Wilfried Ricken
Im Kempken 27
44799 Bochum
Germany
e-mail: `wilfr@hadron.tp2.ruhr-uni-bochum.de`

DirectTEX *Pro*

Version 2.1.2 — 16th March 1997

A complete TEX program package for the Apple Macintosh.

Remark: This program package is ShareWare. Please read the conditions given in the ShareWare instructions at the beginning of this manual.

Contents

1 Introduction

1.1 General

This is a manual for the DirectTEX software package. This package contains several components:

- The DirectTEX *Pro* shell, a program that contains everything needed to use TEX in an integrated environment. The DirectTEX *Pro* shell contains also the driver needed to display and print dvi files.
- The tools TEX, METAFONT, BibTEX, MakeIndex, all the tools from the TeXWare and MFWare packages (like DVIType, ...), and much more (e.g. dvips).
- The latest macros to be used with TEX, e.g. LATEX 2$_\varepsilon$, LATEX 2.09, Plain TEX, ...
- All the files needed to generate fonts with METAFONT, e.g. the CMR fonts, the DC fonts, AMS fonts, logo fonts, ...
- Virtual fonts to support all 35 standard PostScript fonts.
- Manual and help files.

The DirectTEX software package has been written by Wilfried Ricken and is ShareWare. You will find some instructions and the ShareWare fees in the next two sections.
Throughout this manual it is assumed that you are familiar with your Macintosh and with the basic principles of TEX. Introductions to TEX and METAFONT as well as the associated tools can be found in e.g. [Knu86], [Sch88], [Kop91], [Kop90].

1.2 Distribution Channels, Contact Addresses

The DirectTEX software package is distributed primarily via Internet. All people that have access to Internet are able to fetch the latest version of the SoftWare via ftp. You may copy the whole distribution to your local computer by using the following commands:

```
ftp hadron.tp2.ruhr-uni-bochum.de
login: anonymous
password: <your e-mail address>
cd pub/directtex
prompt
binary
mget *
quit
```

There are a lot of other ftp servers that will hold a copy of the main distribution server. You may use programs like gopher or archie to search for such servers. All people without Internet access may receive the latest version of the software by sending in

- 10 HD-disks
- a stamped return envelope

Please make sure the HD disks are not corrupt. You will find the address to send the disks to on the title page of this manual. There are still some other distribution channels, e.g. TEX user groups in some countries. Updates are available for registered users. I will notify every registered user if a new version is available, either via e-mail or ordinary mail. You may get an update the same way as described above. If you have problems or other questions you may contact me either via e-mail or ordinary mail. Please, do not phone me. Remember that I'm not a big company with hotline and user support center, but simply a privat person developing software like DirectTEX.

1.3 ShareWare Instructions

The DirectTEX program package by Wilfried Ricken is ShareWare. That means:

- Professional distribution of this software is strictly prohibited. This applies in particular to commercial public domain and shareware distributors.
- Once you received a copy of this program package, please pass it on to other people, e.g. your friends. Notice, however, that you are not allowed to ask for money apart from eventual costs of the diskettes.
- You are allowed to test this package. If after an appropriate amount of time you should find that you don't like it, then do delete it from your collection. Roughly 10 days of real work should be enough; of course, years are definitely too much :-).
- If you consider this software to be of use to you: Register as a DirectTEX user with me. Pay the ShareWare fee by mailing a cheque to me made payable to:

  Wilfried Ricken
  Im Kempken 27
  44799 Bochum
  Germany
  e-mail: `wilfr@hadron.tp2.ruhr-uni-bochum.de`

  I can also send you details about my bank account if you prefer this option. Please remember to always send in a completed registration form, otherwise it is impossible to register you.

The ShareWare fees are:

- The ShareWare fee presently amounts to US-$ 100.– or 150.– DM for up to three installations. This fee covers all future versions of DirectTEX. Each additional installation costs US-$ 20.– or 30.– DM.

