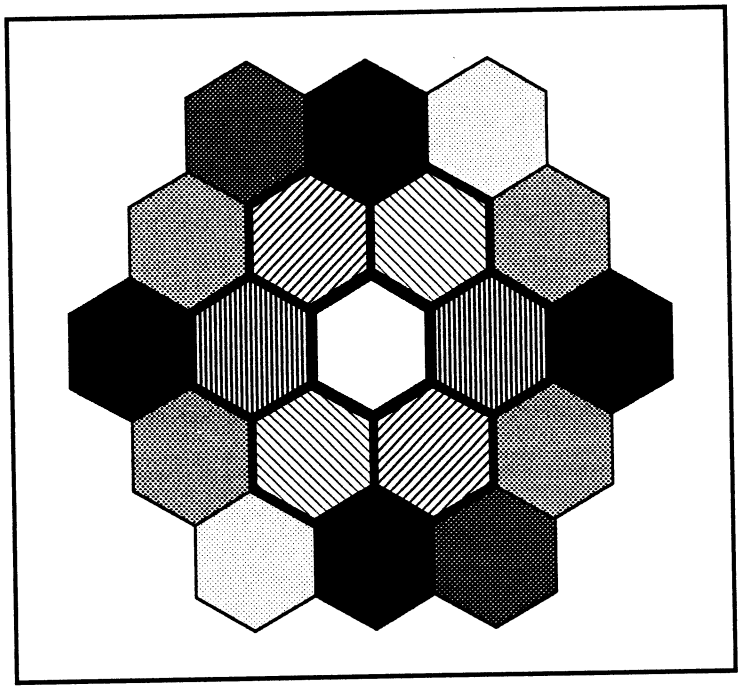

Selected polygon after clicking on the reshape command
After moving a point
After adding and moving a point
After deleting a point

You can also reshape arcs. Reshaping an arc changes its start and end angle, leaving the arc's underlying ellipse unchanged:



Changing Object Fill, Border, and Line Width

You can change the border width, border pattern, and fill pattern of objects and frames by selecting them and then clicking on the Widths, Fills, and Borders icons in the Tools window (see Chapter 4, *Tools Window Icons,* for more information).

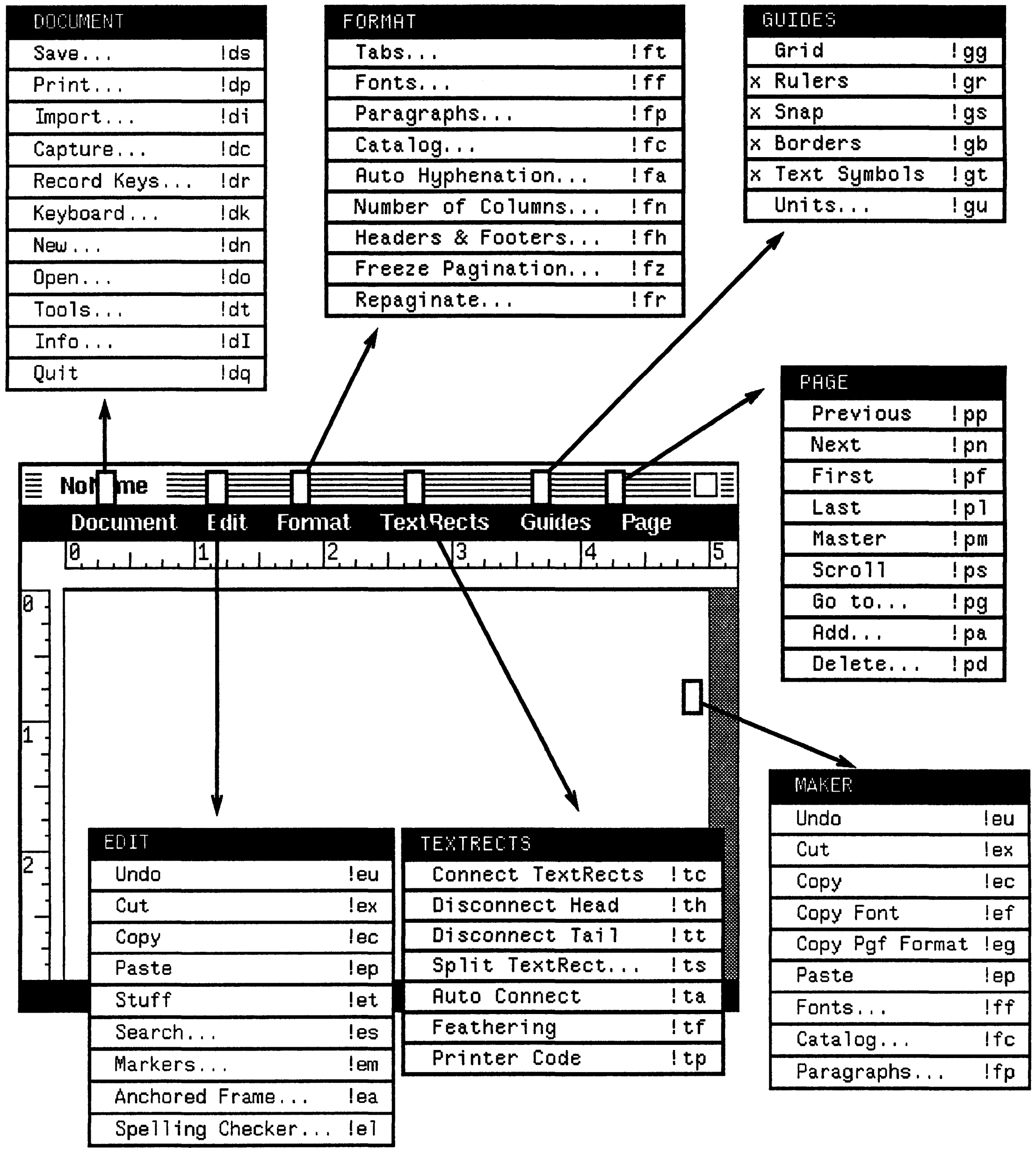


FrameMaker Commands

This chapter covers the FrameMaker commands in alphabetical order.

Menu Overview

You can choose FrameMaker's commands from the menus shown below, from the Tools window, or from the keyboard. See Chapter 2 for instructions on using menus; see Appendix A for information on keyboard commands.



To display a menu, put the arrow pointer on the indicated area and press the right mouse button.

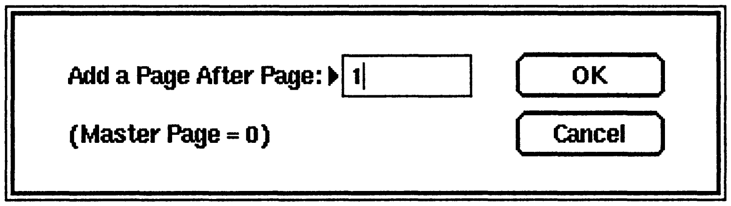
Add Page

Location: Page menu

Standard Keyboard Equivalent: `Esc p a`

Purpose: Use the Add Page command to add a new page after any existing page in the document. The new page's column layout is determined by the master page column layout settings at the time you choose Add Page. Columns on the new page are not linked to columns on other pages in the document. To link them, use the Connect TextRects command. See the Number of Columns command in this chapter and *Master Page* in Chapter 2.

Use: Choose the Add Page command to display the Add Page dialog box:



Add a Page After Page: In this box, type the number of the page that the new page will follow. If the first page of the document is not page 1 (in other words, if you have changed the First Page # in the Headers & Footers dialog box), then this number must be the actual number of the page, not its relative page number within the document. For example, if `Chapter2.doc` begins on page 21, and you want to add a page after the first page in that document, then type **21** in the Add a Page After Page edit box.

Notes:

- FrameMaker automatically creates new pages as needed when you add text to TextRects that have the auto-connect feature turned on. See the Auto Connect command.

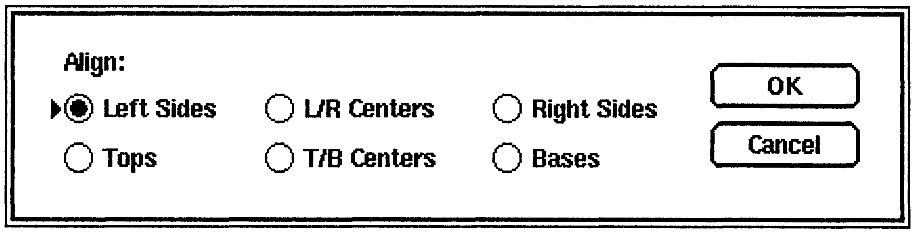
Align

Location: Tools window

Standard Keyboard Equivalent: `Esc o a`

Purpose: Use the Align command to move objects so that their edges or center points all lie on a straight line. The last object you select (known as the *reference object*) is the one to which the other objects align. You can also use this command to align an object within a frame or to set the alignment of TextLines.

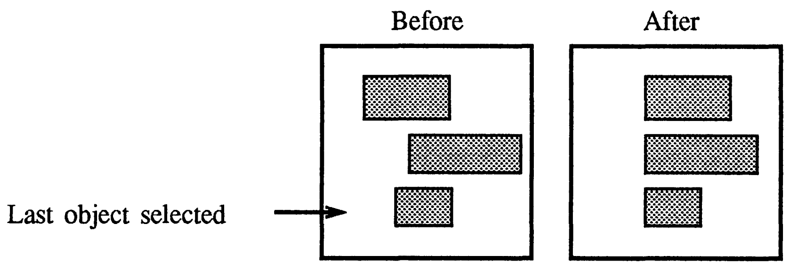
Use: To align objects, select them using the mouse, selecting the reference object last. Then click Align in the Tools window to display the Align dialog box:



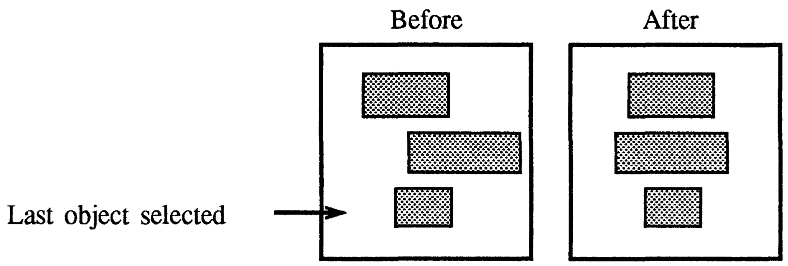
Align:
[✓ Left Sides] L/R Centers Right Sides
Tops T/B Centers Bases

Align: Click one of the settings in this area to specify how objects are to align.

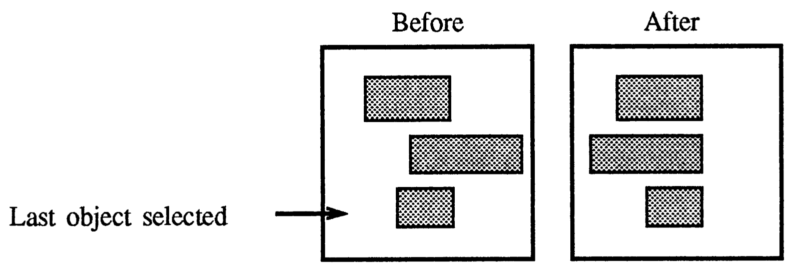
Left Sides: Click Left Sides to move all selected objects so that their left sides line up with the left side of the last selected object:



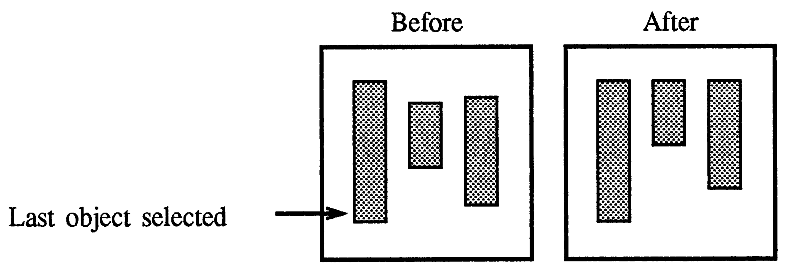
L/R Centers: Click L/R Centers to move all selected objects so that their left/right centers line up with the left/right center of the last selected object:



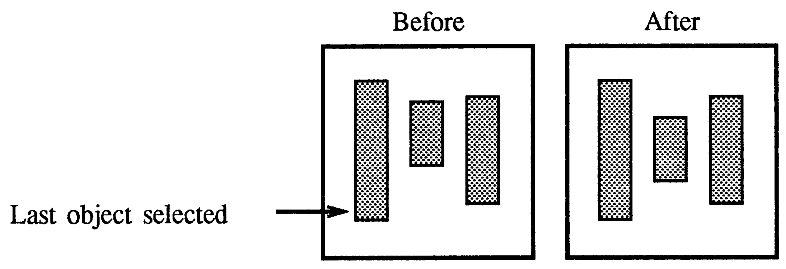
Right Sides: Click Right Sides to move all selected objects so that their right sides line up with the right side of the last selected object:



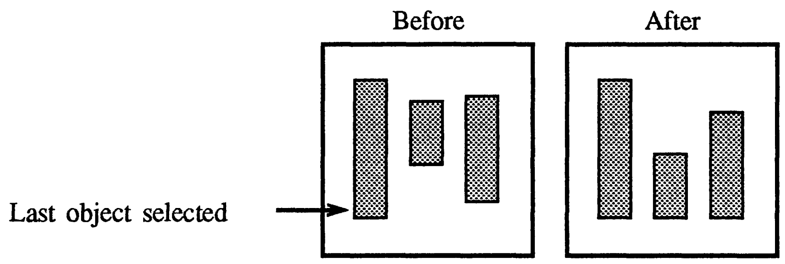
Tops: Click Tops to move all selected objects so that their top edges line up with the top of the last selected object:



T/B Centers: Click T/B Centers to move all selected objects so that their top/bottom centers line up with the top/bottom center of the last selected object:



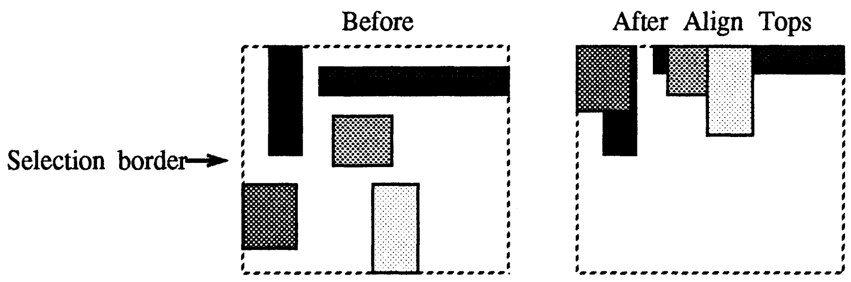
Bases: Click Bases to move all selected objects so that their bases line up with the base of the last selected object:



The base of a TextLine is its baseline (letters such as *y* and *j* hang down below the baseline). The base of other types of objects is their bottom edge.

Notes:

- If you select only one object, the object is aligned within its enclosing frame or page.
- If you select all objects with a selection border (see Chapter 2), the objects are aligned using the selection border:



- If you align a TextLine using the Left Sides, L/R Centers, or Right Sides settings, the TextLine "remembers" the Align setting when it is edited, expanding or shrinking as needed to maintain the position of the desired alignment point. See the TextLine tool in Chapter 4.
- The keyboard commands for paragraph alignment also work for object alignment: Esc j l aligns left sides, Esc j c aligns left/right centers, and Esc j r aligns right sides.

Anchored Frame

Location: Edit menu

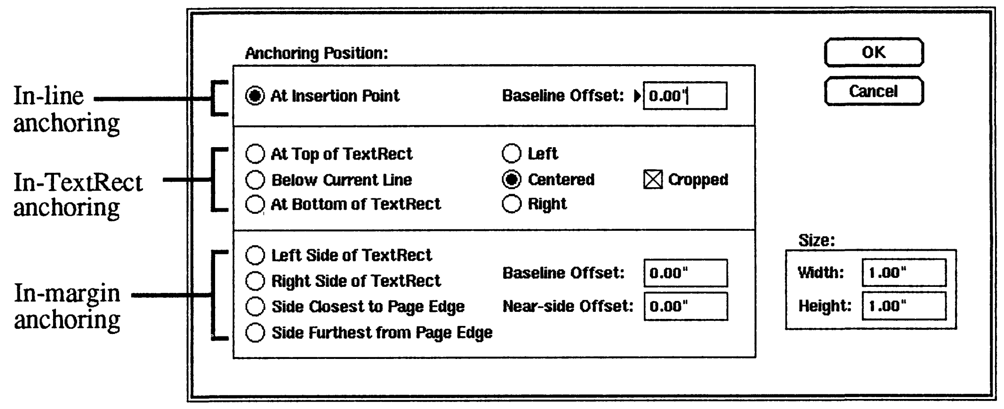
Standard Keyboard Equivalent: `Esc e a`

Purpose: Use the Anchored Frame command to create a new anchored frame and to change the properties of existing anchored frames.

An anchored frame is a frame whose position is tied to a point between two characters in a text flow. When you want to "attach" text or graphic objects to text so that the objects move with the text, put an anchored frame at the appropriate point in the text flow and then put those objects in the anchored frame. In this manual, the pictures of dialog boxes and other graphics are in anchored frames.

Use: To create a new anchored frame, move the arrow pointer to the desired anchor position within some TextRect text, click the middle mouse button to set an insertion point, and then choose Anchored Frame from the Edit menu.

To change the properties of an existing anchored frame, select the frame's anchor symbol (⊥) by highlighting it with the middle mouse button. Or, select the frame itself by clicking on its border or by putting a selection border around it (see *Selecting Objects and Frames* in Chapter 2). Then choose Anchored Frame from the Edit menu to display the Anchored Frame dialog box. If you have selected an existing frame, the dialog box shows the current properties of that frame. Otherwise it shows the settings from the previous use of the Anchored Frame command:



Anchoring Position:

[✓ At Insertion Point] Baseline Offset: 0.00"

At Top of TextRect
Below Current Line
At Bottom of TextRect

Left
[✓ Centered]
Right

[✓ Cropped]

Left Side of TextRect
Right Side of TextRect
Side Closest to Page Edge
Side Furthest from Page Edge

Baseline Offset: 0.00"
Near-side Offset: 0.00"

Size:
Width: 1.00"
Height: 1.00"

Anchoring Position: Choose one of the eight settings to describe the spatial relationship between an anchored frame and its anchor point. These settings fall into three categories:

- In-line anchoring: The frame appears next to the anchor point.
- In-TextRect anchoring: The frame appears in the same TextRect, either above or below the anchor point.
- In-margin anchoring: The frame appears in the left or right margin of the page.

In-Line Anchoring: With in-line anchoring, the frame appears next to the anchor point.

- **At Insertion Point:** Click At Insertion Point to place the frame immediately following the anchor point, as if the frame were a single character. The frame's height contributes to the overall height of the line containing the frame. If the frame is not surrounded by spaces, it is considered part of a word and contributes to wordwrapping decisions. It is usually a good idea to put a space before and after the anchor point, so that the frame is treated as a one-character word (like " a ") and is therefore free to float from line to line when the text before it is edited.
- **Baseline Offset:** In this box, type a value to specify the frame's location relative to the baseline of the surrounding text. An offset of 0 aligns the bottom edge of the frame with the baseline. Positive offset values move the frame up; negative offset values move the frame down.

In-TextRect Anchoring: With in-TextRect anchoring, the frame appears in the same TextRect, either above or below the anchor point.

- **At Top of TextRect**, **Below Current Line**, and **At Bottom of TextRect:** Click one of these settings to place the frame within the same TextRect as the anchor point, as if the frame were a complete line. The At Bottom of TextRect setting provides a rudimentary way to create footnotes.

  If a page contains multiple in-TextRect anchored frames, they are stacked top to bottom within the TextRect in the order that their anchor points occur in the text.
- **Left**, **Centered**, and **Right:** Click one of these settings to control the alignment of in-TextRect anchored frames. If the anchored frame is the full width of the TextRect, these settings do not matter. If the frame is narrower than the TextRect, then the frame is aligned within the TextRect according to these settings.

  For example, if you want a to center a 2"-wide figure within your text column, make the anchored frame just wide enough for the figure and click Centered. You are then free to format the text in one wide column or several narrower ones without concern for the figure's alignment; it is always centered within the column.
- **Cropped:** Turn on the Cropped check box if you want to crop the anchored frame to the width of the TextRect. Turning on Cropped prevents frame contents from spilling over the TextRect edges.

In-Margin Anchoring: Choose one of these settings to place the frame outside of the TextRect containing the anchor point. Use these anchoring positions for margin notes and illustrations.

- **Left Side of TextRect** and **Right Side of TextRect:** Click one of these settings to place the frame to the left or right of the TextRect, generally next to the line containing the anchor point.
- **Side Closest to Page Edge:** Click this setting when you have a two-column layout. When the anchor point is in the left column, the frame appears in the left margin (because the left side of the column is closer to a page edge than the right side). If when you edit text the anchor moves into a righthand column, the frame flips to the right margin.
- **Side Furthest from Page Edge:** Click this setting to put the frame in the widest margin, for example, when you specify one-column, two-sided layouts that have a wide outer margin for illustrations or margin notes. When you click Double Sided in the Headers & Footers dialog box, the columns shift left or right from page to page. For example, the wider margin may be on the left on odd-numbered pages and on the right on even-numbered pages. This setting puts the frame in the widest margin, no matter what page the frame occupies.
- **Baseline Offset:** In this box, type a value to specify the vertical position of the anchored frame relative to the baseline of the text containing the anchor point. An offset of zero causes the bottom edge of the frame to line up with the baseline. Positive offset values move the frame up; negative offset values move the frame down.

  This offset value is "springy"; that is, the anchored frame cannot go above or below the top or bottom of the TextRect containing the anchor point. If the Baseline Offset value you supply would cause the frame to be too high or too low, the frame sits as high or as low as possible. If you edit the text above the anchor point so that the anchor point moves away from the top or bottom of its TextRect, the anchored frame position is readjusted to try to use the requested Baseline Offset.
- **Near-Side Offset:** In this box, type a value to specify the gap between the anchored frame and the TextRect containing the frame's anchor point. Positive values move the frame away from the TextRect; negative values pull the frame back into the TextRect.

  When using negative Near-Side Offset values, note that the text within the TextRect does not automatically adjust to accommodate the frame; you must set paragraph indent values appropriately so that the anchored frame does not obscure the text in the TextRect.
- **Size:** In these boxes, type the desired width and height of the anchored frame.

New/Edit Frame: Click this button to insert a new anchored frame or to change the properties of the selected anchored frame.

Notes:

- It is easier to work with anchored frames when borders are displayed. FrameMaker automatically turns on borders when you choose the Anchored Frame command to add or edit an anchored frame. See the Borders command.
- Once an anchored frame is in place, you can change its Size, Baseline Offset, and Near-Side Offset values by selecting the frame and moving or stretching it using the mouse or arrow keys (see *Moving Objects and Frames within a Page or Frame* in Chapter 2).
- You can select anchored frames as objects or as characters.

  When you select an anchored frame as an *object*, handles appear on the frame, allowing you to manipulate the frame as an object (for example, to stretch it, move it, and fill it).To select an anchored frame as an object, click on its border with the left mouse button or draw a selection border around it (see *Selecting Objects and Frames* in Chapter 2). When drawing a selection border around an object, press and hold down the Shift key before pressing the left mouse button.

  When you select an anchored frame as an object, it behaves much like an unanchored frame. You can stretch it, scale it, and change its fill, pen, and pen width. If you cut or copy the frame, an *unanchored* frame is placed on the Clipboard.

  To select an anchored frame as a *character*, select its anchor symbol using the middle mouse button or one of the keyboard highlighting commands (see Appendix A). Or choose the Search command to search for `\a`.

  When you select an anchored frame as a symbol and you cut or copy it, an *anchored* frame is placed on the Clipboard. If you later select an insertion point and choose the Paste command from the Edit menu, a copy of the anchored frame is inserted. If you want to make many anchored frames with the same properties, make one anchored frame with the properties you want, select the anchor point and frame using the middle mouse button, copy it, and then paste it into new locations as needed.
- In-margin anchored frames may cause the screen display to be incorrect if they intersect other objects or if their position moves due to editing around the anchor point. They may also be displayed incorrectly due to typing around the anchor point. To correct the screen display, choose the Redisplay command from the Window menu.
- To delete an anchored frame, select it by highlighting its anchor symbol or by clicking on its border with the left mouse button. Then press the Delete key or choose the Cut command from the Edit menu.

Auto Connect

Location: TextRects menu

Standard Keyboard Equivalent: `Esc t a`

Purpose: Use the Auto Connect command to turn the auto-connect property of a text flow on and off. The auto-connect property controls what happens when the final TextRect in a text flow overflows.

When you turn on the auto-connect feature and your editing causes the text flow to overflow, a new page is automatically generated, and the TextRects on that new page are automatically connected to the overflowing text flow. The new page is added after the page containing the overflow TextRect. The layout of TextRects on the page matches the layout on the master page. You can change this layout using the Number of Columns command.

When you turn off the auto-connect feature and your editing causes the text flow to overflow, the extra text becomes hidden below the bottom of the last TextRect in the flow. When you display borders, a TextRect containing overflowed text has a solid border across the bottom edge. See the Borders command.

Use: Select any TextRect in the text flow and choose the Auto Connect command to toggle the auto-connect feature on and off. If the auto-connect feature is on for the text flow, a mark (for example, a checkmark or an *x*) appears next to the Auto Connect command in the TextRects menu.

Notes:

- When you turn on the auto-connect feature in a text flow that has overflowed, FrameMaker creates as many new pages as needed to accommodate the extra text.

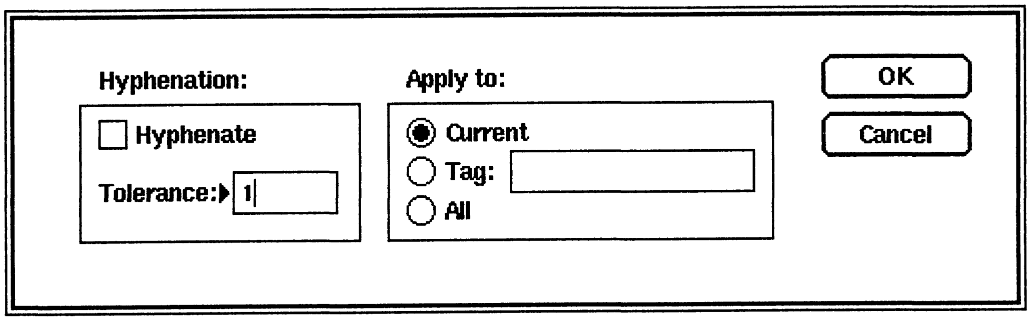
Auto Hyphenation

Location: Format menu

Standard Keyboard Equivalent: `Esc f a`

Purpose: Use the Auto Hyphenation command to change the hyphenation settings for one or more paragraphs without changing other paragraph properties. Hyphenation settings include whether or not the paragraph is hyphenated, and if so, how many lines in a row can be hyphenated. You can also change hyphenation settings using the Paragraphs command.

Use: Choose the Auto Hyphenation command to display the Auto Hyphenation dialog box:



Tolerance: 1

[✓ Current] Tag: All

Hyphenation: In this area, specify the hyphenation settings.

- **Hyphenate:** Turn on this check box to automatically hyphenate the indicated paragraph(s). A check means automatic hyphenation is on; no check means it is off.
- **Tolerance:** In this box, specify how many adjacent lines in a paragraph can be hyphenated. The number 1 means that each hyphenated line in a paragraph cannot be immediately followed by another hyphenated line. The number 2 means that two lines can be hyphenated in a row, but the next line after them cannot be hyphenated. To get unlimited hyphenation, type a large number (for example, 99).

Apply To: Click a setting in this area to specify which paragraphs are affected by the new settings.

- **Current:** Click Current to apply the new settings to the paragraph containing the insertion point and all selected paragraphs (if any).
- **Tag:** Click Tag to apply the new settings to all paragraphs with the indicated paragraph tag.
- **All:** Click All to apply the settings to all paragraphs in the document.

Notes:

- This version of FrameMaker uses algorithmic hyphenation. The algorithm is fairly conservative. In some cases it avoids hyphenation rather than making errors.
- *Discretionary hyphens* are characters that mark acceptable hyphenation points within words. If a word does not need to be hyphenated (because it is not at the end of a line), the discretionary hyphen is not visible (when you turn on text symbols, you can see discretionary hyphens on the screen, but they take up no space and do not appear when the document is printed). If a word does need to be hyphenated, FrameMaker splits the word based on the discretionary hyphens it contains.

  To insert a discretionary hyphen, put the insertion point where you want the discretionary hyphen and press Alt-hyphen (press and hold down the Alt key and then press the hyphen key).
- To prevent a word from being hyphenated, put a *suppress hyphenation symbol* in the word: Put the insertion point in the word and press Ctrl-hyphen (press and hold down the Ctrl key and then press the Hyphen key).
- FrameMaker supports two characters that look like a hyphen: the en dash (–) and the em dash (—). To insert an en dash, press Ctrl-q Shift-p. To insert an em dash, press Ctrl-q Shift-q.

Back

Location: Tools window

Standard Keyboard Equivalent: `Esc o b`

Purpose: Use the Back command to change the drawing order of a page so that all selected objects or frames are drawn first, before all other objects on the page are drawn. The drawing order controls the appearance of overlapping objects. It also affects the selection of objects with the mouse. If you click with the arrow pointer on overlapping objects, FrameMaker selects the frontmost object. If you are trying to select one object and another object keeps getting in the way, send the interfering object to the back so that you can select the desired object.

Use: Select one or more objects or frames and click Back in the Tools window.

Notes:

- If you use the Back command on a group, all subobjects of the group are sent to the back, but the drawing order within the group is maintained.
- See also the Front command and *Changing Object and Frame Overlap* in Chapter 2.

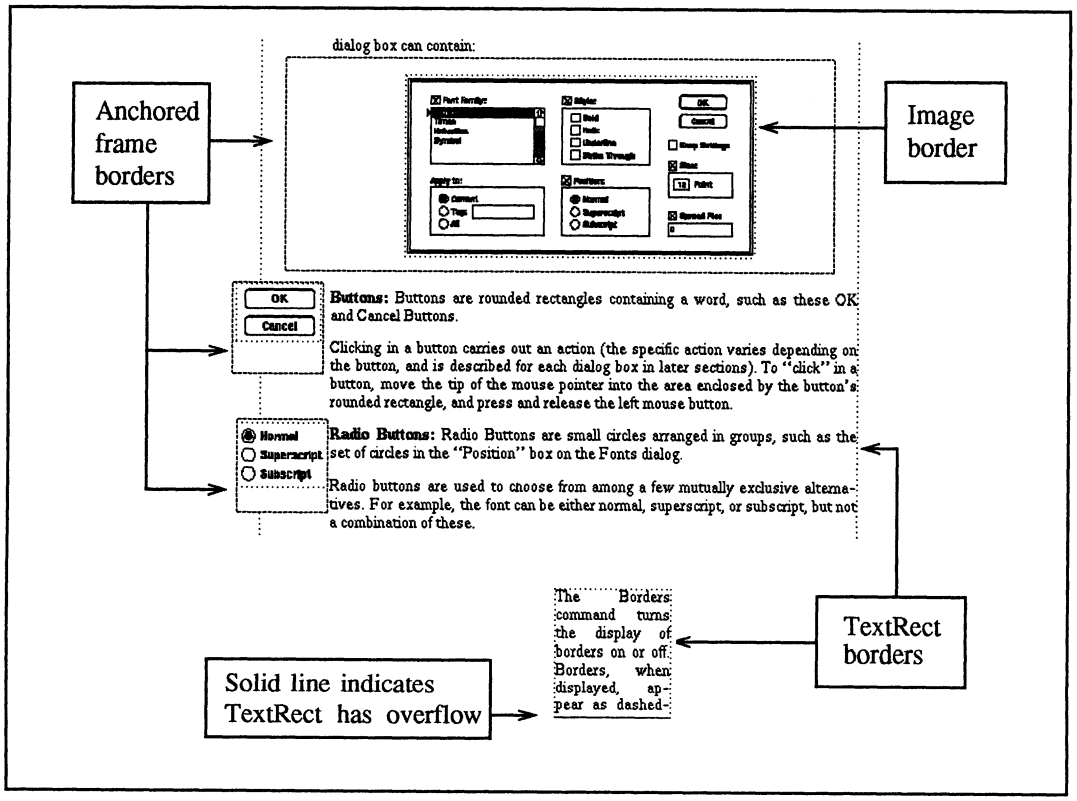
Borders

Location: Guides menu

Standard Keyboard Equivalent: `Esc g b`

Purpose: Use the Borders command to turn the display of borders on and off. When you display borders, they appear as dashed- or dotted-line rectangles around TextRects, frames, imported images, and pages. Display the borders to see how a page is composed and to help you position the mouse when selecting text or scrolling pages.

Use: Choose the Borders command to turn borders on and off. When borders are on, a mark (a checkmark or an *x*) appears next to the Borders command in the Guides menu.



dialog box can contain:

Buttons: Buttons are rounded rectangles containing a word, such as these OK and Cancel Buttons.

Clicking in a button carries out an action (the specific action varies depending on the button, and is described for each dialog box in later sections). To "click" in a button, move the tip of the mouse pointer into the area enclosed by the button's rounded rectangle, and press and release the left mouse button.

Radio Buttons: Radio Buttons are small circles arranged in groups, such as the set of circles in the "Position" box on the Fonts dialog.

Radio buttons are used to choose from among a few mutually exclusive alterna-tives. For example, the font can be either normal, superscript, or subscript, but not a combination of these.

The Borders command turns the display of borders on or off. Borders, when displayed, ap-pear as dashed-

Sample Borders in a Document Window

Notes:

- Borders are displayed on the outside edges of TextRects and imported images. When you draw objects with the snap feature turned on, the borders do not coincide exactly with the ruler division, even though the objects themselves do.

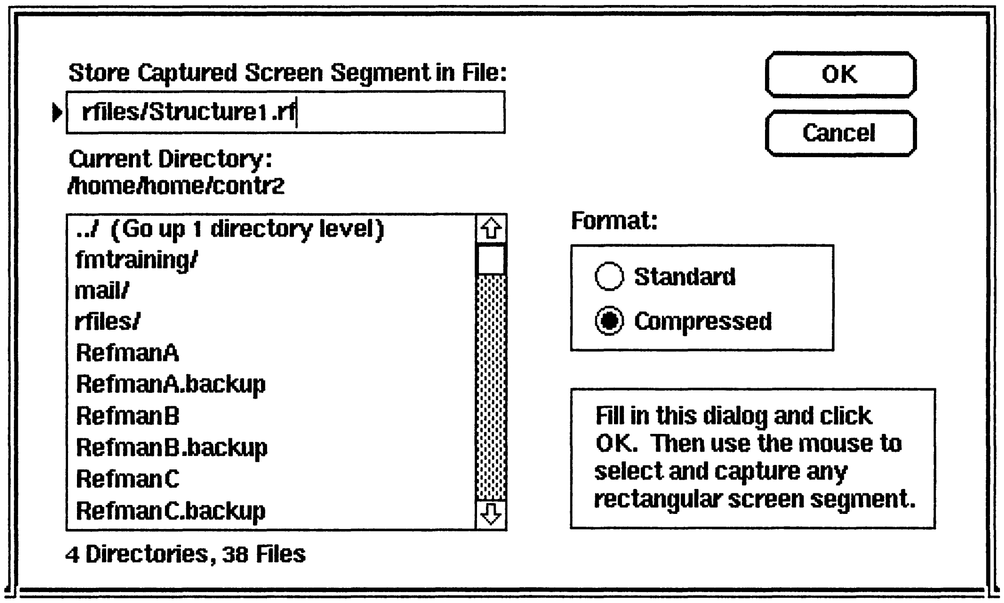
Capture

Location: Document menu

Standard Keyboard Equivalent: `Esc d c`

Purpose: Use the Capture command to store any portion of the workstation screen in a bitmap file (in xwd XYPixmap format). Once you've stored a portion of the screen, you can import it into a FrameMaker document using the Import command.

Use: Choose the Capture command to display the Capture dialog box:



Store Captured Screen Segment in File:
rfiles/Structure1.rf

Current Directory:
/home/home/contr2

../ (Go up 1 directory level)
fmtraining/
mail/
rfiles/
RefmanA
RefmanA.backup
RefmanB
RefmanB.backup
RefmanC
RefmanC.backup

4 Directories, 38 Files

Format: Standard [✓ Compressed]

Fill in this dialog and click OK. Then use the mouse to select and capture any rectangular screen segment.

Store Captured Screen Segment in File: Specifies the name of the image file used to store the captured portion of the screen.

Format: Because xwd bitmap files do not support the compressed format, FrameMaker always creates standard format files regardless of which format setting you choose.

Fill in the desired filename and click OK to display the capture pointer (✧).

To capture a rectangular area of the screen, move the capture pointer to one of its corners, press and hold down the left mouse button, and then drag the capture pointer to the opposite corner. A rectangular *capture border* indicates the area of the screen to be captured. When the correct area is enclosed, release the mouse button; the selected screen portion is written to the requested file.

Notes:

- Use the Import command to import the bitmap file into a FrameMaker document.
- FrameMaker imports only monochrome images. If you want to import an image captured on a color workstation, convert it to a monochrome image before importing it. See the Import command for information on converting color images.
- The screen pixels *under* the capture border are included in the bitmap file.
- You can capture a FrameMaker dialog box even though the Document menu is not available while the dialog box is displayed. To do so, display the dialog box and then press Alt-c. The arrow pointer changes into the capture pointer, and you can select an image to capture as described on the previous page.

  FrameMaker stores the captured dialog box in a file named `Dialog`*xx*`.rf`, where *xx* is a number. To find the exact name of the file, click Cancel in the dialog box that you captured and use the Import command to display the Import dialog box. The filename of the captured file appears in the Import dialog box. The file is put in the current directory named in the Capture dialog box. If you have not used the Capture dialog box since beginning FrameMaker, the file is put in the directory of the document from which you chose the Capture command.

Catalog

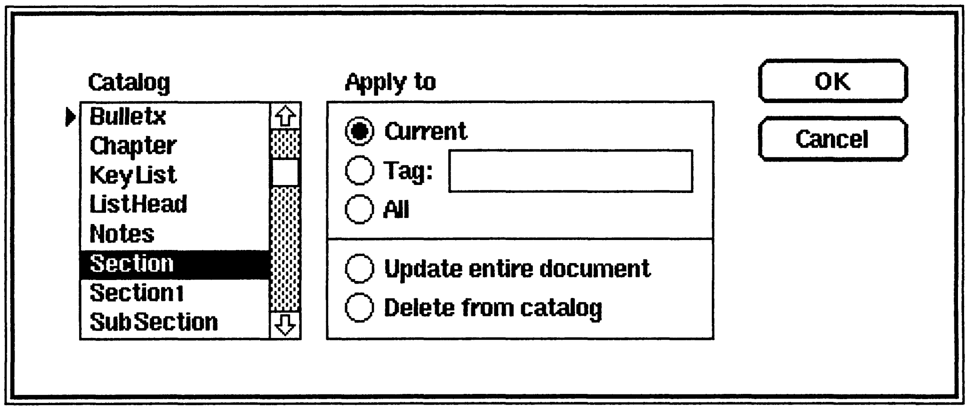
Location: Format menu and Maker menu

Standard Keyboard Equivalents: `Esc f c` and `Esc C`

Purpose: Use the Catalog command to change the paragraph format of one or more paragraphs using paragraph formats you have stored in the Catalog. (You create and store paragraph formats using the Paragraphs command.) Also use the Catalog command to delete paragraph formats from the Catalog.

See *Tagged Paragraphs and the Catalog* in Chapter 2 for a more complete description of setting up and using the Catalog.

Use: Choose the Catalog command to display the Catalog dialog box:



Catalog: In this area, select a paragraph format to be applied.

Apply To: Click one of the settings in this area to specify what to do with the selected paragraph format.

- **Current:** Click Current to apply the selected paragraph format to the paragraph containing the insertion point and to all paragraphs containing selected text.
- **Tag:** Click Tag to apply the selected paragraph format to all paragraphs with the indicated paragraph tag.
- **All:** Click All to apply the selected paragraph format to all paragraphs in the document.
- **Update Entire Document:** Click this setting to apply all paragraph formats in the Catalog to all corresponding paragraphs in the document. A document paragraph corresponds to a paragraph format if the paragraph's tag matches the format's name. Use this setting after importing a Catalog from another document (see the Import command) and to ensure that a document has consistent formatting across all of its tagged paragraphs. A dialog box appears to make sure this is the desired action.

- **Delete from Catalog:** Click this setting to delete the selected paragraph format from the Catalog. A dialog box appears to make sure this is the desired action.

Notes:

- When you store a paragraph format in the Catalog, you are storing tab settings, font settings, and a paragraph name (tag) that is the same as the paragraph format name.
- When you apply a paragraph format from the Catalog to a paragraph in the document, that paragraph's tag is changed to match the paragraph format name.
- When you change a tagged paragraph's format using the Paragraphs command, the associated Catalog entry (if one exists) is not updated unless you turn on the Apply to Catalog check box in the Paragraphs dialog box.
- You can "export" a Paragraph Catalog to a MIF file. For more information, see the Save and Import commands.

Connect TextRects

Location: TextRects menu

Standard Keyboard Equivalent: `Esc t c`

Purpose: Use the Connect TextRects command to connect two unconnected TextRects. When you connect two TextRects, the text they contain flows between them as you edit it.

Using the Connect TextRects command, you can:

- Link two isolated TextRects.
- Put an isolated TextRect at the head or tail of a text flow.
- Insert an isolated TextRect into the middle of a text flow.
- Remove a TextRect from a flow.
- Connect the tail of one text flow to the head of another text flow.

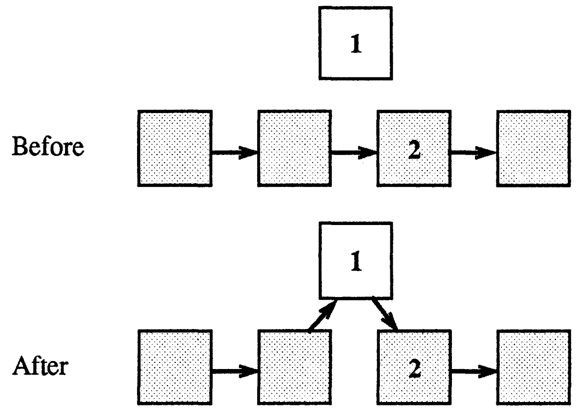
Turn the auto-connect feature on when you want text to flow automatically from one page to the next (see the Auto Connect command). Use the Connect TextRects command, however, to perform complex tasks such as a laying out a magazine or flowing text around a complex graphic.

Use: To connect two TextRects:

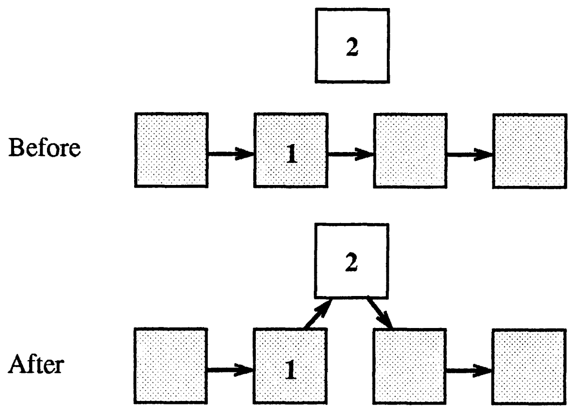
- Select the first TextRect.
- Select the second TextRect. This TextRect need not be on the same page as the first TextRect.
- Choose the Connect TextRects command. Overflow text from the first TextRect flows into the second TextRect.

Examples:

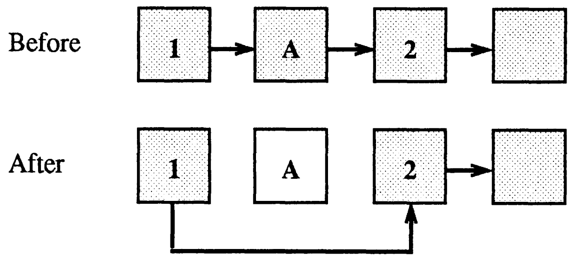
- To insert a TextRect in the middle of an existing flow, select the isolated TextRect (1) first, then select the TextRect to which it should flow (2), and choose the Connect TextRects command. This diagram shows what happens:



- To insert a TextRect into a flow, you can also select the TextRect from the flow first (1), then select the isolated TextRect to which it should flow (2), and choose the Connect TextRects command:



- To remove a TextRect from the middle of a flow (leaving the TextRect empty), select the TextRect before the one to be removed from the flow (1), select the TextRect following the one to be removed (2), and choose the Connect TextRects command:



The text that began in TextRect A now begins in TextRect 2.

Notes:

- You can connect only two TextRects in one step, but you can build TextRect chains in successive steps.
- It is often easier to create a desired page layout using the Split TextRect command than by drawing individual TextRects and then connecting them. See the Split TextRect command for more information.
- See the Disconnect Head/Tail commands and the Cut command for information about removing a TextRect from a flow and from the page at the same time.

Copy

Location: Edit menu and Maker menu

Standard Keyboard Equivalent: `Esc e c`

Purpose: Use the Copy command to copy whatever is currently selected and to store that copy on the FrameMaker Clipboard (an off-screen storage area for objects, frames, text, font settings, and paragraph formats). There is one central Clipboard shared by all FrameMaker document windows.

Once you have copied something to the Clipboard, you can put copies of it into any FrameMaker document using the Paste command.

Text stored on the FrameMaker Clipboard retains all of its font information. If the text includes paragraph symbols, the associated paragraph formats are also stored on the Clipboard. If the text contains anchored frames, they too are stored on the Clipboard.

Important: The Copy command removes the previous contents of the Clipboard.

Use: Select objects, frames, or text using the techniques presented in Chapter 2. Then choose the Copy command.

Notes:

- See also the Cut and Paste commands.
- When you select text in a FrameMaker window, a copy of it is automatically stored in a special X Window text buffer (a holding place in the workstation's memory). You can copy text from the text buffer into any window in the X Window system.

  To copy text from a FrameMaker document to an xterm window or another X Window application:

  1. In the FrameMaker window, select the text you want to copy (for information on selecting text in a document window, see Chapter 2).

     The selected text is saved in the buffer.
  2. Point on the window in which you want to paste the text.
  3. Press and hold down Shift and then click the right mouse button.

     In an xterm window, the text appears at the bottom of the window, after the last prompt.

Copy Font

Location: Maker menu

Standard Keyboard Equivalent: `Esc e f`

Purpose: Use the Copy Font command to copy font settings from whatever text is selected and to store them on the FrameMaker Clipboard (an off-screen storage area for objects, frames, text, font settings, and paragraph formats). Once you have copied font settings to the Clipboard, you can apply them to other blocks of characters using the Paste command.

Important: The Copy Font command removes the previous contents of the Clipboard.

Use: Select a block of text using the techniques presented in Chapter 2. Then choose the Copy Font command. To apply the copied font settings to another block of text, select the text you want to change and choose the Paste command from the Edit menu (or press Ctrl-y).

Notes:

- If you select text that contains a mixture of font settings, only the font information common to all the settings is copied to the Clipboard. For example, if you select a block containing Times 12 plain, Times 12 bold, and Times 12 italic, only Font Family = Times and Point Size = 12 are stored on the Clipboard. If you then select a block and choose the Paste command, only the Family and Point Size of the block are affected. If the block contains a mixture of plain, bold, and other styles, those styles are not affected.
- See also the Font, Copy, Cut, and Paste commands.

Copy Pgf Format

Location: Maker menu

Standard Keyboard Equivalent: `Esc e g`

Purpose: Use the Copy Pgf Format command to copy the paragraph format settings of the current paragraph and to store them in the FrameMaker Clipboard (an off-screen storage area for objects, frames, text, font settings, and paragraph formats). The paragraph format settings include all values you see in the Paragraphs dialog box, plus the tab settings in the document window's top ruler. Once you have copied paragraph format settings to the Clipboard, you can apply them to other paragraphs using the Paste command.

Important: The Copy Pgf Format command removes the previous contents of the Clipboard.

Use: Put the insertion point in a paragraph using techniques described in Chapter 2. Then choose the Copy Pgf Format command. To copy the format settings to another paragraph, put the insertion point in a different paragraph or select a block of paragraphs and choose the Paste command. The paragraph containing the insertion point and any selected paragraphs lose their old format settings and take on the pasted settings.

Notes:

- See also the Paragraphs, Copy, Cut, and Paste commands.

Cut

Location: Edit menu and Maker menu

Standard Keyboard Equivalent: `Esc e x` and `Delete`

Purpose: Use the Cut command to remove the current selection and to place it on the FrameMaker Clipboard (an off-screen storage area for objects, frames, text, font settings, and paragraph formats). You can cut text, objects, and frames from a document, and you can also paste the cut material from the Clipboard into a document using the Paste command.

Text stored on the FrameMaker Clipboard retains all of its font information. If you select text that includes paragraph symbols, the associated paragraph formats are also stored on the Clipboard. If you select text that contains anchored frames, they too are stored on the Clipboard.

Important: The Cut command and the three Copy commands remove the previous contents of the Clipboard.

Use: Select objects, frames, or text using the techniques presented in Chapter 2. Then choose the Cut command.

Notes:

- Cutting TextRects: If you cut a TextRect that is not part of a flow, then both the TextRect and the text it contains are removed from the document and stored on the Clipboard. If the TextRect is part of a flow, however, the TextRect object is removed and stored on the Clipboard, but the text it contained remains in the document. The entire text flow is reformatted to accommodate the displaced text within the remaining TextRects.
- See also the Copy and Paste commands.

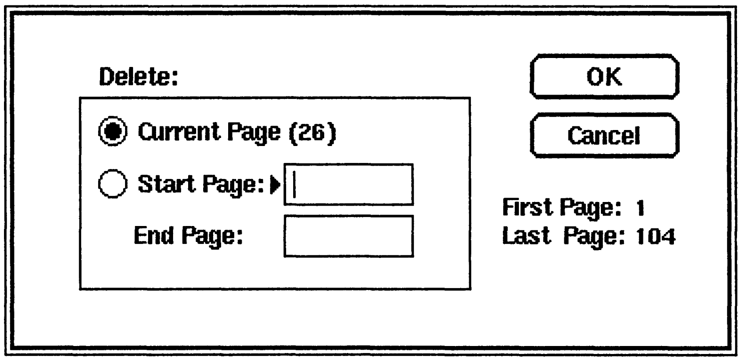
Delete Page

Location: Page menu

Standard Keyboard Equivalent: `Esc p d`

Purpose: Use the Delete Page command to delete a single page or a range of pages from the document.

Use: Choose the Delete Page command to display the Delete Page dialog box:



Delete: Click one of these settings to specify which page(s) to delete.

- **Current Page:** Click Current Page to delete the current page (the page number is indicated next to Current Page and at the bottom of the document window).
- **Start Page** and **End Page:** In these boxes, fill in the range of pages to be deleted.

  If the first page of the document is not page 1 (in other words, if you have changed the First Page # in the Headers & Footers dialog box), then the numbers in these boxes must be the actual page numbers, not the relative page numbers within the document. For example, if `Chapter2.doc` begins on page 21, and you want to delete the first five page of that document, then type `21` in the Start Page edit box and `25` in the End Page edit box.

Important: If the deleted page contains TextRects, both the TextRects and the text they contain are deleted.

Notes:

- If the TextRects on the deleted pages are part of a text flow (a chain of linked TextRects), the Delete Page command repairs the chain by linking the pages before and after the deleted ones.
- You can delete any page except the master page.
- Since you can't undo the Delete Page command, you may want to use the Save command to make a backup copy of your document before you delete pages, in case you later need to restore the deleted pages.

Disconnect Head/Tail

Location: TextRect menu

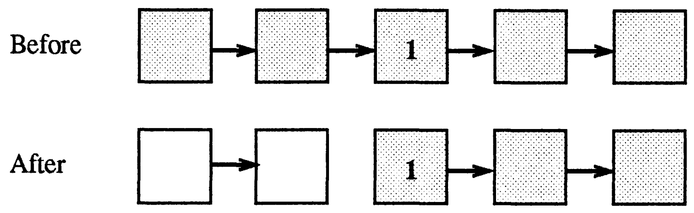
Standard Keyboard Equivalents:

Disconnect Head:	`Esc t h`
Disconnect Tail:	`Esc t t`

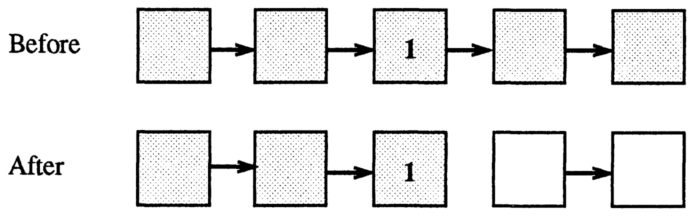
Purpose: Use the Disconnect Head and Disconnect Tail commands to split a text flow into two unconnected flows.

Use: Select one TextRect in the flow and choose one of the Disconnect commands.

- Disconnect Head breaks the flow between the selected TextRect (1) and its predecessor:



- Disconnect Tail breaks the flow between the selected TextRect (1) and the next TextRect in the flow:



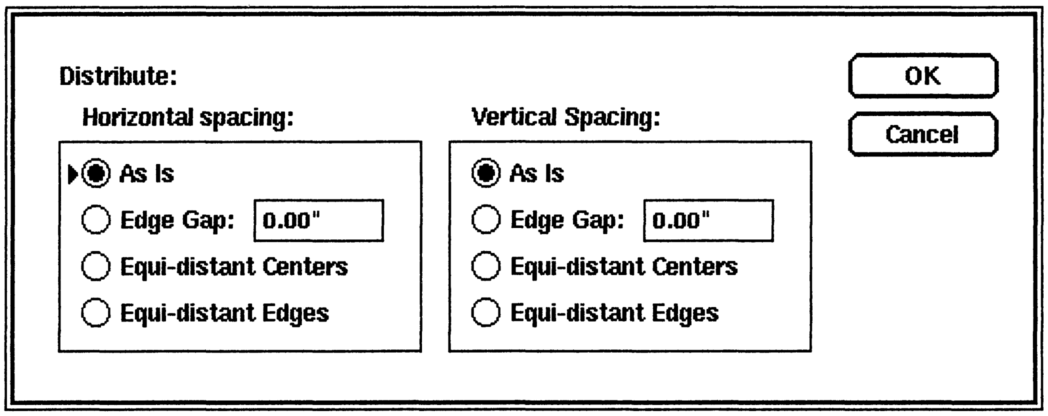
Distribute

Location: Tools window.

Standard Keyboard Equivalent: `Esc o d`

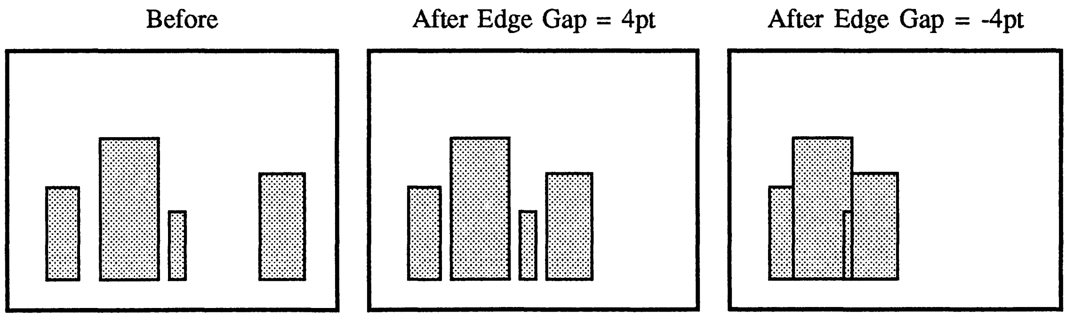
Purpose: Use the Distribute command to move objects horizontally and/or vertically so that they are evenly spaced. (Since the Distribute command uses 1-point resolution, the equidistant settings may actually produce spacing that varies by 1/72" from object to object.)

Use: Select objects using the mouse (see Chapter 2) and then click Distribute in the Tools window to display the Distribute dialog box:


Distribute:
Horizontal spacing:
[✓ As Is] ○ Edge Gap: 0.00" ○ Equi-distant Centers ○ Equi-distant Edges
Vertical Spacing:
[✓ As Is] ○ Edge Gap: 0.00" ○ Equi-distant Centers ○ Equi-distant Edges

Horizontal Spacing: Click one of these settings to specify how the objects should be spaced horizontally.

- **As Is:** Click As Is to not move the objects horizontally. Use this setting to adjust the vertical spacing without affecting the horizontal spacing.
- **Edge Gap:** Click Edge Gap to move all objects (except the leftmost one) to the left or right so that the gap between adjacent objects equals the indicated value. An edge gap of 0 causes the objects to touch. Negative numbers cause the objects to overlap; positive numbers move the objects apart:



Feathering

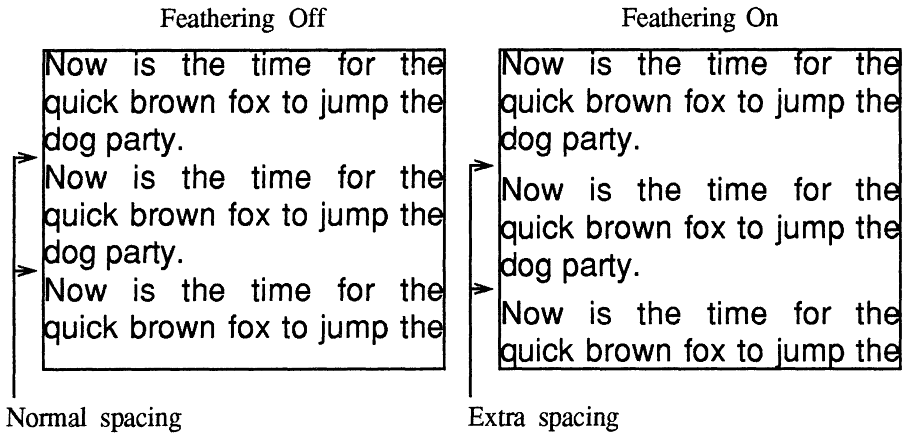
Location: TextRects menu

Standard Keyboard Equivalent: `Esc t f`

Purpose: Use the Feathering command to turn vertical justification on and off within a column.

In a TextRect that has feathering turned on, extra white space is added as needed between each paragraph so that the bottom text line lies against the bottom edge of the TextRect.

When you turn feathering off, no extra space is added between lines in a TextRect, and the final line might not lie against the bottom edge of the TextRect. The difference between feathered and unfeathered TextRects is shown below:

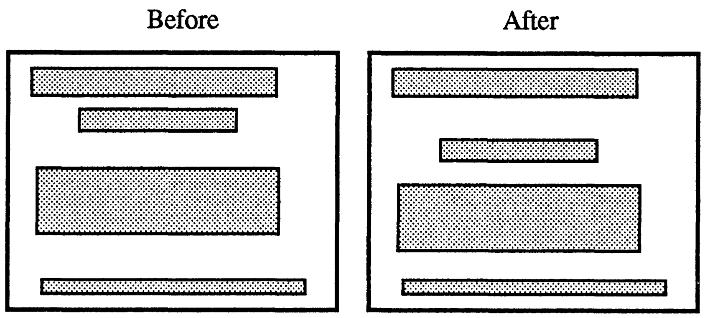

Feathering Off
Feathering On
Now is the time for the quick brown fox to jump the dog party.
Now is the time for the quick brown fox to jump the dog party.
Now is the time for the quick brown fox to jump the
Now is the time for the quick brown fox to jump the dog party.
Now is the time for the quick brown fox to jump the dog party.
Now is the time for the quick brown fox to jump the
Normal spacing
Extra spacing

Use: Select the TextRect you want to feather by putting the insertion point in the text it contains. Then choose the Feathering command to toggle feathering on and off. When you select a TextRect that has feathering turned on, a mark (for example, a checkmark or an *x)* appears next to the Feathering command in the TextRects menu.

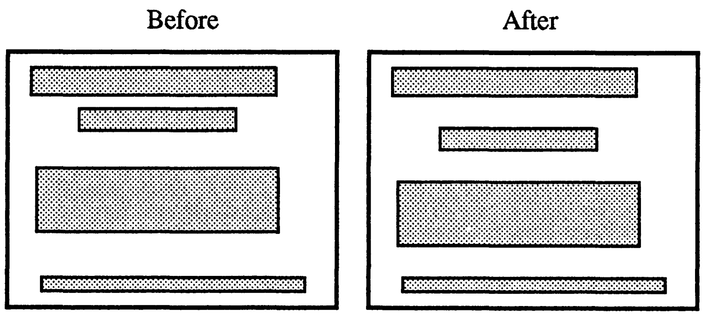
Notes:

- FrameMaker does not feather the lines in a TextRect if the extra space needed between each paragraph exceeds 6 points.
- If there are no paragraph breaks within a feathered TextRect, extra space is added between each line of text.
- If the last line seems to sit too high in a feathered TextRect, make sure that all characters (including the nonprinting ones) have the correct font settings. FrameMaker leaves space for the largest possible descender in the largest font used in the line, even though no character with such a large descender actually occurs on the line (a descender is the portion of a character that extends below the baseline, such as the hook on the letter *j*).

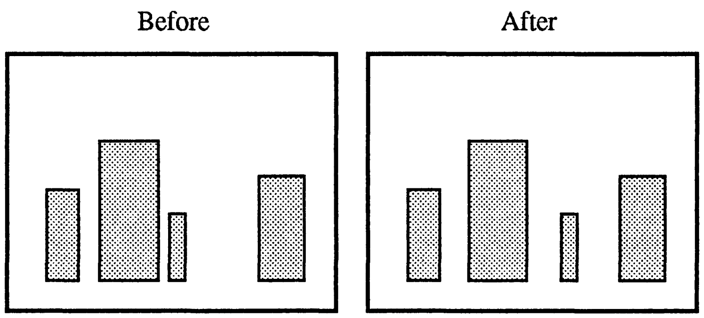
- **Equidistant Centers:** Click this setting to leave the top and bottom objects where they are and to move the other objects up or down so that the distance between all top/bottom centers is equal:



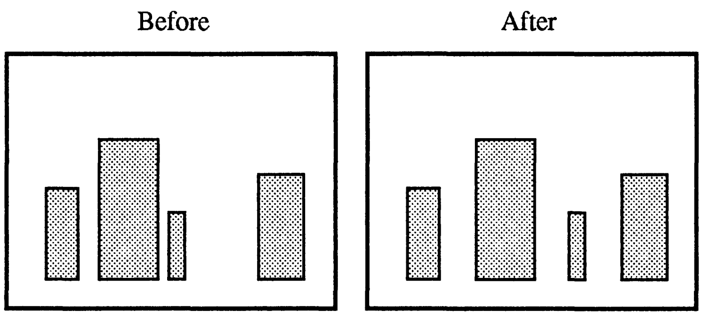
- **Equidistant Edges:** Click this setting to leave the top and bottom objects where they are and to move the other objects up or down so that the gap between all object edges is equal:



- **Equidistant Centers:** Click this setting to leave the left- and rightmost objects where they are and to move the other objects to the left or right so that the distance between all left/right centers is equal:

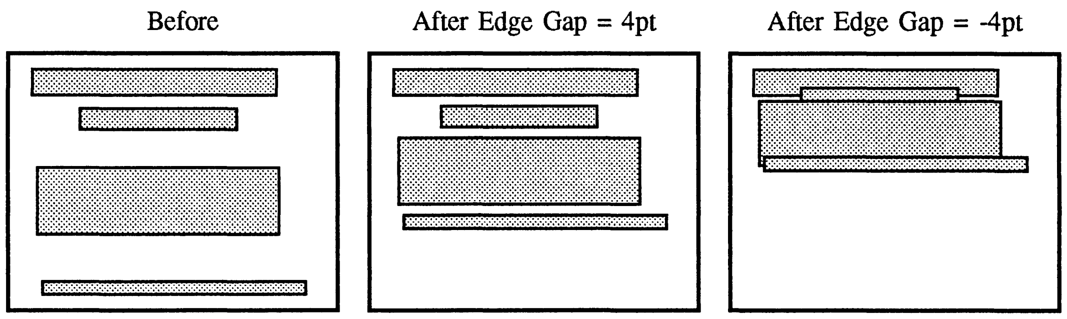


- **Equidistant Edges:** Click this setting to leave the left- and rightmost objects where they are and to move the other objects left or right so that the gap between all object edges is equal:



Vertical Spacing: Click one of these settings to specify how the objects should be moved up or down.

- **As Is:** Click As Is to not move the objects vertically. Use this setting to adjust the horizontal spacing without affecting the vertical spacing.
- **Edge Gap:** Click Edge Gap to move all objects (except the topmost one) up or down so that the gap between adjacent objects equals the indicated value. An edge gap of 0 causes the objects to touch. Negative numbers cause the objects to overlap; positive numbers move the objects apart:



Fonts

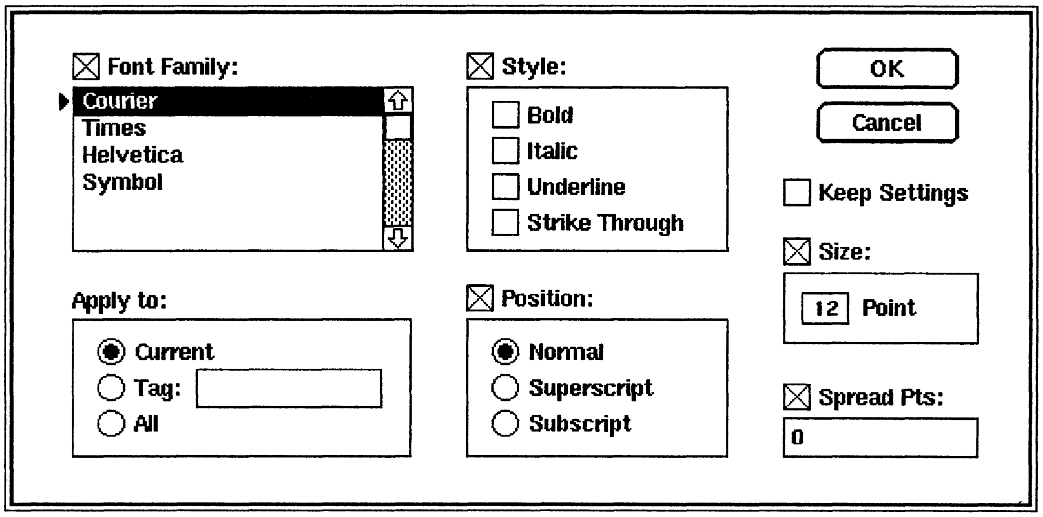
Location: Format menu and Maker menu

Standard Keyboard Equivalents: `Esc f f` and `Esc F`

Purpose: Use the Fonts command to change the font family, size, style, spacing, and position of text. Also use it to set the default font, which is used when TextRects and TextLines are added with the TextRect and TextLine tools (see Chapter 4).

Use: Select the text using the middle mouse button and choose the Fonts command or use a keyboard font command. For more information, see Appendix A, *Keyboard Commands*. To switch fonts while typing in new text, choose the Fonts command after setting the insertion point and before typing the new text.

Choose the Fonts command to display the Fonts dialog box:



☒ Font Family: Courier (selected), Times, Helvetica, Symbol

☒ Style: ☐ Bold ☐ Italic ☐ Underline ☐ Strike Through

☐ Keep Settings

☒ Size: 12 Point

Apply to: ● Current ○ Tag: ○ All

☒ Position: ● Normal ○ Superscript ○ Subscript

☒ Spread Pts: 0

Keep Settings: Turn on Keep Settings to specify that the dialog box settings are to remain the same across each use of the Fonts command until you explicitly change them or until you turn off the Keep Settings check box. Use Keep Settings when you want to apply the same font settings to several different blocks of text (see also the Copy Font command). When you choose the Fonts command and this setting is turned off, the Fonts dialog box appears showing the font settings of the selected text.

Font Family: In this area, select a font family (for example, Times or Helvetica). You can add font families to those that appear in the scroll list. See Appendix D, *Customizing FrameMaker,* for information on adding fonts.

The check box to the left of Font Family controls whether or not this Fonts dialog box affects the font family when it is applied. A check means "set text to the indicated font family"; no check means "do not change the text's font family."

When the Fonts dialog box appears, if there is text selected that contains a mixture of font families, the family of the first selected character is indicated in the scroll list, and the check box to the left of Font Family is turned off.

Style: Turn on one or more settings to specify font style (for example, Plain, **Bold**, *Italic*, Underline, and ~~Strike Through~~). You can produce such combinations as ***Bold Italic*** and *Italic Underline*. If you don't turn on a setting in the Style area, the result is plain text.

The check box to the left of Style controls whether or not the Fonts dialog box affects font style when it is applied. A check means "set text to the indicated style"; no check means "do not change the text's style."

When the Fonts dialog box appears, if there is text selected that contains a mixture of font styles, the style of the first selected character is indicated, and the check box to the left of Style is turned off.

Position: Click one of these settings to specify the position of the text relative to the baseline (Normal, Superscript, or Subscript).

The check box to the left of Position controls whether or not the Fonts dialog box affects font position when it is applied. A check means "set text to the indicated position"; no check means "do not change the text's position."

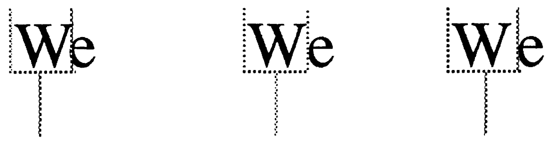
When the Fonts dialog box appears, if there is text selected that contains a mixture of positions, the position of the first selected character is indicated, and the check box to the left of Position is turned off.

Size: In this area, specify font size in points. (For example, standard font sizes for PostScript printers are 7, 8, 9, 10, 12, 14, 18, and 24—although these sizes aren't available for all font families. Other sizes may be available at your site.) Scroll through the available sizes by clicking repeatedly in the Point box.

The check box to the left of Size controls whether or not the Fonts dialog box affects font size when it is applied. A check means "set text to the indicated size"; no check means "do not change the text's size."

When the Fonts dialog box appears, if there is text selected that contains a mixture of sizes, the size of the first selected character is displayed in the scroll list, and the check box to the left of Size is turned off.

Spread Pts: In this box, type the amount of positive or negative space to be added to each character's normal *advance width* (a character's advance width is the distance from its left side to the left side of the following character). Positive values spread the text farther apart; negative values squeeze text together producing an effect known as *kerning*. A Spread Pts value of 0 causes standard character spacing. (Note that all selected characters are affected.) To kern two characters together, select the left one and set its Spread Pts value to a negative number:



Spread points: **-2** **0** **2**

Notes:

- Characters you type in from the keyboard (or import with the Import command) acquire the font characteristics of the insertion point. The insertion point font is reset whenever you move the insertion point using the mouse or keyboard commands, but not when you type normal characters or press the Delete key. The insertion point font is reset based on the font of the nearest visible character on the line, if any. If the insertion point is surrounded by two visible characters or is on an empty line, the font of the character to its right is used, even if that is a nonprinting character such as an end-of-paragraph symbol or an end-of-flow symbol.
- You can change the font characteristics of nonprinting characters just as you can other characters. When you turn on text symbol display, you can see the font characteristics of these nonprinting symbols (see the Text Symbols command for more information).
- Not all combinations of font family, style, and size are supported by FrameMaker. For example, the Symbol family has no bold or italic variations.
- See also the Copy Font command.
- See *Font Commands* in Appendix A for a description of keyboard shortcuts for changing fonts.

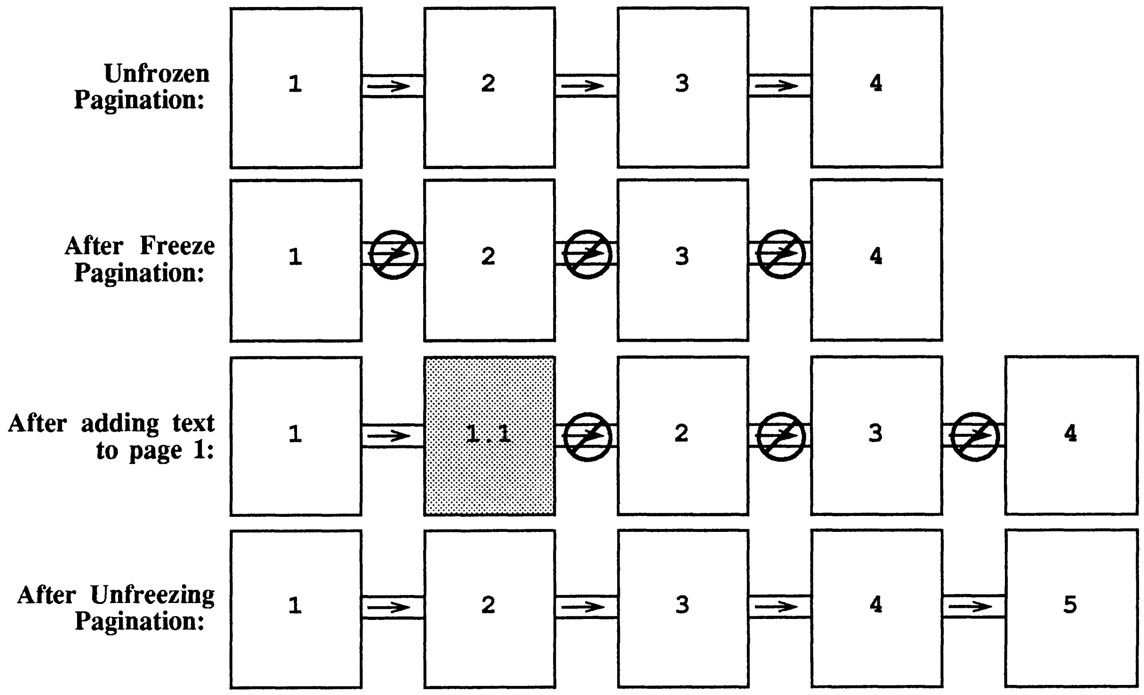
Freeze Pagination

Location: Format menu

Standard Keyboard Equivalent: `Esc f z`

Purpose: Use the Freeze Pagination command to turn *frozen pagination* on and off. When pagination is frozen, edited text (and the resulting changes in paragraph numbering) does not flow from page to page as it usually does. Instead, if you delete text, the text following the deletion stays where it is instead of linking to the text before the deletion; and if you add text while freeze pagination is on, FrameMaker appends *point pages* to *main pages*. Main pages are the ones that exist before you turn freeze pagination on; point pages are the pages that follow a main page when you add text to a main page and it overflows. (They're called point pages because their page number is the preceding main page number, followed by a decimal point and a value to the right of the decimal point—5.1, 5.2, 5.3, and so forth.)

When you freeze pagination, text flows from a main page to its point pages, but text cannot flow from a point page to a main page after it:



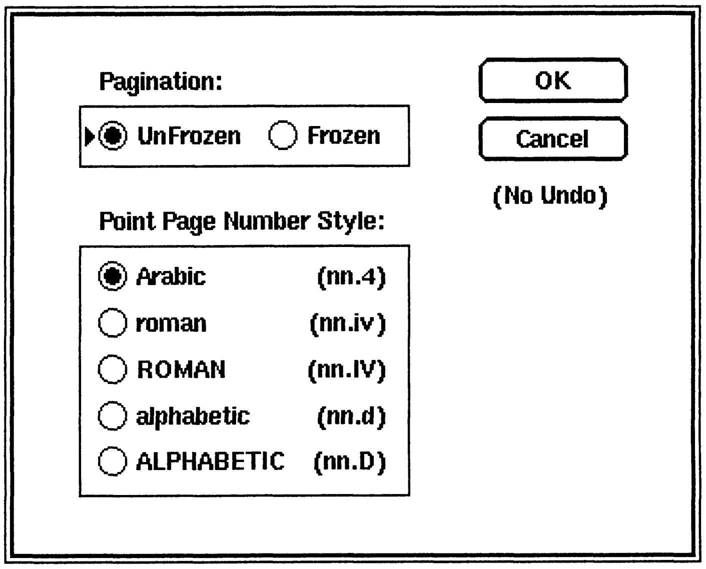
In the above diagram, notice that when you add text to page 1 in a frozen document, FrameMaker adds a point page and numbers it 1.1. Notice also that after adding text to a frozen document, you can turn off freeze pagination, and the document is restored to normal pagination with all text and formatting intact.

This feature serves at least two purposes. First, it is useful when you have published and distributed a document and need to revise it. Some users prefer to leave the original document's page numbering intact and indicate added pages with a special numbering scheme; that way, they can save money by distributing only the revised pages.

Second, editing with frozen pagination often speeds up FrameMaker's performance in very large documents. Since FrameMaker keeps your document formatted while you work on it, it may take a long time to reformat a large document while you're typing or editing. To minimize the formatting time, you can freeze the document's pagination while you are editing, and then turn off freeze pagination when you are done. FrameMaker restores normal pagination as if it had never been frozen. You can also repaginate a range of pages while pagination is frozen using the Repaginate command.

Important: Freeze pagination works only on documents with a single text flow running from page to page; it does not work on documents with custom-designed pages or multiple text flows.

Use: To freeze pagination (or to turn it off if it's frozen), choose the Freeze Pagination command from the Format menu. When freeze pagination is on, a mark (for example, a checkmark or an *x*) appears next to the Freeze Pagination command in the Format menu. The Freeze Pagination dialog box appears:


Pagination: [✓ UnFrozen] Frozen

Point Page Number Style:

[✓ Arabic (nn.4)]
roman (nn.iv)
ROMAN (nn.IV)
alphabetic (nn.d)
ALPHABETIC (nn.D)

Pagination: Click one of these settings to specify whether pagination should be frozen or unfrozen. The dialog box displays the current setting. To change it, click the other setting.

Point Page Number Style: Click one of these settings to specify the numbering style for point pages.

- **Arabic:** The Arabic setting produces numbers such as 1, 1.1, 1.2, and 2.
- **roman:** The roman setting produces numbers such as 1, 1.i, 1.ii, and 2.
- **ROMAN:** The ROMAN setting produces numbers such as 1, 1.I, 1.II, and 2.
- **alphabetic:** The alphabetic setting produces numbers such as 1.a, 1.b, and 1.c. After the 26th point page, the numbering starts at *a* again, but doubles the letters, so the 27th point page would be 1.aa.
- **ALPHABETIC:** The ALPHABETIC setting produces numbers such as 1.A, 1.B, and 1.C.

Notes:

You cannot save a document with frozen pagination in MIF format, and you cannot import MIF files into documents with frozen pagination.

Front

Location: Tools window

Standard Keyboard Equivalent: `Esc o f`

Purpose: Use the Front command to change the drawing order of a page so that all selected objects or frames are drawn last, after all other objects on the page are drawn. The drawing order controls the appearance of overlapping objects. It also affects the selection of objects with the mouse: When you click with the arrow pointer over overlapping objects, FrameMaker selects the frontmost objects.

Use: Select one or more objects or frames and click Front in the Tools window.

Notes:

- When you use the Front command group, all subobjects in the group are brought to the front, but the drawing order within the group is maintained.
- See also the Back command and *Changing Object and Frame Overlap* in Chapter 2.

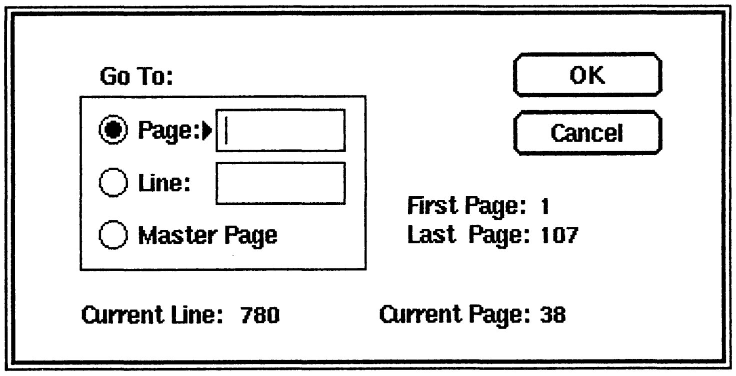
Go To

Location: Page menu

Standard Keyboard Equivalent: `Esc p g` and `Ctrl-g`

Purpose: Use the Go To command to go to any page within the document or to any line within a text flow. If the insertion point is in a line of text, the Go To command also shows the number of the line containing the insertion point.

Use: Choose the Go To command to display the Go To dialog box:



First Page: 1
Last Page: 107
Current Line: 780
Current Page: 38

Go To: In this area, specify where to go.

- **Page:** To go to a specific page, type in the page number and then click OK. Page 0 is the master page. If you specify a page beyond the end of the document, you go to the last page in the document.
- **Line:** To go to a specific line within a text flow (the insertion point must already be within a text flow), type in the line number and then click OK. Line numbers start with 1. If you specify a line number beyond the end of the text flow, you go to the last line in the text flow.
- **Master Page:** To go to the master page, click Master Page and then click OK.

Notes:

- If the first page of the document is not page 1 (in other words, if you have changed the First Page # in the Headers & Footers dialog box), then the numbers in these boxes must be the actual page numbers, not the relative page numbers within the document. For example, if the First Page # is 21, type `22` to go to the second page of the document. If the First Page # is 1, type `2`. The page numbers at the bottom of the document window show relative page numbers and also absolute page numbers when the two differ.

Grid

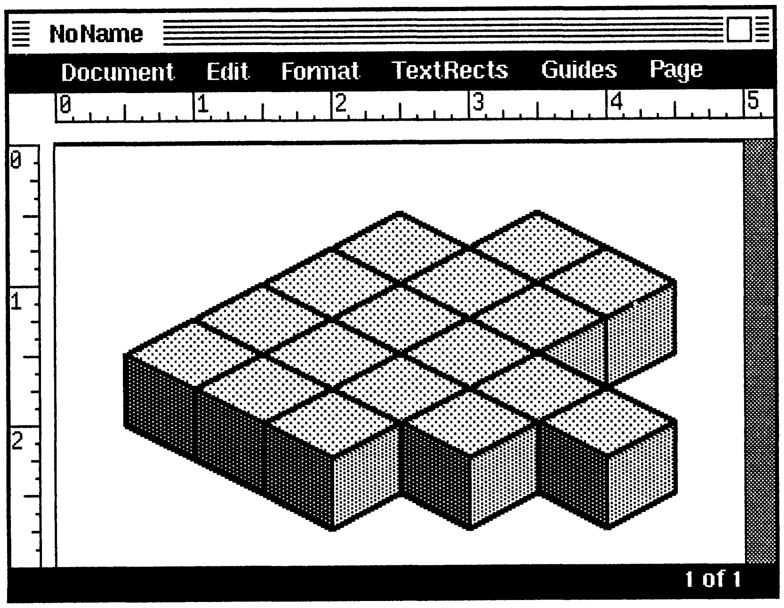
Location: Guides menu

Standard Keyboard Equivalent: `Esc g g`

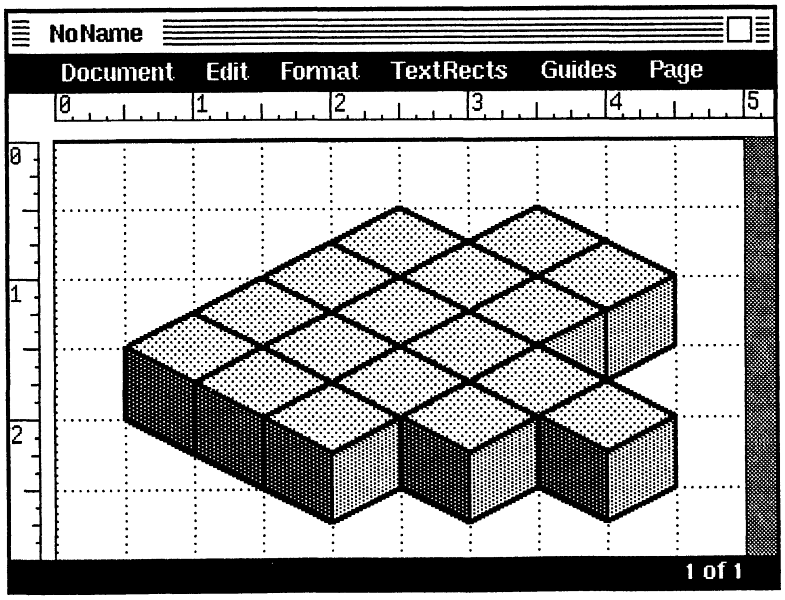
Purpose: Use the Grid command to turn the display of the grid on and off.

Use: Choose the Grid command to toggle the grid on and off. When the grid is visible, a mark (for example, a checkmark or an *x*) appears next to the Grid command in the Guides menu.

You control grid spacing with the Units command from the Guides menu.



No grid



1/2" grid

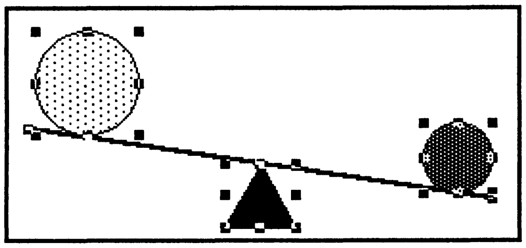
Group

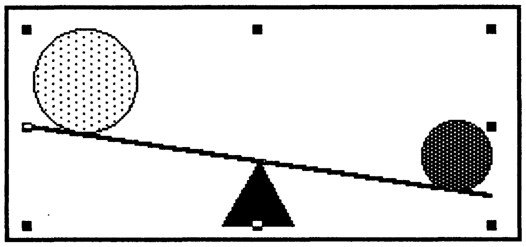
Location: Tools window

Standard Keyboard Equivalent: `Esc o g`

Purpose: Use the Group command to group all selected objects and/or groups into one object. Grouping has no visible effect on the objects when they are printed. You can, however, manipulate grouped objects as a single unit without changing their spatial relationships. Once you group objects, you can move, copy, resize, and flip them as if they were one object.

Use: Select two or more objects and/or groups and click Group in the Tools window. The handles on selected items are replaced by a single set of handles surrounding the entire group.


Handles before Grouping


Handles after Grouping

To select a group, select any of its elements; the entire group is selected.

Notes:

- Objects remain grouped until you explicitly ungroup them with the Ungroup command.
- Grouping is hierarchical; you can group several groups of objects. See the Ungroup command.
- Grouping does not affect drawing order.

