# Musings on Including PostScript and Macintosh Graphics in LaTeX Documents

by

Len Schwer

APTEK, Inc.
4320 Stevens Creek Blvd.
Suite 195
San Jose, CA 95129

July 1989

**Abstract**

The basics of including PostScript and Macintosh graphics into LaTeX documents are discussed. This article is intended to serve as a general overview for the person who is new to including PostScript and Macintosh graphics into LaTeX documents. Because there is no universal way to include such graphics, other than with scissors and glue, this article trys to be as general as possible, but ultimately uses and references specific software and hardware, *i.e.* Cricket Draw, ArborText's DVIPS, and an Apple's LaserWriter+. The reader is assumed to have some knowledge of the Macintosh interface, PostScript programming, and LaTeX document preparation.

# Contents

**1 Background** **3**

**2 Macintosh Graphics** **4**
2.1 Capturing Macintosh Graphics in a PostScript File . . . . . 4
2.2 Capturing Cricket Draw Graphics in a PostScript File . . . . . 6

**3 Transfering Macintosh PostScript Files to Host Machines** **8**

**4 Using DVI Drivers to Include Graphics** **9**
4.1 Background . . . . . 9
4.2 Modifying the LaserPrep File . . . . . 10
4.3 Using BBFIG and PSFIG to Include Macintosh Graphics . . . . . 12
4.4 Using ArborText's DVIPS to Include Macintosh Graphics . . . . . 15

**Appendix I: Listing of Changes to LaserPrep #65** **17**

**Appendix II: Listing of Changes to LaserPrep #68** **19**

**Appendix III: Listing of Arbortext's \PrintMacDraw Macro** **22**

**Appendix IV: Creating Multiple-Graphic Figures with Sub-Captions** **23**

# 1 Background

The concept of including Macintosh graphics in LaTeX documents, without scissors and glue, is very simple:

- Create the graphic with the user's favorite Macintosh application;
- Convert the graphic into its PostScript representation;
- Transfer the PostScript file to the LaTeX host machine;
- Include the PostScript file in the LaTeX document via the DVI-PostScript driver's `\special` command.

While this sounds like a relatively straight forward procedure, it gets complicated, sometimes very complicated. The complications arise from three elements:

1. There are several DVI-PostScript drivers available and they all treat the `\special` command differently. This is not too surprising since most of the DVI-PostScript drivers are Public Domain software, *i.e.* free. Many of the differences are simply syntactical, but some are more subtle and involve differences in the PostScript prologue which precedes the material from the LaTeX document. The good news here is that there is an organized movement within the TeX community to develop standards for DVI drivers and someday users may benefit from these standardization efforts.

2. Not all PostScript devices are the same. The numerous possible conflicts that arise when Macintosh QuickDraw, a PostScript language shorthand, is combined with various PostScript implementations of DVI drivers can and does produce different results on different PostScript devices. One example is the ArborText DVI driver, DVIPS, and an Apple LaserWriter+. This combination of DVI driver and printer cannot apparently be coaxed into including Macintosh figures, according to Arbortext's instructions, while the same DVIPS generated file sent to an NEC PostScript compatible printer works flawlessly!

3. Lastly, but of equal importance, the individuals who decide to dabble in this topic need a working knowledge of the Macintosh interface, PostScript programing, and TeX or LaTeX document preparation.

This article attempts to be general, when possible, by indicating how things are supposed to work, but in many places it is very specific. It is not possible for the author to test every combination of software and hardware. Thus the document serves a dual purpose: (1) it is intended to serve as a general overview for the person who is new to including Macintosh graphics in LaTeX documents and (2) it serves as a step-by-step user guide for those users who have certain software tools and hardware available[1]; this latter purpose provides an in-house user guide and thereby justifies the effort required to create the document. The author has no connection with nor financial interest in any of the commercial software or

[1]Cricket Software's Cricket Draw, Synergy Software/Abelbeck Software' VersaTerm-PRO, Arbortext's DVIPS, and an Apple's LaserWriter+

hardware mentioned in the article; it is just hard to do things without some kind of tools. Where appropriate, mention will be made of equivalent available public domain software and possible sources for obtaining the software.

There are two major sections in this article:

**Macintosh Graphics.** This section describes how to capture or convert Macintosh graphics into an equivalent PostScript representation;

**Using DVI Drivers to Include Graphics.** This section describes what needs to be done within the LaTeX document and to the Macintosh generated PostScript file to successfully include the graphic without scissors and paste.

These two major sections are connected by a shorter, and probably more self evident, section:

**Transfering Macintosh PostScript Files to Host Machines.** A step-by-step user guide on transferring text files from the Macintosh using Synergy Software's VersaTerm-PRO implementation of the *Kermit* file transfer protocol.

# 2 Macintosh Graphics

The Macintosh was designed to be used with an Apple LaserWriter printer. The LaserWriter printer is a PostScript device. Thus all Macintosh applications need to support PostScript, if they are to allow printing. What we would like to do is capture the PostScript representation of a graphic in a file, transfer the file to the host machine where LaTeX is used, and include the PostScript graphic file in a LaTeX document in a compatible manner.

In this section we will describe two methods for capturing the PostScript representation of a graphic into a file. The first method is applicable to all Macintosh applications that allow printing to an Apple LaserWriter. The second method is specific to the object oriented drawing application Cricket Draw.

## 2.1 Capturing Macintosh Graphics in a PostScript File

Unfortunately, few Macintosh applications, other than Cricket Draw, provide a menued option for generating a PostScript file, as will be described in the following sub-section. This is probably due to the existence of a universal technique for capturing any Macintosh application printer-directed output into a PostScript file. The technique is quite simple:

> After completing the graphic, select the *Print* option under the **File** menu. This produces the 'Print Dialog Box' which allows the user to select various options before sending a graphic to the printer. In the upper righthand corner of the 'Print Dialog Box' is an OK button. Click the mouse down, but do not release, on the OK button. With the mouse still clicked down on the OK button, depress and hold down the F key and then the Command (Apple or Flower) key, this combined key stroke is usually refered to as *Command-F*, then release the mouse button. A dialog box should appear stating that a PostScript file is now being created.

A file named 'PostScript0', or more generally 'PostScript*n*' where *n* is incremented by one for each PostScript file generated, will be created in the window where the generating application resides; the Macintosh *FINDER* may also be used to locate these PostScript files.