Upon registration, you will receive a license and a serial number. Moreover, registered users of DirectTEX have the following advantages:

- You will immediately be notified of a new version. Updates may be received via the normal distribution channels without additional fees.
- Should you encounter a problem during installation or use of this software (but not regarding TEX, LATEX or similar software), feel free to contact me. This holds also in case you discover bugs in my software or wish to propose improvements. Please contact me only by E-Mail or ordinary Mail. Do *not* phone me !
- A short remark: Unregistered users of DirectTEX may send me bug reports or other comments about DirectTEX, but do not expect any answer from me.
- One more remark: Please do support this software package even in case you never payed ShareWare fees up to now. Its development took a lot of work and time. I hope there will be people who understand and act appropriately.

Furthermore: Just compare all the features of this TEX and METAFONT package to others that are available for the Apple Macintosh. You will find that there is none as complete as the present one, neither public domain nor commercial. Also compare the costs of a commercial package to the ShareWare fee I ask for, taking into account the possibilities offered to influence further development of the present package.

1.4 Serial Number Mechanism

In order to prevent people from using DirectTEX without paying the ShareWare fee DirectTEX includes a license and serial number mechanism. Every user who wants to register with me must supply an user and organisation name. After receiving the ShareWare fee I will send back a license number which is unique for each registration and a serial number which simply is computed out of the user and organisation name and the license number.
If you do not register as a user with me, you will see the registration dialog every time you start a TEX tool. Note that DirectTEX is fully functional even if you do not register. Therefore it is possible to try DirectTEX for a limited amount of time without paying the ShareWare fee.

1.5 Hardware and Software Requirements

This software runs on every Apple Macintosh with a 68020 or higher processor or a PowerPC processor. For 680x0 systems at least 4MB RAM are required, but 8MB RAM are recommended. For PowerPC systems, at least 8MB RAM are required. DirectTEX requires System 7.x, System 7.5 is recommended.
Should you encounter any problems when using other configurations, let me know of them, giving an exact description of your hard- and software environment.

1.6 General Information about DirectTEX and DirectTEX *Pro*

DirectTEX is a complete implementation of all programs conceived and written by DONALD E. KNUTH under his TEX and METAFONT project, as well as additional programs written by several other people. For the Apple Macintosh, I have implemented all programs as tools running under the DirectTEX *Pro* shell. Additionally, some commands have been implemented as scripts that are executed by the DirectTEX *Pro* command interpreter.
The DirectTEX software package contains also the `TeX--XeT` extension of TEX. This extension allows bi-directional typesetting. This feature is very useful for all people writing from right to left. There are four new TEX primitives: `\beginL`, `\endL`, `\beginR` and `\endR`. Their use is straightforward: Simply write `\beginR This sentence is reversed.\endR` to get the result desrever si ecnetnes sihT. This enhancement to TEX has been written by D.E. Knuth and rewritten by Peter Breitenlohner to avoid changes in the `dvi` files which would require new `dvi` drivers. Many thanks to Peter for sharing his work with other people!

2 A Quick Tour of DirectTEX

2.1 Creating TEX Files

In the following sections I will explain the basic usage of DirectTEX *Pro* using a little example. Note that if something unexpected happens you should try to find the source for it and should read the sections in this manual that describe the program that failed.

By a double-click on the DirectTEX *Pro* icon, DirectTEX *Pro* will start up. After some time you will see the DirectTEX *Pro* menu bar at the top of your screen. Now select `New...` from the file menu. A dialog box will appear that lets you specify the name and location of the file to create. Type in `HelloWorld.tex` and select a directory where to store the new TEX file, then click `New` (see figure 1). A new window will appear which will be empty (since we created a new file).

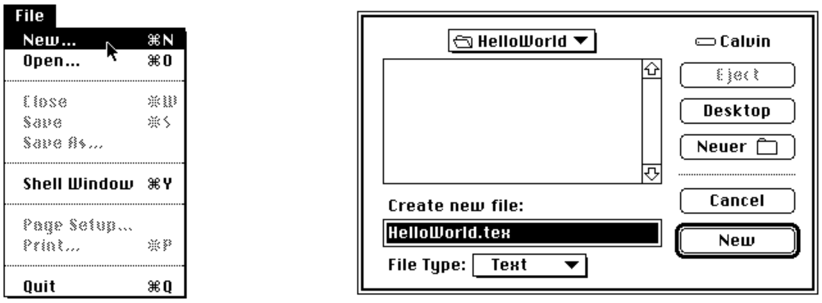


Figure 1: Creating a new TEX file

Now you are ready to type in your text. In this example we will create a new LATEX 2_ε document. E.g. enter

```
\documentclass{article}
\begin{document}
Hello, world.
\end{document}
```

This document contains some LATEX 2_ε commands and the text you want to typeset. Select `Save` from the `File` menu to save your new document to disk.