Headers & Footers

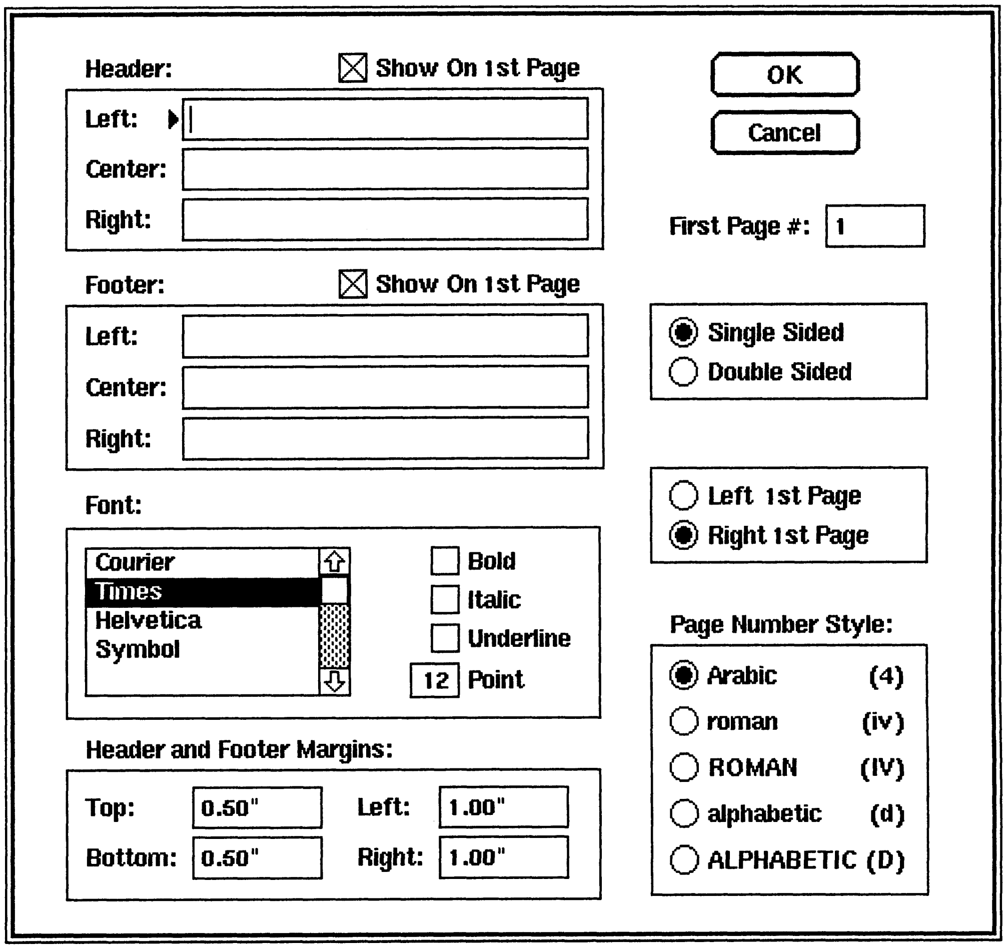
You can set up page headers and footers using the Headers & Footers command or using the master page. Each is described below.

Location: Format menu

Standard Keyboard Equivalent: `Esc f h`

Purpose: Use the Headers & Footers command to set up automatic headers and footers for each page, including automatic page numbering, running headers and footers, and odd/even page layout.

Use: Choose the Headers & Footers command to display the Headers & Footers dialog box:



Header: [✓ Show On 1st Page]
Left:
Center:
Right:

Footer: [✓ Show On 1st Page]
Left:
Center:
Right:

Font: Courier, [✓ Times], Helvetica, Symbol
Bold, Italic, Underline
12 Point

Header and Footer Margins:
Top: 0.50" Left: 1.00"
Bottom: 0.50" Right: 1.00"

OK
Cancel

First Page #: 1

[✓ Single Sided] Double Sided

Left 1st Page [✓ Right 1st Page]

Page Number Style:
[✓ Arabic (4)]
roman (iv)
ROMAN (IV)
alphabetic (d)
ALPHABETIC (D)

Header and **Footer:** In these areas, type in the left, center, and right header and footer for each page. Each item can be longer than the edit box provided; typing in a long string scrolls text off the left edge of the edit box.

You can print the current page number in any of these six header and footer items by typing a number sign (#) within the text. For example, specifying a Right Footer of `Page #` would print something like `Page 1` on the first page, `Page 2` on the second, and so forth.

You can also define running headers and/or footers (headers or footers that change as the headings in your document change). You print running headers or footers using the *dollar variables* `$1` or `$2` in one or more of the Header or Footer strings. FrameMaker then searches the text on each document page for a marker whose type is Header/Footer $1. If one is found, its marker text replaces `$1` in the header/footer string. If the page does not contain such a marker, then previous pages are searched, from bottom to top, until a Header/Footer $1 marker is found, and its text is used. If no such marker is found, the `$1` characters are dropped from the header/footer string on that page. The `$2` characters in a header/footer string work in the same way, except that they are replaced by text from Header/Footer $2 markers.

As an example, you might put a Header/Footer $1 marker at the start of each major section in your document. The marker text for each marker would be the name of the section (such as Introduction, Basics, Advanced Topics, Conclusion). To have the section name appear in the right header on each page, you might type the string `My Document: $1` in the Right Footer item. On each document page, this string is expanded to show the words *My Document* followed by text from the current Header/Footer $1 marker: *My Document: Introduction*, *My Document: Basics*, and so forth.

See the Markers command for more information on markers.

Font: In this area, select the font family, click the style, and set the size of the header and footer text. All automatic header and footer items have the same font.

Header and Footer Margins: In these boxes, type the left, right, top, and bottom margins for the headers and footers. These values are *indents* from the corresponding edges of the page. The header text sits just below the top margin, and the footer text's baseline lies at the bottom margin.

First Page #: In this box, type the number of the first page in terms of the automatic page numbering displayed in the header and footer. This setting allows you to break a single document into multiple files. By setting each file's starting page number appropriately, you can print and combine all of the individual files to create a continuously numbered document. See Chapter 5 for a more automatic way to handle multifile page numbering.

Single Sided or **Double Sided:** Click one of these settings to specify single- or double-sided page layouts.

For single-sided layouts, the headers and footer layouts on each page are identical: The left header and footer are always on the left of the page, and the right header and footer are always on the right. For double-sided layouts, the left and right headers and footers alternate on each page, so that the left and right items switch position on successive pages.

Text margins are also affected by the Single Sided or Double Sided setting. For double- sided layouts, the left and right margins alternate if they are unequal, so that the wider margin flips back and forth between the left and right sides of the page.

Use the Double Sided setting for documents that are to be printed on both sides of the paper. For example, this document uses the Double Sided setting.

Left or **Right 1st Page:** Click one of these settings to specify whether the first page is a left page or a right page. This setting is meaningful only if Double Sided layout is turned on. In this case, clicking Right 1st Page causes the page margins, headers, and footers to be normal (that is, left on the left and right on the right) on the first, third, fifth, and all other odd pages, and to be swapped (left on the right and right on the left) on the second, fourth, sixth, and all other even pages. Clicking Left 1st Page causes them to be reversed on the first, third, fifth, and all other odd pages and to be normal on the even pages. Each chapter in this document uses the Right 1st Page setting.

Page Number Style: Click one of these settings to specify how automatic page numbers in the header and footer are printed. Settings include numeric (1, 2, 3), lowercase roman (i, ii, iii), uppercase roman (I, II, III), lowercase alphabetic (a, b, c), and uppercase alphabetic (A, B, C).

Notes:

- You can also set up headers and footers using the master page. Place text and/or graphics along the top or bottom edge of the master page. Using the master page allows for more flexibility than is provided using the Headers & Footers command. For example, it allows multiline, mixed font, text and graphic headers and footers. It does not provide for different layouts on odd and even pages, however, and does not allow automatic page numbering (these last two features are provided using the Headers & Footers command).
- You can create headers and footers using a mixture of both master page items and items from the Headers & Footers command. For example, you can create a document using the Headers & Footers command to print the header and footer text and the master page to print rules in the footer.

Help

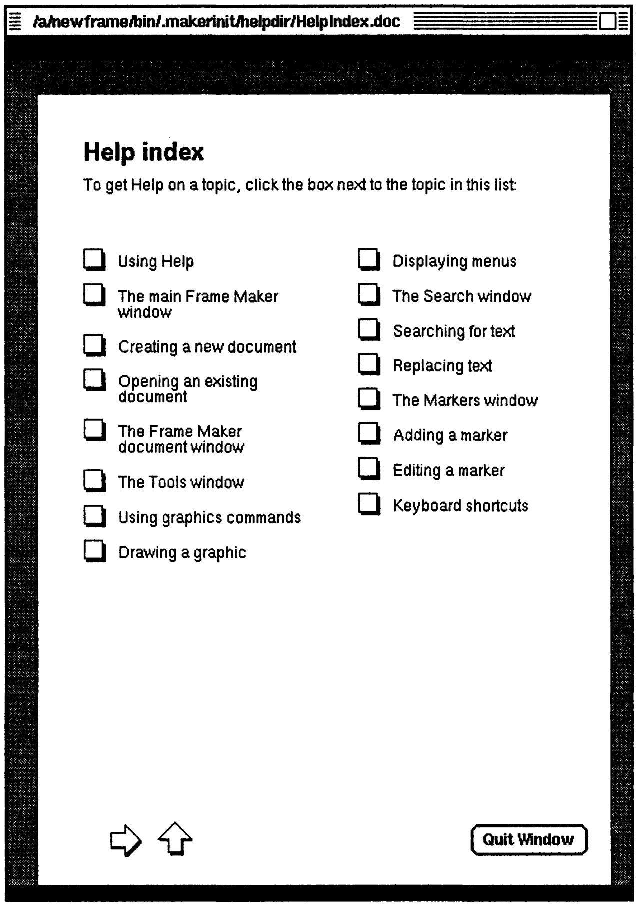
Location: Main FrameMaker window

Standard Keyboard Equivalent: `Esc m h`

Purpose: Use the Help command to display on-line information about FrameMaker.

Use: To use the Help command, click HELP in the main FrameMaker window. The first page of Help appears, showing the Help index. To get Help on a topic, click a box next to a topic in the index. For instructions on using Help, click the box next to *Using Help*. To quit Help, click Quit Window at the bottom of the Help window.

The Help index looks something like this:



/a/newframe/bin/.makerinit/helpdir/HelpIndex.doc

Help index

To get Help on a topic, click the box next to the topic in this list:

- ❑ Using Help
- ❑ The main Frame Maker window
- ❑ Creating a new document
- ❑ Opening an existing document
- ❑ The Frame Maker document window
- ❑ The Tools window
- ❑ Using graphics commands
- ❑ Drawing a graphic
- ❑ Displaying menus
- ❑ The Search window
- ❑ Searching for text
- ❑ Replacing text
- ❑ The Markers window
- ❑ Adding a marker
- ❑ Editing a marker
- ❑ Keyboard shortcuts

Because Help is actually a FrameMaker document window, you can manipulate it just as you do other FrameMaker document windows. See your window system documentation for information on manipulating document windows.

Import

Location: Document menu

Standard Keyboard Equivalent: `Esc d i`

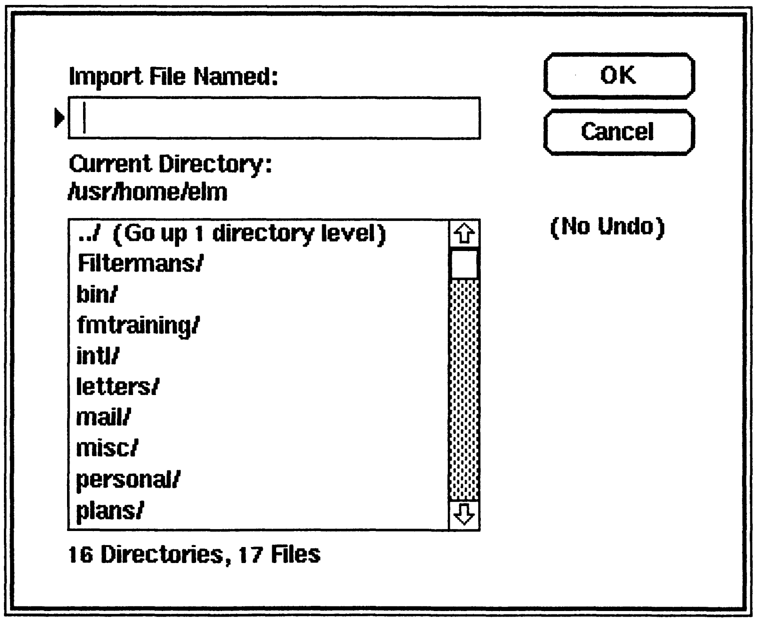
Purpose: Use the Import command to bring data from other files into a FrameMaker document. You can import the following types of files:

- X11 xwd format files (both XYPixmap and ZPixmap versions)
- X11 bitmap files
- X10 xwd format files
- Compressed and uncompressed Sun rasterfiles
- Encapsulated PostScript (EPS) files
- ASCII text files
- Maker Markup Language (MML) files
- Maker Interchange Format (MIF) files

How you import each of these file types is described below.

Your system administrator may have configured FrameMaker so you can import additional types of files, such as troff or DCA. Consult your system administrator on the types of files you can import at your site.

Use: Choose the Import command to display the Import dialog box:



Current Directory:
/usr/home/elm

```
../ (Go up 1 directory level)
Filtermans/
bin/
fmtraining/
intl/
letters/
mail/
misc/
personal/
plans/
```

16 Directories, 17 Files

Import File Named: In this box, type the name of the file to be imported. Use the scroll list to select a filename in the current directory or any directory you specify.

The Import scroll list initially lists the contents of the current document's directory. When FrameMaker imports the file, it uses a relative pathname (based upon the current document's directory) unless you specify an absolute pathname in this box.

OK: Click OK to read the file. If FrameMaker cannot read the file, it tells you and displays the Import dialog box again. If it can read the file, FrameMaker determines the file's type. FrameMaker groups files into three classes: image files, ASCII files, and MIF files. Each type is processed differently, as described below.

Importing Image Files

With FrameMaker, you can import two types of image files: bitmap files and Encapsulated PostScript (EPS) files. You can scale and crop both types of images to any dimension on the page. The steps you follow to import the two types of images vary slightly.

Importing Bitmap Files

- After importing the image, you can scale and crop it to any dimension on a page. You scale the image either by stretching it as you do rectangles or by using the Image File Scaling Options dialog box.

  When you import a bitmap image, the Image File Scaling Options dialog box appears:

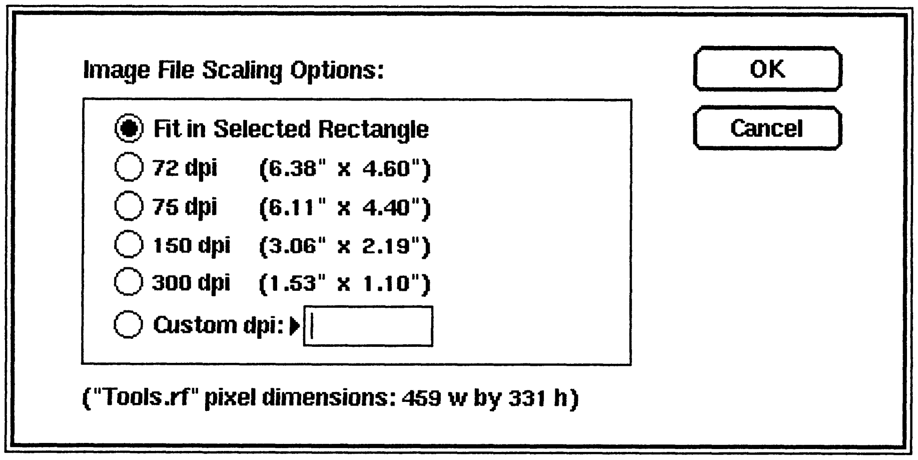


Image File Scaling Options:

[● Fit in Selected Rectangle]
○ 72 dpi (6.38" x 4.60")
○ 75 dpi (6.11" x 4.40")
○ 150 dpi (3.06" x 2.19")
○ 300 dpi (1.53" x 1.10")
○ Custom dpi:

("Tools.rf" pixel dimensions: 459 w by 331 h)

Use this dialog box to match dots in the bitmap to printer dots for undistorted printing. For the best printed results, choose a dots-per-inch (dpi) value that divides evenly into your printer's resolution. For example, for a 300-dpi printer, the following values are good choices: 75, 100, 150, and 300. However, 72 dpi is a poor choice (the image would be distorted when printed).

FrameMaker uses the screen as if it has 72 dpi. For this reason, images that look clear on the screen are distorted when you print them, and images that print clearly are slightly distorted on the screen:

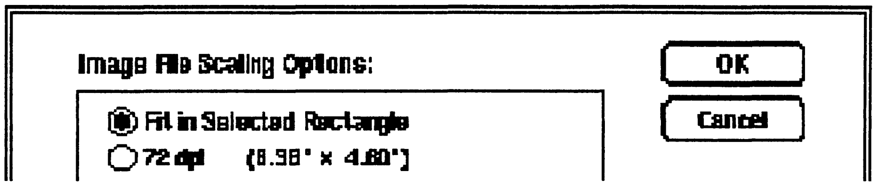


Image File Scaling Options:

[✓ Fit in Selected Rectangle]
72 dpi (8.98" x 4.60")

This image shows how part of the dialog box on the previous page looks when the page is viewed on the screen in a document window. Notice that the buttons and text are distorted on the screen, but they print correctly. This dialog box is imported at 100 dpi.

Importing Encapsulated PostScript Files

- You can import any Encapsulated PostScript (EPS) file that is in the ASCII 2.0 EPS format. To import an EPS file, type the filename in the Import dialog box and click OK. The image appears on the screen. Scale the image by stretching it just as you do rectangles (see *Graphics Editing Overview* in Chapter 2).
- An EPS file is a special kind of PostScript file you can create with a high-quality graphics application. Unlike PostScript TextRects, which appear on the screen as PostScript code, when you import an EPS file, you see on the screen a representation of the printed image.
- Like bitmap images, you can scale and crop EPS images. Unlike bitmap images, however, when you print a document containing an EPS image, FrameMaker sends PostScript code describing the image to the printer. The printed image is of very high quality, even if you've scaled the image (scaling a bitmap image, on the other hand, can produce uneven results when you print the document).
- Currently, most Macintosh applications create a type of EPS file that FrameMaker is unable to read directly (they generally create Macintosh 1.1 EPS files). FrameMaker, on the other hand, reads only ASCII 2.0 EPS files, so you must convert EPS files created on the Macintosh to a format that FrameMaker can read before uploading them to your workstation. For information about a small Macintosh program that converts Macintosh 1.1 EPS files to ASCII 2.0 EPS files, refer to /postscript/FrameEPSF/README in your main Frame directory.

Notes on Imported Image Files

This short section contains information relating to both bitmap files and EPS files.

- When you import an image into a document, FrameMaker doesn't put a copy of the image in the document file. Instead, it stores only the name of the image file. FrameMaker reads the image file when you open or print the document or when you scale the image within FrameMaker. If you change the image file, FrameMaker automatically updates your document.

  If you move or copy a FrameMaker document containing imported images, be sure to move the image files as well. If FrameMaker can't find an image when you open a document, it prompts you for the image file's correct pathname.
- Normally, FrameMaker imports the image into the center of the current page. To import an image directly into a frame or anchored frame, select the frame before choosing the Import command; FrameMaker imports the image into the center of the frame you selected.

  You can also select an existing image, square, or rectangle before using the Import command. In this case, the imported image replaces the selected object. The upper-left corner of the image will be placed at the upper-left corner of the rectangle. The imported image will be scaled as you specified in the Image File Scaling Options dialog box: If you chose Fit in Selected Rectangle, the image file will be scaled to fit inside the rectangle while retaining its original proportions.
- **Converting color image files:** FrameMaker cannot directly import color and gray scale image files, such as color EPS files or bitmap files captured on a color workstation. You can, however, use the shell script `fmcolor` to convert color bitmap files so you can import them. For instructions on using `fmcolor`, type **`fmcolor`** at the UNIX prompt.
- **Changing a bitmap image:** You can examine and change the scaling of a bitmap image by selecting it and then choosing the Import command. The Import dialog box appears, showing the filename of the image. When you click OK, the Image File Scaling Options dialog box appears. You can see (and change, if you want) the scaling of the image. When you're done, click OK to import the image file again and scale it as specified.
- **Cropping an imported image:** If an image contains portions you do not want printed, you can crop it in two ways:
  1. You can put the image inside a frame. Draw a frame (see Chapter 4), select the frame, and import the image into that frame. If the image has already been imported, cut it from its old location and paste it into the new frame using the Cut and Paste commands. Then shrink or stretch the frame as needed to cover the unwanted borders of the image (you can also scale or reposition the image within the frame).
  2. Cover the image with rectangles or polygons whose fill is White and whose pen is None. This is like placing white paper over the unwanted

portions of the image. An advantage to this approach is that FrameMaker displays the page faster than if you had used a frame. The disadvantages to this approach are that drawing the rectangle usually takes more time and that the rectangle may obscure other items.

- **Putting a border around an image:** Images can have a border pen pattern and line width just like rectangles. When you first import an image, it has a border pattern of None and the thinnest line width. You can change either of these settings by selecting the image and then clicking on a different border pattern and/or width in the Tools window. However, the border shown in this way is based on the actual edges of the image. If you want to outset the border from the image, draw a rectangle around the image to create the border (be sure to set the rectangle's fill pattern to None).

Importing Text Files

FrameMaker can import standard text files (ASCII files) that are both *line-oriented* (such as C programs and shell scripts) and *paragraph-oriented* (such as files containing document text). When you import a line-oriented file, each line in the file becomes a paragraph in the FrameMaker document. These paragraphs are typically one line long, but they may be wordwrapped to create more lines in some paragraphs. When you import a paragraph-oriented text file, FrameMaker merges multiple lines from the text file into paragraphs in the FrameMaker document.

There is nothing *within* a standard text file to tell FrameMaker whether the file is line-oriented or paragraph-oriented, so FrameMaker uses an arbitrary convention: If the file's name ends with the four characters `.txt`, then it is assumed to contain paragraph-oriented text. Otherwise it is assumed to contain line-oriented text.

Importing line-oriented text files: To import a line-oriented text file, put the insertion point at the place where you want the file's text to be inserted and use the Import command. Each character from the file is brought into the document as if it had been typed directly at the keyboard, so you can set the font and paragraph type at the insertion point before using the Import command. Each line ending in the text file becomes an end-of-paragraph symbol in the FrameMaker document.

Caution: When you specify a file to Import, FrameMaker tries to figure out what kind of file it is. If the file is not one of the types that FrameMaker recognizes (for example, MML, MIF, and image), then FrameMaker assumes that the file is a line-oriented text file. If you try to import a data file (perhaps a database file or an object code file), FrameMaker does its best to read the file in as a text file—but you may not like the results. If you start to import such a file and change your mind, press Ctrl-c to stop the import process.

Importing paragraph-oriented text files: To import a paragraph-oriented text file, rename it if needed so that the last four characters of its name are `.txt`. Then put the insertion point at the place where you want the file's text to be inserted and use the Import command. It is a good idea to put the insertion point at the start of a paragraph when importing a `.txt` file, since FrameMaker imports the text as complete paragraphs.

FrameMaker tries to group lines from `.txt` files into paragraphs using blank lines and indented lines to detect paragraph boundaries. Each blank line or set of adjacent blank lines signals a paragraph break (the blank lines do not end up in the FrameMaker document). Also, each indented line starts a new paragraph.

The imported paragraphs have tags of the form Indent *n* , where *n* is a number indicating the indent level of the paragraph as it occurred in the text file. You can change the default formats for these tags. See Appendix D, *Customizing FrameMaker,* for more information.

Importing tables from standard text files: In standard text files, columns are usually separated by tab symbols. The number of tabs between one column and the one to its right varies depending on the number of characters in the left column. In FrameMaker, on the other hand, each column in a table is separated by a single tab symbol, no matter how wide the entry to its left is. Therefore, tables imported into FrameMaker from standard text files do not line up correctly without some cleanup within FrameMaker.

Importing MML Files

MML stands for Maker Markup Language. An MML file is a standard ASCII text file that uses markup statements to define the formatting of text. Chapter 6 provides a complete description of MML and its use.

To import an MML file, put the insertion point where the MML file's contents should be inserted and use the Import command.

Importing MIF Files

MIF stands for Maker Interchange Format. A MIF file is a standard ASCII text file that uses a custom language to define FrameMaker document text and/or graphics. Chapter 7 provides a complete description of MIF and its use.

MIF has a wide variety of import uses—many of them involve bringing data into FrameMaker from other computer programs. But there are two important uses for MIF that involve no other computer programs: importing a Paragraph Catalog from one document into another and merging separate FrameMaker documents into a single document.

Importing a MIF Catalog: You can save a document's Paragraph Catalog as a MIF file (see the Save command). To import such a MIF Catalog file into a document file, open the document file and use the Import command, specifying the MIF Catalog's filename. The paragraph formats in the MIF file are *merged* into the document's Catalog as follows: If the MIF file and the document's Catalog both contain a paragraph format with the same name, the MIF file version replaces the one in the Catalog. Other paragraph formats from the MIF file are added to the document's Catalog.

Merging FrameMaker documents: To append `FileB.doc` onto the end of `FileA.doc`, follow these steps:

- Open `FileB.doc`.
- Use the Save command to save a copy of `FileB.doc` using MIF format. Save the MIF version in a file called `FileB.mif` (see the Save command for more information).
- Open `FileA.doc`.
- Go to the last page in `FileA.doc`.
- Use the Import command. Specify `FileB.mif` as the file to import.

The contents of `FileB.mif` are added to the end of `FileA.doc`. `FileB.doc` is unaffected by this merger.

A small variation on the above procedure can be used to merge `FileB.mif` into the *middle* of `FileA.doc`. Go to a page within `FileA.doc` and use the Import command. FrameMaker imports `FileB.mif` after whatever page is visible when you choose the Import command.

Info

Location: Main FrameMaker window and Document menu

Standard Keyboard Equivalent: `Esc m i` and `Esc d I`

Purpose: Use the Info command to display a dialog box showing license, version, copyright, address, and telephone number information on the running version of FrameMaker.

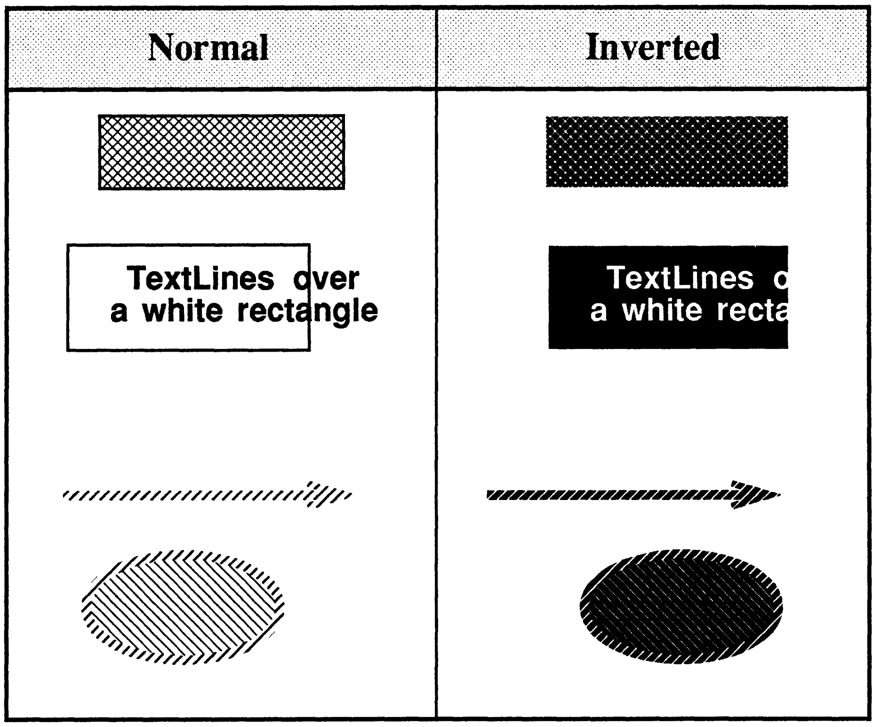
Use: Click INFO in the main FrameMaker window or choose the Info command from the Document menu. When you are done reading the dialog box, click OK to continue or click License to display the License dialog box. For more information, see the License command.

Invert

Location: Tools window

Standard Keyboard Equivalent: `Esc o i`

Purpose: Use the Invert command to change black to white and white to black in selected objects, toggling between *normal* and *inverted* states. In the inverted state, those portions of an object that usually appear black are drawn in white, and the portions that are normally white are drawn in black:



Samples of Normal and Inverted Objects

Use: Select one or more objects or groups and then click Invert in the Tools window. Clicking Invert repeatedly switches the selected objects back and forth between normal and inverted states.

Notes:

- You can invert all objects (except TextRects). When you invert a frame containing objects, however, the objects within the frame are not inverted.
- The Invert command affects both object fill patterns and object borders.

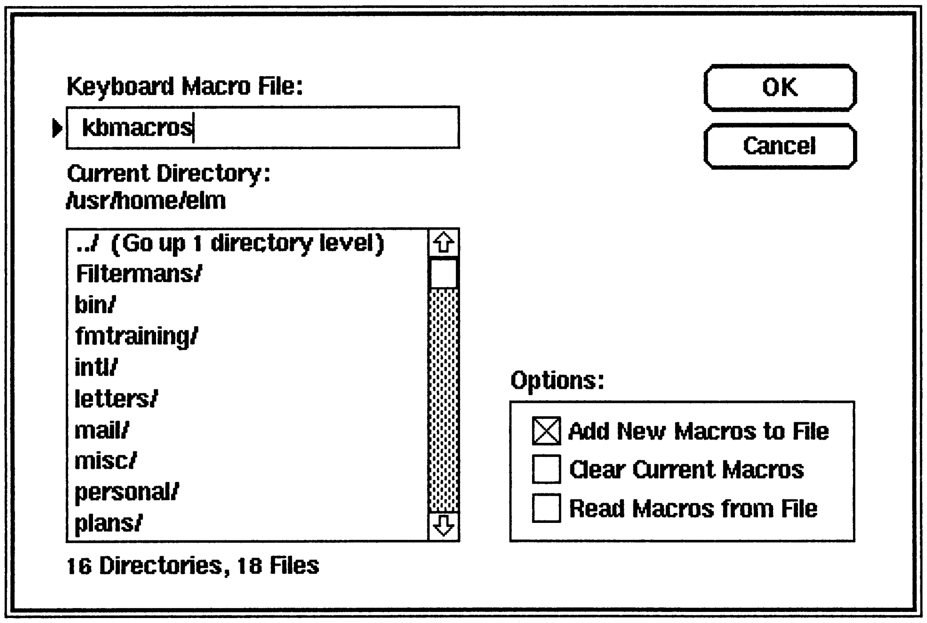
Keyboard

Location: Document menu

Standard Keyboard Equivalent: `Esc d k` and `Esc R`

Purpose: Use the Keyboard command to save recorded keyboard macros in a file, to clear recorded keyboard macros, and to read previously saved keyboard macros from a macro file. See the Record Keys command for additional information.

Use: Choose the Keyboard command to display the Keyboard dialog box:


Keyboard Macro File:
kbmacros
Current Directory:
/usr/home/elm
../ (Go up 1 directory level)
Filtermans/
bin/
fmtraining/
intl/
letters/
mail/
misc/
personal/
plans/
16 Directories, 18 Files
Options:
[✓ Add New Macros to File]
Clear Current Macros
Read Macros from File

Keyboard Macro File: In this box, type the name of a macro file to store macros in or to read macros from, depending on the selected settings. Use the scroll list to select a filename in the current directory or any directory you specify. When you start FrameMaker, the Keyboard scroll list initially lists the contents of your home directory.

The file `kbmacros` in your home directory is a good place to store macros that you want to use all of the time. Each time you start FrameMaker, it looks for a file called `kbmacros`. FrameMaker looks for `kbmacros` in three places: the directory where FrameMaker is started, your home directory, and finally in the `.makerinit` directory. If a `kbmacros` file is found in one of these places, the macros it contains are read in automatically. Once a `kbmacros` file is found, FrameMaker does not look for others.

Options: Turn on one or more of these settings. They are processed in the order listed in the dialog box: Add first, Clear next, and Read last.

- **Add New Macros to File:** Turn on this setting to append new macros to the specified keyboard macro file. New macros are those recorded since the last time the Add or Clear setting was used or since FrameMaker was started, whichever is most recent.

This setting works as follows: It copies (appends) the contents of `~/kblog` to the specified file, and then it renames `~/kblog` to `~/kblog.backup`.

- **Clear Current Macros:** Turn on this setting to remove all keyboard macros, including newly recorded macros and those read in from macro files. This setting sets the keyboard to its built-in configuration, and then reads the file `kbmap`, if it exists. See Appendix D, Customizing FrameMaker, for information on the `kbmap` file and on the keyboard base configuration.
- **Read Macros from File:** Turn on this setting to merge keyboard macros from the specified keyboard macro file into the current keyboard configuration. Because this is a merge operation, you can have a primary macro file that defines macros common to all your work and then have secondary macro files that are each used for a subset of your work. If each secondary file defines macros for keys not used in the primary set, you can read macros from different secondary macro files without losing your primary macros.

Notes:

- See Appendix D, *Customizing FrameMaker,* for general information on keyboard configurations, keyboard macros, and keyboard macro files.

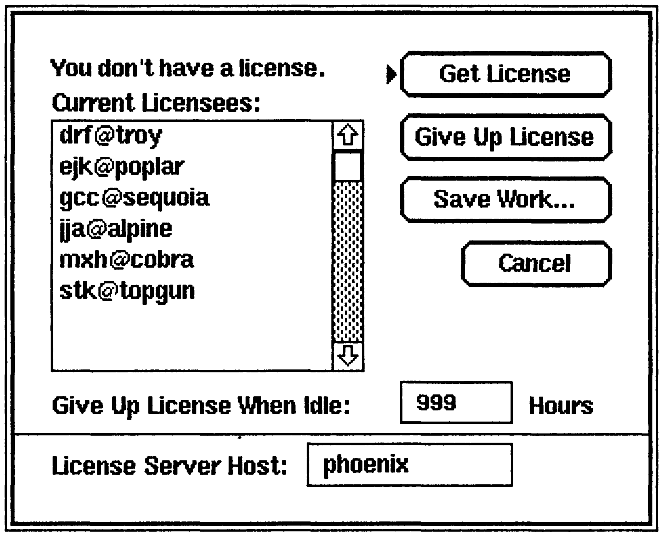
License

Location: The License command appears as a button in the Info dialog box. You can choose the License command even when you are unable to obtain a license.

Purpose: You can use the License command for several purposes:

- To obtain a license
- To give up your license
- To save work completed before you lost your license
- To set your personal time-out period (how long you can leave FrameMaker idle without losing your license)
- To specify your license server host (the host running the licensing program)

Use: Click INFO in the main FrameMaker window to display the Info dialog box and then click License to display the License dialog box:


You don't have a license.
Current Licensees:
drf@troy
ejk@poplar
gcc@sequoia
jja@alpine
mxh@cobra
stk@topgun
Give Up License When Idle: 999 Hours
License Server Host: phoenix

Current Licensees: This scroll list displays the users who have obtained licenses from the host displayed in the License Server Host edit box. The status line above the scroll list tells you whether you have a license.

Get License: Click this button to try to obtain a license.

Give Up License: Click this button to give up your license.

Save Work: Click this button if you have lost your license and you have any unsaved documents open.

Give Up License When Idle: In this edit box, specify how many hours FrameMaker should remain idle before you lose your license. The number you specify must be a whole number (at least 1 hour). Your system administrator may have set an upper limit on the amount of time you can leave FrameMaker idle without losing your license. See *Licensing Instructions* in Chapter 9 for more information.

License Server Host: This edit box displays your current license server host. Edit the contents of this edit box to change license server hosts. For example, you would do this if the primary license server host is not available and you need to switch to the backup server. For more information, see your system administrator.

Notes:

- When you use FrameMaker, you do so with either a reserved or a floating license. A reserved license is reserved solely for your use and is always available to you. Floating licenses are available to users on a first-come, first-served basis. As a result, if you would normally use FrameMaker with a floating license, you may occasionally find that all such licenses are in use (that is, you are unable to use FrameMaker). In addition, your system administrator specifies how many hours FrameMaker can remain idle before a user loses a license. As a result, if you use FrameMaker and then leave it idle for some time, you may lose your license.

  Even if you don't have a license, you still can:
  - Manipulate FrameMaker windows
  - Save any unsaved work
  - Use the License command
  - Quit FrameMaker

  You cannot continue to use other FrameMaker features, however, until you obtain a license.
- If you are using a floating license and do not intend to use FrameMaker for a while, but do not want to quit FrameMaker, we recommend that you give up your license; your license then becomes available for another user.
- See Appendix H, *Licensing Procedures,* for a description of Frame's licensing policy and instructions for setting up and customizing licenses.
- Changes you make to the Give Up License When Idle and License Server Host settings take effect when you click Get License.

Markers

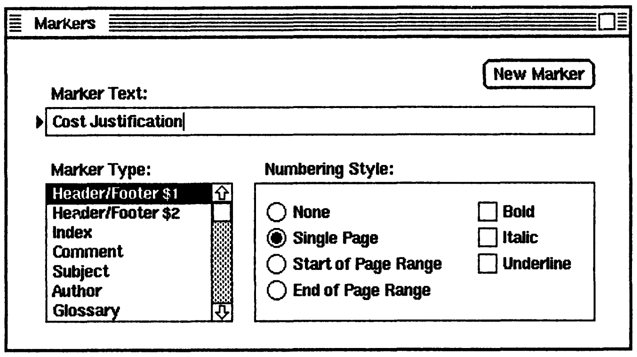
Location: Edit menu

Standard Keyboard Equivalent: `Esc e m`

Purpose: Use the Markers command to store and edit markers and to view marker contents in document text. Unlike a standard dialog box, the Markers window can remain on the screen while you work within a document or UNIX window.

Markers are special nonprinting characters stored within document text. Each marker "contains" a text string and other information described below. The text stored inside markers is used for running headers and footers, for indexing and similar list-generation tasks, and for embedding nonprinting comments within a document.

Use: Choose the Markers command to display the Markers window.



Marker Text: Cost Justification

Marker Type: [✓ Header/Footer $1] Header/Footer $2 Index Comment Subject Author Glossary

Numbering Style: None [✓ Single Page] Start of Page Range End of Page Range

☐ Bold ☐ Italic ☐ Underline

After the window appears, you may want to move it to a position on the screen where it does not overlap document windows. See your window system documentation for information on manipulating windows.

To create a new marker, move the arrow pointer to the desired position within some TextRect text and click with the middle mouse button to set an insertion point. Then move the arrow pointer into the Markers window, fill in the desired values, and click the New Marker button or press the Enter key to store the new marker. Markers are not visible within the text unless text symbols are turned on using the Text Symbols command from the Guides menu. The symbol for a marker is a large "tee" (T).

To edit an existing marker, select it using the middle mouse button, keyboard selecting commands, or the Search command (search for \m). If the selected text contains one marker, the Markers window is updated to show the marker's contents, and the New Marker button is changed to read Edit Marker. If the selected text contains more than one marker, the Markers window shows the contents of the first Marker in the selected text. Modify the values in the Markers window as needed and click the Edit Marker button to store the new values.

To delete a marker, select the marker and press the Delete key.

Items in the Markers window have the following meanings:

Marker Type: Specifies the marker's type. Use header/footer-type markers to set up running headers and footers (see the Headers & Footers command). Use other marker types in conjunction with the `fmbook` program, described in Chapter 5, to create automatically generated lists such as indexes.

Marker Text: Specifies the marker's "contents." For header/footer-type markers, the marker text is displayed within the header/footer strings. For index-type markers, the marker text specifies index entries using a powerful syntax described in Chapter 5.

Numbering Style: These settings are relevant only for markers used with the `fmbook` program. See *Preparing Documents for an Index* in Chapter 5.

New/Edit Marker: Clicking in this button stores a new marker or updates the settings of an existing marker.

Notes:

- *Preparing Documents for an Index* in Chapter 5 contains useful tips for working with markers.
- You can change the list of marker types by editing a configuration file. See Appendix D, *Customizing FrameMaker,* for more information.

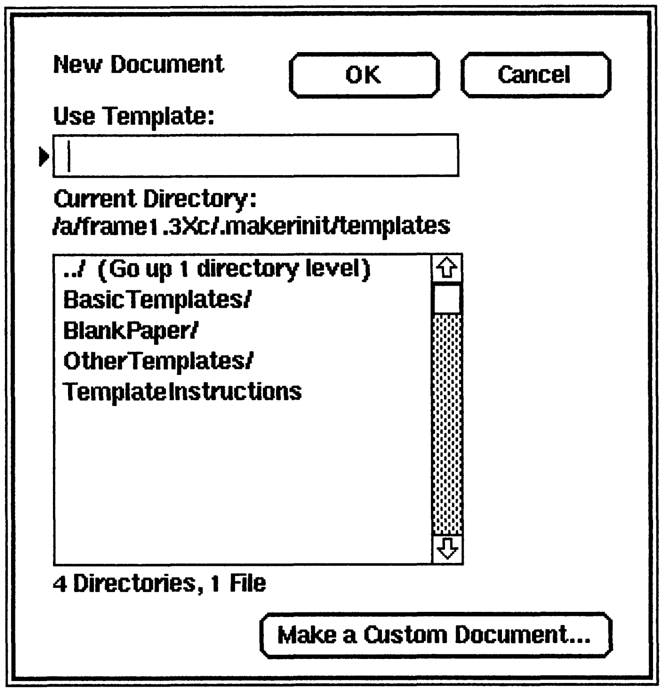
New

Location: Main FrameMaker window and Document menu

Standard Keyboard Equivalent: `Esc d n`

Purpose: Use the New command to create a new document (in memory only). To save a new document to disk, use the Save command from the Document menu.

Use: Click NEW in the main FrameMaker menu or choose New from the Document menu to display the New dialog box:



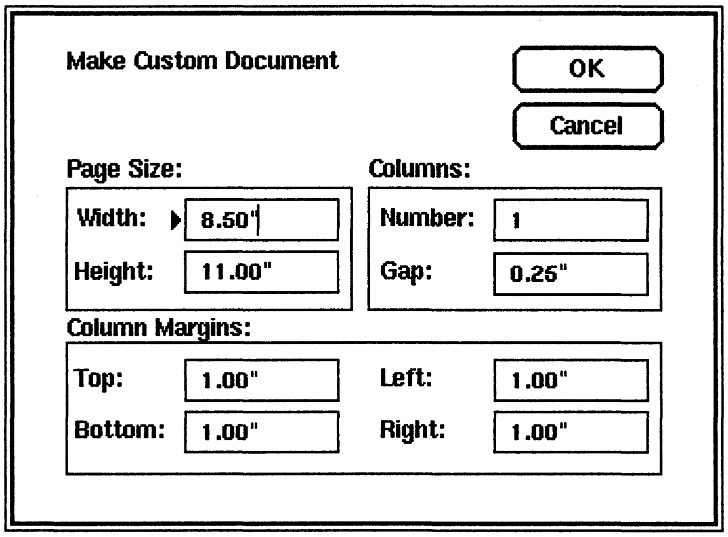
Current Directory:
/a/frame1.3Xc/.makerinit/templates

../ (Go up 1 directory level)
BasicTemplates/
BlankPaper/
OtherTemplates/
TemplateInstructions

4 Directories, 1 File

Use Template: In this box, type the name of the template from which you want to create the document. Use the scroll list to select a template in the current directory or any directory you specify. For more information about the templates and template directory, see the notes for this command.

Make a Custom Document: When you click Make a Custom Document, the Make Custom Document dialog box appears. In this dialog box, you can set the page size, the number of columns, and the column margins for the new document.


Make Custom Document

Page Size:
Width: 8.50"
Height: 11.00"

Columns:
Number: 1
Gap: 0.25"

Column Margins:
Top: 1.00"
Bottom: 1.00"
Left: 1.00"
Right: 1.00"

Page Size: In this area, specify the width and height. The maximum value for both width and height is 14". However, FrameMaker does not gracefully support page sizes that are taller than the screen (sizes taller than 11.66" are problematic). After you create a document, you cannot change its page size.

Columns: In this area, specify the number of columns (between 1 and 10) and the gap between columns. The *gap* is the horizontal distance between each column. The columns are equal in size, with equal gaps. The column width is calculated so that the left edge of the left column is at the specified left margin and the right edge of the right column is at the right margin.

The margin and column settings affect the layout of the document's first page when FrameMaker creates it and the layout of each new page that you add to the document. You can, however, change these settings using the Number of Columns command from the Format menu. In addition, once a page is created, you can change its margin and column settings with the Number of Columns command.

Margins: In this area, specify the top, bottom, left, and right margins for the TextRect(s) automatically generated on each new page (see *FrameMaker Concepts* in Chapter 2). The values represent distances "in" from each edge. For example, a right margin of 1.5" means a right margin 1.5 inches in from the right edge of the page. If the document is double-sided, the Left margin means the edge against the binding, and the Right margin means the outer edge (see the Headers & Footers command).

These margins do not affect where you place headers and footers (you can place them anywhere on the page). Most printers, however, cannot print along the outer

1/4" of each page, so if you place your headers and footers very close to the edge of the page, they may not print.

OK: Click OK to open the template you have selected or to create a new document with the custom dimensions you've specified. A window appears in the upper-left corner of the screen showing the new document's first page, with a text insertion point active in the first column on the page. New custom documents appear with rulers, borders, and text symbols visible and snap turned on. These settings can be turned off using commands from the Guides menu.

Notes:

- When you first choose the New command, FrameMaker looks for the following directories and lists the contents of the first one it finds:

  1. `./fmtemplates` (the `fmtemplates` subdirectory in the directory in which FrameMaker was started)
  2. `~/fmtemplates` (the `fmtemplates` subdirectory in your home directory)
  3. *install_dir*`/fmtemplates` (the `fmtemplates` subdirectory in the directory in which FrameMaker was installed)
  4. ~ (your home directory)

  The *install_dir*`/fmtemplates` directory contains standard templates provided with FrameMaker.

- If you would like to use your own templates as well as the standard templates, create an `fmtemplates` directory in your home directory and create a symbolic link to our standard templates.

  The following series of commands, for example, make the necessary directory and create a link to the standard templates:

  ```
  % cd ~
  % mkdir fmtemplates
  % cd fmtemplates
  % ln -s install_dir/fmtemplates StandardTemplates
  ```

  After creating your `~/fmtemplates` directory, create your templates as you would any other document and save them in the new `~/fmtemplates` directory.

- The International version of FrameMaker includes templates for languages other than US English. For more information, see *International Version Template Defaults* in Appendix D.

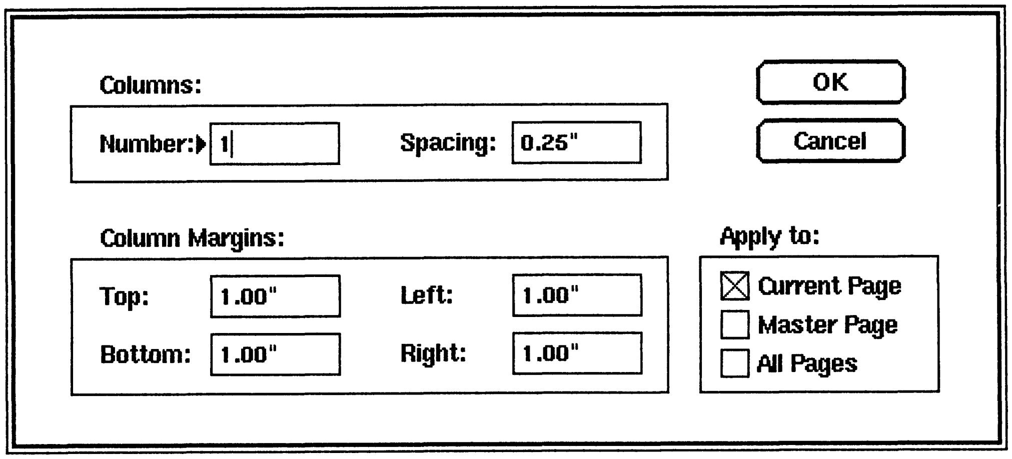
Number of Columns

Location: Format menu

Standard Keyboard Equivalent: `Esc f n`

Purpose: Use the Number of Columns command to change the number and/or margins of TextRects (columns) automatically generated on each new page (see Chapter 2) and to reformat existing pages whose TextRects were automatically generated in the first place. Using this command on custom-designed pages is not likely to match your intentions.

Use: Choose the Number of Columns command to display the Number of Columns dialog box:



Columns:

Number: 1 Spacing: 0.25"

Column Margins:

Top: 1.00" Left: 1.00"

Bottom: 1.00" Right: 1.00"

Apply to:

☒ Current Page
☐ Master Page
☐ All Pages

Columns: In these boxes, type the number and spacing of columns. The number of columns can range from 1 to 10. Spacing is the horizontal distance between columns. The generated columns are equal in size, with equal intercolumn spacing. The column size is calculated so that the left edge of the left column coincides with the specified left margin and the right edge of the right column coincides with the right margin.

Column Margins: In these boxes, type the Top, Bottom, Left, and Right margins for the columns. The values represent distances "in" from each edge. For example, if you specify a right margin of 1.5", you get a right margin 1.5 inches in from (left of) the right edge of the page. If you specify a double-sided document, the Left margin really means the edge against the binding, and the Right margin means the margin along the outer edge (see the Headers & Footers command).

Apply To: Click one or more of these settings to specify which pages are affected by the new column and margin settings.

- **Current Page:** Turn on this check box to apply the settings to the page visible when you choose the Number of Columns command. If the master page is visible, the effect is equivalent to that described under the Master Page setting below.

  The process FrameMaker uses when reformatting pages is as follows: Find the first TextRect on the page (in draw order) that has the auto-connect feature turned on. If that TextRect is connected to other TextRects on the page, delete those TextRects, moving their contents back into the first TextRect and connecting the last TextRect's tail to the first TextRect. Finally, split and resize the first TextRect as needed to produce the correct number and position of columns.
- **Master Page:** Turn on this check box to apply the specified settings to the master page. The master page settings affect the column layout of new pages as you add them to a document. When you display the master page, you can see the settings.
- **All Pages:** Turn on this check box to apply the specified settings to all document pages except the master page. This setting uses the logic described above for Current Page, applying it to each document page.

Notes:

- On the master page, the page margin settings appear as rectangles.

Open

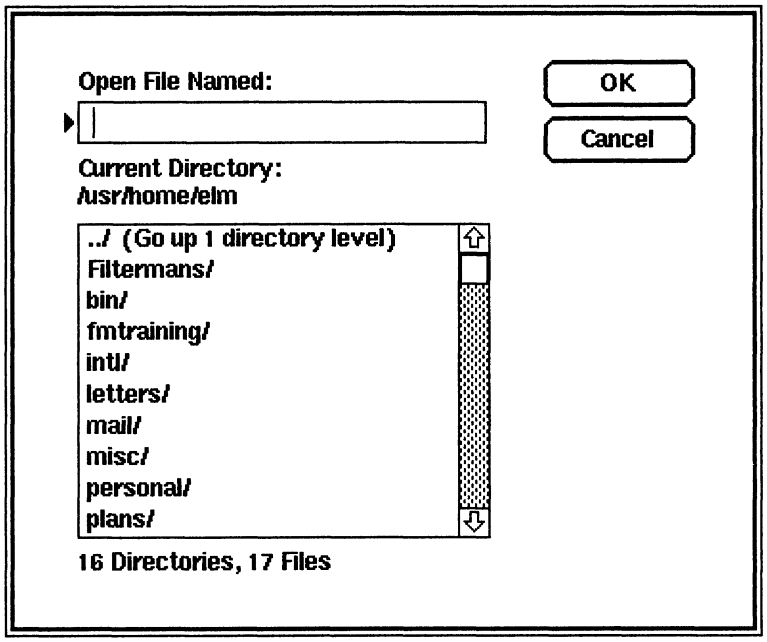
Location: Main FrameMaker window and Document menu

Standard Keyboard Equivalent: `Esc d o`

Purpose: Use the Open command to display a previously saved FrameMaker document for viewing, editing, and/or printing. You can also use this command to open standard text files, MML files, and MIF files.

Opening a document places a copy of the document in memory; it does not affect the disk file. To update the disk file after editing a document, choose the Save command from the Document menu.

Use: Choose the Open command to display the Open dialog box:


Open File Named:

Current Directory:
/usr/home/elm

../ (Go up 1 directory level)
Filtermans/
bin/
fmtraining/
intl/
letters/
mail/
misc/
personal/
plans/

16 Directories, 17 Files

Open File Named: In this box, type the name of the file to be opened. You can also use the scroll list to select a filename in the current directory or any directory you specify. When you start FrameMaker, the Open scroll list initially lists all the files in the directory in which you started FrameMaker. If you open a document in another directory, the second directory's contents are listed when you next choose the Open command.

OK: Click OK to open the specified file. If the document can be opened, a window is placed on the screen showing the document's first page. The window's position and display settings match those settings at the time the document was last saved. If your screen is not large enough to display an entire page, FrameMaker displays as much of the page as possible and adjusts the vertical ruler to show accurate measurements. If the file cannot be opened, FrameMaker displays a message explaining why.

When you use the International version of FrameMaker, you can create paragraphs with a Language setting other than USEnglish. If you later try to open the documents with a U.S. version of FrameMaker, the Language setting in all paragraphs will be reset to USEnglish.

See Chapters 7 and 8 for information on opening MML and MIF files.

> **Note:** You can open and edit a MIF or MML file as if it were a standard text file. To do so, use the Open command and fill in the MIF/MML filename. Then hold down the Alt key and click OK. Pressing the Alt key stops FrameMaker from interpreting the MIF or MML in the file; FrameMaker reads in the file's text for editing.

Notes on Opening Text Files

FrameMaker can open standard text files (ASCII files) that are both *line-oriented* (such as C programs and shell scripts) and *paragraph-oriented* (such as files containing document text). When opening a line-oriented file, each line in the file becomes a paragraph in the FrameMaker document. These paragraphs are typically one line long, but they may be wordwrapped to create more lines in some paragraphs. When opening a paragraph-oriented text file, FrameMaker merges multiple lines from the text file into paragraphs in the FrameMaker document.

There is nothing *within* a standard text file to tell FrameMaker whether the file is line-oriented or paragraph-oriented, so FrameMaker uses an arbitrary convention: If the file's name ends with the four characters `.txt`, then it is assumed to contain paragraph-oriented text. Otherwise it is assumed to contain line-oriented text.

Opening line-oriented text files: To open a line-oriented text file, use the Open command and fill in the desired text file's name.

The document and paragraph format used for line-oriented text files is determined by files in `.makerinit`. After FrameMaker determines that the text file is not a MIF or MML file and that its name does not end with `.txt`, it searches `.makerinit` for a file named *xxx*`Template.doc`, where *xxx* is the suffix of the file being opened. For example, if the text file `foo.c` is being opened, FrameMaker looks for the file `cTemplate.doc`. If it fails to find such a file, it looks for the file called `ASCIITemplate.doc`. If it cannot find that file, it uses the default document setup for the New command.

Using this file-naming convention, you can set up our own default document formats in `.makerinit`. Do so by creating, in `.makerinit`, an empty FrameMaker document with the correct margins, number of columns, Paragraph Catalog, and initial font setting. FrameMaker opens the appropriate document and then brings in the text file, much as if each character in the file were typed into the document.

Each end-of-line character in the text file causes a paragraph ending in the FrameMaker file.

Opening paragraph-oriented text files: To open a paragraph-oriented text file, rename it if needed so that the last four characters of its name are `.txt`. Then use the Open command and fill in the desired text file's name.

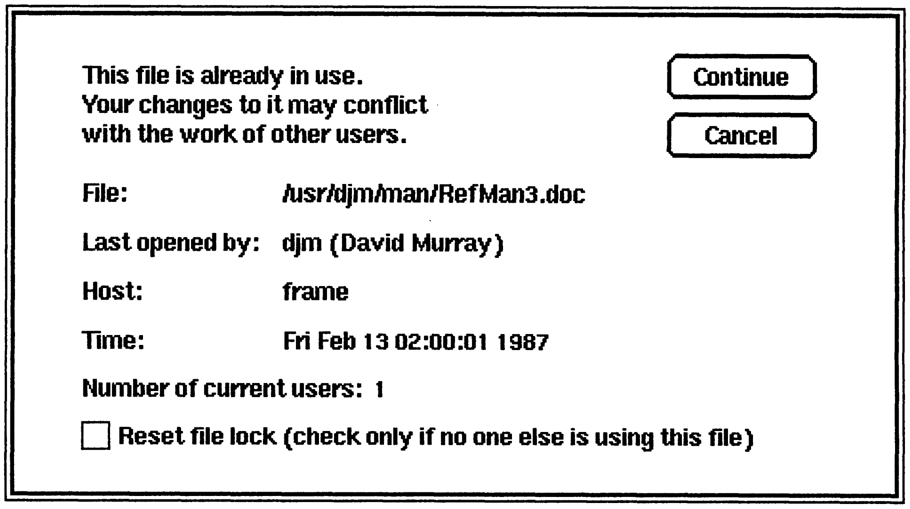
FrameMaker tries to group lines from `.txt` files into paragraphs using blank lines and indented lines to detect paragraph boundaries. Each blank line or set of adjacent blank lines signals a paragraph break (the blank lines do not end up in the FrameMaker document). Also, each indented line starts a new paragraph.

The imported paragraphs have tags of the form Indent *n* , where *n* is a number indicating the indent level of the paragraph as it occurred in the text file. You can change the default formats for these tags. See Appendix D, *Customizing FrameMaker,* for more information.

Notes on File Locking

FrameMaker offers a simple kind of file locking to help prevent multiple users from trying to edit the same document at the same time.

If a user tries to open a file to which he or she has write access and that file is already being edited, FrameMaker displays the following dialog box:



This file is already in use.
Your changes to it may conflict
with the work of other users.

File: /usr/djm/man/RefMan3.doc

Last opened by: djm (David Murray)

Host: frame

Time: Fri Feb 13 02:00:01 1987

Number of current users: 1

☐ Reset file lock (check only if no one else is using this file)

The dialog box shows the name of the file being opened, the name of the person who already has this file open (this may be the same user who is opening a second copy of the document), the name of that person's workstation, and the time the file was opened. The number of current users is also shown, to indicate if more than one other person has the file open.

Continue: Clicking Continue opens the file, even though it is already open elsewhere. FrameMaker has no problem with a file being opened by many users at one time. It is possible, however, for one user to overwrite the work of others:

The last person to save his or her work destroys all work done by others on the file while he or she was editing.

Cancel: Clicking Cancel does not open the document; it returns you to the Open dialog box where you can choose another file to open.

Reset File Lock: Click this check box if you are sure that no one else is really editing the file (to clean up an erroneous lock file). FrameMaker uses lock files to keep track of what files are being edited at any point in time (the lock file for `foo.doc` would be `foo.doc.lck`). FrameMaker may be unable to delete these lock files when it should, due to a problem such as a power failure or system crash.

Page

Location: Page menu

Standard Keyboard Equivalent: See below.

Purpose: Use the Page Previous, Next, First, Last, and Master commands to move to various pages within the document:

- **Previous:** Choose Previous to display the previous page of the document. Invoking the Previous command from page 1 displays the master page (the master page is really page 0). On the master page, the Previous command has no effect.

  Keyboard equivalent for Previous: `Esc p p`
- **Next:** Choose Next to display the next page of the document. Invoking the Next command from the master page displays page 1. On the final document page, the Next command has no effect.

  Keyboard equivalent for Next: `Esc p n`
- **First:** Choose First to display the first page of the document (not the master page).

  Keyboard equivalent for First: `Esc p f`
- **Last:** Choose Last to display the final page of the document.

  Keyboard equivalent for Last: `Esc p l`
- **Master:** Choose Master to display the document's master page.

  Keyboard equivalent for Master: `Esc p m`

Notes:

- See the Go To and Scroll commands also.
- For more keyboard equivalents, see Appendix A, *Keyboard Commands.*

Paragraphs

FrameMaker supports paragraph formatting changes using the Paragraphs command and also using the mouse. Each function is described below.

The Paragraphs Command

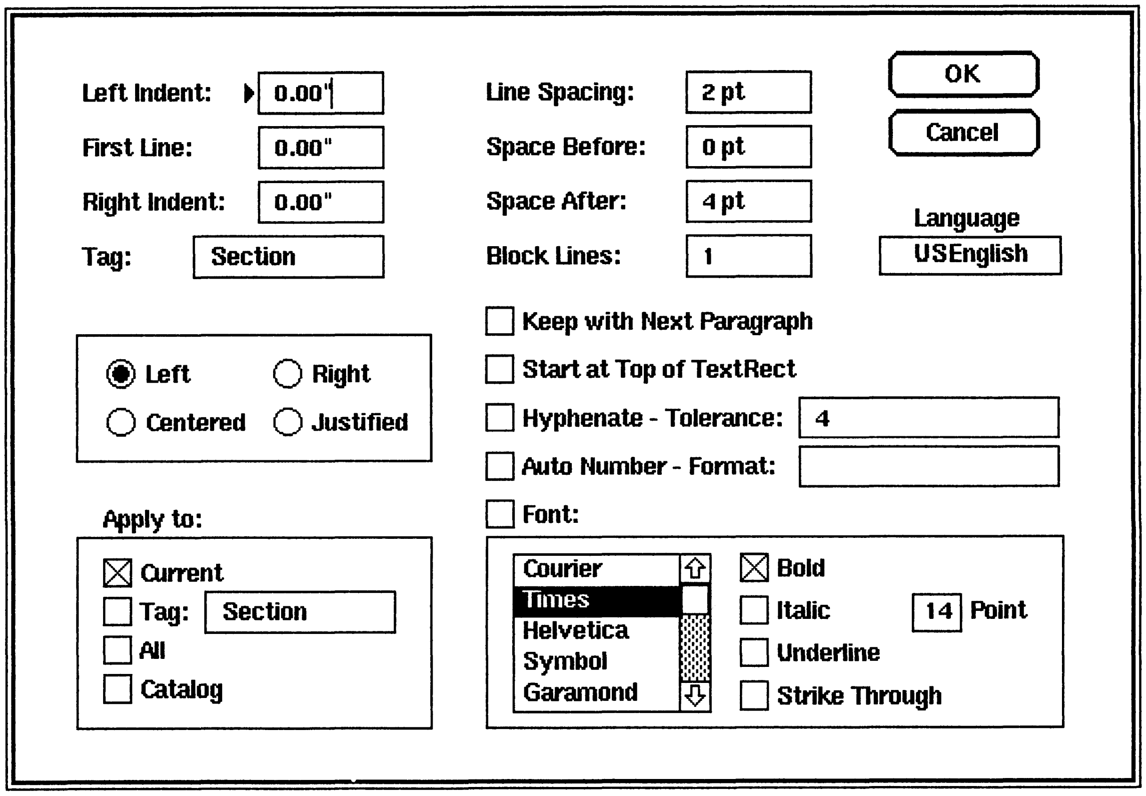
Location: Format menu and Maker menu

Standard Keyboard Equivalent: `Esc f p`

Purpose: Use the Paragraphs command to set formatting properties of one or more paragraphs and to store paragraph formats in the Catalog for later use with the Catalog command.

Use: To change the properties of a specific paragraph, put the insertion point anywhere within that paragraph and then use the Paragraphs command. You can also change the properties of several adjacent paragraphs by selecting them and using the command (all selected paragraphs are affected the same way by the command).

Choose the Paragraphs command to display the Paragraphs dialog box:


Left Indent: 0.00"
First Line: 0.00"
Right Indent: 0.00"
Tag: Section
Line Spacing: 2 pt
Space Before: 0 pt
Space After: 4 pt
Block Lines: 1
Language: USEnglish
[✓ Left] Right Centered Justified
Keep with Next Paragraph: ☐
Start at Top of TextRect: ☐
Hyphenate - Tolerance: ☐ 4
Auto Number - Format: ☐
Font: ☐ Times (Courier, Times, Helvetica, Symbol, Garamond)
Bold: ☒
Italic: ☐
Underline: ☐
Strike Through: ☐
14 Point
Apply to: ☒ Current ☐ Tag: Section ☐ All ☐ Catalog

When the dialog box first appears, it shows the settings of the paragraph containing the insertion point, if any, or the default paragraph settings (the settings given to the first paragraph of a TextRect drawn using the TextRect tool).

Left Indent: In this box, type a value to specify the left margin for all lines in the paragraph *except* the first line. This value is the distance from the left edge of the TextRect containing the paragraph.

First Line: In this box, type a value to specify the left margin for the first line of the paragraph. This value is the distance of the first line margin from the left edge of the TextRect containing the paragraph. The First Line can be greater than the Left Indent, for indented first lines, or less than the Left Indent, for "outdented" first lines (sometimes called *hanging indents*) such as paragraphs beginning with a bullet. It can also be the same as the Left Indent, for paragraphs like this one.

Right Indent: In this box, type a value to specify the right margin for all lines in the paragraph. This value is the distance of the margin from the right edge of the TextRect containing the paragraph.

Tag: In this box, type a name for this paragraph. If you apply the format to one or more paragraphs, then those paragraphs acquire the specified tag as their name. If you store the format in the Catalog, the format's Catalog name is the specified tag. The tag can be blank; we recommend, however, that you give each paragraph a tag and that you assign the same tag to all paragraphs that are of the same type.

We recommend that you do not use accented characters (such as ê, é, and è) in a tag name. Accented characters will be dropped from the tag name if you save the file in MIF (Maker Interchange Format). For more information about MIF, see Chapter 7.

Justification: Click one of these settings to specify left, right, centered, or justified (left and right) margin alignment. This paragraph uses the Left setting. See the on-line demonstration document (`Demo.doc`) for examples of other types of alignment.

Line Spacing: In this box, specify the amount of white space below each line of the paragraph (including the last line) in points (1 point = 1/72"). This spacing is often called *leading* (rhymes with "heading"). Values can range from 0 to 100 points. When you specify 0 line spacing, lines in the paragraph are packed as close together as possible (notice the distance between the bottom of the *q* on the second line and the top of the *h* in the third line):

Now is the time for the
quick brown fox to jump
the dog party.

Specify traditional double-spacing, as on a typewriter, by specifying a Line Spacing value that is equivalent to the prevailing point size of characters in the paragraph. That is, if the paragraph consists mostly of 12-point characters, use 12-point line spacing to achieve double-spacing. Most paragraphs on this page use a 12-point font with 2-point line spacing, often referred to as "12 on 14."

Space Before and **Space After:** Specifies the amount of white space, in addition to the white space provided by the line spacing, above the first line and below the final line of the paragraph. Values can range from 0 to 100 points. Add space between paragraphs rather than add empty paragraphs because adding space provides finer spacing control, makes it easier to change spacing at a later time, and requires less memory.

Block Lines: In this box, specify the minimum number of paragraph lines that must be kept in the same TextRect, both at the beginning and end of the paragraph. For example, a block size of 2 means that the first two lines of the paragraph must be kept together and that the last two lines of the paragraph must be kept together. This setting is a form of widow and orphan control. Use a large value (such as 100) to keep all lines of a paragraph on the same page.

Language: In this area, specify the language to be used when hyphenating and checking spelling for the paragraph. This setting allows you to include a paragraph in a different language from the other paragraphs in your document; FrameMaker will apply the appropriate hyphenation and spelling rules to each paragraph. If you're using the U.S. version of FrameMaker, only USEnglish is available. If you're using the International version of FrameMaker, you can choose from languages such as USEnglish, UKEnglish, Dutch, French, and German. Scroll through the available languages by clicking repeatedly in the Language box.

For information about setting the language of TextLines in your document, see the Spelling Checker command.

Keep with Next Paragraph: Turn on this setting to place the entire paragraph on the same page as the start of the following paragraph, if possible. If the paragraph would normally fit on a page, but the next paragraph would not, then both paragraphs are moved to the next page, leaving the first page with extra white space. This setting is another form of widow and orphan control.

Use this setting with paragraphs such as section titles or table headings. For example, all of the section headings in this document are single-line paragraphs that have the Keep with Next Paragraph setting checked.

Start at Top of TextRect: Turn on this setting to specify that the paragraph always starts at the top of a TextRect. Use this feature when you want the paragraph to start at the top of a column.

Checking this setting on an existing paragraph or applying a cataloged format with this setting turned on can produce slightly disorienting results on the screen, because the paragraph may jump to a new page.

Hyphenate and **Tolerance:** Turn on this setting and type a number in the edit box to specify whether or not the paragraph should be formatted using hyphenation, and if so, how many lines in a row may be hyphenated. For more complete information on hyphenation and tolerance, see the Auto Hyphenation command.

Auto Number and **Format:** Turn on this setting and type a string in the edit box to specify whether or not the paragraph begins with a FrameMaker-provided numbering string and, if so, the format for that string.

The basic idea of automatic paragraph numbering is that the number for one paragraph is derived by looking at the number of some previous paragraph within the same text flow.

The paragraph number can consist of one counter (for numbers such as 1, 2, 3, 4) or multiple counters (for numbering such as 1.1, 1.1.1, 1.1.2, 1.2). The value for each counter is determined by looking at the corresponding counter in the previous numbered paragraph and either using the same value or incrementing or decrementing the value. Consult the example below to see exactly how this works.

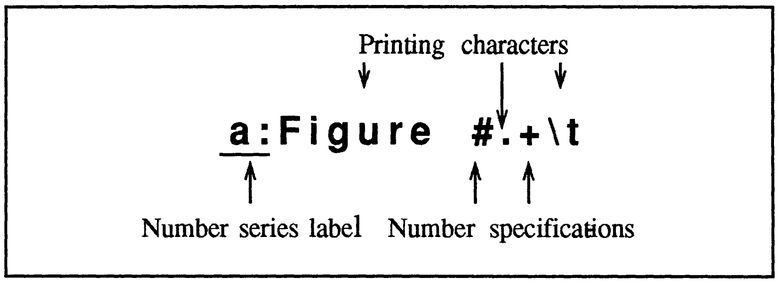
When you turn on the check box to the left of Auto Number, the paragraph is automatically numbered by FrameMaker. The number string is placed at the start of the paragraph.

Specify the format for the automatic number string in the edit box to the right of the Auto Number.

A complete automatic number string's format specification consists of:

- An optional number series label.
- Optional number specifications. A number specification is either a specific number (1, 33, or 128), a number sign (#), or a plus sign (+).
- Optional printing characters.

Here is a sample:



Anatomy of a Number Format Specification

Number series label: The number series label, if present, must be at the start of the format specification; it consists of any single printable character followed by a colon (:). A numbered paragraph's value is based on the number of the previous numbered paragraph *in the same flow* that has the same numbering series (see the example below). All format strings that do not begin with a series label are given a series value of 0 if they contain number specifications or a series value of 1 if they do not contain number specifications.

Number specifications: Each number specification sets the value of a counter. The number format can contain an essentially unlimited number of counters. For each counter:

- A specific number causes FrameMaker to print exactly that number and resets the value of the corresponding counter.
- A number sign (#) causes FrameMaker to print the value of the corresponding counter from the previous numbered paragraph in the same numbering series.
- A plus sign (+) causes FrameMaker to increment and then print the value of the corresponding counter from the previous numbered paragraph in the same numbering series (++ increments the counter twice).

Printing characters: Printing characters are all characters not part of the number series label or number specifications. They include space characters and may include `\t` to indicate a tab character. To accomplish formatting such as bulleted paragraphs, as shown in the example that follows, the format string can contain *only* printing characters.

Example: In the following paragraphs, the underlined text is the automatic number string FrameMaker creates. The bracket (>) represents a tab character.

Number Format String	Appearance in the Document
`2.\t`	2.> This is paragraph A.
`+\t`	3> This is paragraph B
`#.1\t`	3.1> This is paragraph C
`f:Figure +.`	Figure 1. This is paragraph D
`+.\t`	4.> This is paragraph E
`e:(+)`	(1) This is paragraph F
`#.+\t`	4.1> This is paragraph G
`•\t`	•> This is paragraph H
`•\t`	•> This is paragraph I
`•\t`	•> This is paragraph J
`#.+\t`	4.2> This is paragraph K
`f:Figure +.`	Figure 2. This is paragraph L
`#.#.+\t`	4.2.1>This is paragraph M
`#.#.+\t`	4.2.2>This is paragraph N
`e:(+)`	(2) This is paragraph O

Paragraph A	- Uses the default numbering series (series 0). Sets the first counter to 2, prints the 2, and then prints a period and a tab.
Paragraph B	- Uses the series 0. Increments the counter from 2 (in the previous series 0 paragraph) to 3, prints the 3, and then prints a tab.
Paragraph C	- Series 0 again. This format string has two counters. The # sign is counter one. It means print the same value that was printed for counter one in the previous series 0 paragraph. The 1 is counter two. It means set counter two to 1.
Paragraph D	- Uses numbering series f. Prints the word Figure and then a space. Since this is the first paragraph to use numbering series f, the + causes a value of 1 to be printed. The string ends with a period and a space.
Paragraph E	- Uses numbering series 0 again. Increments the value of counter one (the 3 in paragraph C becomes a 4).
Paragraph F	- Uses numbering series e (could stand for *equation*). Since this is the first paragraph to use numbering series e, the + causes a value of 1 to be printed.
Paragraph G	- Back to series 0. The # sign picks up the value 4 from paragraph E. The + increments the value of counter two, which was 0 in paragraph E, causing a 1 to be printed.
Paragraph H	- Contains no number specifications, so uses numbering series 1. Prints a bullet character and tab at the start of the paragraph.
Pgfs I & J	- Same as paragraph H.
Paragraph K	- Same as paragraph G.
Paragraph L	- Same as paragraph D. The + increments counter one in series f from 1 to 2.
Pgfs M & N	- These introduce a third counter to series 0 and pick up their values from paragraph K.
Paragraph O	- Same as paragraph F. The + increments counter one in series e from 1 to 2.

The tab and margin settings for a numbered paragraph are important. For example, the bullet formats in the example above (`•\t`) should have associated tab and margin settings to create the proper indenting. These settings would work well: Left Indent = .5", First Line = 0, and a left tab at .5".

To type a bullet, hold down the Alt key and type a period (Alt-period).

Font: Select or turn on the settings in the Font area to specify the font family, style, and size of the automatic number generated by FrameMaker, if any. These settings work like items in the Fonts dialog box.

If you click the check box to the left of Font, you *force the font* of the paragraph, setting the paragraph's *default font*. Forcing the font is extremely useful in conjunction with the Catalog. For example, you probably want section headings to be in one size and style with standard body text in another size and style. By checking the Font box and storing the appropriate formats in the Catalog, you can set both paragraph format and font in one step whenever you apply a Catalog item to one or more paragraphs.

When you apply a force font format to an existing paragraph, FrameMaker changes all the font attributes (family, size, and style) that match the original default font. For example, if you change a force font paragraph in which all the text is in the original default font, FrameMaker changes all the text in the paragraph to the new font.

If you have any font changes in the paragraph, however, FrameMaker retains those changes. For example, consider a force font paragraph with a default font of plain 12-point Times. The paragraph contains a bold 12-point Times word:

> This sample paragraph shows how **local** font changes are handled within force font paragraphs.

If you apply a new force font format to this paragraph, with a default font of plain 10-point Helvetica, the bold word switches to bold 10-point Helvetica:

> This sample paragraph shows how **local** font changes are handled within force font paragraphs.

Note that if the original paragraph isn't a force font paragraph, then FrameMaker doesn't retain font changes within the paragraph when you force a font.

To *remove* font changes within a paragraph, select the paragraph and choose the Font command from the Format menu. You can also apply the paragraph format with force font turned off and then apply it again with force font turned back on.

Apply To: Choose one or more of the settings in this area to specify what to do with the specified paragraph format:

- **Current:** Turn on this setting to apply the specified paragraph format to the paragraph containing the insertion point and to all paragraphs containing selected text.
- **Tag:** Turn on this setting to apply the specified paragraph format to all paragraphs whose tags match the tag specified in the edit box to the right of Tag in the Apply To area.
- **All:** Turn on this setting to apply the specified paragraph format to all paragraphs in the current document.
- **Catalog:** Turn on this setting to save the specified paragraph format in the Catalog for later use under the name indicated by the paragraph tag.

Notes:

- When you apply a paragraph format to a paragraph in the document, that paragraph's tag is changed to match the name in the Tag edit box in the upper-left corner of the Paragraphs dialog box (below Right Indent).
- Tab settings, while not displayed in the Paragraphs dialog box, are part of the paragraph format. When you apply the paragraph format settings to other paragraphs and to the Catalog as indicated in the Apply To area, the tab settings are also applied.

- When the Paragraphs dialog box first appears, it shows the format settings of the paragraph containing the insertion point, even if other paragraphs contain selected text.
- To uniformly change the format of a few scattered paragraphs, put the insertion point in the first paragraph you want to change, choose the Paragraphs command to display the Paragraphs dialog box, and fill in the new settings. Then click OK or press the Enter key. In addition to changing the selected paragraph's settings, you also store those settings in a memory buffer called the *Last Paragraph Settings*.

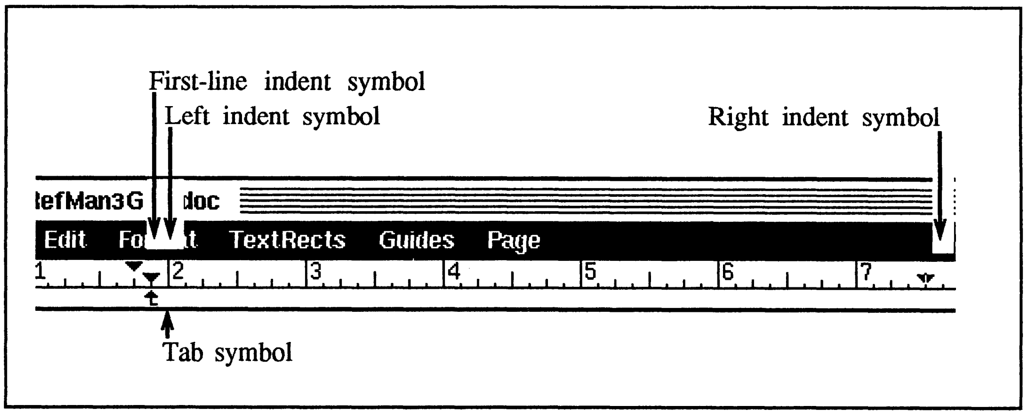
  Now, put the insertion point in a paragraph whose format you want to make match the first paragraph and press Esc j j (the *Paragraph Bang* command). This command applies the current Last Paragraph Settings to the paragraph containing the insertion point and to all paragraphs containing selected text.
- For more information on keyboard shortcuts for changing paragraph formatting, see *Paragraph Commands* in Appendix A.
- Also see the Copy Pgf Format and Catalog commands.

Paragraph Formatting with the Mouse

You can change the Left Indent, First Line, and Right Indent paragraph settings with the mouse. To do this, you must first display the rulers (see the Rulers command).

To change one of these settings using the mouse, follow these steps:

- Click the middle mouse button in the appropriate paragraph. Notice that symbols appear below the ruler at the top of the page to show the paragraph's current margin and tab settings:



Ruler Showing Margin and Tab Symbols

- Move the arrow pointer so that its tip is within the symbol for the setting being changed.
- Press and hold down the left mouse button.

- Move the arrow pointer left or right without releasing the mouse button to reposition the symbol as needed.
- Release the mouse button when the symbol is in the correct position.

Notes:

- You cannot move these symbols to positions outside of the enclosing TextRect's left or right sides.
- Each indent's exact position is displayed by the down-facing tip of the triangle.
- It is often useful to turn on the snap feature (using the Snap command from the Guides menu) before changing these settings with the mouse. When snap is on, these symbols "snap" to the nearest ruler division as the mouse is moved, making it easy to set margins and indents to exact locations using the mouse.

Paste

Location: Edit menu and Maker menu

Standard Keyboard Equivalent: `Esc e p` and `Ctrl-y`

Purpose: Use the Paste command to paste whatever is stored on the FrameMaker Clipboard into the current document at the insertion point. The Clipboard is an invisible storage area for objects, frames, text, font settings, and paragraph formats. All FrameMaker document windows share the same Clipboard, so you can cut or copy information, including paragraph format and font information, from one document and paste the information into another document.

Paste has no effect on the contents of the Clipboard, so you can paste the same information repeatedly.

Pasting text: When you put the insertion point in some text and choose the Paste command, the Clipboard text is inserted at the insertion point. If Clipboard text includes end-of-paragraph symbols, then the paragraph formats associated with those symbols are also brought into the document at the insertion point. Otherwise, the pasted text acquires the format of the paragraph containing the insertion point. The pasted text retains its font settings.

Pasting an object:

- If you paste, with nothing selected, onto a page that is the same size as the page from which the objects were cut or copied, the objects are pasted at their original coordinates on the current page.

  If you paste, with nothing selected, onto a page whose size is different from the size of the source page, the objects are pasted in the center of the current page.

  In these cases, the objects are pasted onto the page surface, not into any anchored frames on the page, even if an anchored frame happens to lie "under" the location where the objects are pasted. To paste an object into a frame, see below.
- If you paste with an object selected, the Clipboard objects are pasted into the page or frame containing the selected object, a bit below and to the right of the selected object.
- To paste objects into a frame, select the frame into which you want to paste the objects, and then choose the Paste command. The objects are pasted into the center of the frame.

Pasting font settings: First copy the desired font settings to the Clipboard using the Copy Font command. Then select text and use the Paste command. The selected text takes on the font settings from the Clipboard.

Pasting paragraph formats: First copy the desired paragraph format to the Clipboard using the Copy Pgf Format command. Then put the insertion point in a paragraph or select a block of paragraphs and use the Paste command. The paragraph containing the insertion point and all selected paragraphs take on the previously copied paragraph format.

Notes:

- See also the Cut, Copy, Copy Font, Copy Pgf Format and XPaste commands.

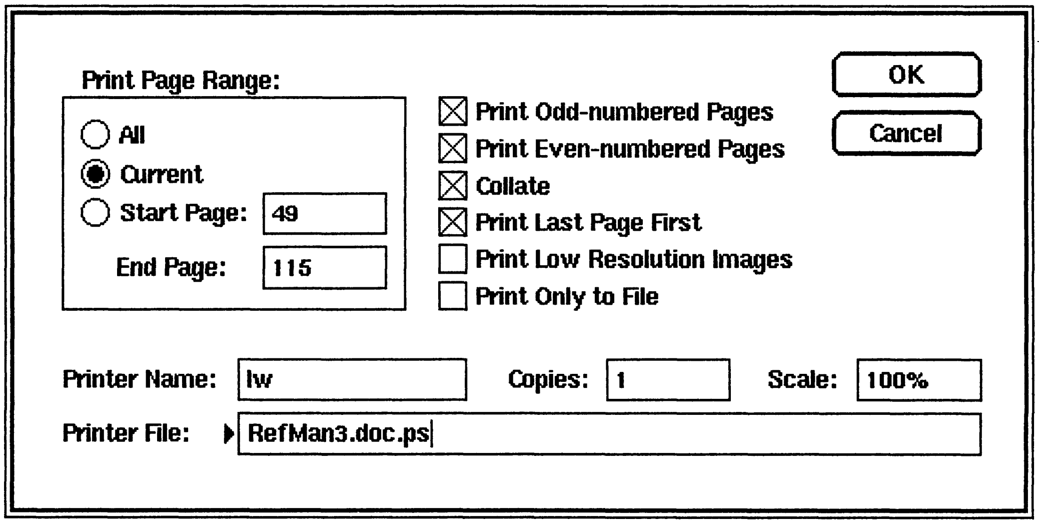
Print

Location: Document menu

Standard Keyboard Equivalent: `Esc d p`

Purpose: Use the Print command to print all or part of a document.

Use: Choose the Print command to display the Print dialog box:



Print Page Range:
( ) All
(●) Current
( ) Start Page: 49
End Page: 115

☒ Print Odd-numbered Pages
☒ Print Even-numbered Pages
☒ Collate
☒ Print Last Page First
☐ Print Low Resolution Images
☐ Print Only to File

Printer Name: lw
Copies: 1
Scale: 100%
Printer File: RefMan3.doc.ps

Print Page Range: Click one of the settings in this area to specify the document pages to print.

- **All:** Click All to print all pages.
- **Current:** Click Current to print the page that is visible when the Print command is invoked.
- **Start Page** and **End Page:** In the Start Page and End Page edit boxes, type the range of pages to be printed. The page numbers specified in Start and End Page are relative from the start of the document, as displayed in the headers and footers (see the Go To command for more information).

Print Odd-Numbered Pages and **Print Even-Numbered Pages:** Turn on these check boxes to specify the pages to be printed within the overall requested page range. These settings are ignored if you click Current in the Print Page Range area.

Normally you would turn on both of these settings to print both odd and even pages. You can, however, use these settings to print a document on both sides of the paper. To do this, give one command to print odd pages only (one side of the paper), reinsert the paper in the printer, and use a second command to print even pages only (the other side of the paper). First figure out how your printer feeds paper so that you do not end up overprinting or having pages appear upside down.

Collate: You may want to turn on this setting when you are printing multiple copies of multipage documents.

If Collate is turned on, all pages for the first copy are printed together, and then the process is repeated for each additional copy. You take the copies from the printer and use them in order, without having to shuffle papers (that is, collate them). While this setting is convenient, it also requires more printing time since each page has to be sent to the printer and imaged in the printer for each copy of the page.

If Collate is off, all copies of the first page are printed together, and then the process is repeated for each additional page. As a result, you need to collate the pages by hand. The document prints much more quickly with Collate turned off, however, because each individual page is sent to the printer once and imaged once. The additional copies of each page are printed at the fastest possible speed.

Print Last Page First: Turn this setting on if pages come out of your printer with the printed side facing up; by printing the last page first, the pages come out of the printer in the correct order. For printers with pages that come out with the printed side facing down, turn this setting off.

Print Low Resolution Images: Turn on this setting to print draft copies of a document. When you turn on this setting, imported images print faster but may not be as sharp as when this setting is turned off.

Print Only to File and **Printer File:** When you turn on Print Only to File, the document is not sent to your system printer. Instead, FrameMaker creates a print file and stores it in the file specified in the Printer File edit box. (A print file is a description of the FrameMaker document in a language specific to your printer. The type of print file FrameMaker creates depends on the printer FrameMaker is configured to use.)

Once you have a print file, you can edit it, convert it for use with other printers, or send it to a service bureau for phototypesetting. You can also send the print file to your printer using the *lp* command.

FrameMaker generates a temporary print file even when Print Only to File is turned off. FrameMaker tries to create that file in the `/tmp` directory. If it cannot create the file or if the file becomes larger than 1 megabyte, FrameMaker tries to create the file in your home directory. Then FrameMaker issues an *lp –dprintername* command.

Printer Name: In this edit box, specify the printer's device name. The default printer name is usually *ps* (for a PostScript printer). You can, however, change this default. See Appendix D, *Customizing FrameMaker,* for more information on printer names.

Copies: In this edit box, specify the number of copies to print.

Scale: In this box, type a number to specify a scaling factor for all printed pages; 100% is no scaling, 50% is half size, and 200% is twice the unscaled size. The printed image is printed at the lower-left corner of the paper for all scaling values. For example, if you want to enlarge something by 200%, draw it in the lower-left corner of a standard letter-size document. Arbitrarily large and small scale values are allowed, but all printing is cropped to the printable page area supported by the printer.

OK: Click OK to initiate printing as specified by the above items. FrameMaker prints in two phases. Phase one ties up FrameMaker for a few seconds (the pointer turns into an hourglass) and generates a background process. Phase two, controlled by the background process, builds the page description file (either PostScript or imPRESS) and sends the file to the printer queue.