The reader may be wondering at this point, with such an easy and universal way of generating PostScript file representations of graphics, *i.e.* Command-F, why bother with any other technique, including the soon to be described Cricket Draw option technique, for generating PostScript files. The answer is that the PostScript files generated using the Command-F technique are not quite PostScript files. They are more correctly referred to as *QuickDraw* files. QuickDraw is a special PostScript shorthand created by Apple Computer[2]. NOTE: QuickDraw files will not produce a graphic image when sent to a PostScript printer, unless a special initialization file has been previously sent to the printer.

The QuickDraw initialization file is commonly called a *LaserPrep* file and various versions of it are known as AppleDict Version #*nn*, where *nn* is the current version number, *e.g.* AppleDict Version #65 (a.k.a. LaserPrep 65). The LaserPrep is a dictionary that translates QuickDraw into PostScript. The resulting translation does produce a printable graphic image on PostScript printers. Usually the LaserPrep file is only download once to a PostScript printer connected to a Macintosh. This PostScript dictionary called 'md' (Macintosh Dictionary?) remains resident in the printer's volatile memory until the printer is powered down. A copy of the Macintosh's current LaserPrep file may be generated by following the above described Command-F procedure, but substituting *Command-K* before releasing the mouse clicked down on the OK button in the 'Print Dialog Box'. Using the Command-K key sequence again generates a file named 'PostScript*n*' that contains both the LaserPrep file and the contents generated by the Command-F key sequence, *i.e.* the LaserPrep file is a prologue to the QuickDraw file; the boundary between the LaserPrep and QuickDraw files may be found by search for the first occurrence of the string `%%EOF` which indicates the last line of the LaserPrep file.

One more note about the LaserPrep file, near the bottom of the LaserPrep file are lines of hexadecimal numbers followed by several lines of zeros. The lines of hexadecimal numbers have very long lengths and will break most file transfer programs. These lines may be wrapped to a shorter line length by inserting carriage returns at an appropriate distance along the string. Alternatively, for the less faint-of-heart, these lines may be deleted from the LaserPrep file. More specifically, the lines between and including:

```
currentfile ok userdict/ ...
...
cleartomark
```

may be deleted. According to Bill Woodruff[3]

> "The dictionary [LaserPrep] includes some special encrypted assembly-language procedures that are proprietary to Apple Computer. If you check the "Faster

[2]Some very interesting comments on the development of QuickDraw are provided in a brief article by Bill Woodruff entitled, *PostScript and the Macintosh: A History*, **MacTech Quarterly**, Volume 1, Number 2, Summer 1989, pp.119-120.

[3]*op. cit.*

BitMap Printing" option in the generic Macintosh Page Setup dialog, for example, you activate an Apple bitmap smoothing routine that will not work if the installation of 'md' recognizes the printer is not from Apple."

## 2.2 Capturing Cricket Draw Graphics in a PostScript File

The Cricket Draw object oriented drawing program provides a menu option for saving graphics as a PostScript file. Cricket Draw provides this option to allow users without PostScript printers to save graphics in files which can be sent to other users or service bureaus for typesetting. It needs to be emphasized here that *this* file is a true PostScript file and not a QuickDraw file that requires the LaserPrep prologue. A true PostScript file is much easier to deal with and should be preferred over the corresponding QuickDraw file. Although, like any other Macintosh application, the QuickDraw equivalent can be generated via Command-F if desired.

The following steps are suggested[4] for obtaining a PostScript file representation.

After you have completed your graphic in the Cricket Draw application:

1. Under the **Edit** menu select the option *Select All*; this causes all the objects in the graphic to be selected.

2. Under the **Arrange** menu select the option *Group*; this causes all the selected objects to be treated a one object.

3. Under the **Arrange** menu select the option *Reflect on X*; this causes the entire graphic to be inverted vertically.

4. Under the **File** menu select the option *New*; this will create a new window where the PostScript representation of the graphic will be generated.

5. The just selected *New* option under the **File** menu produces a dialog box with two options for the type of file to be created, select the *PostScript* option. A new window will appear, named PostScript #1, overlaying the current graphic and some of the available menus will change.

6. Under the **Goodies** menu select the option *Generate PostScript*; this produces a listing of the PostScript representation of the graphic in the newly created window PostScript #1.

7. (OPTIONAL) The contents of the newly created PostScript window may be edited while remaining in the Cricket Draw application using the standard Macintosh editing features and the optional features under the **Search** menu.

8. Under the **File** menu select the *Save As...* option. This produces a dialog box with three options, select the *Complete* option. This dialog also allows the user to change the file's name and specify the folder where the file will be save.

[4]This sequence of steps was suggested to the author by Steve Strickland. The author is grateful to Mr. Strickland for starting him on the right track and, in part, for inspiring this document.

9. (OPTIONAL) The contents of the graphic window should also be saved, because the PostScript file representation **can not** be converted back into a graphic image by Cricket Draw. Thus if changes to the graphic are required, they will need to be made to a copy of the original graphic file.

As mentioned above, the PostScript file representating the graphic may be edited while the PostScript window is active in Cricket Draw. The saved file may also be edited by the user's favorite editor after exiting from Cricket Draw. Editing the PostScript file allows knowledgeable PostScript programmers to alter the program[5]. Novice PostScript programmers may also find it necessary or convenient to edit the PostScript file for the following reasons:

- The above steps 1-3 may be omitted if the PostScript file is edited to include one line as indicated:

  ```
  /vmstate save def
  1 -1 scale 30 -762 translate % Add this line after above line
  ```

  This added line changes the Cricket Draw coordinates to PostScript coordinates.

- Certain PostScript printers will object to the character string '|_____'in the PostScript file. All[6] occurrences of this string in the PostScript file should be deleted if the printer finds them offensive.

- Special symbols or characters, *e.g.* the partial differential symbol $\partial$, which are generated by certain combined key strokes, *e.g.* Option-d with the font type Symbol, need to be replaced by their PostScript octal equivalent, *e.g.* \266 for the partial differential.

A final word of advice. Always print your edited PostScript file, using a public domain application such as Adobe's SendPS, *before* downloading the file to the host machine and leaving the Macintosh. This may expose other potential problems with the Cricket Draw generated PostScript:

- One common problem is the presence of empty text boxes, *i.e.* text boxes that were created, but no text was entered, and the empty box was not deleted. These empty text boxes *must* be removed[7] otherwise the PostScript error message:

  ```
  %%[ Error: nocurrentpoint; Offending Command: currentpoint]%%
  ```

  will occur during an attempt to print the file.

[5]The editing may be performed interactively, with on-screen viewing of both the altered PostScript file and graphic, using the Macintosh application LaserTalk by Emerald City Software. (NOTE: A PostScript printer must be connected to the Macintosh to use the software interactively)

[6]This is know to cause a problem on Apple LaserWriter+ printers with the Symbol font family.