◇ Be sure to always save the file to disk before you typeset it. Otherwise you will *not* typeset your last changes, but the last saved version.

2.2 Creating Format and Base Files

Now that you have created a new document you probably want to typeset it. However, before you can do that you have to initialize the macro package it uses (in this example the LATEX 2_ε macro package). Note that you have to initialize a macro package only once, after initialization you may use it for different documents without initializing it again. The macro packages that are already initialized will appear in the `Formats` menu.

METAFONT is a program that may be used to create the fonts needed to display and print `dvi` files created by TEX. However, before you can use METAFONT you must initialize a base file that METAFONT will use. This may be done in the same way as initializing a format file for TEX.

To initialize a format or base file select `Initialize Formats...` from the `Formats` menu. A dialog box (see figure 2) will appear which offers you a wide choice of available formats. Now select `LaTeX (TeX)` and `Plain (MF)` from the list and click `Init` to initialize the LATEX 2_ε macro package and the plain base for METAFONT. Note that another window will appear on

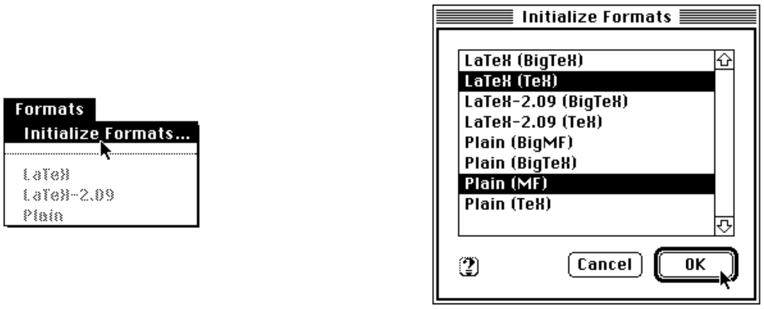


Figure 2: Selecting a format to initialize

the screen, the DirectTEX *Pro* shell window. After some time the LATEX 2$_\varepsilon$ macro package and the plain base should be initialized and the shell window will contain some output of TEX and METAFONT.
You may have a look at the `Formats` menu: There should now be an additional item `LaTeX` at the bottom of the menu. Now that we have a new document and the appropriate format file we will typeset our new document.

- ⋄ Be sure to initialize the plain base file for METAFONT. Otherwise the automatic generation of missing fonts will not function properly.
- ⋄ If the plain base for METAFONT is missing, it is initialized automatically in newer versions of DirectTEX, so you may skip this step.

2.3 Translating and viewing TEX Files by hand

You may now simply enter a command like `LaTeX HelloWorld` into the shell window. However, before you can do this you must make sure that the current directory is set to the directory where your `HelloWorld.tex` file resides. To set the current directory, use the item `Set Directory...` in the `Directory` menu. The dialog box that appears lets you select the current directory (see figure 3).

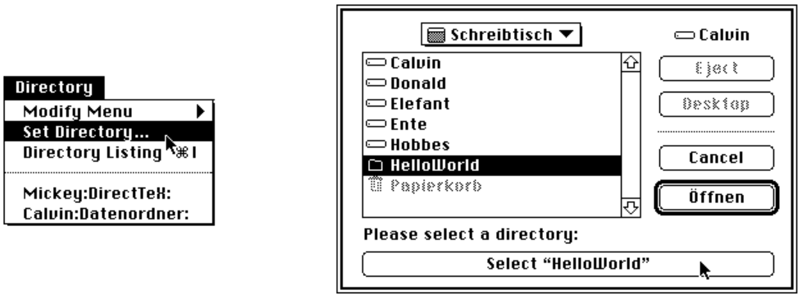


Figure 3: Setting the current directory

Now you may execute the command `LaTeX HelloWorld` to typeset your first TEX file. If the shell window is not visible simply select `Shell Window` from the `File` menu to display it again. Type the command and type `Return` or `Enter` (see figure 4). TEX will start up and typeset your document, creating a file `HelloWorld.dvi` that contains the typesetted document in encoded form.