Managing the Printer Queue

FrameMaker uses the *lp* facility to print, so print jobs go into *lp*'s printer queue. To list the jobs in the printer queue, use the *lpstat -t* command. This command lists print jobs in the order they are being processed and shows the request id, user name, and size of each print request.

To remove a job from the print queue, use the *cancel* command. Specifying a request id cancels the associated request even if it is currently printing. For example, to remove the request id lpt -1762, type this command:

```
cancel lpt-1762
```

Specifying a printer name cancels the request which is currently on the specified printer. For example, to remove the current job on printer lpt, type this command:

```
cancel lpt
```

Manual Paper Feed

If you are using a PostScript printer, you can use its manual feed mode when you want to print on odd-sized paper. For example, you may want to print on an envelope or odd-sized transparency. You may also want to insert special paper (like your company's letterhead) without removing and refilling the paper cassette. For more information about using your PostScript printer's manual feed mode, see your PostScript printer manual.

There are two ways to use the PostScript printer's manual feed mode with FrameMaker. You can:

- Change the print file for each document that you want to print in manual feed mode.
- Toggle FrameMaker's feed mode back and forth between manual and automatic paper feed, allowing you to print a number of documents in manual feed mode without changing individual print files.

To change the print file for a document so that it will print in manual feed mode:

- When you print the document, turn on the Print Only to File setting in the Print dialog box and specify an appropriate name for the print file.
- Add the lines shown in bold below to the PostScript print file you created in the step above.

```
%!
%%Pages: (atend)
%%DocumentFonts: (atend)
%%EndComments
%
% Frame Maker PostScript Prolog 1.3, for use with Maker 1.3
% Copyright (c) 1986,87,88 by Frame Technology, Inc. All rights reserved.
%
% Known Problems:
%  Due to bugs in Transcript, the 'PS-Adobe-' is omitted from line 1
/FMversion (1.2) def
/FrameDict 123 dict def
statusdict begin          <------------  Add lines in Bold
/manualfeed true def
end
% The readline in 23.0 doesn't recognize cr's as nl's on AppleTalk
FrameDict /tmprangecheck errordict /rangecheck get put
errordict /rangecheck {FrameDict /bug true put} put
FrameDict /bug false put
mark
```

If you use FrameMaker to edit the file, be sure to specify Text Only in the Save dialog when you're done.

- Use the UNIX command *lp* to print the PostScript file.

  The PostScript printer will pause before each page for you to feed pages manually.

To toggle FrameMaker's print mode back and forth between manual and automatic paper feed:

- Create your own `.makerinit` directory as follows:

```
% cd ~
% mkdir .makerinit
```

- Create files in your `.makerinit` directory as follows:

```
% cd .makerinit
% cp $FMHOME/.makerinit/postscript_prolog postscript_prolog.original
% cp postscript_prolog.original postscript_prolog.manual
```

- Edit the file `postscript_prolog.manual`. Make the same changes to it as shown above.

- Create a shell script like the one below (called `change`) to switch back and forth between automatic and manual feed.

```
#!/bin/csh
if ($1 == "man") then
echo "Frame Maker set to Manual Feed"
rm ~/.makerinit/postscript_prolog
ln -s ~/.makerinit/postscript_prolog.manual ~/.makerinit/postscript_prolog
else
echo "Frame Maker set to Auto Feed"
rm ~/.makerinit/postscript_prolog
ln -s ~/.makerinit/postscript_prolog.original ~/.makerinit/postscript_prolog
endif
```

- To make the script executable, type **`chmod +x change`**.

Whenever you want to change FrameMaker to manual paper feed mode, type **`change man`**; when you want to change FrameMaker back to automatic paper feed mode, type **`change`**.

Notes:

- Printing speed varies depending on page complexity, from about 14 seconds for each page of short, single-font text, to several minutes for each page containing large image files. Also, objects filled with gray-scale patterns print much faster than objects filled with bit patterns.
- Most printers cannot print along the outer 1/4" of each page, so if you place text or graphics very close to the edge of the page, they may not print.
- For information on dealing with printer problems, see your release notice.

Printer Code

Location: TextRects menu

Standard Keyboard Equivalent: `Esc t p`

Purpose: Use the Printer Code command to mark a TextRect so that FrameMaker knows it contains printer code (Printer Code TextRects can only contain PostScript). When such a TextRect is encountered while printing, its contents are sent directly to the printer as commands rather than formatted as text.

Printer Code TextRects give you access to the full power of the PostScript page description language. Use this feature to create special effects within FrameMaker. For example, you can create rotated text and text using very small or very large point sizes. Unlike imported Encapsulated PostScript (EPS) files, however, images created with the Printer Code command appear as printer code on the screen (see the Import command for more information on importing EPS files).

FrameMaker sets up the PostScript stack for use by the code in the TextRect: It pushes the TextRect's bottom-left page-relative coordinates, width, and height onto the PostScript stack before the imbedded PostScript code is stored in the PostScript file. FrameMaker also resets the origin (0,0 point) to the bottom-left corner of the TextRect. Your PostScript code is responsible for controlling its own scaling and clipping if needed.

Use: To set up a printer code TextRect, select the TextRect and choose the Printer Code command. If the TextRect you select is currently a printer code TextRect, a mark (for example, a checkmark or an *x*) appears next to the Printer Code command on the TextRects menu. Use the Printer Code command to turn the TextRect's printer code setting on or off as needed.

Notes:

- You can set only isolated TextRects as printer code TextRects.
- It is a good idea to print a document containing printer code TextRects to a file before sending the document directly to your printer. You can then check your PostScript code and see how FrameMaker sets up the stack before your code gets executed.
- PostScript errors within a printer code TextRect cause the rest of your document to fail to print.
- A printer code TextRect may reference external PostScript files, rather than putting all of the desired PostScript within the TextRect. To include an external PostScript file, place an include statement at the start of a line within the printer code TextRect as follows: `#include "filename"`. The double quotation characters are necessary. The filename should be a full path, to

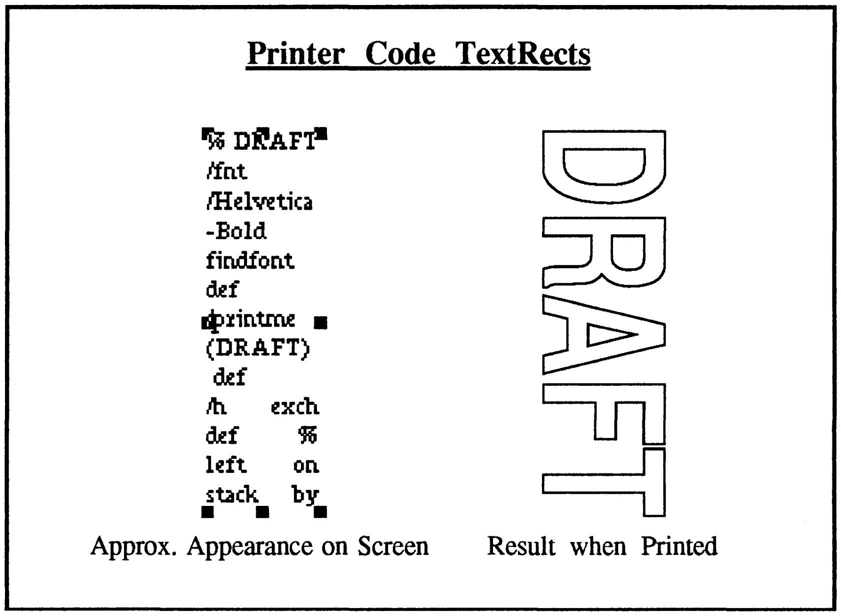
ensure that FrameMaker can find the include file. If it is not a full path, the include file must be in the directory where you start FrameMaker.

- You can use FrameMaker to generate PostScript files for use in PostScript TextRects. To do so, make a single-page FrameMaker document with the desired image positioned at the lower-left corner of the page. Use the Print command to print the document to a file. Edit the file to remove the PostScript prolog (delete all the lines up to and including the comment `%%EndProlog`). You can also edit the PostScript in other ways to produce special effects. Finally, save the resulting file and use it in a printer code TextRect via an `#include` statement (described in the previous bullet).
- For examples of PostScript (in TextRects) that produce large or rotated letters, see the on-line document `BigLetters.doc`. To open the document, click OPEN in the Main FrameMaker window, type the filename `BigLetters.doc` in the Open File Named edit box, and click OK.

Example 1:

The following PostScript code prints the word *DRAFT* in an outline font, rotated 90°. You can change the word *DRAFT* in line 3 to any other string. The string is scaled on printout to match the dimensions of the printer code TextRect:

```
% DRAFT
/fnt /Helvetica-Bold findfont def
/printme (DRAFT)  def
/h exch def % left on stack by Maker
/w exch def % ditto
/scaler 12 def % avoid PS rounding problems
fnt scaler scalefont setfont
/hsize h printme stringwidth
pop div scaler mul def
/wsize w (X) false charpath flattenpath
pathbbox /t exch def pop pop pop t
div scaler mul def
fnt [hsize 0 0 wsize 0 0] makefont setfont
newpath 0 h moveto -90 rotate
printme false charpath .5 setlinewidth stroke
```

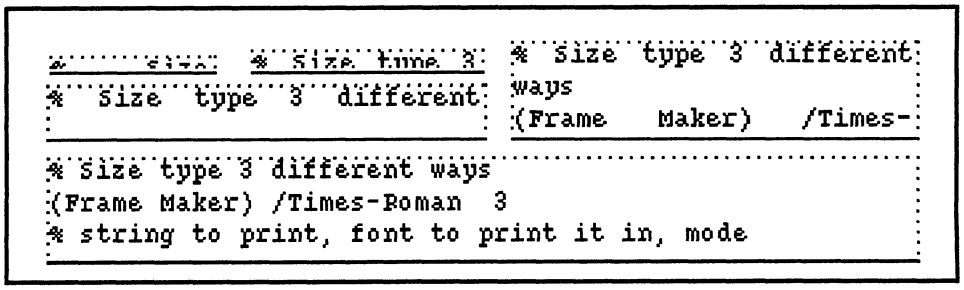


```
% DRAFT
/fnt
/Helvetica
-Bold
findfont
def
/printme
(DRAFT)
 def
/h    exch
def    %
left   on
stack  by
```

Example 2:

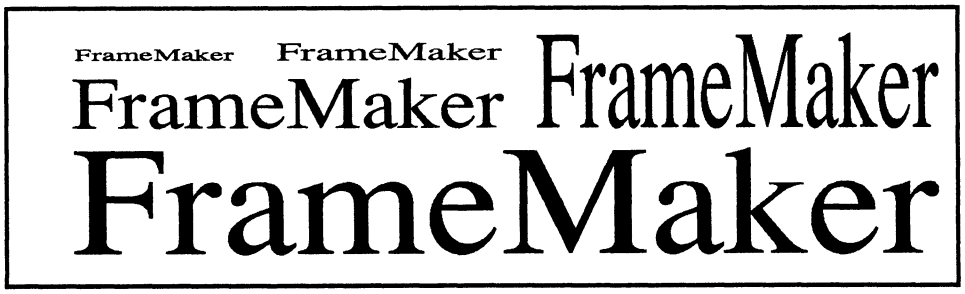
This PostScript code prints a string scaled to within its enclosing TextRect. The second line of code defines the string to print, the font to use, and the type of scaling to use, as defined in the comments on lines 3 through 6:

```
% Size type 3 different ways
(FrameMaker) /Times-Roman  3
% string to print, font to print it in, mode
% mode = 1 to fill width,
% mode = 2 to fill height
% mode = 3 to fill both
/mode exch def
findfont /fnt exch def
/printme exch def
/h exch def % left on stack by Maker
/w exch def % ditto
0 0 moveto
/scaler 12 def % avoid PS rounding problems
fnt scaler scalefont setfont
/wsize w printme stringwidth
pop div scaler mul def
/hsize h (X) false charpath flattenpath
pathbbox /t exch def pop pop pop t
div scaler mul def
mode 1 eq {/hsize wsize def} if
mode 2 eq {/wsize hsize def} if
fnt [wsize 0 0 hsize 0 0] makefont setfont
printme show
```



```
% Size type 3
% Size type 3 different    % Size type 3 different
                           ways
                           (Frame Maker) /Times-
% Size type 3 different ways
(Frame Maker) /Times-Roman  3
% string to print, font to print it in, mode
```

Approximate Screen Image



Printed Output

Quit

Location: Document menu

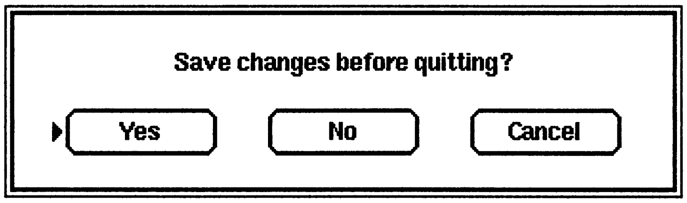
Standard Keyboard Equivalent: `Esc w q` and `Esc d q`

Purpose: Use the Quit command to remove windows from the screen (and from memory) and also to quit FrameMaker.

Use: To quit a document window, move the mouse pointer into the window you want to quit, and select the Quit command from the Document menu. To quit FrameMaker, point on "QUIT" in the main FrameMaker window and click any mouse button.

Quitting a Document Window

FrameMaker knows whether you've changed a document since the last time you saved it. If you haven't changed the document since saving it, FrameMaker removes the window from the screen and memory when you choose the Quit command. If you have changed the document, however, the Quit dialog box appears:



Save changes before quitting?

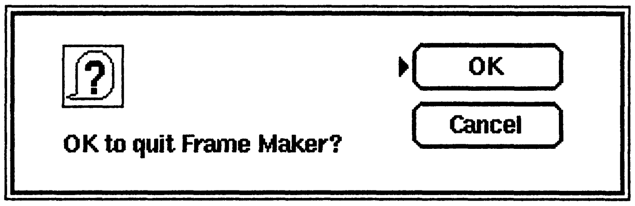
- **Yes:** Click Yes to display the Save dialog box (see the Save command). When you click OK or Cancel in the Save dialog box, FrameMaker removes the document from the screen and memory.
- **No:** Click No to remove the document window from the screen and from memory without saving the document.
- **Cancel:** Click Cancel to remove the Quit dialog box and to leave the document window on the screen.

Notes:

- Do not use window manager commands to destroy or kill a window.
- To close the Tools, Search, and Spelling Checker windows, use one of the keyboard Quit commands or use your window manager to iconify them.

Quitting FrameMaker

When you choose QUIT from the main FrameMaker window, FrameMaker checks to see if you have any unsaved work (including changes to documents and macros you've created without saving). If you don't have any unsaved work, FrameMaker asks you to confirm that you want to quit:


OK to quit Frame Maker?

Click OK to quit FrameMaker. Click Cancel to continue working with FrameMaker.

If you have unsaved work, FrameMaker displays the Quit dialog box (see the illustration on the previous page). When quitting FrameMaker, the buttons in the Quit dialog box have the following meanings:

- **Yes:** Click Yes to display the Save dialog box once for each open document you've changed without saving (see the Save command). When you click OK or Cancel in each successive Save dialog box, FrameMaker removes the corresponding document window from the screen and memory. After you quit all the document windows, FrameMaker removes any remaining FrameMaker windows from the screen and memory.
- **No:** Click No to remove all FrameMaker windows from the screen and memory without saving documents.
- **Cancel:** Click Cancel to remove the Quit dialog box from the screen and to leave all FrameMaker windows on the screen.

Record Keys

Location: Document menu

Standard Keyboard Equivalent: `Ctrl-]` and `Esc d r`

Purpose: The Record Keys command lets you assign a series of keystrokes to a shorter series of keystrokes in order to shorten repetitive keyboard tasks. The short key sequence that invokes a longer sequence is called a *trigger*, and the longer sequence is called a keyboard *macro.*

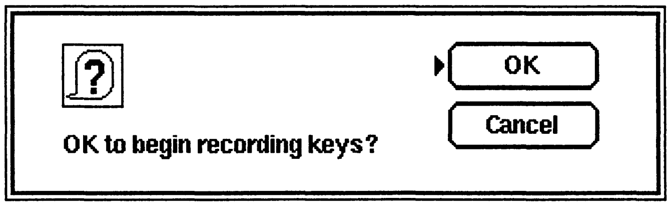
Keyboard macros can include text strings, editing commands, menu commands, and dialog box commands, so you can build anything from simple typing macros to powerful editing macros. For example, you could record a macro to display the Catalog, select a Catalog entry, and apply it to the current paragraph. Once this macro is defined, you can apply the same Catalog item to any paragraph by putting the insertion point in the paragraph and pressing the macro trigger key(s).

Keyboard macros can include keys that are themselves triggers for longer key sequences. For example, suppose you assign "FrameMaker" to Ctrl-1 and " is fun." to Ctrl-2. You can then assign Ctrl-1 Ctrl-2 to the key sequence Ctrl-3. When you press Ctrl-3, "FrameMaker is fun." is inserted. If you later change Ctrl-2 to " is powerful.", then pressing Ctrl-3 inserts "FrameMaker is powerful."

After you create a keyboard macro, you can store it in a macro file using the Keyboard command. When FrameMaker starts up, it looks for a file named `kbmacros` and automatically sets up your keyboard with macros in that file. You can also use the Keyboard command to remove macros that have been recorded and to read in customized macro files as they become appropriate for a certain editing job. See the Keyboard command and Appendix D, *Customizing FrameMaker,* for more information on these topics.

Use: Recording a keyboard macro has three steps: recording the contents of the macro (such as "FrameMaker" in the example above), recording the trigger (such as the key sequence Ctrl-1), and recording an optional comment.

- **Step 1. Recording the contents of the macro:** Choose Record Keys from the Document menu to display the first Record Keys dialog box:

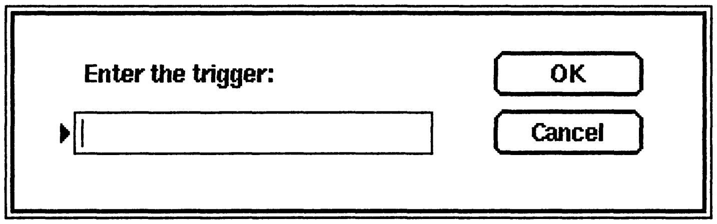

OK to begin recording keys?

Click OK to begin recording the body of the macro, or click Cancel if you do not want to record keys.

After you click OK, you can record up to 150 keystrokes. (FrameMaker interprets a single keystroke as a sequence of 1, 2, or 3 keystrokes in a row.) As you press each key, its normal function will be carried out, and it will also be recorded. You can verify that keys are being recorded by looking at the Document menu: A mark (for example, a checkmark or an *x*) appears next to the Record Keys command when keys are being recorded.

If you press an incorrect key while recording, stop recording, as explained in step 2 below, and start over.

- **Step 2. Recording the trigger:** When you are done recording keys, choose the Record Keys command from the Document menu or press Ctrl-] to turn off recording and to display the Enter Trigger dialog box:

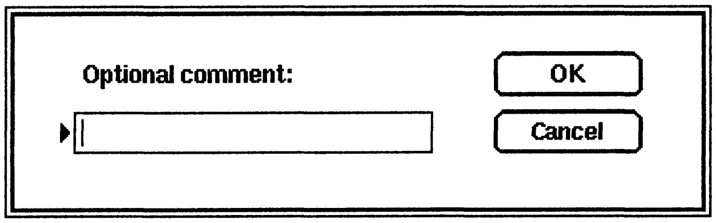


To enter the trigger, press a function key or other keys to define the macro trigger. As you press each key, one or more characters appear in the trigger edit box. FrameMaker interprets keys as a sequence of 1, 2, or 3 characters in a row. Unprintable characters are shown in the dialog box using the asterisk character (*). The trigger can contain up to 15 characters (remember that each key can represent more than 1 character).

The Enter key and Ctrl-c can be part of a trigger, so pressing these keys is not equivalent to clicking OK or Cancel. However, the Delete and Backspace keys can be used to remove characters from the end of the trigger (again, if you press the wrong key, you must delete all of its corresponding characters).

When you are done entering the trigger, do not press Enter or Ctrl-c (these keys would just be added to the trigger). Instead, use the mouse to click OK or Cancel. You can also press Ctrl-], which is equivalent to clicking OK (only in this dialog box).

- **Step 3. Recording an optional comment:** After you've entered the trigger, the Comment dialog box appears:



The comment is useful if you use the Keyboard command to store the macro in a file for later use: The macro itself is stored in a somewhat obtuse format (easy for a computer to understand, but difficult for people!). The comment will be placed in the file before the macro, so that you can look at the file to find out what macros it contains.

To enter an optional comment, type a description for the macro you have recorded, such as `Ctrl-1 : Frame Maker`. In this dialog box, pressing Enter and Ctrl-c have their standard meanings of OK and Cancel.

Do not type angle brackets (< or >) in your optional comments; FrameMaker cannot read these symbols in a macro file.

OK: Click OK to store the macro. You can now use the macro by putting the insertion point at the appropriate spot (if it is not already there) and pressing the trigger key(s).

Cancel: Click Cancel to remove the Comment dialog box from the screen. No keyboard macro is created, and key recording is no longer active.

Notes:

- When you record a macro, a copy of it is written to a file called `kblog` in your home directory. You can look at `kblog` to see how macros are represented and to edit them. Note that `kblog` gets overwritten every time FrameMaker is started up. Use the Keyboard command to save your macros to a more permanent file.
- Beware of disabling already used key sequences. (For example, if you use Ctrl-y as a macro trigger, you disable the Ctrl-y Paste command.) You may, however, want to use the same trigger for two macros if you want to improve a macro or delete it; if two or more macros have the same trigger, FrameMaker recognizes only the most recently recorded one.
- While recording macros that work with dialog boxes, it is important to use keys that set dialog box settings to specific values, as opposed to using keys that toggle the state of settings, so that you can always be sure of the selected value.

Example:

Here's an example showing the proper style for recording macros that drive dialog boxes. Don't be alarmed by its apparent length; soon you'll be able to set up complicated macros very quickly. (See *Using Dialog Boxes* in Chapter 2.)

Suppose you want to record a macro so that pressing F8 1† causes the font of the word containing the insertion point to be set to Courier 12 bold. Place the insertion point within a word and use these keystrokes:

Keystrokes	Explanation
`Ctrl-]`	Choose the Record Keys command
`Enter`	Respond to "OK to begin recording keys?" dialog box
`Esc h w`	Highlight the word
`Esc f f`	Display the Fonts dialog box
`Alt-Tab`	Make sure dialog's keyboard focus is at the Font Family
`Alt-Up arrow`	Select the first item in the Font Family scroll list
`Tab`	Move to Current radio button
`1`	Force Current to be on
`Tab` 4 times	Move to Style check box
`1`	Force Style to be on
`Tab`	Move to Bold check box
`1`	Force Bold to be on
`Tab 0`	Force Italic to be off
`Tab 0`	Force Underline to be off
`Tab 0`	Force Strike Through to be off
`Tab 1`	Force Position to be on
`Tab 1`	Force Normal to be on
`Tab` 3 times	Move to Keep Settings check box
`0`	Force Keep Settings to be off
`Tab 1`	Force Size to be on
`Tab Alt-Up arrow`	Move to Point scroll box and force it to 1st setting (7 pt)
`Space` 4 times	Advance scroll box to 5th setting (12 pt)
`Tab 1`	Force Spread Pts to be on
`Ctrl-u`	Erase whatever Spread Pts value is there
`0`	Fill in a Spread Pts value of 0
`Enter`	Apply the new font setting
`Ctrl-]`	End recording and bring up the Trigger dialog box
`F8 1`	Press the F8 function key followed by the 1 key to assign this to the sequence F8 1
`Ctrl-]`	End recording the trigger; display Comments dialog box
`F8 1:Courier 12 Bold`	Enter comment text
`Enter`	Done

† That's the `F8` key followed by the `1` key. You could do this task using the keyboard instead of the Fonts dialog, but we wanted to show you how to record macros that use dialog boxes.

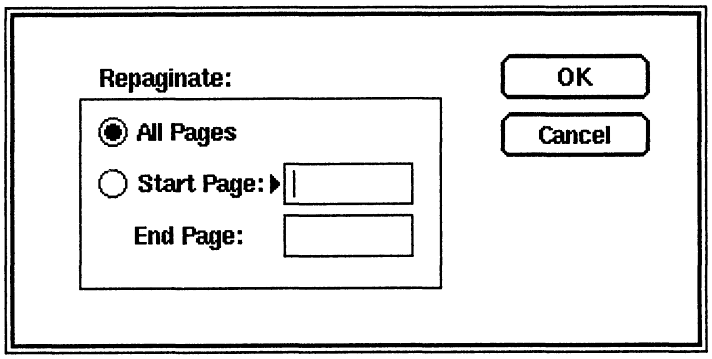
Repaginate

Location: Format menu

Standard Keyboard Equivalent: `Esc f r`

Purpose: Use the Repaginate command to recompute page breaks across two or more pages of a frozen document. See the Freeze Pagination command for a description of frozen pagination.

Use: Choose the Repaginate command to display the Repaginate dialog box:

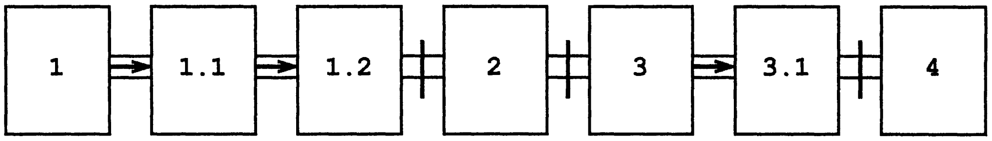


All Pages: Click All Pages to repaginate all pages. This is equivalent to filling in the first and last page numbers in the Start Page and End Page edit boxes.

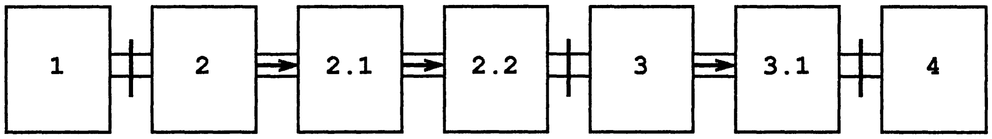
Start Page and **End Page:** In this box, fill in the first and last page numbers to specify a range of pages to repaginate. The numbers refer to *main page* numbers, as opposed to *point page* numbers, and are relative page numbers (in other words, if the first page of the document is page 21, and you want to repaginate the first five pages of the document, you type `21` as the first page and `25` as the last page). All pages starting at the Start Page and continuing through the End Page and its extensions are repaginated.

To repaginate a range of pages, FrameMaker opens the text flow between those pages, allows the text to flow within those pages, and then freezes the text flow between the resulting main pages again. Pages outside of the repaginated area are not affected:

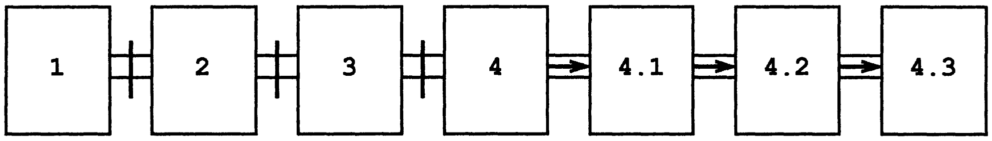
Before repaginating:



After repaginating pages 1 through 2:



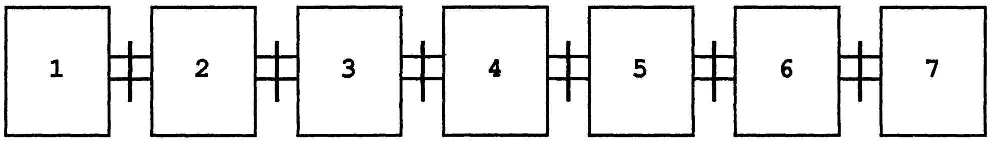
After repaginating all:



Notes:

- Instead of the Repaginate All setting, you can use the Freeze Pagination command to unfreeze pagination. If you then freeze pagination again, you get the following page configuration:

After Unfreeze Pagination followed by Freeze Pagination:



FrameMaker provides different functions for the Repaginate All and Freeze/Unfreeze commands so that you can achieve both pagination effects.

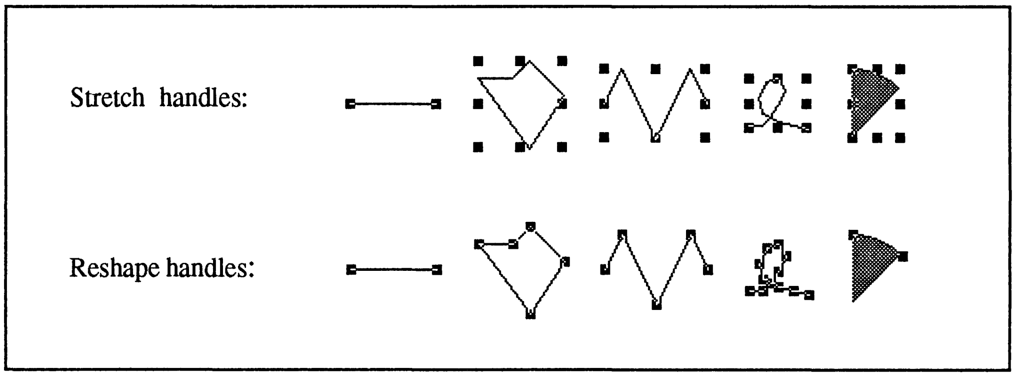
- After you have repaginated a range, you may find extra pages. You can delete these pages using the Delete Page command.

Reshape

Location: Tools window

Standard Keyboard Equivalent: `Esc o r` (the letter o as in *object*)

Purpose: Use the Reshape command to reshape polygons, polylines, lines, arcs, and freehand shapes. This command replaces stretch handles with reshape handles.



Stretch and Reshape Handles

Use: To reshape an object, select it with the left mouse button and choose the Reshape command from the Tools window. The object's handles change from stretch handles to reshape handles. The object is still selected, so other commands, such as the Cut command or clicking a fill pattern, still affect the object while it has reshape handles.

Once the object has reshape handles, you can move the points that define its shape, and you can add or delete points to alter the shapes more. (You cannot, however, add points to or delete them from arcs.)

- To move a point, drag the point using the left mouse button.
- To add a point, move the arrow pointer to a position along the object's border, hold down the Ctrl and Shift keys, and click the right mouse button. A new point appears at the position of the arrow pointer. You can then move the new point like any other point.
- To delete a point, move the arrow pointer so it's touching the point you want to delete, hold down the Shift and Ctrl keys, and click the left mouse button.

Notes:

- Reshaping an arc changes the arc start or end angle (depending on which handle is dragged).
- See *Graphics Editing Overview* in Chapter 2 for more information.

Rulers

Location: Guides menu

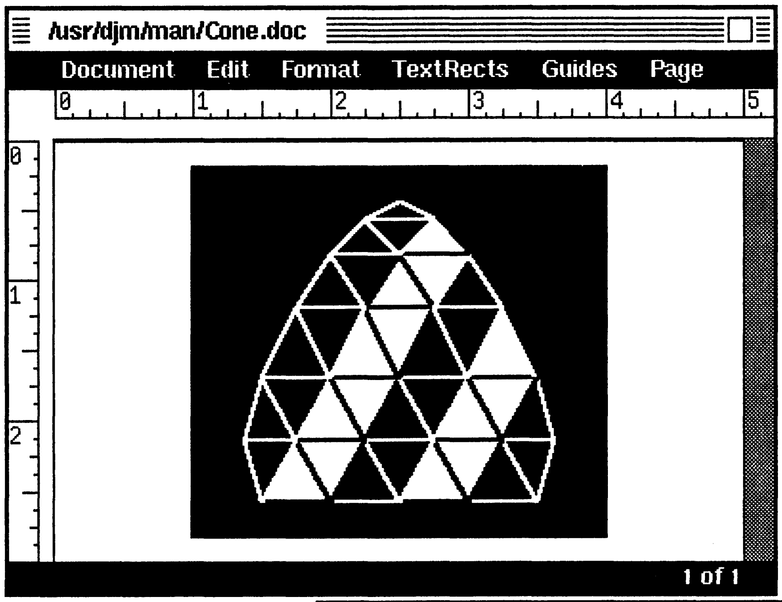
Standard Keyboard Equivalent: `Esc g r`

Purpose: Use the Rulers command to turn on and off the display of top and side rulers.

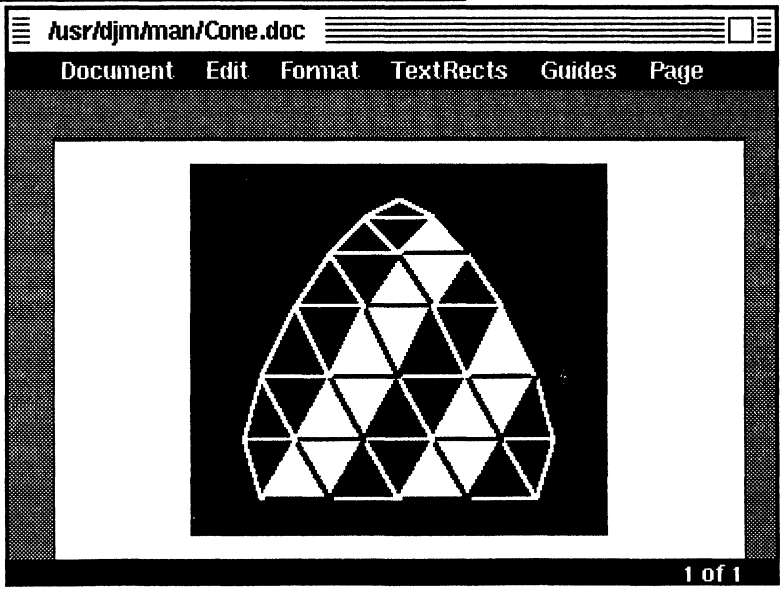
Use: When rulers are visible, a mark (for example, a checkmark or an *x*) appears next to the Ruler command in the Guides menu.

Notes:

- You can change the spacing of ruler divisions using the Units command from the Guides menu.



Rulers on



Rulers off

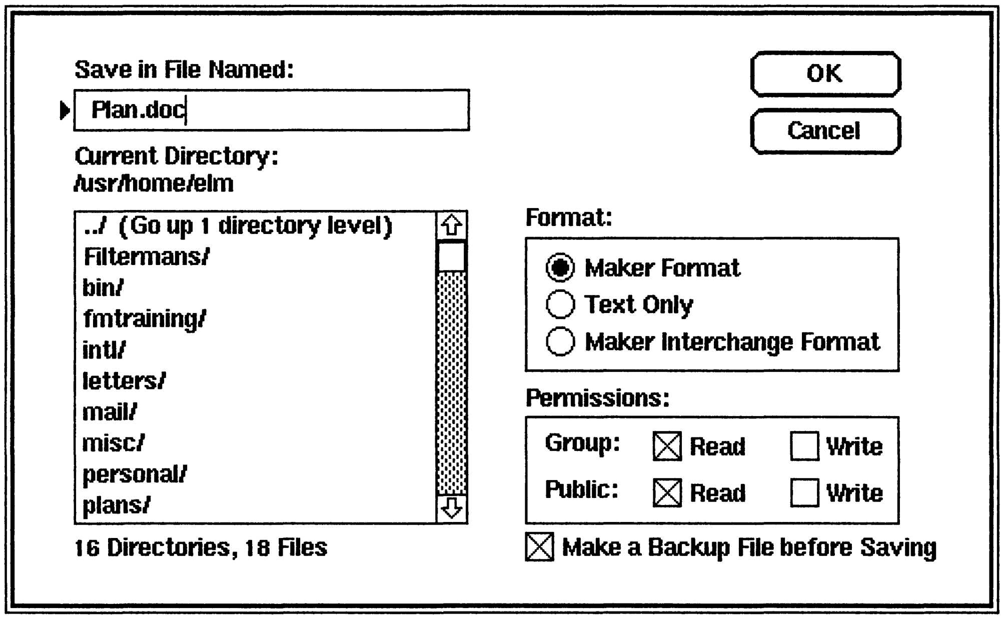
Save

Location: Document menu

Standard Keyboard Equivalent: `Esc d s`

Purpose: Use the Save command to write a copy of the document to a disk file.

Use: Choose the Save command to display the Save dialog box:



Save in File Named: Plan.doc

Current Directory: /usr/home/elm

../ (Go up 1 directory level)
Filtermans/
bin/
fmtraining/
intl/
letters/
mail/
misc/
personal/
plans/

16 Directories, 18 Files

Format: [✓ Maker Format] Text Only Maker Interchange Format

Permissions:

Group:	[✓ Read]	Write
Public:	[✓ Read]	Write

[✓ Make a Backup File before Saving]

Save in File Named: In this box, type the name of the file FrameMaker is to save. When you choose the Save command, the edit box contains the current name of the document, but you can use the scroll list to select another filename in the current directory or any directory you specify. The Save scroll list initially lists all files in the document's current directory. If you are saving a new document that has not been saved before, the scroll list lists the contents of the last directory in which you opened a document.

To save the file, you must have write access to the specified directory and file.

Permissions: Turn on one or more of these settings to specify the read and write access provided to others in the user's group and to all other users.

Format: Click one of these settings to specify the format in which to save the file.

- **Maker Format:** Click Maker Format to specify the standard format for FrameMaker documents; this format is the most compact and requires the least amount of time to save and open.
- **Text Only:** Click Text Only to write all the text from TextRects only (this format does not write TextLines). Each end-of-line in the FrameMaker document, including line endings within paragraphs, is written as a "new line"

character in the text file; because of this, you should reformat the text so that the line breaks occur exactly where you want them before saving the file. To reformat the text, either widen the TextRect or make the font size smaller.

- **Maker Interchange Format:** See Chapter 7 for an explanation of Maker Interchange Format.

Make a Backup File before Saving: Turn on this setting to make a backup of the file being saved by renaming the old file as *filename*`.backup` before saving the new version.

OK: Click OK to save the document in the specified file. If the file can be written, the arrow pointer changes to an hourglass while the document is being saved; after a few moments, the arrow pointer reappears, indicating that the document was saved, and the name at the top of the document window is updated to match the specified filename. If the file cannot be written, an error message is displayed.

Notes:

- The dis[illegible]ace required to store documents using the Maker Format setting varies depending on page size and content. The following table shows various file sizes for several chapters in this reference manual and for `Demo.doc` (final versions of these documents may have different sizes):

Document	# Pages	File Size	Bytes/Page
Ref Man Ch. 1	2	6976	3488
Ref Man Ch. 2	36	175252	4868
Ref Man Ch. 3	130	551031	4238
Ref Man Ch. 4	16	82988	5186
Ref Man Ch. 5	10	30684	3068
Ref Man Ch. 6	48	293360	6111
Ref Man Ch. 7	20	93377	4666
Ref Man Ch. 8	26	129264	4971
Demo.doc	32	267002	8343

  Chapters 3 through 8 represent typical sizes for technical documentation. Chapter 3 (the one you are reading) requires fewer bytes per page because many of its pages are not full. Chapter 4 requires more bytes per page because it contains many short paragraphs. And finally, `Demo.doc` requires the most bytes per page because each page has its own layout and because it contains many short paragraphs.

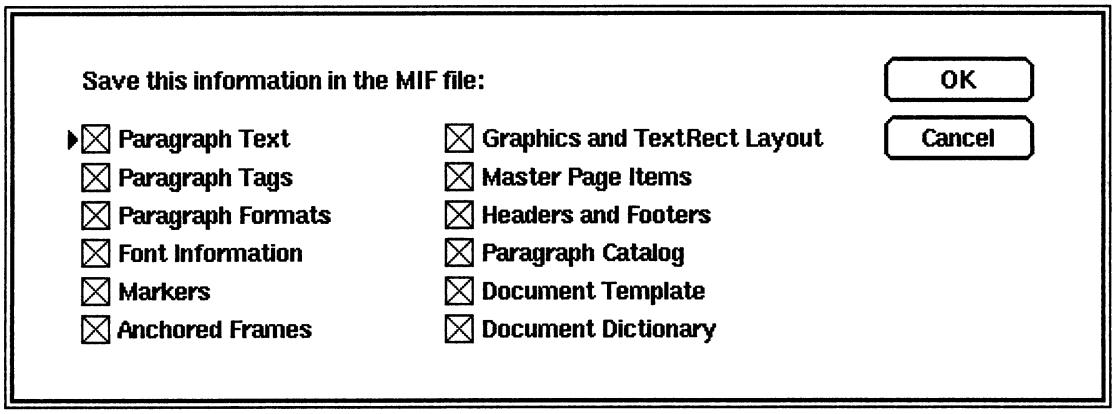
  The figures in the table above show the actual document file sizes. They do *not* include the space required for imported image files, which are stored external to the document files. All of the image files for this reference manual use approximately 1 megabyte of storage.

- Before saving "on top of" an existing file, FrameMaker always renames the existing file by adding `.backup.tmp` to its name. After FrameMaker successfully writes the new file, if the Make a Backup File before Saving setting is turned on, FrameMaker renames the `.backup.tmp` file by removing the `.tmp`. If no backup has been requested, FrameMaker deletes the `.backup.tmp` file.

If for some reason FrameMaker or your system should crash while your file is being saved, you can recover the most recent usable version by using the `.backup.tmp` file.

Notes on Saving MIF Files

When you request MIF format in the Save dialog box and click OK, the MIF Save dialog box appears, allowing you to specify what gets saved in the MIF file:



Save this information in the MIF file:

☒ Paragraph Text
☒ Paragraph Tags
☒ Paragraph Formats
☒ Font Information
☒ Markers
☒ Anchored Frames
☒ Graphics and TextRect Layout
☒ Master Page Items
☒ Headers and Footers
☒ Paragraph Catalog
☒ Document Template
☒ Document Dictionary

The settings you choose vary depending on what you intend to do with the MIF file.

- To export the document's Catalog (as described in the Import command), turn off all settings except Paragraph Catalog.
- To save the document for merging into another FrameMaker file, turn on all settings *except* Master Page Items, Headers and Footers, and Document Template. It does no harm to leave these settings turned on, but they are unnecessary and take up extra space in the MIF file.
- If the file has no graphics except those in anchored frames and you want to save it for merging into another FrameMaker document, turn on only the left column settings (and perhaps the Paragraph Catalog if you want). By leaving out the graphics and TextRect layout objects, you can import the resulting MIF file into a document using different column and margin settings.
- If the format of all paragraphs in the document matches the format descriptions in the document's Paragraph Catalog, you can save considerable space in the MIF file by turning off the Paragraph Formats setting; Paragraph Tags allows each paragraph's format to be looked up in the Paragraph Catalog.

The following table lists and explains the categories of information that can be saved in a MIF file.

This MIF category:	Saves this information:
Paragraph Text	All text that appears in TextRects.
Paragraph Tags	The tag names used in the document, in the order in which they occur.
Paragraph Formats	The paragraph formats for each paragraph in the document, in the order in which they occur.
Font Information	The font information for all the text in the document.
Markers	The marker information that was entered in the Marker dialog box, plus the page number on which the marker appears.
Anchored Frames	The information that was entered in the Anchored Frame dialog box; the position of the frame; the name, position, and attributes of any objects inside the frame; the text and position of any TextLines in the frame; the filename of imported raster files and their position.
Graphics and TextRect Layout	For each page, the page type, number, and size; the position and attributes of each TextRect, non-anchored frame, object, and TextLine.
Master Page Items	For the Master Page, the page number format, page size, text margins, number of columns, the position and attributes of objects, and the position and text of TextLines.
Headers and Footers	The header and footer text that was entered in the Headers & Footers dialog box, along with the header and footer margins.
Paragraph Catalog	All the paragraph formats stored in the Catalog.
Document Template	The information entered in the Make a Custom Document dialog box, whether the header and footer start on the first page, the page layout, whether pagination is frozen, and whether the first page starts on the right or left.
Document Dictionary	A list of the words in the document's dictionary.

Scale

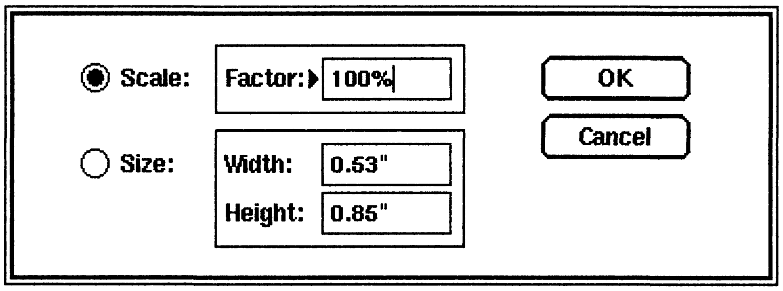
Location: Tools window.

Standard Keyboard Equivalent: `Esc o z`

Purpose: Use the Scale command to change the size of a selected object or group by indicating numeric proportions rather than by stretching it with the mouse.

Use: To use the Scale command, follow these steps:

- Select a *single* object or group.
- Click Scale in the Tools window to display the Scale dialog box:



[✓ Scale:] Factor: 100%

Size: Width: 0.53" Height: 0.85"

Scale and **Factor:** In the Factor box, type the scaling factor for the object. A scaling factor of 100% represents the object's original size; a scaling factor of 50% shrinks the object to half its size; and a scaling factor of 200% doubles the object's size.

Size, **Width**, and **Height:** In these boxes, instead of indicating a scaling factor, you can specify the exact width and height of the object. When the Scale dialog box appears, these boxes show the object's current width and height. The units used for width and height (inch, centimeter, pica, or point) reflect the Display Units setting in the Units dialog box (see the Units command).

Notes:

- You cannot scale TextLines.
- If the numbers you specify in the Scale command would cause the objects to be moved off the page, FrameMaker beeps and leaves the objects unscaled.
- Since object dimensions are stored in FrameMaker in points, which do not divide exactly into 100ths of an inch or 100ths of a centimeter, certain dimensions cause a rounding error to occur.

Scroll

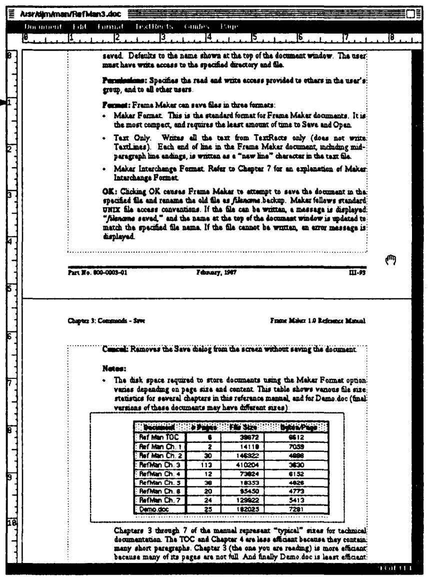
Location: Page menu

Standard Keyboard Equivalent: `Esc p s`

Purpose: Use the Scroll command to view portions of adjacent pages simultaneously in the document window. Use this command to select text across pages and to display the bottoms of pages longer than 11.25".

Use: When you choose the Scroll command, the arrow pointer becomes a scroll pointer (🖐).

To scroll, press and hold down the *left* mouse button and move the scroll pointer either up or down. To display part of the previous page, move the scroll pointer down. To display part of the next page, move the scroll pointer up. To see the scrolled pages, release the mouse button:


Page border (visible only when borders are turned on)
Scroll pointer
/usr/djm/man/RefMan3.doc
Document Edit Format TextRects Guides Page
93 of 113

saved. Defaults to the name shown at the top of the document window. The user must have write access to the specified directory and file.

Permissions: Specifies the read and write access provided to others in the user's group, and to all other users.

Format: Frame Maker can save files in three formats:

- Maker Format. This is the standard format for Frame Maker documents. It is the most compact, and requires the least amount of time to Save and Open.
- Text Only. Writes all the text from TextRects only (does not write TextLines). Each end of line in the Frame Maker document, including mid-paragraph line endings, is written as a "new line" character in the text file.
- Maker Interchange Format. Refer to Chapter 7 for an explanation of Maker Interchange Format.

OK: Clicking OK causes Frame Maker to attempt to save the document in the specified file and rename the old file as *filename*.backup. Maker follows standard UNIX file access conventions. If the file can be written, a message is displayed: "*filename* saved," and the name at the top of the document window is updated to match the specified file name. If the file cannot be written, an error message is displayed.

Part No. 800-0003-01 February, 1987 III-93

Chapter 3: Commands - Save Frame Maker 1.0 Reference Manual

Cancel: Removes the Save dialog from the screen without saving the document.

Notes:

- The disk space required to store documents using the Maker Format option varies depending on page size and content. This table shows various file size statistics for several chapters in this reference manual, and for Demo.doc (final versions of these documents may have different sizes):

Document	# Pages	File Size	Bytes/Page
Ref Man TOC	6	39672	6612
Ref Man Ch. 1	2	14118	7059
Ref Man Ch. 2	30	146922	4896
RefMan Ch. 3	113	410204	3630
RefMan Ch. 4	12	73824	6152
RefMan Ch. 5	38	18353	4826
RefMan Ch. 6	20	95450	4773
RefMan Ch. 7	24	129922	5413
Demo.doc	25	182025	7281

Chapters 3 through 7 of the manual represent "typical" sizes for technical documentation. The TOC and Chapter 4 are less efficient because they contain many short paragraphs. Chapter 3 (the one you are reading) is more efficient because many of its pages are not full. And finally Demo.doc is least efficient

To get out of scroll mode (and return to the arrow pointer), click the middle mouse button within the document window. The window remains scrolled, and you can use any FrameMaker functions on the text.

To reset the window so that only one page is displayed, choose the Next, Previous, or Go To command from the Page menu.

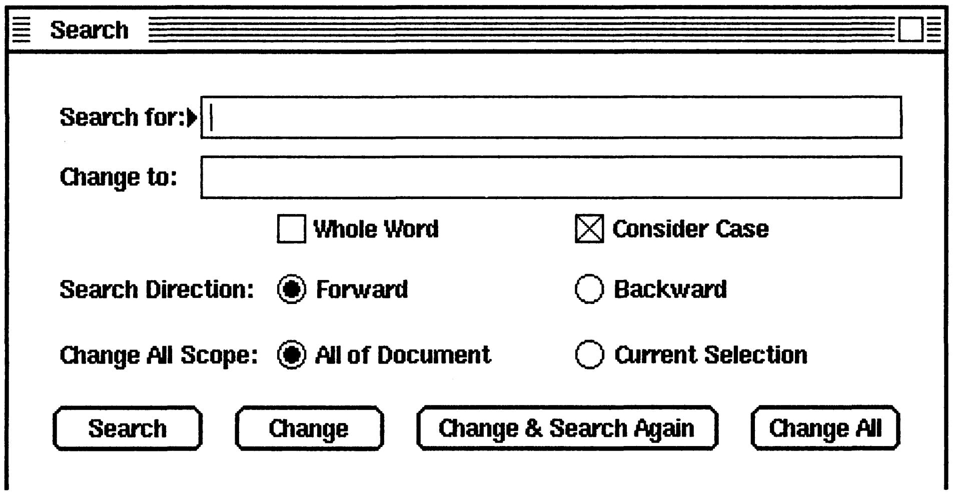
Search

Location: Edit menu

Standard Keyboard Equivalent: `Esc e s`

Purpose: Use the Search command to search through document text for a string of characters and, optionally, to replace the found characters with different characters.

Use: Choose the Search command to display the Search window.



Search for:

Change to:

Whole Word [✓ Consider Case]

Search Direction: [✓ Forward] Backward

Change All Scope: [✓ All of Document] Current Selection

Unlike a standard dialog box, the Search window can remain on the screen while you work within a document or shell window. Once this window is on the screen, you can manipulate it as you do FrameMaker document windows. For example, you might want to move the window so it doesn't overlap your document window(s). See your window system documentation for information on manipulating windows.

To use FrameMaker's search functions, fill in the Search window, set the insertion point in a document, and click one of the four buttons at the bottom of the Search window.

Search For, **Whole Word**, and **Consider Case:** In the Search For edit box, type the string for which to search. Turn on Consider Case to specify that differences between uppercase and lowercase letters are significant. When Consider Case is turned on, *Bill* is considered to be different from *bill.* Turn on Whole Word to prevent substrings from being found. When Whole Word is turned off, the string *alt* is found in *alternative, malt, maltese,* and *alt.*

The Search For string can contain leading, embedded, and trailing spaces, all of which are significant. An empty Search For string is not meaningful.

You can include special character sequences in the Search For string to allow searching for formatting characters such as end-of-paragraph, Alt-Enter, and Tab:

Character String	Character Searched For
\t	Tab
\h	Alt-Enter (hard return)
\n	Alt-Enter (same as \h)
\r	Alt-Enter (same as \h)
\p	Enter (end-of-paragraph)
\a	Anchored frame's anchor symbol
\m	Marker symbol
\s	Space (equivalent to typing a space char)
\\	Backslash (\ )
\space	Hard space
\-	Discretionary hyphen
_	Don't hyphenate
\other	Other (that is, \x searches for *x*)

You can also include standard regular expression characters (such as those used in the UNIX *grep* utility) in the Search For string :

Character String	Character Searched For
^	Matches the beginning of a line
[ab]	Matches any one of the characters in the brackets
[^ab]	Matches any one character not in the brackets
*	Matches zero or more occurrences of the previous character
.	Matches any single character (Note that .* is often useful)
$	Matches the end of a line

Change To: In the Change To edit box, type the replacement string used by the change buttons at the bottom of the Search window. You can use the same special character sequences in the Change To box as listed above under Search For. An empty Change To string is meaningful; it causes the found string to be deleted when a change function is used.

Search Direction: Click Forward or Backward to control the search direction. Searching always starts at the insertion point.

Change All Scope: Click one of these settings to control the range across which the Change All function works. If you turn on Current Selection and then click Change All, only the selected text is affected.

Search: Click Search to search for the Search For string. FrameMaker starts searching at the insertion point, searches in the requested Search Direction, and wraps around the document if it reaches the end without first finding the string. All text objects in a document are searched, page by page.

If the search string is found, the page containing the string is displayed, and the string is selected. At this point, you can use all commands that affect selected text (such as Change, Delete, and Fonts) to modify the found string. If FrameMaker can't find the string in the document, it displays a dialog box.

Change: Click Change to replace the selected text with the Change To text, leaving the pointer at the end of the replaced text.

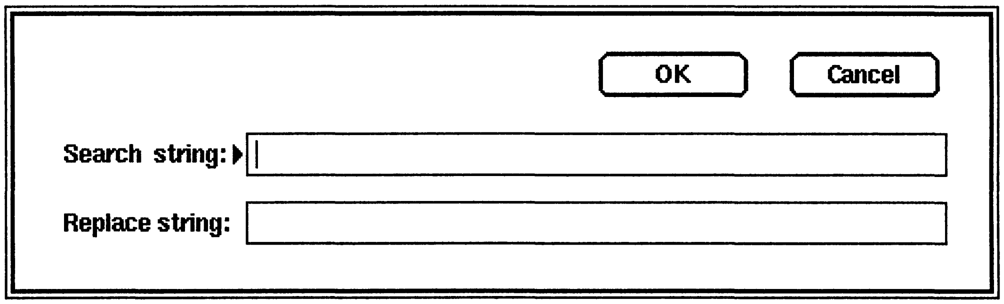
Change & Search Again: Click Change & Search Again to replace the selected text with the Change To text and then to search for the next occurrence of the Search For string.

Change All: Click Change All to search all text in the document (TextRects and TextLines) or to search all of the selected text, and to replace all occurrences of the Search For string with the Change To string. A dialog box appears to make sure you really want to do this. You can stop the Change All function before it is finished by putting the arrow pointer in the document window and pressing Ctrl-c.

Notes:

- For information on keyboard shortcuts for searching and replacing text, see *Search and Replace Commands* in Appendix A.
- While it is useful to be able to work in other windows while the Search window is on the screen, this capability creates a problem for dedicated users of the keyboard, since you must move the arrow pointer into the Search window in order to type text into the window's edit boxes. Because of this, you cannot build a keyboard macro that sets the Search For or Change To strings in the Search window when the arrow pointer is within a document window.

To get around this problem, use the document window keyboard command called Set Search Strings; press Esc s s to display the Set Search Strings dialog box:



Since this is a standard FrameMaker dialog box, all keyboard input is processed by the dialog box regardless of the arrow pointer's location.

In this dialog box, type the Search and Replace strings as needed and press Enter (or press Ctrl-c to cancel). If the Search window is on the screen, it is updated to show changes made using the Set Search Strings dialog box. Once you have set the search strings correctly, you can use the keyboard shortcuts for searching.

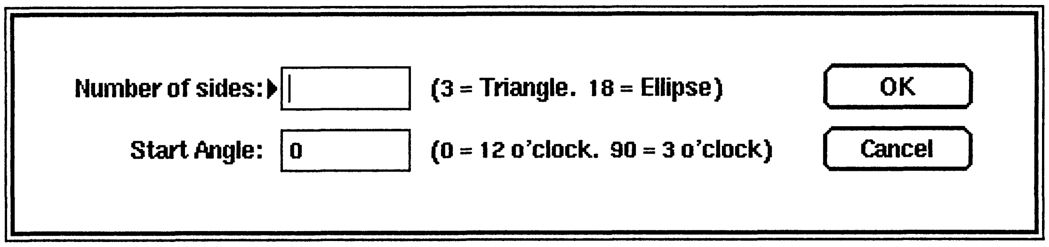
Set # Sides

Location: Tools window

Standard Keyboard Equivalent: `Esc o n`

Purpose: Use the Set # Sides command to convert a selected polygon or ellipse into either a regular polygon with the specified number of sides or an ellipse.

Use: To use the Set # Sides command, select a single polygon, rectangle, square, circle, or ellipse and choose the Set # Sides command in the Tools window. The Set # Sides dialog box appears:



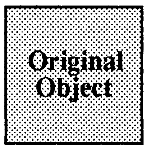
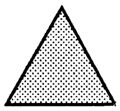
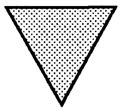
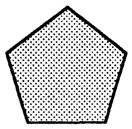
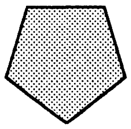
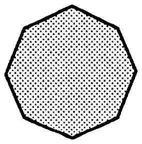
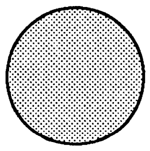
Number of sides: (3 = Triangle. 18 = Ellipse)

Start Angle: 0 (0 = 12 o'clock. 90 = 3 o'clock)

Number of Sides: In this box, type the number of sides to be used when creating the regular polygon. You can indicate any number between 3 and 18 (in FrameMaker, an 18-sided object is an ellipse).

Start Angle: In this box, specify the rotation (in degrees) of the polygon. If you specify 0 degrees, FrameMaker creates a polygon with a vertex at the top of the polygon.

Here are some samples:

						
# Sides:	3	3	5	5	8	18
Start Angle:	0	60	0	36	0	0

Notes:

- FrameMaker does not store or draw ellipses as 18-sided regular polygons; instead, it stores them as ellipse primitives and draws them at full output device resolution. However, to convert a polygon into a true ellipse, you must select the polygon and indicate that you want to convert it to an 18-sided object in order to obtain an ellipse.

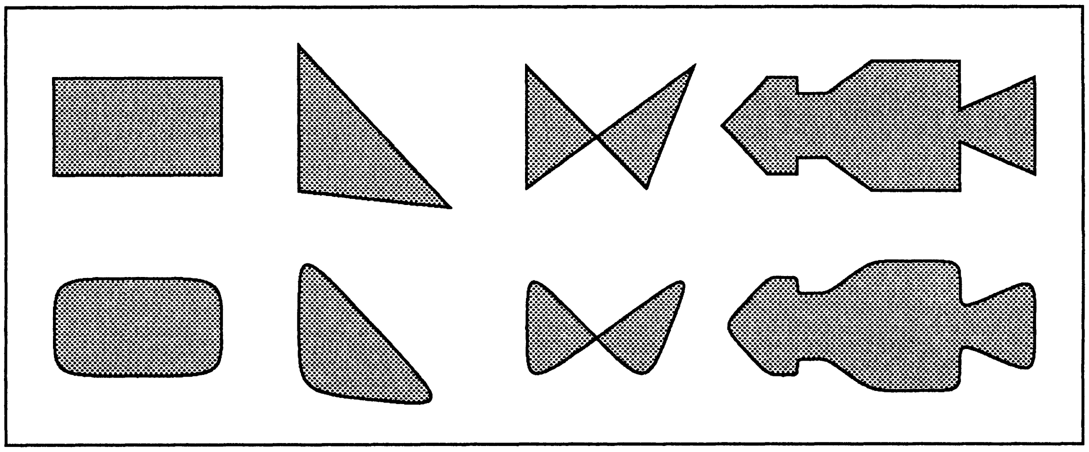
Smooth

Location: Tools window

Standard Keyboard Equivalent: `Esc o s`

Purpose: Use the Smooth command to round the corners of selected polygons, polylines, and freehand lines.

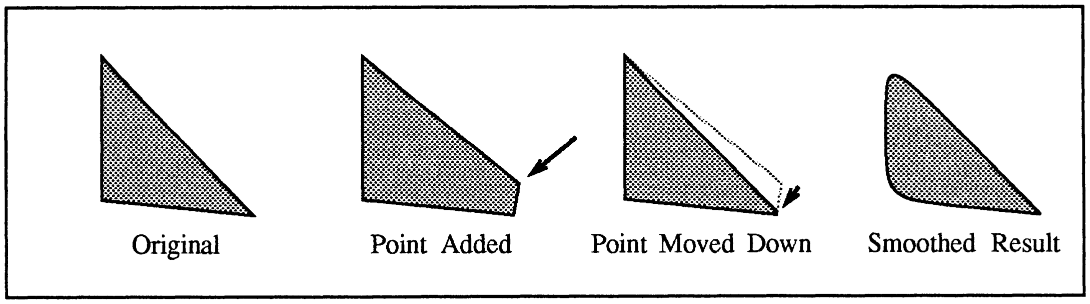
Use: To smooth an object, select it and then choose the Smooth command in the Tools window.



Objects before Smoothing (top) and after Smoothing (bottom)

Notes:

- To leave a corner unsmoothed, use the Reshape command to add an extra point near that corner and then move the new point on top of the original point (see the Reshape command):



Snap

Location: Guides menu

Standard Keyboard Equivalent: `Esc g s`

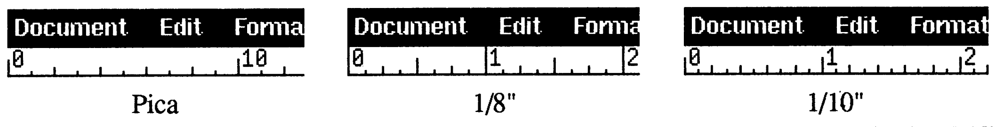
Purpose: Use the Snap command to cause objects that are drawn, moved, and stretched with the mouse to "snap" to positions that coincide with the smallest ruler divisions. Use this command to align and connect objects. New FrameMaker documents appear with the snap feature turned on.

Use: The Snap command toggles the snap feature on and off. When snap is on, a mark (for example, a checkmark or an *x)* appears next to the Snap command in the Guides menu.

Notes:

- To change the snap-point spacing, use the Units command from the Guides menu to change the Ruler Divisions setting.

Snap Points Correspond to the Ruler Tick Marks



If you draw an object while the snap feature is off and later try to move it while snap is on, a problem may occur if the object size does not match the snap points on all sides. If one side is moved to a snap point, the other side does not coincide with a snap point.

When you move such an object, you can control which side is snapped to the ruler divisions by first moving in the direction of the side you want to snap. For example, if you want the left side to be moved to a snap point, move to the left when you first press the mouse button. Then, without releasing the button, you can move left or right as needed; the left side jumps from snap point to snap point.

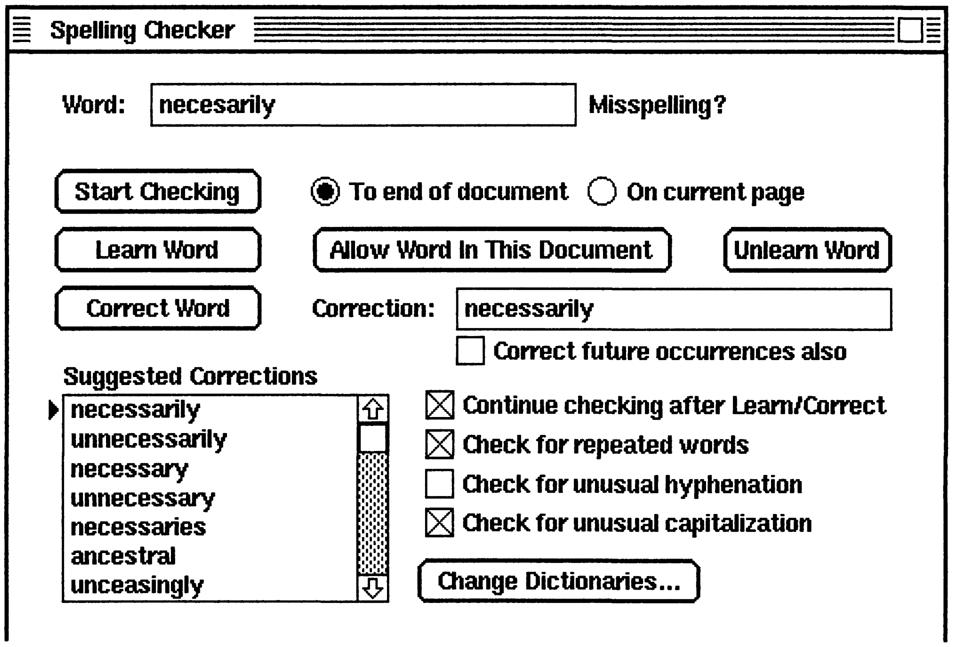
Spelling Checker

Location: Edit menu

Standard Keyboard Equivalent: `Esc e l`

Purpose: Use the Spelling Checker command to access FrameMaker's spell checking features. These features allow you to look for and correct the following types of errors: misspellings, repeated words, incorrect hyphenation, and incorrect capitalization.

Use: Put the insertion point at the appropriate spot in the document (usually, but not always, at the beginning of the document) and choose the Spelling Checker command from the Edit menu. The Spelling Checker window appears. You can also set the insertion point after you choose the Spelling Checker command.


Spelling Checker

Word: necesarily Misspelling?

[✓ To end of document] On current page

Correction: necessarily

☐ Correct future occurrences also

Suggested Corrections
- necessarily
- unnecessarily
- necessary
- unnecessary
- necessaries
- ancestral
- unceasingly

☒ Continue checking after Learn/Correct
☒ Check for repeated words
☐ Check for unusual hyphenation
☒ Check for unusual capitalization

Word: In this box, FrameMaker displays a word or string of characters it considers to be incorrect.

Whenever FrameMaker finds a problematic string, it updates the contents of the Word edit box and displays a status message to the right of the box (Misspelling? in the example shown above). Possible messages include Misspelling, Bad Capitalization, Repeated Word?, Bad (), and Bad Hyphenation. All messages appear with a question mark, indicating that FrameMaker is not sure that there really is an error.

Start Checking: Click Start Checking to begin checking at the start of the word containing the insertion point. Click To End of Document to check from the insertion point to the end of the document; FrameMaker will not check any text before the insertion point. Click On Current Page to check the entire page; FrameMaker checks all text on the page regardless of the location of the insertion point. In either case, FrameMaker checks TextRects in their draw order (the order FrameMaker draws them when you display a page). While checking for problems, FrameMaker displays an hourglass.

When FrameMaker finds a problem, it displays the page containing the problem, selects the problematic string, copies the string into the Word edit box in the Spelling Checker window, and displays a status message describing the suspected problem. At that point you normally click one of the other buttons in the Spelling Checker window, or you correct the word manually by editing it in the document window. If no problem is found, FrameMaker displays Spelling OK in the status area of the Spelling Checker window.

You can also check the spelling of a single word or a text selection. To do so, put the insertion point within the word or select the text to be checked, hold down the Alt key, and click Start Checking. If no text is selected, FrameMaker checks the spelling of the word containing the insertion point. Otherwise it checks the spelling of the selected text. If it finds a misspelling, it selects the misspelling.

Learn Word: Click Learn Word to store the specified word in your current personal dictionary (see notes below). Future occurrences of the word are not considered as misspellings for *all* documents for which you check spelling.

If the word uses all lowercase letters, FrameMaker assumes that the word follows normal capitalization rules (that is, it may be in lowercase or have an initial capital letter). If the word contains capital letters, FrameMaker assumes that the capitalization is significant. If you want the capitalization to not be significant, but the word is capitalized (because it was found at the start of a sentence), you should edit the word that appears in the Word edit box of the Spelling Checker window so that it is all in lowercase before clicking Learn Word.

After FrameMaker stores the word in the dictionary, it normally continues to check spelling, as if you had clicked the Start Checking button again (much like the Change & Search Again button in the Search window). If you do not want to continue checking, turn off Continue Checking after Learn/Correct (the first check box in the lower-right corner of the Spelling Checker window), or hold down the Alt key before clicking Learn Word.

Some errors flagged by FrameMaker cannot be learned, such as repeated words and unusual punctuation patterns. If you try to learn such a sequence, FrameMaker displays a message informing you that the sequence cannot be learned. Unfortunately, that sequence is flagged every time you check the document. To ignore the word, click Start Checking.

Allow Word In This Document: Click this button to store the specified word in the current document's dictionary. The word is not considered a misspelling within this particular document, but it continues to be flagged as a misspelling in other documents. This setting follows the same capitalization, learning, and continuation processing described for Learn Word.

Unlearn Word: Click this button to remove the specified word from both your current personal dictionary and from the document's dictionary, if the word is in either place. Click this button if you click Learn Word or Allow Word by mistake. You can also type any word you want into the Word edit box and click Unlearn Word.

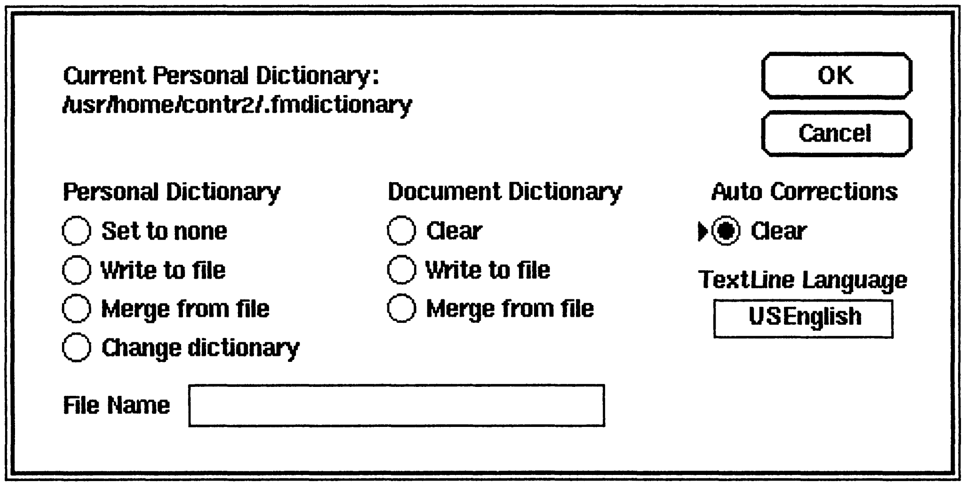
Correct Word: Click this button to replace whatever text is selected in the document window with the text in the Correction edit box. Whenever FrameMaker finds a problem, it suggests a list of possible corrections. The list is displayed in the Suggested Corrections scroll list, sorted from the most likely to least likely suggestion. The most likely suggestion automatically appears in the Correction edit box. You can choose a different correction by clicking it in the scroll list or by typing in a new one in the Correction edit box. Clicking a word in the scroll list automatically copies that word into the Correction edit box.

When you type a word in the Correction edit box and click Correct Word, FrameMaker inserts the word in the document and then checks its spelling. If it doesn't find the word in its dictionary, it selects it. You can then retype the word in the Correction edit box if you made a typo or use the Learn Word function to add the word to your personal dictionary.

If the Correct Future Occurrences Also setting is turned on, then FrameMaker remembers the word and its correction in the auto-correction list. If the same word is found again, FrameMaker automatically corrects the word without any user intervention. The auto-correction list is remembered within a FrameMaker session and affects all documents checked in that session. It cannot be saved and is not associated with a document or dictionary. You can clear the auto-correction list at any time using the Change Dictionaries function described below.

Check for...: These three check boxes control checking for repeated words, unusual hyphenation, and unusual capitalization. FrameMaker's notion of unusual hyphenation is quite restrictive, so this setting is initially turned off. Unusual capitalization includes sentences that start with a lowercase letter and unusual patterns such as *THe* instead of *the*, *The*, or *THE*.

Change Dictionaries: Click this setting to display the Change Dictionaries dialog box, which allows you to manipulate your personal dictionary, the document dictionary, and the auto-corrections list:


Current Personal Dictionary:
/usr/home/contr2/.fmdictionary

Personal Dictionary	Document Dictionary	Auto Corrections
○ Set to none	○ Clear	● Clear
○ Write to file	○ Write to file	TextLine Language
○ Merge from file	○ Merge from file	USEnglish
○ Change dictionary		

File Name

Personal Dictionary: At any moment there can be no more than one active personal dictionary. When you start FrameMaker, `~/.fmdictionary` is your personal dictionary.

The link between FrameMaker and a personal dictionary is fleeting, in that FrameMaker does not keep the file open across spell-checking activities. Rather, it reads the file into a fast lookup table (FLT) at startup and then closes it. When you use the Learn Word function, the word gets added to the personal dictionary by opening, writing, and closing the dictionary file (it also gets added to the FLT). At all times, FrameMaker tries to make sure that the FLT and the personal dictionary file have the same words in them.

- **Set to None:** Click this setting to turn off use of a personal dictionary. The FLT is cleared. After using this setting, you cannot use the Learn Word function until you reestablish a personal dictionary using the Change Dictionary setting defined below.
- **Write to File:** Click this setting to write a copy of the words in the FLT, creating a dictionary file (defined below). The file is written using the name specified in the File Name edit box.
- **Merge from File:** Click this setting to read the dictionary file specified in the File Name edit box. Each word in the file is merged into both the FLT and the current personal dictionary file.
- **Change Dictionary:** Click this setting to clear the FLT and then use the dictionary file specified in the File Name edit box as the current personal dictionary. Each word from the file is read, building a new FLT.

Document Dictionary: The document dictionary is stored within the document, like the Paragraph Catalog. You can clear it, write to a dictionary file, and augment it by reading in a dictionary file.

- **Clear:** Click Clear to remove all words from the current document's dictionary.
- **Write to File:** Click this setting to write a copy of the words in the current document's dictionary to the file specified in the File Name edit box, creating a dictionary file.
- **Merge from File:** Click this setting to read the dictionary file specified in the File Name edit box. Each word in the file is merged into the current document's dictionary.

Clear Auto Corrections: Click this setting to clear all of the auto-corrections established in the current FrameMaker session. See the description of the Correct Word function above for the definition of auto-corrections.

TextLine Language: In this area, specify the language to be used when checking the spelling of TextLines in the document. If you're using the U.S. version of FrameMaker, only USEnglish is available. If you're using the International version of FrameMaker, you can choose from languages such as USEnglish, UKEnglish, Dutch, French, and German. Scroll through the available languages by clicking repeatedly in the TextLine Language box. For information about setting the language of paragraphs in your document, see the Paragraphs command.

Notes:

- Up to four dictionaries can be active at one time. They are the main, site, personal, and document dictionaries. FrameMaker looks through all four dictionaries (if they are all active) before it considers a word to be misspelled.

  The main dictionary contains 80,000 common words such as *the*, *main*, and *active*. It cannot be edited.

  You can, however, edit the site dictionary. The site dictionary is shipped with about 200 technical words in it, which you can delete or add to as needed (its format is described later in these notes). See Appendix D, *Customizing FrameMaker,* for information on where the site dictionary is stored.

  The site dictionary is read only once, during startup. To use a different site dictionary, you must quit and restart FrameMaker.

  There are several types of words that a system administrator would typically put in the site dictionary. One example is your company name. Some names do not need to be added because they are made up of words already in the main dictionary (such as *Frame Technology Corporation*); other names would normally be flagged as misspellings (such as *BondVideo*). Other candidates for inclusion in the site dictionary include your company president's name and the names of products produced by your company (such as *GPX007*).

  The personal and document dictionaries are described above.

- The personal and site dictionary files use a standard format. They contain ASCII text. The first line of the file must be `<MakerDictionary 1.0>`. After that, each word is placed on a line by itself. If these files contain more than 5000 words, spell-checking performance may be slow.

  For an example of the dictionary file format, see the site dictionary.

- When FrameMaker selects a word, you have several possible courses of action.

  Click Learn Word if the word is clearly not a misspelling in this or any other context (such as *academia*).

  Click Allow Word In This Document if the word might be a misspelling in other documents but is clearly not one in this document (such as *Thimk* in a report on a product named Thimk).

  Click Correct Word if the word is clearly a misspelling and the correction is another word that can easily be entered into the Correction edit box.

  Click Start Checking to leave the word unresolved. You would do this if the word could be a misspelling if it occurred anywhere else. For example, you may have the word *usr* in some UNIX documentation. If you store *usr* in your personal dictionary or the document dictionary, then you run the risk of missing cases where *usr* should really be *user*.

  Finally, you may choose to edit the word in the document window. Most often you do that when the misspelling is caused by a missing space, as in *thereason*.

- When you first start to check documents, you may find some words incorrectly flagged as spelling errors. This is because FrameMaker does not know your special vocabulary. After you check several documents, if you use the Learn Word function properly, you find that FrameMaker becomes much more accurate, limiting its feedback to mostly questionable words.

- If you don't know how to spell a word, and FrameMaker is unable to identify a correct spelling for you, try typing the word phonetically (that is, the way it sounds). For example, if you type *justificashun*, FrameMaker will select the misspelled word and suggest the correct spelling, *justification*.

- FrameMaker does not check text in Printer Code TextRects. For more information on Printer Code TextRects, see the Printer Code command in this chapter.

- FrameMaker can check approximately one page per second the first time it checks a document. FrameMaker keeps track of what has already been checked so that it can avoid rechecking unchanged areas. For this reason, once you have a document cleaned up, you can recheck it as you make additions with very fast response time. You can press Esc l r (see *Spelling Checker Commands* in Appendix A) to force FrameMaker to recheck all document text.

- There are several keyboard shortcuts for checking spelling. See *Spelling Checker Commands* in Appendix A.

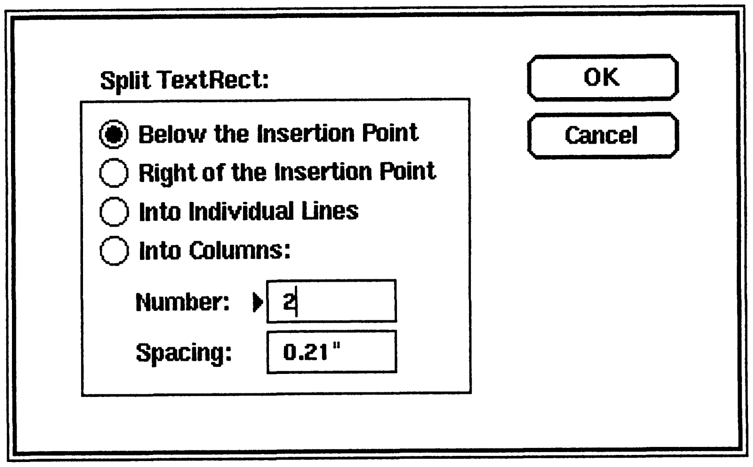
Split TextRect

Location: TextRects menu

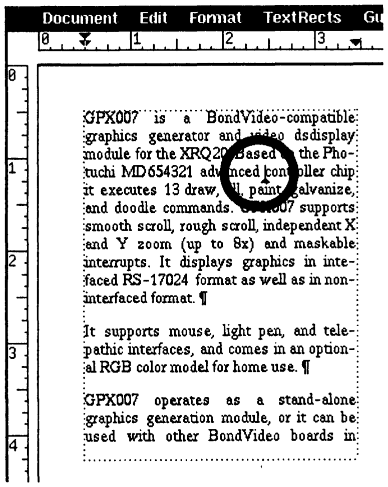
Standard Keyboard Equivalent: `Esc t s`

Purpose: Use the Split TextRect command to split a TextRect into two or more linked TextRects. Use this command when you need to create custom page layouts, run text around objects, and fit copy. See the examples below and in the *FrameMaker User's Manual.*

Use: Select a TextRect with the left or middle mouse button and choose the Split TextRects command from the TextRects menu. The Split TextRect dialog box appears:


Split TextRect:
[✓ Below the Insertion Point]
Right of the Insertion Point
Into Individual Lines
Into Columns:
Number: 2
Spacing: 0.21"

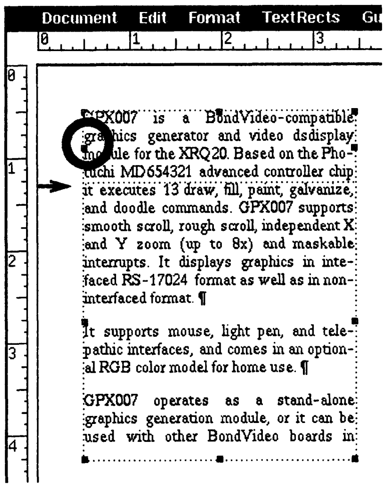
Below the Insertion Point: Click this setting to split the selected TextRect into two linked TextRects, stacked one above the other, as shown below:



GPX007 is a BondVideo-compatible graphics generator and video dsdisplay module for the XRQ20. Based on the Photuchi MD654321 advanced controller chip it executes 13 draw, fill, paint, galvanize, and doodle commands. GPX007 supports smooth scroll, rough scroll, independent X and Y zoom (up to 8x) and maskable interrupts. It displays graphics in intefaced RS-17024 format as well as in non-interfaced format. ¶

It supports mouse, light pen, and telepathic interfaces, and comes in an optional RGB color model for home use. ¶

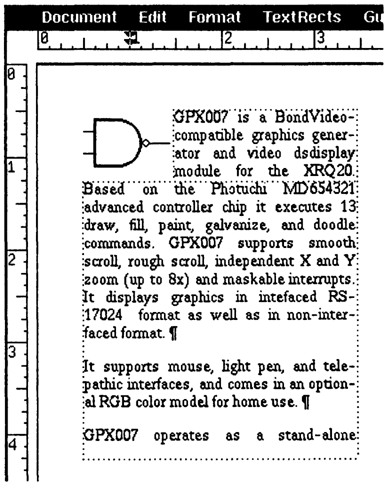
GPX007 operates as a stand-alone graphics generation module, or it can be used with other BondVideo boards in



GPX007 is a BondVideo-compatible graphics generator and video dsdisplay module for the XRQ20. Based on the Photuchi MD654321 advanced controller chip it executes 13 draw, fill, paint, galvanize, and doodle commands. GPX007 supports smooth scroll, rough scroll, independent X and Y zoom (up to 8x) and maskable interrupts. It displays graphics in intefaced RS-17024 format as well as in non-interfaced format. ¶

It supports mouse, light pen, and telepathic interfaces, and comes in an optional RGB color model for home use. ¶

GPX007 operates as a stand-alone graphics generation module, or it can be used with other BondVideo boards in



GPX007 is a BondVideo-compatible graphics generator and video dsdisplay module for the XRQ20. Based on the Photuchi MD654321 advanced controller chip it executes 13 draw, fill, paint, galvanize, and doodle commands. GPX007 supports smooth scroll, rough scroll, independent X and Y zoom (up to 8x) and maskable interrupts. It displays graphics in intefaced RS-17024 format as well as in non-interfaced format. ¶

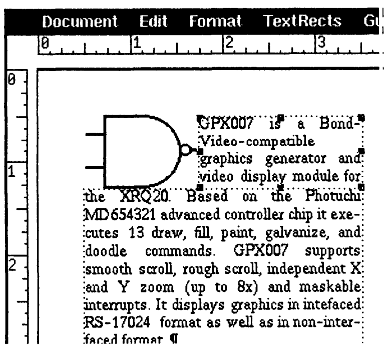
It supports mouse, light pen, and telepathic interfaces, and comes in an optional RGB color model for home use. ¶

GPX007 operates as a stand-alone

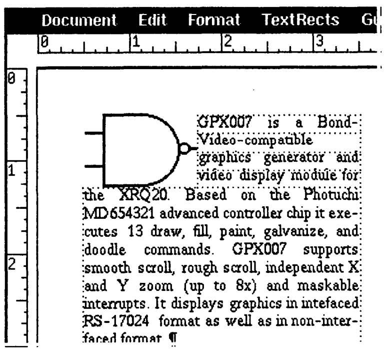
Left: Position the insertion point anywhere in the fourth line, choose the Split TextRect command, click Below the insertion point, and click OK.
Center: This shows the resulting two TextRects (notice the border below the fourth line). The text flow within the two TextRects has not changed. Now, drag the left-center handle to the right to make room for a graphic.
Right: Notice that the text flows to fit the new shape.

Right of the Insertion Point: Click this setting to split the selected TextRect into two linked TextRects placed next to each other.

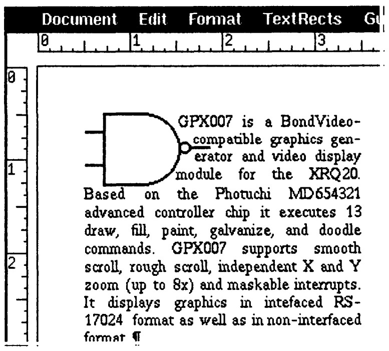
Into Individual Lines: Click this setting to split the selected TextRect into many linked TextRects, one for each line in the original TextRect. Use this setting for copy fitting:



GPX007 is a BondVideo-compatible graphics generator and video display module for the XRQ20. Based on the Photuchi MD654321 advanced controller chip it executes 13 draw, fill, paint, galvanize, and doodle commands. GPX007 supports smooth scroll, rough scroll, independent X and Y zoom (up to 8x) and maskable interrupts. It displays graphics in intefaced RS-17024 format as well as in non-interfaced format. ¶



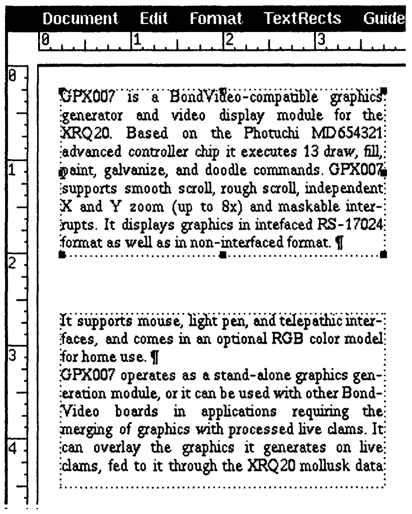
GPX007 is a BondVideo-compatible graphics generator and video display module for the XRQ20. Based on the Photuchi MD654321 advanced controller chip it executes 13 draw, fill, paint, galvanize, and doodle commands. GPX007 supports smooth scroll, rough scroll, independent X and Y zoom (up to 8x) and maskable interrupts. It displays graphics in intefaced RS-17024 format as well as in non-interfaced format. ¶



GPX007 is a BondVideo-compatible graphics generator and video display module for the XRQ20. Based on the Photuchi MD654321 advanced controller chip it executes 13 draw, fill, paint, galvanize, and doodle commands. GPX007 supports smooth scroll, rough scroll, independent X and Y zoom (up to 8x) and maskable interrupts. It displays graphics in intefaced RS-17024 format as well as in non-interfaced format. ¶

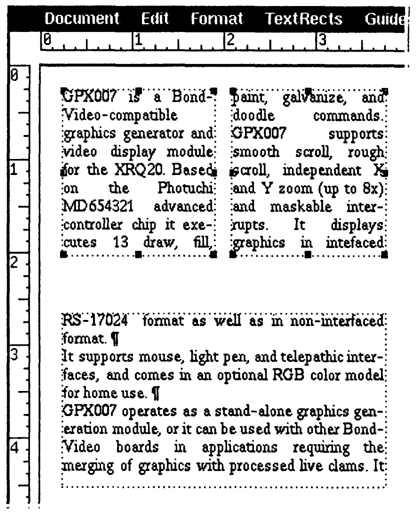
Left: Select the upper TextRect and use the Split TextRect command to split into individual lines.
Center: This shows the resulting four TextRects. The text flow within the four TextRects has not changed. Now, drag the left-center handles in or out to fit around the graphic.
Right: Notice that the text flows to fit the new shape. The TextRects have a transparent fill.