[7]Any easy method for finding these elusive empty entities is to change the default setting (from on to off) of the Text/Bitmaps Display option of the *Preferences* dialog box item under the **Layout** menu.

- Another problem was discovered only after the graphic was included in a LaTeX document and sent to the PostScript printer, an Apple LaserWriter+ with PostScript 42.2. The problem was caused by an arrowed line being slanted, *i.e.* a line with an arrow head at one end was slanted 10° by an inadvertent use of Cricket Draw's slanting tool. The problem was solved by undoing the slanting. The PostScript error message was

  ```
  %%[ Error: stackunderflow; OffendingCommand: exch]%%
  ```

  .

NOTE: The combination of Emerald City's LaserTalk application and the *Appendix D: Cricket Draw's PostScript Procedures* of the Cricket Software's Cricket Draw manual are invaluable in resolving these types of miscellaneous errors.

# 3 Transfering Macintosh PostScript Files to Host Machines

There are several protocols for transferring text files between computers and the various protocols are packaged in several Macintosh applications, both public domain and commercial. This section describes file transfer using the *Kermit*[8] protocol as implemented on the Macintosh in VersaTerm PRO 3.0. If the reader does not have access to a terminal emulator/communications program similar to VersaTerm-PRO, there are suitable public domain Macintosh application available via anonymous FTP on SUMEX-AIM.STANFORD.EDU in the <INFO-MAC> directory:

- MacKermit 0.9(40) - Kermit protocol (FreeWare);
- FreeTerm 3.0B3 - Xmodem and MacBinary protocols (FreeWare);
- Zterm 0.8 - Xmodem, Ymodem, Zmodem, & MacBinary protocols (ShareWare);
- NCSA TelNet - FTP file transfer through LocalTalk or Ethertalk and a Kinetics or Cayman GateWay (FreeWare).

After launching VersaTerm-PRO on the Macintosh, log into the host machine and select the target directory for the Macintosh PostScript files to be downloaded.

1. Invoke Kermit[9] on the host machine; usually by just typing the string `kermit`.
2. Place the host kermit into *Server Mode*; at the Kermit prompt type the string `server`, *e.g.*

   ```
   Kermit-32> server
   ```

[8]Kermit was written by Frank da Cruz and is in the public domain. It may be obtained by writting to: Kermit Distribution, Columbia University Center for Computing Activities, 612 West 115th Street, New York, NY 10025. While Kermit may or may not be a popular file transfer protocol in the Macintosh world, it does seem to exist on most mainframe computers and currently runs on over 200 systems.

[9]The ultimate Kermit reference is by Frank da Cruz, *Kermit A File transfer Protocol*, Digital Press (Order# EY-6705E-DP), 1987, ISBN 0-932376-88-6.

3. Under the VersaTerm PRO **File** menu select the *Kermit Server...* option; this produces the 'Kermit Server Commands' dialog box.

4. Select the *Send File...* option from the 'Kermit Server Command' dialog box and click on the OK button; this produces the standard Macintosh file finder dialog box.

5. Find and select the file to be transfered to the host machine and click on the SEND ONE button or double-click on the file name. VersaTerm PRO displays a thermometer type indicator to show the progress of the file transfer and beeps several times when the transfer is complete.

6. Again invoke the 'Kermit Server Command' dialog box by selecting the *Kermit Server...* option under the **File** menu of Versaterm PRO, or type the keyboard short cut Command-R.

7. If more than one file is to be transferred select the *Send File...* option.

8. If no more files are to be transferred select the *Finish - Exit Server Mode* option and click the OK button; this exits Kermit on the Macintosh and host machine, and returns the user to the host machine.

# 4 Using DVI Drivers to Include Graphics

## 4.1 Background

The general approach for including graphics in LaTeX documents is to use the `\special` command feature of the output device driver. The output device driver converts the intermediate device independent output from LaTeX, device independent (DVI) files, into device dependent files, *e.g.* for PostScript devices:

`text.dvi` $\longrightarrow$ `DVIPS` $\longrightarrow$ `text.ps`

The `\special` command alerts the device driver that some 'extra' material is to be included in the output file. Again for PostScript device drivers, this extra material will be a file containing PostScript, *e.g.* a Macintosh graphic.

The LaTeX document will contain embedded device driver dependent `\special` commands that indicate the filename of the graphic and where in the document the graphic is to be inserted, *e.g.*

```
\begin{figure}
\special{my.figure}
\end{figure}
```

In the above example, the command `\special` alerts the DVI driver to include the file `my.figure` in the document inside the floating *figure* environment. Because the graphic will appear immediately after the text preceeding the `\begin{figure}` environment, it is normally necessary to provide some white space in the document to prevent the graphic from overlaying the text. The two most common ways of providing this necessary white space are:

1. To explicitly include the white space in the figure environment via the LaTeX command \vspace, *e.g.*

```
\begin{figure}
\vspace{3.6in}
\special{my.figure}
\end{figure}
```

2. To implicitly include the white space via an optional argument of the \special command, *e.g.*

```
\begin{figure}
\special{my.figure}{3.6in}
\end{figure}
```

Other options may exist for the \special command, but the filename and amount of white space are the two most common.

In addition to specifying the amount of white space the graphic will occupy in the document, a few more items concerning the graphic need to be addressed:

- The origin of the graphic's coordinate system[10] must be made compatible with the DVI-driver's origin, otherwise it is possible that the graphic may not appear or only partially appear on the printed page.
- The graphic should not alter the current status of the printing device. LaTeX will decide where the graphic will appear. Hence the graphic should not issue page printing commands, but rather leave that instruction to the DVI-driver.
- The graphic should not redefine elements of the DVI-driver environment. The graphic should be a self-encapsulated environment that does not alter the DVI-driver environment in which it is embedded.

## 4.2 Modifying the LaserPrep File

> "LaserPrep was patched and fixed and extended and patched and fixed until it has reached its current state where even within Apple it is considered an embarrassing morass. Yet, it remains the world's most used PostScript program and the fundamental bottleneck through which almost all Macintosh printing is done. But to change it, even slightly, is to move a gigantic tectonic plate on which the entire superstructure of civilized Macintosh printing hangs in fragile balance[11]."

To use any DVI to PostScript driver to include Macintosh PostScript files generated with the Command-F technique described above, the LaserPrep file must be modified. There are two major types of changes:

[10] This is why the Cricket Draw graphic had to be inverted before creating the PostScript file.
[11] Bill Woodruff, *Op. cit.*

1. Changes to keep the LaserPrep from altering the printer's status, *i.e.* changes to `statusdict`.

2. A change to prevent the included PostScript file from issuing a `showpage` or `copypage` command.

The two other changes to LaserPrep #65 are either ineffective, *viz.* changing `LaserWriter` to `XXXXXXXXXX`, or printer specific, *viz.* changing `/legal` to `/letter` for a NEC compatible PostScript printer.