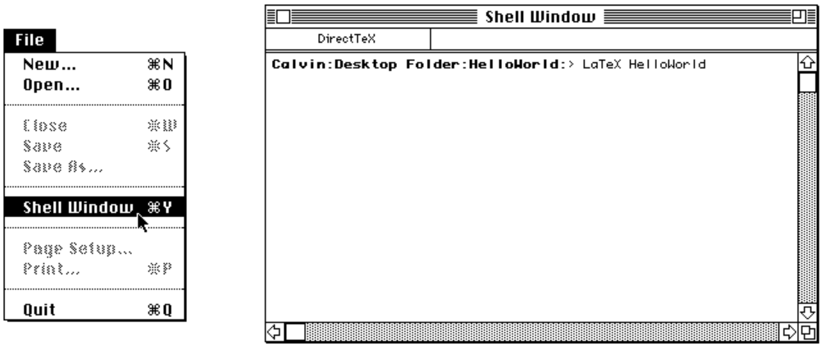


Figure 4: Typesetting `HelloWorld.tex`

Now execute the command `MacDVI HelloWorld`. This will start the program **MacDVI** which is a part of the DirectTEX *Pro* shell, and this program will display the `dvi` file on your computer's screen. However, if you've never used DirectTEX before, there are some fonts missing to display and print `dvi` files. The DirectTEX *Pro* shell is able to create these fonts exactly when they are needed. However, make sure that you have initialized the plain base file before executing `MacDVI HelloWorld`. Otherwise METAFONT is not able to create your missing fonts. After METAFONT has created the missing fonts, a window will appear on the screen displaying the `HelloWorld.dvi` file.
In principle you now know everything to work with your simple TEX file. However, always typing such commands like `LaTeX HelloWorld` may be very boring to you. Therefore DirectTEX *Pro* has some support for you build-in.
Simply bring the window of `HelloWorld.tex` to the front and use the menu item `Use Front Window` from the `TeX` menu. This will enable all disabled menu items in the `TeX` menu. Make sure that `LaTeX` is selected in the `Formats` menu. Now selecting `Typeset Main TeX File` from the `TeX` menu is equivalent to typing `LaTeX HelloWorld`.
For small tasks, this mechanism is powerful enough. However, if you are creating a bigger project with lots of different text files, the DirectTEX *Pro* project manager will help you a lot.

2.4 Creating and Using Projects

In principle there are two ways to create a new project:

- Select `New...` from the `File` menu again. Make sure that you select `Project` as file type at the bottom of the dialog. Name your new project `HelloWorld.prj` and place it in the same folder where `HelloWorld.tex` resides in (see figure 5).

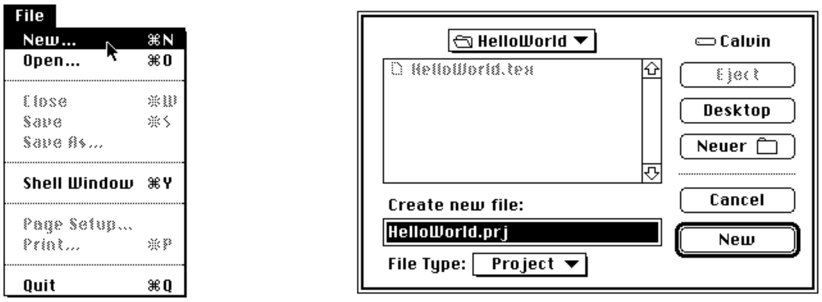


Figure 5: Creating a new project

Some details about a project: A project may contain any number of text files. One of these files must be declared as the main file, and one of these files may be declared as the current file. A project also contains the name of the macro package to use when typesetting the main file.