Into Columns: Click this setting to split the selected TextRect into the specified number of equal-sized, equally spaced columns:



GPX007 is a BondVideo-compatible graphics generator and video display module for the XRQ20. Based on the Photuchi MD654321 advanced controller chip it executes 13 draw, fill, paint, galvanize, and doodle commands. GPX007 supports smooth scroll, rough scroll, independent X and Y zoom (up to 8x) and maskable interrupts. It displays graphics in intefaced RS-17024 format as well as in non-interfaced format. ¶

It supports mouse, light pen, and telepathic interfaces, and comes in an optional RGB color model for home use. ¶
GPX007 operates as a stand-alone graphics generation module, or it can be used with other Bond-Video boards in applications requiring the merging of graphics with processed live clams. It can overlay the graphics it generates on live clams, fed to it through the XRQ20 mollusk data



GPX007 is a Bond-Video-compatible graphics generator and video display module for the XRQ20. Based on the Photuchi MD654321 advanced controller chip it executes 13 draw, fill,

paint, galvanize, and doodle commands. GPX007 supports smooth scroll, rough scroll, independent X and Y zoom (up to 8x) and maskable interrupts. It displays graphics in intefaced

RS-17024 format as well as in non-interfaced format. ¶
It supports mouse, light pen, and telepathic interfaces, and comes in an optional RGB color model for home use. ¶
GPX007 operates as a stand-alone graphics generation module, or it can be used with other Bond-Video boards in applications requiring the merging of graphics with processed live clams. It

Notes:

- To merge two linked TextRects into one, select one of them using the left mouse button and delete it using the Cut command (deleting the TextRect but leaving the text it contained within the text flow). Then stretch the remaining TextRect to the desired size.

Tabs

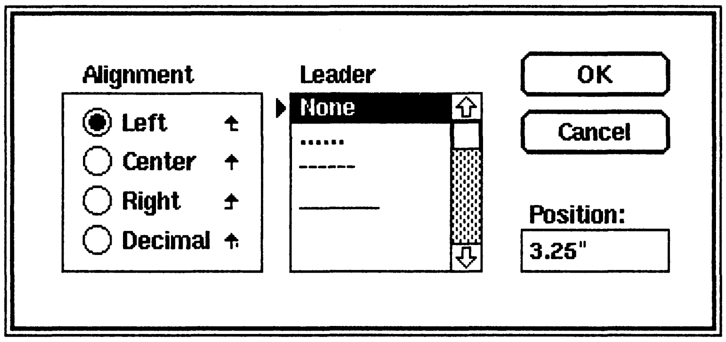
Location: Format menu

Standard Keyboard Equivalent: `Esc f t`

You can add tab stops with the Tabs command from the Format menu or with the mouse. Once you've created tab stops, you can move and delete them with the mouse. Each of these operations is described below.

Adding Tab Stops with the Tabs Command

Use the Tabs command to add new tab stops to the current paragraph. The Tabs command displays the Tabs dialog box:



Alignment: Click one of these settings to specify the kind of tab stop: left, right, center, or decimal.

Leader: Select None for no leader or select one of the optional leader character settings to put a continuous leader pattern between the tab symbol in the document text and the character following the tab symbol.

Position: In this box, type the tab stop position relative to the left edge of the paragraph's enclosing TextRect. If a tab already exists at that location, its attributes are set to match those specified in the dialog box.

OK: Click OK to set the specified tab stop at the specified position and to set the default tab stop type (see *Adding Tab Stops with the Mouse*, next).

Notes:

- When you turn on rulers, the current paragraph's tab settings appear immediately below the ruler. The tab symbols are the four different up-pointing arrows, representing the four different kinds of tab alignment, also seen in the Tabs dialog box.

Adding Tab Stops with the Mouse

When rulers are displayed, you can add tab stops with the mouse (see the Rulers command). When you add tab stops with the mouse, FrameMaker creates them using the Alignment and Leader type displayed in the Tabs dialog box.

To add a tab stop using the mouse:

- Click the middle mouse button in the appropriate paragraph.

  Notice that symbols appear below the ruler at the top of the page to show the paragraph's current tab settings (see the Paragraphs command).
- Move the arrow pointer into the thin area below the top ruler so that its tip points at the place where you want the tab stop.
- Click the left mouse button.

  FrameMaker places an up-pointing arrow showing where you added the tab stop.

Notes:

- The tip of the arrow's head shows the tab stop's exact position.
- Before adding tab stops with the mouse, you may want to turn on the snap feature using the Snap command. With snap on, FrameMaker adds tab stops at the ruler subdivision nearest to where you click the mouse button, making it easier to place tab stops at exact locations.

Moving Tab Stops

With rulers displayed, you can move tab stops using the mouse (see the Rulers command).

To move a tab stop using the mouse:

- Click the middle mouse button in the appropriate paragraph.

  Notice that up-pointing arrow symbols appear below the ruler at the top of the page to show the paragraph's current tab settings.
- Move the arrow pointer so that its tip is within the tab stop symbol to be moved.
- Press and hold down the left mouse button.
- Drag the tab stop symbol left or right by sliding the mouse without releasing the mouse button.
- When the tab stop symbol is at the desired position, release the mouse button.

Deleting Tab Stops

With rulers displayed, you can delete tab stops using the mouse (see the Rulers command.

To delete a tab stop:

- Click the middle mouse button in the appropriate paragraph.

  Notice that up-pointing arrow symbols appear below the ruler at the top of the page to show the paragraph's current tab settings.
- Move the arrow pointer so that its tip is within the tab stop symbol to be deleted.
- Press and hold down the left mouse button.
- Drag the tab stop symbol down into the page area of the document window by sliding the mouse without releasing the mouse button.
- When the tab stop symbol is within the page area, release the mouse button.

  The tab stop is deleted.

Copying Tab Stops from One Paragraph to Another

A paragraph's tab stops are actually part of its paragraph format (see the Paragraphs command). When a paragraph format is applied to another paragraph, the tab settings are also applied. See the following commands for instructions for copying one paragraph's format to other paragraphs: Copy Pgf Format, Paragraphs, and Catalog.

Using Tabs in the Document

To add a tab, set the insertion point at the appropriate position in the text and press the Tab key. Tab symbols are nonprinting and do not show up on the screen unless text symbols are turned on using the Text Symbols command from the Guides menu. The symbol for a tab is (❯).

Unlike a typewriter, successive tab symbols on a line of text correspond to successive tab stops, one for one. So, to move the insertion point to the third tab stop, the line must contain three tab symbols. The advantage of this approach is that tabbing remains correct even if characters are deleted to the left of the tab symbols. For example, in a columnar table there is one tab stop for each column, and each row uses a single tab symbol to separate each column entry, no matter how narrow each entry is.

Text Symbols

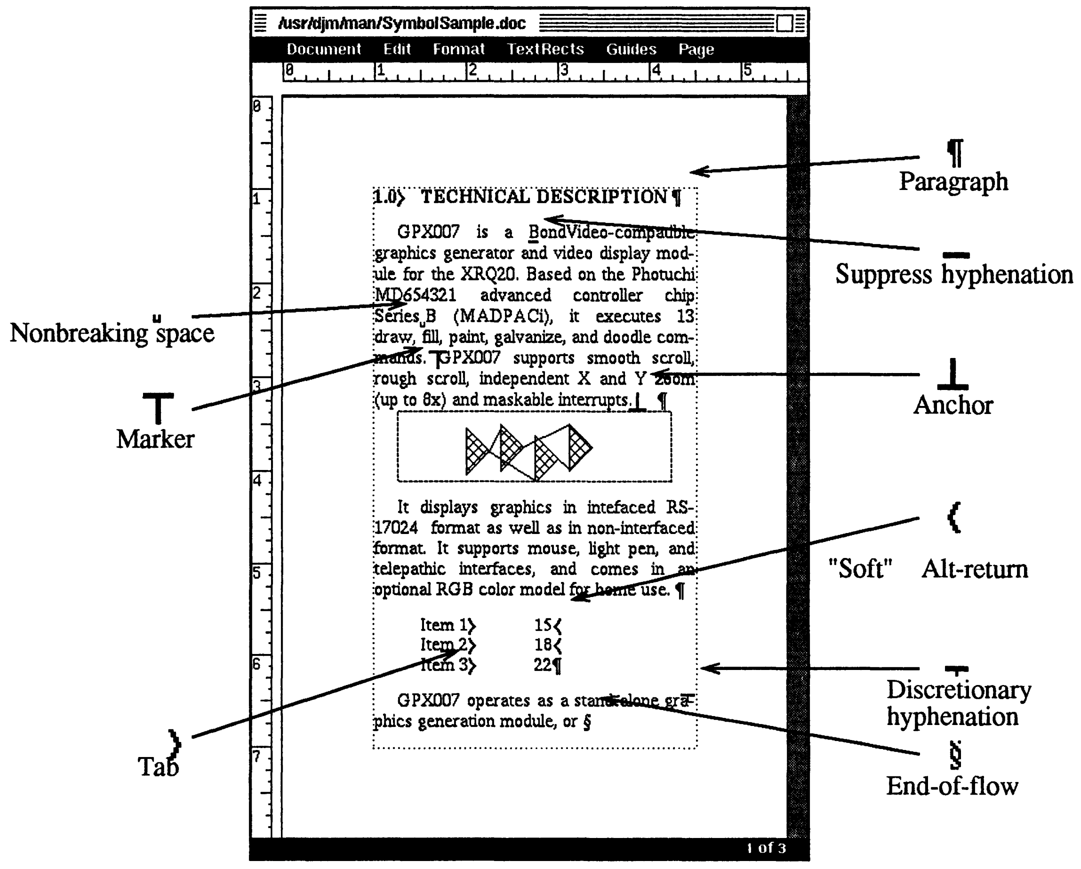
Location: Guides menu

Standard Keyboard Equivalent: `Esc g t`

Purpose: Use the Text Symbols command to turn the display of nonprinting characters on and off. Nonprinting characters include end-of-paragraph, end-of-flow, tab, nonbreaking space, and so on. When display of text symbols is on, special symbols appear on the screen to indicate the presence of nonprinting characters.

Use: When the display of text symbols is on, a mark (for example, a checkmark or an *x)* appears next to the Text Symbols command in the Guides menu.

The symbols for nonprinting characters are shown below:



1.0› TECHNICAL DESCRIPTION ¶

GPX007 is a BondVideo-compatible graphics generator and video display module for the XRQ20. Based on the Photuchi MD654321 advanced controller chip Series B (MADPACi), it executes 13 draw, fill, paint, galvanize, and doodle commands. GPX007 supports smooth scroll, rough scroll, independent X and Y zoom (up to 8x) and maskable interrupts. ¶

It displays graphics in intefaced RS-17024 format as well as in non-interfaced format. It supports mouse, light pen, and telepathic interfaces, and comes in an optional RGB color model for home use. ¶

```
Item 1›     15‹
Item 2›     18‹
Item 3›     22¶
```

GPX007 operates as a stand-alone graphics generation module, or §

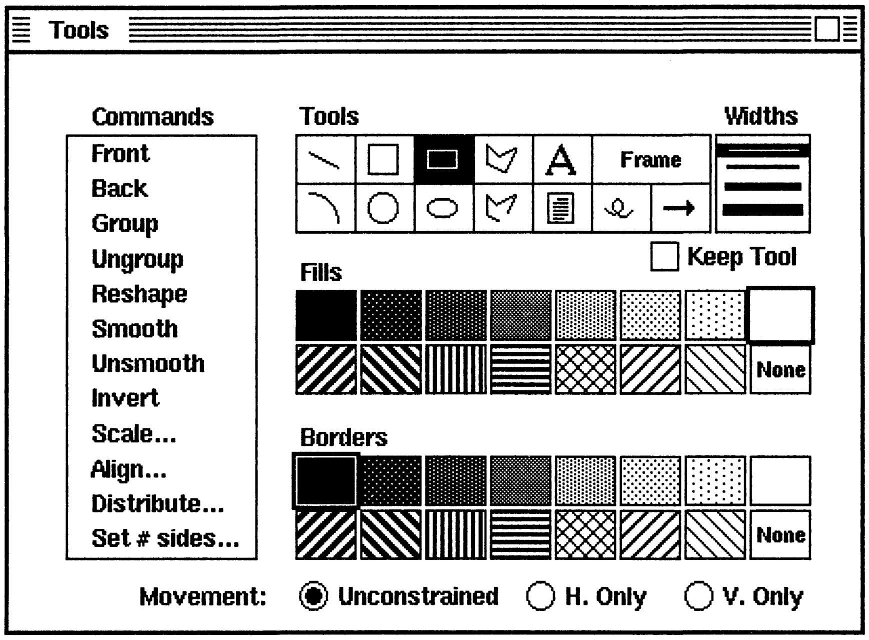
Tools

Location: Main FrameMaker window

Standard Keyboard Equivalent: `Esc d t`

Purpose: Use the Tools command to open the Tools window. If the Tools window is already open, click TOOLS in the main FrameMaker window to move the Tools window to the front of all windows on the screen.

Use: Click TOOLS in the main FrameMaker window to display the Tools window:



The individual commands listed in the Tools window are described in this chapter. Other items in the Tools window are described in Chapter 4.

You can manipulate the Tools window as you do FrameMaker document windows (for example, moving it so it doesn't overlap a document window). Iconifying the Tools window closes it. See your window system documentation for information on manipulating windows.

Undo

Location: Edit menu and Maker menu

Standard Keyboard Equivalent: `Esc e u`

Purpose: Use the Undo command to reverse the effect of the previous command. Most FrameMaker commands are reversible with the Undo command.

Use: To undo an action, choose Undo from the Edit or Maker menu.

Notes:

- Issuing two Undo commands in a row causes no net change in your document; the second Undo command undoes the effect of the first.
- If you use the Undo command after typing a sequence of characters, the command removes all characters since the insertion point was placed.
- If you use the Undo command after pressing the Delete key in text, it undoes *one* deletion. If you hold down the Delete key to delete several characters, the Undo command restores only the last character you deleted.

Ungroup

Location: Tools window

Standard Keyboard Equivalent: `Esc o u g`

Purpose: Use the Ungroup command to separate groups into the components that were selected when the group was created.

Use: To take apart a group, select it and click Ungroup in the Tools window. If the group itself consists of grouped objects, you'll have to choose the Ungroup command more than once to ungroup the objects completely.

See the Group command for more information.

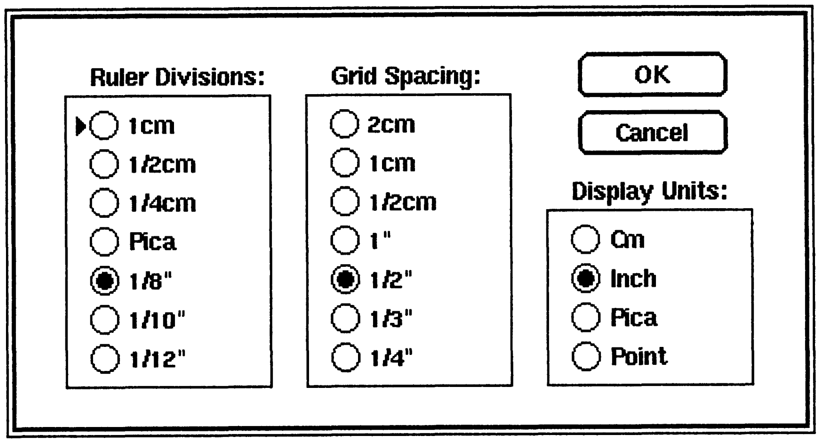
Units

Location: Guides menu

Standard Keyboard Equivalent: `Esc g u`

Purpose: Use the Units command to select ruler divisions, grid spacing, and display units.

Use: Choose the Units command to display the Units dialog box:



Ruler Divisions: 1cm 1/2cm 1/4cm Pica [✓ 1/8"] 1/10" 1/12"

Grid Spacing: 2cm 1cm 1/2cm 1" [✓ 1/2"] 1/3" 1/4"

Display Units: Cm [✓ Inch] Pica Point

Ruler Divisions: Click one of these settings to specify the tick-mark spacing of rulers displayed in the document window. Ruler divisions also affect the Snap command.

Grid Spacing: Click one of these settings to specify the spacing of grid lines.

Display Units: Click one of these settings to specify the units to use to display dimensions in the document window status line and in dialog boxes. This setting sets the assumed units for dimensions entered in dialog boxes and the default units used if the document is saved using the MIF format (see the Save command).

Notes:

- The three areas in the Units dialog box are independent of each other. Normally you choose settings that make sense together, such as 1/8" ruler, 1/2" grid, and Inch display units. A second reasonable set is 1 cm ruler, 1/2 cm grid, and cm display units. It is possible, however, to choose settings that do not go together well, such as Pica ruler, 1 cm grid, and Point display units.

Unsmooth

Location: Tools window.

Standard Keyboard Equivalent: `Esc o u s`

Purpose: Use the Unsmooth command to remove smoothing from selected objects whose corners were rounded with a previous Smooth command.

Use: Select the objects you want to unsmooth and choose the Unsmooth command from the Tools window.

See the Smooth command for more information.

X Paste

Location: Edit menu

Standard Keyboard Equivalent: `Esc e t`

Purpose: Use the X Paste command to copy unformatted text between windows. When you select text in many X Window applications, a copy of the selected text is automatically stored in a special text buffer (a holding place in the workstation's memory).

The X Paste command pastes text from this buffer into a FrameMaker document or dialog box at the insertion point, as if you typed the characters from the keyboard.

Use: To copy text from an xterm window into a FrameMaker document window:

1. In the xterm window, point at the beginning of the text you want to copy.
2. Press and hold down Shift *and* the middle mouse button and drag to the end of the text you want to copy.

   The text is selected.
3. Release both the Shift key and the mouse button.

   The selected text is saved in the buffer.
4. In the FrameMaker window, put the insertion point where you want to paste the text.
5. Choose the X Paste command from the Edit menu.

   The text appears in the document at the insertion point.

To paste a filename into a FrameMaker dialog box (for example, the Open dialog box):

1. In the xterm window, use the *ls* command to list the filename on the screen.
2. Still in the xterm window, select the filename using the steps described above.
3. Display the FrameMaker dialog box, put the insertion point in the edit box in which you want the filename, and type `Esc e t`.

   The filename you selected is pasted into the edit box.

Notes:

- This special text buffer is similar to FrameMaker's Clipboard, but it is less powerful because it holds only unformatted text. The FrameMaker Clipboard can hold formatted text, graphics, font settings, and paragraph formats. Use the Clipboard to copy information between FrameMaker windows. For information about the Clipboard, see the Cut, Copy, and Paste commands.
- To copy text from a FrameMaker document to an xterm window:

  1. In the FrameMaker window, select the text you want to copy (for

information on selecting text in a document window, see Chapter 2).

The selected text is saved in the buffer.

2. Point on the xterm window in which you want to paste the text.
3. Press and hold down Shift and then click the middle mouse button.

   In an xterm window, the text appears at the bottom of the window, after the last prompt.

Tools Window Icons

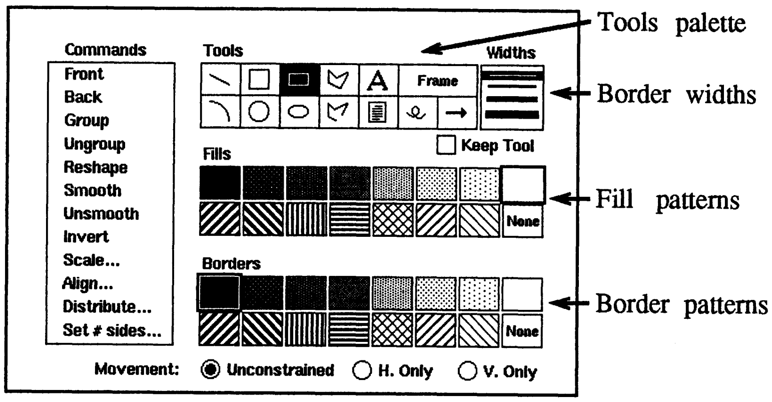
This chapter describes how to use the icons in the Tools window. Display the Tools window by clicking TOOLS in the main FrameMaker window (see the Tools command in Chapter 3). Commands displayed in the Tools window are described in Chapter 3.

Tools Palette

Overview

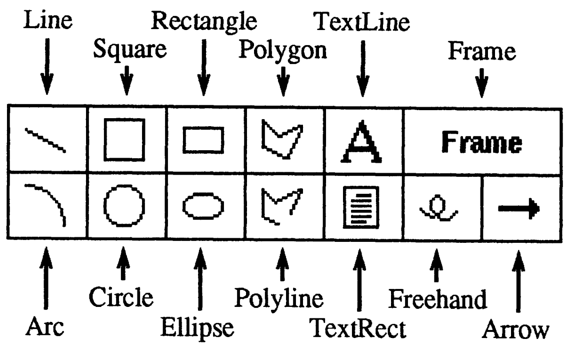
The Tools window contains a palette of tools for drawing graphic and text objects.

To use a tool, click on its icon with the left mouse button. The icon inverts to show that it is selected, and the current settings in the Widths, Fills, and Borders areas are outlined. For example, the Tools window below shows the Rectangle tool selected, along with the thinnest line width, white fill pattern, and black border pattern:



When a tool is selected and the arrow pointer is moved into a document window, the pointer changes into a drawing pointer. The drawing pointer's shape is a reticule (⌖) for all tools except the TextLine, which uses an I-beam pointer (I).

The following diagram shows the meaning of each tool icon in the Tools palette:

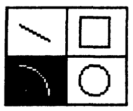


If you select the wrong tool, click a different tool. If you decide not to draw an object, click in the document window with the middle mouse button to deselect the tool.

Instructions for using each tool are provided in the following sections.

Arc

To draw an arc or filled "pie-slice," follow these steps:


Arc tool

- Click the Arc tool with the left mouse button or press Esc 1 a while in the document window.
- Check and, if needed, change the Widths, Fills, and Borders settings in the Tools window.
- Move the arrow pointer to where the arc should begin. If rulers are displayed, tick marks on each ruler show the position of the arrow pointer to aid in accurate placement.
- Press and hold down the left mouse button.
- Move the arrow pointer while holding down the mouse button until the arc stretches to the desired endpoint. Dimension information appears in the status line as you draw the arc.
- Release the mouse button.

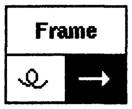
Notes:

- The arc orientation (concave or convex) is controlled by the initial direction you move the mouse after pressing the left mouse button. In general, if you move the arrow pointer so that it traces the desired arc shape, you will achieve the desired orientation.

  If you start to draw an arc and it appears with the wrong orientation as you draw (concave instead of convex or vice versa), do not release the mouse button. Instead, move the arrow pointer back to the starting point and drag the arc out again. Be sure to move the mouse along the desired arc path.

  You can also flip an arc (by dragging one of its handles) to switch its orientation after it is drawn.
- The Arc tool always creates 90 degree arcs. After drawing the arc, you can reshape it to cover any number of degrees and to have any start and end angle. To reshape an arc, select it, click Reshape in the Tools window, and drag one or both of the arc's endpoint handles to create the desired shape. As you move the arc's handle, the arc's start angle, end angle, and percent of an ellipse appear in the status line.
- To change an arc's border width or its border or fill pattern, select the arc and click the appropriate setting in the Tools window. For information on moving and stretching an arc, see *Graphics Editing Overview* in Chapter 2.

Arrow



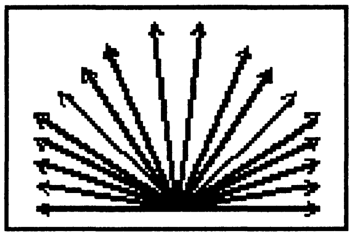
Arrow tool

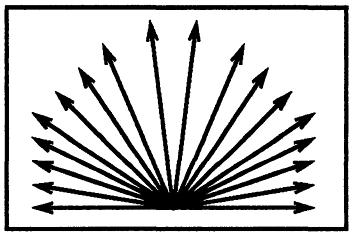
To draw an arrow:

- Click the Arrow tool with the left mouse button or press Esc 1 w while in the document window.
- Check and, if needed, change the Widths and Borders settings in the Tools window.
- Move the arrow pointer to where the tail of the arrow should begin. (The tail is the end without the point.) If rulers are displayed, tick marks on the rulers show the position of the arrow pointer to aid in accurate placement.
- Press and hold down the left mouse button.
- Move the arrow pointer while holding down the mouse button until the arrow stretches to the desired endpoint. Dimension information appears in the status line as the arrow is stretched.
- Release the mouse button.

Notes:

- The Movement settings in the Tools window affect the way that arrows are drawn. For example, to draw a horizontal arrow, click H. Only before drawing the arrow.
- The arrow is drawn with the arrowhead at the point where the mouse button is released.
- To change an arrow's border width or border pattern, select the arrow and click the appropriate setting in the Tools window. For information on moving, stretching, and flipping an arrow, see *Graphics Editing Overview* in Chapter 2.
- Arrows look much better when printed than they do on the screen:


Screen Appearance


Printed Result

- You can change the printed appearance of arrows if you're adventuresome and know something about PostScript programming. See Appendix D, *Customizing FrameMaker,* for more information.

Circle

To draw a circle:

Circle tool

- Click the Circle tool with the left mouse button or press Esc 1 c while in the document window.
- Check and, if needed, change the Widths, Fills, and Borders settings in the Tools window.
- Move the arrow pointer to one corner of the rectangle that the circle should fill. If the rulers are displayed, tick marks on the rulers show the position of the arrow pointer to aid in accurate placement.
- Press and hold down the left mouse button.
- Move the arrow pointer while holding down the mouse button until the circle is stretched to the desired size. Dimension information appears in the status line as the circle is stretched.
- Release the mouse button.

Notes:

- To change a circle's border width or its border or fill pattern, select the circle and click the appropriate setting in the Tools window.
- You can stretch a circle into an ellipse. For information on moving and stretching a circle, see *Graphics Editing Overview* in Chapter 2.

Ellipse

To draw an ellipse:

Ellipse tool

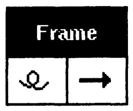
- Click the Ellipse tool with the left mouse button or press Esc 1 e while in the document window.
- Check and, if needed, change the Widths, Fills, and Borders settings in the Tools window.
- Move the arrow pointer to one corner of the rectangle that the ellipse should fill. If the rulers are displayed, tick marks on the rulers show the position of the arrow pointer to aid in accurate placement.
- Press and hold down the left mouse button.
- Move the arrow pointer while holding down the mouse button until the ellipse is stretched to the desired size. Dimension information appears in the status line as the ellipse is stretched.
- Release the mouse button.

Notes:

- To change an ellipse's border width or its border or fill pattern, select the ellipse and click the appropriate setting in the Tools window.
- For information on moving or stretching an ellipse, see *Graphics Editing Overview* in Chapter 2.

Frame

To draw an *unanchored* frame:



Frame tool

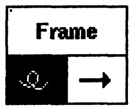
- Click the Frame tool with the left mouse button or press Esc 1 m while in the document window.
- Check and, if needed, change the Widths, Fills, and Borders settings in the Tools window.
- Move the arrow pointer to where one corner of the frame should begin. If rulers are displayed, tick marks on the rulers show the position of the arrow pointer to aid in accurate placement.
- Press and hold down the left mouse button.
- Move the arrow pointer while holding down the mouse button until the frame is stretched to the desired size. Dimension information appears in the status line as the frame is stretched.
- Release the mouse button.

Notes:

- To change a frame's border width or its border or fill pattern, select the frame and click the appropriate setting in the Tools window.
- For information on moving and stretching a frame, see *Graphics Editing Overview* in Chapter 2.
- Create new *anchored* frames with the Anchored Frame command (see Chapter 3).

Freehand

To draw a freehand shape:



Freehand tool

- Click the Freehand tool with the left mouse button or press Esc 1 f while in the document window.
- Check and, if needed, change the Widths and Borders settings in the Tools window.
- Move the arrow pointer to the point where the freehand shape should begin. If the rulers are displayed, tick marks on the rulers show the position of the arrow pointer to aid in accurate placement.
- Press and hold down the left mouse button.
- Move the arrow pointer while holding down the mouse button. As you move the pointer, the freehand shape appears along the path of the drawing pointer.
- Release the mouse button.

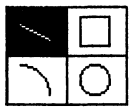
Notes:

- To change a freehand shape's border width or border pattern, select the freehand shape and click the appropriate setting in the Tools window.

- For information on moving, stretching, smoothing, flipping, and reshaping a freehand shape, see *Graphics Editing Overview* in Chapter 2.
- Add and delete points from a freehand line with the Reshape command (see Chapter 3).
- It is often easier to draw the desired freehand shape using the Polyline tool together with the Reshape and Smooth commands.
- You cannot fill a freehand shape with a pattern. To draw an irregular filled shape, use the Polygon tool and create a rough approximation of the shape. Then use the Reshape command to refine and add points to the shape. Use the Smooth command for further refinement.

Line

To draw a line:


Line tool

- Click the Line tool with the left mouse button or press Esc 1 l (that is, the "one" key followed by the unshifted L key) while in the document window.
- Check and, if needed, change the Widths and Borders settings in the Tools window.
- Move the arrow pointer to where the line should begin. If the rulers are displayed, tick marks on each ruler show the position of the arrow pointer to aid in accurate placement.
- Press and hold down the left mouse button.
- While holding down the mouse button, move the arrow pointer until the line stretches to the desired endpoint. Dimension information appears in the status line as the object is stretched.
- Release the mouse button.

Notes:

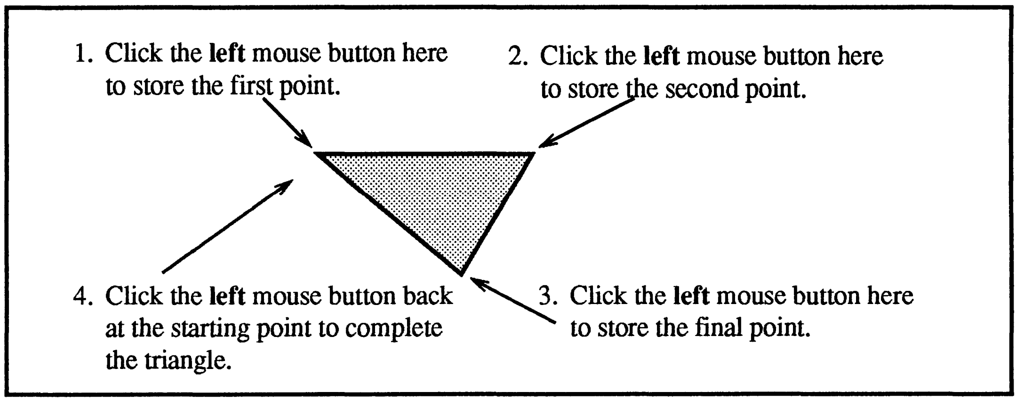
- The Movement settings in the Tools window affect the way that lines are drawn. For example, to draw a straight horizontal line, click H. Only before drawing the line.
- To change a line's border width or border pattern, select the line and click the appropriate setting in the Tools window.
- For information on moving, stretching, and flipping a line, see *Graphics Editing Overview* in Chapter 2.
- To draw a dashed line, use a nonblack Borders setting.
- To convert a line to a polyline, add points to it with the Reshape command (see Chapter 3).

Polygon

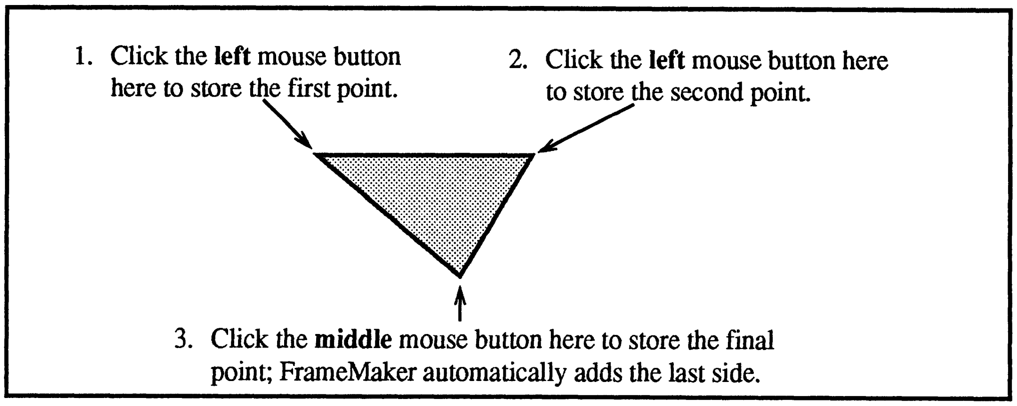
To draw a closed, fillable polygon:

Polygon tool

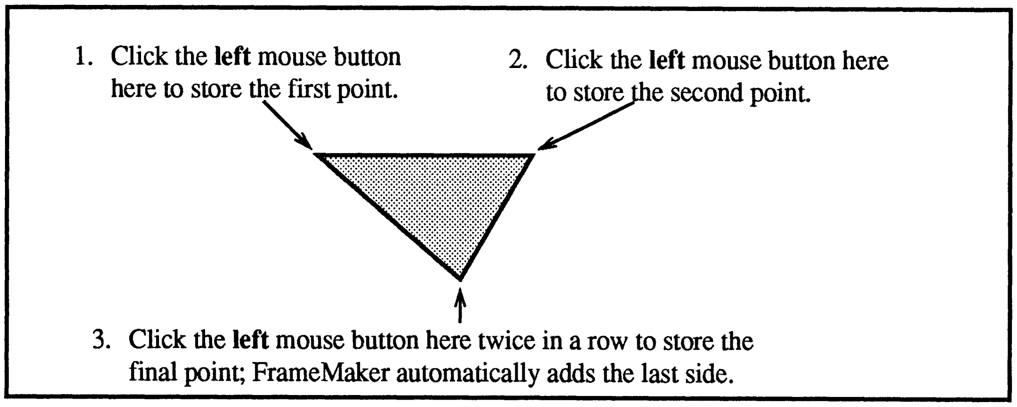
- Click the Polygon tool with the left mouse button or press Esc 1 p g while in the document window.
- Check and, if needed, change the Widths, Fills, or Borders settings in the Tools window.
- Move the arrow pointer to where one point on the polygon should be. If the rulers are displayed, tick marks on the rulers show the position of the arrow pointer to aid in accurate placement.
- Click the left mouse button to store the polygon's starting point.
- Move the arrow pointer until one side of the polygon is stretched to the desired endpoint (no need to keep the mouse button pressed). Dimension information appears in the status line as the line is stretched.
- Click the left mouse button to store the second point. This completes the first side of the polygon.
- Move the mouse to form the second side; when the side is properly positioned, click the left button.
- Repeat the previous step to add additional sides to the polygon, until all but the final side has been drawn. There can be an arbitrarily large number of sides, and they can cross each other to form arbitrarily complex shapes.
- There are three ways to complete the drawing of a polygon. Use whichever one seems most natural to you:
  1. You can move the arrow pointer back to the starting point and click with the left mouse button:


1. Click the **left** mouse button here to store the first point.
2. Click the **left** mouse button here to store the second point.
4. Click the **left** mouse button back at the starting point to complete the triangle.
3. Click the **left** mouse button here to store the final point.

2. You can click with the middle mouse button on the final point instead of using the left mouse button:


1. Click the **left** mouse button here to store the first point.
2. Click the **left** mouse button here to store the second point.
3. Click the **middle** mouse button here to store the final point; FrameMaker automatically adds the last side.

3. You can click with the left mouse button twice in a row on the final point:

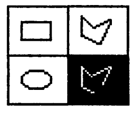

1. Click the **left** mouse button here to store the first point.
2. Click the **left** mouse button here to store the second point.
3. Click the **left** mouse button here twice in a row to store the final point; FrameMaker automatically adds the last side.

Notes:

- To change a polygon's border width or its border or fill pattern, select the polygon and click the appropriate setting in the Tools window.
- For information on moving, stretching, smoothing, flipping, and reshaping a polyline, see *Graphics Editing Overview* in Chapter 2.
- To add or delete points from a polygon, use the Reshape command (see Chapter 3).
- To create a regular polygon, such as a pentagon or hexagon, draw a rectangle of the desired size, select it, and use the Set # Sides command (see Chapter 3).

Polyline

To draw a polyline (a set of connected, unclosed lines):


Polyline tool

- Click the Polyline tool with the left mouse button or press Esc 1 p l (that is, the "one" key, the unshifted P key, and the unshifted L key) while in the document window.
- Check and, if needed, change the Widths and Borders settings in the Tools window.
- Move the arrow pointer to where the polyline should begin. If rulers are displayed, tick marks on the rulers show the position of the arrow pointer to aid in accurate placement.
- Click the left mouse button to store the polyline's starting point.
- Move the arrow pointer until one side of the polyline is stretched to the desired endpoint (no need to keep the mouse button pressed). Dimension information appears in the status line as the line is stretched.
- Click the left mouse button to store the second point, completing the first line segment of the polyline.
- Move the mouse to form the second segment; when the segment is properly positioned, click the left button.
- Repeat the previous step to add additional segments to the polyline until all but the final segment has been drawn. There can be an arbitrarily large number of segments, and they can cross each other to form arbitrarily complex shapes.
- To draw the final segment, move the arrow pointer to the desired endpoint and click the middle mouse button, or click the left mouse button two times in a row without moving the mouse.

Notes:

- To change a polyline's border width or border pattern, select the polyline and click the appropriate setting in the Tools window.
- For information on moving, stretching, smoothing, flipping, and reshaping a polyline, see *Graphics Editing Overview* in Chapter 2.
- To add or delete points from a polyline, use the Reshape command (see Chapter 3).

Rectangle

To draw a rectangle:

Rectangle tool

- Click the Rectangle tool with the left mouse button or press Esc 1 r while in the document window.
- Check and, if needed, change the Widths, Fills, and Borders settings in the Tools window.
- Move the arrow pointer to where one corner of the rectangle should begin. If rulers are displayed, tick marks on the rulers show the position of the arrow pointer to aid in accurate placement.
- Press and hold down the left mouse button.
- Move the arrow pointer while holding down the mouse button until the rectangle is stretched to the desired size. Dimension information appears in the status line as the rectangle is stretched.
- Release the mouse button.

Notes:

- To change a rectangle's border width or its border or fill pattern, select the rectangle and click the appropriate setting in the Tools window.
- For information on moving, stretching, and smoothing a rectangle, see *Graphics Editing Overview* in Chapter 2.

Square

To draw a square:

Square tool

- Click the Square tool with the left mouse button or press Esc 1 s while in the document window.
- Check and, if needed, change the Widths, Fills, and Borders settings in the Tools window.
- Move the arrow pointer to where one corner of the square should begin. If rulers are displayed, tick marks on the ruler show the position of the arrow pointer to aid in accurate placement.
- Press and hold down the left mouse button.
- Move the arrow pointer while holding down the mouse button, until the square is stretched to the desired size. Dimension information appears in the status line as the object is stretched.
- Release the mouse button.

Notes:

- To change a square's border width or its border or fill pattern, select the square and click the appropriate setting in the Tools window.
- You can stretch a square to form a rectangle. For information on moving, stretching, and smoothing a square, see *Graphics Editing Overview* in Chapter 2.

TextLine

To place a TextLine on a page:

TextLine tool

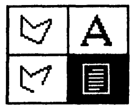
- Click the TextLine tool with the left mouse button or press Esc 1 t l (that is, the "one" key, the unshifted T key, and the unshifted L key) while in the document window.
- Move the arrow pointer to where the text left, center, or right should be (you choose the proper alignment next). If rulers are displayed, tick marks on the rulers show the position of the arrow pointer to aid in accurate placement.
- Click the left mouse button to place an insertion point in the document.
- When you create a TextLine, its left edge and baseline align with the point where you clicked the mouse. To change the alignment, use one of the keyboard alignment commands (Esc j l for left, Esc j c for center, or Esc j r for right). You can also select the TextLine and use the Align command in the Tools window.
- Type in the desired text. Pressing Return moves the insertion point down to the start of a new line and begins a new TextLine object.

Notes:

- The TextLine is created using the *current default font* (the font that appears in the Fonts dialog when no text is selected). To set up the desired font before creating the TextLine, choose the Fonts command with no text selected. To change the font of text in an existing TextLine, select the text and choose the Fonts command.
- Each character in a TextLine can have different font characteristics.
- Unlike TextRects, which are usually opaque, TextLines have transparent white space between each character, allowing text to be placed over graphics without obscuring the area behind the text.
- For information on moving the TextLine, see *Graphics Editing Overview* in Chapter 2.
- For a discussion of when to use TextLines and when to use TextRects, see *Text Objects* in Chapter 2.

TextRect

To place a TextRect on a page:


TextRect tool

- Click the TextRect tool with the left mouse button or press Esc 1 t r while in the document window.
- Check and, if needed, change the Widths, Fills, and Borders settings in the Tools window.
- Move the arrow pointer to where one corner of the TextRect should be. If rulers are displayed, tick marks on the rulers show the position of the arrow pointer to aid in accurate placement.
- Press and hold down the left mouse button.
- Move the arrow pointer while holding down the mouse button until the TextRect is stretched to the desired size. Dimension information appears in the status line as the TextRect is stretched.
- Release the mouse button.

Notes:

- To change a TextRect's border width or its border or fill pattern, select the rectangle and click in the appropriate setting in the Tools window.
- For information on moving and stretching a TextRect, see *Graphics Editing Overview* in Chapter 2.
- Normally, a TextRect has Borders set to None and Fills set to white; however, any other setting can also be used. When typing into a TextRect whose Fills pattern is nonwhite, the TextRect's fill may look incorrect on the screen, and objects behind a transparent TextRect may appear incorrectly. To improve performance, FrameMaker does not always keep the screen display correct. To correct the screen display, use the Redisplay command in the Window menu or press Control-l (that is, the unshifted L key).
- The TextRect is created using the *current default font* (the font that appears in the Fonts dialog when no text is selected). To set up the desired font before creating the TextRect, choose the Fonts command with no text selected. To change the font of text in an existing TextRect, select the text and choose the Fonts command. Each character in a TextRect can have different font characteristics.
- To type text into the TextRect, move the arrow pointer into the TextRect and click with the middle mouse button.
- For a discussion of when to use TextLines and when to use TextRects, see *Text Objects* in Chapter 2.

Keep Tool

The Keep Tool setting controls what happens immediately after you draw an object. When the setting is enabled, the arrow pointer retains its drawing pointer shape, and you can continue to draw additional objects using the same drawing tool. When disabled, the arrow pointer returns to an arrow shape. Keep Tool does not work for the TextLine tool.

Fills

The Tools window provides 16 fill patterns. Arcs, squares, circles, rectangles, ellipses, polygons, TextRects, and frames can be filled with a pattern. When you select a drawing tool, the fill that will be used for objects drawn with the tool appears outlined in the Tools window.

To change the fill pattern of an object using the mouse:

- Select the appropriate object(s).
- Move the arrow pointer into the Fills area in the Tools window.
- Click the left mouse button on the desired fill pattern.

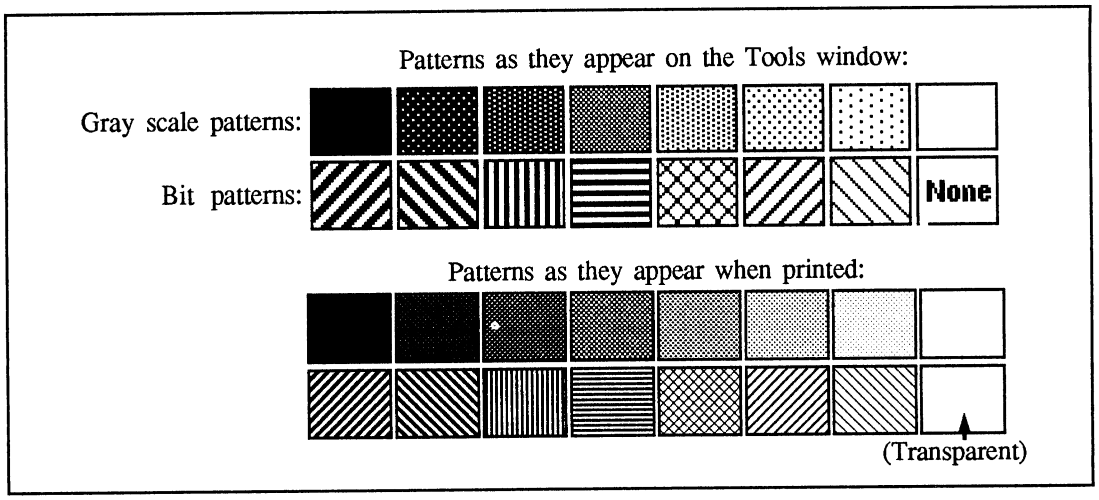
To change the fill pattern from the keyboard:

- Select the appropriate object(s).
- Press Esc 0 f (the second key is the zero key) to set the fill to the first fill pattern (black).
- Press Esc + f to increment the fill pattern to the next available fill pattern.
- Press Esc - f to decrement the fill pattern to the previous fill pattern.

Notes:

- The fill pattern of all selected objects is set when you click a Fills setting in the Tools window. Be certain that the correct objects are selected before clicking a Fills setting.
- The sixteen fill patterns are user-definable. See Appendix D, *Customizing FrameMaker,* for information on customizing fill patterns.
- The standard fill patterns provided with FrameMaker are divided into two types: gray scale patterns (the upper row) and bit patterns (the lower row). Gray scale patterns print much more quickly than bit patterns and usually produce more pleasing results. The illustration below shows how the fill

patterns appear on the screen and when printed on a 300-dpi PostScript printer.


Patterns as they appear on the Tools window:
Gray scale patterns:
Bit patterns:
None
Patterns as they appear when printed:
(Transparent)

Borders

The Tools window provides 16 border patterns. All objects except TextLines can be drawn using a border pattern. When you select a drawing tool, the border pattern that will be used for objects you draw with the tool becomes outlined in the Tools window.

To change the border pattern of one or more objects using the mouse:

- Select the appropriate object(s).
- Move the arrow pointer into the Borders area in the Tools window.
- Click the left mouse button on the desired pattern.

To change the border pattern from the keyboard:

- Select the appropriate object(s).
- Press Esc 0 p (the second key is the zero key) to set the fill to the first border pattern (black).
- Press Esc + p to increment the border pattern to the next available border pattern.
- Press Esc - p to decrement the border pattern to the previous border pattern.

Notes:

- The border pattern of all selected objects is set when you click a Borders setting in the Tools window. Be certain that the correct objects are selected before clicking a Borders setting.
- The sixteen border patterns are user-definable. See Appendix D, *Customizing FrameMaker,* for information on customizing fill patterns.
- The standard border patterns provided with FrameMaker are divided into two types: gray scale patterns (the upper row) and bit patterns (the lower row). Gray scale patterns print much more quickly than bit patterns and usually produce more pleasing results. See the pattern illustration in the previous section.

Widths

The Tools window provides four line widths. All objects except TextLines can be drawn using a border width.

To change the line width of one or more objects using the mouse:

- Select the appropriate object(s).
- Move the arrow pointer into the Widths area in the Tools window.
- Click the left mouse button on the desired width.

To change the border width from the keyboard:

- Select the appropriate object(s).
- Type Esc 0 w (the second key is the zero key) to set the border to the thinnest line width.
- Type Esc + w to increment the line width.
- Type Esc - w to decrement the line width.

Notes:

- The line width of all selected objects is set when you click a Widths setting in the Tools window. Be certain that the correct objects are selected before clicking a Widths setting.
- To have no border, select the thinnest width and then select a Borders setting of None.
- The line widths shown on the screen do not exactly match the widths used when printing. In general, lines are thicker on the screen than they are when printed.
- You can change the thickness of the line widths for both the screen and the printer. See Appendix D, *Customizing FrameMaker*, for more information. The standard values are as follows:

Line Width on Screen	Line Width when Printed
1	.5
2	1
4	3
6	4
(pixels)	(points)

The fmbook Program

Introduction

`fmbook` is a powerful extraction and formatting program that you use to:

- Create a standard table of contents or index using the format supplied with FrameMaker.
- Create a standard table of contents or index using your own format. You can specify your own:
  - Margin and tab settings,
  - Page header, footer, and column specifications,
  - Paragraph formats, and
  - Master page items.

  For an index, you can also control:
  - The sort order of index entries,
  - Entry groupings and group titles, and
  - Page reference separators.
- Create a customized list, such as a list of figures or a list of tables.
- Create custom derivations or use the extracted data for other purposes (for example, in a database application).
- Print an entire book without starting FrameMaker.

`fmbook` can extract and format information from single documents as well as from multiple documents that comprise a single book. For more efficient editing, you can divide books into smaller files, and then use `fmbook` to number pages, compile the index, and print the book as a unit.

fmbook Overview

To create an index, table of contents, or other list, `fmbook` reads FrameMaker documents, extracts text and page number information from those documents, formats and sorts that information, and produces a new file (which we call a *derived document*). Generally, the file it creates is in Maker format so you can open it, edit it, and print it with FrameMaker.

`fmbook`'s extraction process is based on paragraph tags and markers. `fmbook` creates the standard table of contents by extracting all paragraphs from a document whose tag is chapter, section, subsection, or appendix. It creates an index by extracting all marker text associated with the "index" marker type.

As it is executing, `fmbook` uses a number of auxiliary files normally stored in the directory `.makerinit/bookdir` in the directory in which FrameMaker is installed. These files specify the paragraphs or markers to be extracted and the format of the resulting derived document. To customize the format of your table of contents or index, extract different tags or markers, or create other kinds of derived documents (such as a list of figures or tables), you'll need to edit these auxiliary files. Later sections in this chapter explain how to do this.

You can use `fmbook`, creating your own table of contents or index, after reading just a few pages of this chapter. As you modify your design or need to customize indexes and lists, you can read additional sections in this chapter.

Single-File Documents and Books

`fmbook` can create an index, table of contents, or other list, for one document at a time or for several FrameMaker files that you consider to be a single book. For example, you may be writing a book divided into FrameMaker files named `ChapterOne.doc`, `ChapterTwo.doc`, etc. The instructions in the next section describe how to prepare your documents for `fmbook`, whether you are working with a single-file document or with a book that consists of several documents. *Creating a Book File,* later in this chapter, describes how to specify the components of your book.

Preparing Documents

Naming Documents

We suggest that you suffix your FrameMaker filenames with `.doc`.

If you choose not to suffix your FrameMaker filenames with `.doc`, you should avoid putting version information in the filename suffix, for the following reason: `fmbook` names a standard table of contents or index by adding the suffixes `TOC.doc` and `Index.doc` to the filename, truncated to the last period (if any) after the last slash (if any). For example, if your document is named `mydoc.version1`, `fmbook` truncates the name to `mydoc.` and then adds the suffix `TOC.doc`. The resulting table of contents is named `mydocTOC.doc`. If you later create a table of contents for `mydoc.version2`, the new table of contents will have the same name as the old one and will overwrite it.

Numbering the Pages of Documents

If you want page numbers to be printed on each page of a document, use FrameMaker's Headers and Footers command to set up a header or footer string containing the number symbol (#). If you're working with a multiple-file book, set up the header or footer strings appropriately for each document. For example, if you have a two-chapter book with 40 pages in each chapter, you may want to number chapter one's pages from 1 through 40 and chapter two's pages from 41 through 80; or you may want to number the pages 1-1 through 1-40 and 2-1 through 2-40. For more information, see the Headers and Footers command in Chapter 3.

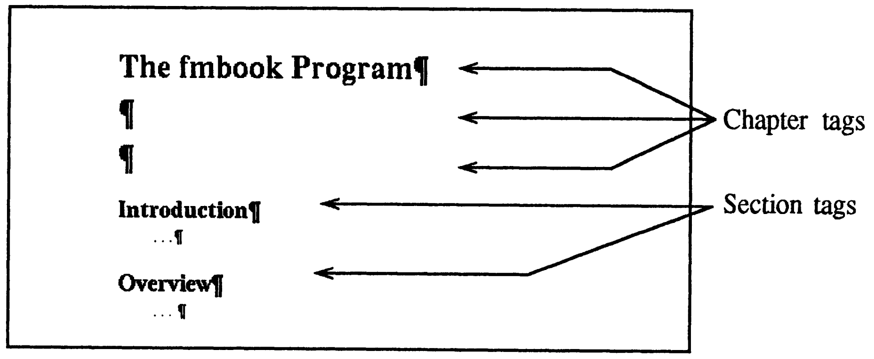
If you intend to print the book with `fmbook` rather than with FrameMaker, and you want to number the pages consecutively throughout the book, you don't need to specify the correct value for each document's starting page number with the Headers and Footers command. Instead, you tell `fmbook` how to number the pages of the book. For more information, see *Controlling Page Numbering for a Book* later in this chapter.

Preparing Documents for a Table of Contents

`fmbook` creates a standard table of contents by extracting paragraphs with the tags of chapter, section, subsection, and appendix from the main document file and listing them along with their page numbers. So make sure that:

- Each chapter, section, subsection, and appendix heading in the document has the appropriate tag.
- Each heading is a single paragraph. In particular, you won't get the desired results if you have a two-line heading that spans two paragraphs. If you want to force a line break in your heading, hold down the Alt key and then press Enter. This forces a new line, but does not begin a new paragraph.

- No other paragraphs in the document have these tags. For example, suppose that you use two paragraph returns to get the extra spacing between a heading and its following text, as shown below:



Sample Document

If the empty paragraphs have the same tag name as the heading above them, `fmbook` interprets them as headings and produces extra entries in the table of contents:

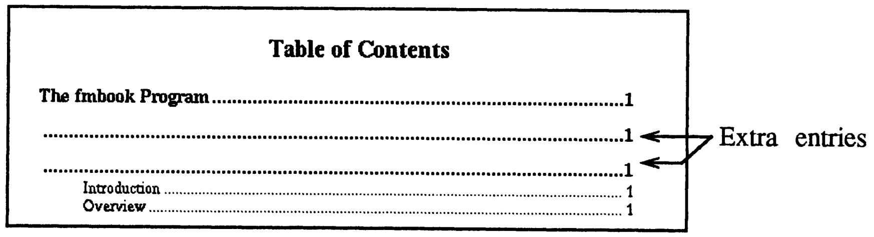


Table of Contents

The fmbook Program.......................................1
....1
....1
Introduction.......................................1
Overview.......................................1

Extra entries

Resulting Table of Contents

To add space below a heading, increase the Space After value in the Paragraph dialog instead of using an empty paragraph.

For more information about tagged paragraphs, see the Paragraphs command in Chapter 3.

If you want to extract paragraphs with tag names other than chapter, section, subsection, and appendix, see *Customizing a Table of Contents* later in this chapter.

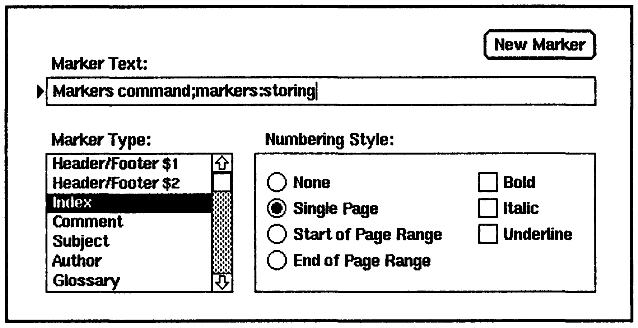
Preparing Documents for an Index

Before `fmbook` automatically extracts, sorts, and formats the indexing information from a document, you must indicate which words and concepts are indexed by inserting index markers in the text.

Insert a New Marker

To insert a marker at some point in text:

1. Display the Markers window using the Markers command on the Edit menu.



Marker Text: Markers command;markers:storing

Marker Type: Header/Footer $1, Header/Footer $2, [✓ Index], Comment, Subject, Author, Glossary

Numbering Style: ○ None [✓ ● Single Page] ○ Start of Page Range ○ End of Page Range ☐ Bold ☐ Italic ☐ Underline

Markers Window

2. In the Marker Type area, select Index.
3. Put the insertion point in your document where you want to insert the marker.
4. Move the arrow pointer into the Markers window and fill in the marker values (described below).
5. Click on the New Marker button. If Text Symbols display is on, a marker symbol (T) appears in the text at the insertion point to show where you stored the new marker.

There are two shortcuts for filling in the marker text for a new marker:

- With the Markers window displayed, highlight in the text a word or group of words that does not already contain a marker. When you release the mouse button, the highlighted text will automatically appear in the Markers window.
- In the text, put the marker at the start of a word and leave the marker text box empty. When `fmbook` finds such a marker, it uses the word to the right of the marker.

Edit an Existing Marker

You can edit both a marker's text and its options. To edit an existing marker:

1. Highlight the marker symbol (it doesn't matter if you also highlight some of the surrounding text). FrameMaker shows the contents of that marker in the Markers window, and the New Marker button changes to Edit Marker.
2. Edit the contents of the Markers window.
3. Click on the Edit Marker button to store the new values.

Delete a Marker

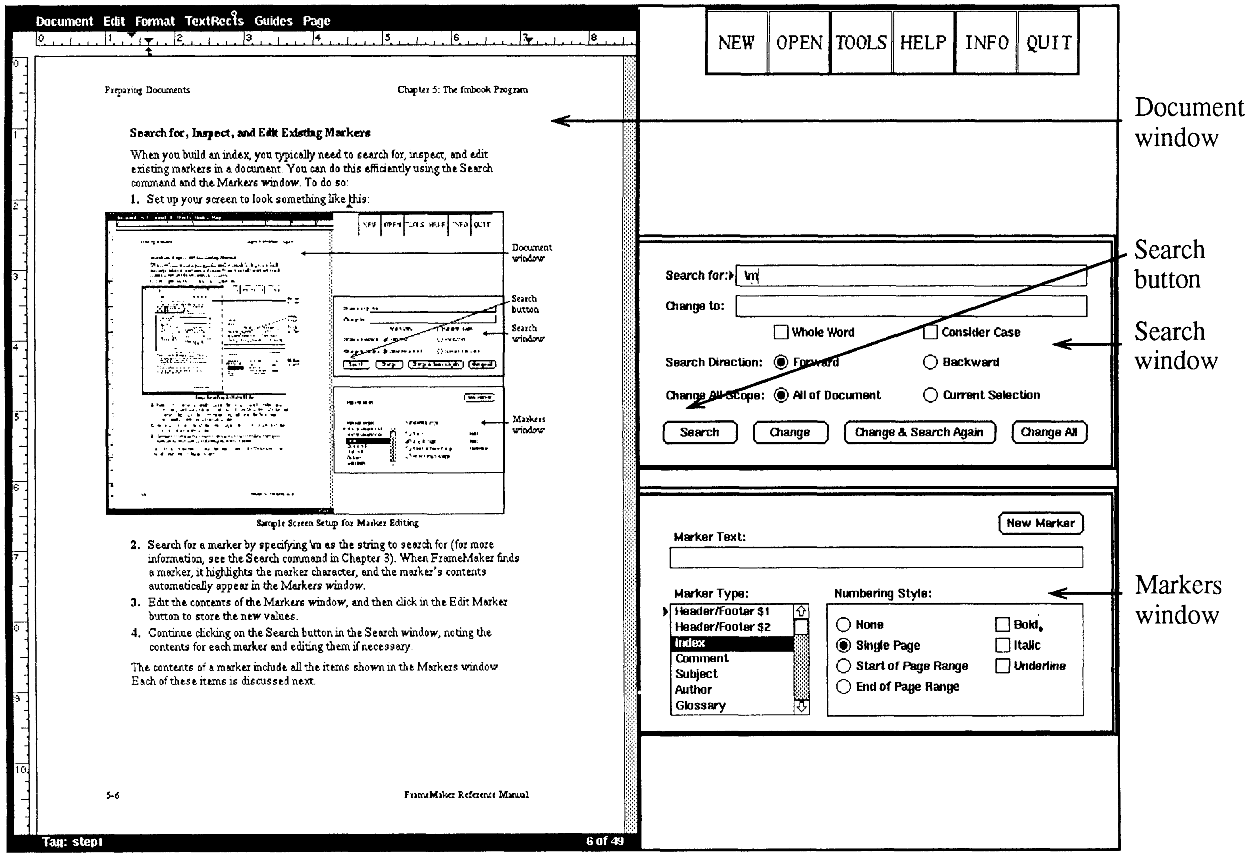
To delete a marker, highlight the marker symbol and press the Delete key.

To highlight a marker without highlighting any surrounding text, search for the string `\m`. For more information, see the Search command in Chapter 3.

Search for, Inspect, and Edit Existing Markers

When you build an index, you typically need to search for, inspect, and edit existing markers in a document. You can do this efficiently using the Search command and the Markers window. To do so:

1. Set up your screen to look something like this:



Search for: \m

Change to:

☐ Whole Word ☐ Consider Case

Search Direction: ● Forward ○ Backward

Change All Scope: ● All of Document ○ Current Selection

Marker Text:

Marker Type: Header/Footer $1, Header/Footer $2, **Index** (selected), Comment, Subject, Author, Glossary

Numbering Style: ○ None ● Single Page ○ Start of Page Range ○ End of Page Range ☐ Bold ☐ Italic ☐ Underline

Sample Screen Setup for Marker Editing

2. Search for a marker by specifying \m as the string to search for (for more information, see the Search command in Chapter 3). When FrameMaker finds a marker, it highlights the marker character, and the marker's contents automatically appear in the Markers window.
3. Edit the contents of the Markers window, and then click in the Edit Marker button to store the new values.
4. Continue clicking on the Search button in the Search window, noting the contents for each marker and editing them if necessary.

The contents of a marker include all the items shown in the Markers window. Each of these items is discussed next.

Marker Type

`fmbook` creates a standard index by extracting the contents of markers whose marker type is "index." You can create indexes from other existing marker types using techniques described later in this chapter. If you want to create new marker types, see Appendix D.

Marker Text

The marker text determines what appears in the index and how it is sorted. The table below shows the available options (assuming that a marker is inserted on page 10 of a document):

Index Entry	Marker Text	Explanation
coffee 10	`coffee`	Simple entry.
tea english 10	`tea:english`	Multiple levels allowed (see the next paragraph); colon separates levels of the index entry.
english tea 10 tea 10	`tea;english tea`	Multiple entries allowed; semicolon separates entries.
68000 10 (entry under S)	`68000[sixtyeightthousand]`	Entry is sorted as if it were the string in brackets; the string in brackets can contain multiple entries separated by semicolons.
68000 10 (entries under both Numerics and S)	`68000[;sixtyeightthousand]`	Semicolon immediately following left bracket causes regular entry to appear as well as other specified entries.

You can also indicate multiple levels within the square brackets, so the following marker text:

68000[;sixtyeightthousand;processor:sixtyeightthousand];microprocessor

generates these index entries:

68000 10	(sorted as a numeric)
68000 10	(sorted as "sixtyeightthousand")
68000 10	(sorted as "processor:sixtyeightthousand")
microprocessor 10	

The standard index format allows three levels, but you can create a custom format that allows any number of levels. For more information, see *Customizing an Index* later in this chapter.

If you want to print a colon, semicolon, or bracket character within an index entry, precede the character with a backslash (\ ) to override its special meaning. For example, to make "2:1 ratio" an index entry, enter **`2\:1 ratio`** as the marker text.

Numbering Style

The numbering style options determine how the page number appears in an index entry.

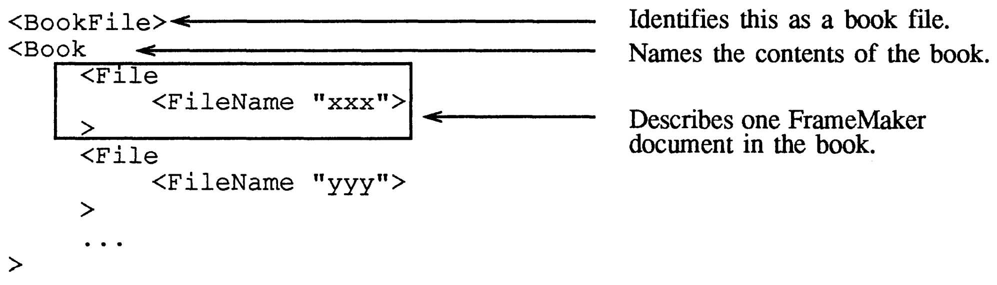
If you want:	Then do this:
An entry with a single page number	Select a Numbering Style of Single Page.
An entry such as "Mock, see Ovaltine"	Use a marker whose text is "Mock, see Ovaltine" and whose Numbering Style is None.
An entry such as "Ducks 23-37"	On page 23, put an index marker whose text is "Ducks" and whose Numbering Style is Start of Page Range; on page 37, put another index marker whose text is "Ducks" and whose Numbering Style is End of Page Range. If fmbook finds an unmatched start or end marker, it creates an entry with ?? as a page reference.
Different kinds of page references	Use the Bold, Italic, and Underline options. For example, you could use the Bold option for the marker representing the principal reference; the index entry would then print the main page reference in bold text and the other page references in regular text.

Creating a Book File

When you want `fmbook` to treat several FrameMaker files as a single book, you create a book file. The book file specifies the files that make up the book, the order in which `fmbook` will process them, and how `fmbook` will number the pages. If you want to create a table of contents or index for a single document, skip this section and go directly to *Creating a Standard Table of Contents and Index* later in this chapter.

Defining a Book

To define a book, create a file like the following (using FrameMaker or any UNIX editor):



```
<BookFile>
<Book
    <File
        <FileName "xxx">
    >
    <File
        <FileName "yyy">
    >
    ...
>
```

Identifies this as a book file.
Names the contents of the book.
Describes one FrameMaker document in the book.

The order of <File...> statements determines the order in which `fmbook` assembles the individual documents into a book. For example, the book file below defines a book with three chapters named Ch1, Ch2, and Ch3.

```
<BookFile>
<Book  <------------------------------  Each <Book... has a matching >.
    <File
        <FileName "Ch1">
    >
    <File <-----------------------  Each <File... has a matching >.
        <FileName "Ch2">
    > <---------------------------------------------
    <File
        <FileName "Ch3">
    >
> <------------------------------------------------------
```

Caution: A book file must be an ASCII text file. If you create or edit your book file using FrameMaker, remember to save it using the Text Only format in the Save dialog.

Controlling Page Numbering for a Book

You can tell `fmbook` how to number the pages in your book. For example, you may want a preface to be numbered in lowercase roman numerals, with the first chapter in the book beginning at page 1, and the other chapters beginning with the next odd page number.

Starting Page Number

To control the starting page number of a file, add a <StartPageNum...> statement to each <File...> statement as needed. For example:

```
<File
    <FileName "Preface.doc">
    <StartPageNum 1>    <--------  Preface.doc's page numbers
>                                  begin with 1.
```

The choices for <StartPageNum...> are:

Statement	Meaning
<StartPageNum *n*>	Starts numbering with the value specified.
<StartPageNum NextOdd>	Starts numbering with the next odd number. If used with the first file in a book, this is equivalent to <StartPageNum FromDocument>.
<StartPageNum NextEven>	Starts numbering with the next even number. If used with the first file in a book, this is equivalent to <StartPageNum FromDocument>.
<StartPageNum NextPageNum>	Starts numbering with the next number. If used with the first file in a book, this is equivalent to <StartPageNum FromDocument>.
<StartPageNum FromDocument>	Uses the Start Page # value found in the FrameMaker file's Headers and Footers dialog.

A <StartPageNum...> statement will remain in effect until it is changed in a subsequent <File...> statement. For example, if you want each chapter of your book to begin on an odd page, you need to include a <StartPageNum NextOdd> statement only in the first chapter's <File...> statement; `fmbook` then numbers the first chapter, and subsequent chapters, using NextOdd.

The initial default is <StartPageNum FromDocument>.

To print page numbers on each page of a document, you must use FrameMaker's Headers and Footers command to set up a header or footer string containing a number symbol (#). For more information, see *Numbering the Pages of Documents* earlier in this chapter.

Page Numbering Style

To control the page numbering style of a file, add a <PageNumStyle...> statement to each <File...> statement as needed. For example:

```
<File
     <FileName "Preface.doc">
     <StartPageNum 1>
     <PageNumStyle lcroman>   <-----------
>
```

`Preface.doc`'s pages are numbered *i*, *ii*, *iii*, *iv*, and so forth.

The choices for <PageNumStyle...> are:

Statement	**Meaning**
<PageNumStyle arabic> *or* <PageNumStyle numeric>	1, 2, 3,...
<PageNumStyle lcroman>	*i*, *ii*, *iii*, *iv*,...
<PageNumStyle ucroman>	I, II, III, IV,...
<PageNumStyle lcalpha>	a through z, then aa, ab,..., zz, aaa, and so forth
<PageNumStyle ucalpha>	A through Z, then AA, AB, and so forth
<PageNumStyle FromDocument>	Uses the Page Number Style value found in the FrameMaker file's Headers and Footers dialog.

Like <StartPageNum...>, a <PageNumStyle...> statement remains in effect until changed in a subsequent <File...> statement.

The initial default is <PageNumStyle FromDocument>.

If you have point pages in your document, you can use a <PointPageNumStyle...> statement, in the same way as <PageNumStyle...>, to control the style of point pages. For more information about point pages, see the Freeze Pagination command in Chapter 3.

Page Number Prefix and Suffix

To control the prefix and suffix for the page numbers in a file, add <PageNumPrefix...> and <PageNumSuffix...> statements to each <File...> statement as needed. For example:

```
<File
     <FileName "Ch3.doc">
     <StartPageNum 1>
     <PageNumStyle arabic>
     <PageNumPrefix "[3-"
     <PageNumSuffix "]"
>
```

Table of contents and index entries for Ch3 have the following page numbers: [3-1], [3-2], [3-3], and so forth.

Note that the <PageNumPrefix...> and <PageNumSuffix...> statements affect only the page numbers that appear in table of contents and index entries. To print the document's page numbers on each page with the prefix and suffix, you put those characters in the appropriate header and/or footer.

Using fmbook

You can run `fmbook` either in an interactive or a noninteractive mode. This section describes interactive use of the program; noninteractive use is described in *Using the fmbook Command Line Interface* later in this chapter.

Starting fmbook

To start `fmbook`, type **`fmbook`** in a shell window. After `fmbook` loads its fonts, the following appears:

```
fmbook 1.3  (using /usr/frame/.makerinit)

Type the word HELP and press the
Enter key for a list of commands.

Command?
```

Note: If you plan to use `fmbook` several times in a sitting, we recommend that you leave it running. Also, you can reduce unnecessary typing if you start `fmbook` from the directory containing the document files you're using. If you are editing the document or book files at the same time, remember to save any document or book files that `fmbook` uses as often as necessary to ensure that `fmbook` is using the current version of those documents.

Verify that fmbook is on your UNIX path by typing this command: *echo $PATH.* If `fmbook` is not on your UNIX path. If not, you must either give a full pathname to start `fmbook` or change your working directory to *install_dir*`/bin` before starting `fmbook`,or add `fmbook` to your path. (*Install_dir* is the directory in which FrameMaker is installed.)

Getting Help

While working in `fmbook`, you can always get a complete list of commands by typing **`help`** or `?`.

```
Command? help
fmbook 1.10
(case is not significant in built-in commands)
Command                Meaning
-------------------- -----------------------------------
 Show or Status        Show all scheduled actions
 Book [name]           Start working with a new .book or .doc file
                       (or, just enter a file name)
 Go                    Do all scheduled actions
 Done                  Do all scheduled actions, then exit program
 Quit                  Forget everything and exit program
 Clear                 Unschedule all printing and creating actions
 Reset                 Reset all print options
 Terse                 Turn off status messages
 Verbose               Turn on  status messages
 Help                  Prints this help message
 TOC                   Schedule creation of standard table of
                       contents
 Index                 Schedule creation of standard index
 Copies nnn            Set number of copies for printing
 Scale nnn             Set scaling value    for printing
 PrintFile  name       Set output file name for printing
 Printer   [name]      Set device name used for printing
   Create  [docname] Schedule   creation [of just docname]
 NoCreate  [docname] Unschedule creation [of just docname]
   Print   [docname] Schedule   printing [of just docname]
 NoPrint   [docname] Unschedule printing [of just docname]
 Collate   Yes/No      Turn on/off collation for printing
 LowRes    Yes/No      Turn on/off lowres printing
 Reverse   Yes/No      Turn on/off reverse printing order
 EvenPages Yes/No      Enable/Disable printing of even pages
 OddPages  Yes/No      Enable/Disable printing of odd pages
 KeepPrintSettings Yes/No       Enable/Disable forced print
                                option settings
```

The actual command list displayed on your system may be different since it is possible to add custom commands at individual sites. For more information, see *Creating a Custom List and Creating Alternative Indexes* later in this chapter. For information on changing the list of commands, see Appendix D.

The individual commands shown above are explained in detail later in this chapter.

Using fmbook

Throughout this chapter, we use sample filenames when describing `fmbook` (for example, `myfile.doc`). You, of course, will use your own filenames. In addition, we've chosen to end all FrameMaker filenames with `.doc`, but this is not a requirement.

To use `fmbook`:

1. Tell `fmbook` what file you want to work with.

   ```
   Command? book myfile.doc
   (Doc myfile.doc) <----------
   Command?
   ```

   `fmbook` confirms the file and indicates that this is a single-file document, not a book.

   At the command prompt, you can type just the filename, omitting the command **book**.

2. Tell `fmbook` what action(s) to perform for the file (this is called *scheduling actions*).

   ```
   Command? index
   Create myfileIndex.doc
   Command? toc
   Create myfileTOC.doc
   Command?
   ```

3. Tell `fmbook` to carry out all the scheduled actions.

   ```
   Command? go
   ```

Note that `fmbook` saves the commands and executes them only when you give the **go** command. At that time, `fmbook` carries out all scheduled actions and then prompts you for the next command; `fmbook` doesn't clear the actions you scheduled. In this example, if you type **go** a second time, `fmbook` creates the index again; if you schedule another action, such as **toc**, and then type **go**, `fmbook` will create a table of contents *and* an index.

To display a list of all scheduled actions, type **show** at the command prompt. `fmbook` will display a list of the scheduled actions, along with various printing settings (similar to those found in the FrameMaker's Print dialog).

To clear all user-scheduled actions before continuing, type **clear** at the command prompt.

To exit the program, type either **done** or **quit**. The command **done** causes `fmbook` to carry out all scheduled actions before exiting the program; **quit** causes `fmbook` to exit without carrying out any of the scheduled actions.

Your system administrator may have changed the default actions of `fmbook`'s **index** and **toc** commands, so you may wish to consult your local documentation. Your system administrator may also have added commands that allow you to derive other documents automatically, such as a list of tables and a list of figures. Once again, check the local documentation to see which ones are supported. See *Creating a Custom List* later in this chapter, for information about creating such commands.

Printing a Book

To print a book using `fmbook`, you identify the book and then issue one or more commands to control the printing:

1. Start `fmbook` as usual.
2. Specify the book file. For example, if the name of the book file is `Refman.book`, you would type the command as follows:

```
Command? book Refman.book
(Book Refman.book) <-------
Command?
```

fmbook confirms the file and indicates that this is a book, not a single-file document.

3. Use one of the first three commands to print the file or files you want:

Command	**Effect**
`print`	Schedules all files in the book for printing, except for those specified in any following **`noprint`** commands.
`print` ***`filename`***	Schedules the specified file for printing.
`print` ***`filename x y`***	Schedules pages *x* through *y* of the specified file for printing.
`noprint`	Cancels printing of all files in the book.
`noprint` ***`filename`***	Cancels printing of the specified file.

Note that you need to specify the book file (not a single document file) to `fmbook` even though you may want to print just one file in the book. You need to do so because `fmbook` has to know the complete set of files that make up the book in order to assign the right page numbers.

Notice that you can use wild card characters, such as the asterisk (*) and period (.), in the filenames for the **`print`** and **`noprint`** commands. For more information about this syntax, see the documentation for the UNIX editor ED.

4. Tell `fmbook` to carry out all the scheduled actions.

```
Command? go
```

Changing Print Settings

When you print a document using FrameMaker, you specify how you want it printed by choosing options in the Print dialog. When you save the document, the current settings of the Print dialog are saved with the file.

When you schedule a document for printing with `fmbook`, you can:

- Use the document's Print dialog settings.
- Use new settings that you specify in a book file. For example, you may want to change the printing scale for every document in the book.
- Override the document and book file print settings by typing commands within `fmbook`.

To view `fmbook`'s current print settings, type the **`status`** command. `fmbook` displays the current settings, as follows:

```
Scheduled actions for ...
...
(KeepPrintSettings No; EvenPages Yes; OddPages Yes; LowRes No;
 Collate Yes; Reverse Yes; Scale 100; Copies 1; To printer)

Command?
```

No means that fmbook will use the document or book file settings.

Normally, `fmbook` uses the document or book file print settings.

To use new print settings in your book file.

1. Add new commands to the end of the file, like the following:

```
<BookFile>
<Book
    <File
        <FileName "Ch1">
        <StartPageNum 1>
        <PageNumStyle numeric>
    >
    <File
        <FileName "Ch2">
        <StartPageNum NextOdd>
    >
    <Scale      70>
    <LowRes    Yes>
>
```

Overrides Scale and LowRes settings in files Ch1 and Ch2.

2. Type the **`book`** command and schedule files for printing.

To override the document and book file print settings by typing commands within `fmbook`:

1. At the `fmbook` command line, enter the commands for the settings you want, such as **`reverse no`**, **`copies 3`**, or **`lowres yes`**.
2. Type the command **`keepprintsettings yes`**. When KeepPrintSettings is set to Yes, `fmbook`'s print settings will override the settings found in any subsequent document or book file.
3. Type a book command and schedule your files for printing.

To reset `fmbook` to use the print settings in subsequent book or document files rather than its own settings:

1. If KeepPrintSettings is set to Yes, type the following command:

```
Command? keepprintsettings no
```

Writing MIF or Document Files

You can use `fmbook` to read FrameMaker documents and write them as MIF files and, conversely, read MIF files and write them as documents. Using this feature, for example, you can convert all the documents in a book to MIF files. (See Chapter 7, *Maker Interchange Format,* for information on MIF.)

Here are the commands you use to convert documents to MIF and vice versa:

- **`writemif`**
- **`writedoc`**
- **`nowrite`**

The **`writemif`** and **`writedoc`** commands schedule files for writing (if you are working with a book file, these commands schedule the writing of an entire book). To turn off scheduled writing, use the **`nowrite`** command (similar to the **`noprint`** command). The **`show`** command lists the documents that are scheduled for writing.

To convert documents to MIF files, or to convert MIF files to documents:

1. Start `fmbook` as usual.
2. Specify the book file. For example, if the book file is `Refman.book`, type this command:

```
Command? book Refman.book
(Book Refman.book) <-------
Command?
```

fmbook confirms the file and indicates that this is a book, not a single-file document.

3. Depending on the type of files you want to write, use the **`writedoc`** or **`writemif`** command.

This command:	Does this:
`writemif`	Schedules all files in the book for writing in MIF format, except for those files specified in any following **`nowrite`** commands.
`writedoc` ***`filename`***	Schedules the specified file for writing in FrameMaker document format.
`nowrite`	Cancels writing of all files in the book.
`nowrite` ***`filename`***	Cancels writing of the specified file.

When you convert a document file with the suffix `.doc` to a MIF file, `fmbook` creates a file with the same name, but with the suffix `.mif`. If the file doesn't have the suffix `.doc`, `fmbook` appends the suffix `.mif` to the file. For example:

This command:	Creates this MIF file:
`writemif Chapter1.doc`	`Chapter1.mif`
`writemif Proposal`	`Proposal.mif`
`writemif SalesFig.87`	`SalesFig.87.mif`

`fmbook` follows similar file naming rules when you write a document file, but appends the suffix `.doc` to the created file. For example:

This command:	Creates this document file:
`writedoc Chapter1.mif`	`Chapter1.doc`
`writedoc Proposal`	`Proposal.doc`
`writedoc SalesFig.87`	`SalesFig.87.doc`

`fmbook` does any deriving before writing the files you specify, so you always end up with the proper version of any derived files. `fmbook` writes files with the proper page numbers after the chapters in front have expanded. Using this feature, you can update documents in a book that use consecutive page numbering across chapters.

Creating a Standard Table of Contents and Index

Once you've prepared your documents, you can create a table of contents and index by starting `fmbook` and giving the following commands:

```
command? book Refman.book   <—— You can specify a single document file here.
(book Refman.book)
command? toc                <—— Schedules the creation of a table of contents.
Create RefmanTOC.doc
command? index              <—— Schedules the creation of an index.
Create RefmanIndex.doc
command? go
(Reading Ch1.doc, 10 page)
(Reading Ch2.doc, 61 pages)
(Reading Ch3.doc, 43 pages)
(Creating fmbookTOC.doc - Done.  Created 117 entries, 3 pages)
(Creating fmbookIndex.doc - Done.  Created 280 entries, 4 pages)
Command?
```

You can then open, edit, and print the table of contents and index with FrameMaker.

> **Note:** If your book file or any of its document files are open while `fmbook` is running, remember to save them before typing **go** in the `fmbook` window; otherwise `fmbook` uses the old versions of the files.

You can create a table of contents or an index as a part of your book. Embedding the table of contents or index allows you to print the entire book, including the table of contents or index, with `fmbook`; it also allows the length of the table of contents to affect the page numbering of the chapters that follow it. For more information, see *Creating an Embedded Table of Contents, Index, or Custom List* later in this chapter.

Editing the Table of Contents or Index

To edit the table of contents or index, open it with FrameMaker and edit it as you would any other document. It is easy to change the paragraph formats for the entire document. All paragraphs have one of just a few paragraph tags, so you can reformat the entire document by changing the format of each of the tags and applying those changes to similarly tagged paragraphs and the Paragraph Catalog. For more information, see the Paragraphs command in Chapter 3.

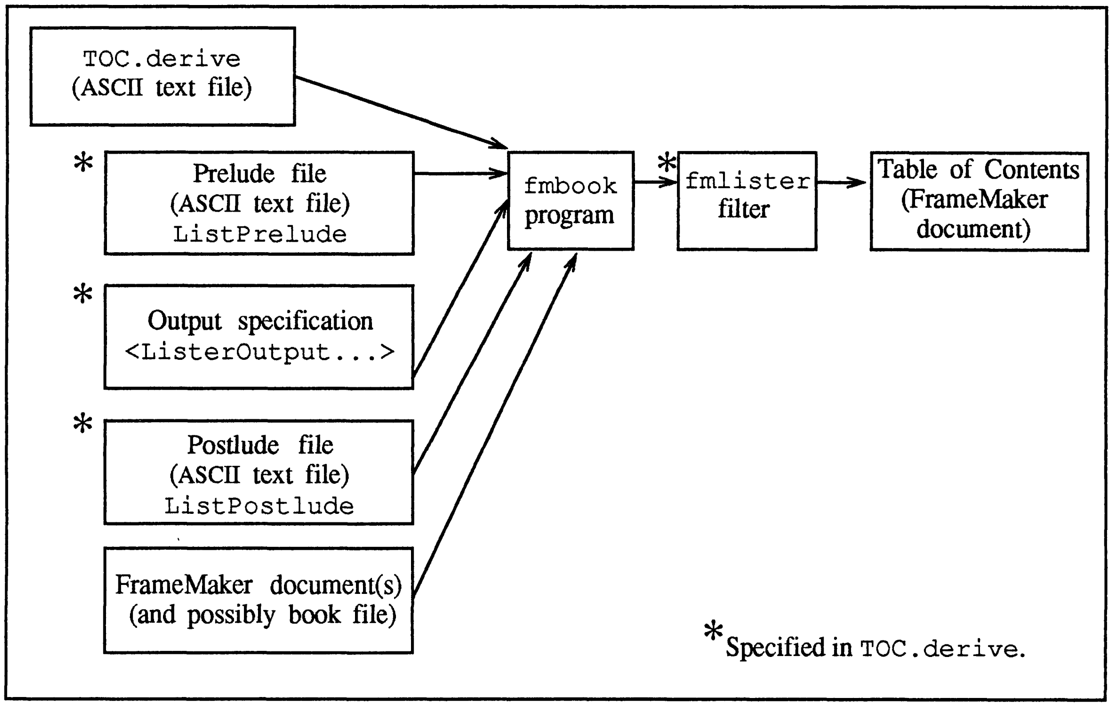
If you want `fmbook` to create a table of contents and index using your own format and layout, see *Customizing a Table of Contents* and *Customizing an Index* later in this chapter.

Customizing a Table of Contents

You can customize a table of contents in a number of ways. For example, you can extract different tags, change the default format, or add trailing text and graphics. To learn how to customize your table of contents, read the next two sections, *Understanding How fmbook Creates a Standard Table of Contents* and *Editing fmbook's Auxiliary Files*; then turn to the appropriate heading in this section.

Understanding How fmbook Creates a Standard Table of Contents

When `fmbook` creates a table of contents or other similar list for your document or book, it uses a number of auxiliary files (pictured below).



In particular, `fmbook` uses an ASCII text file named `TOC.derive` (normally found in the directory `.makerinit/bookdir` in the directory in which FrameMaker is installed) to determine the following:

- The filename or suffix for the table of contents.
- The name of the ASCII text file that specifies the format of the table of contents. The default name of this *prelude file* is `ListPrelude`.
- The names of the paragraph tags whose text is extracted for the table of contents and the specification for the output that `fmbook` generates for each tag. The default name for this *output specification* is <ListerOutput...>.
- The name of the ASCII text file that specifies any additional text or graphics you want to include at the end of the table of contents. The default name of this *postlude file* is `ListPostlude`.

- The name of the *filter* (backend program) to which `fmbook` passes its output for the table of contents. (`fmbook` does not create a table of contents. Instead, it extracts the text contained in paragraphs with particular tags and passes the information on to other programs.) The standard filter is called `fmlister`.