Listings of the differences between the Macintosh LaserPrep #65 and #68 files[12] and a suitably modified version for use with ArborText's DVIPS driver are given in Appendices I and II.

The modified version of LaserPrep #68 should be compatible with the public domain PostScript device driver DVI2PS[13], when used with the appropriate TEX prologue. DVI2PS, the TEX prologue file, modified LaserPrep #68 file, and associated information is available via anonymous FTP from LINC.CIS.UPENN.EDU (130.91.6.8) in the sub-directory DIST/PSFIG, files DVI2PS.TAR.Z and LPREP68.TAR. These files are in Unix TAR format and in the case of the DVI2PS file a Unix TAR and compressed binary format. However, the uncompressed and unTARed DVI2PS file does contain modifications for VMS systems and the unTARed LPREP68 file contains plain text files.

The modified LaserPrep file should be given an appropriate name, *i.e.* LaserPrep68.ps, and placed in a directory where DVIPS or DVI2PS can find and include it into the DVI file. A suggested directory location is one of the directories searched by the logical `TeX$Inputs`. Also, the modified LaserPrep file must be included in the LATEX document before the occurrence of the first Macintosh PostScript file. The LaserPrep file is usually included before the `\begin{document}` statement *e.g.* using ArborText's `\special` command,

```
\special{ps: plotfile LaserPrep68.ps global}
```

It is important to use the appropriate modified version of LaserPrep file that corresponds to the Command-F generated Macintosh graphic. To determine which version is appropriate edit the Command-F generated file and search for the following line near the top of the file

```
%%IncludeProcSet: "(AppleDict md-figure)" 68 0
```

The number 68 indicates that modified LaserPrep #68 should be used.

To determine the current version number of the LaserPrep file on your Macintosh either: select the LaserPrep icon in the SYSTEM folder and use *Get Info* under the **File** menu, or when attempting to print a document from the Macintosh look just to the left of the 'OK' button in the Print Dialog Box. In either case the number should be 4.0 or 5.2: 4.0 is equivalent to LaserPrep #65 and 5.2 is equivalent to LaserPrep #68; basically Macintosh system software 5.x, LaserPrep Version 4.0, and 'md' Version 65 all go together and Macintosh system software 6.x, LaserPrep Version 5.x, and 'md' Version 68 all go together, simple huh? This means that technically LaserPrep files such as LaserPrep #68 are refering to the version of the Apple dictionary and not the actual LaserPrep version number. Hopefully, someday soon Apple will get all these numbers in sync.

[12] The hexadecimal strings at the bottom of the LaseRprep files have been deleted.

[13] DVI2PS was not tested by the author.

## 4.3 Using BBFIG and PSFIG to Include Macintosh Graphics

**Acknowledgement:** Both BBFIG and PSFIG were developed and placed in the public domain[14] by Trevor J. Darrell. This subsection borrows quite liberally from Mr. Darrell's very nice documentation *Incorporating PostScript and Macintosh Figures in* TEX. Current versions of the software and documentation are available via anonymous FTP from LINC.CIS.UPENN.EDU (130.91.6.8) in the sub-directory DIST/PSFIG. The author, and undoubtly many other PSFIG users, are indebted to Mr. Darrell for his outstanding programming skills and his unselfish willingness to share this software and knowledge with others. The author hopes that this document continues Mr. Darrell's spirit of freely sharing knowledge.

PSFIG is a TEX macro package that facilitates the inclusion of arbitrary PostScript figures in LATEX documents. The real advantage of using the PSFIG macros is that they work within the Adobe constructs for Encapsulated PostScript Files (EPSF)[15]. For the PSFIG user, that means that graphics can be easily placed within a LATEX document, see for example, in this document *Appendix IV: Creating Multiple-Graphic Figures with Sub-Captions*, and that the same EPS file included in the LATEX document can be printed directly to a PostScript device; other systems for including PostScript graphics require a `currentpoint` to be set and that the `showpage` be explicitly disabled, thus disabling direct printing of the file.

To properly locate a PostScript figure, the DVI driver must know the size of the figure and its relative position on the printed page. This information is implicitly available within the various graphic constructs used in the PostScript file. The information *should* also be available explicitly in the BoundingBox comment[16] of the PostScript file's prologue.

The bounding box encloses all the marks made on a page as a result of executing (printing) a PostScript program. The BoundingBox comment has four parameters:

`%%BoundingBox:` $ll_x\ ll_y\ ur_x\ ur_y$

The four integer parameters, in units of points, represent the coordinates of the lower left ($ll_x$,$ll_y$) and upper right ($ur_x$,$ur_y$) corners of the bounding box in the default user (creator) coordinate system.

Although good PostScript programming practice dictates that the BoundingBox comment be provided in PostScript files, few Macintosh programs provide this information. Either the BoundingBox parameters are not specified, *e.g.*

`%%BoundingBox: ? ? ? ?`

which is the case for all Command-F generated PostScript files, or they specify the entire $8.5 \times 11$ page, *e.g.*

[14] Copyright notice from the PSFIG source: "All software, documentation, and related files in this distribution of psfig/tex are Copyright (c) 1987 Trevor J. Darrell. Permission is granted for use and non-profit distribution of psfig/tex providing that this notice be clearly maintained, but the right to distribute any portion of psfig/tex for profit or as part of any commercial product is specifically reserved for the author."

[15] *Encapsulated PostScript Files Specification Version 2.0*, Adobe Systems Inc., 1585 Charleston Road, P.O. Box 7900, Mountain View, CA 94039-7900, (415)961-4400, 16 January 1989.

[16] *PostScript Language Reference Manual*, Appendix C: Structuring Conventions, Adobe System Inc., Addison-Wesley Publishing Co., Inc., p.268, (1985).

```
%%BoundingBox:  0 0 612 792
```

which occurs in Cricket Draw generated PostScript files.

To supply the proper bounding box information one could measure the required dimensions, as discussed in the preceeding section's description of \PrintMacDraw, or use the PostScript utility BBFIG which calculates the bounding box parameters from the graphic information implicit in the PostScript file. The BBFIG PostScript file is preappended to the PostScript file[17] for which the bounding box is to be determined, and the combined file is sent to a PostScript printer. The result is a printed image of the PostScript graphic with the bounding box drawn around the graphic and bounding box parameters listed below the bounding box; the bounding box parameters are also returned via the print job's log file, if the log file feature is implemented for the host machine's print queue.