So after creating your project you have to do some things:

- Add text files to your project. In our example simply add the file `HelloWorld.tex`. Do this by clicking the `Add...` button and selecting the file in the dialog box that appears.
- Make one of these files the main file. You do this by selecting this file and clicking the `Main` button. Optionally make one of these files the current file. In our example you should select `HelloWorld.tex` in the list and click the `Main` and `Current` button.
- Select the format that should be used from the `Formats` menu. In our example simply select `LaTeX` from the `Formats` menu.

• Bring the window of `HelloWorld.tex` to the front and select `Create Project...` from the `TeX` menu. DirectTEX *Pro* will create a project named `HelloWorld.prj` in the same folder as your text file `HelloWorld.tex`, add `HelloWorld.tex` to your project, make it the main and current file and make `LaTeX` the format to use. This is much faster than creating a new project from scratch as decribed above.

Now have a look at the `TeX` menu: The first eight menu items may only be used if you have a valid TEX project. Each item does exactly the job described by the name of the item. These items may be used instead of commands like `LaTeX HelloWorld` or `MacDVI HelloWorld` entered in the shell window.

2.5 Handling TEX errors

Now we will add an error to our document `HelloWorld.tex` just to see how TEX handles errors. Simply change your document to

```
\documentclass{article}
\begin{document}
Hello, world.
\foo
\end{document}
```

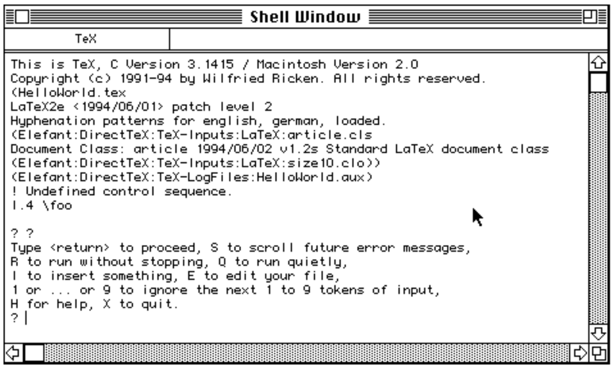


Figure 6: Shell window after TEX finds an error

The macro `foo` will cause an error because it is undefined. To see what happens typeset `HelloWorld.tex` again, using the `Typeset Main TeX File` menu item from the `TeX` menu. Now after a short time TEX will stop, displaying a question mark and the blinking cursor in the shell window (see figure 6). Simply type `?` and `Return` to get a list of options you have.
One of the options is typing `e` to edit your file. After typing `e` TEX stops the compilation and opens your document. Furthermore, it selects the line where it detects an error. Note that in this case this *is* the line where we made the error. However, there are lots of cases where TEX will detect an error only some time after the point where the error occurred.

- ⋄ TEX will always open your text file with the application that created it. Therefore you should always create your text files with your preferred text editor.

In our example remove the line and typeset the document again, thus getting the original output again, without errors.
Now you have seen a little example how to use DirectTEX. If you want to learn more about TEX now you should read one of the books mentioned earlier and use DirectTEX to try out the examples you probably will find in these books. By this way you will learn more about TEX and more about the usage of DirectTEX.

3 The DirectTEX *Pro* shell

Sorry, but the information in this chapter isn't up-to-date. Please wait for the final manual to appear.

3.1 Introduction

The DirectTEX *Pro* shell is an integrated environment for using TEX and METAFONT. It contains a simple text editor, a `dvi` file previewer and printer, a simple command language with a lot of build-in commands and a shell for executing commands by hand. It is possible to add menus to the menu bar by using the command language and to connect the items of these menus to commands to execute when that item is selected. So in principle every user is enabled to configure it's own environment.
Using the DirectTEX *Pro* shell is as straightforward as using nearly every Macintosh application. So the following sections will describe only the main aspects.
If you ever have problems using the DirectTEX *Pro* shell you might use the online help. Simply select the appropriate item within the help menu at the right side of your menu bar. DirectTEX *Pro* supplies help for all standard menu items, all windows and nearly all dialog boxes. Note that DirectTEX *Pro* is not able to supply online help for all menus added by the command language.