Editing fmbook's Auxiliary Files

To customize a table of contents or to create another similar list, you edit one or more of the auxiliary files (`TOC.derive`, `ListPrelude`, and `ListPostlude`) described in the previous section. `fmbook` looks for all of these files first in your current directory, then in your home directory, and finally in a `.makerinit/bookdir` directory. (FrameMaker looks for a `.makerinit/bookdir` directory first in your current directory, then in your home directory, and finally in the directory in which FrameMaker was installed.) You can override the version in `.makerinit/bookdir` by putting a different version in your home directory or in the directory from which you run `fmbook`. You can also change the version in `.makerinit/bookdir`, but you should do this with care, since changes to `.makerinit` in FrameMaker's install directory may affect other users of `fmbook`.

To edit `TOC.derive`, use any UNIX editor or FrameMaker. If you use FrameMaker to edit `TOC.derive`, however, remember to select the Text Only format in the Save dialog when saving the file, since `fmbook` expects `TOC.derive` to be in ASCII text format.

For instructions on creating and editing the prelude and postlude files, see *Changing the Format of a Table of Contents* and *Adding Trailing Text and Graphics to a Table of Contents* later in this chapter.

Extracting Different Tags

When creating a table of contents, `fmbook` normally extracts paragraphs whose tags are named chapter, section, subsection, and appendix. These tags create a three-level heading structure, with paragraphs tagged as chapter and appendix at the same level. For a number of reasons, however, this default may not meet your needs. For example:

- You may not want subsection entries in your table of contents.
- You may want to include a fourth level of headings.
- You may have two different tags that represent the same heading level (such as section and sectiontop).
- You may have used tags named chap, sect, subsect, and app.

To extract different tags for your table of contents, edit the file `TOC.derive`:

1. Display `TOC.derive`. Normally, `TOC.derive` looks like this:

```
<DerivedDocument
    <FileNameSuffix "TOC.doc">
    <DefaultDerive Yes>
    <A_List>
    <ExtractTag
        <Name "chapter">
        <ListerOutput "TOC chapter">
    >
    <ExtractTag
        <Name "Chapter">
        <ListerOutput "TOC chapter">
    >
    <ExtractTag
        <Name "section">
        <ListerOutput "TOC section">
    >
    ...
>
```

Specifies a tag for extraction.

Identifies the name of the tag to be extracted.

Specifies the tag name to be given to the extracted information.

Your copy may look somewhat different if the `TOC.derive` file has been customized for your site.

2. Edit the file as needed. For example, to use tags named chap and sect, but no others, edit the file to look like this:

```
<DerivedDocument
    <FileNameSuffix "TOC.doc">
    <DefaultDerive Yes>
    <A_List>
    <ExtractTag
        <Name "chap">
        <ListerOutput "TOC chapter">
    >
    <ExtractTag
        <Name "sect">
        <ListerOutput "TOC section">
    >
>
```

To include a fourth-level heading with the tag subsubsection, add the following lines before the last line of the file (containing just >):

```
    <ExtractTag
        <Name "subsubsection">
        <ListerOutput "TOC subsubsection">
    >
```

If two different tags in your document file represent the same heading level (such as section and sectiontop), your `TOC.derive` file should contain:

```
    <ExtractTag
        <Name "section">
        <ListerOutput "TOC section">
    >
    <ExtractTag
        <Name "sectiontop">
        <ListerOutput "TOC section">
    >
```

3. Save the file in ASCII text format.
4. Run `fmbook` to create the table of contents.

Changing the Format of a Table of Contents

`fmbook` uses the default format shown below for a table of contents.

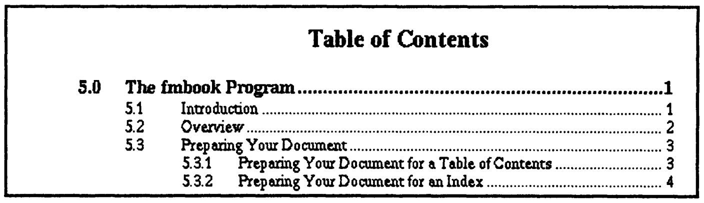


Table of Contents

5.0 The fmbook Program 1
5.1 Introduction 1
5.2 Overview 2
5.3 Preparing Your Document 3
5.3.1 Preparing Your Document for a Table of Contents 3
5.3.2 Preparing Your Document for an Index 4

Standard Format for a Table of Contents

For a number of reasons, however, this default may not meet your needs. For example, you might not use numbered paragraphs for headings. You might want different margins and tab settings, typefaces and sizes, or your own page headers, footers, and numbering scheme. You might even want a double-sided page format, rules, or graphics on every page. For example, you might want your table of contents to look like this:

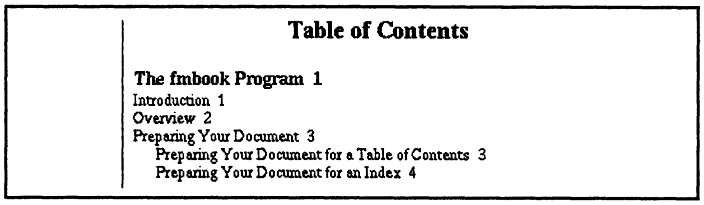


Table of Contents

The fmbook Program 1
Introduction 1
Overview 2
Preparing Your Document 3
Preparing Your Document for a Table of Contents 3
Preparing Your Document for an Index 4

Custom Format for a Table of Contents

This section describes how to make changes such as these. If your needs are beyond the scope of the information in this section, see *Creating a Custom Derivation* later in this chapter.

To change the default format for your table of contents, edit the prelude file `ListPrelude`. You can edit `ListPrelude` directly, but to do so requires some knowledge of Maker Interchange Format (for more information about MIF, see Chapter 7). Instead, we recommend that you do the following:

1. Create the table of contents with `fmbook`.
2. Open the table of contents with FrameMaker.
3. Reformat the table of contents as you wish.
4. Make sure that you apply any changes in paragraph formats to the Paragraph Catalog. For more information, see the Paragraphs command in Chapter 3.
5. Delete the table of contents entries (leaving just the title, if any, and any top-of-page graphics you may have added), so that your document becomes just a template in which `fmbook` can insert entries.

6. Save the file in the appropriate directory, in Maker Interchange Format, as `ListPrelude`. Be sure to save all the information you need. At the very least, you'll want to save the Paragraph Catalog. In addition:

If you set up:	**Then save:**
Page margins	Document Template
Page headers, footers, or numbering	Headers and Footers
A title for the table of contents	Paragraph Text and Paragraph Tags
Rules or other graphics or text on the master page	Master Page Items
An anchored frame at the top of the first page	Anchored Frames

7. Close or quit the table of contents document.
8. Edit the new `ListPrelude` file by deleting its first few lines, down to the first # symbol.

```
<MIFFile 1.01> # Generated by FrameMaker 1.1


include(mif_read.m4)
#
# /usr/test/sampleTOC.doc
# Options:
...
```

Delete these lines.

Note: To use FrameMaker to edit `ListPrelude`, open the file by holding down the Alt key as you press the Enter key or as you click OK in the Open dialog. Opening the file in this manner forces FrameMaker to read the file as text instead of interpreting the MIF it contains. Also remember to select the Text Only format in the Save dialog when saving the edited file.

9. Create the table of contents again.
10. Check the resulting table of contents. If it isn't quite right, repeat steps 1 through 9 above to refine your format.

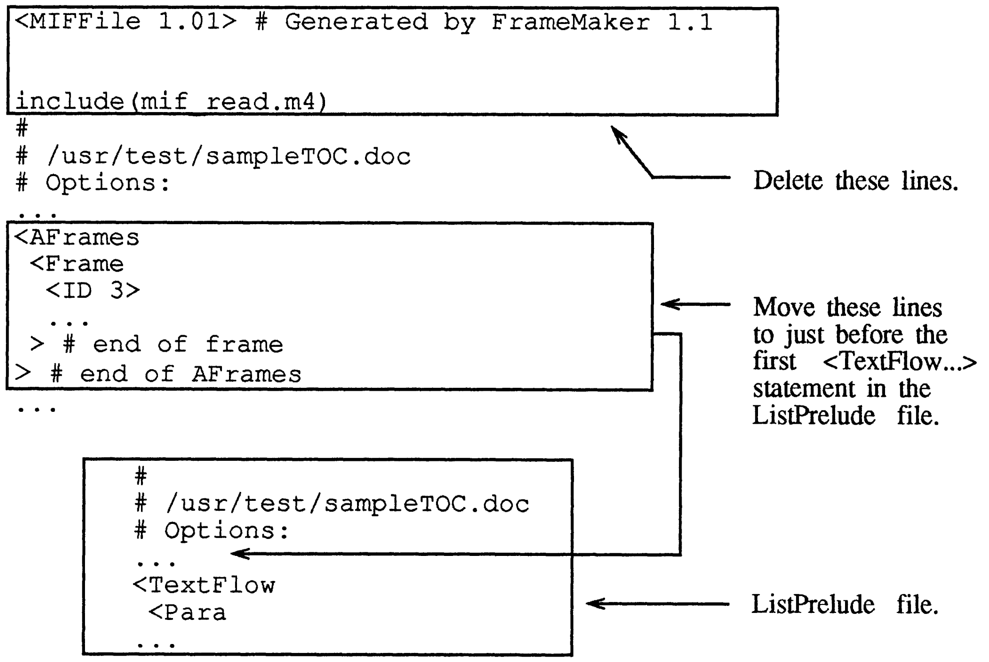
Adding Trailing Text and Graphics to a Table of Contents

To add text (for example, a paragraph of explanatory text) or graphics (for example, a rule or other graphic element) to the end of your table of contents, edit the postlude file, `ListPostlude`. You can edit `ListPostlude` directly, but to do so requires some knowledge of Maker Interchange Format (for more information about MIF, see Chapter 7). Instead, we recommend that you do the following:

1. Create the table of contents with `fmbook`.
2. Open the table of contents with FrameMaker.
3. Add the text and graphics you want at the end of the table of contents.
4. Delete all of the table of contents, except for the text and graphics you've just added.
5. Save the file in the appropriate directory, in Maker Interchange Format, as `ListPostlude`. Be sure to save all the information you need. If you added text,

save Paragraph Tags, Paragraph Text, and Paragraph Formats; if you added anchored graphics, also save Anchored Frames.

6. Edit the new `ListPostlude` file by deleting its first few lines, down to the first # symbol. If you added graphics at the end of the table of contents, move the <AFrames...> statement in `ListPostlude` to the appropriate place in your `ListPrelude` file. For more information, see Chapter 7. .



```
<MIFFile 1.01> # Generated by FrameMaker 1.1


include(mif_read.m4)
#
# /usr/test/sampleTOC.doc
# Options:
...
<AFrames
 <Frame
  <ID 3>
  ...
 > # end of frame
> # end of AFrames
...
```

```
#
# /usr/test/sampleTOC.doc
# Options:
...
<TextFlow
 <Para
...
```

Delete these lines.
Move these lines to just before the first <TextFlow...> statement in the ListPrelude file.
ListPrelude file.

Note: To use FrameMaker to edit `ListPostlude`, open the file by holding down the Alt key as you press the Enter key or as you click OK in the Open dialog. Opening the file in this manner forces FrameMaker to read the file as text instead of interpreting the MIF it contains. Also remember to select the Text Only format in the Save dialog when saving the edited file.

7. Create the table of contents again.
8. Check the resulting table of contents. If it isn't quite right, repeat steps 1 through 7 above to refine your format.

Creating a Custom List

Many documents require derived lists other than the standard table of contents. For example, you may need a list of figures, a list of tables, or other similar list. This section describes how to create such lists. If your needs are beyond the scope of the information in this section, see *Creating a Custom Derivation* later in this chapter.

Preparing Documents for a Custom List

Make sure that all paragraphs have the appropriate tag names. For example, each figure title might have a tag of figure, each table title might have a tag of table, and so forth.

Defining a Custom List

To define a different kind of list, create a new file similar to `TOC.derive`:

1. Copy `TOC.derive` into the appropriate directory with a new prefix (the prefix will be the `fmbook` command you use to create the list, such as `LOF` for a list of figures); the suffix must be `.derive`. For example, if you create `LOF.derive` to derive a list of figures, you would type the `fmbook` command **`lof`** to create the list.
2. Specify the name of your custom list by changing the second line of your new `.derive` file. It must be of the form <FileName "*xxx*"> or <FileNameSuffix "*xxx*">. In the first case, `fmbook` assigns the name you specify to the list it creates; in the second case, `fmbook` assigns a name using the suffix that you specify (for more information about how this filename is constructed, see *Preparing Documents for a Table of Contents* earlier in this chapter).
3. Specify the paragraphs you want to extract by changing the <ExtractTag...> statements. For more information, see *Extracting Different Tags* earlier in this chapter.

```
<DerivedDocument
     <FileNameSuffix "LOF.doc">
     <DefaultDerive Yes>
     <A_List>
     <ExtractTag
          <Name "figure">
          <ListerOutput "LOF figure">
     >
>
```

Specifies the name of your custom list.

Identifies the name of the tag to be extracted.

Specifies the tag name to be assigned to the extracted information.

4. Save the file in ASCII text format.
5. Create your list by typing the appropriate `fmbook` command (see step 1 above).

```
Command? lof
Create myfileLOF.doc
Command? go
```

Changing the Format of a Custom List

`fmbook` normally uses the `ListPrelude` and `ListPostlude` files to determine the format of the list. If your table of contents and other custom lists have the same page layout and titling information, you can simply modify `ListPrelude` and `ListPostlude` to include formatting information for the new paragraph tags in your custom list. To learn how to modify these files, see *Changing the Format of a*

Table of Contents and *Adding Trailing Text and Graphics to a Table of Contents* earlier in this chapter.

It is much more likely, however, that you want a custom title for your list, and possibly a custom format and layout as well. To achieve this, you create new prelude and postlude files and change your `.derive` file.

1. First, create the new prelude and postlude files:

   Create your list with `fmbook`, using the default format for a table of contents. Then follow the instructions in *Changing the Format of a Table of Contents* and *Adding Trailing Text and Graphics to a Table of Contents*, earlier in this chapter, but save the files with different names (*not* as `ListPrelude` and `ListPostlude`). For example, for a list of figures you might name the files `LOFPrelude` and `LOFPostlude`.

2. Next, change your `.derive` file to tell `fmbook` that the format specifications are in the files you just saved (and *not* in `ListPrelude` and `ListPostlude`). Normally, your `.derive` file begins like this:

   ```
   <DerivedDocument
       <FileNameSuffix "TOC.doc">
       <DefaultDerive Yes>
       <A_List>
       <ExtractTag>
       ...
   ```

   The <A_List> statement above is an abbreviation built into `fmbook`. It is equivalent to:

   ```
   <PreludeFileName "ListPrelude">
   <PostludeFileName "ListPostlude">
   <Filter "fmlister">
   <MifToMaker Yes>
   ```

   Add either or both of the following two lines (as needed) just below the <A_List> statement:

   ```
   <PreludeFileName "xxx">
   <PostludeFileName "yyy">
   ```

   where *xxx* and *yyy* are the names of *your* prelude and postlude files. These statements override the statements in <A_List>.

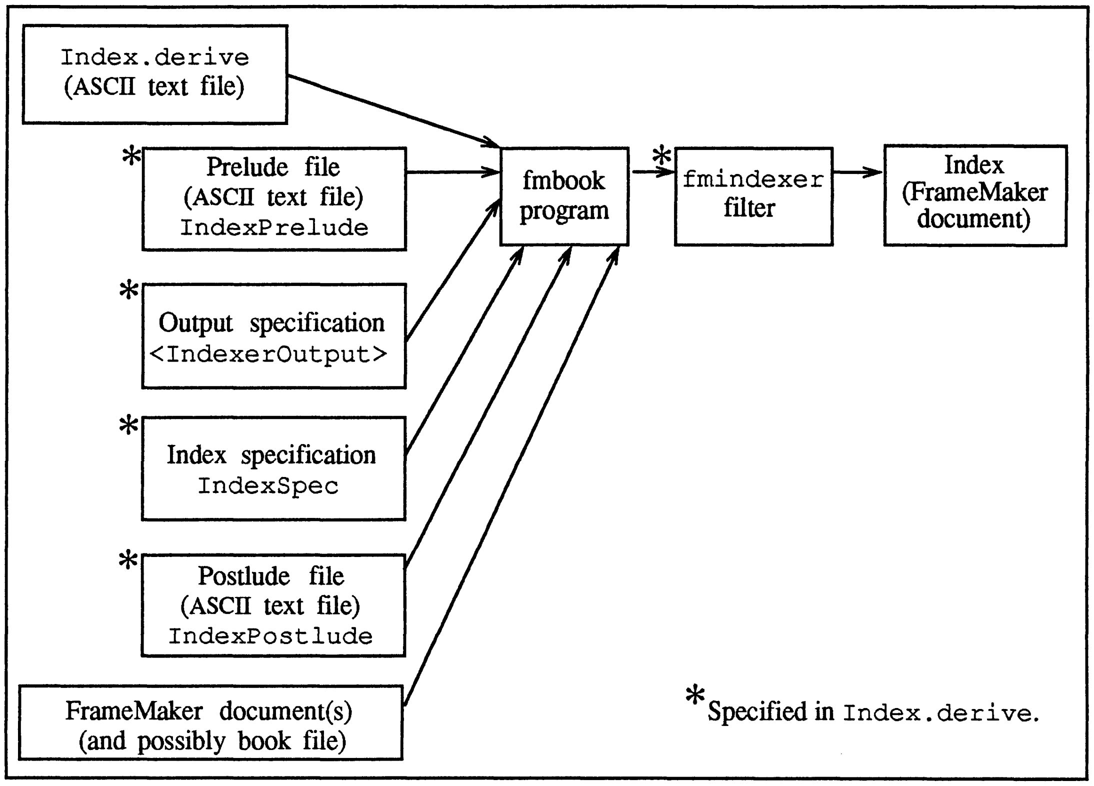
3. Save the file in ASCII text format.
4. Finally, run `fmbook` again to create your custom list.

Customizing an Index

You can customize an index in a number of ways. For example, you can extract different markers, change the default format, and add trailing text and graphics. To learn how to customize your index, read the next section, *Understanding How fmbook Creates a Standard Index*; then turn to the appropriate heading in this section.

Understanding How fmbook Creates a Standard Index

When `fmbook` creates an index for your document, it uses a number of auxiliary files (pictured below).



In particular, `fmbook` uses an ASCII text file named `Index.derive` (normally found in the directory `.makerinit/bookdir` in the directory in which FrameMaker was installed) to determine the following:

- The filename or suffix for the index.
- The name of the ASCII text file that specifies the format of the index. The default name of this *prelude file* is `IndexPrelude`.
- The specification for the output that `fmbook` generates for each marker. The default name for this *output specification* is:<IndexerOutput>.
- The name of the ASCII text file that specifies the page reference separators, the sort order, and the index groups and labels. The default name of this file is `IndexSpec`.
- The name of the ASCII text file that specifies any additional text or graphics you want to include at the end of the index. The default name of this *postlude file* is `IndexPostlude`.

The name of the *filter* (backend program) to which `fmbook` passes its output for the index. (`fmbook` does not create the index. Instead, it extracts the contents of markers with a particular marker type and passes the information on to other programs.) The standard filter is called `fmindexer`.

To customize an index, you edit one or more of the auxiliary files (`Index.derive`, `IndexSpec`, `IndexPrelude`, and `IndexPostlude`) described above. The same rules apply when editing `Index.derive` and `IndexSpec` as when editing `TOC.derive` for a table of contents. For more information about editing these files, see *Editing fmbook's Auxiliary Files* earlier in this chapter. You'll find instructions for creating and editing the prelude and postlude files later in this chapter.

Extracting Different Markers

When creating an index, `fmbook` normally extracts "index markers." However, you may want to extract different markers. For example, you may want to include "author markers" as well as "index markers" in your index. This section explains how to extract a different marker type or additional marker types for your index. If you want to create several different indexes, each containing a different marker type (for example, separate title, author, and subject indexes), see *Creating Alternative Indexes* later in this chapter. If you want to create a new marker type, see Appendix D.

To extract different markers for your index, edit the file `Index.derive`:

1. Display `Index.derive` (for more information, see *Editing fmbook's Auxiliary Files* earlier in this chapter). Normally, `Index.derive` looks like this:

```
<DerivedDocument
     <FileNameSuffix "Index.doc">
     <DefaultDerive AsNeeded>
     <An_Index>
     <ExtractMark                  ←  Specifies a marker for extraction.
          <Name "Index">           ←  Identifies the name of the marker to be
          <IndexerOutput>             extracted.
     >
>
```

   Your copy may look somewhat different if the `Index.derive` file has been customized for your site.

2. Edit the file as needed. For example, to use only markers named Author, edit the file to look like this:

```
<DerivedDocument
     <FileNameSuffix "Index.doc">
     <DefaultDerive AsNeeded>
     <An_Index>
     <ExtractMark
          <Name "Author">          ←  Identifies the name of the marker to
          <IndexerOutput>             be extracted.
     >
>
```

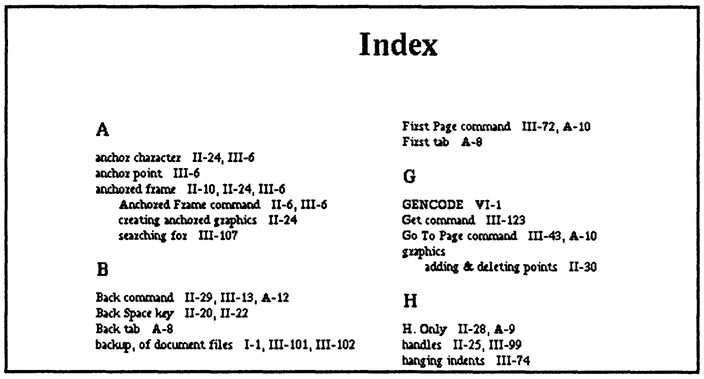
To use markers Index *and* Author, edit the file to look like this:

```
<DerivedDocument
    <FileNameSuffix "Index.doc">
    <DefaultDerive AsNeeded>
    <An_Index>
    <ExtractMark
        <Name "Index">                fmbook will extract both markers.
        <IndexerOutput>
    >
    <ExtractMark
        <Name "Author">
        <IndexerOutput>
    >
>
```

3. Save the file in ASCII text format.
4. Create the index.

Changing the Format of an Index

`fmbook` uses the default format shown below for an index.



Index

A

anchor character II-24, III-6
anchor point III-6
anchored frame II-10, II-24, III-6
  Anchored Frame command II-6, III-6
  creating anchored graphics II-24
  searching for III-107

B

Back command II-29, III-13, A-12
Back Space key II-20, II-22
Back tab A-8
backup, of document files I-1, III-101, III-102

First Page command III-72, A-10
First tab A-8

G

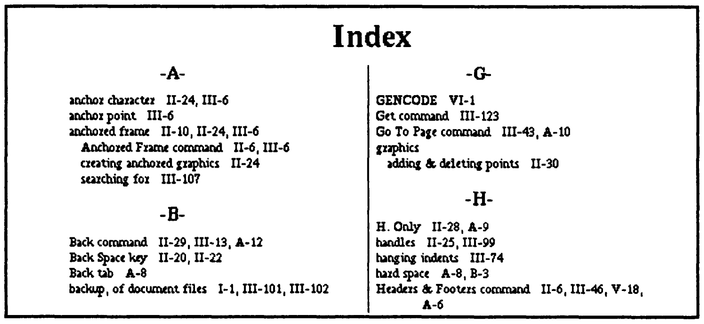
GENCODE VI-1
Get command III-123
Go To Page command III-43, A-10
graphics
  adding & deleting points II-30

H

H. Only II-28, A-9
handles II-25, III-99
hanging indents III-74

Standard Format for an Index

For a number of reasons, however, this default may not meet your needs. You may want a different number of columns on a page, different margins and tab settings, typefaces and sizes, or your own page headers, footers, and numbering scheme. You might even want a double-sided page format or graphics on every page. For example, you might want your index to look like this:



Index

-A-

anchor character II-24, III-6
anchor point III-6
anchored frame II-10, II-24, III-6
  Anchored Frame command II-6, III-6
  creating anchored graphics II-24
  searching for III-107

-B-

Back command II-29, III-13, A-12
Back Space key II-20, II-22
Back tab A-8
backup, of document files I-1, III-101, III-102

-G-

GENCODE VI-1
Get command III-123
Go To Page command III-43, A-10
graphics
  adding & deleting points II-30

-H-

H. Only II-28, A-9
handles II-25, III-99
hanging indents III-74
hard space A-8, B-3
Headers & Footers command II-6, III-46, V-18, A-6

Custom Format for an Index

This section describes how to make changes such as these. If your needs are beyond the scope of the information in this section, see *Creating a Custom Derivation* later in this chapter.

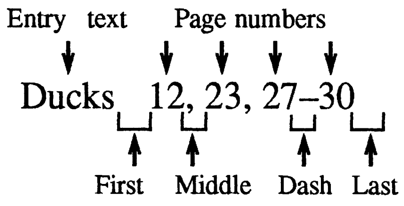
To change the default format for your index, you edit the prelude file `IndexPrelude`. This file plays exactly the same role for an index as `ListPrelude` does for a table of contents or other list. You edit the two files in exactly the same way. For specific instructions, see *Changing the Format of a Table of Contents* earlier in this chapter. Remember, however, that you are editing `IndexPrelude`, *not* `ListPrelude`.

Adding Trailing Text and Graphics to an Index

To add text (for example, a paragraph of explanatory text) or graphics (for example, a rule or other graphic element) to the end of your index, edit the `IndexPostlude` file. This file plays exactly the same role for an index as `ListPostlude` does for a table of contents or other list. You edit the two files in exactly the same way. For specific instructions, see *Adding Trailing Text and Graphics to a Table of Contents* earlier in this chapter. Remember, however, that you are editing `IndexPostlude`, *not* `ListPostlude`.

Changing Page Reference Separators

The page reference separators for the standard index normally look like this:



- To separate the page list from the entry text, there are three spaces (First).
- Between each two page numbers there is a comma and a space (Middle).
- Any two page numbers that constitute a start/end pair have their page numbers separated by an en dash (Dash)
- Nothing follows the last page number (Last).

To change these separators, you edit the file `IndexSpec`:

1. Display `IndexSpec`. Normally, the first few lines of `IndexSpec` look like this:

   ```
   <First  "   ">     Before first page number
   <Middle ", ">      Between page numbers
   <Last   "">        After last page number
   <Dash   "\xD0 ">   Between Start/End pairs
   ```

   Your copy may look somewhat different if the `IndexSpec` file has been customized for your site.

2. Edit the file as needed. See Appendix B for any special character codes you may need.

3. Save the file in ASCII text format.
4. Create the index.

Changing Entry Sort Order

To change the sort order of index entries:

1. Display `IndexSpec`. Normally, the sort order portion of `IndexSpec` begins this way:

```
<SortAs "a" A
     \x87 \x88 \x89 \x8A \x8B \x8C
\xE7 \xCB \xE5 \x80 \xCC \x81 \xBB >
     # aacute agrave acircumflex adieresis atilde aring
     # Aacute Agrave Acircumflex Adieresis Atilde Aring
ordfeminine
<SortAs "ae" \xAE \xBE > # AE ae
<SortAs "b" B>
<SortAs...
```

These character codes are sorted as if they were "a".

These are comment lines.

Your copy may look somewhat different if the `IndexSpec` file has been customized for your site.

2. Edit the file as needed. See Appendix B for any special character codes you may need.
3. Save the file in ASCII text format.
4. Create the index.

Changing Entry Groupings and Group Titles

In the standard index, entries are grouped one letter at a time (all the A's, then the B's, and so forth), and each group has an uppercase letter as a title. You may want to change this, for example, to use different titles or to group entries two letters at a time.

To change the index entry groupings or group titles:

1. Display `IndexSpec`. The end of the `IndexSpec` file has entries such as:

```
<GroupTitle "A">
```

This means that there is a title (A) above the set of A entries (if any) in the final index. Note that the <GroupTitle...> string is treated like any other entry in the index; it is subject to the results of the <SortAs...> statements to determine its sorting position.

2. Edit the file as needed.

For example, if you want the titles to appear as -A-, edit the file to look like this:

```
<GroupTitle "-Symbols-[\x07 ]">
<GroupTitle "-Numerics-[0]">
<GroupTitle "-A-[A]">
<GroupTitle "-B-[B]">
<GroupTitle "-C-[C]">
...
```

Specifies how the title is sorted.

Specifies how the title appears in the index.

To group the titles two letters at a time, edit the file to look like this:

```
<GroupTitle "-Symbols-[\x07 ]">
<GroupTitle "-Numerics-[0]">
<GroupTitle "A-B[A]">
<GroupTitle "C-D[C]">
...
```

Specifies that all entries from A up to but not including C are grouped together under the title A-B.

3. Save the file in ASCII text format.
4. Create the index.

Creating Alternative Indexes

This section describes how to create several different indexes for a document (for example, separate title, author, and subject indexes).

To prepare your documents, insert markers with the appropriate marker types. For more information, see *Preparing Documents for an Index*, earlier in this chapter.

Defining an Alternative Index

To define a different kind of index, you create a new auxiliary file similar to `Index.derive`, as follows:

1. Copy `Index.derive` into the appropriate directory with a new name. The new file's prefix will be the `fmbook` command you use to create the custom index (such as `Author` for an author's index); it's suffix must be `.derive`. For example, if you create `Author.derive` to derive an author's index, you would type the `fmbook` command **`author`** to create the index.
2. Change the second line of your new `.derive` file. It must be of the form <FileName "*xxx*"> or <FileNameSuffix "*xxx*">. In the first case, `fmbook` assigns the name you specify to the list it creates; in the second case, `fmbook` assigns a name using the suffix that you specify (for more information about how this filename is constructed, see *Preparing Documents for an Index* earlier in this chapter).

3. Change the <ExtractMark...> statements to reflect the markers you want `fmbook` to extract. For more information, see *Extracting Different Markers* earlier in this chapter.

```
<DerivedDocument
     <FileNameSuffix "Author.doc">  <—— Specifies the name of your
     <DefaultDerive AsNeeded>             custom  list.
     <An_Index>
     <ExtractMark
          <Name "Author">   <———————— Specifies the marker to be extracted.
          <IndexerOutput>
     >
>
```

4. Save the file in ASCII text format.
5. Create your index by typing the appropriate `fmbook` command (see step 1 above).

```
Command? **author**
Create myfileAuthor.doc
Command? **go**
```

Changing the Format of an Alternative Index

`fmbook` normally uses `IndexPrelude` and `IndexPostlude` to determine the format of the index and `IndexSpec` to determine page reference separators, sort order, and index groupings and titles. If all of your indexes have the same page layout and titles, you don't need to make any changes to these files.

To create a custom format for your new index (for example, to change the page header or footer or index title), you create new prelude and postlude files (and possibly a new index specification file) and change your `.derive` file.

1. First, create the new prelude and postlude files:

   Create your index with `fmbook`, using the existing prelude, postlude, and index specification files. Then follow the instructions in *Customizing an Index*, earlier in this chapter, for editing the `IndexPrelude`, `IndexPostlude`, and `IndexSpec` files, but save the files with different names (*not* as `IndexPrelude`, `IndexPostlude`, and `IndexSpec`).
2. Next, change your `.derive` file to tell `fmbook` that the format specifications are in the files you just saved. Normally, your `.derive` file begins this way:

```
<DerivedDocument
     <FileNameSuffix "Index.doc">
     <DefaultDerive AsNeeded>
     <An_Index>
     <ExtractMark>
     ...
>
```

   The <An_Index> statement above is an abbreviation built into `fmbook`. It is equivalent to:

```
<PreludeFileName "IndexPrelude">
<PostludeFileName "IndexPostlude">
<Filter "fmindexer">
<FilterParm0 "IndexSpec">
<MifToMaker Yes>
```

Add one or more of the following three lines (as needed) just below the <An_Index> statement:

```
<PreludeFileName "xxx">
<PostludeFileName "yyy">
<FilterParm0 "zzz">
```

where *xxx*, *yyy*, and *zzz* are the names of *your* prelude, postlude, and index specification files. These statements override the statements in <An_Index>.

3. Save the file in ASCII text format.
4. Create the index.

Creating an Embedded Table of Contents, Index, or Custom List

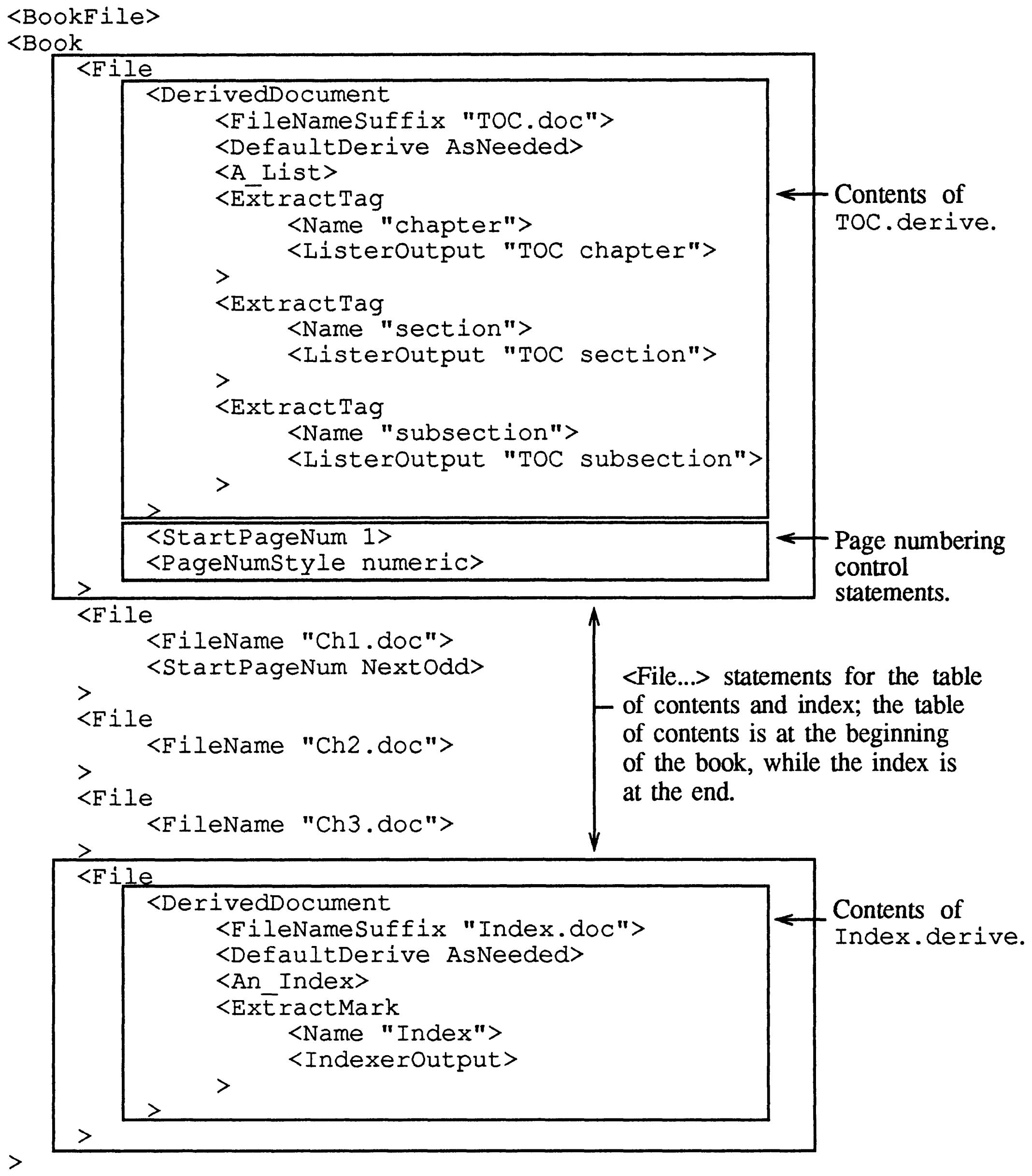
You can create a table of contents or an index as a part of your book. Embedding the table of contents or index allows you to print the entire book, including the table of contents or index, with `fmbook`; it also allows the length of the table of contents to affect the page numbering of the chapters that follow it.

Preparing the Book File

To embed an index, table of contents, or other similar list in a book file, you insert the contents of your `Index.derive`, `TOC.derive`, or other custom `.derive` files into your book file, as follows:

1. Add a <File...> statement to your book file for each derived document you want to include (index, table of contents, or other list).
2. In place of the <FileName...> statement, substitute the contents of the appropriate `.derive` file.
3. Add page numbering control statements as needed.
4. Check your book file carefully to make sure that every start angle bracket (<) has a corresponding end angle bracket (>).

For example, if your book contains a table of contents, three chapters, and an index, your book file might look like this:


Contents of TOC.derive.
Page numbering control statements.
<File...> statements for the table of contents and index; the table of contents is at the beginning of the book, while the index is at the end.
Contents of Index.derive.

```
<BookFile>
<Book
    <File
        <DerivedDocument
            <FileNameSuffix "TOC.doc">
            <DefaultDerive AsNeeded>
            <A_List>
            <ExtractTag
                <Name "chapter">
                <ListerOutput "TOC chapter">
            >
            <ExtractTag
                <Name "section">
                <ListerOutput "TOC section">
            >
            <ExtractTag
                <Name "subsection">
                <ListerOutput "TOC subsection">
            >
        >
        <StartPageNum 1>
        <PageNumStyle numeric>
    >
    <File
        <FileName "Ch1.doc">
        <StartPageNum NextOdd>
    >
    <File
        <FileName "Ch2.doc">
    >
    <File
        <FileName "Ch3.doc">
    >
    <File
        <DerivedDocument
            <FileNameSuffix "Index.doc">
            <DefaultDerive AsNeeded>
            <An_Index>
            <ExtractMark
                <Name "Index">
                <IndexerOutput>
            >
        >
    >
>
```

A <DerivedDocument...> statement can appear in a book file either after the main <Book...> statement or within a <File...> statement (as shown above). In the first case, `fmbook` considers the derived document to be strictly an output file that contains information about the book. In the second case, however, `fmbook` considers the derived document to be a part of the book itself. In this case, its page count may affect the page numbering of the rest of the book. For instance, adding another chapter to your book may cause the derived table of contents to spill onto another page. This, in turn, may immediately make the information in the table of contents (and any other derived files) obsolete, since the page numbering may have changed in all of the chapters. On the infrequent occasions when this occurs, `fmbook` rederives all the derived documents with the new page numbering information.

Scheduling the Table of Contents and Index

When you type the command **`book Refman.book`**, `fmbook` automatically schedules the creation of the derived documents specified in the book file. After you type the **`go`** command, `fmbook` creates those derived documents and then unschedules them. `fmbook` won't create them again on subsequent **`go`** commands unless you changed the book file or one of the documents in the book after `fmbook` last created the derived documents. If you did change the book file or one of the documents in the book, `fmbook` notices this the next time you issue a **`show`** or **`go`** command and schedules the creation of the derived documents again. This is called *as-needed deriving*. If there is *no* existing copy of the derived file, `fmbook` schedules the creation of a new one.

You can also change the default for any derived document by putting a <DefaultDerive Yes> or <DefaultDerive No> statement in your book file. When you put <DefaultDerive Yes> in a <DerivedDocument...> statement, `fmbook` schedules the derivation of the document whether or not it is up to date.

Finally, you can use the **`create`** and **`nocreate`** commands to schedule or deschedule the creation of a derived document, regardless of the statements in your book file, as follows:

Command	**Effect**
`nocreate`	Deschedules the creation of all derived documents in your book file.
`nocreate` ***`filename`***	Deschedules the creation of the specified file.
`create`	Schedules the creation of all derived documents in your book file.
`create` ***`filename`***	Schedules the creation of the specified file.

If you want to work only on your index, type **`nocreate`** to deschedule the creation of all derived documents. Then type **`create RefmanIndex.doc`** to schedule the creation of just the index. Repeated **`go`** commands then derive a new index each time.

fmbook Limitations

There is no limit on the number of different documents that can make up a single book, though the largest single document must be able to fit into memory. `fmbook` has more memory than FrameMaker does, so this should never be a problem. There is no limit on the number of <ExtractTag...> and <ExtractMark...> statements, though UNIX imposes a limit of about 20 simultaneous files that can be derived at once. There is also no restriction on the size of a derived list or index other than available disk space. Note that `fmbook` does not use any temporary files other than intermediate copies of the derived files and any files created by the programs called by <Filter...> statements (see the next section, *Creating a Custom Derivation* for more information about filters). The standard filters, `fmlister` and `fmindexer`, do not make temporary files and have no restrictions on their input size.

Creating a Custom Derivation

Earlier sections of this chapter explained how to customize an index, table of contents, or other similar list. If the changes you want to make are beyond the scope of those earlier sections, you need to understand a little more about how `fmbook` derives a file. Before continuing, you should read *Understanding How fmbook Creates a Standard Table of Contents* and *Understanding How fmbook Creates a Standard Index* earlier in this chapter.

Understanding <ExtractTag...> and <ExtractMark...> Statements

<ExtractTag...> and <ExtractMark...> statements, used in <DerivedDocument...> statements (either in a `.derive` file or in a book file), control `fmbook`'s extraction process. The <ExtractTag...> statement normally looks like this:

```
<ExtractTag                                   Identifies the name of the tag to
     <Name "chapter">  <-------------         be extracted.
     <ListerOutput "TOC chapter"> <--         Provides the tag name and output
>                                             specification for the extracted
```

The <ExtractMark...> statement normally looks like this:

```
<ExtractMark                                  Identifies the name of the marker to
     <Name "Index"> <-----------------        be extracted.
     <IndexerOutput> <----------------        Provides the output specification for
>                                             the extracted information.
```

Understanding <ListerOutput...> and <IndexerOutput> Statements

Both <ListerOutput...> and <IndexerOutput> are shorthand for specific cases of the general <Output...> statement. <ListerOutput...> is an abbreviation for the following:

```
<CleanUpWhiteSpace Yes>
<Output                                       Identifies the name of the paragraph
     <String "tagname">  <-----------         tag for this paragraph of output.
     <Tab>
     <Text>  <------------------------        Specifies the paragraph text.
     <Tab>
     <PageNumPrefix> <PageNum> <PageNumSuffix>
     <NewLine>             <-----------       Indicates page number information.
>
```

The output specified by <ListerOutput...> matches the input that the backend processing program `fmlister` expects. For each input line, `fmlister` assumes that information will be separated by <Tab> and terminated by <NewLine>.

Likewise, <IndexerOutput> is an abbreviation for the following:

```
<CleanUpWhiteSpace Yes>
<Output
    <Text> <----------------------- Indicates marker text.
    <Tab>
    <Order> <---------------------- Indicates relative page number from the
    <Tab>                           beginning of the book; useful for sorting
    <PageRefStyle>                  index entries by the order of their actual
    <Tab>                           appearance in the book.
    <PageRefFont> <---------------- Specifies page numbering style.
    <Tab>
    <PageNumPrefix> <PageNum> <PageNumSuffix>
    <NewLine>
>                        <--------- Indicates page number information.
```

This output matches the input format that the backend processing program `fmindexer` expects. For each input line, `fmindexer` assumes that marker information will be separated by <Tab> and terminated by <NewLine>.

<PageRefStyle> can have values of 0, 1, 2, or 3 for None, Single Page, Start of Page Range, or End of Page Range, respectively. <PageRefFont> has the value 0 for regular text, plus 1 for Bold, plus 2 for Italic, plus 4 for Underline, plus 8 for Strikeout. For more information, see *Controlling Page Numbering for a Book* earlier in this chapter.

Note that tabs that appear in <Text> are represented as \t, so they won't be confused with the output from the <Tab> statement (a real tab character). Likewise, carriage returns appear as \n. Also, nonprintable characters appear in \x format, along with a few other characters, such as the right angle bracket (>), with special backslash (\) representations.

Changing the <Output...> Statement

The output format for each tag and mark must match the input format expected by the filters `fmlister` and `fmindexer`, respectively.

While `fmlister` and `fmindexer` expect specific input formats, you can make some changes to the <Output...> statements that pass data to them, as long as you separate the appropriate pieces of information with <Tab> and end the output with <NewLine>. For example, the standard table of contents contains the paragraph text at the left of each line with the page number on the right. But you may want to create a table of contents with the page number on the left, as shown below:

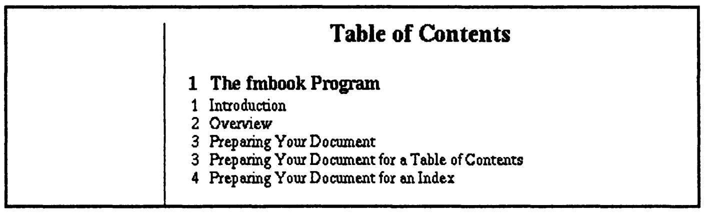


Table of Contents

1 The fmbook Program
1 Introduction
2 Overview
3 Preparing Your Document
3 Preparing Your Document for a Table of Contents
4 Preparing Your Document for an Index

Custom Format for a Table of Contents

To create this table of contents, you need to interchange the page number and the text for each entry. If you want the page numbers right-aligned, you also need to precede each page number with a tab character. To do this, use the following <Output...> statement in each <ExtractTag...> statement (in the <DerivedDocument...> statement that specifies the table of contents):

```
<Output
     <String "TOC chapter">
     <Tab>
     <String "\t">                                          Indicates a tab character.
     <PageNumPrefix> <PageNum> <PageNumSuffix>
     <Tab>
     <Text>                                                 Page number and
     <NewLine>                                              paragraph information
>                                                           have been interchanged.
```

First, we interchanged the statements referring to the paragraph text and the page numbering. Since `fmlister` expects this information only as text strings separated by a <Tab>, we didn't confuse it by passing it the page number before the text string. Second, we inserted the <String "\t"> statement before the page numbering information so that each line of the table of contents would begin with a tab character, allowing the page numbers to be aligned with a flush right tab stop. `fmlister` then interpreted the \t as part of the page numbering string. Of course, you also need to modify `ListPrelude` if you want your table of contents to be formatted correctly. For more information, see *Customizing a Table of Contents* earlier in this chapter.

Finally, you can specify up to 10 text strings to be associated with each file in a book, and insert those strings in your <Output...> statement. For more information, see *Book File Syntax Description* later in this chapter.

Creating Your Own Filter for fmbook

Tables of contents and indexes are created using the backend programs `fmlister` and `fmindexer`, respectively. You can, however, specify that `fmbook` pass its output directly to a file without any postprocessing; replace <A_List> or <An_Index> in your `.derive` file with its expanded form and delete the <Filter...> statements. For more information, see *Changing the Format of a Custom List* and *Changing the Format of an Alternative Index* earlier in this chapter.

You can also create your own custom backend program to derive other documents or to use the extracted data for other purposes (for example, in a database application). `fmbook` writes out data in the following order:

- The contents of the prelude file (if any was specified) followed by a null character (\0).
- The contents of the postlude file (if any was specified) followed by another null character.
- The extracted data, followed by EOF (end of file).

`fmbook` drops null characters in the prelude and postlude files.

The program named in a <Filter...> statement will be started up via a UNIX `execlp` call (that is, it will be searched for down your entire path, as if you had given the command to the shell). `fmbook` passes a number of arguments to the filter, as follows:

This argument:	**Contains:**
argv[0]	The filter program name.
argv[1]	The name of the file that the filter should input; this is a temporary filename (with a complete path).
argv[2]	The name of the file that the filter should output; this is a temporary filename (with a complete path), so that fmbook won't overwrite an old derived file until the new one is complete. After the filter produces the temporary file, fmbook renames the final output file to the correct name. Also, if <MifToMaker> is turned on, fmbook converts it from MIF format to FrameMaker format.
argv[3]	The path to the `.makerinit` directory (to help the filter find default specification files such as `IndexSpec`).
argv[4]	The string from <FilterParm0...>.
argv[5-13]	The string from <FilterParm[1-9]...>; use the argument positions in ascending order, since the first null FilterParm ends the list of arguments.

Using the fmbook Command Line Interface

You can use `fmbook` in a noninteractive mode by typing commands on the command line or in shell scripts (for an example, see *install_dir*`/bin/fmprint`). The functionality of interactive mode is retained, however, because all interactive commands have a corresponding command line version. The command:

```
fmbook myfile.doc -i -p
```

for example, schedules an index and printing for `myfile.doc`. There's an implicit **-g** command (which is similar to the interactive **go** command) at the end of the line, so the command above also creates the index and prints the document. With the **-g** command, you can schedule several actions and carry them out, by typing all the commands on one command line. For instance, the command:

```
fmbook myfile.doc -p -c2 -g MobyDuck.book -p
```

prints two copies of `myfile.doc` and one copy of `MobyDuck.book`.

The full list of commands follows. Note that parameters must directly follow the command letter, with no intervening spaces.

Command	**Meaning**
-c[*number*]	Copies
-C[*number*]	Scale (see **-z**)
-d[*filename*]	Derive (Create) [*filename*]
-D[*filename*]	NoDerive (NoCreate) [*filename*]
-e	EvenPages Yes
-E	EvenPages No

-f [*name*]	PrintFile
-F[*name*]	Printer
-g	Go
-i	Index
-I [*name*]	Input named `.derive` file
-k	KeepPrintSettings Yes
-K	KeepPrintSettings No
-l	LowRes Yes
-L	LowRes No
-m	Convert a MIF file to a document file
-M	Convert a document file to a MIF file
-o	OddPages Yes
-O	OddPages No
-p [*filename*]	Print [*filename*], where [*filename*] is optional
-P [*filename*]	NoPrint [*filename*]
-q	Quit
-r	Reverse Yes
-R	Reverse No
-s	Show current action list
-t	TOC
-v	Verbose
-V	Terse
-x	Reset
-X	Clear
-y	Collate Yes
-Y	Collate No
-z [*number*]	Scale (see **-C**)

Book File Syntax Description

This section describes the syntax of a book file. Notice that:

- An asterisk (*) indicates that any number of instances of this statement may appear in succession (for example, you can put several <File...> statements in one <Book...> statement, as shown below).
- You can put <Comment...> statements between other statements.
- `fmbook` will ignore any text not within angle brackets (< >).
- Case is not significant in an unquoted string.

Some statements in a book file contain arguments. These arguments take the following values:

Argument	**Values**
{boolean}	Yes or No
{maybe}	Yes, No, or AsNeeded
number	Any integer
"string"	Any text string, enclosed in quotes

Book File Syntax

A book file consists of the following:

```
<BookFile>
<Book
     <File...>*
     <Comment
      These optional statements set default printing options:
     >
     <Copies    number>
     <Collate   {boolean}>
     <Scale     number>
     <LowRes    {boolean}>
     <Reverse   {boolean}>
     <OddPages  {boolean}>
     <EvenPages {boolean}>
     <PrintFile "string">
     <Printer   "string">
>
<DerivedDocument...>
```

The statements in a book file have the following meanings:

Statement	Meaning
<BookFile>	Identifies this as a book file to fmbook; required in the first line of a book file.
<Book...>	Contains the specification of the book in a series of <File...> statements.
<File...>	Contains the specification of one file that is part of the book; a separate <File...> statement is required for each file in the book.
<Copies number> <Collate {boolean}> <Scale number> <LowRes {boolean}> <Reverse {boolean}> <OddPages {boolean}> <EvenPages {boolean}> <PrintFile "string"> <Printer "string">	Specify file printing options; override settings in FrameMaker file.

<File...> Syntax

A <File...> statement consists of the following:

```
<File
    <FileName "string">
    <DerivedDocument...>
    <StartPageNum {startpagespec}>
    <PageNumStyle {pagestylespec}>
    <PointPageNumStyle {pagestylespec}>
    <Str0 "string">
    <Str1 "string">
    <Str2 "string">
    <Str3 "string">
    <Str4 "string">
    <Str5 "string">
    <Str6 "string">
    <Str7 "string">
    <Str8 "string">
    <Str9 "string">
    <PageNumPrefix "string">
    <PageNumSuffix "string">
>
```

The statements in a <File...> statement have the following meanings:

Statement	Meaning
<FileName...>	Specifies the name of a FrameMaker file that contains a portion of the book.
<DerivedDocument...>	fmbook creates this portion of the book from the contents of the rest of the book. Either a <DerivedDocument...> statement or a <FileName...> statement appears in a <File...> statement, but not both.
<StartPageNum {startpagespec}>	Determines starting page number of the file; overrides page numbering found in the file. {startpagespec} can be an integer, NextPageNum, NextEven, NextOdd, or FromDocument.
<PageNumStyle {pagestylespec}> *and* <PointPageNumStyle {pagestylespec}>	Determines page numbering style; overrides page numbering found in the file. {pagestylespec} can be arabic, numeric, lcroman, ucroman, lcalpha, ucalpha, or FromDocument.
<Str[0-9] "string">	Associates strings with the file, for later reference in the extraction process. For more information, see the <DerivedDocument...> Syntax section later in this chapter.
<PageNumPrefix "string">	Specifies the prefix to the page number of the file (affects contents and index page references only).
<PageNumSuffix "string">	Specifies the suffix to the page number of the file (affects contents and index page references only).

<DerivedDocument...> Syntax

A <DerivedDocument...> statement, whether in a separate `.derive` file or embedded in a book file, may contain:

```
<DerivedDocument
    <FileName         "string">
    <FileNameSuffix   "string">
    <DefaultDerive    {maybe}>
    <A_List>
    <An_Index>
    <MifToMaker       {boolean}>
    <PreludeFileName  "string">
    <PostludeFileName "string">
    <Filter           "string">
    <FilterParm0 "string">
    <FilterParm1 "string">
    <FilterParm2 "string">
    <FilterParm3 "string">
    <FilterParm4 "string">
    <FilterParm5 "string">
    <FilterParm6 "string">
    <FilterParm7 "string">
    <FilterParm8 "string">
    <FilterParm9 "string">
    <ExtractTag  {extractspec}>*
    <ExtractMark {extractspec}>*
>
```

The statements in a <DerivedDocument...> statement have the following meanings:

Statement	**Meaning**
<FileName "string">	Specifies the name of the output file that contains the derived information.
<FileNameSuffix "string">	Indicates that the output filename should be the input filename, truncated back to the last period (if any) after the last slash (if any), with the specified suffix.
<DefaultDerive {maybe}>	Yes causes fmbook to schedule the document automatically when it reads the book file. AsNeeded causes fmbook to schedule the document only if the book file or any of the files in the book is newer than the existing derived file. No requires the user to schedule creation of the document specifically when running fmbook.
<A_List> and <An_Index>	Specifies filter, prelude file, and postlude file for a standard table of contents or index; subsequent statements can override these settings. See <A_List> Statement and <An_Index> Statement later in this chapter.
<MifToMaker {boolean}>	Yes causes `fmbook` to convert output to Maker format; No leaves output format determined by the filter (`fmlister` and `fmindexer` both output MIF).
<PreludeFileName "string">	Specifies the file that the specified filter copies in front of the extracted data.
<PostludeFileName "string">	Specifies the file that the specified filter copies after the extracted data.
<Filter "string">	Specifies the name of the filter (such as `fmlister` or `fmindexer`) that processes the extracted data.

<FilterParm[0-9] "string">	Specifies parameters that fmbook will pass to the specified filter.
<ExtractTag {extractspec}>	Causes fmbook to extract the contents of paragraphs with a specific tag; see {extractspec} Syntax later in this chapter.
<ExtractMark {extractspec}>	Causes fmbook to extract the contents of markers of a specific type; see {extractspec} Syntax later in this chapter.

The only required field for a <DerivedDocument...> is <FileName...> (or <FileNameSuffix...>), though you won't get any significant output without at least one <ExtractTag...> or <ExtractMark...> statement.

<A_List> Statement

The <A_List> statement is shorthand for:

```
<PreludeFileName "ListPrelude">
<PostludeFileName "ListPostlude">
<Filter "fmlister">
<MifToMaker Yes>
```

<An_Index> Statement

The <An_Index> statement is shorthand for:

```
<PreludeFileName "IndexPrelude">
<PostludeFileName "IndexPostlude">
<Filter "fmindexer">
<FilterParm0 "IndexSpec">
<MifToMaker yes>
```

{extractspec} Syntax

An {extractspec} can contain:

```
<Name "string">
<CleanUpWhiteSpace {boolean}>
<Output {outputspec}>
<ListerOutput "string">
<IndexerOutput>
```

The statements in an {extractspec} have the following meanings:

Statement	Meaning
<Name "string">	Specifies the tag name or marker type that fmbook extracts; required statement in an {extractspec}.
<CleanUpWhiteSpace {boolean}>	Affects the output commands; see {outputspec} Syntax later in this chapter.
<Output...>	Specifies the output from fmbook. {outputspec} parameters are explained below; you must include either an <Output...>, <ListerOutput...>, or <IndexerOutput> statement in an {extractspec}.
<ListerOutput "string">	Use in place of an <Output...> statement for the standard table of contents or other list; see {outputspec} Syntax later in this chapter.
<IndexerOutput>	Use in place of an <Output...> statement for the standard index; see {outputspec} Syntax later in this chapter.

{outputspec} Syntax

An {outputspec} can contain one or more of the following:

Statement	**Meaning**
<Name>	Specifies the tag name or marker type.
<Text>	Contains paragraph or marker text; if <CleanUpWhiteSpace Yes> statement appears, fmbook removes leading and trailing white space, and turns embedded spaces and hard returns into single spaces (fmbook does not alter tabs).
<PageNum>	Specifies formatted page number on which fmbook found the paragraph or marker.
<Order>	Identifies relative page number, counting the first page of the book as page 1, on which fmbook found the paragraph or marker.
<AutoNumString>	Contains numbering string associated with an automatically numbered paragraph (nothing for an unnumbered paragraph and a marker).
<String "string">	Specifies a text string.
<Tab>	Specifies a real tab (0x09); use to separate information on an output line.
<NewLine>	Specifies a real newline (0x0A); use to terminate output line.
<Str[0-9]>	Identifies associated string specified in the <File...> statement.
<PageNumPrefix>	Identifies associated string specified in the <File...> statement.
<PageNumSuffix>	Identifies associated string specified in the <File...> statement.
<PageRefStyle>	Specifies marker page numbering type (0, 1, 2, 3 for None, Single Page, Start of Page Range, End of Page Range, respectively); when used in an <ExtractTag...> statement, outputs 0.
<PageRefFont>	Specifies marker page numbering font (0 for regular text, plus 1 for Bold, plus 2 for Italic, plus 4 for Underline, plus 8 for Strikeout); when used in an <ExtractTag...> statement, outputs 0.

<ListerOutput...> Statement

The <ListerOutput...> statement, which specifies the output for a standard table of contents or other list, is an abbreviation for:

```
<CleanUpWhiteSpace Yes>
<Output
     <String "tagname"> <Tab>
     <AutoNumString> <Text> <Tab>
     <PageNumPrefix> <PageNum> <PageNumSuffix>
     <NewLine>
>
```

<IndexerOutput> Statement

The <IndexerOutput> statement, which specifies the output for a standard index, is an abbreviation for:

```
<CleanUpWhiteSpace Yes>
<Output
     <Text> <Tab>
     <Order> <Tab>
     <PageRefStyle> <Tab>
     <PageRefFont> <Tab>
     <PageNumPrefix> <PageNum> <PageNumSuffix>
     <NewLine>
>
```

Common Problems

This section describes some common problems you may encounter when you start using `fmbook`. Each section below includes a description of the symptoms you may see and their causes, with specific instructions for solving the problem.

Problems Formatting the Standard Table of Contents

The default format of the table of contents assumes that the entries are automatically numbered paragraphs. If your entries are not automatically numbered, the page numbers appear immediately to the right of the text on each line of the table of contents, as shown below:

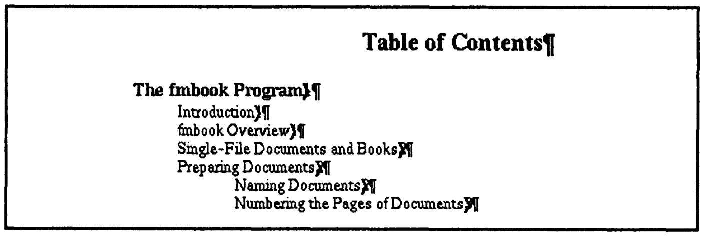


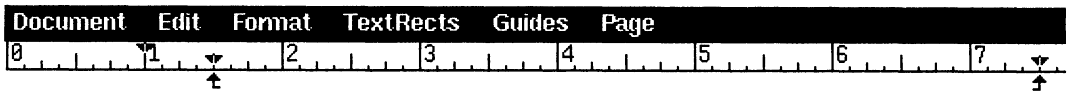
Table of Contents¶

The fmbook Program〉¶
Introduction〉¶
fmbook Overview〉¶
Single-File Documents and Books〉¶
Preparing Documents〉¶
Naming Documents〉¶
Numbering the Pages of Documents〉¶

With Text Symbols display on, as in the example above, the tab symbol (〉) obscures the page number.

To solve this problem:

1. Place an insertion point in the first entry in your table of contents. The tab settings that appear under the ruler at the top of the document window look something like this:



2. Remove the leftmost tab.
3. Display the Paragraphs dialog, apply to Current, Tag, and Catalog, and then click OK. All paragraphs in the table of contents that have this tag are now formatted correctly.
4. Repeat steps 1 through 3 for each tag in the table of contents.

If you want `fmbook` to use the new format automatically whenever it creates a table of contents, see *Changing the Format of a Table of Contents* earlier in this chapter.

Table of Contents Entries in the Wrong Order

If the table of contents entries appear in the wrong order, the document probably contains multiple TextRects on a page. `fmbook` searches a page in *draw order*, (back to front) rather than top to bottom. You can solve this problem one instance at a time, by using the Back command (in the Tools window) to change the draw order. If you do so, redisplay the page and note carefully the order in which FrameMaker draws the TextRects on the page. We suggest, however, that you avoid this problem completely by placing the headings in the main TextRect or in anchored frames, rather than in separate TextRects.

Blank Entries in the Table of Contents

If the table of contents contains blank entries, an empty paragraph (with a tag name that `fmbook` extracted for the Table of Contents) was probably left in the FrameMaker document. For more information, see *Preparing Documents for a Table of Contents* earlier in this chapter.

Extra Paragraphs after the Table of Contents or Index Title

If the table of contents or index contains an extra empty paragraph just below the title, an extra paragraph mark was probably left in the template when the prelude file was created. To solve this problem:

1. Repeat the procedure for creating your own `ListPrelude` or `IndexPrelude` file, described earlier in this chapter, in *Changing the Format of a Table of Contents* and *Changing the Format of an Index*.
2. Before you save the format in Maker Interchange Format, turn Text Symbols on (from the Guides menu). Your table of contents or index template should contain only a title, whose last line ends with an end-of-flow symbol (§) rather than a paragraph mark (¶). If the last line of the title contains a paragraph mark, delete it.

Question Marks in Index Entry Page References

Question marks can appear as a page reference in an index entry (for example 23-?? or ??-35). This happens when your marker has a Numbering Style of Start of Page Range but there is no corresponding marker for End of Page Range, or vice versa. To solve this problem, check the Numbering Style of the marker that created the entry. If it is correct, add a marker where needed to specify the other end of the page range.

fmbook Error Messages

Bad .derive Commands

`fmbook` error messages containing the words `bad .derive commands` indicate an error in either a `.derive` file or a book file. Check the following:

- Did you save the files in ASCII text format (as Text Only)?
- Are there spelling errors on the line indicated in the first error message?
- Are angle brackets (usually right angle brackets) missing in the line indicated in the first error message or in the line just above that?

MIF

`fmbook` error messages containing the acronym MIF (for Maker Interchange Format) indicate an error in the prelude or postlude file. Check the following:

- Did you save the files in ASCII text format (as Text Only)?
- Did you delete all the lines down to the first # in the files?
- Did you accidentally delete any other lines? If you're not sure, follow the instructions in this chapter to create the files again.

Maker Markup Language

Introduction

Use the Maker Markup Language (MML) to enter formatted text documents into FrameMaker using a standard (nongraphic) text file. The MML language allows access to many, but not all, features of FrameMaker.

MML documents can define the following document constructs:

- Document page size and column layout (including all options that are available with the New command)
- Header and footer layouts (including all options available in the Headers & Footers dialog box)
- Paragraph Catalog (including all options available in the Paragraphs dialog box)
- Font changes (including most options available in the Fonts dialog box)
- Document text with varying fonts and paragraph formats (all document text in an MML file is assumed to comprise one text flow)
- Anchored frames containing graphics, embedded in the document text

MML documents cannot define the following document constructs:

- Unanchored frames and graphics
- Irregular column layouts
- Column layouts that vary from page to page
- Multiple text flows

If you need to create documents using standard text files, and those documents need to include FrameMaker constructs not supported by MML, use the Maker Interchange Format (MIF) instead (see Chapter 7).

With MML, you can create formatted documents using both a *GENCODE* style of markup, in which document format and content are separate, and a typesetting style of markup, in which formatting specifications are mixed with text.

This chapter contains:

- The procedure for creating an MML document and bringing it into FrameMaker
- An overview of MML file format and syntax
- A description of each MML statement
- Sample MML files
- MML error messages

Creating and Using MML Documents

An MML document is a standard ASCII text file containing MML statements and text broken up into paragraphs. Its filename suffix should be `.mml`. You can create the file with any standard text editor. You can also create it using FrameMaker as a simple text editor.

> **Caution:** If you use FrameMaker to create or edit an MML file, remember to select Text Only format in the Save dialog box when saving the file.

Once an MML file has been prepared and saved, it can be opened or imported into FrameMaker, provided that its filename ends with `.mml`. The `.mml` filename suffix alerts FrameMaker that the file needs to be *filtered* before it is brought in. FrameMaker uses an external filtering program called `mmltomif` to convert the MML file into a temporary MIF file. FrameMaker then reads the MIF version of the document and produces a fully editable FrameMaker representation. See Appendix D, *Customizing FrameMaker,* for more information on how `.mml` and other ASCII files are processed by the FrameMaker Open and Import commands.

After the MML document has been opened or imported, it can be modified, printed, and saved using any of FrameMaker's standard commands. Note, however, that any changes made to the document in FrameMaker are not reflected back in the original MML file. Thus, if the MML file is meant to serve as the master source for the document, all changes must be made to the MML file itself.

Overview of MML File Format and Syntax

The contents of an MML file consist of markup statements and regular text (that is, the textual content of the document).

Markup statements begin with a less-than symbol (<) and continue up to a balancing greater-than symbol (>). These symbols are referred to as left and right braces. For example, `<section>` might cause a new section to be started, while `<family Times>` would switch fonts. Case is not significant in statement names, so `<FaMiLy Times>` would work just the same.

All text not enclosed in braces is assumed to be regular document text. Within the regular text, adjacent nonblank lines are all considered to be in the same paragraph, and blank lines are used to separate paragraphs. Paragraph markup statements also signal paragraph boundaries.

If the text needs to contain a < or > character (that is, one that should appear in the FrameMaker document as opposed to indicating the start of a markup statement), it must be preceded by a backslash character (\< or \>).

The Sections of an MML File

An MML file is divided into the following sections, in this order:

- Optional identification line: Contains the statement `<MML n.n>`, where `n.n` represents the MML language version number. When `mmltomif` reads the file, this statement is ignored, and is not required.
- Optional Macro Definitions Section: Contains `<!DefineMacro>` and `<!DefineChar>` statements to define simple, parameter-less macros used in subsequent markup statements.
- Optional Font Definitions Section: Contains `<!DefineFont>` statements to define named sets of font attributes for use later within both regular text and other markup statements.
- Optional Paragraph Format Definitions Section: Contains `<!DefinePar>` and/or `<!DefineTag>` statements to define named sets of paragraph attributes for use in setting the format associated with regular text paragraphs.
- Optional Document Layout Section: Contains document layout statements used to define things like paper size, number and spacing of columns, and headers and footers.
- Document Text Section: The first non–white space character outside of a markup statement signals the start of the document text section. This section contains regular ASCII text intermingled with font, special character, anchored frame, and paragraph-related markup statements.

MML Statements

The general form of a markup statement is:

```
< StatementName OptionalDataFields >
```

The format of data fields is indicated using the following conventions:

`[char]`	is a single character (or a backslash equivalent, such as `\t` for tab).
`[string]`	is any sequence of characters enclosed by double quotation marks (`"abc"`). To use double quotation marks in a string, type `\"`.
`[commentstring]`	is any sequence of characters terminated by >.
`[name]`	is a simple alphanumeric string starting with a nonspace character and terminated by a space or >.
`[number]`	is an integer.
`[boolean]`	is `Yes`, `No`, `Y`, or `N` and all case variations of these.
`[measure]`	is a real number (which may contain a decimal point and digits to the right of the decimal point) followed by an optional units of measurement: `"` (for inch), `pt`, `cm`, `mm`, or `pica`.
`[unit]`	is `inch`, `in`, `cm`, `mm`, `pica`, `pc`, or `pt`
`[lrcd]`	is `left`, `l`, `right`, `r`, `center`, `c`, `leftright`, `lr`, `decimal`, or `d`

Control and Macro Statements

`<MML [commentstring]>`

It is suggested that an `<MML>` statement be placed at the start of all MML files. The comment string is ignored when the file is read. The `<MML>` statement is not required.

The following statements can appear anywhere in the input file:

`<Comment [commentstring]>`

Used to place comments in the MML file. The comment string is ignored when the file is read.

`<!DefineMacro [name] [string]>`

Causes all occurrences of the newly defined statement `<name>` to be replaced by the `[string]`. Note that `[string]` is rescanned each time `<name>` is encountered.

`<!DefineChar [char] [string]>`

Works like **`<DefineMacro>`**, except [char] is any single character, and will NOT be in <braces>. This statement facilitates the remapping of foreign or otherwise special keyboards. For instance, suppose that the Yen symbol is represented by character code 0xFE in the MML file, and further note that FrameMaker considers character 0xB4 to be the Yen symbol. We can give the statement **`<!DefineChar \xFE "<Character \\xB4>" >`** to cause MML to turn all 0xFE's into Yens for FrameMaker.

`<Include [string]>`

Reads in the file named in **`[string]`** as MML input. If the include filename is not a full pathname (one that starts with **/** or ~), the `mmltomif` program has a number of options to control where it searches for the include file. In the normal case, `mmltomif` is called by FrameMaker, and is instructed to search for include files in the directory containing the main MML file being processed. You can also run `mmltomif` directly to access other include file options. See Appendix D, *Customizing FrameMaker,* for more information.

`<Units [unit]>`

Establishes default units for all measurements. If a **`<Units>`** statement appears in the file, it must come before the Font Definitions section. (default = **`inch`**)

`<!Alias [newname] [currentname]>`

Creates a new statement name that is a synonym for an existing statement name. For example, you could do **`<!Alias !MD !DefineMacro>`** and then do **`<!MD bi "<bold><italic>">`**. You could then use **`<bi>`** within the document text to set the current font to bold italic.

`<EndOfInput>`

Causes all remaining text in the MML file to be ignored. This can be useful for debugging an MML file or for temporarily modifying an MML file so that only part of it is read by FrameMaker.

`<Message [string]>`

Causes the specified **`[string]`** to be printed on the message window. This is useful mainly for debugging.

Font Statements

MML's font statements provide a control of character font that is similar to the control provided by the FrameMaker Fonts command (see Chapter 3).

Any of the font statements can appear in the Font Definitions, Paragraph Definitions, Document Layout, and Document Text Sections of an MML file, except for `<!DefineFont>`, which can only appear in the Font Definitions Section.

`<family [name]>`

Change font family. `[family]` is typically `Times`, `Helvetica`, `Courier`, or `Symbol`. Case is significant in the family name. The complete list of family names comes from a configuration file. See Appendix D, *Customizing FrameMaker,* for information.
(default = `Times`)

`<italic>`
`<noitalic>`
`<bold>`
`<nobold>`
`<underline>`
`<nounderline>`
`<strike>`
`<nostrike>`

These statements turn on and off the various font style variations.

`<plain>`

Same as `<nobold> <noitalic> <nounderline> <nostrike>`.
(default style = `plain`)

`<oblique>`
`<nooblique>`

These statements are synonyms for `<italic>` and `<noitalic>`.

`<superscript>`
`<subscript>`
`<normal>`

These statements change the relative position of the text baseline. They work like the other font properties. For example:

`e<superscript>i*pi<normal>=-1` gives $e^{i*pi}=-1$

`<pts [number]>`

Changes font point size. For example, `<pts 10>` changes to 10pt.
(default = `12`)

`<!DefineFont [name] [fontstatements]>`
Executes the list of `[fontstatements]` (the statements defined in this section) and then establishes the current font characteristics as a named font type, so that henceforth, the `<name>` statement will cause all the same font settings in effect at the end of the original `<!DefineFont>` statement to return. See the sample file at the end of this chapter for examples of the `<!DefineFont>` statement and its use.

Paragraph Statements

MML's paragraph statements provide a control of paragraph formatting similar to the control provided by FrameMaker's Paragraphs command (see Chapter 3).

All paragraph statements can appear within the Paragraph Definitions Section and between paragraphs in the Document Text Section, except for `<!DefinePar>` and `<!DefineTag>`, which can appear only in the Paragraph Definitions Section.

`<par>`
Ends a paragraph. The font characteristics currently in effect do not change, nor do the paragraph settings. Two or more consecutive new lines act the same as `<par>`. A new paragraph does not begin until a real character is seen. The `<par>` statement is most useful within macro definitions.

`<LeftIndent [measure]>`
Changes the prevailing paragraph left indentation. Only the value in effect at the start of paragraph text affects that paragraph. (default = `0"`)

`<RightIndent [measure]>`
Similar to LeftIndent. (default = `0"`)

`<FirstIndent [measure]>`
LeftIndent for the first line of a paragraph. (default = `.25"`)

`<SpaceBefore [measure]>`
White space above the paragraph. Usually specified in points, as in `6pt`. (default = `0pt`)

`<SpaceAfter [measure]>`
White space below the paragraph. (default = `0pt`)

`<Leading [measure]>`
Space between lines within the paragraph. (default = `2pt`)

Caution: Be sure to remember that the `<SpaceBefore>`, `<SpaceAfter>`, and `<Leading>` values are *not* assumed to be in points. If you specify `<Leading 2>`, this means 2 units in the current default unit system, which is usually inches. So `<Leading 2>` would put 2" of leading between each paragraph line.

`<Alignment [lrc]>`
Alignment of paragraph lines: Left, Right, Center, or Left and Right. (default = `lr`)

`<AutoNumber [boolean]>`
If `<AutoNumber Yes>` is specified, there must also be a valid `<NumberFormat>` string (see the Paragraphs command for a description of Number Format strings). (default = `No`)

`<NumberFormat [string]>`
Ignored unless `<AutoNumber Yes>` is specified. (default = "")

`<Hyphenate [boolean]>`
See the Paragraphs command in Chapter 3. (default = `No`)

`<ColumnTop [boolean]>`
See the Paragraphs command in Chapter 3. (default = `No`)

`<WithNext [boolean]>`
See the Paragraphs command in Chapter 3. (default = `No`)

`<ForceFont [boolean]>`
Only useful within a `<!DefinePar>` statement. If a paragraph format is defined with `<ForceFont Yes>`, then the current font settings are stored as part of the paragraph definition. Whenever the defined paragraph format name is found in the Document Text section, the current font is set to the font characteristics present when the named paragraph was defined. (default = `Yes`)

`<Tolerance [number]>`
See the Paragraphs command in Chapter 3. (default = `4`)

`<BlockSize [number]>`
See the Paragraphs command in Chapter 3. (default = `1`)

`<TabStopType [lrcd]>`
Establishes the tab type for all subsequent defined tab stops until the next `<TabStopType>` statement. (default = `l`)

`<TabStopLeader [char]>`
Establishes the tab leader character for all subsequent defined tab stops until the next TabStopLeader statement. (default = `" "`)

`<TabStops <TabStop [dimension]> <TabStop [dimension]> >`
Defines a set of tab stops. Each `TabStop` substatement sets a tab stop at the indicated position. The tab type and associated leader character are determined by the most recent preceding `TabStopType` and `TabStopLeader` statements which may be freely intermingled among the `TabStops` substatements.

To clear all tabs, use an empty tab stop list: `<TabStops>`.

`<!DefinePar [name] [parstatements]>`

Executes the list of `[parstatements]` (the statements defined in this section) and then establishes the current paragraph settings as a named paragraph type. Henceforth, the `<name>` statement will cause all the same paragraph settings in effect at the end of the original `<!DefinePar>` statement to return. Paragraph settings include the current font settings if `ForceFont` is active. When an MML document is opened or imported by FrameMaker, the resulting FrameMaker document will contain a Paragraph Catalog entry for each paragraph format defined in the MML file using `<!DefinePar>` statements. When an MML paragraph whose format is `<name>` is brought in to FrameMaker, its tag will be `name`. See the sample file at the end of this chapter for examples of the `<!DefinePar>` statement and its use.

`<!DefineTag [name]>`

Similar to `<!DefinePar>`, in that it establishes a paragraph format name. However, unlike `<!DefinePar>`, `<!DefineTag>` does not cause a Catalog entry to be generated when the MML document is brought in to FrameMaker.

`<!DefineTag>` is useful when you intend to import an MML file into an existing FrameMaker document that already has the correct Paragraph Catalog set up. When an MML paragraph is brought in to FrameMaker preceded by a `<!DefineTag [name]>` statement, the FrameMaker document's Catalog will be searched for a paragraph format whose name matches `[name]`. If such a format is found, the MML paragraph's format will be set to match the corresponding format in the FrameMaker Catalog.

Note: MML tag names cannot have a space in them (although tag names in FrameMaker can).

Document Layout Statements

MML's document layout statements provide a control of document layout similar to the control provided by FrameMaker's New and Headers & Footers commands (see Chapter 3).

These statements may appear in the Document Layout Section:

`<PageWidth [measure]>`
See the New command in Chapter 3. (default = `8.5"`)

`<PageHeight [measure]>`
See the New command in Chapter 3. (default = `11"`)

`<TopMargin [measure]>`
See the New command in Chapter 3. (default = `1"`)

`<BottomMargin [measure]>`
See the New command in Chapter 3. (default = `1"`)

`<LeftMargin [measure]>`
See the New command in Chapter 3. (default = `1"`)

`<RightMargin [measure]>`
These margins are offsets from the corresponding edge of the paper and define the area occupied by text columns. (default = 1")

`<Columns [number]>`
See the New command in Chapter 3. (default = `1`)

`<ColumnGap [measure]>`
See the New command in Chapter 3. (default = `.3"`)

`<LeftHeader [string]>`
See the Headers & Footers command in Chapter 3. Establishes [string] as the left-aligned page header. (default = `""`)

`<CenterHeader [string]>`
`<RightHeader [string]>`
`<LeftFooter [string]>`
`<CenterFooter [string]>`
`<RightFooter [string]>`
Similar to `<LeftHeader>`. Other header/footer strings as documented in the Headers & Footers command.

`<HeaderFont>`
Establishes the current font characteristics as those to be used in all of the header and footer strings. Typically preceded by a set of font statements or a named font definition. (default = `Times 12 Plain`)

`<HeaderTopMargin [measure]>`
See the Headers & Footers command in Chapter 3. (default = `.5"`)

`<HeaderBottomMargin [measure]>`
See the Headers & Footers command in Chapter 3. (default = `.5"`)

`<HeaderLeftMargin [measure]>`
See the Headers & Footers command in Chapter 3. (default = `1"`)

`<HeaderRightMargin [measure]>`
See the Headers & Footers command in Chapter 3. These margins are offsets from the corresponding edge of the paper.
(default = `1"`)

`<HeaderPageNumberStyle [style]>`
See the Headers & Footers command in Chapter 3. Where [style] is one of **`Arabic`**, **`UCRoman`**, **`LCRoman`**, **`UCAlpha`** or **`LCAlpha`**. (default = **`Arabic`**)

`<FirstPageHeader [boolean]>`
See the Headers & Footers command in Chapter 3. (default = **`Yes`**)

`<FirstPageFooter [boolean]>`
See the Headers & Footers command in Chapter 3. (default = **`Yes`**)

`<DoubleSided [boolean]>`
See the Headers & Footers command in Chapter 3. No means single sided.
(default = **`No`**)

`<FirstPageLeft [boolean]>`
No means the first page is considered to be a right page. This statement is only meaningful if preceded by a **`<DoubleSided Yes>`** statement.
(default = **`Yes`**)

`<FirstPageNumber [number]>`
See the Headers & Footers command in Chapter 3. (default = **`1`**)

Document Text Statements

The document text section contains:

- Text outside of markup statements
- Font statements and references to named font sets (see <!DefineFont>)
- Paragraph statements and references to named paragraph formats (see <!DefinePar> and <!DefineTag>)
- <Character> statements (see below)
- Macros defined with <!DefineMacro> and <!DefineChar>

Special characters may be included in regular document text using a C-like convention: \t represents a tab, \n a hard return, and **`\xnn`** (where **`nn`** is a 1- or 2-digit hexadecimal number *terminated by a space*). This represents any character value in the range 20..7E and 80..FE (other values are ignored). Refer to Appendix B, *Character Set*, for an explanation of character code values.

`<Character [number]>`

Represents a character code value in the range 32..126 and 128..254 (\x20..\x7E and \x80..\xFE) Other values are ignored. To use hexadecimal values in a `<Character>` statement, be sure to leave a space between the number and the right bracket (i.e., `<Character \86 >`). Useful for entering characters outside the printing ASCII range. May occur within document text and within definitions of macros that are used within document text.

There are also the following predefined macros, which expand to appropriate `<Character>` statements: `<Tab>`, `<HardReturn>`, `<HardSpace>`, `<DiscHyphen>`, `<Bullet>`, `<Cent>`, `<Yen>`, `<Dagger>`, and `<DoubleDagger>`.

`<AFrame <BRect 0 0 w h> [other MIF Frame substatements]>`

Creates an anchored frame, with the anchor point following the character that precedes the `<AFrame>` statement. The `<AFrame>` statement must contain a `<BRect>` statement, giving the frame's width and height. The `<BRect>` statement is actually a Maker Interchange Format (MIF) statement (not an MML statement). Following the `<BRect>` statement there may be other MIF `<Frame>` statements, including the `<FrameType>` statement (used to define how the frame's position is related to the anchor point's position).

A minimal `<AFrame>` statement would be:

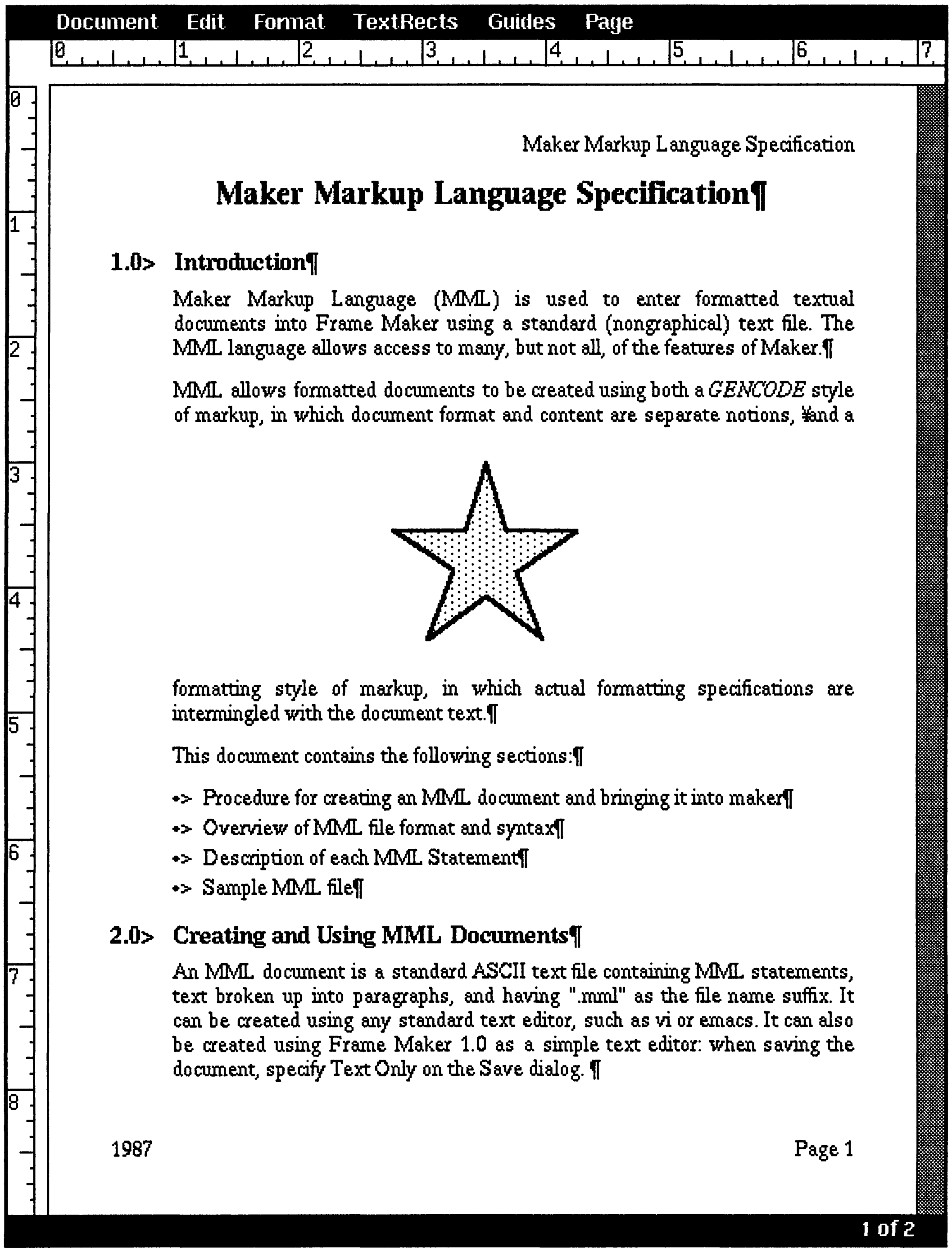
```
<AFrame <BRect 0 0 4" 2"> >
```

This statement places an empty 4"-by-2" anchored frame in the document (the default frame type is `<FrameType Below>`, which corresponds to the Below Current Line setting in the Anchored Frame dialog box (see the Anchored Frame command in Chapter 3).

The sample file at the end of this chapter contains an example of an `<AFrame>` statement that includes graphics. See Chapter 7 for a complete description of MIF graphics statements you can place within a MIF `<Frame>` statement.