The accuracy of BBFIG in determining the bounding box information is not as good as may be needed in some circumstances. Experience has shown that the x-coordinates, *i.e.* $ll_x$ and $ur_x$, need to have 10-20 points (72.72 points to the inch) added to them and that the y-coordinates, *i.e.* $ll_y$ $ur_y$, need to have about 15 points subtracted from those issued by BBFIG. These differences are only noticeable when large figures, *i.e.* nearly full page originals, are included, especially if the PSFIG *clip* option is used. In most applications the figures are scaled down and not clipped, so the coordinate differences are negligible. However, if you notice your figures missing parts of wandering around the page, check the bounding box information. Of course the exact bounding box information can be obtained by simply printing the figure and using a ruler, in points if possible, to measure the four coordinates. Measuring the bounding box coordinates is also necessary for graphics from some Macintosh applications such as Cricket Software's Cricket Graph. For some unknown reason, Cricket Graph figures contain an invisible (un-stroked) path around the entire edge of the paper.

Once the proper bounding box information has been added to the BoundingBox comment of a PostScript file, it may be easily included in a LaTeX document by using the TeX macro PSFIG.

To include a PostScript figure with PSFIG, simply load the PSFIG macros at the beginning of your document with

```
\input{psfig}
```

then invoke the macro

```
\psfig{figure=input}
```

where *input* is the name of a PostScript file. PSFIG will automatically position the figure at the current place on the page, and reserve the proper amount of space in LaTeX so that it does not conflict with any other objects.

For example, if we have a file called 'piechart.ps' which contains the PostScript code to draw a pie chart, and it was the first figure in our non-page reversed[18] document, we would use the commands

[17] If the PostScript file was generated with using Command-F (QuickDraw), then a *non-modified* LaserPrep must also be preappended to the QuickDraw file.

[18] A page reversed document prints the last page first and first page last.

```
\input{psfig}
\centerline{\psfig{figure=piechart.ps}}
```

Since no mention of size was made in the above example, PSFIG draws the figure at its natural size (as if it was printed directly by a PostScript printer.) If the pie's natural size is several inches across, which is a little large, the pie can be reduced with:

```
\centerline{\psfig{figure=piechart.ps,height=1.5in}}
```

The `height` option specifies how tall the figure should be on the page. Since no `width` was specified, the figure will be scaled equally in both dimensions. By specifying both a `height` and a `width`, figures can be scaled disproportionately, with interesting results.

There are a few caveats associated with using PSFIG:

- For PSFIG to find the natural size of a figure, the figure must have a proper bounding box comment; see previous bounding box discussion.
- Some versions of LaTeX will fail to properly center a lone figure in a centering environment; a good work-around is to precede the figure with a hard space.
- On very large documents with many figures, the printer memory allocated to DVIPS may have to be limited.
- The `\psfig` macro will be confused by extra white space or new lines in its argument.

Certain PostScript figures (such as large bitmap images being transmitted at 9600 baud) can tie up a slower PostScript device such as an Apple LaserWriter for quite some time. To circumvent this, a figure may be printed in draft mode, which will reserve the same space on the page, but it will just print the name of the file[19] from which the figure is derived, and will not actually include it. The macro `\psdraft` will switch into draft mode, and all subsequent PSFIG macros will produce draft figures. The macro `\psfull` will switch out of draft mode.

The preceeding discussion of PSFIG is appropriate for Cricket Draw generated PostScript graphics and *any* other type of EPS file. The only exception to this broad statement are QuickDraw files generated by the Command-F option. QuickDraw files require a suitably modified LaserPrep file be preappended to the QuickDraw file. PSFIG provides an easy mechanism for including the LaserPrep file as described below.

There is a `prolog` option in the PSFIG macro to specify a file that should be preappended to the figure. The name of the prolog is, of course, site dependent; we have used `$disk3:[arbortex.psfig.lprep]lprep68.pro`. For example, if you had a file 'frog.mac' that contained the Macintosh code to draw Kermit, he could be included with:

```
\psfig{figure=frog.mac,prolog=$disk3:[arbortex.psfig.lprep] lprep68.pro}
```

[19]The current implementation of PSFIG does not support underscores (_) in file names for draft mode. Since PSFIG places the file name on the page using TeX commands in draft mode and hence reserved LaTeX characters cannot be used in the file name.

If there are many such figures, it is probable that the repeated inclusion of the `mac.pro` file will cause a significant increase in file size and transmission time. An alternative method is to load the Mac.Pro file once globally, so that it will be available throughout the rest of the document. Use

```
\psglobal{\$disk3:[arbortex.psfig.lprep]lprep68.pro}
```

at the beginning of your document to achieve this effect. For this to work properly, the `\psglobal` must appear before any Macintosh figures, and the final output must not be page reversed [1].

Recent experience has shown that the use of `\psglobal` with the Arbortext DVIPS driver causes some type of conflict for Macintosh graphics included inside a figure environment, *i.e.* the figure prints by itself on a page separate from the LaTeX document. Two solutions are: always include a prolog along with each Macintosh graphics (brute force) or include the `\psglobal` inside the first figure environment (lucky hacque).

## 4.4 Using ArborText's DVIPS to Include Macintosh Graphics

There are two elements involved in using ArborText's DVIPS driver to include Macintosh graphics:

1. Modifying the LaserPrep file to be compatible with the DVIPS generated PostScript environment;

2. Using Arbortext's TeX macro for including Macintosh PostScript files.

The modification of the LaserPrep file has already been discussed. What follows is a discussion of an alternative TeX macro for including Macintosh graphics called \PrintMacDraw. Although the Arbortext macro is adequate, the PSFIG macro is more robust and should be the preferred method for including Macintosh graphics. The following discussion simply serves to illustrate another technique for including Macintosh graphics and may be of pedagogical value.

As mentioned previously, more than just the filename of the PostScript graphic needs to be specified when including figures into LaTeX documents. In particular, the overall size of the figure and its location on the page must be transmitted to the DVIPS driver to properly locate the figure within the LaTeX document. To transmit this information to DVIPS, Arbortext has supplied a convenient TeX macro; a listing of the macro is given in Appendix III.

The \PrintMacDraw macro has six required arguments:

1. Name of Macintosh generated PostScript file;

2. Figure height (inches);

3. Figure width (inches);

[1] It is possible to use psglobal in a page reversed document; place it just before the last figure in your document. This is living dangerously, and you do so at your own risk.