3.2 The Text Editor

The integrated text editor is very simple — it is based on the facilities of the operating system for editing text. You may edit up to 32K of text with this text editor. For small projects, this is sufficient, but for greater projects it is strongly suggested to use an external text editor like Alpha, MPW oder BBEdit. I personally use the integrated editor for example when writing a short letter or things like that.
The text editor has the usual commands available for editing text, namely cut, copy, paste and clear. You may select the font for screen display. The keyboard layout is similar to that of MPW, so using the arrow keys together with some modifier keys like shift or command always has a special meaning.
The text editor has no undo, so if you don't like that simply use an external editor. Another hint: Remember to save your text before you run it through TEX.

3.3 The Command Language

The command language integrated into the DirectTEX *Pro* shell has been designed to meet certain needs of TEX users. The most important thing about this command language is it's ability to handle strings containing any character of the standard Macintosh character set. Other command languages, like the one integrated in the MPW shell, are not able to do that, and therefore often have problems when dealing with arbitrary Macintosh file names.
The language is strictly line-oriented, that means that it is impossible to write commands that span over multiple lines. It supports command substitution, variable expansion and input/output redirection.
Executing a command is done in three steps of simple text manipulation: Command substitution, variable expansion and removal of special characters. The resulting text is interpreted as a sequence of words separated by spaces. The first word is the name of the commando to execute, all other words are the arguments for that command.

3.3.1 Command substitution

Everything between ` and ` is interpreted as a command and is replaced, including the `, by the output of this command. You may type something like

```
for i `files | grep '\.tex$'` [latex $i
```

to run all files ending with `.tex` in the current directory through the command `latex`. The DirectTEX *Pro* shell will interpret `files | grep '\.tex$'` as a subcommand and replace it by its output, namely a list of all files ending with `.tex`.

3.3.2 Variable expansion

DirectTEX *Pro* will interpret every combination of characters, numbers and the underscore behind a `$` sign as the name of an environment variable and replace it, including the `$` sign, by its value. Note that DirectTEX *Pro* will insert the value of the variable in a way that it will be interpreted as a single word, even if it contains spaces. DirectTEX *Pro* will quote the contents of the variable. Here is an example:

```
set foo_bar "1 2 3"
debug echo $foo_bar
```

will output

```
Debugging Information:
======================
Tool-Name = echo
Number of arguments = 1

List of arguments:
==================
    "1 2 3"

1 2 3
```

If the variable doesn't exist, the name is replaced by two single quotes `''`.

3.4 Special characters

There are some characters that have a special meaning:

- Single quotes `'`: Everything between two single quotes is taken verbatim and not interpreted any further. No command substitution and no variable expansion is taking place.
- Double quotes `"`: They may be used to group things together to form a single word. Spaces and tabs are taken verbatim and are not interpreted as separators anymore. However, command substitution and variable expansion is still active.
- A single `[`: This one may be used to take the remainder of the command line verbatim, up to the next newline character. This one is very handy for the `if` and `for` commands because these commands expect a complete command as a single argument. A small example you saw already under command substitution.
- The pipe characters | and ||: These two symbols may be used to separate two commands from each other. `a | b` will call command `b` only if command `a` return a status code of zero (which means success), and `a || b` only if `a` returns a status code not equal zero (which means failure).

- The input/output redirection characters >, $\geq$, Σ, >>, $\geq\geq$, $\Sigma\Sigma$ and <: These symbols may be used to redirect input or output from or to a file. < stands for standard input, > for standard output, $\geq$ for error output and Σ for both standard and error output. Using >>, $\geq\geq$ or $\Sigma\Sigma$ will not overwrite the file, but append the data to the file. Use something like `files > foo` to print a list of all files in the current directory into the file `foo`.

- The escape character ∂: Use this character to remove the special meaning of every other character. For example, use `echo hallo` ∂> `hallo` to get an output like `hallo > hallo`.

- The command substitution character `` ` ``: Everything between two of these characters is executed as a subcommand and replaced by the standard output of this command.

- The variable character `$`: This character marks the beginning of a variable name. Everything up to the next character that is neither a letter nor a number nor an underscore is taken as the name of the variable. It's a good idea to use a ∂ to signal the end of the name if a letter or underscore follows.

4 A Detailed View of DirectTEX *Pro*

Sorry, but the information in this chapter isn't up-to-date. Please wait for the final manual to appear.