Samples

This section contains sample files showing an MML representation of the document in the illustration below. First an include file containing formats is



Maker Markup Language Specification

Maker Markup Language Specification¶

1.0> Introduction¶

Maker Markup Language (MML) is used to enter formatted textual documents into Frame Maker using a standard (nongraphical) text file. The MML language allows access to many, but not all, of the features of Maker.¶

MML allows formatted documents to be created using both a *GENCODE* style of markup, in which document format and content are separate notions, ¥and a

formatting style of markup, in which actual formatting specifications are intermingled with the document text.¶

This document contains the following sections:¶

•> Procedure for creating an MML document and bringing it into maker¶
•> Overview of MML file format and syntax¶
•> Description of each MML Statement¶
•> Sample MML file¶

2.0> Creating and Using MML Documents¶

An MML document is a standard ASCII text file containing MML statements, text broken up into paragraphs, and having ".mml" as the file name suffix. It can be created using any standard text editor, such as vi or emacs. It can also be created using Frame Maker 1.0 as a simple text editor: when saving the document, specify Text Only on the Save dialog. ¶

1987 Page 1

shown, followed by the file containing the document text. The two files could be merged into one; we use an include file here simply to show how it is done and because keeping format information and document content in separate files is a useful technique. These files are included on the FrameMaker distribution tape, in the `docs` directory, so you can open them, copy them, and make your own variations.

```
 -------------------------- First File: Formats.mml --------------------------
<MML 1.00 - Formats.mml  Sample standard font, paragraph, and document
            formats>
<Comment  *** Define fonts for Title, Section, Body, Headers and Footers.
              Most of the defaults are good, so we just specify family,
              size, and style. "ft" stands for "font for Titles", "fs" is
              "font for sections," etc.>
<!DefineFont ft
            <Family          Times>
            <pts             18>
            <Bold>
>
<!DefineFont fs
            <Family          Times>
            <pts             14>
            <Bold>
>
<!DefineFont fb
            <Family          Times>
            <pts             12>
            <Plain>
>
<!DefineFont fhf
            <Family          Times>
            <pts             10>
            <Plain >
>
<Comment *** Set appropriate font for a Title paragraph and
                 define its format: >
<!DefinePar Title
            <ft>
            <Alignment       Center >
            <SpaceAfter      12pt>
            <ForceFont       Yes>
>

<Comment *** set font and define other paragraph formats>
<!DefinePar Section
            <fs>
            <Alignment       Left >
            <LeftIndent      0.50">
            <FirstIndent     0.00">
            <RightIndent     0.00">
            <Leading         0pt>
            <SpaceBefore     9pt>
            <SpaceAfter      9pt>
            <AutoNumber      Yes >
            <NumberFormat    "+.0\t">
            <TabStops <TabStop0.50"> >
>
<!DefinePar Body
            <fb>
            <Alignment       LeftRight >
            <LeftIndent      0.50">
            <FirstIndent     0.50">
            <RightIndent      0.00">
            <Leading          2pt>
            <SpaceBefore      0pt>
            <SpaceAfter       10pt>
            <ForceFont        Yes>
            <AutoNumber       No >
            <TabStops>
>

<!DefinePar BulletItem
            <Alignment        Left >
            <LeftIndent       0.75">
            <FirstIndent      0.50">
            <RightIndent      0.00">
            <Leading          2pt>
            <SpaceBefore      0pt>
            <SpaceAfter       3pt>
            <ForceFont        Yes>
            <AutoNumber       Yes>
            <NumberFormat     "\xA5 \t">
      <Comment *** \xA5 is the bullet character. \t is a tab character.>
            <TabStops
                              <TabStop   0.75">
            >
>

<Comment  *** Document Layout descriptions. Most of the default settings
              are good. >
<PageWidth                    7.00">
<PageHeight                   9.00">
<TopMargin                    0.75">
<BottomMargin                 0.75">
<LeftMargin                   0.50">
<RightMargin                  0.50">
<HeaderTopMargin              0.40">
<HeaderBottomMargin           0.46">
<HeaderLeftMargin             0.50">
<HeaderRightMargin            0.50">

-------------------------- Second File: Sample.mml --------------------------
<MML 1.00 -- Sample.mml   A sample mml file>

<Comment *** Include the font, paragraph,and document definitions from another
             file.  By keeping the formats in different files than the
             document text, all documents can be assigned a new format by
             just changing one file:
>
<Include "Formats.mml">

<Comment *** Define a few macros just to show how it is done.  Would normally
             put such standard macros in an include file:
>
<!DefineMacro                 if "<Italic>">
<!DefineMacro                 pf "<Plain>" >
<!DefineMacro                 bf "<Bold>"  >

<Comment *** Set up Headers and Footers. The next line sets the font.>
<HeaderFont <fhf>>
```

```
<RightHeader                "Maker Markup Language Specification">
<LeftFooter                 "1987">
<RightFooter                " Page #">

<Comment *** Start of Document Text ***>
<Title>
Maker Markup Language Specification

<Section>
Introduction

<Body>
Maker Markup Language (MML) is used to enter formatted textual documents into
FrameMaker using a standard (nongraphical) text file. The MML language allows
access to many, but not all, of the features of Maker.

<Comment *** The following Body paragraph contains an anchored frame.
            The AFrame statement is equivalent to a MIF Frame
            statement (see the MIF documentation for details).
            Inside the frame is a star. We just show this here
            so you can see how it is done.
 >

MML allows formatted documents to be created using both a <if>GENCODE
<pf>style of markup, in which document format and content are separate
notions, <AFrame <BRect 0 0 4 2> <FrameType Below>
  <Polygon
   <Pen 0> <PenWidth '1.0'> <Fill 6> <Inverted No >
   <NumPoints 10>
   <Point  2.03" 0.29"> <Point  2.19" 0.83"> <Point  2.76" 0.83">
   <Point  2.28" 1.17"> <Point  2.49" 1.71"> <Point  2.03" 1.36">
   <Point  1.56" 1.71"> <Point  1.76" 1.15"> <Point  1.28" 0.83">
   <Point  1.86" 0.83">
  > # end of Polygon
> and a formatting style of markup, in which actual formatting
specifications are intermingled with the document text.

This document contains the following sections:

<BulletItem>
Procedure for creating an MML document and bringing it into maker

Overview of MML file format and syntax

Description of each MML Statement

Sample MML file

<Section>
Creating and Using MML Documents

<Body>
An MML document is a standard ASCII text file containing MML statements,
text broken up into paragraphs, and having ".mml" as the file name suffix.
It can be created using any standard text editor, such as vi or emacs.
It can also be created using FrameMaker 1.1 as a simple text editor: when
saving the document, specify Text Only on the Save dialog box.
<Comment *** Would be followed by additional such lines>
```

MML Interpreter Messages

When FrameMaker reads an MML file, it invokes a filter program called `mmltomif`. `mmltomif` interprets the MML file and produces a temporary MIF file. While `mmltomif` is reading the MML file it may detect unexpected character sequences (sometimes referred to as errors). It responds by printing messages on the message window. Even if it finds an error, `mmltomif` will continue to process the MML file and do its best to read in as much of the document as possible.

This section lists the messages produced by `mmltomif`, along with their meanings. Words in *`this font`* indicate variable words in a message. A line number is printed along with most messages when they appear on the screen.

General Messages

`Bad boolean:` *`UnexpectedString`*

`mmltomif` expected to see **`Yes`** or **`No`**. The value **`No`** is assumed.

`Bad digit: '%c'.`

`mmltomif` encountered a nonnumeric character when expecting a number.

`Bad hex digit:` *`DigitChar`*

In the middle of a hex number, there is a nonhex character. The result is not likely to be what was intended.

`Bad lrcd:` *`UnexpectedString`*

`mmltomif` expected to see **`Left`**, **`Right`**, **`Center`**, **`Decimal`**, or **`LeftRight`**.

`Bad real number:` *`UnexpectedChar`*

A non–real number character appeared in the middle of a real number.

`Bad unit:` *`UnexpectedString`*

`mmltomif` expected to see a valid unit (**`inch`**, **`cm`**, etc.).

`Bad style:` *`UnexpectedString`*

`mmltomif` expected to see **`arabic`**, **`lcroman`**, **`ucroman`**, **`lcalpla`**, **`ucalpha`**.

`Could not open` *`filename`*

The named include file could not be found.

`Did not find end of comment on line` *`LineNumber`*

The comment that began on the specified line did not end by the time the file was completely read.

`Expected string, not` *`UnexpectedChar`*

`mmltomif` was expecting to see a string starting with a double quotation mark

(") rather than the unexpected character shown in the message.

`Junk at end of command:` *`JunkString`*

`mmltomif` expected to see a right brace character (>).

`Keyword too long`

While looking for a macro name or other keyword, a very long token was found.

`MML MSG:` *`MessageString`*

Not an error—generated by a user **`<Message>`** statement.

`Never finished defining '%s'.`

`mmltomif` encountered a <!Define...> statement within a <!Define...> statement (for example, a <DefineChar> statement within a <!DefinePar> statement). You must finish the first <!Define...> statement before beginning another.

`mmltomif` ignores the first <!Define..> statement and continues reading the file. The results, however, are unlikely to be what you had intended.

`Null name not allowed`

The empty macro name (<>) is not allowed.

`String too long`

A very long string was found. The maximum string length is 1000 characters; characters beyond that are truncated.

`Tab commands only allowed within <TABSTOPS...>`

The statements **`<TabsStopType>`**, **`<TabStopLeader>`**, and **`<TabStop>`** can appear only within a **`<TabStops>`** statement.

`Undefined:` *`MacroName`*

There is no definition for this macro. The undefined macro is ignored.

`Unexpected right bracket`

A right brace with no matching left brace has appeared.

`\x needs ending space`

Characters specified with \x must end with a space. Similar to `Bad hex digit` message.

Command Line Messages

`Bad Option` *`CommandLineOptionChar`*

`mmltomif` did not recognize this option character. The option is ignored.

`Could not find` *`FileName`*

Could not find the specified input file. `mmltomif` exits.

`Could not write` *`FileName`*

Could not open the specified output file for writing. `mmltomif` exits.

`Too many -I options`

`mmltomif` exits. The maximum number of -I options is 100.

`Usage: mmltomif [-Ipath] [input [output]]`

`mmltomif` displays this if it is started without any parameters.

Fatal Error Messages

`Could not open tmp file`

`mmltomif` could not open its temporary file.

`Hash table overflow`

Too many macros defined. The maximum number is 1000.

`Input stack overflow`

Too many nested include files (maybe in an include loop). The maximum nesting depth is 100.

`internal error`

MML has encountered an internal error. Please call Frame Technical Support.

`Out of memory!`

`mmltomif` was unable to complete the translation of the MML file because it ran out of memory. To free memory for `mmltomif`, quit some FrameMaker document windows or kill other processes.

`Tab buffer overflow`

Too many different tab stop settings. There can be up to 40 tab stops for a paragraph.

Maker Interchange Format

Maker Interchange Format (MIF) is a group of statements that describe all text and graphics understood by FrameMaker in an easily parsed, readable text file. MIF provides a way to exchange information, from a simple bar graph to a complex document, between FrameMaker and other applications while preserving graphics and document structure and format. You can write programs that convert graphics or documents into a MIF file and then import the MIF file into FrameMaker with the graphics and document formats intact.

MIF and FrameMaker

Using FrameMaker's Save command, you can save a FrameMaker document in MIF. After FrameMaker saves a MIF file (in the form described in this chapter), it starts a shell script. (For more information on the shell script and where it is stored, see *Filters for Import, Open, and Save* in Appendix D.)

Typically, this shell script does nothing to the MIF file. You can, however, change the script so that when you save a file with a particular file suffix, it executes a filter you have written. For example, the script could execute a filter to translate the MIF file to DCA format.

FrameMaker follows similar steps when reading a file. When you open or import a file, FrameMaker checks if it is a MIF file or if it has a suffix listed in a configuration file (again, see Appendix D for more information). If it's a MIF file, FrameMaker reads the file directly, translating it into a FrameMaker document.

If the file has a suffix listed in the configuration file, FrameMaker starts a shell script that executes a filter based on the file's suffix. The filter then converts the foreign file to a MIF file. FrameMaker interprets the MIF file, storing the results in a FrameMaker document.

Because FrameMaker uses a shell script instead of reading the file directly, you can customize FrameMaker to start automatically a filter you have written.

What This Chapter Contains

This chapter contains an overview of MIF, including its file layout and syntax conventions, followed by descriptions of the statements and their syntax. The statements are described in the order that they would most likely occur in a MIF file. If a statement (such as <Font>) can occur in many places in the file, we describe it in the order that it might *first* occur.

A good way to become familiar with MIF is to examine examples of MIF files. You can look at a MIF file by saving a FrameMaker document in MIF (see the Save command in Chapter 3); keep in mind, however, that the files FrameMaker produces often contain more information than other programs need to generate.

MIF File Overview

There are three kinds of MIF statements: control, markup, and comment. Control statements allow you to include files and define macros you can use in markup statements. The MIF file identification line, the <Units> statement, and the <Verbose> statement are also considered control statements.

Markup statements define structural units, graphics, document parameters, special characters, and other component information. Document text is embedded within markup statements.

There are two ways to include comments in a MIF file: you can use a <Comment> statement, described later in this chapter, or you can put comments on a line after a number symbol (#). We recommend, however, that you use the <Comment> statement. See *Comment Statement* later in this chapter.

MIF File Layout

A MIF file consists of the following sections:

- MIF file identification line
- Macro statements
- MIF defaults
- Paragraph Catalog
- Anchored frames
- Document template definitions
- Page layout descriptions (including graphics but excluding text flows)
- Document text flows

Each section, except the MIF file identification line, is optional. Most sections have a main statement which uses substatements to describe document elements.

The beginning of the file has three purposes: to identify the file and the conversion program, to read in and define macros, and to establish the units used and the debugging setting (<Verbose>).

After that, the file contains the Paragraph Catalog, which is a series of <Pgf> statements within one <Catalog> statement. The Paragraph Catalog in MIF is exactly like the one within FrameMaker: Each <Pgf> statement defines settings and has a tag to identify it.

After the Catalog, the file may include a <Document> statement, which controls features such as page size, margins, headers and footers. Since the MIF interpreter assumes the same page defaults that the FrameMaker New command does, this statement is necessary only if you want to override those default settings.

Next is the <Dictionary> statement, which lists words in the user dictionary.

Also at the beginning of the file, before any document text, is the <AFrames> statement, which describes all anchored frames in the document as well as their contents. Further down in the file, where the document contents are described, the file must include a short <AFrame> statement for each anchored frame; this short version need only supply an ID number to match it to the frame's description in the first <AFrame> statement (for example, <AFrame 4>).

The <Page> statement describes the page layout of each page, including its TextRect dimensions and text flow, as well as the objects and frames on that page. A MIF file should include a <Page> statement for each page in the document, including the master page. (There should be only one master page <Page> statement.)

The <TextFlow> statement represents the actual text in the document. The text is expressed in units of paragraphs <Para> and paragraph lines <ParaLine>.

Current State

The MIF interpreter keeps track of certain states within the document:

- current paragraph format
- current font
- current page
- current frame
- current TextRect
- current fill pattern
- current border pattern
- current line width

Markup Statement Syntax Conventions

In the statement descriptions that follow, we will use these conventions to describe syntax:

<token data>

Token represents one of the MIF keywords (such as `Catalog`) listed in the following statement descriptions; *data* is one or more numbers, a string, a keyword, or nested markup statements.

All markup statements, as well as all data portions of a markup statement, are optional. If the token following the left bracket (<) is not recognized, all text up to the corresponding right bracket (>) is ignored.

There are several general types of data items in a markup statement. We use the terms and symbols in the following section to identify data items:

`string`	-	Left apostrophe (`` ` ``), zero or more characters, right apostrophe ('). Example: `` `ab cdef ghij' `` See the following paragraph for more information on special characters in string data.
`n`	-	An integer whose range depends on the associated token.
`d`	-	A decimal number signifying a dimension. If no units are specified, the current default unit is assumed (see the Units statement in "MIF Interpreter Controls," earlier in this chapter). You can name the units explicitly, as in `1.11"`, `72pt`, `8.3cm`, etc.
`w h`	-	A pair of dimensions representing width and height.
`x y`	-	Coordinates of a point. Same syntax as a dimension. Coordinates originate at the upper left corner of the enclosing frame.
`l t r b`	-	A set of coordinates representing left, top, right, and bottom indents.
`npt`	-	An integer representing number of points.
`item | item`	-	The vertical bar separates items in a list of possible data values and indicates that only one of the items can be in the statement.
`<token>`	-	Brackets around a single statement name indicate a nested statement. The entire expanded statement may occur at this point.

The character set used within MIF string data is FrameMaker's character set (see Appendix B, *Character Set*). However, because a MIF file must contain only ASCII characters and because of MIF parsing requirements, you must represent certain characters with backslash (\) sequences:

Character	**Representation**
Tab	`\t`
>	`\>`
'	`\q`
`	`\Q`
\	`\\`

Also, all FrameMaker characters whose value is above the ASCII range (greater than `\x7f`) are represented using `\x` notation in the string. The hex digits must be followed by a space. The following example shows a FrameMaker line and its representation in a MIF string:

FrameMaker:	Some `symbols': >\Ø¿!
MIF:	`` `Some \Qsymbols\q: \> \\\xaf \xc0 !' ``

MIF File Identification Line

The MIF file identification line must be the first line of the file, with no leading white space. The form of the line is:

```
<MIFFile n > # p
```

The number *n* indicates the version number of the MIF language used in the file, and *p* is a comment showing the name and version number of the program that generated the file. MIF is by design compatible across versions, so a MIF interpreter should be able to parse any MIF file, although the results may sometimes differ from the user's intentions.

Comment Statement

```
<Comment    comment text>
```

Programs can include comments in MIF files they create. FrameMaker ignores any text between the <Comment and the matching > (as well as text after the number sign). Here are some examples:

```
<Comment  -  The  following  statements  define  the
paragraph formats>


<Comment <These statements have been removed:
<Font <FBold> <FItalic>>
>>


# You can include comments after number signs, but we
# recommend using the Comment statement.
```

Macro Statements

MIF supports two macro statements that allow you to include information from other files. Although these statements usually appear near the top of a MIF file, you need not put them in this position.

The two macros are in this form:

- `define(`*name, replacement*`)`
- `include(`*file*`)`

In general, you would use an include statement to read a header file containing define statements that a filter needs to translate a file. By isolating the data in a header file, you can simplify the process of changing important mappings.

In all MIF files FrameMaker creates, the second line is

```
include(mif_read.m4)
```

This statement includes a macro file that ensures compatibility with earlier versions of FrameMaker. Versions after 1.1 ignore this macro file.

Notes on Macros

Macro names must be enclosed in brackets, just as statement names are. That is, after defining a macro such as `define(Bold, Font <FBold Yes>)`, you should then write `<Bold>` in MIF statements. (The interpreter will understand <Bold> to mean <Font <FBold Yes>>.) As in the rest of a MIF file, the left and right angle brackets must be balanced.

Units Statement

```
<Units              Uin    |        # assumed units for dimensions
                    Ucm    |
                    Umm    |
                    Upica  |
                    Upt
>
```

You can specify the default units used for dimensions and coordinates with the <Units> statement. If this statement isn't present, the default unit is `Uin`. A <Units> statement remains in effect until another valid <Units> statement is encountered. When FrameMaker writes a MIF file, it uses the document's current display units, set using the Units command on the Guides menu.

Verbose Statement

```
<Verbose            Yes | No>       # turn verbose on/off
```

Debugging a MIF input file (that is, a MIF file that FrameMaker is trying to open or import) can be difficult. Obvious errors are recorded in the message window. But when Verbose mode is turned on (<Verbose Yes>), FrameMaker writes a more detailed stream of processing descriptions to the file `miflog` in the user's home directory. This file can get quite long, but may be essential while writing a program to create MIF for input to FrameMaker. A <Verbose> statement may occur unnested, or within markup statements, as explained later in this chapter. A <Verbose> statement remains in effect until the interpreter encounters another valid <Verbose> statement.

Catalog Statement

```
<Catalog
     <Units>                        # Defined earlier in this chapter
     <Verbose>                      # Defined earlier in this chapter
     <Pgf>                          # Define paragraph formats
     <Pgf>
      ...
>
```

The next subsection defines the format of the <Pgf> statement.

Pgf Statement

```
<Pgf
     <PgfTag       string>          # Paragraph tag string
     <PgfLanguage  USEnglish |      # Hyphenation & Spell check language
                   UKEnglish |
                   French    |
                   Dutch     |
                   German    >
     <PgfAlignment Left   |         # Alignment
                   Center|
                   Right  |
                   LeftRight  >  <PgfAutoNumYes | No ># Autonumber
flag
     <PgfHyphenate Yes | No >       # Hyphenation flag
     <PgfColumnTop Yes | No >       # Column break flag
     <PgfWithNext  Yes | No >       # Orphan control flag
     <PgfForceFont Yes | No >       # Font forcing flag
     <PgfSplit     Yes | No >       # Straddles frozen pages
     <Font>                         # Paragraph's default font (see Font
                                    # Statement, next subsection)
     <PgfTolerance n>               # Hyphenation tolerance
     <PgfBlockSize n>               # Widow/orphan lines
     <PgfLIndent   d>               # Left margin
     <PgfFIndent   d>               # First line left margin
     <PgfRIndent   d>               # Right margin
     <PgfLeading   d>               # Line spacing within paragraph
     <PgfSpBefore  d>               # Space before paragraph
     <PgfSpAfter   d>               # Space after paragraph
     <PgfNumFormat string>          # Autonumber formatting string
     <PgfNumString string>          # Current paragraph's number
     <PgfNumTabs   n>               # Number of tab stops in para.
     <TabStop                       # A TabStop statement
        <TSX       x>               # Horizontal position of stop
        <TSType    Left    |        # TabStop alignment
                   Center  |
                   Right   |
                   Decimal >
        <TSLeader  n>               # TabStop leader character
                                    # code, in decimal
     >                              # End of TabStop statement
     <TabStop>                      # More TabStops as needed
>                                   # End of Pgf statement
```

As each <Pgf> statement is processed (within or outside a <Catalog> statement), its substatements affect the *current paragraph format* definition. When text is entered, it is formatted in the current paragraph format. To change the current paragraph format, you need to specify the changes explicitly in a <Pgf> statement or name the tag of a paragraph format stored in the Catalog.

For example, suppose a MIF file contains a <Pgf> statement that accepts the FrameMaker defaults instead of specifying paragraph formats explicitly. A <Pgf> statement after this one might explicitly change the indent values to <PgfLIndent 1.5 in> and <PgfRIndent 2 in>, leaving other paragraph formats unchanged. This second statement would change the current paragraph format definition, which means that if a <Pgf> statement with the tag "body" followed it, the body format would be changed to reflect the new indent values.

```
<Pgf
   <PgfTag  body>
>
<Comment   Current paragraph format is
defaults>
<Pgf
   <PgfTag  note>
   <PgfLIndent 1.5 in>
   <PgfRIndent 2 in>
>
<Comment   Current paragraph format is
defaults + indent changes>
<Pgf
   <PgfTag  body>
>
<Comment   Current paragraph format is
defaults + indent changes>
```

At the end of a <Pgf> statement, the current paragraph format definition is stored in the Catalog. Although it often isn't necessary, we recommend that all <Pgf> statements in a <Catalog> statement contain all possible <Pgf> substatements; by explicitly specifying each paragraph option, you can avoid unpredictable results. When FrameMaker writes a MIF <Pgf> statement, it specifies all substatements.

Font Statement

```
<Font
    <FFamily      string>       # Name of font family
                                # i.e., Times, Courier, etc.
    <FSize        n>            # Size in points
    <FPlain       Yes | No>     # Flag
    <FBold        Yes | No>     # Flag
    <FItalic      Yes | No>     # Flag
    <FUnderline   Yes | No>     # Flag
    <FStrike      Yes | No>     # Flag
    <FDX          npt>          # Horizontal kern pts
    <FDY          npt>          # Vertical kern pts
    <FDAX         npt>          # Horizontal origin advance
    <FNoAdvance   Yes | No>     # Flag
>                               # End of Font statement
```

All substatements in the <Font> statement are optional. Like the <Pgf> substatements, all <Font> substatements reset the interpreter's notion of the *current font.* The <FDX>, <FDY>, and <FDAX> statements control kerning, spreading, superscripting, and subscripting. <FDY> controls superscripting (a positive value results in superscript, a negative value in subscript). <FDAX> controls kerning and spreading by altering the character advance (the distance between a character's origin and the next character's origin); in FrameMaker, <FDAX> is affected by the Spread setting in the Fonts dialog box. <FDX>, which is affected by Alt-Left Arrow and Alt-Right Arrow, also controls horizontal movement.

The value of the kerning statements should almost always be zero. The best way to understand their use is to save a FrameMaker document in MIF and examine the MIF file.

Document Statement

The MIF file may contain a statement that sets certain document defaults. However, the MIF interpreter assumes the same defaults as the FrameMaker New command, so a <Document> statement, with one or more substatements, is necessary only to override unwanted default settings. In many applications, you can use the default settings because the MIF file doesn't need to control these settings. However, if a MIF generator is concerned with any of these settings, it should generate MIF to set them explicitly; the user can change the default document settings in FrameMaker any time.

The <Document> statement must precede all other statements that represent document content. You specify the parameters as markup statements nested within the overall document statement.

A <Document> statement does not need all these substatements, and the substatements can occur in any order.

```
<Document                                 # Document parameters
    <DPageSize      w h>                  # Paper size
    <DMargins       l t r b>              # Default text margins
    <DColumns       n>                    # Default number of columns
    <DColumnGap     d>                    # Default column gap
    <DHeadOnFirst Yes | No>               # Display header on first page
    <DFootOnFirst Yes | No>               # Display footer on first page
    <DStartPage     n>                    # Starting page number
    <DTwoSides      Yes | No>             # Two-sided layout
    <DFrozenPages Yes | No>               # True if Freeze Pagination on
    <DParity        FirstLeft |           # First page is left or right
                    FirstRight>
    >                                     # End of optional doc params
```

Dictionary Statement

The MIF file may contain a <Dictionary> statement, which lists all the words in the document's dictionary.

```
<Dictionary
    <OKWord 'word'>                       # Words in dictionary
    <OKWord 'word'>
>                                         # End of Dictionary
```

Anchored Frames Statement

The MIF file may contain an <AFrames> statement near the beginning, which lists the contents of all anchored frames. Later in the file, where an anchored frame actually occurs, a short <AFrame> statement provides only an ID matching the frame to the statement that described it before in the file.

```
<AFrames
    <Frame>                               # Complete Frame statements
>                                         # End of AFrames
```

For a description of the <Frame> statement, see *Object and Frame Statements* later in this chapter.

Page Statement

```
<Page
    <PageNum      n | n.n>       # Page num for additive pages
    <PageType     MasterPage |   # Kind of page
                  BodyPage>
    <Units>                      # Defined earlier in this chapter
    <Verbose>                    # Defined earlier in this chapter
    <PageSize     w h>           # Ignored by FrameMaker 1.1
    <Frame | Objects>            # Frames and/or objects in the page

    # Remaining items are for master pages only
    <Columns      n>             # Default number of columns
    <ColumnGap    d>             # Default column gap
    <HeaderL      string>        # Left header string
    <HeaderC      string>        # Center header string
    <HeaderR      string>        # Right header string
    <FooterL      string>        # L,C,R footer strings
    <FooterC      string>
    <FooterR      string>
    <HFMargins    l t r b>       # Header/footer margins
    <HFFont       <Font>>
    <NumStyle     Arabic  |      # Display style for page number
                  UCRoman |
                  LCRoman |
                  UCAlpha |
                  LCAlpha >
>                                # End of Page statement
```

FrameMaker uses the <PageNum> statement when it writes a MIF file, but it ignores it when reading a MIF file. Each <Page> statement adds a new page to the document.

Only one <Page> statement should have `<PageType MasterPage>`. If the document already has a master page, frames and objects in the master page <Page> statement will be added to the existing master page, and other properties in the <Page> statement will override the current settings.

Object and Frame Statements

Each <Page> statement has nested within it <Object> and <Frame> statements. If a frame contains objects and other frames, the frames and objects are listed in the order that they are drawn (object underneath first).

For the <Object> and <Frame> statements, the interpreter keeps track of the current page and current frame. If the interpreter encounters a <Frame> statement before a <Page> statement, it assumes the frame is on the current page. Similarly, if the interpreter encounters an <Object> statement before a <Frame> and/or <Page> statement, it assumes the object is on the current frame or page. When FrameMaker opens a MIF file, the default current page is page 1, and the default current frame is the page frame for page 1. When FrameMaker imports a MIF file, the default current page is the first page visible when the import command is invoked, and the current frame is the single currently selected frame on that page, if there is one; if there is no currently selected frame, then the current frame is the page frame for that page.

An object statement consists of the object type, generic data common to all objects, and specific data about the object. Its form is:

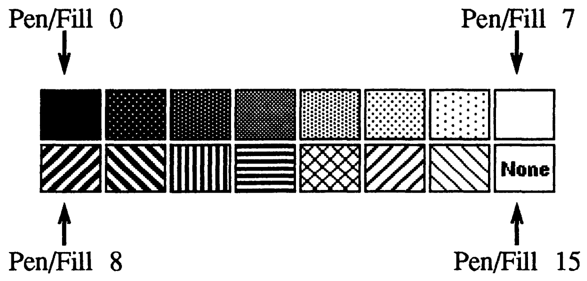
<ObjectType <GenericObjectData> <TypeSpecificData> >

Each object type and its corresponding data are described in the following subsections. The *<GenericObjectData>* consists of the following statements:

```
<ID          n>           # Object ID #
<Pen         n>           # Border pattern index
<PenWidth    n>           # Border line thickness in points
<Fill        n>           # Fill pattern index
<Invert      Yes | No>    # Invert pixop flag
<GroupID     n>           # ID of parent group object
<Units>                   # Defined earlier in this chapter
<Verbose>                 # Defined earlier in this chapter
```

The <ID> is necessary only if other objects refer to the object. For example, anchored frames, groups, and linked TextRects require the <ID>.

The <Pen> and <Fill> values refer to selections in FrameMaker's Tools window pattern table, as shown below:



You can customize FrameMaker to use patterns different from the standard set. See Appendix D, *Customizing FrameMaker*, for information on changing the definition of the fill and pen patterns.

<PenWidth> values are in points and can include decimal fractions. However, when the MIF file is read, <PenWidth> values map to one of the four pen widths available within FrameMaker. The four standard values are .5, 1, 3, and 4 points, though you can redefine them (see Appendix D, *Customizing FrameMaker*, for more information). Each <Pen>, <Fill>, <PenWidth>, or <Invert> statement resets the MIF interpreter's corresponding default value. If an Object statement

doesn't include one of these statements, the MIF interpreter will use the current default value for the object data.

The <GroupID> statement is necessary only if the object belongs to a group. All objects in the same immediate group have the same <GroupID>, which is the ID of the group's parent object.

In FrameMaker, patterns are not directly associated with a document, but rather with the FrameMaker application itself. Each FrameMaker document simply contains indexes into FrameMaker's current pattern and line width table, so you cannot define a document's patterns directly; you can only specify the values 0 through 15.

Descriptions of each type of object statement follow.

Arc Statement

```
<Arc
    <GenericObjectData>          # Defined above
    <ArcRect      l t w h>       # Underlying ellipse rect
    <ArcTheta     n>             # Start angle
    <ArcDTheta    n>             # Arc angle length
>                                # End of Arc Statement
```

The arc is a segment of an ellipse whose bounding rectangle is defined in <ArcRect>. <ArcTheta> specifies the starting point of the arc in degrees. Zero corresponds to 12 o'clock, 90 to 3 o'clock, 270 to 9 o'clock, etc. <ArcDTheta> corresponds to the length of the arc, with positive and negative values corresponding to clockwise and counter-clockwise extents, respectively.

Arrow Statement

```
<Arrow
    <GenericObjectData>          # Defined at beginning of section
    <TailPoint    x y>           # Coordinate of Tail
    <HeadPoint    x y>           # Coordinate of Head
>                                # End of Arrow
```

Ellipse Statement

```
<Ellipse
    <GenericObjectData>          # Defined at beginning of section
    <BRect        l t w h>       # Position, in enclosing
                                 # page/frame coordinates
>                                # End of Ellipse statement
```

The <Ellipse> statement describes circles and noncircular ellipses. <BRect> values specify the left, top, width, and height of the rectangle that encloses the ellipse.

ImportObject Statement

```
<ImportObject
    <GenericObjectData>            # Defined at beginning of section
    <BRect       l t w h>          # Position, in enclosing
                                   # page/frame coordinates
    <ImportObFile string>          # Object's filename
    <ImportObType BitMap>          # Bitmap includes EPS files
    <BitMapDpi   n>                # Scaling info for image file
    >                              # End of ImportObject statement
```

An `ImportObject` is an object whose contents are defined in an external file. Although `ImportObType` can only be `BitMap`, this type includes both bitmap files and Encapsulated PostScript (EPS) files. When reading the image file, FrameMaker determines whether the file is a bitmap or EPS file.

<BRect> values specify the left, top, width, and height of the object's bounding rectangle. <ImportObFile> specifies the name of the file containing the imported data.

For imported image files (both bitmaps and EPS files), you can use the <BitMapDpi> field to override the <BRect> width and height when the object is printed. For bitmaps, this value sets the image's printing resolution (in dots per inch). For EPS files, however, it sets the scaling factor—a value of 72 prints the image at 100% (the actual size of the image in the file); a value of 144 prints the image at twice the size.

For more information on importing image files, see the Import command in Chapter 3.

Frame Statement

```
<Frame
    <GenericObjectData>         # Defined at beginning of section
    <BRect       l t w h>       # Position, in enclosing
                                # page/frame coordinates
    <FrameType   NotAnchored |  # Whether frame is anchored
                 Inline   |     # If anchored, where?
                 Top      |
                 Below    |
                 Bottom   |
                 Left     |
                 Right    |
                 Near     |
                 Far      >
    <BLOffset    d>             # BaseLine Offset
    <NSOffset    d>             # Near Side Offset
    <AnchorAlign Left     |     # Only for anchored frames
                 Center   |
                 Right    >
    <Cropped     Yes | No >     # Per dialog box
    <Object | Frame....>        # 0 or more object or frame
                                # statements, listed in draw order.
>                               # End of Frame statement
```

Unless the <GenericObjectData> indicates otherwise, the MIF interpreter assumes a frame has no pen (Pen = 15), a pen width of 1, and a fill of index 7 (white). <BRect> values specify the left, top, width, and height of the frame. The assumed <FrameType> is `NotAnchored`. In fact, if a <Frame> statement is within a <Page> statement, the frame can't be an anchored frame (all the anchored frames are described in the <AFrames> statement). Usually, a <Frame> statement contains a list of <Object> and <Frame> statements, defining the contents of the frame, and listed in draw order from back to front.

Group Statement

```
<Group
    <GenericObjectData>         # Defined at beginning of section
>                               # End of Group statement
```

The <Group> statement uses only the <ID> and <GroupID> statements. A group object gathers elemental objects and allows groups to be nested. When the MIF interpreter encounters a <Group> statement, it searches all objects within the current frame for those whose GroupID matches the ID of the <Group> statement. These objects are then collected to form the group. All objects with the same GroupID must be listed in the MIF file before their associated <Group> statement is listed. If multiple <Group> statements have the same ID, the results will be unpredictable.

Polygon Statement

```
<Polygon
    <GenericObjectData>          # Defined at beginning of section
    <Smoothed     Yes | No>      # Smoothing on?
    <NumPoints    n>             # Number of vertices
    <Point        x y>           # Frame-relative point position
    ....                         # More Points as needed
>                                # End of Polygon statement
```

If the <Smoothed> statement is not present, the default is <Smoothed No>. The <NumPoints> statement is optional (FrameMaker determines the number of points in the polygon by counting the <Point> statements.)

Polyline Statement

```
<PolyLine
    <GenericObjectData>          # Defined at beginning of section
    <Smoothed     Yes | No>      # Smoothing on?
    <NumPoints    n>             # Number of vertices
    <Point        x y>           # Frame-relative point position
    ....                         # More points as needed
>                                # End of PolyLine statement
```

The <PolyLine> statement is used for both simple and complex lines. A simple line is represented as a <PolyLine> with <NumPoints 2>. If the <Smoothed> statement is not present, then the default is <Smoothed No>. The <NumPoints> statement is optional (FrameMaker determines the number of points in the polyline by counting the <Point> statements).

Rectangle Statement

```
<Rectangle
    <GenericObjectData>          # Defined at beginning of section
    <Smoothed     Yes | No>      # Smoothing on?
    <BRect        l t w h>       # Position, in enclosing
                                 # page/frame coordinates
>                                # End of Rectangle statement
```

<BRect> values specify the left, top, width, and height of the rectangle. The <Rectangle> statement also defines squares.

TextLine Statement

```
<TextLine
    <GenericObjectData>         # Defined at beginning of section
    <TLOrigin     x y>          # Alignment point origin
    <TLAlignment  Center|       # Alignment
                  Left   |
                  Right >
    <String       string>       # Printable ASCII in `'
    <Char         n>            # Non-ASCII character code
    <Font>                      # Embedded font change
>                               # End of TextLine statement
```

<TLOrigin> specifies the baseline (y) and the left, center, or right edge of the TextLine (x), depending on the <TLAlignment>.

Following the <TLAlignment> statement is a list of one or more <String> statements with each <String> preceded by an optional <Font> statement. <Char> statements provide codes for characters outside the printable ASCII range. You can define macros that make <Char> statements more readable. In fact, there are several predefined constants for character values: `Tab`, `HardReturn`, `EndOfPara`, `EndOfFlow`, `HardSpace`, `DiscHyphen`, `Bullet`, `Cent`, `Pound`, `Yen`, `EnDash`, `EmDash`, `Dagger`, and `DoubleDagger`.

TextRect Statement

```
<TextRect
    <GenericObjectData>         # Defined at beginning of section
    <BRect        l t w h>      # Position, in enclosing
                                # page/frame coordinates
    <TRNext       n>            # ID of next rect in flow
    <TRNextFlow   n>            # ID of next unfrozen rect
    <TRFeather     Yes | No>    # Flag
    <TRAutoConnect Yes | No>    # Flag
    <TRPrintCode   Yes | No>    # Flag
    <Frame>                     # 0 or more anchored frames
>                               # End of TextRect statement
```

<BRect> values specify the left, top, width, and height of the TextRect. <TRNext> indicates the ID of the next TextRect in the flow. If there is no next <TextRect>, you can either use a <TRNext 0> statement or omit the entire <TRNext> statement. The flags correspond directly to commands on the FrameMaker TextRects menu; we explain them in Chapter 3.

TextFlow Statement

```
<TextFlow
    <Font>                     # Sets current font values
    <TextRectID   n>           # Where the following text goes
    <Para>                     # Defines a paragraph
    <Verbose>
    <Units>
>                              # End of TextFlow statement
```

Typically, the <TextFlow> statement consists of a list of embedded <Para> statements (see the next statement description). There may be optional <Font> statements, which affect all following statements until the next <Font> statement.

When the first <TextFlow> statement is encountered, a default text flow environment is set up. The default environment consists of a Current TextRect, Current PgfProperties, and Current FontProperties. The <TextFlow> statement can override all these defaults.

After a <TextFlow> statement is encountered, all non-<TextFlow> statements will be ignored.

Most MIF generators will put all document text in one <TextFlow> statement. However, if there are subsequent <TextFlow> statements, the interpreter assumes it has the same settings (current paragraph format, current font, etc.) as the previous text flow. To divert the flow into a new, unlinked TextRect, there must be a <TextRectID> statement at the start of the new <TextFlow> statement. The <TextRectID> statement resets the current TextRect definition so that subsequent text is placed within the identified TextRect; it is only necessary if you want to reset the TextRect defaults. Again, most MIF generators do not use the <TextRectID> statement.

Para Statement

```
<Para
    <Pgf>                      # Sets current paragraph values
    <PgfTag       string>      # Forces pgf lookup in Catalog
    <ParaLine>                 # Defines line; see next subsection
    <Verbose>
    <Units>
>                              # End of Para statement
```

The <Para> statement usually consists of a list of embedded <ParaLine> statements. A paragraph by default has the same `Pgf` settings as its predecessor.

The optional <Pgf> statement, however, can override this. If a <PgfTag> statement is used, the MIF interpreter searches the document's Paragraph Catalog for a <Pgf> definition with the same tag string. If the tag string exists, then the Catalog's <Pgf> definition is used. If no corresponding Catalog <Pgf> is found, the <Pgf> definition of the previous paragraph is used, except that its tag string is reset to the tag in the <PgfTag> statement.

ParaLine Statement

```
<ParaLine
    <TextRectID   n>           # Where the following text goes
    <String       string>      # Printable ASCII in `'
    <Char         >            # A non-ASCII character code
    <Font>                     # Embedded font change
    <AFrame       n>           # ID of embedded anchored frame
    <Marker>                   # Embedded marker
>                              # End of ParaLine statement
```

A typical line statement consists of either one <String> statement or several <String> statements with interspersed <Char>, <Font>, <Marker>, and <AFrame> statements.

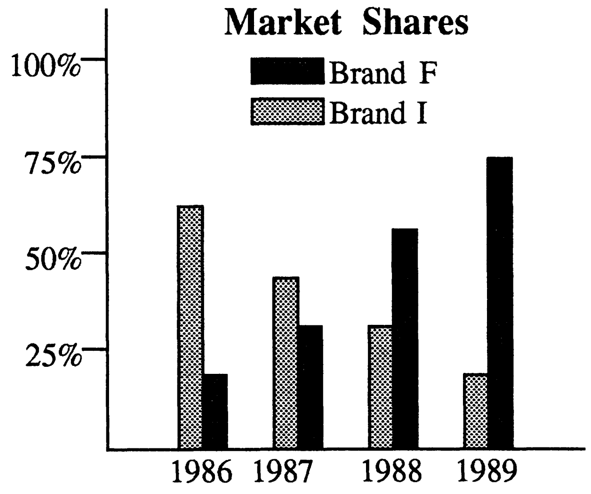
Marker Statement

```
<Marker
    <MType        n>           # Marker type number
    <MText        string>      # Marker text string
    <MStyle       Plain |      # Marker page number format
                  Start |
                  End   |
                  None  >
    <FPlain       Yes | No >   # Marker number font style
    <FBold        Yes | No >   # Marker number font style
    <FItalic      Yes | No >   # Marker number font style
    <MCurrPage    n>           # Marker's current page
>                              # End of marker statement
```

Examples

Bar Chart Example

This example shows a bar chart and the MIF file that describes it. To draw the bar chart, you can open or import the MIF file in FrameMaker. Normally, you would create an anchored frame in a document, select the frame, and then import this file. (The MIF file is on the FrameMaker installation tape in `docs/BarChart.mif`.) Here is an anchored frame into which `BarChart.mif` has been imported:



Bar Chart Generated by the Following MIF Statements

```
<MIFFile 1.00> # Generated by SomeChartPack 1.4
include(mif_read.m4)
<Comment  Draw the title>
  <TextLine <GroupID 1>
   <Font <FFamily `Times'> <FSize 14> <FPlain Yes> <FBold Yes >
         <FDX 0> <FDY 0> <FDAX 0> <FNoAdvance No >
   >
<Comment  End of Font>
   <TLOrigin  1.85" 0.21"> <TLAlignment Center > <String `Market Shares'>
  >
<Comment  End of TextLine>
<Comment  Draw the legend>
  <Rectangle <GroupID 1>
   <BRect  1.36" 0.33" 0.38" 0.13">
   <Fill 1>
 >
<Comment  End of Rectangle>
  <Rectangle <GroupID 1>
   <Fill 4>
   <BRect  1.36" 0.54" 0.38" 0.13">
  >
```

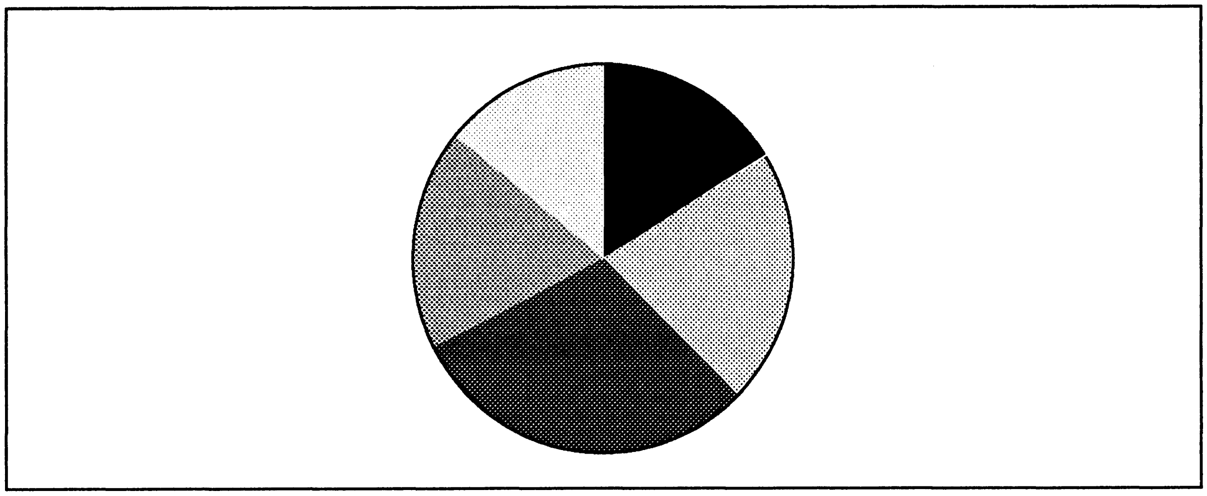
```
<Comment  End of Rectangle>
  <TextLine <GroupID 1>
   <Font <FSize 12> <FPlain Yes > >
<Comment  End of Font>
   <TLOrigin  2.04" 0.46"> <TLAlignment Center > <String `Brand F'>
  >
<Comment  End of TextLine>
  <TextLine <GroupID 1>
   <TLOrigin  2.01" 0.67"> <TLAlignment Center > <String `Brand I'>
  >
<Comment  End of TextLine>
<Comment  Set up default pen and pen width>
  <Pen 0>
  <PenWidth `0.500 '>
<Comment  Draw axis>
  <PolyLine <GroupID 1>
   <NumPoints 3> <Point  0.60" 0.08"> <Point  0.60" 2.35">
                 <Point  3.10" 2.35">
  >
<Comment  End of PolyLine>
<Comment  Draw tickmarks on y axis>
  <PolyLine <GroupID 1>
   <NumPoints 2> <Point  0.60" 1.83"> <Point  0.47" 1.83">
  >
<Comment  End of PolyLine>
  <PolyLine <GroupID 1>
   <NumPoints 2> <Point  0.60" 1.33"> <Point  0.47" 1.33">
  >
<Comment  End of PolyLine>
  <PolyLine <GroupID 1>
   <NumPoints 2> <Point  0.60" 0.83"> <Point  0.47" 0.83">
  >
<Comment  End of PolyLine>
  <PolyLine <GroupID 1>
   <NumPoints 2> <Point  0.60" 0.33"> <Point  0.47" 0.33">
  >
<Comment  End of PolyLine>
<Comment  Label the axes>
  <TextLine <GroupID 1>
   <Inverted No >
   <TLOrigin  1.08" 2.51"> <TLAlignment Center > <String `1986'>
  >
<Comment  End of TextLine>
  <TextLine <GroupID 1>
   <TLOrigin  1.52" 2.51"> <TLAlignment Center > <String `1987'>
  >
<Comment  End of TextLine>
  <TextLine <GroupID 1>
   <TLOrigin  2.08" 2.51"> <TLAlignment Center > <String `1988'>
  >
<Comment  End of TextLine>
  <TextLine <GroupID 1>
   <TLOrigin  2.58" 2.51"> <TLAlignment Center > <String `1989'>
  >
```

```
<Comment  End of TextLine>
  <TextLine <GroupID 1>
   <TLOrigin  0.46" 1.92"> <TLAlignment Right >  <String `25%'>
  >
<Comment  End of TextLine>
  <TextLine <GroupID 1>
   <TLOrigin  0.46" 1.42"> <TLAlignment Right >  <String `50%'>
  >
<Comment  End of TextLine>
  <TextLine <GroupID 1>
   <TLOrigin  0.46" 0.92"> <TLAlignment Right >  <String `75%'>
  >
<Comment  End of TextLine>
  <TextLine <GroupID 1>
   <TLOrigin  0.46" 0.42"> <TLAlignment Right >  <String `100%'>
  >
<Comment  End of TextLine>
<Comment  Draw the bars>
  <Rectangle <GroupID 1>
   <Fill 4>
   <BRect  0.97" 1.10" 0.13" 1.25">
  >
<Comment  End of Rectangle>
  <Rectangle <GroupID 1>
   <BRect  1.47" 1.47" 0.13" 0.88">
  >
<Comment  End of Rectangle>
  <Rectangle <GroupID 1>
   <BRect  1.97" 1.72" 0.13" 0.63">
  >
<Comment  End of Rectangle>
  <Rectangle <GroupID 1>
   <BRect  2.47" 1.97" 0.13" 0.38">
  >
<Comment  End of Rectangle>
  <Rectangle <GroupID 1>
   <Fill 1>
   <BRect  1.10" 1.97" 0.13" 0.38">
  >
<Comment  End of Rectangle>
  <Rectangle <GroupID 1>
   <BRect  1.60" 1.72" 0.13" 0.63">
  >
<Comment  End of Rectangle>
  <Rectangle <GroupID 1>
   <BRect  2.10" 1.22" 0.13" 1.13">
  >
<Comment  End of Rectangle>
  <Rectangle <GroupID 1>
   <BRect  2.60" 0.85" 0.13" 1.50">
  >
<Comment  End of Rectangle>
<Comment  Put it all in a group so it is easy for the user to
move/scale after importing>
```

```
  <Group
   <ID 1>
  >
<Comment  End of Group>
<Comment  End of MIFFile>
```

Pie Chart Example

This example shows a simple pie chart. Note that the chart is positioned in the upper right corner of the coordinate space rather than centered on some assumed page size. This is a good idea since the image may be imported into any size page or frame; if it were centered on an 8.5 x 11" page and imported into a 3x3 frame, the image would lie outside the bounds of the frame, and therefore be invisible.



Pie Chart Generated by the Following MIF Statements

```
<MIFFile 1.00> # Generated by xyzgrapher 3.5
include(mif_read.m4)
<Comment  All dimensions given in point units (1pt = 1/72")>
<Units Upt >
  <Pen 0>
  <PenWidth .5>
  <Fill 0>
  <Arc <Inverted No > <GroupID 1>
   <ArcRect  12 11 144 144 > <ArcTheta 0> <ArcDTheta 58>
  >
<Comment  End of Arc>
  <Arc <Fill 5> <GroupID 1>
    <ArcRect  12 11 144 144 > <ArcTheta 59> <ArcDTheta 77>
  >
<Comment  End of Arc>
  <Arc <Fill 2> <GroupID 1>
    <ArcRect  12 11 144 144 > <ArcTheta 135> <ArcDTheta 108>
  >
```

```
<Comment  End of Arc>
  <Arc <Fill 4> <GroupID 1>
    <ArcRect  12 11 144 144 > <ArcTheta 243> <ArcDTheta 66>
  >
<Comment  End of Arc>
  <Arc <Fill 6> <GroupID 1>
    <ArcRect  12 11 144 144 > <ArcTheta 309> <ArcDTheta 51>
  >
<Comment  End of Arc>
  <Group <ID 1> >
<Comment  End of Group>
<Comment  End of MIFFile>
```

MIF Messages

When FrameMaker reads a MIF file, it may detect unexpected character sequences (sometimes referred to as errors). It responds by printing messages on the message window. Even if it finds an error, FrameMaker will continue to process the MIF file and do its best to read in as much of the document as possible.

This section lists the MIF messages produced by FrameMaker along with their meanings. Words in *this font* indicate variable words in a message. The general form of all MIF message lines is

`MIF: Line` *`LineNum`*`:` *`Message`*

The *Message* portion consists of one of the messages listed in the following *Syntax Messages* section. The *LineNum* is approximate since it is the absolute line number in the file after all macros in the file have been expanded.

Syntax Messages

`Char out of range (`*`CharValue`*`).`

A character in a <Char> statement , or a character expressed using \x in a string, is out of range.

`Missing or unexpected token or number.`

An unknown statement or data name was found, or a number is missing on the indicated line.

`Missing dimension.`

A necessary dimension value was not found in a statement.

`Nesting Too Deep, Skipping statement. Done Skipping (`*`LineNum`*`).`

Too many nested frames. The maximum nesting depth is 10.

`ParameterName out of range (BadValue).`

A statement's data value was too large or too small.

`--- Skipping these chars: ---------- Done skipping.`

A statement was found in an invalid context (for example, a <DParity> statement outside of a <Document> statement).

`String too long, overflow ignored.`

The maximum string length is 255 characters.

`Unbalanced Right Bracket.`

A right bracket character (>) was found that has no corresponding left bracket character.

`Unexpected Right Bracket.`

A right bracket character (>) was found where a data value was expected, or was found outside of a statement

Object Messages

`Cannot find TextRect id TextRectIDNumber`

The ID specified in a `TextRectID` statement has no corresponding defined TextRect object.

`Could not connect to TRNextFLow id TextRectIDNumber`

The ID specified in a <TRNextFlow> statement has no corresponding defined TextRect object.

`Could not connect to TRNext id =TextRectIDNumber`

The ID specified in a <TRNext> statement has no corresponding defined TextRect object.

`Could not find anchored frame FrameIDNumber.`

The frame ID specified in an <AFrame> statement has no corresponding defined frame.

`Empty Group (ID=GroupIDNumber)`

No defined objects have a GroupID that matches the ID in a <Group> statement.

`Family/size/style combination not available.`

The requested combination of font family, size, and style is not available.

`Insufficient Memory!`

FrameMaker was unable to allocate memory for one of its work buffers. It would be best to free some swap space and perhaps restart FrameMaker if this happens.

`Object ignored. Must come before textflow statements.`

All object statements must come before the first <TextFlow> statement in a MIF file.

`Unable to start new object`

FrameMaker was unable to allocate memory for a new object. If this happens, free some swap space and perhaps restart FrameMaker.

`Unable to store marker`

The marker table is full, probably due to FrameMaker running out of swap space. If this happens, free some swap space and perhaps restart FrameMaker.

`Unavailable Font Size (`*`RequestedSize`*`).`

The requested font size is not available.

`Unknown Font Family.`

The requested font family is not available.

`WARNING: Circular text flow was found and cut.`

The MIF file defined a set of linked TextRects resulting in a circular text flow (the last TextRect in the flow is linked to the first or to one in the middle). FrameMaker attempted to resolve the problem by breaking a link.

`WARNING: Circular text flow. Document should not be used.`

The MIF file defined a set of linked TextRects resulting in a circular text flow (the last TextRect in the flow is linked to the first or to one in the middle). FrameMaker was unable to resolve the problem. A FrameMaker file will open, but it should not be used.

Verbose Debugging Messages

`Could not merge onto page %0.`

A statement is trying to merge graphics onto the page specified in the message, but the page isn't present in the document. The objects are merged onto the current page instead.

`Following TabStops will determine actual number of tabs`

The <PgfNumTabs> statement is present in MIF for use by other programs that read MIF files. When FrameMaker reads a MIF file, it determines the number of tabs stops in a paragraph by counting the number of <TabStop> statements.

`Processing opcode#` *`Token`*

Shows what statement is being processed.

`PageSize ignored - use DPageSize earlier`

Once a document's page size has been defined, the page size must be consistent across all pages in the document.

`MCurrPage ignored when reading.`

When reading a MIF file, FrameMaker displays this message if it encounters an <MCurrPage> statement within a <Marker> statement. Page numbers for markers are ignored when reading MIF files (the page number is determined by the page on which the marker occurs).

`NumPoints will be determined by Point statements.`

The <NumPoints> statement is present in MIF for use by other programs that read MIF files. When FrameMaker reads a MIF file, it determines the number of points in a polygon by counting the number of <Point> statements.

`PenWidth mapped to` *`NewPenWidth`*`.`

FrameMaker maps all requested pen width values to the set available within FrameMaker.

`Unexpected characters.`

When reading a MIF file, FrameMaker displays this message if it expects a statement identifier, but instead encounters characters that do not make up a known statement identifier. FrameMaker skips the unexpected characters, but displays them to help you debug the problem.

Keyboard Commands

This appendix lists the following FrameMaker keyboard commands:

- Border pattern commands
- Deletion commands
- Document menu commands
- Drawing tool commands
- Edit menu and related commands
- Fill pattern commands
- Font commands
- Format menu commands
- Guides menu commands
- Highlighting commands
- Insertion point commands
- Kerning/object moving commands
- Main window commands
- Miscellaneous
- Mouse button modifiers
- Movement constraints
- Page menu commands
- Paragraph commands
- Search and replace commands
- Spelling checker commands
- TextRect menu commands
- Tools window commands
- Width commands

Each category appears in a separate table. Each table (except for the Mouse Button Modifiers table) has the same format.

The first column in the table shows the function codes (in hexadecimal notation) for each command. We show these values to allow you to change FrameMaker's built-in key-to-command mappings (you can change which keys perform specific commands). To configure your keyboard, you need these function codes (see Appendix D, *Customizing FrameMaker,* for more information).

The second column shows the standard key sequences you use to choose each command. The third column describes the command.

You'll frequently use several keys in keyboard commands. We use the following abbreviations for these keys:

This abbreviation:	**Means:**
Esc	The key labeled *Esc*
Ctrl	The key labeled *Ctrl*
Alt	The keys labeled *Alt.*
Shift	The key labeled *Shift*
[F1]	The function key labeled *F1*; function keys are always enclosed in square brackets
space	The spacebar
minus or hyphen	The key labeled with a hyphen (-)
period	The key labeled with a period (.)
zero	The key labeled with the number *0*
one	The key labeled with the number *1*
l	The unshifted *L* key

Most keyboard commands involve pressing two or three keys, either together or in succession. For example:

When we say:	**We mean:**
Esc d i	Press and release three keys in succession: the *Esc* key, the unshifted *D* key, and the unshifted *I* key.
Alt-d	Press and hold down the *Alt* key and then press the unshifted *D* key.
Shift-[F1]	Press and hold down the *Shift* key and then press the function key labeled *F1.*
Ctrl-q Ctrl-space	Press and hold down the Ctrl key and then press the letter *q* and the space bar.

Border Pattern Commands

Use these commands to modify the border pattern of all selected objects. The arrow pointer must be in the document window.

Function Code	Key Sequence	Description
\x3A3	Esc zero p	Set to first border pattern (black)
\x3A4	Esc + p	Increment border pattern
\x3A5	Esc minus p	Decrement border pattern

Deletion Commands

Use these commands to delete characters, starting at the insertion point and going forward or backward as indicated. For example, Alt-d deletes all characters between the insertion point and the end of the word to the right of the insertion point, including the trailing space. You can use the Delete key to delete selected objects as a shortcut for the Cut command.

Function Code	Key Sequence	Description
\x112	Delete, Backspace	Delete back one character
\x113	Ctrl-d	Delete forward one character
\x114	Alt-Delete, Alt-Backspace	Delete back to start of word
\x115	Alt-d	Delete forward to end of word
\x116	Ctrl-k	Delete forward to end of line
\x117	Alt-k	Delete forward to end of sentence
\x118	Delete, Backspace	Delete selected text/selected objects
\x119	Ctrl-u	Delete backward from insertion point to start of line

Document Menu Commands

Use these commands as shortcuts for using the Document menu commands. The arrow pointer must be in the document window.

Function Code	Key Sequence	Description
\x310	Ctrl-x Ctrl-w, Ctrl-x Ctrl-s, Esc d s	Save
\x311	Esc d p	Print
\x312	Esc d i, Esc I	Import
\x251	Esc d c, Alt-c	Capture
\x252	Esc d r, Ctrl-]	Record Keys
\x254	Esc d k	Keyboard
\x378	Esc d q	Quit

Drawing Tool Commands

Use these commands as shortcuts for selecting drawing tools in the Tools window. The arrow pointer must be in the document window.

Function Code	Key Sequence	Description
\x38C	Esc one l	Line
\x38D	Esc one r	Rectangle
\x38E	Esc one s	Square
\x38F	Esc one p g	Polygon
\x390	Esc one p l	Polyline
\x391	Esc one a	Arc
\x392	Esc one c	Circle
\x393	Esc one e	Ellipse
\x394	Esc one t l	TextLine
\x395	Esc one t r	TextRect
\x396	Esc one f	Freehand
\x397	Esc one w	Arrow
\x398	Esc one m	Frame
\x399	Esc one one	Select last-used tool
\x3A9	Esc one k	Toggles Keep Tool setting

Edit Menu and Related Commands

Use these commands as shortcuts for using the Edit menu commands as well as other editing functions. The arrow pointer must be in the document window.

Function Code	Key Sequence	Description
\x320	Esc e u	Undo
\x321	Ctrl-w, Esc e x, Esc x	Cut
\x322	Alt-w, Esc e c	Copy
\x323	Ctrl-y, Esc e p, Esc v	Paste
\x324	Alt-y, Esc e t, Esc i	X Paste
\x325	Esc e a	Anchored Frame
\x326	Ctrl-s, Esc e s	Search
\x327	Esc e m	Markers
\x32A	Esc e f	Copy Font
\x32B	Esc e g	Copy Pgf Format
\x32C	Esc m k	Insert new marker. Like New Marker button.

Fill Pattern Commands

Use these commands to modify the fill pattern of all selected objects in a window. The arrow pointer must be in the document window.

Function Code	Key Sequence	Description
\x3A6	Esc zero f	Set to first fill pattern (black)
\x3A7	Esc + f	Increment fill pattern
\x3A8	Esc minus f	Decrement fill pattern

Font Commands

Use these commands as shortcuts for using the Fonts command to set the font characteristics of all selected text and selected TextLines in the window. The arrow pointer must be in the document window.

Function Code	Key Sequence	Description
\x230	Esc c b, [F2]	Set characters to bold
\x231	Esc c i, [F3]	Set characters to italic
\x232	Esc c u, [F4]	Set characters to underline
\x23E	Esc c s, [F5]	Set characters to strike through
\x233	Esc c p, [F1]	Set characters to plain
\x234	Esc c +	Set characters to superscript
\x235	Esc c minus	Set characters to subscript
\x236	Esc c =	Set characters to normal
\x237	Esc c >, Esc + n	Scroll to next font family
\x238	Esc c <, Esc minus n	Scroll to previous font family
\x239	Esc c], Esc + s	Increment text size
\x23A	Esc c [, Esc minus s	Decrement text size
\x23B	Esc c Left Arrow	Squeeze spacing 1 point
\x23C	Esc c Right Arrow	Spread spacing 1 point
\x23D	Esc c c	Repeat last font-related command

Format Menu Commands

Use these commands as shortcuts for using the Format menu commands. The arrow pointer must be in the document window.

Function Code	Key Sequence	Description
\x330	Esc f t, Esc T	Tabs
\x331	Esc f f, Esc F	Fonts
\x332	Esc f p, Esc P	Paragraphs
\x333	Esc f c, Esc C	Catalog
\x334	Esc f a	Auto Hyphenation
\x335	Esc f n	Number of Columns
\x336	Esc f h	Headers & Footers
\x337	Esc f z	Freeze Pagination
\x338	Esc f r	Repaginate

Guides Menu Commands

Use these commands as shortcuts for using the Guides menu commands. The arrow pointer must be in the document window.

Function Code	Key Sequence	Description
\x350	Esc g g	Grid
\x351	Esc g r	Rulers
\x355	Esc g s	Snap
\x356	Esc g b	Borders
\x357	Esc g t	Text Symbols
\x358	Esc g u	Units

Highlighting Commands

Use the first five commands to select text to the right of the insertion point. If text is already selected, these commands extend the selection. The arrow pointer must be in the document window containing the insertion point.

Function Code	Key Sequence	Description
\x247	Esc h c	Select current character, then next one
\x248	Esc h w	Select current word, then next one
\x249	Esc h l	Select current line, then next one
\x24A	Esc h s	Select current sentence, then next one
\x24B	Esc h p	Select current paragraph, then next one
\x24C	Esc h b	Shift selecting left 1 character
\x24D	Esc h f	Shift selecting right 1 character
\x24E	Esc h zero	Clear selecting

Insertion Point Commands

Use these commands, which are based on EMACS commands, to move the insertion point. The arrow pointer must be in the document window containing the insertion point.

Function Code	Key Sequence	Description
\x100	Home	Top of TextRect
\x101	Up Arrow, Ctrl-p	Up 1 line
\x102	Down Arrow, Ctrl-n	Down 1 line
\x103	Right Arrow, Ctrl-f	Right 1 character
\x104	Left Arrow, Ctrl-b	Left 1 character
\x105	Ctrl-a	Beginning of line
\x106	Ctrl-e, End	End of line
\x107	Alt-b	Beginning of word
\x108	Alt-f	End of word
\x109	Alt-a	Beginning of sentence
\x10A	Alt-e	End of sentence
\x10B	Alt-[	Beginning of paragraph
\x10C	Alt-]	End of paragraph
\x10E	Shift-Home	Bottom of TextRect
\x10F	Alt-{	Beginning of flow
\x110	Alt-}	End of flow

Kerning/Object Moving Commands

Use these commands to create mathematical formulas in text and to move graphics with precise control. The arrow pointer must be in the document window containing selected text or one selected object/group.

Function Code	Key Sequence	Description
\x11A	Alt-Up Arrow, Ctrl-Up Arrow	Move 1 point up
\x11B	Alt-Down Arrow, Ctrl-Down Arrow	Move 1 point down
\x11C	Alt-Left Arrow, Ctrl-Left Arrow	Move 1 point left
\x11D	Alt-Right Arrow, Ctrl-Right Arrow	Move 1 point right
\x121	Shift-Alt-Up Arrow	Move 6 points up
\x122	Shift-Alt-Down Arrow	Move 6 points down
\x123	Shift-Alt-Left Arrow	Move 6 points left
\x124	Shift-Alt-Right Arrow	Move 6 points right
\x11E	Alt-Home, Ctrl-Home	Remove all kerning (text only)
\x11F	F10	Make characters overlap (text only)
\x120	F11	Make characters stack (text only)

Main Window Commands

Use these commands as shortcuts for using the commands in the main FrameMaker window. Most of these commands are also in the Document menu. The arrow pointer must be in the document window.

Function Code	Key Sequence	Description
\x300	Esc m n, Esc d n	New
\x301	Ctrl-x Ctrl-f, Esc m o, Esc d o	Open
\x302	Esc m t, Esc d t	Tools
\x303	Esc m h	Help
\x304	Esc m i, Esc d I	Info

Miscellaneous

Use back tab and first tab in dialog boxes (see *Using Dialog Boxes* in Chapter 2).

Use the open line command, which is an EMACS-like option, as a shortcut to pressing the Enter key and then pressing the Left Arrow key.

Use Ctrl-q to enter special characters, as explained in Appendix B, *Character Set.*

Use Ctrl-t to switch the two characters that surround the insertion point. There must be an active insertion point. (Use this key sequence to correct misspellings such as *piont.* Put the insertion point between the *i* and the *o* and press Ctrl-t.)

Use the nonbreaking space, discretionary hyphen, and suppress hyphenation while editing text (see the Auto Hyphenation command in Chapter 3).

Use Ctrl-c to interrupt the Import and Change All commands and as a shortcut for clicking Cancel in a dialog box.

Function Code	**Key Sequence**	**Description**
\x220	Shift-Tab	Back tab (as in dialog boxes)
\x222	Alt-Tab	First tab (as in dialog boxes)
	Ctrl-o	Open line (same as Enter Left Arrow)
\x223	Ctrl-q	Insert a special character (see Appendix B)
\x224	Ctrl-t	Transpose characters
\x226	Alt-space	Nonbreaking space (not word delimiter)
\x227	Alt-hyphen	Discretionary hyphen
\x22E	Ctrl-hyphen	Suppress hyphenation
\x250	Ctrl-c	Abort process like Import or Change All

Mouse Button Modifiers

Use these keys to change the effect of pressing the mouse buttons. The arrow pointer must be in the document window. To use one of the modified mouse button commands, press and hold down the indicated key and press the indicated mouse button. There are no function codes for the mouse button modifiers.

Key Sequence	Mouse Button	Description
Ctrl	Right	Force an object to move
Ctrl	Left	Force an object to stretch or shrink
Shift	Right	Use quick-copy feature for graphics
Shift	Left	Force a selection border to be drawn
Shift-Ctrl	Right	Add a point to an object
Shift-Ctrl	Left	Delete a point from an object
Ctrl	Middle	Use quick-copy feature for text
Shift	Middle	Extend text selection

Movement Constraints

Use these commands as shortcuts for clicking the H. Only, V. Only, and Unconstrained settings in the Tools window. The arrow pointer must be in the document window.

Function Code	Key Sequence	Description
\x352	Esc g h	H. Only
\x353	Esc g v	V. Only
\x354	Esc g n	Unconstrained

Page Menu Commands

Use these commands as shortcuts for using the Page menu commands. The arrow pointer must be in the document window.

Function Code	Key Sequence	Description
\x360	Alt-v, Esc p p, [F6]	Previous
\x361	Ctrl-v, Esc p n, [F7]	Next
\x362	Esc p f	First
\x363	Esc p l	Last
\x364	Esc p m	Master
\x365	Esc p s	Scroll
\x366	Ctrl-g, Esc p g	Go To
\x367	Esc p a	Add
\x368	Esc p d	Delete

Paragraph Commands

Use these commands as shortcuts for using the Paragraphs command. Esc j l, Esc j c, and Esc j r can also be used when objects are selected, as shortcuts for the Align command's Left Sides, L/R Centers, and Right Sides settings. The arrow pointer must be in the document window.

Function Code	Key Sequence	Description
\x240	Esc j +, Esc + l	Increment line spacing
\x241	Esc j minus, Esc minus l	Decrement line spacing
\x242	Esc j c	Center paragraph
\x243	Esc j l	Left justify paragraph
\x244	Esc j r	Right justify paragraph
\x245	Esc j f	Fully justify paragraph
\x246	Esc j j	Repeat last paragraph-related command

TextRect Menu Commands

Use these commands as shortcuts for using the TextRects menu commands. The arrow pointer must be in the document window.

\x340	Esc t c	Connect TextRects
\x341	Esc t h	Disconnect Head
\x342	Esc t t	Disconnect Tail
\x343	Esc t s	Split TextRect
\x344	Esc t a	Auto Connect
\x345	Esc t f	Feathering
\x346	Esc t p	Printer Code

Tools Window Commands

Use these commands as a shortcut for selecting commands in the Tools window. (Note that *o* stands for *object.*) The arrow pointer must be in the document window.

Function Code	**Key Sequence**	**Description**
\x380	Esc o f	Front
\x381	Esc o b	Back
\x382	Esc o g	Group
\x383	Esc o u g	Ungroup
\x384	Esc o r	Reshape
\x385	Esc o s	Smooth
\x386	Esc o u s	Unsmooth
\x387	Esc o i	Invert
\x388	Esc o z	Scale
\x389	Esc o a	Align
\x38A	Esc o d	Distribute
\x38B	Esc o n	Set # Sides

Search and Replace Commands

Use these commands as shortcuts for clicking buttons in the Search window. The arrow pointer must be in the document window. See the Search command in Chapter 3.

Function Code	Key Sequence	Description
\x228	Alt-Ctrl-r, Esc s p	Search backward
\x229	Alt-Ctrl-s, Esc s n	Search forward
\x22A	Ctrl-%, Esc r o	Change
\x22B	Esc r g	Change All
\x22C	Esc r a	Change & Search Again
\x22D	Esc s s	Display Set Search String dialog box

Spelling Checker Commands

Use these commands as shortcuts to using the Spelling Checker functions. The arrow pointer must be in the document window.

Function Code	Key Sequence	Description
\x32D	Esc e l	Display the Spelling Checker window
\x3C0	Esc l s	Start checking current selection
\x3C1	Esc l e	Start checking to end of document
\x3CB	Esc l p	Start checking current page
\x3C2	Esc l c w	Correct word
\x3C3	Esc l a p	Add word to personal dictionary (Learn)
\x3C4	Esc l a d	Add word to document dictionary
\x3C5	Esc l a c	Add word and correction to auto corrections
\x3C6	Esc l x p	Delete word from personal dictionary
\x3C7	Esc l x d	Delete word from document dictionary
\x3C8	Esc l c a	Clear auto corrections
\x3C9	Esc l c d	Change dictionaries
\x3CA	Esc l r	Clear no-need-to-recheck flags

Width Commands

Use these commands to modify the border and/or line width of all selected objects in the window. The arrow pointer must be in the document window.

Function Code	Key Sequence	Description
\x3A0	Esc zero w	Set to thinnest width
\x3A1	Esc + w	Increment line width
\x3A2	Esc minus w	Decrement line width

Character Set

This appendix lists the character set used for FrameMaker documents and shows how to type each character in the set.

The assignment of specific characters to specific code values is referred to as *character encoding*. FrameMaker uses two kinds of encoding: standard and symbol. *Symbol* encoding is used for FrameMaker's Symbol font, and *standard* encoding is used for all other fonts. The table on the following pages shows both types of encoding.

The first column in the table shows the character code value, in hexadecimal notation. The main purpose for showing these values is for use with the Maker Markup Language (MML) and the Maker Interchange Format (MIF) (see Chapters 6 and 7 for more information).

The second column shows the key sequence used to type each character. The third and fourth columns show the character names and shapes for standard encoding, and the final two columns show the characters used in symbol encoding.

We use the following abbreviations when indicating a key sequence:

This abbreviation:	**Means:**
Esc	The key labeled *Esc*
Ctrl	The key labeled *Ctrl*
Alt	The keys labeled *Alt*
Shift	The key labeled *Shift*
space	The spacebar
comma	The key labeled with a comma (,)
minus or hyphen	The key labeled with a hyphen (-)
period	The key labeled with a period (.)
zero	The key labeled with the number *0*
one	The key labeled with the number *1*
l	The unshifted *L* key

Most key sequences involve pressing two or more keys, either together or in succession. For example:

When we say:	**We mean:**
Ctrl-hyphen	Press and hold down the Ctrl key and then press the hyphen key.
Ctrl-q Ctrl-space	Press and hold down the Ctrl key and then press the letter *q* and the space bar.
Ctrl-q b	Press and hold down the Ctrl key and then press the letter *q*. Then release both keys and press the letter *b*.
Esc d i	Press and release three keys in succession: the *Esc* key, the unshifted *D* key, and the unshifted *I* key.
Alt-space	Press and hold down the Alt key and then press the space bar.

Notes:

- You can use FrameMaker's macro capability to assign commonly typed special characters to more simple key sequences (see the Record Keys command in Chapter 3 for more information).
- The on-line document `HelpSymbols.doc` is a quick reference to the information in this appendix. To see this on-line document, choose the Open command, type **`HelpSymbols.doc`** in the Open File Named edit box, and click OK.
- The characters in the range below code \x20 are *control codes*. Rather than causing characters to be printed, these codes affect how the surrounding text is formatted. These characters can been seen on the screen when viewing a document window if text symbols are turned on (see the Text Symbols command in Chapter 3).
- Many of the character values are reserved for future use. Some of the reserved values cause characters to appear on the screen in a document window. These values, however, may cause different characters to appear when printed.

FrameMaker Character Set

Hex		Standard Set		Symbol Set	
Code	Key Sequence	Char Name	Graphic	Char Name	Graphic
\x04	`Alt-space`	hard space			
\x05	`Ctrl-hyphen`	nohyphen			
\x06	`Alt-hyphen`	discretionaryhyphen			
\x07	`<not typeable>`	softhyphen			
\x09	`Tab`	tab			
\x0a	`Enter`	endofparagraph			
\x0d	`Alt-Enter`	hardreturn			
\x20	`space`	space		space	
\x21	`!`	exclam	!	exclam	!
\x22	`"`	quotedbl	"	universal	∀
\x23	`#`	numbersign	#	numbersign	#
\x24	`$`	dollar	$	existential	∃
\x25	`%`	percent	%	percent	%
\x26	`&`	ampersand	&	ampersand	&
\x27	`Ctrl-'`	quotesingle	'	suchthat	∍
\x28	`(`	parenleft	(	parenleft	(
\x29	`)`	parenright	)	parenright	)
\x2a	`*`	asterisk	*	asteriskmath	∗
\x2b	`+`	plus	+	plus	+
\x2c	`comma`	comma	,	comma	,
\x2d	`-`	hyphen	-	minus	−
\x2e	`.`	period	.	period	.
\x2f	`/`	slash	/	slash	/
\x30	`0`	zero	0	zero	0
\x31	`1`	one	1	one	1
\x32	`2`	two	2	two	2
\x33	`3`	three	3	three	3
\x34	`4`	four	4	four	4
\x35	`5`	five	5	five	5
\x36	`6`	six	6	six	6
\x37	`7`	seven	7	seven	7
\x38	`8`	eight	8	eight	8
\x39	`9`	nine	9	nine	9
\x3a	`:`	colon	:	colon	:
\x3b	`;`	semicolon	;	semicolon	;

FrameMaker Character Set - *(continued)*

Hex		Standard Set		Symbol Set	
Code	Key Sequence	Char Name	Graphic	Char Name	Graphic
\x3c	<	less	<	less	<
\x3d	=	equal	=	greater	=
\x3e	>	greater	>	greater	>
\x3f	?	question	?	question	?
\x40	@	at	@	congruent	≅
\x41	A	A	A	Alpha	Α
\x42	B	B	B	Beta	Β
\x43	C	C	C	Chi	Χ
\x44	D	D	D	Delta	Δ
\x45	E	E	E	Epsilon	Ε
\x46	F	F	F	Phi	Φ
\x47	G	G	G	Gamma	Γ
\x48	H	H	H	Eta	Η
\x49	I	I	I	Iota	Ι
\x4a	J	J	J	theta1	ϑ
\x4b	K	K	K	Kappa	Κ
\x4c	L	L	L	Lambda	Λ
\x4d	M	M	M	Mu	Μ
\x4e	N	N	N	Nu	Ν
\x4f	O	O	O	Omicron	Ο
\x50	P	P	P	Pi	Π
\x51	Q	Q	Q	Theta	Θ
\x52	R	R	R	Rho	Ρ
\x53	S	S	S	Sigma	Σ
\x54	T	T	T	Tau	Τ
\x55	U	U	U	Upsilon	Υ
\x56	V	V	V	sigma1	ς
\x57	W	W	W	Omega	Ω
\x58	X	X	X	Xi	Ξ
\x59	Y	Y	Y	Psi	Ψ
\x5a	Z	Z	Z	Zeta	Ζ
\x5b	[	bracketleft	[	bracketleft	[
\x5c	\	backslash	\	therefore	∴
\x5d	]	bracketright	]	bracketright	]
\x5e	^	asciicircum	^	perpendicular	⊥

FrameMaker Character Set - (continued)

Hex Code	Key Sequence	Standard Set		Symbol Set	
		Char Name	Graphic	Char Name	Graphic
\x5f	_	underscore	_	underscore	_
\x60	`Ctrl-``	grave	`	radicalex	‾
\x61	`a`	a	a	alpha	α
\x62	`b`	b	b	beta	β
\x63	`c`	c	c	chi	χ
\x64	`d`	d	d	delta	δ
\x65	`e`	e	e	epsilon	ε
\x66	`f`	f	f	phi	ϕ
\x67	`g`	g	g	gamma	γ
\x68	`h`	h	h	eta	η
\x69	`i`	i	i	iota	ι
\x6a	`j`	j	j	phi1	φ
\x6b	`k`	k	k	kappa	κ
\x6c	`l`	l	l	lambda	λ
\x6d	`m`	m	m	mu	μ
\x6e	`n`	n	n	nu	ν
\x6f	`o`	o	o	omicron	ο
\x70	`p`	p	p	pi	π
\x71	`q`	q	q	theta	θ
\x72	`r`	r	r	rho	ρ
\x73	`s`	s	s	sigma	σ
\x74	`t`	t	t	tau	τ
\x75	`u`	u	u	upsilon	υ
\x76	`v`	v	v	omega1	ϖ
\x77	`w`	w	w	omega	ω
\x78	`x`	x	x	xi	ξ
\x79	`y`	y	y	psi	ψ
\x7a	`z`	z	z	zeta	ζ
\x7b	{	braceleft	{	braceleft	{
\x7c	|	bar	|	bar	|
\x7d	}	braceright	}	braceright	}
\x7e	~	asciitilde	~	similar	~
\x7f		*Reserved*		*Reserved*	
\x80	`Esc % A`	Adieresis	Ä	*Reserved*	
\x81	`Esc * A`	Aring	Å	*Reserved*	

FrameMaker Character Set - *(continued)*

Hex		Standard Set		Symbol Set	
Code	Key Sequence	Char Name	Graphic	Char Name	Graphic
\x82	`Esc comma C, Shift-F9`	Ccedilla	Ç	*Reserved*	
\x83	Esc ' E	Eacute	É	*Reserved*	
\x84	Esc ~ N	Ntilde	Ñ	*Reserved*	
\x85	Esc % O	Odieresis	Ö	*Reserved*	
\x86	Esc % U	Udieresis	Ü	*Reserved*	
\x87	Esc ' a	aacute	á	*Reserved*	
\x88	Esc ` a	agrave	à	*Reserved*	
\x89	Esc ^ a	acircumflex	â	*Reserved*	
\x8a	Esc % a	adieresis	ä	*Reserved*	
\x8b	Esc ~ a	atilde	ã	*Reserved*	
\x8c	Esc * a	aring	å	*Reserved*	
\x8d	Esc comma c, F9	ccedilla	ç	*Reserved*	
\x8e	Esc ' e	eacute	é	*Reserved*	
\x8f	Esc ` e	egrave	è	*Reserved*	
\x90	Esc ^ e	ecircumflex	ê	*Reserved*	
\x91	Esc % e	edieresis	ë	*Reserved*	
\x92	Esc ' i	iacute	í	*Reserved*	
\x93	Esc ` i	igrave	ì	*Reserved*	
\x94	Esc ^ i	icircumflex	î	*Reserved*	
\x95	Esc % i	idieresis	ï	*Reserved*	
\x96	Esc ~ n	ntilde	ñ	*Reserved*	
\x97	Esc ' o	oacute	ó	*Reserved*	
\x98	Esc ` o	ograve	ò	*Reserved*	
\x99	Esc ^ o	ocircumflex	ô	*Reserved*	
\x9a	Esc % o	odieresis	ö	*Reserved*	
\x9b	Esc ~ o	otilde	õ	*Reserved*	
\x9c	Esc ' u	uacute	ú	*Reserved*	
\x9d	Esc ` u	ugrave	ù	*Reserved*	
\x9e	Esc ^ u	ucircumflex	û	*Reserved*	
\x9f	Esc % u	udieresis	ü	*Reserved*	
\xa0	Ctrl-q space	dagger	†	*Reserved*	
\xa1	Ctrl-q !	*Reserved*		Upsilon1	ϒ
\xa2	Ctrl-q "	cent	¢	minute	′
\xa3	Ctrl-q #	sterling	£	lessequal	≤
\xa4	Ctrl-q $	section	§	fraction	⁄

FrameMaker Character Set - (continued)

Hex Code	Key Sequence	Standard Set Char Name	Standard Set Graphic	Symbol Set Char Name	Symbol Set Graphic
\xa5	`Ctrl-q %`, `Alt-period`	bullet	•	infinity	∞
\xa6	`Ctrl-q &`	paragraph	¶	florin	ƒ
\xa7	`Ctrl-q '`, `F8`	germandbls	ß	club	♣
\xa8	`Ctrl-q (`	*Reserved*		diamond	♦
\xa9	`Ctrl-q )`	*Reserved*		heart	♥
\xaa	`Ctrl-q *`	*Reserved*		spade	♠
\xab	`Ctrl-q +`	acute	´	arrowboth	↔
\xac	`Ctrl-q ,`	dieresis	¨	arrowleft	←
\xad	`Ctrl-q -`	*Reserved*		arrowup	↑
\xae	`Ctrl-q .`	AE	Æ	arrowright	→
\xaf	`Ctrl-q /`	Oslash	Ø	arrowdown	↓
\xb0	`Ctrl-q 0`	*Reserved*		degree	°
\xb1	`Ctrl-q 1`	*Reserved*		plusminus	±
\xb2	`Ctrl-q 2`	*Reserved*		second	″
\xb3	`Ctrl-q 3`	*Reserved*		greaterequal	≥
\xb4	`Ctrl-q 4`	yen	¥	multiply	×
\xb5	`Ctrl-q 5`	*Reserved*		proportional	∝
\xb6	`Ctrl-q 6`	*Reserved*		partialdiff	∂
\xb7	`Ctrl-q 7`	*Reserved*		bullet	•
\xb8	`Ctrl-q 8`	*Reserved*		divide	÷
\xb9	`Ctrl-q 9`	*Reserved*		notequal	≠
\xba	`Ctrl-q :`	*Reserved*		equivalence	≡
\xbb	`Ctrl-q ;`	ordfeminine	ª	approxequal	≈
\xbc	`Ctrl-q <`	ordmasculine	º	ellipsis	…
\xbd	`Ctrl-q =`	*Reserved*		arrowvertex	⏐
\xbe	`Ctrl-q >`	ae	æ	arrowhorizex	⎯
\xbf	`Ctrl-q ?`	oslash	ø	carriagereturn	↵
\xc0	`Ctrl-q @`	questiondown	¿	aleph	ℵ
\xc1	`Ctrl-q A`	exclamdown	¡	Ifraktur	ℑ
\xc2	`Ctrl-q B`	*Reserved*		Rfraktur	ℜ
\xc3	`Ctrl-q C`	*Reserved*		weierstrass	℘
\xc4	`Ctrl-q D`	florin	ƒ	circlemultiply	⊗
\xc5	`Ctrl-q E`	*Reserved*		circleplus	⊕
\xc6	`Ctrl-q F`	*Reserved*		emptyset	∅
\xc7	`Ctrl-q G`	guillemotleft	«	intersection	∩

FrameMaker Character Set - *(continued)*

Hex		Standard Set		Symbol Set	
Code	Key Sequence	Char Name	Graphic	Char Name	Graphic
\xc8	Ctrl-q H	guillemotright	»	union	∪
\xc9	Ctrl-q I	ellipsis	…	propersuperset	⊃
\xca	Ctrl-q J	*Reserved*		reflexsuperset	⊇
\xcb	Esc ` A	Agrave	À	notsubset	⊄
\xcc	Esc ~ A	Atilde	Ã	propersubset	⊂
\xcd	Esc ~ O	Otilde	Õ	reflexsubset	⊆
\xce	Ctrl-q N	OE	Œ	element	∈
\xcf	Ctrl-q O	oe	œ	notelement	∉
\xd0	Ctrl-q P	endash	–	angle	∠
\xd1	Ctrl-q Q	emdash	—	gradient	∇
\xd2	Ctrl-q R, Alt-`	quotedblleft	“	registerserif	®
\xd3	Ctrl-q S, Alt-'	quotedblright	”	copyrightserif	©
\xd4	Ctrl-q T, `	quoteleft	‘	trademarkserif	™
\xd5	Ctrl-q U, '	quoteright	’	product	Π
\xd6	Ctrl-q V	*Reserved*		radical	√
\xd7	Ctrl-q W	*Reserved*		dotmath	·
\xd8	Esc % y	ydieresis	ÿ	logicalnot	¬
\xd9	Esc % Y	Ydieresis	Ÿ	logicaland	∧
\xda	Ctrl-q Z	fraction	⁄	logicalor	∨
\xdb	Ctrl-q [	currency	¤	arrowdblboth	⇔
\xdc	Ctrl-q \	guilsinglleft	‹	arrowdblleft	⇐
\xdd	Ctrl-q]	guilsinglright	›	arrowdblup	⇑
\xde	Ctrl-q ^	fi	ﬁ	arrowdblright	⇒
\xdf	Ctrl-q _	fl	ﬂ	arrowdbldown	⇓
\xe0	Ctrl-q `	daggerdbl	‡	lozenge	◊
\xe1	Ctrl-q a	periodcentered	·	angleleft	〈
\xe2	Ctrl-q b	quotesinglbase	‚	registersans	®
\xe3	Ctrl-q c	quotedblbase	„	copyrightsans	©
\xe4	Ctrl-q d	perthousand	‰	trademarksans	™
\xe5	Esc ^ A	Acircumflex	Â	summation	∑
\xe6	Esc ^ E	Ecircumflex	Ê	parenlefttp	⎛
\xe7	Esc ' A	Aacute	Á	parenleftex	⎜
\xe8	Esc % E	Edieresis	Ë	parenleftbt	⎝
\xe9	Esc ` E	Egrave	È	bracketlefttp	⎡
\xea	Esc ' I	Iacute	Í	bracketleftex	⎢

FrameMaker Character Set - (continued)

Hex		Standard Set		Symbol Set	
Code	Key Sequence	Char Name	Graphic	Char Name	Graphic
\xeb	`Esc ^ I`	Icircumflex	Î	bracketleftbt	⎣
\xec	`Esc % I`	Idieresis	Ï	bracelefttp	⎧
\xed	`` Esc ` I ``	Igrave	Ì	braceleftmid	⎨
\xee	`Esc ' O`	Oacute	Ó	braceleftbt	⎩
\xef	`Esc ^ O`	Ocircumflex	Ô	braceex	⎪
\xf0		*Reserved*		*Reserved*	
\xf1	`` Esc ` O ``	Ograve	Ò	angleright	〉
\xf2	`Esc ' U`	Uacute	Ú	integral	∫
\xf3	`Esc ^ U`	Ucircumflex	Û	integraltp	⌠
\xf4	`` Esc ` U ``	Ugrave	Ù	integralex	⎮
\xf5	`Ctrl-q u`	dotlessi	ı	integralbt	⌡
\xf6	`Ctrl-q v`	circumflex	ˆ	parenrighttp	⎞
\xf7	`Ctrl-q w`	tilde	˜	parenrightex	⎟
\xf8	`Ctrl-q x`	macron	¯	parenrightbt	⎠
\xf9	`Ctrl-q y`	breve	˘	bracketrighttp	⎤
\xfa	`Ctrl-q z`	dotaccent	˙	bracketrightex	⎥
\xfb	`Ctrl-q {`	ring	˚	bracketrightbt	⎦
\xfc	`Ctrl-q |`	cedilla	¸	bracerighttp	⎫
\xfd	`Ctrl-q }`	hungarumlaut	˝	bracerightmid	⎬
\xfe	`Ctrl-q ~`	ogonek	˛	bracerightbt	⎭

FrameMaker Messages

This appendix lists in alphabetical order all the messages FrameMaker displays. In most cases, we explain why the message appears and what you can do to solve the problem.