4. Lower left x-coordinate of figure (inches from left side of paper);

5. Lower left y-coordinate of figure (inches from bottom of paper);

6. Desired figure magnification (1.000 = unmagnified).

An example of the \PrintMacDraw syntax is

```
\PrintMacDraw{macdraw.fig}{6.25}{5}{1.25}{2.5}{1.0}
```

All of the above measurements are for the unmagnified dimensions of the figure as they appear when the figure is printed by itself on a page. Normally the figure is printed, before it is to be included in a LaTeX document, and the dimensions are recorded. These dimensions may also be altered to affect cropping of the figure. \PrintMacDraw centers the cropped region in the current LaTeX environment, *i.e.* it centers the figure in a box of width `\hsize`. If you specify a figure magnification other than 1.0, the figure is scaled appropriately, and LaTeX reserves space for a correspondingly scaled box which is still centered in the current environment.

# Appendix I: Listing of Changes to LaserPrep #65

```
************
File Mac_LaserPrep65
    9   /LW{save statusdict/product get(LaserWriter)anchorsearch
   10   exch pop{length 0 eq{1}{2}ifelse}{0}ifelse exch restore}bind def
******
File Modified_LaserPrep65
    9   %-les 11-Oct-88 Changed LaserWriter to XXXXXXXXXX suggested by Arbortext
   10   %-les 11-Oct-88-% /LW{save statusdict/product get(LaserWriter)anchorsearch
   11   /LW{save statusdict/product get(XXXXXXXXXXX)anchorsearch
   12   exch pop{length 0 eq{1}{2}ifelse}{0}ifelse exch restore}bind def
************
************
File Mac_LaserPrep65
   13   ok{statusdict begin 9 sccinteractive 3 ne exch 0 ne or{9 0 3
          setsccinteractive}if end}if
   14   /md 250 dict def md begin
******
File Modified_LaserPrep65
   15   %-cwb 29-Aug-1988-% ok{statusdict begin 9
          sccinteractive 3 ne exch 0 ne or{9 0 3 setsccinteractive}if end}if
   16   /md 250 dict def md begin
************
************
File Mac_LaserPrep65
   47   statusdict begin waittimeout 300 lt{/waittimeout
          300 def}if end
          scaleby96{ppr aload pop 4 -1 roll add 2 div 3 1 roll add 2 d
   48   /fr{3 index 3 index xl ppr aload pop 3 -1 roll 2 mul add 3 1 roll
          exch 2 mul add
******
File Modified_LaserPrep65
   49   %-cwb 29-Aug-1988-% statusdict begin waittimeout 300 lt{/waittimeout
          300 def}if end
   50   scaleby96{ppr aload pop 4 -1 roll add 2 div 3 1 roll add 2 div 2 copy
   51   translate .96 dup scale neg exch neg exch translate}if}bdf
   52   /fr{3 index 3 index xl ppr aload pop 3 -1 roll 2 mul add 3 1 roll
          exch 2 mul add
************
************
File Mac_LaserPrep65
  360   /cp{not{userdict/#copies 0 put}if ma 0 gt{{t1 the{exit}if}loop}if
          {copypage}{showpage}ifelse pm restore}bdf
  361   /px{0 3 1 roll tp tt}bdf
******
File Modified_LaserPrep65
  364   /cp{not{userdict/#copies 0 put}if ma 0 gt{{t1 the{exit}if}loop}if
  365   %-cwb 29-Aug-1988-% {copypage}{showpage}ifelse
  366   pop %-cwb 29-Aug-1988-%
  367   pm restore}bdf
  368   /px{0 3 1 roll tp tt}bdf
************
************
```

```
File Mac_LaserPrep65
  381   md begin/bigs[lnop lnop/legal load userdict/a4 known{/a4 load}{lnop}
          ifelse lnop lnop lnop lnop lnop]def
  382   /smalls[lnop userdict/note known{/note load}{dup}ifelse lnop
          userdict/a4small known{/a4small load}{lnop}ifelse 2 index lnop
******
File Modified_LaserPrep65
  388   %-cwb 29-Aug-1988, replace /legal with /letter (below)-%
  389   md begin/bigs[lnop lnop/letter load userdict/a4 known{/a4 load}{lnop}
          ifelse lnop lnop lnop lnop lnop]def
  390   /smalls[lnop userdict/note known{/note load}{dup}ifelse lnop
          userdict/a4small known{/a4small load}{lnop}ifelse 2 index lnop
************

Number of difference sections found: 5
Number of difference records found: 13
```

# Appendix II: Listing of Changes to LaserPrep #68

NOTE: All the lines ending with %TRUN% have been intentionally truncated to simplify the comparative listing in this Appendix.

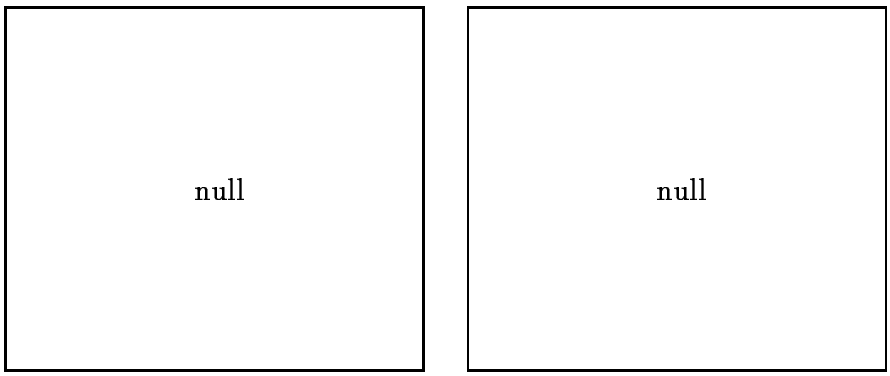
```
************
File Mac_LaserPrep68
    1   %%Title: "Laser Prep -- The Apple PostScript Dictionary (md)"
    2   %%Creator: Apple Software Engineering
******
File Modified_LaserPrep68
    1   %!Title: "Laser Prep -- The Apple PostScript Dictionary (md)"
    2   %%Creator: Apple Software Engineering
************
************
File Mac_LaserPrep68
    5   %  CopyRight Apple Computer, Inc. 1984,1985,1986,1987,1988 All Rights Reserved.
    6   %%EndComments
    7   %%BeginProcSet: "(AppleDict md)" 68 0
    8   statusdict begin waittimeout 300 lt{0 60 300 setdefaulttimeouts}if end
    9   currentfile statusdict begin serverdict begin product (LaserWriter II NT) e%TRUN%
   10   /execjob {31 sendpcmd 4 eq tonerlight //execjob exec} bind def end end eexe%TRUN%
   11   35de8eabfc7fa5eac0431edc501ad43f5fcbdf9fdd321cce93b525f4439dd94696bf56ac13a%TRUN%
   12   d27489a69d6b69c6a841468b46720b75ad65650700e0c528e7af61e7e3e821b59445c44b698%TRUN%
   13   7b1c0b5203cfc46868d6e3c0d66db79174e7091e32e307679732da062e440e41dabd36a161b%TRUN%
   14   6eb9e36ec45bad2a0b82829a72e0efa2d87332c482196e690361168271c55053341ab3
   15   /sc {60 45 {abs exch abs 2 copy add 1 gt{1.0 sub dup mul exch 1.0 sub dup m%TRUN%
   16   ifelse}setscreen} bind def statusdict begin product(LaserWriter II)anchorse%TRUN%
   17   {pop pop/letter [/letter load /exec load /sc load /exec load]cvx def/legal %TRUN%
   18   /lettersmall [/lettersmall load /exec load /sc load /exec load]cvx def/a4sm%TRUN%
   19   systemdict/currentpacking known{currentpacking true setpacking}if
   20   /LW{save statusdict/product get(LaserWriter)anchorsearch
   21   exch pop{length 0 eq{1}{2}ifelse}{0}ifelse exch restore}bind def
   22   /LW+{LW 2 eq}bind def
   23   /ok{systemdict/statusdict known dup{LW 0 gt and}if}bind def
   24   ok{statusdict begin 9 sccinteractive 3 ne exch 0 ne or{9 0 3 setsccinteract%TRUN%
   25   /md 250 dict def md begin
******
File Modified_LaserPrep68
    5   % Modified by T. Darrell 1/89 [TJD]
    6   % ) CopyRight Apple Computer, Inc. 1984,1985,1986,1987,1988 All Rights Reserved.
    7   %%EndComments
    8   %%BeginProcSet: "(AppleDict md)" 68 0
    9   %%[TJD]%statusdict begin waittimeout 300 lt{0 60 300 setdefaulttimeouts}if end
   10   %%[TJD]%currentfile statusdict begin serverdict begin product (LaserWriter %TRUN%
   11   %%[TJD]%/execjob {31 sendpcmd 4 eq tonerlight //execjob exec} bind def end %TRUN%
   12   %%[TJD]% HEX CODE DELETED
   13   %%[TJD]%/sc {60 45 {abs exch abs 2 copy add 1 gt{1.0 sub dup mul exch 1.0 s%TRUN%
   14   %%[TJD]%ifelse}setscreen} bind def statusdict begin product(LaserWriter II)%TRUN%
   15   %%[TJD]%{pop pop/letter [/letter load /exec load /sc load /exec load]cvx de%TRUN%
   16   %%[TJD]%/lettersmall [/lettersmall load /exec load /sc load /exec load]cvx %TRUN%
   17   systemdict/currentpacking known{currentpacking true setpacking}if
   18   %%[TJD]%/LW{save statusdict/product get(LaserWriter)anchorsearch
```

```
   19   %%[TJD]%exch pop{length 0 eq{1}{2}ifelse}{0}ifelse exch restore}bind def
   20   /LW{0}def
   21   /LW+{LW 2 eq}bind def
   22   /ok{systemdict/statusdict known dup{LW 0 gt and}if}bind def
   23   %%[TJD]%ok{statusdict begin 9 sccinteractive 3 ne exch 0 ne or{9 0 3 setscc%TRUN%
   24   /md 250 dict def md begin
************
************
File Mac_LaserPrep68
   56   /txpose{fNote{smalls}{bigs}ifelse pgs get exec pxs pys scale ppr aload pop %TRUN%
   57   xflip yflip and{pop exch neg exch translate 180 rotate 1 -1 scale ppr 3 get%TRUN%
   58   xflip yflip not and{pop exch neg exch translate pop 180 rotate ppr 3 get pp%TRUN%
   59   {noflips{translate pop pop 270 rotate 1 -1 scale}if xflip yflip and{transla%TRUN%
   60   xflip yflip not and{translate pop pop 90 rotate ppr 3 get ppr 1 get neg sub%TRUN%
   61   statusdict begin waittimeout 300 lt{/waittimeout 300 def}if end scaleby96{p%TRUN%
   62   /fr{4 copy ppr aload pop 3 -1 roll add 3 1 roll exch add 6 2 roll 3 -1 roll
******
File Modified_LaserPrep68
   55   /txpose{
   56   %[TJD] fNote{smalls}{bigs}ifelse pgs get exec
   57    pxs pys scale
   58    ppr aload pop por{
   59    noflips{
   60    pop exch neg exch translate pop 1 -1 scale
   61    }if
   62    xflip yflip and{
   63    pop exch neg exch translate 180 rotate 1 -1 scale
   64    ppr 3 get ppr 1 get neg sub neg ppr 2 get
   65    ppr 0 get neg sub neg translate
   66    }if
   67    xflip yflip not and{
   68    pop exch neg exch translate pop 180 rotate
   69    ppr 3 get ppr 1 get neg sub neg 0 translate
   70    }if
   71    yflip xflip not and{
   72    ppr 1 get neg ppr 0 get neg translate
   73    }if
   74    }{
   75    noflips{
   76    translate pop pop 270 rotate 1 -1 scale
   77    }if
   78    xflip yflip and{
   79    translate pop pop 90 rotate 1 -1 scale
   80    ppr 3 get ppr 1 get neg sub neg ppr 2 get
   81    ppr 0 get neg sub neg translate
   82    }if
   83    xflip yflip not and{
   84    translate pop pop 90 rotate ppr 3 get ppr 1 get
   85    neg sub neg 0 translate
   86    }if
   87    yflip xflip not and{
   88    translate pop pop 270 rotate ppr 2 get ppr 0 get
   89    neg sub neg 0 exch translate
   90    }if
```