***directoryname* is for a different release.**

When starting, FrameMaker looks for the `.makerinit` directory. The `.makerinit` directory listed in the message is for a different release of the FrameMaker software from the main executable file.

***filename* cannot be opened for writing.**

You are using the Save command. FrameMaker is unable to save the document either because you specified a directory that doesn't exist or because you don't have write permission to the directory.

***filename* does not exist.**

You are using the Open or Import command and have typed a filename that does not exist.

***filename* is a directory.**

You are using the Open or Import command and have typed a directory name instead of a filename.

***filename* is not a monochrome image.**

FrameMaker is trying to load an image from the named file and has failed because the image is not monochrome. FrameMaker can load only monochrome images.

***filename* is not a readable image.**

You are using the Import command, and FrameMaker is unable to read the file you specified.

***filename* is not an image file.**

FrameMaker is trying to load an image from the named file and has failed because the file doesn't contain an image.

***filename* is not an importable file.**

You are using the Import command, and FrameMaker is unable to import the file you specified.

***filename* is unreadable.**

You are using the Import command, and FrameMaker is unable to read the file you specified.

An (unknown) error has occurred reading this file.

You are using the Open command, and a system error occurred when FrameMaker read the file. Look in the console window for UNIX error messages.

Bad eps file *filename.*

You are printing a document containing an Encapsulated PostScript (EPS) file whose format has been corrupted.

Bad page range.

You have specified an incorrect page range in the Delete Page dialog box (the start page must be smaller than the end page). Display the Delete Page dialog box again and retype the page range.

Cannot allocate bitmap pool.

You are trying to start FrameMaker, and there is insufficient swap space memory for FrameMaker to run. You need to terminate some other process(es) to free enough memory to start FrameMaker.

Cannot allocate colors. Aborting.

FrameMaker was unable to use the background and foreground colors specified on your command line or X Window resource file, and it could not access the default colors (black and white). Set up your workstation so that black, white, and any other colors you want are available. See Appendix D in the *FrameMaker Reference Manual* for more information on X window resource options. See your workstation documentation for instructions.

Cannot allocate memory pool.

You are trying to start FrameMaker, and there is insufficient swap space memory for FrameMaker to run. You need to terminate some other process(es) to free enough memory to start FrameMaker.

Cannot allocate startup memory.

You are trying to start FrameMaker, and there is insufficient swap space memory for FrameMaker to run. You need to terminate some other process(es) to free enough memory to start FrameMaker.

Cannot be undone. OK to continue?

Due to memory limitations, you cannot undo some FrameMaker actions. This message warns you that the Undo command cannot undo the command you chose. Click OK to perform the command or click Cancel to stop the command.

You may want to use the Save command before carrying out this action. Then, if you do not like the results, you can quit the document without saving and reopen the saved version.

Commands that may require too much memory to be undone include Paragraphs, Freeze and Unfreeze Pagination, Delete Page, and Change All (in the Search window).

Cannot complete printing.

This is a general error message that the printer driver displays when it can't finish printing a document. Usually, it appears with a more specific message describing the problem. If it appears by itself, however, it probably means the printer driver ran out of disk space (when you print a document, FrameMaker creates a temporary disk file that it sends to your printer). You should free some disk space in your home directory and try printing the document again.

Cannot connect to X server *name*. Check $DISPLAY environment variable and -d option. Possibly run "xhost <client machine>" on the server.

The display was not specified on the command line or in the $DISPLAY variable, there is no server running on the client workstation, the named server does not exist, or the named server does not allow a connection to this client. See *Changing FrameMaker X Window Options* in Appendix D for information on the `display` option. If the server does not allow a connection to this client, run `xhost` *client_machine* on the server. See your window system documentation for more information.

Cannot convert image *filename*.

You are printing a document containing an imported image, and an error has occurred because the print driver does not have enough memory to handle the image.

Cannot delete master page.

You have chosen the Delete command from the Page menu and have specified the master page in the Delete dialog box. The master page cannot be deleted from a document.

Cannot directly import a maker file.

You are using the Import command and have specified a FrameMaker Maker Format file as the file to import. FrameMaker cannot directly import Maker format files. However, it is possible to merge files using MIF Save and Import. See the Save and Import commands in Chapter 3 for instructions.

Cannot display modeless dialog box.

You have chosen the Markers, Search, or Spelling Checker command, and FrameMaker is unable to display the window associated with the command. You may have too many windows open. Quit some windows and try again.

Cannot display the Tools window.

You have chosen the Tools command, and FrameMaker is unable to display the Tools window. You may have too many windows open. Quit some windows and try again.

Cannot initialize user interface toolkit interface.

You are trying to start FrameMaker, and there is insufficient swap space memory for FrameMaker to run. You need to terminate some other process(es) to free enough memory to start FrameMaker.

Cannot find any document language data files.

FrameMaker is unable to start because it can't find any hyphenation and spelling files. FrameMaker looks for these language files in the directory `.makerinit/langdir`. For more information, see Appendix D.

Cannot find desired default document language files. Setting default document language to *languageName*.

FrameMaker is unable to start because it can't find the hyphenation and spelling files for the default document language specified in the file `.makerinit/languages`. FrameMaker uses the language shown in the message as the default document language. See Appendix D for more information or call your local Data General service representative for help.

Cannot find insertion point image.

FrameMaker cannot find one of the files in the `.makerinit` directory. See your FrameMaker Release Notice or call your local Data General service representative for more information.

Cannot find .makerinit directory. Please read Installation Instructions.

You are trying to start FrameMaker, and FrameMaker cannot find the `.makerinit` directory. See your FrameMaker Release Notice or call your local Data General service representative for more information.

Cannot handle an image of that size.

You are importing a bitmap file (sometimes called a raster file). The image file's dimensions are too large or too small to be imported. Neither dimension of the image can be smaller than 8 pixels or larger than 2000 pixels.

To import an image larger than FrameMaker supports, you must crop the image using an image editor.

Cannot import a color image file.

You are using the Import command, and you have specified a color or gray-scale image file as the file to import. FrameMaker can import only monochrome images.

Cannot import an image file.

You are using the Import command, and FrameMaker is unable to import the file because it has a bad format.

Cannot import MIF when pagination is frozen.

You are using the Import command in a document where auto pagination is turned off (see the Freeze Pagination command in Chapter 3). To import a MIF file into this document, you must turn on auto pagination.

Cannot initialize commands.

You are trying to start FrameMaker, and there is insufficient swap space memory for FrameMaker to run. You need to terminate some other process(es) to free enough memory to start FrameMaker.

Cannot initialize file locking.

You are trying to start FrameMaker, and there is insufficient swap space memory for FrameMaker to run. You need to terminate some other process(es) to free enough memory to start FrameMaker.

Cannot initialize font database interface.

You are trying to start FrameMaker, and there is insufficient swap space memory for FrameMaker to run. You need to terminate some other process(es) to free enough memory to start FrameMaker.

Cannot initialize formatter.

You are trying to start FrameMaker, and there is insufficient swap space memory for FrameMaker to run. You need to terminate some other process(es) to free enough memory to start FrameMaker.

Cannot initialize imager interface

You are trying to start FrameMaker, and there is insufficient swap space memory for FrameMaker to run. You need to terminate some other process(es) to free enough memory to start FrameMaker.

Cannot initialize operating system interface

You are trying to start FrameMaker, and there is insufficient swap space memory for FrameMaker to run. You need to terminate some other process(es) to free enough memory to start FrameMaker.

Cannot initialize platform dependent user interface

You are trying to start FrameMaker, and there is insufficient swap space memory for FrameMaker to run. You need to terminate some other process(es) to free enough memory to start FrameMaker.

Cannot initialize user interface

You are trying to start FrameMaker, and there is insufficient swap space memory for FrameMaker to run. You need to terminate some other process(es) to free enough memory to start FrameMaker.

Cannot load image file *filename*.

You are working with or printing a document containing an imported image. If you are working with a such a document, FrameMaker is unable to load the image file from disk. If you are printing, an error has occurred because the printer driver does not have enough memory to handle the image.

Cannot lock this file for your use.

You are trying to open or save a file in a directory in which you do not have write permission, but you do have permission to modify the file. This is important only if you are working in an environment where someone else might change the file and save it while you are working on it. If no one is likely to change the file except you, click OK/Continue and proceed as usual.

To lock a file for your use, FrameMaker must create a lock file in the same directory as the real file. Since you do not have write permission in the file's directory, FrameMaker cannot create the lock file. You may continue to use the file, and other users can open and save the file while you are working on it.

Cannot open *filename*.

You are trying to open a file using the Open or Import commands, or the Change Dictionaries dialog box. FrameMaker cannot open the file you specified either because it does not exist or because you do not have read permission for the file.

Cannot open an image file--use Import.

You are using the Open command on an image file.

Cannot open *filename* because *reason*.

You are trying to start FrameMaker or have chosen a command that displays a dialog box. This message represents a problem caused by deleting necessary files in the `.makerinit` directory. Make sure that the file listed in the message exists and that you have read permission to it (see Appendix D for more information on where the resource files are located). If you still have a problem, call your local Data General service representative for help.

Cannot open dictionary *filename*.

This message appears when you start FrameMaker. FrameMaker is unable to open the default personal dictionary `~/.fmdictionary`. Normally, you create the default personal dictionary using the Learn Word button when you check the spelling in a document.

Cannot open fonts.

You are trying to start FrameMaker, and FrameMaker cannot open its fonts for one of the following reasons: it cannot find the directory `.makerinit/fontdir`; it cannot find the file `fontlist` within that directory; the `fontdir` directory does not contain usable font files. See your FrameMaker Release Notice or call your local Data General service representative for more information.

Cannot open *filename* for writing

You used the Change Dictionaries dialog box to write a dictionary file to disk, and FrameMaker is unable to write to the file either because the file doesn't exist or because you don't have write permission to it.

Cannot open hyphenation tables.

You are trying to start FrameMaker. This message represents a problem caused by deleting necessary files in the `.makerinit` directory. See your FrameMaker Release Notice or call your local Data General service representative for more information.

Cannot open image file *filename*.

You have chosen the Open command and have tried to open an image file. An image file can be imported only into a document. See the Import command in Chapter 3.

Cannot open include file *filename*.

You are printing a document to a PostScript printer and an error occurred when the printer driver attempted to open the named include file.

Cannot open maker font.

You are trying to start FrameMaker. This message represents a problem caused by deleting necessary files in the `.makerinit` directory. See your FrameMaker Release Notice or call your local Data General service representative for more information.

Cannot open marker labels.

You are trying to start FrameMaker. This message represents a problem caused by deleting necessary files in the `.makerinit` directory. See your

FrameMaker Release Notice or call your local Data General service representative for more information.

Cannot open ruler font.

You are trying to start FrameMaker. This message represents a problem caused by deleting necessary files in the `.makerinit` directory. See your FrameMaker Release Notice or call your local Data General service representative for more information.

Cannot open tabs.

You are trying to start FrameMaker. This message represents a problem caused by deleting necessary files in the `.makerinit` directory. See your FrameMaker Release Notice or call your local Data General service representative for more information.

Cannot open this type of file.

You are using the Open command on a file of unknown type. It is not a Maker document, MIF document, text file, or image file.

Cannot proceed because cannot find *languageName* user interface directory.

FrameMaker is unable to start because it can't find the directory for the user interface language specified in the file `.makerinit/languages`. See Appendix D for more information or call your local Data General service representative for help.

Cannot proceed because cannot find .makerinit directory.

You are trying to start FrameMaker, and FrameMaker cannot find the `.makerinit` directory. See your FrameMaker Release Notice or call your local Data General service representative for more information.

Cannot proceed because cannot find version *versionNumber* of .makerinit directory.

FrameMaker is unable to start because it can't find a version of the `.makerinit` directory that matches the program version. See your FrameMaker Release Notice or call your local Data General service representative for help.

Cannot read in font info.

You are trying to start FrameMaker. This message represents a problem caused by deleting necessary files in the `.makerinit` directory. See your FrameMaker Release Notice or call your local Data General service representative for more information.

Cannot read in marker info.

You are trying to start FrameMaker. This message represents a problem caused by deleting necessary files in the `.makerinit` directory. See your

FrameMaker Release Notice or call your local Data General service representative for more information.

Cannot read in misc info.

You are trying to start FrameMaker. This message represents a problem caused by deleting necessary files in the `.makerinit` directory. See your FrameMaker Release Notice or call your local Data General service representative for more information.

Cannot read macro file.

You are using the Keyboard command and have asked to read macros from a file. FrameMaker cannot read the specified macro file, or the file is not a macro file. Check the file and the filename in the Keyboard dialog box and try again.

Cannot read this version.

You are using the Open command and have tried to read a FrameMaker file that is not compatible with the version of FrameMaker that you are using.

This can happen when you have two versions of FrameMaker on your system and you are using the old version to open a file saved with the new one (in Maker format). To get around this, open the file with the new version, save it in Maker Interchange Format, and try again.

Cannot save MIF when pagination is frozen. Save in Maker Format.

You are using the Save command and have checked the Maker Interchange Format setting for a document with frozen pagination. You can save a document with frozen pagination only in Maker format.

Cannot scale image *filename*.

You are printing a document to a PostScript printer and an error occurred when the printer driver tried to scale the named image file. Most likely, this happens because the printer driver does not have enough memory to handle the image.

Cannot scale imPRESS documents.

In the Print dialog box, you have specified a value other than 100% in the Scale edit box. When printing imPRESS documents, you can print only at a scale of 100%.

Cannot set default font. Check font list.

You are trying to start FrameMaker. The `.makerinit/fontdir/fontlist` file has specified a default font that does not exist in `.makerinit/fontdir`.

This is most likely the result of FrameMaker using the wrong `.makerinit` directory, probably one from an earlier version of

FrameMaker. See your FrameMaker Release Notice or call your local Data General service representative for more information.

Cannot shift image *filename*.

You have chosen to print to an imPRESS document containing bitmaps. FrameMaker is unable to print the image named in the message because it doesn't have enough memory. To free memory for the printer driver, quit some document windows or kill some other processes.

Cannot start FrameMaker's windows.

You are trying to start FrameMaker without first starting your window system, or the window system is full and there are no more windows available.

Cannot stretch byte-reversed images.

FrameMaker cannot stretch the selected image. Try reversing bytes using the `fmcolor` filter. For instructions on using `fmcolor`, type **`fmcolor`** at the UNIX prompt.

Cannot undo.

You are using the Undo command, and your previous action (if any) cannot be undone. If the previous action has devastated your document, you may want to quit without saving and reopen the previously saved version.

Cannot write to file.

FrameMaker can't write to the file you specified either because the file doesn't exist or because you don't have write permission to it. Make sure you typed the filename correctly.

Cannot write to macro file.

You are using the Keyboard command and have asked to add new macros to a file. FrameMaker cannot open the specified file for writing. Check the file's permissions and/or the filename in the Keyboard dialog box and try again.

Cannot write to printfile.

You are using the Print command, and FrameMaker is unable to open the file into which it is to write printer code. This occurs when you have turned on the Print Only to File check box, and you do not have write permission for the specified Printer File.

Catalog empty.

You are using the Catalog command, and no paragraph formats have been stored in the Catalog. The Paragraphs command is used to store paragraph formats in the Catalog.

Choose Catalog member.

You are using the Catalog command, and have not selected an item from the list at the left of the Catalog dialog box. Select a paragraph format from the list by clicking on its name and try again.

Color not recognized: *name*.

The server did not recognize the color specified in your command line or your X Window resource file. Instead, FrameMaker uses the default background color (white) or foreground color (black). For more information on color options, see *Changing FrameMaker X Window Options* in Appendix D.

Could not import image.

You are using the Import command, and FrameMaker failed to import the image.

Could not find postscript_prolog.

You have chosen to print a document on a PostScript printer. FrameMaker is unable to print the document because it can't find the file `postscript_prolog` in `.makerinit`.

Could not find string *stringNumber* in file *filename*.

FrameMaker is unable to display a message either because the resource file containing the message has an error in it, or because the file has been corrupted. See Appendix D for information on where the string resource files are located or call your local Data General service representative for help.

Could not put up dialog box.

You have selected a command that requires FrameMaker to display a dialog box, and FrameMaker is unable to display the dialog box.

Current paragraph not visible.

You have selected the Tabs command to add a tab stop, and the paragraph containing the insertion point isn't visible. Display the page containing the insertion point and select the Tabs command again.

Delete Page cannot be undone. OK to continue?

Using the Delete command, you have specified pages to delete from a document. This message warns you that FrameMaker cannot undo the Delete command due to memory limitations. Click OK to delete the pages anyway or click Cancel to keep from deleting the pages.

Dimension too big.

You are using the Scale command, and the specified scaling factor or dimension would create an object that is too large. Objects can be up to 2000 points in each dimension.

Do you REALLY want to save in Maker format? (MIF is safer).

FrameMaker has run out of memory and is attempting to save your files before exiting. In the process it has displayed the Save dialog box for an open file, suggesting that you use the MIF save setting. You have switched to Maker format. FrameMaker is advising that, due to memory problems, you are less likely to lose your file if you save in MIF format. Click OK to try saving in Maker format, or click Cancel to switch back to MIF format.

Edit Marker failed.

You are using the Edit Marker button in the Markers window, and an unexpected problem has occurred. You should save your current work and restart FrameMaker. Please call your local Data General service representative to report this problem.

Empty document.

You are using the Print command, and there is nothing in the document to print.

Enter tag name in Tag box.

You are using the Paragraphs command and have turned on the Tag check box in the Apply To area of the Paragraphs dialog box. You haven't, however, typed a name in the Tag edit box. Type a tag or turn off the check box. See the Paragraphs command in Chapter 3 for more information.

Expected '*versionNumber*' saw '*versionNumber*'.

You have chosen to print a document. FrameMaker is unable to print the document because your FrameMaker version and your printer driver version do not match.

File cannot be opened for writing.

You are using the Save or Capture command, and the file you are trying to save to cannot be opened for writing by FrameMaker, either because you do not have write access to the file or its directory or because the file system is full.

File does not have read permission.

You are using the Open command, and FrameMaker cannot read the file because you don't have read permission to it.

File is a directory.

You are using the Open or Save command, and the filename you typed in the dialog box is a directory, not a file. Check the filename in the Open or Save dialog box or use the *ls* command in a shell window.

File is not a dictionary.

You are using the Change Dictionaries dialog box and chose to read a dictionary file. The file you specified, however, isn't a dictionary file. Display the Change Dictionaries dialog box and type the filename again in the File Name edit box. Or, if you typed the correct filename, edit the file to have the correct file format (see Spelling Checker command in Chapter 3 for the correct form).

First select two TextRects.

You are using the Connect TextRects command. Before you can connect two TextRects, you must select them. See the Connect TextRects command in Chapter 3 for more information.

$FMHOME is not set.

You are trying to start FrameMaker without using the FrameMaker start-up script *install_dir*`/bin/maker` (this script correctly sets the environment variable FMHOME). Use the UNIX *cd* command to change to the directory in which FrameMaker was installed and type `maker`.

Font unavailable.

You are using the Fonts, Paragraphs, or the Headers & Footers command. The combination of font family, style, and size you have specified is not available on the version you are running. Select a different set of font settings and try again.

Full screen cannot be accessed.

You are using the Capture command, and FrameMaker has been unable to gain access to the entire screen. This is an unexpected condition. Please call your local Data General service representative if this message appears.

Giving up ... bye!

You are trying to start FrameMaker, and FrameMaker is unable to find the `.makerinit` directory. FrameMaker has displayed a dialog box for you to give the full pathname of the `.makerinit` directory, and you have clicked Cancel in that dialog box. FrameMaker cannot run unless it can find the `.makerinit` directory, so FrameMaker exits.

Having trouble communicating with the license server.

You are unable to use a Frame product because your machine is unable to communicate with the license server host. See your system administrator for help. For more information, see *Problems with the License Server Host and Server Process* in Appendix H of the *FrameMaker Reference Manual.*

Incompatible font from file *filename*.

Incompatible fontmetric from file *filename*.

You are starting FrameMaker. The named file does not exist in a format usable by FrameMaker. This may indicate that FrameMaker has found the wrong `.makerinit` directory (probably from an earlier version of FrameMaker stored elsewhere on your system). Try restarting FrameMaker and watching the console window to see which `.makerinit` it is using. See your FrameMaker Release Notice or call your local Data General service representative for more information.

Insufficient memory.

You are starting FrameMaker and you do not have enough swap space memory for FrameMaker to run. You normally need at least 2.2 Mb of free swap space to start FrameMaker.

Insufficient memory to handle selection.

You are trying to select a complex image, but there is not enough memory for FrameMaker to perform the selection.

Insufficient memory to initialize Maker windows.

FrameMaker has run out of memory.

Insufficient memory to scale the bitmap.

You are trying to scale a bitmap image, but you do not have enough memory for FrameMaker to scale it.

Insufficient memory (2MB needed).

You are trying to start FrameMaker, and there is insufficient swap space memory for FrameMaker to run. You need to terminate some other process(es) to free enough memory to start FrameMaker.

Internal error *errornumber.*

FrameMaker has encountered an internal error. Please contact your local Data General service representative and describe the actions leading up to this error.

Invalid connection.

You are using the Connect TextRects command. See the Connect TextRects command in Chapter 3 for more information.

Junk (*characters*) after command: *commandname*.

An extra character was found after a command.

Kit number: *message*.

This message is an aid to debugging window and server problems. Please make a note of it and contact your local Data General service representative.

Last action in another document.

You have chosen the Undo command with the pointer in the wrong document. If you have the flash feature of your window system set, FrameMaker flashes the window of the appropriate document after this message.

License server's protocol is not being followed.

You are unable to run a Frame product. The most likely cause is that the date in the user's operating system is more than four hours different from the license server host's date. The system administrator must change the date on the user's machine so that it matches the date of the license server host. For more information, see *Problems obtaining a license* in Appendix H of the *FrameMaker Reference Manual.*

Maker must be run with contrasting foreground and background colors. Aborting.

Either your command line or one of your X Window resource files specified foreground and background colors that are too similar. This message can also occur when you've specified a only one color (either foreground or background) and it is too close to the default for the other. See *Changing FrameMaker X Window Options* in Appendix D for information on changing these options, or try using the **`maker -fg white -bg black`** command to start FrameMaker.

Markers cannot be put in this type of line.

You are trying to put a marker into a text line. Markers can be put only in text inside a TextRect.

Missing <FrameFinalUpdate ...> line in the frameusers file.

Without the <FrameFinalUpdate> line in the `frameusers` file, you won't be able to use a Frame product. Your system administrator must run the `fmlicense` script and add the <FrameFinalUpdate> line. See Appendix H in the *FrameMaker Reference Manual.*

Missing <Product ...> line in the frameusers file.

Without the <Product> line in the `frameusers` file, you won't be able to use a Frame product. Your system administrator must run the `fmlicense` script and add the <Product> line. See Appendix H in the *FrameMaker Reference Manual.*

Missing <Vendor ...> line in the frameusers file.

Without the <Vendor> line in the `frameusers` file, you won't be able to use a Frame product. Your system administrator must run the `fmlicense` script and add the <Vendor> line. See your FrameMaker Release Notice for more information.

Negative dimension.

You are using the Scale command, and you have specified a negative dimension.

New Marker failed.

You are using the New Marker button in the Markers window, and FrameMaker has run out of memory. Save your current work and restart FrameMaker.

No active document.

You are using the Search window, and the document in which you want to search and/or replace has no insertion point.

No correction specified.

In the Spelling Checker window, you clicked Correct Word, and the Correction edit box is empty.

No current line.

In the Go To dialog box, you chose to go to a line number, and there is no current line (no line containing the insertion point).

No current paragraph.

You are using the Catalog, Copy Pgf Format, Paragraphs, or Tabs command, and you have not indicated which paragraph you wish to change.

No file specified.

You are using the Change Dictionaries dialog box and have chosen to read or write a dictionary file without specifying a filename.

No insertion point.

The function you are trying to perform requires an insertion point in a document.

No memory available for license server process.

The license server process was unable to continue running due to a lack of memory on the license server host. Your system administrator must terminate some processes(es) on the license server host and start the license server process again. See Appendix H in the *FrameMaker Reference Manual.*

No new macros.

You are using the Keyboard command with the Add New Macros to File setting turned on, and there are no new macros to be saved.

No objects selected.

You are using the Distribute command from the Tools window, and no objects have been selected.

No odd pages and no even pages specified.

You are using the Print command, and both Print Odd-Numbered Pages and Print Even-Numbered Pages are turned off; there are no pages to be printed.

No pages to delete.

You are trying to delete pages from a document in which the only page is a master page (you can't delete the master page). Make sure you have the arrow pointer in the document window from which you want to delete pages.

No personal dictionary.

You have chosen a function requiring a current personal dictionary, and the personal dictionary is set to none.

No printfile.

You have chosen to print a document. FrameMaker is unable to print the document because your FrameMaker version and your printer driver version do not match.

No room for document.

You are using the Open command, and there isn't enough memory. Try closing some documents or killing some processes.

No room in memory pool for frame image. You may need to change your .makerinit/bitmaps file.

FrameMaker is trying to load an image that is too big to fit in the bitmap memory pool. You need to close other documents containing bitmap images, or increase the pool size number given in `.makerinit/bitmaps`.

No room in memory pool for image *filename*. You may need to change your .makerinit/bitmaps file.

FrameMaker is trying to load an image that is too big to fit in the bitmap memory pool. You need to edit the pool size number given in `.makerinit/bitmaps` if you want to see this image on the screen. The pool size must be larger than the product of the bitmap's (displayed) width and height (in points) divided by 8.

No room in memory pool for raster image. You may need to change your .makerinit/bitmaps file.

FrameMaker is trying to load an image that is too big to fit in the bitmap memory pool. You need to close other documents containing bitmap images, or increase the pool size number given in `.makerinit/bitmaps`.

No TextRects with Auto Connect on this page.

You have chosen the Number of Columns command to change the format of your document. The Number of Columns command, however, works only on TextRects that have Auto Connect on. FrameMaker was unable to find any such TextRects in the range of pages you specified in the Number of Columns dialog box. See the Number of Columns command in Chapter 3 for more information.

No Undo for Connect TextRects.

You have connected two TextRects with the Connect TextRects command and then chosen the Undo command. You cannot undo the Connect TextRects command. You can, however, disconnect the TextRects using the Disconnect Head or the Disconnect Tail command.

Not all languages used in this document are available. OK to proceed?

You are opening a FrameMaker document, and FrameMaker is unable to find the hyphenation and spelling files for all the languages used in the document. If you click OK, FrameMaker uses the default document language for all paragraphs for which it can't find the language files. If you edit these paragraphs, FrameMaker hyphenates them based on the default document language. For more information on where FrameMaker searches for language files and the default document language, see Appendix D.

Not a monochrome image.

You are using the Import command, and you have specified a color or gray-scale image file as the file to import. FrameMaker can import only monochrome images.

Not enough memory to fetch string *stringNumber* from file *filename*.

You are trying to start FrameMaker, and there is insufficient swap space memory for FrameMaker to run. You need to terminate some other process(es) to free enough memory to start FrameMaker.

Not enough memory to load image *filename*. *Number* bytes needed.

You have chosen to print a document containing bitmaps. FrameMaker is unable to scale the bitmap named in the message because it doesn't have enough memory. To free memory for the printer driver, quit some document windows or kill other processes.

Not enough memory to scale image *filename*.

FrameMaker is trying to scale an image to a different size and doesn't have enough memory to do it.

Nothing to cut.

You are using the Cut command and haven't selected anything to cut.

Nothing to paste.

You are using the Paste command, and the clipboard is empty.

OK to begin recording keys?

When you select the Record Keys command from the Document menu, FrameMaker displays this alert box to confirm that you want to begin recording a keyboard macro. Click OK to begin recording a macro. Click Cancel to avoid recording a macro.

OK to change the printer code setting on this TextRect?

You are using the Printer Code command to tell FrameMaker to treat the contents of a TextRect as printer commands rather than as printable text.

OK to delete *tagname*?

You are using the Catalog command and have clicked the Delete from Catalog setting to remove the named paragraph format from the Catalog.

OK to overwrite existing Catalog entry?

You are using the Paragraphs command and have clicked the Catalog setting in the Apply To area of the dialog box in order to store the paragraph format in the Catalog. FrameMaker is warning you that there is already an entry in the Catalog with the same name as the one you are trying to store.

OK to use TextRect which is not visible?

You are using the Connect TextRects command, and one of the TextRects is not visible. This usually happens if you are trying to join two TextRects on different pages.

Out of memory.

FrameMaker needs more memory. Try closing some documents or killing some processes.

Page size too big.

In the New dialog box, you have selected a custom page size with a width or height greater than 14 inches.

Pagination already frozen.

Pagination already unfrozen.

You are using the Auto Pagination command to freeze (or unfreeze) a document whose pagination is already frozen (or unfrozen).

Please keep your console visible in case something important is displayed there later

This message appears in the console window when you start FrameMaker. Since FrameMaker sometimes displays messages in the console window, you should keep it visible.

Printing failed. Could not write all of file.

You are using the Print command, and a system error occurred that prevented FrameMaker from writing a printer file. Look in your console window for UNIX error messages.

Printing failed - could not run *printerdrivername*.

FrameMaker is unable to start the print driver named in the message. When this message appears, you won't be able to print documents. This message can appear for a number of system-related reasons: You may have too many processes running, you may not have execute access to the `.makerinit` directory or the print driver file, or the files may have been accidentally removed.

Problem: File does not contain a valid image.

You are using the Import command, and FrameMaker is unable to import the image because the file has a bad format.

Problem: Image has invalid dimensions.

You are using the Import command, and FrameMaker is unable to import the file either because it's not an image file or it has a bad format. See the Import command in Chapter 3 for information on the types of image files you can import.

Problem: Image is in color.

You are using the Import command to import a color image. FrameMaker can import only monochrome images. You can convert color images to monochrome with the shell script fmcolor. See the Import command in Chapter 3 for more information on converting color images.

Put insertion point in TextRect.

You selected either the Auto Connect, Feathering, or Printer Code command without selecting the TextRect you want the command to affect. Put the insertion point in a TextRect and select the command again.

Scale factor too large or too small.

You are using the Scale command and have specified a scaling factor that is less than 10% or greater than 500%.

Select one anchored frame to be reanchored.

You are using the Anchored Frame command, and you have selected more than one anchored frame or object.

Select one object for adjustment.

You have pointed on an object and pressed either the Alt-arrow or Shift-arrow keys to move the object, and you have more than one object selected.

Select one object to scale.

You are using the Scale command, and you have more than one object selected.

Select one square/rectangle/circle/ellipse.

You are using the Set # Sides command, and you have selected more than one object.

Select one TextRect.

You selected the Disconnect Head or Disconnect Tail command without first selecting the TextRect you want to disconnect. Put the insertion point in the TextRect and select the command again.

Set the insertion point inside a TextRect.

You selected the Split TextRect command without selecting the TextRect you want to split. Put the insertion point in the TextRect and select the command again.

Set Marker Type.

You are using the Markers window to insert a new marker or edit an existing one, and you have not selected a marker type.

Set the insertion point inside a TextRect, or select an existing anchored frame.

You are using the Anchor command, and there is neither an insertion point for a new anchored frame nor a selected anchored frame to be edited.

Sorry, no room. Macro ignored.

You tried to record a keyboard macro. FrameMaker was unable to record the macro because it didn't have enough memory (you can have about 100 macros in memory at once). Use the Keyboard command to save and then clear the current macros. Then edit the `kbmacros` file and remove any unwanted or surplus macros. See the Record Keys and Keyboard commands in Chapter 3 for more information.

Spooling *filename* to your home directory. You may need to remove this file after printing.

You are printing a document, and the print file is too large to fit into `/tmp`. FrameMaker then tries to write the print file in your home directory. Normally print files are removed after printing, and if a print file is created in your home directory that does not have execute permission set for "other," the print file may not get deleted.

String resource file version *number* does not match executable version *number*.

You are trying to start FrameMaker, and the string resource files (files FrameMaker uses to display messages) are a different version from the version of FrameMaker you are using. The string resource files are located in the `.makerinit/<language>` directories.

Syntax error in dialog box file.

You have selected a command that displays a dialog box. FrameMaker is unable to display the dialog box because the resource file containing the dialog box's definition has an error in it, or the file has been corrupted. (See Appendix D for information on where the dialog box resource files are located.)

Syntax error in string *stringNumber* of file *filename*.

FrameMaker is unable to display a message either because the resource file containing the message has an error in it, or because the file has been corrupted. See Appendix D for information on where the string resource files are located or call your local Data General service representative for help.

Temporary file cannot be created.

You are importing or opening a text file, and FrameMaker is unable to make a temporary file for use in processing the text. This can happen if you do not have read/write permission in `/tmp` or your home directory or if you have too many files open at once. Check your access to `/tmp` and your home directory. If you still have trouble, call your local Data General service representative.

The following document may have been scrambled.

FrameMaker has run out of memory and is attempting to save your files before exiting. In the process it has brought up the Save dialog box for an open file, which may be in an inconsistent state due to the memory failure. Save the file in Maker Interchange Format so that you can patch it up later.

The selection is too large for X cut buffers.

FrameMaker has run out of memory.

The spelling checker cannot handle *specific string.*

The spelling checker cannot handle this capitalization.

The spelling checker cannot handle this word.

The spelling checker cannot handle repeated words.

You clicked Learn Word or Allow Word in This Document in the Spelling Checker window, and the spelling checker is unable to add the word in the Word edit box to your personal or document dictionary. The spelling checker cannot learn words that it questions as repeated, unusually capitalized, or unusually hyphenated.

There are no licenses available for you.

You tried to use a Frame product, but were unable to because the maximum number of allowed users has been reached. Try getting a license again later. You can click the License button in the Info dialog box to see which users currently have a license to use the product.

There are too many windows open. Please close one.

You have chosen a command that opens a FrameMaker window. FrameMaker is unable to open another window because it has reached its limit. Quit a FrameMaker window and try again.

There is not enough memory for FrameMaker to keep running...

FrameMaker has unexpectedly run out of memory and is going to quit. If you want to save your work, click Yes. FrameMaker displays the Save dialog box for each document with unsaved changes. Save in Maker Interchange Format so that you can later patch up any errors that may have been caused by the memory failure. Notice, however, that documents whose pagination is frozen can be saved only in Maker format.

There is not enough memory! Please free some memory...

FrameMaker has estimated that its available memory is too low to safely carry out your most recent command. Before proceeding, free some memory using one or more of the steps suggested in the dialog box.

There was a problem reading your file. Please check your file system, disk, and network.

You are trying to open a FrameMaker document. FrameMaker is unable to open the file due a problem with your file system, disk, or network. Try opening the file again. If the problem persists, see your system administrator for help.

This command is for use on documents with frozen pagination.

You are using the Repaginate command in a document with unfrozen pagination. See the Freeze Pagination and Repaginate commands in Chapter 3 for more information.

This connection is not possible.

You are using the Connect TextRects command on two TextRects that cannot be connected. See the Connect TextRects command in Chapter 3 for more information.

This document uses unavailable fonts. OK to continue?

You have chosen to open a document that was formatted for another printer type or font set. If you click OK, FrameMaker reformats the document using available fonts. If you don't want to reformat the document, click Cancel.

This file is too small to be a Maker file.

You are using the Open command on a file that looks like a FrameMaker file, and is too small to have any useful data. If it is a corrupted FrameMaker file, you can delete it.

This is a Maker 0.6 document--see Installation Instructions.

You are using the Open command to open a FrameMaker version 0.6 file. Your current version of FrameMaker no longer reads these files directly. There is a utility to convert them into Maker Interchange Format that *can* be read by FrameMaker. For more information, call your local Data General service representative.

This is demonstration software. No license is necessary.

You are running a demonstration version of FrameMaker and you clicked the License button in the Info dialog. You don't need a license to use a demonstration version of FrameMaker.

This product has been idle for too long or was unable to freshen its license quickly enough.

If you see this message, please contact your local Data General service representative.

This type of anchored frame cannot be moved.

You tried to move the selected anchored frame by pointing on it with the mouse or by pressing one of the arrow keys. You can move only anchored frames that are anchored in-line or in-margin. See the Anchored Frame command in Chapter 3 for more information.

This vendor's password has expired.

The temporary password for the Frame product you are using has expired. To continue using this product, you must obtain a new password from Frame.

Too many fonts for printer.

You have chosen to print an imPRESS document that uses more than 47 fonts. To print the document, you must reduce the number of fonts to be less than 48.

Too many windows.

You are using the Open or New commands, and the window system is full and there are no more windows available. Close some documents or kill some processes and try again.

Translation failure.

FrameMaker is unable to print your document. Most likely, your FrameMaker version and your printer driver version do not match. Please call your local Data General service representative for help.

Unknown command-line options: *options*.

FrameMaker did not recognize the named option. Try using the full option name. See *Changing FrameMaker X Window Options* in Appendix D for more information.

Unknown user (or yellow pages busy).

You tried to save a document, using the ~ character in the filename. Unfortunately, the system can't expand the ~ character to a user's home directory either because the user you specified doesn't exist or the yellow pages is busy. Make sure you type the correct user name in the Save dialog box and try again.

Warning: max number of images = *m*, max memory for images = *n*.

You are starting FrameMaker, and your `.makerinit/bitmaps` file has specified some unexpected values for the bitmap memory pool. This file allows you to control the maximum number and amount of memory that FrameMaker can allocate to bitmaps. For more information, see *Image Cache Size,* in Appendix D.

Warning: overflowed textrect on page *n*.

You are printing a document, and there is some text that is not going to appear on the printout because it is below the bottom of a TextRect on the indicated page.

Where is .makerinit?

You are starting FrameMaker, and it cannot find the `.makerinit` directory. If you know where it is located, type the pathname in the dialog box. Otherwise see your FrameMaker Release Notice for more information or call your local Data General service representative for help.

Window data lock broken after time limit exceeded by pid *n*.
The offending process was sent SIGXCPU.

This is a UNIX system message that can occur when FrameMaker takes extra time to redraw all or part of a document. You can ignore it.

Write failed. Is your file system full?

You are using the Save command, and a UNIX system error has occurred during the save. A typical case is when your file system is full and your document's file cannot be completely written.

Write permission not set on this file.

You are using the Save command, and you are trying to save a document into a file for which you do not have write permission. Try saving to a different name, or modify the permissions on the file or its directory as appropriate.

Wrong version number for dictionary *filename*.

At the Change Dictionaries dialog box, you tried to read a dictionary file. The file is in the correct format, and was created for a different version of FrameMaker.

X Error: *message*. Request code: *message*.

A FrameMaker request resulted in an X window server error. If there is a problem with one of FrameMaker's windows, you will see a "Kit number" message. Otherwise, there is a server problem. See your window system documentation for more information.

You are not allowed to use this product.

You tried to use a Frame product, but you are not allowed to use this product. In the `frameusers` file, you are either specified in a <Disallowed> statement or your system administrator has not named you in an <Allowed> statement. See your system administrator for help.

You can only anchor to text inside a TextRect.

You are using the Anchor command to create a new anchored frame, and the insertion point is in a text line.

You can only make isolated TextRects into printer code.

You are using the Printer Code command on a TextRect that has connections to other TextRects. You must cut these connections before using the Printer Code command. See the Disconnect Head and Disconnect Tail commands in Chapter 3 for more information.

You need to start the window system before running FrameMaker.

You are trying to start FrameMaker outside of the window system, or the window system is full and there are no more windows available.

Your maintenance contract has expired.

You can't get a license to use a Frame product because the expiration date in the `frameusers` file has passed. To use this product, you must call Frame and purchase more maintenance.

Your Maker file is truncated--probably an accident when using another program.

You are using the Open command to open a file, and FrameMaker is unable to open it because the file isn't complete.

Zero copies specified.

You are using the Print command, and you have filled in the Copies box with a 0 (zero) or blank. Change the entry and try again.

Customizing FrameMaker

This appendix lists the features of FrameMaker you can customize. You customize FrameMaker by changing configuration files in a directory named `.makerinit` or by specifying FrameMaker X Window options.

Changing the .makerinit Directory

Before you change the files in the `.makerinit` directory, we suggest you make backups in case you want to restore the original versions.

When you start FrameMaker, it searches in this order for `.makerinit`:

- The current directory.
- Your home directory.
- The directory in which FrameMaker is installed (we refer to in this directory as *install_dir*).

If FrameMaker finds a configuration file in one of the directories, it won't search any further for that file. For example, if it finds a file in `.makerinit` in the current directory, it won't search *for that file* in `.makerinit` in your home directory or in the directory in which FrameMaker was installed. FrameMaker will, however, search these directories for other configuration files.

If you're a system administrator, you can change `.makerinit` in the directory in which FrameMaker was installed; these changes apply to all users who share that `.makerinit` directory (including users of FrameWriter).

If you're a user of FrameMaker, you can customize FrameMaker for your personal use by following these steps:

1. Create a `.makerinit` directory in your home directory. (You may also need to create subdirectories.)
2. Copy the file you want to change from *install_dir*/`.makerinit` into the directory you created.
3. Edit the file in this directory.
4. If FrameMaker is running, quit it and then start it again so the changes take effect.

Font List

FrameMaker determines which fonts to load at startup by reading the file `.makerinit/fontdir/fontlist`. FrameMaker is shipped with two font list files: `fontlist.lwstd` (the standard list for a LaserWriter and the default) and `fontlist.lwpstd` (the standard list for a LaserWriter Plus). Using `fminstall`, you can change the file `fontlist` so that it will be a symbolic link to either `fontlist.lwstd` or `fontlist.lwpstd`, depending on what kind of printer you specify during the installation procedure (for information on symbolic links, see the *ln* command in your UNIX manual).

You can save memory overhead and reduce startup time by making a custom `fontlist` that specifies only those fonts you actually use in your documents. To do so, copy one of the supplied font list files to `~/.makerinit/fontdir/fontlist` and remove the unwanted font file references. Then restart FrameMaker.

fmbook Help

You can change the `fmbook` help message (you see this message when you type a question mark (?) at the `fmbook` prompt). To change this message, edit the text file `help_fmbook` in `.makerinit/bookdir`.

Patterns

The 16 fill and border patterns displayed in the FrameMaker Tools window are defined in the `patterns` file in `.makerinit`. Each pattern is defined using 16 hexadecimal digits. Two hex digits (8 bits) represent one row in the pattern, and there are 8 rows. Changing these patterns affects only the screen display.

To change the patterns used for printing, you must edit the `postscript_prolog` file (search for *`fillprocs`* in the file). The standard definitions are:

```
/fillprocs
         [{0.00 grayness} <----------  Always black
          {0.10 grayness}
          {0.30 grayness}
          {0.50 grayness}
          {0.70 grayness}
          {0.90 grayness}      ------  Always white
          {0.97 grayness}    /
          {1.00 grayness} <-
          {<0F1E3C78F0E1C387> 8 1 dpi 16 div setpattern}
          {<0F87C3E1F0783C1E> 8 1 dpi 16 div setpattern}
          {<CCCCCCCCCCCCCCCC> 8 1 dpi 16 div setpattern}
          {<FFFF0000FFFF0000> 8 1 dpi 16 div setpattern}
          {<8142241818244281> 8 1 dpi 16 div setpattern}
          {<03060C183060C081> 8 1 dpi 16 div setpattern}
          {<8040201008040201> 8 1 dpi 16 div setpattern}
          {} <-----------------
          {1.00 grayness}      \
          {0.90 grayness}         Always none
          {0.70 grayness}
          {0.50 grayness}
          {0.30 grayness}
          {0.10 grayness}
          {0.03 grayness}
          {0.00 grayness}
          {<F0E1C3870F1E3C78> 8 1 dpi 16 div setpattern}
          {<F0783C1E0F87C3E1> 8 1 dpi 16 div setpattern}
          {<3333333333333333> 8 1 dpi 16 div setpattern}
          {<0000FFFF0000FFFF> 8 1 dpi 16 div setpattern}
          {<7EBDDBE7E7DBBD7E> 8 1 dpi 16 div setpattern}
          {<FCF9F3E7CF9F3F7E> 8 1 dpi 16 div setpattern}
          {<7FBFDFEFF7FBFDFE> 8 1 dpi 16 div setpattern}
          {}
         ] def
```

The first 8 patterns use the PostScript printer's built-in gray-scale mechanism: `{0.00 grayness}` produces black, `{1.00 grayness}` produces white, and the values between produce varying shades of gray. The second 8 patterns are geometric bit patterns defined using the same hex digit scheme as in the `patterns` file (the empty procedure { } is a placeholder for the None pattern). The last 16 patterns define inverted fill patterns. You can change any of the patterns in the `postscript_prolog` file except black (0 and 16), white (7 and 23), and None (15 and 31).

Tab Leaders and Decimal Tab Characters

The list of tab leaders displayed in the Tabs dialog box is read from the `tabs` file in `.makerinit`. Its standard contents are:

```
The entries in this file specify the tableaders that appear in the
tab dialog box. The numerical codes are in the range [0,255]

<DecimalTab 46> '.', use 44 for ',' tab

<TabLeader  32> ' '
<TabLeader  46> '.'
<TabLeader  45> '-'
<TabLeader  95> '_'
```

The `tabs` file also determines what character is used in decimal tabs. To change the decimal tab character to a comma, open the `.makerinit/tabs` file and change the line

```
<DecimalTab 46> '.'
```

to

```
<DecimalTab 44> ','
```

You can change the decimal tab character and the tab leader characters to other characters as well. To specify the character, look up the character in Appendix B of the *FrameMaker Reference Manual* and convert its hexadecimal code to a decimal code. Then use that decimal number in the <DecimalTab> or <TabLeader> statement.

The numbers (such as 32) are decimal character codes (the characters shown in single quotation marks are comments). See Appendix B, *Character Set*, to find other characters usable as tab leaders. Note that the appendix lists character codes in hexadecimal notation; you will have to convert those values to decimal for use in this file.

International Version Language Defaults

The default language in the international version is UKEnglish. To change FrameMaker's default language, edit the `languages` file in `.makerinit` according to the instructions at the beginning of the file.

The `languages` file contains the options for the user interface language (such as English or French) and the default language (such as Dutch, French, German, UKEnglish, or USEnglish) for all TextLines and new paragraphs in documents.

Once you have changed the default user interface language in this file, the language you've chosen will be the default when you start FrameMaker with the `imaker` command.

International Version Unit Defaults

The `units` file determines the default units used for new documents. You will find the `units` file in `.makerinit/`*language*, where *language* is the name of your default user interface language. For example, if you start the French version, FrameMaker will use the default units specified in `.makerinit/french/units`.

To change the defaults for rulers, grid spacing, and display units, edit the `units` file according to the instructions at the beginning of the file.

Note: When you choose the New command, the default units are used only in custom documents; the Frame templates already have unit settings.

International Version Template Defaults

FrameMaker finds templates by searching for a directory named `fmtemplates`. FrameMaker searches in this order for `fmtemplates`:

- The current directory.
- Your home directory.
- The directory in which FrameMaker is installed (we refer to this directory as *install_dir*).

The `fmtemplates` directory in the installation directory is actually a symbolic link to a language subdirectory in `.makerinit` such as `templates.french` or `templates.ukenglish`. The default symbolic link is to the USEnglish templates. To change the default templates, remove the symbolic link in the installation directory and create a new one. For example, if you want French templates as a default, follow these steps:

```
cd install_dir
rm fmtemplates
ln -s .makerinit/templates.french fmtemplates
```

Line Widths

FrameMaker supports four different line and border widths, which are read from the `widths` file in `.makerinit` at startup. The standard definitions in the `widths` file are:

```
<LineWidths 1 0.5>
<LineWidths 2 1.0>
<LineWidths 4 3.0>
<LineWidths 6 4.0>
```

The four LineWidth statements correspond to the four line widths shown in the Tools window, from top to bottom. For each line width, the first value specifies the width shown on the screen in pixels, and the second value is the width in points used for printing. For example, the thinnest line is displayed one pixel wide and printed 1/2 point wide. You should avoid changing the screen display widths, but you can freely change printing widths.

Arrow Drawing

The procedures used to draw arrows when printing are defined in the `postscript_prolog` file. That file allows users to produce any kind of arrow, in addition to FrameMaker's default style. It is beyond the scope of this manual to explain how to write PostScript code for generating arrows. See the arrow drawing procedure in `postscript_prolog` named `/W`.

Default Printer Name

FrameMaker prints using the UNIX *lp* utility, specifying which printer to use by name. The names understood by *lp* vary from site to site and are defined in the file called `/etc/printcap` on each user's workstation. If you have a PostScript printer at your site, the name is usually one of the following: `lw`, `ps`, `postscript`, or `PostScript`.

FrameMaker's Print dialog box allows you to specify the name of the printer to use (see the Print command in Chapter 3). The printer name (and indeed all Print dialog box settings) is stored along with the contents of a document; when you print a document, the printer name shown in the dialog box is the same as the printer name used when you last printed the document.

For new documents, the default printer name comes from the file called `printer` in `.makerinit`. For sites using a PostScript printer, the typical printer name specified in `printer` is `ps`. You can change this name to any other name corresponding to the printer you have at your site.

Default Print Spooler

When FrameMaker prints a document, it searches for the file `FMlpr` in `.makerinit`. `FMlpr` is a shell script that sends the FrameMaker print file to *lp*. You can edit `FMlpr` to filter the print file before sending it to the printer. For example, you could create a filter that adds crop marks to the document pages.

Spelling Checker Dictionaries

You can customize the site dictionary file for your site, adding and deleting words as you see fit. The site dictionary is a text file that FrameMaker reads at startup. You can add words to the site dictionary using a standard text editor or FrameMaker (be sure to save the dictionary as Text Only).

At startup, FrameMaker searches for the name of the site dictionary in the `dictionaries` file in `.makerinit`. By default, FrameMaker searches for the site dictionary in the file `site.dict` in *install_dir*/`.makerinit`. To specify a different site dictionary, edit the `dictionaries` file. You can, for example, create several site dictionaries. By changing the `dictionaries` file and restarting FrameMaker, you can easily switch between them.

Marker Names

The 26 marker names that appear in the Markers window come from the file `.makerinit/`*language*`/markers`. When markers are stored in a document, their marker type is stored as a number from 0 to 25—the marker names are not significant. You should not change the first 10 marker names, however, because FrameMaker and the `fmbook` program assign special meanings to those marker names. To change the remaining marker names, edit the `markers` file with a text editor or FrameMaker (be sure to save the file as Text Only).

If your marker name uses a character with a hexadecimal code greater than 7e (many foreign language characters fall into this category), type a backslash followed by the character's hex code (the hex codes are listed in Appendix B). For example, to create a marker type called Mémoire, you need the hex code of the *é* character (\x8e). Your entry in the `Markers` file would look like this: `<M\x8emoire>`

Keyboard Mappings

When FrameMaker starts up it reads two keyboard configuration files: first `kbmap` and then `kbmacros`.

FrameMaker searches in this order for `kbmap`:

- `./.makerinit/`*language*
- `~/.makerinit/`*language*
- *install_dir*`/.makerinit/`*language*

If FrameMaker finds the file in one place, it reads the file and stops searching. To customize `kbmap` for your own needs, create a `kbmap` file in your home directory. Keep in mind, however, that the `kbmap` file must be in a language subdirectory of `.makerinit`.

The `kbmap` file ties key codes to FrameMaker function codes and character codes, which are defined in Appendix B, *Character Set,* Appendix A, *Keyboard Commands*, and in the file `.makerinit/edit.h`. For example, if the `kbmap` file contains the line `<a : b>`, FrameMaker displays the letter *b* when you type a lowercase *a*. A more practical example is `<\!r : \x384>`, which causes the key sequence Esc r to invoke the Reshape command (`\!` represents the Esc key and `\x384` is the function code for the Reshape command). The `kbmap` file contains more detailed documentation on its syntax and use.

The `kbmacros` file ties key codes to other key codes, creating keyboard macros. FrameMaker searches in this order for `kbmacros`:

- The current directory.
- Your home directory.
- *install_dir*/`.makerinit`

See the Record Keys and Keyboard commands for more information.

Mouse Timing Controls

You can customize two aspects of mouse timing: the double-click speed and the autoscroll speed. Both of these controls are described below.

Double-clicking is a shortcut for highlighting a word in text (see *Text Editing Overview* in Chapter 2 for instructions). FrameMaker recognizes a double-click by noting the elapsed time between each click (as well as the distance traveled by the arrow pointer). If FrameMaker receives a mouse click and the elapsed time and movement since the previous mouse click is small, FrameMaker assumes you've double-clicked. The maximum time interval that separates double-clicks is defined in the file named `mouse` in `.makerinit`. The standard entry is:

```
              seconds      microseconds
<DoubleClick    0             250000    >
<AutoScroll     0             75000    >
```

FrameMaker creates the timing speeds by adding the values in the two columns (250000 microseconds is one-fourth of a second; 75000 microseconds is less than one-tenth of a second).

If FrameMaker treats a double-click as if it were two single clicks, increase the time interval specified in the `mouse` file. If you highlight words when you didn't mean to, on the other hand, decrease the time interval.

Also defined in the `mouse` file is the autoscroll speed. This value applies only to scroll lists. Specifically, when you point on the scroll arrows to the right of a scroll list and press and hold down the mouse button, the items in the list scroll automatically. The autoscroll speed sets the speed with which the items in the list scroll (the smaller the value, the faster they scroll).

Program Messages

You can change the messages FrameMaker displays in its alert boxes, messages that appear in the console window, and even the text of FrameMaker's dialog boxes. Also, you can change special icons FrameMaker uses. In general, you'll make these changes only if you're customizing FrameMaker for a different language.

For your reference, here are the special directories containing FrameMaker's messages and dialog box icons (FrameMaker searches for them in the *install_dir*/`.makerinit` directory only):

This directory:	**Contains files defining:**
`.makerinit/`*language*`/srre`	Alert boxes and console messages
`.makerinit/dbdir`	Special dialog box icons
`.makerinit/`*language*`/dbre`	FrameMaker dialog boxes
`.makerinit/`*language*`/icondir`	FrameMaker icons

Image Cache Size

When you open a document containing imported image files (this includes both bitmap and EPS files), FrameMaker doesn't read all the images into memory. Instead, FrameMaker reads an image only when you display or print a page containing the image or when you scale it within FrameMaker. In the process, it removes from memory those images it no longer needs.

When FrameMaker starts, one memory buffer is allocated to hold the screen representation of imported image files. The buffer is also used to hold screen images of anchored frames at certain times. The size of this buffer is read from the `bitmaps` file in `.makerinit` at startup. The standard setting provides for up to 128 images in memory at once, using a 200,000-byte buffer size. If you do not use imported image files and anchored frames, you can reduce the size of this buffer to save some memory space (this is *not* recommended). On the other hand, if you're editing documents with many imported images, you can improve FrameMaker's performance by using a larger buffer size.

Templates for Opening Text Files

When you open standard text files whose name does not end in `.txt`, FrameMaker searches for a template document to determine what page size and formatting values to use. The default template document is called `ASCIITemplate.doc`. FrameMaker searches in this order for `ASCIITemplate.doc`:

- The directory in which FrameMaker was started.
- Your home directory.
- The `.makerinit` directory.

The standard `ASCIITemplate.doc` file uses a page size and font that are similar to a standard ShellTool window. To set up a different template, use the New command to create an empty document with the desired page size and number of columns and save the document using the name `ASCIITemplate.doc` in a `.makerinit` directory. You can also put headers and footers, a Paragraph Catalog, and master page graphics in the template document, but you should leave the TextRect(s) on page 1 empty. You can highlight the end-of-flow symbol in the

TextRect at the start of page 1 and set its font, tab settings, and other paragraph formatting values.

You can also set up a more powerful template system using filename suffixes. FrameMaker does not search for `ASCIITemplate.doc` unless it cannot find *xxx*`Template.doc`, where *xxx* is the suffix of the text file being opened. For example, you can create a template file called `cncTemplate.doc`. That template will then be used automatically by FrameMaker whenever you open a standard text file whose suffix is `.cnc`.

FrameMaker searches for *xxx*`Template.doc` files in the same places that it searches for `ASCIITemplate.doc`.

Filters for Import, Open, and Save

FrameMaker provides a mechanism for filtering, reading, and writing foreign files. A *foreign file* is one that is not a FrameMaker document file, an image file (bitmap or EPS), or a MIF file.

When you open or import a foreign file, FrameMaker searches the file called `suffixlist` in `.makerinit` for a filename suffix that matches the foreign file's suffix. If it does not find a matching suffix, it opens the file as a standard text file. If it does find a matching suffix, it starts up the shell script in `.makerinit` called `MifRead`. `MifRead`'s job is to read the foreign file and produce a corresponding temporary MIF file. When `MifRead` completes, FrameMaker reads and then deletes the temporary file.

The standard `MifRead` includes at least two filters, `txttomif` and `mmltomif`:

```
#! /bin/sh
# Shell file to convert various formats to Mif
#
# Edit as needed to execute other filters;
# The cases below should match the contents of file .makerinit/suffixlist
#
# Two filters are provided with standard Maker: One reads files with the
# suffix ".txt" and converts them to mif, attempting to recognize paragraphs.
# The second reads and converts MML files. See the Reference Manual for
# more information.
#
# Inputs are:
# $1 = input file to be filtered (full path, example: /usr/drf/foo.bar )
# $2 = output file from filter   (full path, example: /usr/drf/foo.tmp )
# $3 = input file's path         (example: /usr/drf         )
# $4 = .makerinit's directory
# $5 = maker's bin directory
#
# See what the file type is, and run the appropriate program
case $1 in
        *.txt) $5/txttomif <$1 >$2;;
        *.mml) $5/mmltomif -I$3 -I~ -I$4 <$1 >$2;;
esac
```

The C language source code file for `txttomif` is in the `source` directory on the FrameMaker distribution tape. Use this as a starting point for writing your own filters.

FrameMaker provides a similar filtering mechanism when saving a file in MIF format. It generates the MIF output and then executes `MifWrite`. You can use a case statement similar to the one in `MifRead` to run the MIF file through the appropriate filter based on the goal filename's extension.

Changing FrameMaker X Window System Options

For the most part, X Window system options control the way FrameMaker windows appear on your screen. In general, you can specify X Window system options either as part of the command you use to start FrameMaker (on the command line) or in an X Window system resource file. Exceptions are noted in the following option descriptions.

When you start FrameMaker, it searches the following places, in order, for X Window system options:

- The command line.
- The resource line or resource file listed on the command line.
- The resource file listed in the shell variable `XENVIRONMENT`, or, if there is no such variable, the file `~/.Xdefaults-`*display*`/maker` (where *display* is the

name of the server, for example, `~/.Xdefaults-MyWorkstation/maker`).

- The server's resources, or, if it has none, the `.Xdefaults` file.
- The file `/usr/lib/X11/app-defaults/maker`.
- The FrameMaker program's defaults.

In this section, italics represent variables for which you must substitute an actual value. For example, when a window's location is given as +*x*+*y*, you specify a location by substituting actual coordinates for *x* and *y*. We refer to the directory in which you installed FrameMaker as *install_dir*.

Changing Options on the Command Line

You can specify FrameMaker X Window system options by including them in the command that you use to start FrameMaker. To specify an option, type **`maker`**, a space, a minus sign, the option name, a space, and the option status. Begin each additional option with a space and a hyphen. For example, the following command starts FrameMaker using black for the background color of windows and white for the foreground (text) color:

```
% maker -background black -foreground white
```

Options that you specify on the command line take precedence over options that you have stored in a resource file. You can use a resource file to specify the way you want FrameMaker to operate normally and use command-line options for special cases.

Creating Resource Files

An X Window system resource file is a collection of application program options. It consists of one or more resource lines. Each resource line has one of the following formats:

*application_name*option*: *status*
application_name.option: *status*

The line contains the application name, an asterisk or a period, the option name, a colon, one or more spaces, and the option status. If you omit the application name, the resource line applies to all applications and you must begin the resource line with an asterisk. Do not type a minus sign before the option name. To continue a resource line onto a second line, type a backslash (\) at the end of the first line. Resource lines that describe FrameMaker options use **`maker`** as the application name. For example, the following resource line sets the position of the main FrameMaker window:

```
maker*mgeometry: +200+50
```

For more information on resource files, see your window system documentation.

Using Resource Files

There are three command-line options you can use to specify resources: `-xrm`, `-xrf`, and `-name`.

The `-xrm` "*resource_line*" option lets you include a resource line on the command line. For example:

```
% maker -xrm "maker*background: black"
```

The `-xrf` *file_name* option lets you specify a resource file in which FrameMaker X Window options have been stored. FrameMaker starts up using the options in the resource file you named. For example:

```
% maker -xrf ~/MyResourceFile
```

You can save different sets of FrameMaker options in different resource files and use the `-xrf` option to specify a particular resource file when you start Frame-Maker, or you can assign different application names in the same resource file and use the `-name` option when you start FrameMaker. For example, if you normally use FrameMaker with black text on a white background but sometimes like to use white text on a black background, your resource file might contain the following lines:

```
b_on_wMaker*background: white
b_on_wMaker*foreground: black
w_on_bMaker*background: black
w_on_bMaker*foreground: white
```

When you start FrameMaker, use the `-name` option to specify one of the two names. For example, to use white-on-black windows, type:

```
% maker -name w_on_bMaker
```

Specifying a Display Workstation

If you are running on a remote system, use the `-display` *host*:*server#*.*screen#* option to specify the name of the workstation on which to display FrameMaker. The *host* is the workstation on which to display FrameMaker, *server#* is the number of the server on which the program is running, and *screen#* is the screen number. You can use `-d` as an abbreviation for `-display`, and you can only use this option on the command line. FrameMaker looks for this information in the following locations in order:

- On the command line.
- As part of an `-xrm` *resource_line*.
- In the resource file specified with the `-xrf` option.
- In the shell variable `DISPLAY`.

Message Window Options

When you start FrameMaker, it normally creates an xterm window to use for displaying messages. You have three options for customizing this message window. You can start FrameMaker without a message window, with an iconic message window, or with a message window that has a custom size and position.

The `-noconsole` option instructs FrameMaker not to create a new window. Instead, it prints messages in the shell window in which you entered the **`maker`** command. For example:

```
% maker -noconsole
```

The `-noconsole` option is only available as part of the command you give to start FrameMaker. FrameMaker does not read it from a resource file.

The `-iconic` option starts FrameMaker with an iconic message window, for example:

```
% maker -iconic
```

The `-iconic` option is only available as part of the command you give to start FrameMaker. FrameMaker does not read it from a resource file.

The `-geometry` *w*x*h*+*x*+*y* option lets you specify the size and location of the message window. *w* is the window's width in characters, *h* is it's height in characters, *x* is its x-coordinate in pixels, and y is its *y*-coordinate in pixels. For example, the following command creates a message window in the upper left corner of your display (at the point (0,0)). It is 40 characters wide and 10 characters high.

```
%  maker -geometry 40x10+0+0
```

You can specify the `-geometry` *w*x*h*+*x*+*y* option in a resource file. However, you must use **`fmconsole`** as the application name. For example, the following resource file line also creates a 40-character wide and 10-character high message window in the upper left corner of your display:

```
fmconsole*geometry: 40x10+0+0
```

Background and Foreground Colors

Use the -background *color* (or -bg *color*) option to specify the background color of FrameMaker windows, and use the `-foreground` *color* (or `-fg` *color*) option to name the foreground (text) and border color for FrameMaker windows. The default background color is white, and the default foreground color is black. You must use contrasting colors. The following command starts FrameMaker using windows with a black background and a white foreground:

```
% maker -bg black -fg white
```

See your window system documentation for a list of colors you can use.

File

Use the `-file` *file_name* (or `-f` *file_name*) option to name a file for FrameMaker to open as it starts up. For example, to start FrameMaker with the file `MyLetter.doc` open, use the following command:

```
% maker -file MyLetter.doc
```

Static Color Displays

FrameMaker requires that foreground and background color pixel values differ by only one bit (or plane). Monochrome displays already provide this default, and you can assign appropriate colors on PseudoColor displays. On StaticColor displays, however, the chosen colors (or the default black and white) may not have appropriate values, and FrameMaker will respond with this error message:

```
maker: Cannot allocate colors. Aborting.
```

On these displays, you must select two colors from the colormap which vary by only one bit. You must use these colors as FrameMaker foreground and background colors.

Menu Bar

Use the `-menubar` option to turn on and off the display of the FrameMaker menu bar. When the menu bar is turned off, you cannot display FrameMaker menus, though you can still use keyboard shortcuts. The menu bar is turned on by default. To turn off the menu bar, use the following command:

```
% maker -menubar
```

To turn on the menu bar, use the following command:

```
% maker +menubar
```

Main Menu Location

Use the `-mgeometry` *+x+y* (or `-mg` *+x+y*) option to specify the location of FrameMaker's main window. For example, to position the menu at the point (0,0), use the following command:

```
% maker -mg +0+0
```

Menu Fonts

Use the `-menufont` *font_name* option to specify a font for FrameMaker menus. You can use any X Window system font. The default menu font does not support special German characters. So, if you are using the German version of FrameMaker, you should always choose a different menu font.

If you are using release 2 of the X Window system, the names of files in the directory `/usr/lib/X11/fonts` are valid font names for your system. If you are using release 3 of the X Window system, the names of files in the directory `/usr/lib/X11/fonts/`*resolution*`dpi/fonts.dir` are valid font names (*resolution* is the font resolution in dots per inch). To examine fonts, use the `xfd` program. See your window system documentation for more information on fonts and the `xfd` program.

When font names are very long, you can abbreviate them by using an asterisk to represent the part of the name that is not unique. For example, the following command changes the menu font to a font whose name ends with the characters `98-iso8859-1`:

```
% maker -menufont *98-iso8859-1
```

Title Text

Use the `-title` *text* option to specify a name to display in the title bar of the main FrameMaker window. (If your window system does not display title bars, this option is ignored.) For example, to display `My_Custom_FrameMaker` in the title bar, enter the following command:

```
% maker -title My_Custom_FrameMaker
```

This option is useful when you are running two versions of FrameMaker. Use it to give each main window a different title.

X Window System Options Summary

The following table summarizes FrameMaker's X Window system options:

Option	Use
`-background` *color*	Specifies a background color for FrameMaker windows
`-display` *host:server#.screen#*	Names a display workstation when you are running FrameMaker from a remote server
`-file` *file_name*	Names a file to open as FrameMaker starts
`-foreground` *color*	Specifies a foreground color for FrameMaker windows
`-menubar`	Displays FrameMaker windows without the menu bar
`+menubar`	Displays the menu bar in FrameMaker windows (default)
`-mgeometry` *+x+y*	Specifies the location of the main window
`-menufont` *font_name*	Specifies a font for FrameMaker menus
`-title` *text*	Specifies a title for the main window

`-geometry` *w*x*h*+*x*+*y*	Specifies the size and location of the FrameMaker message window
`-iconic`	Starts FrameMaker with an iconic message window
`-noconsole`	Starts FrameMaker without a message window
`-xrf` "*file_name*"	Specifies a resource file from which to obtain FrameMaker X Window system options
`-xrm` "*resource_line*"	Specifies an X Window system option in resource-line format

Hypertext Documents

With FrameMaker or FrameWriter, you can create locked documents (including hypertext documents) for viewing only. Using hypertext commands and the lock command, you can write documents such as on-line manuals and on-line help. Then you can view these documents in FrameMaker, FrameWriter, or FrameViewer. FrameViewer lets you look at locked documents; if you have a FrameViewer license, you don't need a FrameMaker or FrameWriter license to browse through locked documents.

When a document is locked, all menus except one (a pop-up Document Window menu) disappear; users can move through the document either by paging forward and backward or by clicking on hypertext links which you create before locking the document. With hypertext links, you can create links to information in the same document or in different documents. (See "Creating a hypertext document," later in this appendix, for a description of hypertext and how it works.)

This document explains how to:

- Lock and unlock documents
- Create hypertext documents
- Browse through a locked document
- Start FrameViewer

Locking and unlocking documents

When a document is locked, a user can only view it, using the commands in the pop-up Document Window menu or the hypertext links in the document. Before you lock a document, you should turn off the display of the grid, the ruler, text symbols, and borders and deselect everything. A locked document will be displayed exactly as it appeared when you locked it except that the menu bar and status line won't appear.

When you unlock a document, it is like any other FrameMaker document. You can add and edit hypertext links, but the links are inactive while the document is unlocked.

One command acts as a toggle, locking or unlocking a document. Make sure the cursor is in the document window; then type

```
Esc D l k
```

This command is case-sensitive, so be sure to type a capital *D* and then a lowercase *l* and *k*.

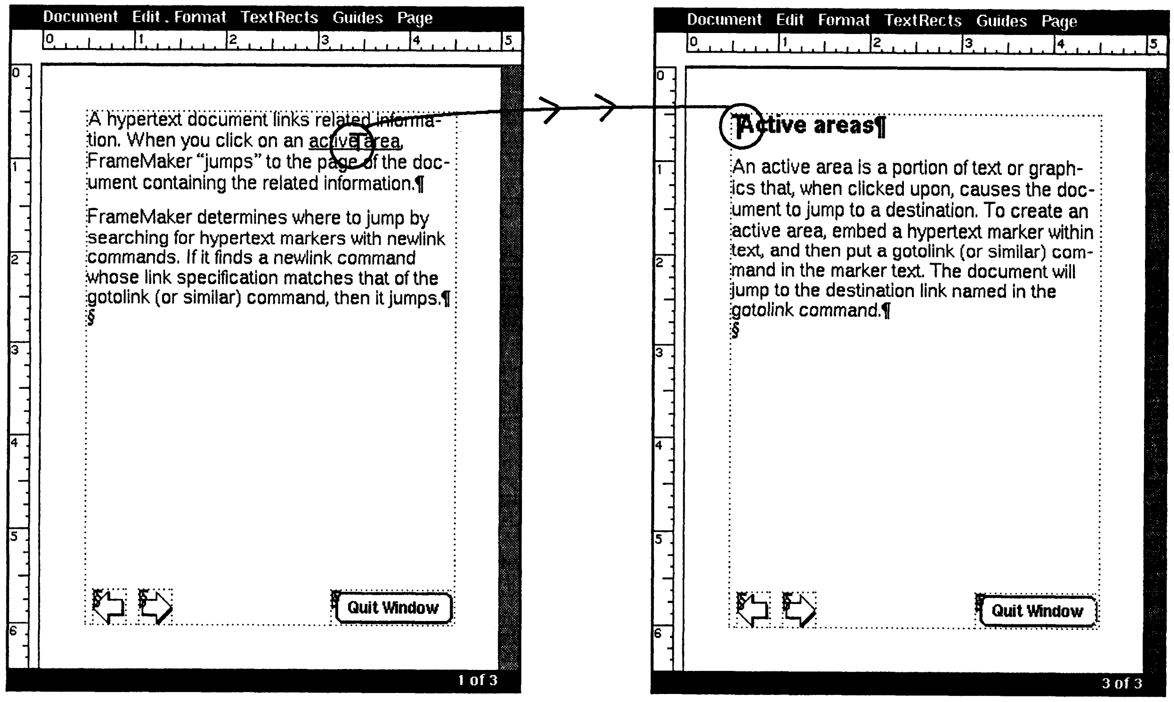
Creating a hypertext document

In our discussion of hypertext, we will use the following terms:

Term:	**Definition:**
link	An association between pieces of information that can span documents. When you click on the link's *active area*, you
jump to	the link's *destination.*
active area	The origin of a link; where users click to *jump* to the related information. A font change (or a whole paragraph with no font change) signifies the beginning and end of an active area.
destination	The end of a link; where users *jump* when they click on an active area.
jump	Move from the active area to the related information.

How hypertext works

Hypertext refers to the linking of related information in an arbitrary way. In a hypertext document, you create links between information so that when users click on, say, a word for more information, the display will jump to the page or document containing the explanation. Our on-line help, for example, contains links; when users want more explanation or details, they can click underlined text or icons, and the page containing the linked information is displayed. Here is an example of a hypertext jump within a document:


Document Edit Format TextRects Guides Page
Quit Window
1 of 3
Document Edit Format TextRects Guides Page
Quit Window
3 of 3

A hypertext document links related informa-tion. When you click on an active area, FrameMaker "jumps" to the page of the doc-ument containing the related information.¶

FrameMaker determines where to jump by searching for hypertext markers with newlink commands. If it finds a newlink command whose link specification matches that of the gotolink (or similar) command, then it jumps.¶
§

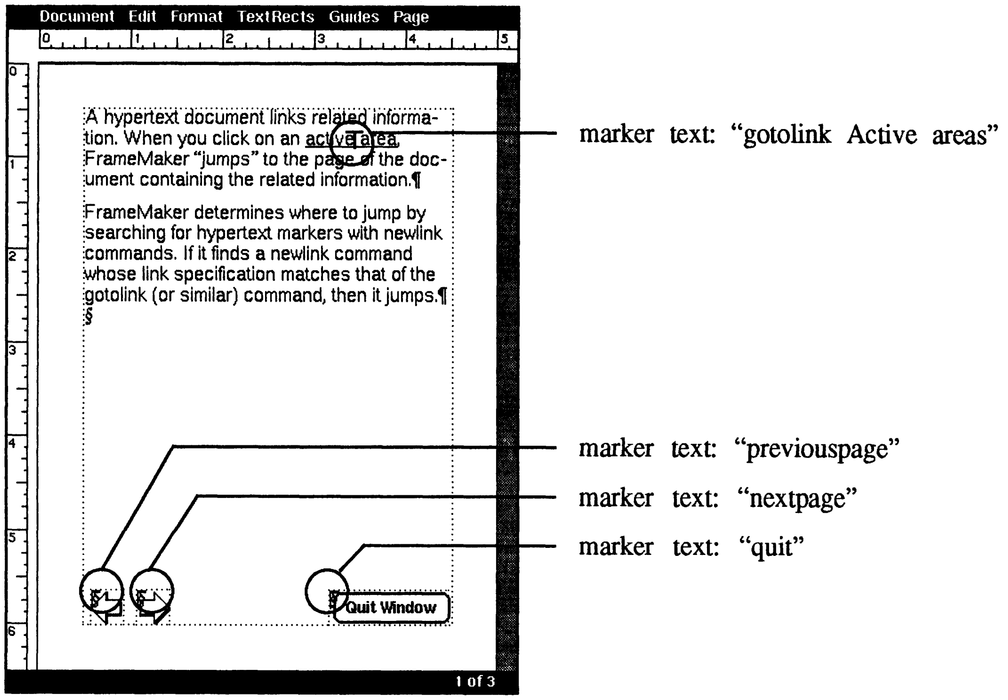
Active areas¶

An active area is a portion of text or graph-ics that, when clicked upon, causes the doc-ument to jump to a destination. To create an active area, embed a hypertext marker within text, and then put a gotolink (or similar) com-mand in the marker text. The document will jump to the destination link named in the gotolink command.¶
§

You can also use hypertext links, as we do in our on-line help icons, to move sequentially through a document, to return to a link's origin, or even to quit documents.

Creating links

To plot the course of links, you insert hypertext markers within text or over graphics that are the origins and destinations of jumps. Then you embed special hypertext commands within the marker text. These commands are described in the section titled "Hypertext marker commands," later in this document. Here is an example of a document with hypertext markers embedded in it:


Document Edit Format TextRects Guides Page
marker text: "gotolink Active areas"
marker text: "previouspage"
marker text: "nextpage"
marker text: "quit"
Quit Window
1 of 3

A hypertext document links related information. When you click on an active area, FrameMaker "jumps" to the page of the document containing the related information.¶

FrameMaker determines where to jump by searching for hypertext markers with newlink commands. If it finds a newlink command whose link specification matches that of the gotolink (or similar) command, then it jumps.¶
§

Once you have placed markers with hypertext commands and locked the document, the user can jump to related information by clicking the active area.

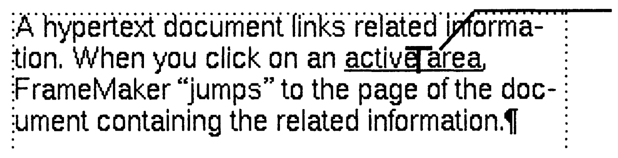
Creating active areas in text

You can place active areas within text or over graphics. For instructions on creating active areas in graphics, see the next section, "Creating active areas in graphics."

To create an active area in text:

1. Select the text you want to activate and change its font.

   In our on-line help, we underline the active areas in text. By changing the font, you let users know where they should click for more information. But the font change is also necessary for the jumps to work properly because font changes define the boundaries of the active area. When the user clicks on text, the program searches forward and backward from the click point for font changes. (If there are no font changes in the paragraph the user clicked on, then the whole paragraph is considered the active area.)


A hypertext document links related information. When you click on an active area, FrameMaker "jumps" to the page of the document containing the related information.¶

user clicks here

The program searches backward and detects a font change in the space before the "a" in "active"; it searches forward and finds a font change after the word "area." Using the font changes as a guide, the program determines that the underlined text is the active area.

2. Insert a hypertext marker within the active area's boundaries.

   Once the program has determined the active area, it will search the area for a hypertext marker and execute the hypertext command in the first marker it finds. (See "Inserting a hypertext marker" and "Hypertext marker commands," later in this appendix, for more information about hypertext markers.)

Creating active areas in graphics

You may want to place an active area in graphics so that when users click on any part, or a specific part, of a graphic, they will jump to a destination. Since markers must be inserted in text, however, you must use a TextRect to make an active area in a graphic:

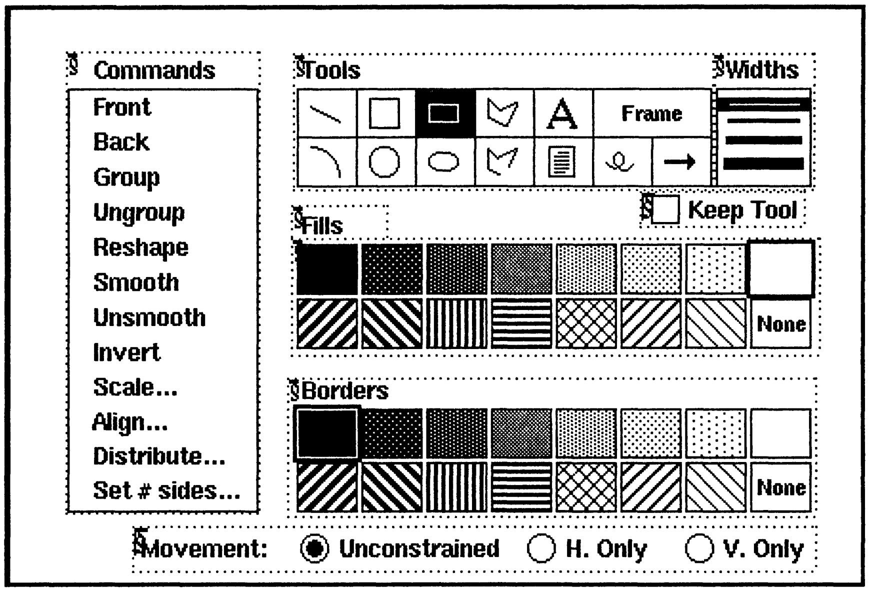
1. Choose the TextRect tool from the Tools window and draw a TextRect with fill "None" over the portion you want to make active.
2. Put the insertion point in the TextRect and insert a hypertext marker. (See the next section, "Inserting a hypertext marker," for more information.)

   Since there is no actual text in the TextRect, the marker will be inserted at the upper left corner of the TextRect, just in front of the end-of-flow symbol.

Once you have placed a TextRect over the active area, the entire area of the TextRect will be active. (In this case, a whole paragraph with no font change signifies the active area.)

Note: To edit a marker or check the marker text, you need to select the marker. But it is usually difficult to see and select the marker at the beginning of a TextRect. To select the marker, click with the middle mouse button in the TextRect, and drag the mouse to the left and up.

Example: In our on-line help, we superimposed several TextRects over portions of the Tools window graphic so that users could click on specific areas for explanations. The illustration below shows the borders of the superimposed TextRects and the hypertext markers in the top left corner of each TextRect:



Inserting a hypertext marker

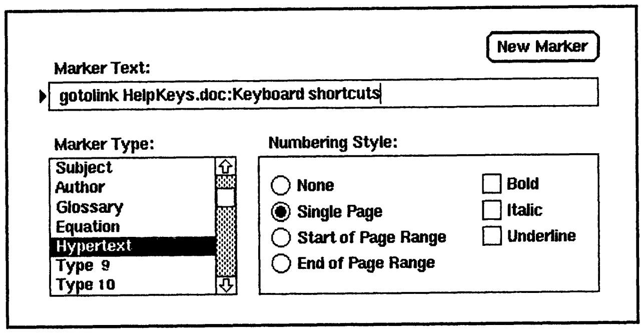
Hypertext markers identify the origins and destinations of jumps. You can chart the course of jumps by inserting hypertext markers in the appropriate places and embedding hypertext marker commands in the marker text.

To insert a hypertext marker:

1. Put the insertion point where you want to create an active area or destination.

   To create an active area within text, place the insertion point in the middle of the word or phrase so that the program will recognize the font change at the beginning and end of the active area. (See “Creating active areas in text,” earlier in this appendix.) If the marker is in a graphic, the insertion point will normally be at the top left corner of the TextRect. (See the previous section, “Creating active areas in graphics.”)
2. Choose the Markers command from the Edit menu.
3. Select “Hypertext” in the “Marker Type:” scroll list.

4. In the "Marker Text:" text box, type the appropriate hypertext command. (See "Hypertext marker commands," later in this appendix.) You can type up to 255 characters in the text box.



Marker Text: gotolink HelpKeys.doc:Keyboard shortcuts

Marker Type: Hypertext

Numbering Style: Single Page

5. Click on the "New Marker" button.

To edit the marker text, select the marker and display the Markers window. Then edit the marker text and click on the "Edit Marker" button.

Hypertext marker commands

In this section, we describe the commands used in hypertext markers to define active areas and destinations. The commands are in alphabetical order.

alert

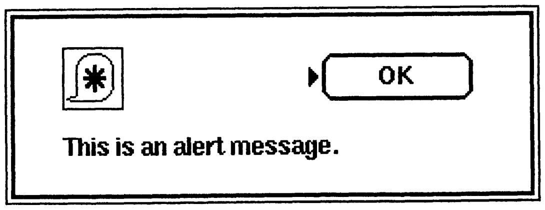
`alert` *message* displays an alert box containing a message.

The message length is limited to the number of characters that can fit on one line in the alert box. When a user clicks on a hypertext marker containing an alert command, a box containing the message is displayed in the middle of the screen. Users must then click OK before they can do anything else.

Example: The command

```
alert This is an alert message
```

will display this alert box:



This is an alert message.

gotolink

`gotolink` *linkspec* searches for and jumps to a hypertext marker containing a `newlink` command with a matching linkname.

Linkspec syntax

The `gotolink`, `newlink`, and `previouslink` commands require a link specification, abbreviated as "linkspec." A link specification is text identifying the destination of a jump—a destination in the same document or in another document. There are two forms of link specifications:

- *linkname*

  The short form of the link specification is the linkname by itself. The linkname is the name of the destination used in the `newlink` command. If you are creating a link to a document heading, we suggest you use the heading name as the linkname to simplify the maintenance of your documents. For example, to jump from some underlined text to text with the heading "Using Help," you could use this command:

  $\texttt{gotolink}\boxed{\texttt{Using Help}} \leftarrow$ linkname

  The program would search in the same document for a marker containing this command:

  ```
  newlink Using Help
  ```

  Besides using destinations linked to text or graphics, you can also use the keywords `firstpage` and `lastpage` as all or part of the link specification. `firstpage` causes the document to jump to the first page of the current document or any document explicitly named in the link specification, and `lastpage` causes the document to jump to the last page.

  Note: `firstpage` and `lastpage` are not stored on the stack, so you can't use `previouslink` to return to the original page after jumping to `firstpage` or `lastpage`. See the description of `previouslink` later in this appendix.

  Examples:

  ```
  gotolink lastpage

  previouslink firstpage
  ```

- *filename:linkname*

  The long form of a link specification consists of the filename (the name of the document in which the destination link occurs) and the linkname. The long form is necessary when the destination link is in another document. Here are two examples of commands with long link specifications:

  ```
  gotolink HelpMarkers:firstpage

  gotolink HelpKeys1.doc:Keyboard shortcuts
  ```

(Notice that you can use the keywords `firstpage` and `lastpage` in the long form, too.)

Although the `gotolink` command requires the name of the destination document if you're jumping to another document, the `newlink` command in the destination marker does not. You can use the short form of the link specification (the linkname) in the `newlink` command.