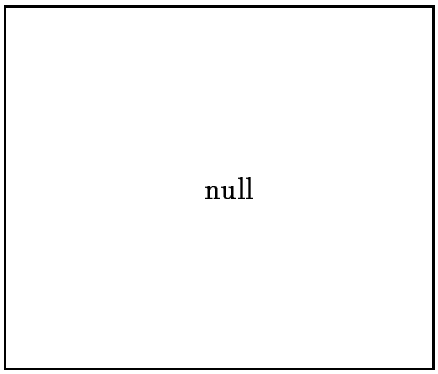
```
   91    }ifelse
   92    statusdict begin waittimeout 300 lt{
   93    /waittimeout 300 def
   94    }if end
   95    scaleby96{
   96    ppr aload pop 4 -1 roll add 2 div 3 1 roll add 2 div
   97    2 copy translate .96 dup scale neg exch neg exch translate
   98    }if
   99   }bdf
  100   /fr{4 copy ppr aload pop 3 -1 roll add 3 1 roll exch add 6 2 roll 3 -1 roll
************
************
File Mac_LaserPrep68
  374   /cp{not{userdict/#copies 0 put}if ma 0 gt{{t1 the{exit}if}loop}if{copypage}%TRUN%
  375   /px{0 3 1 roll tp tt}bdf
******
File Modified_LaserPrep68
  412   %%[TJD]%/cp{not{userdict/#copies 0 put}if ma 0 gt{{t1 the{exit}if}loop}if{c%TRUN%
  413   /cp{ pop pop pm restore } bdf
  414   /px{0 3 1 roll tp tt}bdf
************
************
File Mac_LaserPrep68
  402
  403   %%EndProcSet
******
File Modified_LaserPrep68
  441   %%[TJD]%currentfile ok userdict/stretch known not and{eexec}{flushfile}ifelse
  442   %%[TJD]%
  443   %%[TJD]%currentfile ok userdict/smooth4 known not and{eexec}{flushfile}ifelse
  444   %%[TJD]%
  445   %%EndProcSet
************

Number of difference sections found: 5
Number of difference records found: 72
```

# Appendix III: Listing of Arbortext's \PrintMacDraw Macro

```
\def\PrintMacDraw#1#2#3#4#5#6{
 \vskip 10pt \nobreak
 \hbox to \hsize{%
  \hss
  \dimen0=#3in%
  \hbox to #6\dimen0{%
   \dimen0=#2in%
   \vbox to #6\dimen0{
     \vss
     \special{ps::[asis, begin]
        0 SPB
        /MacDrawCheckPoint save def
          md begin /cp {pop pop} def end
          Xpos Ypos translate
          #6 dup scale
          #4 72 mul neg #5 72 mul neg translate
     }
     \special{ps: plotfile #1 asis}
     \special{ps::[asis,end]
        MacDrawCheckPoint restore
        0 SPE
     }
   }%
   \hss
  }
  \hss
 }
 \vskip 10pt
}
```




(a) Sub-caption (b) Sub-caption




(c) Sub-caption

Figure 1: Example of sub-captioned multiple-graphic figure.

# Appendix IV: Creating Multiple-Graphic Figures with Sub-Captions

Figure 1 shows an example of a multiple-graphic figure with sub-captions created within a LaTeX document environment using the PSFIG macros. Multiply captioned figures can be easily created using the PSFIG macros and ArborText's DVIPS.

There are at least as many ways to position figures on a page using LaTeX commands as there are configurations of figures on a page. For sub-captioned figures, possibly the easiest method is to maintain a close association between the graphic and the sub-caption. This can be easily accomplished via the LaTeX commands `\parbox` and `\centerline`. Figure 1 was created using the following sequence of LaTeX commands[20]:

```
\begin{figure}
\psdraft
     \centerline{\parbox[t]{1.8in}{
          \centerline{\fbox{\psfig{figure=sample.epsf,height=1.8in}}}
```

[20]See the discussion near the end of this Appendix about some minor changes.

```
          \centerline{(a) Sub-caption}}
%
          \hspace*{.5in}
%
          \parbox[t]{1.8in}{
          \centerline{\fbox{\psfig{figure=sample.epsf,height=1.8in}}}
          \centerline{(b) Sub-caption}} }
     \vspace*{.5in}
          \centerline{\hbox{\fbox{
          \psfig{figure=sample.epsf,height=1.8in} }}}
          \centerline{(c) Sub-caption}
     \vspace*{.2in}
\caption{Example of sub-captioned multiple-graphic figure.}
\label{fig:multi_figs}
\psdraft
\end{figure}
```

The same commands as above with comments:

```
\begin{figure}          % begin figure environment
\psdraft                % PSFIG will only print the graphic's
%                       %   file name using the BoundingBox information
%                       %   to center this text. (Demonstration Only)
%
\centerline{            % begin center environment for top two figures
    \parbox[t]{1.8in}{ % begin parbox for top left figure
          \centerline{ % center top left figure inside parbox
             \fbox{    % draw a box around figure (Demonstration Only)
               \psfig{figure=sample.epsf,height=1.8in}}}
%                       % place sub-caption in parbox with top left figure
          \centerline{(a) Sub-caption}}
%                       % end parbox environment
    \hspace*{.5in}      % include a horizontal space between top figures
%
    \parbox[t]{1.8in}{ % begin parbox for top right figure
          \centerline{ % center top right figure inside parbox
             \fbox{    % draw a box around figure (Demonstration Only)
               \psfig{figure=sample.epsf,height=1.8in}}}
%                       % place sub-caption in parbox with top right figure
          \centerline{(b) Sub-caption}} }
%                       % end parbox environment and end center environment
\vspace*{.2in}          % vertical space between top and bottom figures
%
\centerline{            % begin center environment for bottom figure
     \hbox{             % for a single figure the hbox is easier
%                       % locate bottom figure inside hbox
       \fbox{           % draw a box around figure (Demonstration Only)
          \psfig{figure=sample.epsf,height=1.8in} }
```

```
           }}          % end hbox and centerline environments
%                      % center bottom figure sub-caption
          \centerline{(c) Sub-caption}
%
\vspace*{.2in}         % add vertical space between sub and main captions
%                      % main caption
\caption{Example of sub-captioned multiple-graphic figure.}
\label{fig:multi_figs} % main caption label
\psfull                % toggles off the \psdraft mode (Demonstration Only)
\end{figure}           % end figure environment
```

A listing of the Encapsulated PostScript File (EPSF) to be used with the above example follows:

```
%!PS-Adobe-2.0 EPSF-2.0
%%Title: sample.epsf
%%Creator: Len Schwer
%%CreationDate: 14-JUN-89
%%BoundingBox: 100.0 100.0 400.0 365.0
%%Pages: 1
%%EndComments
%%Trailer
```

However, the actual demonstration was constructed without the sample EPSF file (to make this document self contained) by providing PSFIG with the BoundingBox arguments in the call, *e.g.*

```
\psfig{figure=sample.epsf,height=1.8in,bbllx=100bp,bblly=100bp,
                                        bburx=400bp,bbury=365bp}
```

NOTE: Although the above call to the PSFIG macro is shown on two lines, the PSFIG macro will be confused by extra white space or new lines in its argument.