When a `gotolink` command contains a relative pathname, FrameMaker searches for the relative pathname in the following directories:

1. The directory of the document you're jumping from.
2. *startup_dir*/`.makerinit`/*language*/`helpdir`
3. ~/`.makerinit`/*language*/`helpdir`
4. *install_dir*/`.makerinit`/*language*/`helpdir`

(*startup_dir* stands for the directory in which you started FrameMaker. *language* stands for the user interface language.)

If the destination link doesn't exist, then the document jumps to the top of the current document's text flow. (If the link specification includes an existing document name, then the document jumps to the top of the document named in the link specification.)

Example: In the document `HelpIndex.doc`, we use the command

```
gotolink HelpKeys1.doc:Keyboard shortcuts
```

This command searches the document `HelpKeys1.doc` for a hypertext marker containing the command

```
newlink Keyboard shortcuts
```

When the appropriate `newlink` command is found, the program opens the document and displays it in the document window. If the link doesn't exist, then the first page of `HelpKeys1.doc` is displayed. The program jumps to the top of the current document's text flow if the named document doesn't exist.

Note: Link specifications are case-sensitive; the characters in the `newlink` command's linkname, including any trailing spaces, must match those in the `gotolink` command's linkname exactly, just as the linkname in a `previouslink` command must match that of the associated `newlink` command.

newlink

`newlink` *linkspec* defines a destination link whose linkname matches one specified in a `gotolink` command.

Example: The command

```
newlink Keyboard shortcuts
```

identifies a destination link. See the description of the `gotolink` command for an explanation of *linkspec* and examples of how the `gotolink` and `newlink` commands work together.

Note: Even if the matching `gotolink` command identifies the filename of the destination marker, the `newlink` command in the destination marker requires only the linkname, not the filename.

nextpage

`nextpage` displays the next page of the document. This command isn't stored on the stack, so if you click on a marker containing a `nextpage` command, you can't return to the original page with `previouslink`. (See the `previouslink` command.)

Example: In FrameMaker's on-line help documents, we use `nextpage` in the right-arrow icons (except on the last page of a document).

previouslink

There are two versions of the `previouslink` command: `previouslink` and `previouslink` *linkspec*.

`previouslink` jumps back to the origin of the most recent link.

The program remembers the origins of the last 12 jumps by storing them on a stack, with the most recent jump on top. You can keep jumping back until you reach the first origin marker or have jumped back 12 times. You cannot, however, return to the origin of the `previouslink` command; in other words, you cannot jump "forward."

You also can't use `previouslink` to jump back to `nextpage` or `previouspage` commands or `gotolink` commands containing `firstpage` or `lastpage`.

`previouslink` *linkspec* jumps to the link specification only if there are no more remembered origins on the stack. Otherwise, it also jumps back to the previous link. See the description of the `gotolink` command for an explanation of *linkspec*.

Example: In our on-line help documents, we use `previouslink` in our up-arrow icons so that users can return to their previous jumping point.

previouspage

`previouspage` displays the previous page of the document. This command isn't stored on the stack, so if you click on a marker containing a `previouspage` command, you can't return to the original page with `previouslink`. (See the `previouslink` command.)

The user can also display the previous page by pressing F6 or choosing "Previous Page" from the Document Window pop-up menu.

Example: In FrameMaker's on-line help documents, we use `previouspage` in our left-arrow icons (except on the first page of the document).

quit

`quit` removes the current FrameViewer window from the screen.

This command is the same as the Quit command on the Document menu in FrameMaker. If you have edited a document and locked it without saving changes, FrameMaker or FrameWriter will prompt you to save them before quitting. (FrameViewer users will not be asked to save work since they cannot edit locked documents.)

If the user has shift-clicked on links, causing destinations to be displayed in their own windows, then the `quit` command will remove only the current window. The command `quitall` will remove all FrameViewer windows.

Example: In our on-line help documents, we use `quit` in the Quit Window button.

quitall

`quitall` removes all FrameViewer windows from the screen (including closed ones in icon form).

The user can open more than one window for a locked document (or linked documents) by shift-clicking on a link. If you have edited a document and locked it without saving changes, FrameMaker or FrameWriter will prompt you to save them before quitting. (FrameViewer users will not be asked to save work since they cannot edit locked documents.)

Example: The command

```
quitall
```

will cause FrameMaker or FrameWriter to prompt the user to save if there are unsaved changes in any FrameViewer windows, and then will remove all FrameViewer windows.

Creating a series of linked documents

When you create on-line documents, you should consider whether you want to write one long document or several shorter documents. The advantage of confining the information to one document is that the user can jump from one page to the next very quickly. A long document, however, takes longer to open.

A shorter document, on the other hand, will load much more quickly. In our on-line help, for example, we decided to link a series of help documents so that the initial document (the Help index) would open quickly. We also limited the other documents to six pages so that there would be little delay when the user clicked on an active area.

Tips on linking a series of documents

- Put the links in later.

  You may want to wait until the structure of your documents is fairly set before inserting hypertext commands. If you change filenames or the organization of your documents after you've created links, some links may become obsolete.

- Use heading names in `newlink` commands.

  If the destination of a link is a heading, use the heading name as the linkname in the `gotolink` and `newlink` commands.

- Create a book file that refers to all the documents.

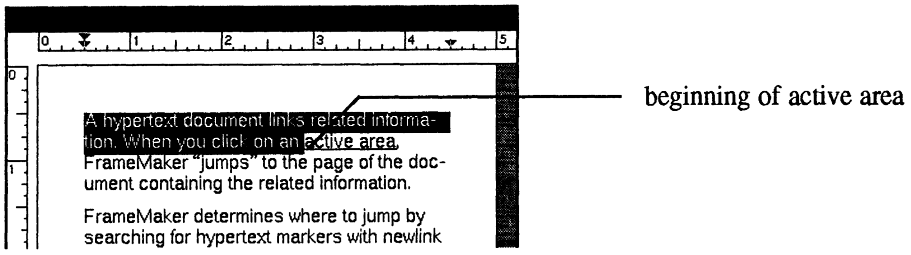
  By creating a book file comprising all your documents, you can use fmbook to print the documents and also to compile an index of all the hypertext markers in the documents. This index is useful when you debug your links; you can use it, for example, to make sure that the link specifications in `gotolink` and `newlink` commands match. For instructions on creating a book file, see Chapter 6, "The fmbook Program."

- Use Shift-Ctrl-click to check for active areas.

  If your links aren't working properly, one cause may be "hidden" font changes—a space or a period, for example, in a font different from the main text font. If a user clicks on such an area, the program will consider it an active area but won't be able to find a hypertext marker.

  You can see what the program considers an active area by holding the Shift and Ctrl keys and clicking in a paragraph with the left mouse button. If there is no font change in the paragraph, the whole paragraph will be highlighted. (Remember that the program considers a whole paragraph without font changes to be the active area.)

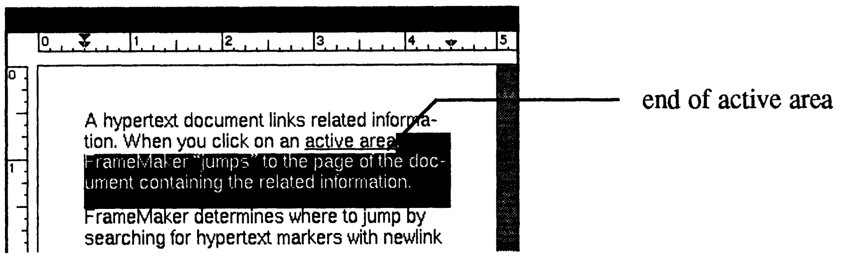
If you Shift-Ctrl-click before an active area, the program will highlight from the beginning of the paragraph to the font change, which means that the first unhighlighted character is the beginning of the active area.



A hypertext document links related information. When you click on an active area, FrameMaker "jumps" to the page of the document containing the related information.

FrameMaker determines where to jump by searching for hypertext markers with newlink

When you Shift-Ctrl-click after an active area, the program will highlight from the font change to the end of the paragraph, which means that the character in front of the highlighted passage is the end of the active area.



A hypertext document links related information. When you click on an active area, FrameMaker "jumps" to the page of the document containing the related information.

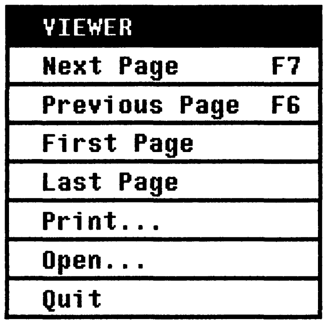
FrameMaker determines where to jump by searching for hypertext markers with newlink

And finally, if you Shift-Ctrl-click on a font change, the program will highlight the area with the changed font (in other words, what it considers the active area).

Browsing through a locked document

When you are reading a locked hypertext document, there are two ways to move through the document. You can:

- Page forward and backward with the Previous Page (F6) and Next Page (F7) commands on the pop-up Document Window menu. To display the pop-up menu, place the mouse arrow anywhere inside the document window and press the right mouse button.



- Jump from link to link by clicking with the left or middle mouse button on active areas. You can also press Shift while clicking to display the destination page or document in a new window.

Starting FrameViewer

FrameViewer is a program which allows you to view locked documents created in FrameMaker or FrameWriter. To start FrameViewer, type

`viewer -f` *filename*

at the UNIX prompt. If you are creating several on-line documents for users, you may want to write a program or shell script that starts FrameViewer for users automatically.

Adding Macintosh PostScript Fonts

PostScript printers contain built-in fonts. Typically, these include Times, Helvetica, Courier, and Symbol in a variety of styles (for example, bold, italic, and bold-italic). FrameMaker comes ready to use these fonts.

You can, however, obtain many more PostScript fonts from Adobe Systems and other vendors. You can use these fonts with FrameMaker, but you must upload them to your FrameMaker workstation and install them in FrameMaker. This appendix describes how to add such fonts to FrameMaker (you must have FrameMaker Version 1.11 or later).

If you have a LaserWriter Plus and want to configure FrameMaker to use the LaserWriter Plus fonts, you don't need this appendix. You configure FrameMaker to use these fonts when you install FrameMaker.

What you need

To complete the instructions in this appendix, you need the following hardware and software:

- A Macintosh computer and your FrameMaker workstation.
- True PostScript fonts on Macintosh disks. You can use fonts from vendors other than Adobe only if the fonts are in Adobe's format. Some font packages claim to be "Standard Macintosh Format PostScript Fonts," but aren't. This appendix won't work for these products. You have true PostScript fonts only if the font disks contain all the files described in the section *Upload the Macintosh files* later in this appendix.
- An RS-232 cable connecting the serial ports of the Macintosh and FrameMaker workstation (if the Macintosh and FrameMaker workstation are connected through a network, you don't need a serial cable).
- Communication software for transferring files between the Macintosh and the FrameMaker workstation. This software must be able to transfer *either* the Data Fork or the Resource Fork of the Macintosh file. Also, the communications software must be able to transmit binary data without changing it during transmission.
- A licensed copy of FrameMaker installed on your FrameMaker workstation.

General steps for adding fonts

To add PostScript fonts to FrameMaker, you follow these general steps:

1. Connect the serial ports of the Macintosh and the FrameMaker workstation.
2. Start the communication software you'll use to transfer the files from the Macintosh to the FrameMaker workstation.
3. Upload the files from the Macintosh to the FrameMaker workstation.

4. Using conversion programs provided with FrameMaker, convert the Macintosh files to a format FrameMaker and your printer can use.
5. Move the converted files to a specific directory beneath FrameMaker's .makerinit directory.
6. List the new fonts in a special file that FrameMaker reads when it starts.
7. If your printer doesn't have the new fonts built-in, edit the converted printer font files, and then download them to the printer. (If your printer has its own hard disk, you may be able to download printer fonts from the Macintosh directly to the printer.)

This appendix describes in detail how to convert the Macintosh files and change FrameMaker's .makerinit directory. It doesn't, however, explain how to connect the Macintosh to the FrameMaker workstation or how to use communications software to upload files. In addition, it only briefly covers the topic of downloading printer fonts.

For more information on uploading files, see the manuals accompanying your communications software. For more information on downloading printer fonts to your PostScript printer, see Adobe System's *Supporting Downloadable PostScript Fonts,* available from Adobe Technical Support.

Upload the Macintosh files

An Adobe Systems Typeface Package usually has two disks: one disk contains *printer font files,* and the other contains *screen font files* and Adobe Font Metric (or *AFM)* files. Some complex fonts may have more disks, but they still contain these three file types.

The printer font files are scalable representations of the character images that the PostScript printer needs to print the characters. Screen font files are bitmap images of the fonts at a particular point size that FrameMaker uses to display characters. The AFM files contain typographic information FrameMaker uses to decide where to break lines and place characters.

Basically, you must transfer all the data on the font disks to the workstation running FrameMaker. You can use any of the different systems available for moving data between machines (provided it meets the requirements listed in the section *What you need* earlier in this appendix).

AFM files

The AFM files are on the Printer Fonts Disk, and always have the suffix AFM. Usually, there are four AFM files, one for each font style. For example, Adobe's Garamond font package (#9) includes the following AFM files:

This AFM file:	Contains this style:
GaramLig.AFM	Light

GaramBol.AFM	Bold
GaramLigIta.AFM	Light Italic
GaramBolIta.AFM	Bold Italic

The AFM files are ASCII text files (unlike the binary printer and screen font files). Using your communications software, transfer the Data forks of the AFM files in Text mode (as opposed to Binary mode).

Because the AFM files are text files, the font installation will still work if your communications software transforms carriage returns to line feeds during transmission. Also, don't worry if the communication software changes the file names; you'll change them again yourself later.

Printer font files

If the fonts you're transferring are built into your printer, you don't need to transfer the printer font files from the Macintosh to the FrameMaker workstation—skip ahead to the *Screen font files* section. Transfer the printer font files only if you plan to download fonts from the FrameMaker workstation to your printer.

Like the AFM files, there are also usually four printer font files, one for each font style. For example, Adobe's Garamond font package includes the following printer font files:

This printer font file:	Contains this style:
GaramLig	Light
GaramBol	Bold
GaramLigIta	Light Italic
GaramBolIta	Bold Italic

Transfer the Resource Forks of these files in Binary format.

If you're using Kermit to transfer these files, be sure to set *both* sides of the communication channel to binary mode—Kermit won't detect an error if you send a file in binary mode with the receiver in text mode. If you don't set both sides to binary, you'll have problems later when you try to use the fonts.

Screen font files

There are usually two screen font files: one containing the plain style, and one containing the other styles. For Garamond, the screen font files are Garamond Plain and Garamond Other.

Transfer the Resource Fork of these files in Binary mode.

Now that you've transferred all the files, you no longer need the Macintosh. You perform the rest of the steps on the FrameMaker workstation.

Before continuing, you may want to save all the files you transferred on a permanent backup tape. You may want this backup if a new version of FrameMaker requires an installation procedure that refers back to these files.

You may also want to create separate working directories for the three kinds of files and move them before converting them. In the conversion process, you create many more files, and each must have a specific name. By putting them in separate working directories, you ease the task of checking their file names and, later, of moving them.

Find the "official" font names

As we mentioned earlier, the communications software may change the file names when it transfers the files to the FrameMaker workstation. For example, Kermit changes the file names to lowercase and removes any spaces. In the rest of this appendix, we use the lowercase file names in examples.

In the tasks that follow, you convert the files you transferred to the FrameMaker workstation into files FrameMaker and your printer can use. To convert the files, you need to know the "official" PostScript font names, which are different from the file names you've been using so far.

The official font names appear in the AFM files you transferred. Use the fgrep command to search these files for "FontName"—the font names appear immediately following this word. For example, to search the Garamond AFM files, type this command in the directory containing the AFM files:

```
fgrep FontName garam*.afm
```

The following lines appear on the screen:

Official Font Names

```
garambol.afm:FontName Garamond-Bold
garambolita.afm:FontName Garamond-BoldItalic
garamlig.afm:FontName Garamond-Light
garamligita.afm:FontName Garamond-LightItalic
```

The official font names for the Garamond family are Garamond-Bold, Garamond-BoldItalic, Garamond-Light, and Garamond-LightItalic. The names of your fonts will be similar in structure.

The capitalization and punctuation of the official font names are significant. In the steps that follow, you must use the exact form of the official font names—these are the only names the PostScript printer recognizes. If you type them incorrectly, the installation procedure may appear to work, but FrameMaker documents that refer to the misspelled fonts won't print properly (missing fonts will print in Courier).

Convert the AFM files

The first files to convert to FrameMaker format are the AFM files. To convert these files, you use the program fmAfm2bfm, which converts AFM files into more compact *bfm* files (we call them bfm files because they have the suffix .bfm). This program, along with the other programs referenced in this chapter, are stored in install_dir/bin. This program takes two parameters, as shown here:

```
fmAfm2bfm      AFMFileToConvert    OfficialFontname.bfm
```

For example, to convert the Garamond family files, you type the following commands:

```
fmAfm2bfm      garamlig.afm        Garamond-Light.bfm
fmAfm2bfm      garamligita.afm     Garamond-LightItalic.bfm
fmAfm2bfm      garambol.afm        Garamond-Bold.bfm
fmAfm2bfm      garambolita.afm     Garamond-BoldItalic.bfm
```

No error messages should appear during the conversion.

Convert the screen font files

Converting the screen font files is a two-stage process. First you run the fmScreen2mfonts program to read the screen font files and write out a number of individual Macintosh font files. Then using another conversion program, you convert each of the Macintosh font files into FrameMaker *bfont* files.

Create individual Macintosh font files

Use the fmScreen2mfonts program to create the individual Macintosh font files. This program takes a single parameter, as shown here:

```
fmScreen2mfonts FontFileToConvert
```

For example, to convert the Garamond screen font files, you first type this command:

```
fmScreen2mfonts garamondplain
```

This produces these Macintosh font files:

```
Garamond9.mfont
Garamond10.mfont
Garamond11.mfont
Garamond12.mfont
Garamond14.mfont
Garamond18.mfont
Garamond24.mfont
```

To convert the other Garamond screen font file, you type this command:

```
fmScreen2mfonts garamondother
```

This produces these Macintosh font files:

```
B_Garamond_Bold9.mfont
B_Garamond_Bold10.mfont
B_Garamond_Bold11.mfont
B_Garamond_Bold12.mfont
B_Garamond_Bold14.mfont
B_Garamond_Bold18.mfont
B_Garamond_Bold24.mfont
I_Garamond_LightItalic9.mfont
I_Garamond_LightItalic10.mfont
I_Garamond_LightItalic11.mfont
I_Garamond_LightItalic12.mfont
I_Garamond_LightItalic14.mfont
I_Garamond_LightItalic18.mfont
I_Garamond_LightItalic24.mfont
BI_Garamond_BoldItalic9.mfont
BI_Garamond_BoldItalic10.mfont
BI_Garamond_BoldItalic11.mfont
BI_Garamond_BoldItalic12.mfont
BI_Garamond_BoldItalic14.mfont
BI_Garamond_BoldItalic18.mfont
BI_Garamond_BoldItalic24.mfont
```

With families other than Garamond, you might get different point sizes. Notice that the Macintosh font file names are similar to—but not exactly the same as—the official font names.

Convert the Macintosh font files

Now you must convert the Macintosh font files into FrameMaker bfont files. For each Macintosh font file, you run the program fmMfont2bfont, again using the official font name in the resulting bfont file name. This time, however, you also include the point sizes as part of the resulting file names.

For example, to convert the Garamond Macintosh font files, you type these commands:

```
fmMfont2bfont  Garamond9.mfont   Garamond-Light9.bfont
fmMfont2bfont  Garamond10.mfont  Garamond-Light10.bfont
fmMfont2bfont  Garamond11.mfont  Garamond-Light11.bfont
fmMfont2bfont  Garamond12.mfont  Garamond-Light12.bfont
fmMfont2bfont  Garamond14.mfont  Garamond-Light14.bfont
fmMfont2bfont  Garamond18.mfont  Garamond-Light18.bfont
fmMfont2bfont  Garamond24.mfont  Garamond-Light24.bfont

fmMfont2bfont  B_Garamond9_Bold.mfont    Garamond-Bold9.bfont
fmMfont2bfont  B_Garamond10_Bold.mfont   Garamond-Bold10.bfont
fmMfont2bfont  B_Garamond11_Bold.mfont   Garamond-Bold11.bfont
fmMfont2bfont  B_Garamond12_Bold.mfont   Garamond-Bold12.bfont
fmMfont2bfont  B_Garamond14_Bold.mfont   Garamond-Bold14.bfont
fmMfont2bfont  B_Garamond18_Bold.mfont   Garamond-Bold18.bfont
fmMfont2bfont  B_Garamond24_Bold.mfont   Garamond-Bold24.bfont

fmMfont2bfont I_Garamond_LightItalic9.mfont  Garamond-LightItalic9.bfont
fmMfont2bfont I_Garamond_LightItalic10.mfont Garamond-LightItalic10.bfont
fmMfont2bfont I_Garamond_LightItalic11.mfont Garamond-LightItalic11.bfont
fmMfont2bfont I_Garamond_LightItalic12.mfont Garamond-LightItalic12.bfont
fmMfont2bfont I_Garamond_LightItalic14.mfont Garamond-LightItalic14.bfont
fmMfont2bfont I_Garamond_LightItalic18.mfont Garamond-LightItalic18.bfont
fmMfont2bfont I_Garamond_LightItalic14.mfont Garamond-LightItalic24.bfont

fmMfont2bfont BI_Garamond_BoldItalic9.mfont  Garamond-BoldItalic9.bfont
fmMfont2bfont BI_Garamond_BoldItalic10.mfont Garamond-BoldItalic10.bfont
fmMfont2bfont BI_Garamond_BoldItalic11.mfont Garamond-BoldItalic11.bfont
fmMfont2bfont BI_Garamond_BoldItalic12.mfont Garamond-BoldItalic12.bfont
fmMfont2bfont BI_Garamond_BoldItalic14.mfont Garamond-BoldItalic14.bfont
fmMfont2bfont BI_Garamond_BoldItalic18.mfont Garamond-BoldItalic18.bfont
fmMfont2bfont BI_Garamond_BoldItalic24.mfont Garamond-BoldItalic24.bfont
```

Again, you should see no error messages when you convert the files.

Create downloadable printer fonts

You don't need to create downloadable printer fonts if either of the following is true:

- Your printer has the new fonts built-in.
- Your printer has its own hard disk. In this case, you can probably connect a Macintosh directly to the printer and download the fonts. For more information, see the manual that came with your printer or Adobe's *Supporting Downloadable PostScript Fonts.*

If your printer doesn't have the fonts built-in, however, follow the steps in this section to create printer font files you can download to your printer.

Creating downloadable printer fonts is a two-stage process. First you run the fmAdobeMacFont program to convert the printer font files you transferred. Then you edit the resulting files with a text editor, adding a PostScript command to them.

Convert the printer font files

Use the fmAdobeMacFont program to convert the printer font files. As you did with earlier conversion programs, you use the official font name in a parameter, but add an suffix of .ps, as shown here:

```
fmAdobeMacFont   FileToConvert    OfficialFontname.ps
```

For example, to convert the Garamond printer files, type these commands:

```
fmAdobeMacFont   garamlig         Garamond-Light.ps
fmAdobeMacFont   garamligita      Garamond-LightItalic.ps
fmAdobeMacFont   garambol         Garamond-Bold.ps
fmAdobeMacFont   garambolita      Garamond-BoldItalic.ps
```

Again, you should see no error messages when you convert the files.

Now check the font names in the *ps* files to make sure they match the font names in the AFM files. Use the fgrep command to search these files for "FontName" as you did earlier when finding the official font names.

For example, to search the Garamond ps files, type this command in the directory containing the files:

```
fgrep FontName Garamond*.ps
```

The following lines appear on the screen:

Official Font Names

```
Garamond-Bold.ps:/FontName /Garamond-Bold def
Garamond-BoldItalic.ps:/FontName /Garamond-BoldItalic def
Garamond-Light.ps:/FontName /Garamond-Light def
Garamond-LightItalic.ps:/FontName /Garamond-LightItalic def
```

Disregard the slashes (/) before the names. Notice the font names match the font names in the AFM files, which is necessary for the system to work.

Edit the ps files

Before you can download the ps files, you must change them slightly with a text editor. To create a downloadable ps file, you delete some text from the first line in the file and you add a line of PostScript commands near the top of the file.

First, delete the text from the first line in the file. You should delete all the text from the exclamation point to the end of the line. For example, here's the first few lines of the Garamond-BoldItalic.ps file, showing the text to delete.

Text to delete

```
%!PS-AdobeFont-1.0: Garamond-BoldItalic 001.002
%%CreationDate: Fri Jul 10 13:20:15 1987
%%VMusage: 38216 44487
% ITC Garamond is a registered trademark of International Typeface
% Corporation.
```

Now add the line of PostScript commands after the comments at the top of the file (PostScript comments begin with the percent sign [%]). Here's the line you add:

Printer Password

```
serverdict begin 0 exitserver
```

Your system administrator may have set a password on your printer that will keep you from downloading the ps files unless you replace the "0" with it. For information on the printer password, see your system administrator or the manual that came with your printer. For information on the PostScript commands, see Adobe's *Supporting Downloadable PostScript Fonts* and *PostScript Language Reference Manual.*

The following example shows the first few lines of the Garamond-BoldItalic.ps file after we deleted the text from the first line and added the PostScript command:

PostScript comments

```
%!
%%CreationDate: Fri Jul 10 13:20:15 1987
%%VMusage: 38216 44487
% ITC Garamond is a registered trademark of International Typeface
% Corporation.
serverdict begin 0 exitserver
11 dict begin
```

PostScript command you add

After adding this PostScript command, you can download the files to your printer. For general information on downloading fonts, see *Download fonts to the printer* later in this appendix.

Change the .makerinit directory

Now you're ready to "tell" FrameMaker about the new fonts. To do this, you put the bfont and bfm files into the proper .makerinit directory. Then you add their names to a special file (called the fontlist file) that FrameMaker reads when it starts.

Choose the directory to change

First, determine the .makerinit directory you want to change. If you're the system administrator adding fonts for all FrameMaker users at a site, change the main .makerinit directory (you probably need to be logged in as root to change this directory).

If you want to install fonts for yourself only, however, make your own .makerinit directory and change it. We suggest you use the fmcopy command to copy the main .makerinit into your home directory (the .makerinit directory occupies about 2M of disk space). Then change this personal copy of .makerinit. When you start FrameMaker, it will read the .makerinit in your home directory instead of the main .makerinit directory.

To create a personal copy of .makerinit in your home directory, type these commands:

```
cd ~
mkdir .makerinit
fmcopy $FMHOME/.makerinit .makerinit
```

To verify that FrameMaker is reading the correct .makerinit directory, quit FrameMaker, and start it again. Watch the Console window—FrameMaker lists the name of the .makerinit directory it is reading when it starts. For FrameMaker to recognize your changes, the .makerinit directory you change must be listed in the Console window.

For the rest of this appendix, we use /usr/maker/.makerinit in the examples. You, of course, should type the name of the .makerinit directory you are changing.

Move the bfm and bfont files

Now move the bfm and bfont files from your work directory to the .makerinit/fontdir.postscript directory. For example, to move the Garamond bfm and bfont files, type these commands:

```
mv Garamond*.bfm    /usr/maker/.makerinit/fontdir.postscript
mv Garamond*.bfont  /usr/maker/.makerinit/fontdir.postscript
```

If you're adding fonts for multiple FrameMaker users, make sure the permissions on the bfm and bfont files allow the users to read them. If you want, you can now remove the *.afm and *.mfont files from your work directory, since you no longer need them.

Edit the fontlist file

When starting, FrameMaker reads the fontlist file to determine what fonts to load. The fontlist file lists the names of all the bfont and bfm files comprising a font. To use the new bfont and bfm files, you must list their file names in this file. The fontlist file is located in the .makerinit/fontdir.postscript directory.

Before changing the fontlist file, make a copy of the original file in case you want to return to it. Type these commands:

```
cd /usr/maker/.makerinit/fontdir.postscript
cp fontlist fontlist.bu
```

Since the fontlist file is a standard ASCII text file, you can edit it with any standard text editor.

The fontlist has three major sections. First is the list of FontFamily entries, followed by the ScreenFont entries and the FontMetric entries (these sections, however, can actually be in any order). In the FontFamily section, you type a font family name and a Family ID number; in the ScreenFont and FontMetric sections, you list all the family's bfont and bfm files.

Add the font family name

The FontFamily section, which is typically at the top of the fontlist file, looks something like this (there may be more font families than shown below):

Font Family Name

```
<FontFamily  Courier    0>
<FontFamily  Times      1>
<FontFamily  Helvetica  2>
<FontFamily  Symbol     3>
```

Family ID

Following the syntax of the font families already listed, add the family name and Family ID. The family name is the name that appears in FrameMaker's scroll lists. You are free to use any name you want. It can't contain spaces, however, and it should be short enough to fit in the scroll lists.

The Family ID is a number you use in the other sections of the fontlist file to "tie" the family to its bfont and bfm files. You can use any small integer here (a good choice is the next available Family ID).

For example, to add Garamond to the standard family list, add this line to the end of the FontFamily list:

```
<FontFamily  Garamond  4>
```

List each screen font file

The ScreenFont section is typically the next section in the fontlist file. In this section, you list each bfont file you created, along with the Family ID, font style, and point size. For example, here are the lines you would add for Garamond:

```
                  bfont file name            Family ID  font style  point size
                     \                            \         |         /
                      ↘                            ↘        ↓        ↙
<ScreenFont   Garamond-Light9.bfont             4       0       9>
<ScreenFont   Garamond-Light10.bfont            4       0       10>
<ScreenFont   Garamond-Light12.bfont            4       0       12>
<ScreenFont   Garamond-Light14.bfont            4       0       14>
<ScreenFont   Garamond-Light18.bfont            4       0       18>
<ScreenFont   Garamond-Light24.bfont            4       0       24>
<ScreenFont   Garamond-Bold9.bfont              4       1       9>
<ScreenFont   Garamond-Bold10.bfont             4       1       10>
<ScreenFont   Garamond-Bold12.bfont             4       1       12>
<ScreenFont   Garamond-Bold14.bfont             4       1       14>
<ScreenFont   Garamond-Bold18.bfont             4       1       18>
<ScreenFont   Garamond-Bold24.bfont             4       1       24>
<ScreenFont   Garamond-LightItalic9.bfont       4       2       9>
<ScreenFont   Garamond-LightItalic10.bfont      4       2       10>
<ScreenFont   Garamond-LightItalic12.bfont      4       2       12>
<ScreenFont   Garamond-LightItalic14.bfont      4       2       14>
<ScreenFont   Garamond-LightItalic18.bfont      4       2       18>
<ScreenFont   Garamond-LightItalic24.bfont      4       2       24>
<ScreenFont   Garamond-BoldItalic9.bfont        4       3       9>
<ScreenFont   Garamond-BoldItalic10.bfont       4       3       10>
<ScreenFont   Garamond-BoldItalic12.bfont       4       3       12>
<ScreenFont   Garamond-BoldItalic14.bfont       4       3       14>
<ScreenFont   Garamond-BoldItalic18.bfont       4       3       18>
<ScreenFont   Garamond-BoldItalic24.bfont       4       3       24>
```

The bfont file name is the exact name of the bfont file. The Family ID is the number you used in the FontFamily section (the Family ID is the only thing tying the bfont files to the family). The font style is a number between 0 and 3, defining the style as shown in the following table.

Use this number:	**For this style:**
0	Plain
1	Bold
2	Italic, Slanted, or Oblique
3	Bold-Italic, Bold-Slanted, or Bold-Oblique

The point size, of course, is the point size of the font.

List each printer font file

Finally comes the FontMetric section, which is quite similar to the ScreenFont section in its layout. The only difference is that it lists bfm files, which are independent of font size. For this reason, several entries may refer to a single bfm file. You must still specify each entry so it matches the corresponding entry in the ScreenFont list. For example, here are the lines you would add for Garamond:

```
                 bfont file name          Family ID  font style  point size
                       \                       \          |          /
<FontMetric   Garamond-Light.bfm               4     0      9>
<FontMetric   Garamond-Light.bfm               4     0      10>
<FontMetric   Garamond-Light.bfm               4     0      12>
<FontMetric   Garamond-Light.bfm               4     0      14>
<FontMetric   Garamond-Light.bfm               4     0      18>
<FontMetric   Garamond-Light.bfm               4     0      24>
<FontMetric   Garamond-Bold.bfm                4     1      9>
<FontMetric   Garamond-Bold.bfm                4     1      10>
<FontMetric   Garamond-Bold.bfm                4     1      12>
<FontMetric   Garamond-Bold.bfm                4     1      14>
<FontMetric   Garamond-Bold.bfm                4     1      18>
<FontMetric   Garamond-Bold.bfm                4     1      24>
<FontMetric   Garamond-LightItalic.bfm         4     2      9>
<FontMetric   Garamond-LightItalic.bfm         4     2      10>
<FontMetric   Garamond-LightItalic.bfm         4     2      12>
<FontMetric   Garamond-LightItalic.bfm         4     2      14>
<FontMetric   Garamond-LightItalic.bfm         4     2      18>
<FontMetric   Garamond-LightItalic.bfm         4     2      24>
<FontMetric   Garamond-BoldItalic.bfm          4     3      9>
<FontMetric   Garamond-BoldItalic.bfm          4     3      10>
<FontMetric   Garamond-BoldItalic.bfm          4     3      12>
<FontMetric   Garamond-BoldItalic.bfm          4     3      14>
<FontMetric   Garamond-BoldItalic.bfm          4     3      18>
<FontMetric   Garamond-BoldItalic.bfm          4     3      24>
```

For FrameMaker to use a particular font (meaning a particular size and style in a family), you must put an entry for that font in both the FontMetric and ScreenFont sections of the fontlist file. For example, if an entry for Garamond-Bold 9 point appears only in the FontMetric section, FrameMaker displays a "Font not available" message when you try to use the font.

Start FrameMaker

After changing the fontlist file, you're ready to see if you can use the new fonts. Start FrameMaker and watch the Console window; FrameMaker should use the .makerinit directory to which you added the fonts. Open a new document and display the font dialog box—the font families you added should appear in the scroll list. Type some text and see if you can change it to the various sizes and styles you added to the fontlist.

If your printer has the new fonts built-in, test to make sure you can print using the new font. Create a test document that uses all the sizes and styles of the new font, and then print it.

If your printer doesn't have the new fonts built-in, you must download the ps files you created before you can print a document using the new fonts (see the next section for more information).

Download fonts to the printer

This section briefly describes how to download fonts. For detailed information, see Adobe's *Supporting Downloadable PostScript Fonts*.

Some PostScript printers have long-term storage (typically a hard disk) where you can store new fonts permanently. With this type of printer, you have to download the font only once, and the printer then accesses the font when needed. In addition, you may not even need to upload the printer font files from the Macintosh to the FrameMaker workstation, because you can connect the Macintosh directly to the printer to download the fonts. For more information on adding fonts to this type of printer, see the instruction manuals that accompany the printer and the Adobe font box.

Other PostScript printers (notably the LaserWriter) have no long-term storage. When you download fonts to this type of printer, they remain in the printer's memory only until the you turn the printer off.

To download fonts to the printer, you send the ps files you edited to the printer using the lp command. For example, to download the Garamond-Light font to the printer, type this command:

```
lp Garamond-Light.ps
```

The printer now "knows" Garamond-Light, and will print any document containing this font correctly. When you turn the printer off, however, Garamond-Light is no longer resident in the printer.

One disadvantage of downloading fonts to this type or printer is that you have to keep track of when the printer is turned off so you can repeat the lp command. Also, these printers tend to have a limited amount of memory in which to store downloaded fonts; you may run out of memory after downloading more than few fonts.

Ideally, you would want your fonts to be downloaded for you whenever you used them in a document, and then have them removed from the printer's memory when the document is printed. Currently, however, Adobe's Transcript spooling software doesn't handle this for you.

You can, if you want, perform these functions yourself: When you print your document, turn on the Print Only to File option on the Print dialog. This creates a

PostScript file describing your document. Then use a text editor to insert the *unedited* ps file created by fmAdobeMacFont directly into the PostScript file. Put the ps file right after the first few lines of the PostScript file that begin with %%. If you print a document this way, it may take longer than normal because PostScript font definitions are usually large.

Adding new PostScript font sizes

Currently, you can't use font sizes in FrameMaker that aren't available as screen fonts. You can, however, use a font scaling program on the screen font files to create new sizes. While the resulting screen fonts may not have look nice on the screen, they look fine when you print the document.

One such font scaling program is Fontastic, which runs on the Macintosh. Fontastic reads an Adobe screen font set, scales one of the sizes to create a new font size, and writes a font set that includes the new size.

After using Fontastic (or a different scaling program), you upload the resulting file to the FrameMaker workstation just as you do regular screen font files. The steps for converting it also remain the same. When you edit the fontlist file, however, be sure to add the new font sizes, or FrameMaker won't recognize them.

Adding IBM PC PostScript Fonts

PostScript printers contain built-in fonts. Typically, these include Times, Helvetica, Courier, and Symbol in a variety of styles (for example, bold, italic, and bold-italic). FrameMaker comes ready to use these fonts.

You can, however, obtain many more PostScript fonts from Adobe Systems and other vendors. You can use these fonts with FrameMaker, but you must upload them to your FrameMaker workstation and install them in FrameMaker. This appendix describes how to add such fonts to FrameMaker.

If you have a LaserWriter Plus and want to configure FrameMaker to use the LaserWriter Plus fonts, you don't need this appendix. You configure FrameMaker to use these fonts when you install FrameMaker.

What you need

To complete the instructions in this appendix, you need the following hardware and software:

- An IBM PC or compatible computer and a FrameMaker workstation.
- True PostScript fonts on PC disks. You can use fonts from vendors other than Adobe only if the fonts are in Adobe's format. Some font packages claim to be "Standard PC Format PostScript Fonts," but aren't. This appendix won't work for these products. You have true PostScript fonts only if the font disks contain all the files described in the section *Upload the PC files* later in this appendix.
- An RS-232 cable connecting the serial ports of the PC and FrameMaker workstation (if the PC and FrameMaker workstation are connected through a network, you don't need a serial cable).
- Communication software for transferring files between the PC and the FrameMaker workstation. This software must be able to transmit binary data without changing it during transmission.
- A licensed copy of FrameMaker or FrameWriter *Version 1.3* installed on your FrameMaker workstation. Version 1.3 includes two conversion programs necessary to add PostScript files to FrameMaker: fmAbf2bfont and fmAdobePCFont.

General steps for adding fonts

To add PostScript fonts to FrameMaker, you follow these general steps:

1. Connect the serial ports of the PC and the FrameMaker workstation.
2. Start the communication software you'll use to transfer the files from the PC to the FrameMaker workstation.
3. Upload the files from the PC to the FrameMaker workstation.

4. Using conversion programs provided with FrameMaker, convert the PC files to a format FrameMaker and your printer can use.
5. Move the converted files to a specific directory beneath FrameMaker's .makerinit directory.
6. List the new fonts in a special file that FrameMaker reads when it starts.
7. If your printer doesn't have the new fonts built-in, edit the converted printer font files, and then download them to the printer. (If your printer has its own hard disk, you may be able to download printer fonts from the PC directly to the printer.)

This appendix describes in detail how to convert the PC files and change FrameMaker's .makerinit directory. It doesn't, however, explain how to connect the PC to the FrameMaker workstation or how to use communications software to upload files. In addition, it only briefly covers the topic of downloading printer fonts.

For more information on uploading files, see the manuals accompanying your communications software. For more information on downloading printer fonts to your PostScript printer, see Adobe System's *Supporting Downloadable PostScript Fonts,* available from Adobe Technical Support.

Upload the PC files

An Adobe Systems PC Typeface Package usually has four disks: Disk 1 contains *printer font files,* and Disk 3 (Supplemental Disk A) contains *screen font (or ABF) files* and Adobe Font Metric (or *AFM)* files. Some complex fonts may have more disks, but they still contain these three file types.

The printer font files are scalable representations of the character images that the PostScript printer needs to print the characters. Screen font files are bitmap images of the fonts at a particular point size that FrameMaker uses to display characters. The AFM files contain typographic information FrameMaker uses to decide where to break lines and place characters.

Basically, you must transfer all the data on the font disks to the workstation running FrameMaker.

You can use any of the different systems available for moving data between machines, provided it can transfer the data *in binary format* without altering it.

If you use Kermit to transfer the files, be sure to set *both* sides of the communication channel to binary mode—Kermit won't detect an error if you send a file in binary mode with the receiver in text mode. If you don't set both sides to binary, you'll have problems later when you try to use the fonts.

AFM files

The AFM files are on the disk labeled Supplemental Disk A, and always have the extension AFM. Usually, there are four AFM files, one for each font style. For example, Adobe's Palatino font package (#1) includes the following AFM files:

This AFM file:	Contains this style:
`POR______.AFM`	Light
`POB______.AFM`	Bold
`POI______.AFM`	Light Italic
`POBI_____.AFM`	Bold Italic

Underscore characters

Transfer these files in binary format. Don't worry if the communication software changes the filenames; you'll change them again yourself later.

Printer font files

If the fonts you're transferring are built into your printer, you don't need to transfer the printer font files from the PC to the FrameMaker workstation—skip ahead to the *Screen font files* section. Transfer the printer font files only if you plan to download fonts from the FrameMaker workstation to your printer.

Like the AFM files, there are also usually four printer font files, one for each font style. For example, Adobe's Palatino font package includes the following printer font files:

This printer font file:	Contains this style:
`POR______.PFB`	Light
`POB______.PFB`	Bold
`POI______.PFB`	Light Italic
`POBI_____.PFB`	Bold Italic

Transfer these files in binary format.

Screen font files

Usually, there is one screen font file (or *ABF* file) for each font style and point size. For example, Adobe's Palatino font package includes four screen font files (one for light, bold, light italic, and bold italic) for each of the 5 point sizes. For example, the following table shows the file names for the 10-point screen font.

This screen font file:	**Contains this style and point size:**
`POR__10_.ABF`	Light, 10 point
`POB__10_.ABF`	Bold, 10 point
`POI__10_.ABF`	Light Italic, 10 point
`POBI_10_.ABF`	Bold Italic, 10 point

In binary format, transfer the ABF files for each point size and style.

Now that you've transferred all the files, you no longer need the PC. You perform the rest of the steps on the FrameMaker workstation.

Before continuing, you may want to save all the files you transferred on a permanent backup tape. You may want this backup if a new version of FrameMaker requires an installation procedure that refers back to these files.

You may also want to create separate working directories for the three kinds of files and move them before converting them. In the conversion process, you create many more files, and each must have a specific name. By putting them in separate working directories, you ease the task of checking their file names and, later, of moving them.

Find the "official" font names

The communications software may change the file names when it transfers the files to the FrameMaker workstation. For example, Kermit changes the file names to lowercase and removes any spaces. In the rest of this appendix, we use the lowercase file names in examples.

In the tasks that follow, you convert the files you transferred to the FrameMaker workstation into files FrameMaker and your printer can use. To convert the files, you need to know the "official" PostScript font names, which are different from the file names you've been using so far.

The first conversion program, fmAbf2bfont, creates files with the official font names. You use these names when running the other conversion programs.

For example, the official font names for the Palatino family are Palatino-Roman, Palatino-Bold, Palatino-Italic, and Palatino-BoldItalic. The names of your fonts will be similar in structure.

The capitalization and punctuation of the official font names are significant. In the steps that follow, you must use the exact form of the official font names—these are the only names the PostScript printer recognizes. If you type them incorrectly, the installation procedure may appear to work, but FrameMaker documents that refer to the misspelled fonts won't print properly (missing fonts will print in Courier).

Convert the screen font files

You must convert the PC screen font files into FrameMaker *bfont* files (we call them bfont files because they have the extension .bfont). For each PC font file, you run the program fmAbf2bfont, this will create a bfont file with the official font name.

For example, to convert the Palatino PC font files, you type these commands:

```
fmAbf2bfont    por__10_.abf
fmAbf2bfont    por__12_.abf
fmAbf2bfont    por__14_.abf
fmAbf2bfont    por__18_.abf
fmAbf2bfont    por__24_.abf

fmAbf2bfont    pob__10_.abf
fmAbf2bfont    pob__12_.abf
fmAbf2bfont    pob__14_.abf
fmAbf2bfont    pob__18_.abf
fmAbf2bfont    pob__24_.abf

fmAbf2bfont    poi__10_.abf
fmAbf2bfont    poi__12_.abf
fmAbf2bfont    poi__14_.abf
fmAbf2bfont    poi__18_.abf
fmAbf2bfont    poi__24_.abf

fmAbf2bfont    pobi_10_.abf
fmAbf2bfont    pobi_12_.abf
fmAbf2bfont    pobi_14_.abf
fmAbf2bfont    pobi_18_.abf
fmAbf2bfont    pobi_24_.abf
```

When you convert a file, you should see the official font family name followed by the point size and the extension `.bfont`. For example:

```
fmAbf2bfont    por_10.abf                 You type
Creating Palatino-Italic10.bfont          You see
         Official font name
```

When you convert files from some font families (for example, Garamond), the point size isn't part of the file name. For these files, you must move the file to the appropriate name using the UNIX *mv* command. For example:

```
     mv    Garamond-Italic.bfont    Garamond-Italic10.bfont
```

Convert the AFM files

The second files to convert to FrameMaker format are the AFM files. To convert these files, you use the program fmAfm2bfm, which converts AFM files into more compact *bfm* files (we call them bfm files because they have the extension .bfm). This program takes two parameters, as shown here:

```
fmAfm2bfm        AFMFileToConvert OfficialFontname.bfm
```

For example, to convert the Palatino family files, you type the following commands:

```
fmAfm2bfm        por_____.afm   Palatino-Roman.bfm
fmAfm2bfm        pob_____.afm   Palatino-Bold.bfm
fmAfm2bfm        poi_____.afm   Palatino-Italic.bfm
fmAfm2bfm        pobi____.afm   Palatino-BoldItalic.bfm
```

No error messages should appear during the conversion.

Create downloadable printer fonts

You don't need to create downloadable printer fonts if either of the following is true:

- Your printer has the new fonts built-in.
- Your printer has its own hard disk. In this case, you can probably connect a PC directly to the printer and download the fonts. For more information, see the manual that came with your printer or Adobe's *Supporting Downloadable PostScript Fonts.*

If your printer doesn't have the fonts built-in, however, follow the steps in this section to create printer font files you can download to your printer.

Creating downloadable printer fonts is a two-stage process. First you run the fmAdobePCFont program to convert the printer font files you transferred. Then you edit the resulting files with a text editor, adding a PostScript command to them.

Convert the printer font files

Use the fmAdobePCFont program to convert the printer font files. As you did with earlier conversion programs, you use the official font name in a parameter, but add an extension of .ps, as shown here:

```
fmAdobePCFont  FileToConvert  OfficialFontname.ps
```

For example, to convert the Palatino printer files, type these commands:

```
fmAdobePCFont  por______.pfb  Palatino-Roman.ps
fmAdobePCFont  pob______.pfb  Palatino-Bold.ps
fmAdobePCFont  poi______.pfb  Palatino-Italic.ps
fmAdobePCFont  pobi____.pfb   Palatino-BoldItalic.ps
```

Again, you should see no error messages when you convert the files.

Now check the font names in the *ps* files to make sure they match the font names in the AFM files. Use the fgrep command to search these files for “FontName”.

For example, to search the Palatino ps files, type this command in the directory containing the files:

```
fgrep FontName Palatino*.ps
```

The following lines appear on the screen:

Official Font Names

```
Palatino-Roman.ps:/FontName /Palatino-Roman def
Palatino-Bold.ps:/FontName /Palatino-Bold def
Palatino-Italic.ps:/FontName /Palatino-Italic def
Palatino-BoldItalic.ps:/FontName /Palatino-BoldItalic def
```

Disregard the slashes (/) before the names. Notice the font names match the font names that you saw when creating the bfont files, which is necessary for the system to work.

Edit the ps files

Before you can download the ps files, you must change them slightly with a text editor. To create a downloadable ps file, you delete some text from the first line in the file and you add a line of PostScript commands near the top of the file.

First, delete the text from the first line in the file. You should delete all the text from the exclamation point to the end of the line. For example, here's the first few lines of the Palatino-BoldItalic.ps file, showing the text to delete.

Text to delete

```
%!PS-AdobeFont-1.0
%%CreationDate: Thu Mar 12 16:47:51 PST 1987
%%VMusage: 39037 47835
% Copyright (c) 1981 Allied Corporation.  Copyright (c) 1985, 1987 Adobe
% Systems Incorporated.  All rights reserved.  This record material
% and the data recorded thereon is the property of Allied Corporation
% and Adobe Systems Incorporated, or its licensors, and may not be
% reproduced, used, displayed, modified, disclosed or transferred in
% any manner without the express written approval of Allied Corporation
% and Adobe Systems Incorporated. Palatino is a
% trademark of Allied Corporation.
```

Now add the line of PostScript commands after the comments at the top of the file (PostScript comments begin with the percent sign [%]). Here's the line you add:

Printer Password

```
serverdict begin 0 exitserver
```

Your system administrator may have set a password on your printer that will keep you from downloading the ps files unless you replace the "0" with it. For information on the printer password, see your system administrator or the manual that came with your printer. For information on the PostScript commands, see Adobe's *Supporting Downloadable PostScript Fonts* and *PostScript Language Reference Manual.*

The following example shows the first few lines of the Palatino-BoldItalic.ps file after we deleted the text from the first line and added the PostScript command:

PostScript comments

```
%!
%%CreationDate: Thu Mar 12 16:47:51 PST 1987
%%VMusage: 39037 47835
% Copyright (c) 1981 Allied Corporation.  Copyright (c) 1985, 1987 Adobe
% Systems Incorporated.  All rights reserved.  This record material
% and the data recorded thereon is the property of Allied Corporation
% and Adobe Systems Incorporated, or its licensors, and may not be
% reproduced, used, displayed, modified, disclosed or transferred in
% any manner without the express written approval of Allied Corporation
% and Adobe Systems Incorporated. Palatino is a
% trademark of Allied Corporation.
serverdict begin 0 exitserver
14 dict begin
```

PostScript command you add

After adding this PostScript command, you can download the files to your printer. For general information on downloading fonts, see *Download fonts to the printer* later in this appendix.

Change the .makerinit directory

Now you're ready to "tell" FrameMaker about the new fonts. To do this, you put the bfont and bfm files into the proper .makerinit directory. Then you add their names to a special file (called the fontlist file) that FrameMaker reads when it starts.

Choose the directory to change

First, determine the .makerinit directory you want to change. If you're the system administrator adding fonts for all FrameMaker users at a site, change the main .makerinit directory (you probably need to be logged in as root to change this directory).

If you want to install fonts for yourself only, however, make your own .makerinit directory and change it. We suggest you use the fmcopy command to copy the main .makerinit into your home directory (the .makerinit directory occupies about 2M of disk space). Then change this personal copy of .makerinit. When you start FrameMaker, it will read the .makerinit in your home directory instead of the main .makerinit directory.

To create a personal copy of .makerinit in your home directory, type these commands:

```
cd ~
mkdir .makerinit
fmcopy $FMHOME/.makerinit .makerinit
```

To verify that FrameMaker is reading the correct .makerinit directory, quit FrameMaker, and start it again. Watch the Console window—FrameMaker lists the name of the .makerinit directory it is reading when it starts. For FrameMaker to recognize your changes, the .makerinit directory you change must be listed in the Console window.

For the rest of this appendix, we use /usr/maker/.makerinit in the examples. You, of course, should type the name of the .makerinit directory you are changing.

Move the bfm and bfont files

Now move the bfm and bfont files from your work directory to the .makerinit/fontdir.postscript directory. For example, to move the Palatino bfm and bfont files, type these commands:

```
mv Palatino*.bfm /usr/maker/.makerinit/fontdir.postscript
mv Palatino*.bfont /usr/maker/.makerinit/fontdir.postscript
```

If you're adding fonts for multiple FrameMaker users, make sure the permissions on the bfm and bfont files allow the users to read them. If you want, you can now remove the *.afm and *.abf files from your work directory, since you no longer need them.

Edit the fontlist file

When starting, FrameMaker reads the fontlist file to determine what fonts to load. The fontlist file lists the names of all the bfont and bfm files comprising a font. To use the new bfont and bfm files, you must list their file names in this file. The fontlist file is located in the .makerinit/fontdir.postscript directory.

Before changing the fontlist file, make a copy of the original file in case you want to return to it. Type these commands:

```
cd /usr/maker/.makerinit/fontdir.postscript

cp fontlist fontlist.bu
```

Since the fontlist file is a standard ASCII text file, you can edit it with any standard text editor.

The fontlist has three major sections. First is the list of FontFamily entries, followed by the ScreenFont entries and the FontMetric entries (these sections, however, can actually be in any order). In the FontFamily section, you type a font family name and a Family ID number; in the ScreenFont and FontMetric sections, you list all the family's bfont and bfm files.

Add the font family name

The FontFamily section, which is typically at the top of the fontlist file, looks something like this (there may be more font families than shown below):

Font Family Name

```
<FontFamily  Courier    0>
<FontFamily  Times      1>
<FontFamily  Helvetica  2>
<FontFamily  Symbol     3>
```

Family ID

Following the syntax of the font families already listed, add the family name and Family ID. The family name is the name that appears in FrameMaker's scroll lists. You are free to use any name you want. It can't contain spaces, however, and it should be short enough to fit in the scroll lists.

The Family ID is a number you use in the other sections of the fontlist file to "tie" the family to its bfont and bfm files. You can use any small integer here (a good choice is the next available Family ID).

For example, to add Palatino to the standard family list, add this line to the end of the FontFamily list:

```
<FontFamily  Palatino  4>
```

List each screen font file

The ScreenFont section is typically the next section in the fontlist file. In this section, you list each bfont file you created, along with the Family ID, font style, and point size. For example, here are the lines you would add for Palatino:

```
             bfont file name              Family ID  font style  point size
                   ↘                           ↘         ↓          ↙
<ScreenFont  Palatino-Roman10.bfont            4         0          10>
<ScreenFont  Palatino-Roman12.bfont            4         0          12>
<ScreenFont  Palatino-Roman14.bfont            4         0          14>
<ScreenFont  Palatino-Roman18.bfont            4         0          18>
<ScreenFont  Palatino-Roman24.bfont            4         0          24>
<ScreenFont  Palatino-Bold10.bfont             4         1          10>
<ScreenFont  Palatino-Bold12.bfont             4         1          12>
<ScreenFont  Palatino-Bold14.bfont             4         1          14>
<ScreenFont  Palatino-Bold18.bfont             4         1          18>
<ScreenFont  Palatino-Bold24.bfont             4         1          24>
<ScreenFont  Palatino-Italic10.bfont           4         2          10>
<ScreenFont  Palatino-Italic12.bfont           4         2          12>
<ScreenFont  Palatino-Italic14.bfont           4         2          14>
<ScreenFont  Palatino-Italic18.bfont           4         2          18>
<ScreenFont  Palatino-Italic24.bfont           4         2          24>
<ScreenFont  Palatino-BoldItalic10.bfont       4         3          10>
<ScreenFont  Palatino-BoldItalic12.bfont       4         3          12>
<ScreenFont  Palatino-BoldItalic14.bfont       4         3          14>
<ScreenFont  Palatino-BoldItalic18.bfont       4         3          18>
<ScreenFont  Palatino-BoldItalic24.bfont       4         3          24>
```

The bfont file name is the exact name of the bfont file. The Family ID is the number you used in the FontFamily section (the Family ID is the only thing tying the bfont files to the family). The font style is a number between 0 and 3, defining the style as shown in the following table.

Use this number:	**For this style:**
0	Plain
1	Bold
2	Italic, Slanted, or Oblique
3	Bold-Italic, Bold-Slanted, or Bold-Oblique

The point size, of course, is the point size of the font.

List each printer font file

Finally comes the FontMetric section, which is quite similar to the ScreenFont section in its layout. The only difference is that it lists bfm files, which are independent of font size. For this reason, several entries may refer to a single bfm file.

You must still specify each entry so it matches the corresponding entry in the ScreenFont list. For example, here are the lines you would add for Palatino:

```
             bfont file name             Family ID  font style  point size

<FontMetric  Palatino-Roman.bfm              4         0          10>
<FontMetric  Palatino-Roman.bfm              4         0          12>
<FontMetric  Palatino-Roman.bfm              4         0          14>
<FontMetric  Palatino-Roman.bfm              4         0          18>
<FontMetric  Palatino-Roman.bfm              4         0          24>
<FontMetric  Palatino-Bold.bfm               4         1          10>
<FontMetric  Palatino-Bold.bfm               4         1          12>
<FontMetric  Palatino-Bold.bfm               4         1          14>
<FontMetric  Palatino-Bold.bfm               4         1          18>
<FontMetric  Palatino-Bold.bfm               4         1          24>
<FontMetric  Palatino-Italic.bfm             4         2          10>
<FontMetric  Palatino-Italic.bfm             4         2          12>
<FontMetric  Palatino-Italic.bfm             4         2          14>
<FontMetric  Palatino-Italic.bfm             4         2          18>
<FontMetric  Palatino-Italic.bfm             4         2          24>
<FontMetric  Palatino-BoldItalic.bfm         4         3          10>
<FontMetric  Palatino-BoldItalic.bfm         4         3          12>
<FontMetric  Palatino-BoldItalic.bfm         4         3          14>
<FontMetric  Palatino-BoldItalic.bfm         4         3          18>
<FontMetric  Palatino-BoldItalic.bfm         4         3          24>
```

For FrameMaker to use a particular font (meaning a particular size and style in a family), you must put an entry for that font in both the FontMetric and ScreenFont sections of the fontlist file. For example, if an entry for Palatino-Bold 10 point appears only in the FontMetric section, FrameMaker displays a "Font not available" message when you try to use the font.

Start FrameMaker

After changing the fontlist file, you're ready to see if you can use the new fonts. Start FrameMaker and watch the Console window; FrameMaker should use the .makerinit directory to which you added the fonts. Open a new document and display the font dialog box—the font families you added should appear in the scroll list. Type some text and see if you can change it to the various sizes and styles you added to the fontlist.

If your printer has the new fonts built-in, test to make sure you can print using the new font. Create a test document that uses all the sizes and styles of the new font, and then print it.

If your printer doesn't have the new fonts built-in, you must download the ps files you created before you can print a document using the new fonts (see the next section for more information).

Download fonts to the printer

This section briefly describes how to download fonts. For detailed information, see Adobe's *Supporting Downloadable PostScript Fonts.*

Some PostScript printers have long-term storage (typically a hard disk) where you can store new fonts permanently. With this type of printer, you have to download the font only once, and the printer then accesses the font when needed. In addition, you may not even need to upload the printer font files from the PC to the FrameMaker workstation, because you can connect the PC directly to the printer to download the fonts. For more information on adding fonts to this type of printer, see the instruction manuals that accompany the printer and the Adobe font box.

Other PostScript printers (notably the LaserWriter) have no long-term storage. When you download fonts to this type of printer, they remain in the printer's memory only until the you turn the printer off.

To download fonts to the printer, you send the ps files you edited to the printer using the lp command. For example, to download the Palatino-Roman font to the printer, type this command:

```
lp Palatino-Roman.ps
```

The printer now "knows" Palatino-Roman, and will print any document containing this font correctly. When you turn the printer off, however, Palatino-Roman is no longer resident in the printer.

One disadvantage of downloading fonts to this type of printer is that you have to keep track of when the printer is turned off so you can repeat the lp command. Also, these printers tend to have a limited amount of memory in which to store downloaded fonts; you may run out of memory after downloading more than few fonts.

Ideally, you would want your fonts to be downloaded for you whenever you used them in a document, and then have them removed from the printer's memory when the document is printed. Currently, however, Adobe's Transcript spooling software doesn't handle this for you.

You can, if you want, perform these functions yourself: When you print your document, turn on the Print Only to File option on the Print dialog. This creates a PostScript file describing your document. Then use a text editor to insert the *unedited* ps file created by fmAdobePCFont directly into the PostScript file. Put the ps file right after the first few lines of the PostScript file that begin with %%. If you print a document this way, it may take longer than normal because PostScript font definitions are usually large.

Licensing Procedures

This section provides an overview of FrameMaker's licensing policy and explains how you can control license access using the various licensing options.

Overview

When you buy FrameMaker, you purchase licenses for a number of simultaneous users; these licenses are initially *floating* licenses. Floating licenses allow anyone to use the product (as long as a license is available). For example, if you have purchased licenses for 20 FrameMaker users and 18 users are using FrameMaker simultaneously, then 2 more people can use FrameMaker. If 20 people are using FrameMaker licenses simultaneously, other users are not able to obtain a license. If you have a floating license but know you will not need to use FrameMaker for a while, you can give up your license—without quitting FrameMaker—using the License command. Then anyone waiting for a license can use the License command to obtain the available floating license.

You can keep licenses in their floating state, or you can control the use of licenses more precisely with FrameMaker's other licensing options. With these licensing options, you can:

- Reserve licenses for certain users or workstations and change these reservations without contacting Data General.
- Bar certain users or workstations from using a product.
- Obtain or return a license without exiting and restarting the application.
- Automatically give up a license if the program has been idle for the time period you specify.

You can also create separate work groups with separate control of licenses. For example, suppose you want FrameMaker licenses for two work groups—a group of 3 technical writers and a group of 10 engineers. You would like to control their use of licenses separately. To do this, you would create separate work groups.

When you first set up your site's licenses, Data General provides you with a password for the number of licenses you've purchased for each work group. After that, you're free to allocate the licenses as you wish (as long as you don't try to change the number of licenses allocated to each work group).

Most of this section is devoted to explaining how to exert more precise control over license access.

How Licensing Works

Three main components control the new licensing policy: the license server process, the license server host, and the `frameusers` file. The *license server process* is a program that monitors license access, keeping track of how many users are using a product simultaneously, how long FrameMaker has been idle on a certain machine, and more. The *license server host* is the machine on which the license server process is running. The `frameusers` file contains information used by the license server process, including information such as the host IDs; the password; the number of licenses per product; the maximum time a program can remain idle before a user loses the license; and statements identifying reserved, allowed, and disallowed users.

When you start FrameMaker, you are initially unlicensed. When you begin using FrameMaker, the license server process, which is started automatically, checks the `frameusers` file to make sure that a license is available and that you are allowed to use the program.

If you cannot get a license, FrameMaker displays the alert message "Unable to get a license" along with the reason.

If you get a license, you retain it as long as you keep using FrameMaker. If FrameMaker is idle for the maximum period of time specified in the `frameusers` file (or for the maximum idle time you specify in the License dialog box), then FrameMaker returns to its original unlicensed state.

Controlling the Use of Licenses

You can exercise finer control over license options by editing the `frameusers` file. In it, you can add statements that:

- Reserve licenses for users
- Specify which users and machines can obtain a floating license, implicitly excluding the unnamed users and machines
- Explicitly disallow specified users from using FrameMaker

Editing the frameusers File

The following example shows a `frameusers` file before it has been edited (you can find a similar example in `/usr/opt/frame1.3/.makerinit/frameusers.sample`):

```
<Server1 <HostId \x120051c7>>
<CurrentServer 1>

<Vendor
    <VendorName Data General>
    <Password 123456789>
    <FrameFinalUpdate 8/1/1988>
    <Serial 1-1-1>
    <Product
        <ProductName imaker>
        <MaxUsers 0>
        <MaxIdleTime 3600000> 1000 hours
    >
    <Product
        <ProductName maker>
        <MaxUsers 0>
        <MaxIdleTime 3600000> 1000 hours
    >
    <Product
        <ProductName viewer>
        <MaxUsers 0>
        <MaxIdleTime 3600000> 1000 hours
    >
    <Product
        <ProductName writer>
        <MaxUsers 0>
        <MaxIdleTime 3600000> 1000 hours
    >
```

A `frameusers` file contains the above basic information after you have completed the `fmlicense` script. (Blank lines are ignored, and you can put comments after right brackets.) Be sure not to change the basic licensing information, except for the <MaxIdleTime>, which defaults to 1000 hours. If you alter anything else, your password will no longer be valid, and the license server process won't run. We recommend that you make a backup copy of your `frameusers` file before editing it. That way, you'll have a version with a correct password should you accidentally alter contents that shouldn't be altered.

We recommend that you use the `fmlicense` script to change the password-related fields. But you must use a text editor or FrameMaker to edit the other fields. If you use FrameMaker to edit the `frameusers` file, be sure to save the file as Text Only.

Notice that this `frameusers` file has more than one product name in it. One `frameusers` file can control a whole network's access to all the Frame products you buy. When you edit these <Product> sections, be sure to keep the product names in alphabetical order.

To control license access for each product, you add <Reserved>, <Allowed>, and <Disallowed> statements within the brackets of each <Product> section. The next three sections describe these statements; for an example of an edited `frameusers` file, see *Sample frameusers File* later in this chapter.

Note: After you edit a `frameusers` file, you must run `fmlicense` again to start a new license server process.

Reserving Licenses

When you reserve a license, the user or machine with the reserved license is guaranteed a FrameMaker license. Use a reserved license when you want to ensure that some FrameMaker users will get a license whenever they need one. For example, suppose you have bought 10 licenses for a work group of 5 writers and 10 engineers. The writers use FrameMaker frequently and cannot afford to lose their licenses when they walk away from their work. The engineers use FrameMaker less frequently, and for shorter periods of time. In this situation, you could reserve licenses for the 5 writers and leave the 5 remaining licenses floating for the engineers. (Once you reserve a license, it's removed from the pool of available floating licenses.)

To reserve licenses, you add <Reserved> statements to a `frameusers` file's <Product> section. You can reserve a license for a user, for any user on a certain machine, or for a user on a certain machine only.

To reserve a license for the user Marie, you would use this statement:

```
        <Reserved <User marie>>
```

The name within the <User> brackets must be the user's login name.

To reserve a license for any user on the machine named paris, you would use this statement:

```
        <Reserved <Host paris>>
```

The name within the <Host> brackets must be the machine's host name.

To reserve a license for Marie on the machine paris only, you would use this statement:

```
        <Reserved <Host paris> <User marie>>
```

This constrains Marie's reserved license to the machine paris; she won't have a reserved license on another machine (unless you indicate <User marie> in another <Reserved> statement).

Reserved users may "lose" their licenses if they do not use FrameMaker for a while, but they will obtain licenses as soon as they request them. Since reserving a license reduces the number of floating licenses, the number of reserved licenses per product must be no more than the number of licenses you've purchased for that product (<MaxUsers>). In other words, if you have bought 10 FrameMaker licenses, you can reserve up to 10 of them, but no more.

Allowing Licenses

With the <Allowed> statement, you can designate explicitly all the users who are allowed to use a floating license for each product. When you use <Allowed> statements in a `frameusers` file's <Product> section, users and hosts not named in the <Allowed> or <Reserved> statements cannot obtain a license.

Use the <Allowed> statement when you want to control the use of licenses in a small work group. For example, if you have purchased 5 licenses for 10 users and you want to restrict others from trying to obtain a license, you can list the 10 users in <Allowed> statements. That way, anyone you haven't named cannot obtain a license.

The <Allowed> statement's syntax is the same as the <Reserved> statement's syntax. To allow the user Gunter a floating license, you would add this statement to the `frameusers` file:

```
<Allowed <User gunter>>
```

The name within the <User> brackets must be the user's login name.

To allow any user on the machine munich a floating license, you would add this statement to the `frameusers` file:

```
<Allowed <Host munich>>
```

The name within the <Host> brackets must be the machine's host name.

To allow the user Kenji a floating license only when he's working on the machine kyoto, you would use this statement:

```
<Allowed <Host kyoto> <User kenji>>
```

In a `frameusers` file's <Product> section, you can combine <Reserved> and <Allowed> statements; a user listed in a <Reserved> statement will always get a license when he or she requests it, and allowed users will have access to the remaining floating licenses. However, you cannot combine <Allowed> and <Disallowed> statements within a <Product> section. You can either list all the users who should have access in <Allowed> statements or list the ones who shouldn't have access in <Disallowed> statements. (For this reason, <Allowed> statements are more practical in a smaller work group, and <Disallowed> statements are more practical in a large work group. See the next section, *Disallowing Licenses*, for more information.)

Disallowing Licenses

You can use the <Disallowed> statement to bar specified users and machines from using a product. When you name a user or host in a <Disallowed> statement, that user or host cannot get a license.

Use this method of controlling licenses when you have large work groups and want to exclude only a few users. For example, if you buy 50 licenses for 100 users and want to prevent 3 of those users from obtaining licenses, you could bar the 3 users two ways. You could either list the 97 allowed users in 100 <Allowed> statements, or you could list the 3 disallowed users in <Disallowed> statements.

The <Disallowed> statement's syntax is the same as the <Reserved> and <Allowed> statements' syntax. To prevent the user Juanita from getting a floating license, you would use this statement:

```
<Disallowed <User juanita>>
```

The name within the <User> brackets must be the user's login name.

To prevent any user from getting a floating license while working on the machine kansas, you would use this statement:

```
<Disallowed <Host kansas>>
```

The name in the <Host> brackets must be the machine's host name.

To prevent the user Dorothy from obtaining a floating license on the machine kansas, you would use this statement:

```
<Disallowed <Host kansas> <User dorothy>>
```

In a `frameusers` file's <Product> section, you can combine <Reserved> and <Disallowed> statements, but you cannot combine <Allowed> and <Disallowed> statements. You can either list all the users who should have access in <Allowed> statements or list the ones who shouldn't have access in <Disallowed> statements.

Sample frameusers File

The following example depicts a `frameusers` file with the added <Reserved>, <Allowed>, and <Disallowed> statements:

```
<Server1 <HostId 17000d55>>
<CurrentServer 1>
<Vendor
        <VendorName Data General>
        <Password 524970922>
        <FrameFinalUpdate 1/1/90>
        <Serial 1-1-1>
        <Product
                <ProductName maker>
                <MaxUsers 4>
                <MaxIdleTime 3600> 1 hour
                <Reserved <Host paris> <User marie>>
                <Reserved <User jean>>
                <Allowed <User sylvie>>
                <Allowed <User jacques>>
                <Allowed <User louis>>
                <Allowed <User jeanine>>
                <Allowed <User pierre>>
        >
        <Product
                <ProductName viewer>
                <MaxUsers 10>
                <MaxIdleTime 7200> 2 hours
                <Reserved <Host roma>>
                <Disallowed <User paolo>>
                <Disallowed <User anna>>
              >
      >
```

Notice the combination of <Reserved> and <Allowed> statements for the FrameMaker section and the combination of <Reserved> and <Disallowed> statements for the FrameViewer section. No <Product> section has a combination of <Allowed> and <Disallowed> statements. In the FrameMaker section, we use 7 <Reserved> and <Allowed> statements, even though only 4 users are allowed to have licenses simultaneously. This is possible because not all floating users necessarily want a license at the same time.

After you edit the `frameusers` file, you must run `fmlicense` again to start a new license server process.

Licensing for Multiple Work Groups

If your network comprises several work groups with varying licensing needs, you may want to create a separate work group. It's easier to copy the FrameMaker files and complete a new installation process in a new installation directory. If you do not want multiple copies of FrameMaker on your network, you can create a new work group with its own license server host and `frameusers` file while sharing the installation directory with other work groups. To create a new work group, follow these steps:

1. Choose a new license server host and get its host ID. (There can be only one license server process per host.)

   Before you run `fmlicense` for the new work group, you will need a new password for the new license server host. See the *FrameMaker Release Notice* for instructions on choosing a server host and getting a new password.

2. Log in to the new license server host.

3. Change to the installation directory using this command:

   ```
   cd /usr/opt/frame1.3
   ```

4. Run `fmlicense` with a special argument: the name of a directory that everyone in the new work group can access. Be sure the directory is created before you run `fmlicense`. (To access the directory, users must have read and execute permissions for that directory, and the directory must be mounted on the users' machines.)

   For example, you might type:

   ```
   ./fmlicense /usr/local/workgroup1
   ```

   In addition to creating the new `frameusers` file, `fmlicense` puts the new `autostart` and `env.sh` scripts in this directory. (These two scripts automatically create the necessary environment variables and start the license server process. For more information, see *Information for System Administrators* later in this chapter.)

5. Put the new installation directory, `/usr/opt/frame1.3`, in the path of each user in the new work group. To do this, run `fminstall` in each user's installation directory.

6. Adjust the users' environments so the necessary environment variables are set correctly. You can do this two ways. Either:

   - Create a symbolic link to the work group's central `env.sh` file in each user's `~/.makerinit` directory. The following command, for example, creates this link:

     ```
     ln -s /usr/local/workgroup1/env.sh ~/.makerinit/env.sh
     ```

     OR

   - Edit users' `.login`, `.cshrc`, or `.profile` files, adding a line so that the `env.sh` or `env.csh` script will be sourced; the script will set the necessary environment variables in the users' environments. If your startup shell is the C shell, then add a line such as the following in users' `.cshrc` or `.login` files:

     ```
     source /usr/local/workgroup1/env.csh
     ```

If your startup shell is the Bourne shell, then add a line such as the following in `.profile`:

```
. /usr/local/workgroup1/env.sh
```

7. Edit the new `frameusers` file if you want to control who can use the new license server host. The new `frameusers` file is in the directory you specified when running `fmlicense`. For example, if you use the command:

   ```
   ./fmlicense /usr/local/workgroup1
   ```

   to run `fmlicense`, then the new `frameusers` file will be:

   ```
   /usr/local/workgroup1/frameusers
   ```

Using More than One License Server Host

If a user cannot obtain a license from one server host, he or she can try to get a license from another server host by changing the name of the license server host in the License dialog box. As long as the license server process is running on the specified server, he or she will be eligible for a license.

To prevent users from using other work groups' server hosts, you can add <Allowed> and <Disallowed> statements to each work group's `frameusers` file.

Using the Backup Server Host

The backup server host can run the server process `rpc.frameusersd` if the main server host is not usable. You should not switch to the backup server host temporarily while the main server host is rebooting; the backup server host is intended more as a long-term solution if the license server host is shut down for a while or removed from the network.

After you have installed FrameMaker and set up your `frameusers` file, we recommend that you back up the `frameusers` file on another machine besides the license server host. That way, you will have a copy of the `frameusers` file should anything happen to the server host. If you lose the `frameusers` file and don't have a backup version, you need to repeat the entire installation process, including running `fmlicense` and editing the `frameusers` file, to run a licensed version of FrameMaker.

To switch to the backup server host, follow these steps:

1. Make sure `/usr/opt/frame1.3` is mounted on the backup server host.

   Note: If the machine on which FrameMaker is installed crashes, then you must reinstall FrameMaker on another machine. We recommend that you reinstall it on the backup server host.

   If you intend to use the backup server host for a while, however, you can run `fmlicense` again, but on the backup server host this time. `fmlicense` adjusts the scripts and the `frameusers` file.

2. Copy the `frameusers` file, if it is still intact, to the new `/usr/opt/frame1.3`.

   (If it is not intact, create a new one using the `fmlicense` script.)

3. Run the `fmlicense` script in `/usr/opt/frame1.3`. The script switches the <CurrentServer> field to <CurrentServer 2> (the backup server host). It also fixes the `env.sh`, `env.csh`, and `autostart` scripts so that the values of the necessary environment variables and the path are set correctly.

   Note: If you have multiple work groups, you need to specify a directory when starting `fmlicense`. You also need to change users' environments so that the new installation directory is on their path and their startup files reflect the environment variables' new values. See *Licensing for Multiple Work Groups* earlier in this chapter for instructions.

Problems with the License Server Host and Server Process

The server process could fail to run for several reasons:

- The password in the `frameusers` file is incorrect.
- The password has expired.
- Users don't have permission to *rsh* to the license server host.
- The license server host doesn't exist.
- Another license server host on the network is using the same `frameusers` file as the user's current server host; if the `frameusers` files are identical, then both files have the same serial number, which is not allowed.
- The `frameusers` file has a syntax error. See *Controlling the Use of Licenses* earlier in this chapter for a description of the file's syntax.
- FrameMaker can't find the `frameusers` file. (This happens if the location specified in the `autostart` script is incorrect.)
- The <CurrentServer> field in the `frameusers` file is incorrect. Run `fmlicense` or edit the `frameusers` file to correct this.
- The installation directory in the `autostart` script is incorrect.
- The file `rpc.frameusersd` is missing or corrupted.
- The dates on the network's license server hosts are not close enough; the dates must be within four hours of each other.
- If the Port Mapper is not running, contact Sun Microsystems.

Problems Obtaining a License

If you cannot get a license, FrameMaker displays an explanatory message. If you receive the message "Having trouble communicating with the license server" or "The license server's protocol is not being followed," then one of the following conditions may be the problem:

- You are a disallowed user or are working on a disallowed host.
- You are an allowed user, but no floating licenses are available.
- The server process is not running on the license server host specified in the License dialog box. See the previous section, *Problems with the License Server Host and Server Process*, for possible causes.
- The date on the user's machine differs from the date on the license server host. If the date differs by more than four hours, the user will not be able to obtain a license.

Index

TIPS ORDERING PROCEDURES

TO ORDER

1. An order can be placed with the TIPS group in two ways:
   a) MAIL ORDER – Use the order form on the opposite page and fill in all requested information. Be sure to include shipping charges and local sales tax. If applicable, write in your tax exempt number in the space provided on the order form.

      Send your order form with payment to: Data General Corporation
      ATTN: Educational Services/TIPS G155
      4400 Computer Drive
      Westboro, MA 01581–9973

   b) TELEPHONE – Call TIPS at (508) 870–1600 for all orders that will be charged by credit card or paid for by purchase orders over $50.00. Operators are available from 8:30 AM to 5:00 PM EST.

METHOD OF PAYMENT

2. As a customer, you have several payment options:
   a) Purchase Order – Minimum of $50. If ordering by mail, a hard copy of the purchase order must accompany order.
   b) Check or Money Order – Make payable to Data General Corporation.
   c) Credit Card – A minimum order of $20 is required for Mastercard or Visa orders.

SHIPPING

3. To determine the charge for UPS shipping and handling, check the total quantity of units in your order and refer to the following chart:

Total Quantity	**Shipping & Handling Charge**
1–4 Units	$5.00
5–10 Units	$8.00
11–40 Units	$10.00
41–200 Units	$30.00
Over 200 Units	$100.00

If overnight or second day shipment is desired, this information should be indicated on the order form. A separate charge will be determined at time of shipment and added to your bill.

VOLUME DISCOUNTS

4. The TIPS discount schedule is based upon the total value of the order.

Order Amount	**Discount**
$1–$149.99	0%
$150–$499.99	10%
Over $500	20%

TERMS AND CONDITIONS

5. Read the TIPS terms and conditions on the reverse side of the order form carefully. These must be adhered to at all times.

DELIVERY

6. Allow at least two weeks for delivery.

RETURNS

7. Items ordered through the TIPS catalog may not be returned for credit.
8. Order discrepancies must be reported within 15 days of shipment date. Contact your TIPS Administrator at (508) 870–1600 to notify the TIPS department of any problems.

INTERNATIONAL ORDERS

9. Customers outside of the United States must obtain documentation from their local Data General Subsidiary or Representative. Any TIPS orders received by Data General U.S. Headquarters will be forwarded to the appropriate DG Subsidiary or Representative for processing.

TIPS ORDER FORM

Mail To: Data General Corporation
Attn: Educational Services/TIPS G155
4400 Computer Drive
Westboro, MA 01581 - 9973

BILL TO:	SHIP TO: (No P.O. Boxes - Complete Only If Different Address)
COMPANY NAME____	COMPANY NAME____
ATTN:____	ATTN:____
ADDRESS____	ADDRESS (NO PO BOXES)____
CITY____	CITY____
STATE____ ZIP____	STATE____ ZIP____

Priority Code ____ (See label on back of catalog)

Authorized Signature of Buyer (Agrees to terms & conditions on reverse side)	Title	Date	Phone (Area Code)	Ext.

ORDER #	QTY	DESCRIPTION	UNIT PRICE	TOTAL PRICE

A SHIPPING & HANDLING

□ UPS	ADD
1-4 Items	$ 5.00
5-10 Items	$ 8.00
11-40 Items	$ 10.00
41-200 Items	$ 30.00
200+ Items	$100.00

Check for faster delivery

Additional charge to be determined at time of shipment and added to your bill.

□ UPS Blue Label (2 day shipping)

□ Red Label (overnight shipping)

B VOLUME DISCOUNTS

Order Amount	Save
$0 - $149.99	0%
$150 - $499.99	10%
Over $500.00	20%

Tax Exempt # or Sales Tax (if applicable) ____

ORDER TOTAL	
Less Discount See B	-
SUB TOTAL	
Your local* sales tax	+
Shipping and handling - See A	+
TOTAL - See C	

C PAYMENT METHOD

□ Purchase Order Attached ($50 minimum)
P.O. number is________. (Include hardcopy P.O.)

□ Check or Money Order Enclosed

□ Visa □ MasterCard ($20 minimum on credit cards)

Account Number □□□□□□□□□□□□□□□□ Expiration Date □□□□

____
Authorized Signature
(Credit card orders without signature and expiration date cannot be processed.)

THANK YOU FOR YOUR ORDER

PRICES SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.
PLEASE ALLOW 2 WEEKS FOR DELIVERY.
NO REFUNDS NO RETURNS.

* Data General is required by law to collect applicable sales or use tax on all purchases shipped to states where DG maintains a place of business, which covers all 50 states. Please include your local taxes when determining the total value of your order. If you are uncertain about the correct tax amount, please call 508-870-1600.

Form 702
Rev. 8/87

DATA GENERAL CORPORATION TECHNICAL INFORMATION AND PUBLICATIONS SERVICE TERMS AND CONDITIONS

Data General Corporation ("DGC") provides its Technical Information and Publications Service (TIPS) solely in accordance with the following terms and conditions and more specifically to the Customer signing the Educational Services TIPS Order Form. These terms and conditions apply to all orders, telephone, telex, or mail. By accepting these products the Customer accepts and agrees to be bound by these terms and conditions.

1. CUSTOMER CERTIFICATION

Customer hereby certifies that it is the owner or lessee of the DGC equipment and/or licensee/sub-licensee of the software which is the subject matter of the publication(s) ordered hereunder.

2. TAXES

Customer shall be responsible for all taxes, including taxes paid or payable by DGC for products or services supplied under this Agreement, exclusive of taxes based on DGC's net income, unless Customer provides written proof of exemption.

3. DATA AND PROPRIETARY RIGHTS

Portions of the publications and materials supplied under this Agreement are proprietary and will be so marked. Customer shall abide by such markings. DGC retains for itself exclusively all proprietary rights (including manufacturing rights) in and to all designs, engineering details and other data pertaining to the products described in such publication. Licensed software materials are provided pursuant to the terms and conditions of the Program License Agreement (PLA) between the Customer and DGC and such PLA is made a part of and incorporated into this Agreement by reference. A copyright notice on any data by itself does not constitute or evidence a publication or public disclosure.

4. LIMITED MEDIA WARRANTY

DGC warrants the CLI Macros media, provided by DGC to the Customer under this Agreement, against physical defects for a period of ninety (90) days from the date of shipment by DGC. DGC will replace defective media at no charge to you, provided it is returned postage prepaid to DGC within the ninety (90) day warranty period. This shall be your exclusive remedy and DGC's sole obligation and liability for defective media. This limited media warranty does not apply if the media has been damaged by accident, abuse or misuse.

5. DISCLAIMER OF WARRANTY

EXCEPT FOR THE LIMITED MEDIA WARRANTY NOTED ABOVE, DGC MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE ON ANY OF THE PUBLICATIONS, CLI MACROS OR MATERIALS SUPPLIED HEREUNDER.

6. LIMITATION OF LIABILITY

A. CUSTOMER AGREES THAT DGC'S LIABILITY, IF ANY, FOR DAMAGES, INCLUDING BUT NOT LIMITED TO LIABILITY ARISING OUT OF CONTRACT, NEGLIGENCE, STRICT LIABILITY IN TORT OR WARRANTY SHALL NOT EXCEED THE CHARGES PAID BY CUSTOMER FOR THE PARTICULAR PUBLICATION OR CLI MACRO INVOLVED. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO CLAIMS FOR PERSONAL INJURY CAUSED SOLELY BY DGC'S NEGLIGENCE. OTHER THAN THE CHARGES REFERENCED HEREIN, IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO LOST PROFITS AND DAMAGES RESULTING FROM LOSS OF USE, OR LOST DATA, OR DELIVERY DELAYS, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY THEREOF; OR FOR ANY CLAIM BY ANY THIRD PARTY.

B. ANY ACTION AGAINST DGC MUST BE COMMENCED WITHIN ONE (1) YEAR AFTER THE CAUSE OF ACTION ACCRUES.

7. GENERAL

A valid contract binding upon DGC will come into being only at the time of DGC's acceptance of the referenced Educational Services Order Form. Such contract is governed by the laws of the Commonwealth of Massachusetts, excluding its conflict of law rules. Such contract is not assignable. These terms and conditions constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes all prior oral or written communications, agreements and understandings. These terms and conditions shall prevail notwithstanding any different, conflicting or additional terms and conditions which may appear on any order submitted by Customer. DGC hereby rejects all such different, conflicting, or additional terms.

8. IMPORTANT NOTICE REGARDING AOS/VS INTERNALS SERIES (ORDER #1865 & #1875)

Customer understands that information and material presented in the AOS/VS Internals Series documents may be specific to a particular revision of the product. Consequently user programs or systems based on this information and material may be revision-locked and may not function properly with prior or future revisions of the product. Therefore, Data General makes no representations as to the utility of this information and material beyond the current revision level which is the subject of the manual. Any use thereof by you or your company is at your own risk. Data General disclaims any liability arising from any such use and I and my company (Customer) hold Data General completely harmless therefrom.

134-755-02

moisten & seal

CUSTOMER DOCUMENTATION COMMENT FORM

Your Name ______________________ Your Title ______________

Company ______________________ Phone ______________

Street ____

City ______________________ State __________ Zip ______

We wrote this book for you, and we made certain assumptions about who you are and how you would use it. Your comments will help us correct our assumptions and improve the manual. Please take a few minutes to respond. Thank you.

Manual Title ______________________ Manual No. ______________

Who are you?	☐EDP/MIS Manager	☐Analyst/Programmer	☐Other ________
	☐Senior Systems Analyst	☐Operator	________
	☐Engineer	☐End User	

How do you use this manual? *(List in order: 1 = Primary Use)*

___ Introduction to the product	___ Tutorial Text	___ Other
___ Reference	___ Operating Guide	________

		Yes	No
About the manual:	Is it easy to read?	☐	☐
	Is it easy to understand?	☐	☐
	Are the topics logically organized?	☐	☐
	Is the technical information accurate?	☐	☐
	Can you easily find what you want?	☐	☐
	Does it tell you everything you need to know?	☐	☐
	Do the illustrations help you?	☐	☐

If you wish to order manuals, use the enclosed TIPS Order Form (USA only) or contact your sales representative or dealer.

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 26 WESTBORO, MA 01581

POSTAGE WILL BE PAID BY ADDRESSEE

DataGeneral

Customer Documentation
MS E-111
4400 Computer Drive
P.O. Box 4400
Westboro, MA 01581-9890

134-755-02

FrameMaker®

Reference Manual

069–100332–00

Cut here and insert in binder spine pocket

Data General

Data General Corporation, Westboro, Massachusetts 01580

069-100332-00