4.1 The Environment Variables

4.1.1 General

The environment variables are maintained by the DirectTEX *Pro* shell. A variable named `<name>` is given the value `<value>` by typing

```
Set <name> <value>
```

If it didn't exist yet, it is automatically created by the DirectTEX *Pro* shell. Use the option `-x` to make the variable known generally.
The current value of an environment variable is typed to the screen by the command

```
Echo $<name>
```

In general,

```
$<name>
```

accesses the value of the variable. The names and values of all variables currently defined are listed by the simple command `Set`. You will also see the scope of the variable.

4.1.2 Explanation

All environment variables are defined in the file `DirectTeX Pro Preferences` which you will find inside the `Preferences` folder within your system folder. Note that DirectTEX *Pro* will create a default file if this file doesn't exist.
Now we come to the environment variables defined in the standard installation and their meanings. First we will list all variables that either contain some preferences or the name of a single file.

Name of variable	Meaning
`dt_VersionNumber`	Version of DirectTEX you are using
`dt_TeXFolder`	Pathname of the (server) TEX data folder
`dt_TeXLocal`	Pathname of the (local) TEX data folder
`dt_TeXProjectDir`	Directory of the current project
`dt_TeXProjectName`	Filename (without extension) of the current project
`dt_TeXProjectExt`	Extension of the current project
`dt_TeXFormat`	Current macro package, e.g. plain or LATEX
`dt_TeXCurrentFile`	Name of current TEX file within a project
`dt_MissingFonts`	Filename of the file containing missing fonts
`dt_ErrorFile`	Filename of the file containing error messages
`dt_ResidentFonts`	Filename of the resident font map file
`dt_DeviceDefs`	Filename of the device definition file
`dt_History`	Filename for the DirectTEX *Pro* command history file
`dt_TempExtensions`	List of extensions belonging to temporary files
`dt_DefaultFormat`	Default format to use
`dt_BatchMode`	Set to `on` to use TEX and METAFONT in batch mode

The variables `dt_TeXFolder` and `dt_TeXLocal` contain the pathnames of the TEX data folders. It is assumed that you will put all TEX related files into several folders within these data folders. This procedure is quite reasonable in order to keep track of the large number of TEX files accumulating in the course of time. When using the automated installation of TeX these folders will be called `DirectTeX` and will be located within the root directory of the hard disk you selected for installation. `{dt_TeXFolder}` and `{dt_TeXLocal}` should match if you are using a single installation of DirectTeX. For a network installation, they should be the names of the server data folder, containing the TeX tools and all standard input files, and the local data folder, containing all files needed by the local installation (e.g. the configuration files). Note that the installation program supplied with DirectTeX is able to install either the local or the network version of DirectTeX.

The variables `dt_TeXProjectDir`, `dt_TeXProjectName` and `dt_TeXProjectExt` contain the complete name of the current TeX project, split into pathname, filename without extension and extension. Thus the various TeX tools are called via

```
<tool-name> $dt_TeXProjectName
```

The tools automatically append the required suffices to filenames. The only exception: TeX itself is called via

```
TeX $dt_TeXProjectName$dt_TeXProjectExt
```

This allows to run files like `multicol.drv` through the standard TeX project mechanism.

`dt_TeXFormat` contains the name of the macro package to be used by TeX, the most common ones being `Plain`, `LaTeX` or `BigLaTeX`.

The variables `dt_TeXProjectDir`, `dt_TeXProjectName` and `dt_TeXProjectExt` are easily given appropriate values by opening a DirectTeX *Pro* project. This will give a value to `dt_TeXFormat` too. However, `dt_TeXFormat` can also be changed by selecting one of the menu items at the bottom of the `Formats` menu. You may clear all values by closing all project windows.

`$dt_MissingFonts` contains a list of all fonts **MacDVI** cannot find. This list will be displayed in the missing fonts window. Every time **MacDVI** starts, it will read this file. Then all missing fonts will be added to an internal list of missing fonts. When **MacDVI** finishes its work, it writes out the list of missing fonts. Therefore every missing font will occur only once. If you select the option `Create missing fonts online` in the preferences dialog, **MacDVI** will try to call `MFToPK` when it encounters a missing font. If `MFToPK` succeeds, no lines will be appended to `$dt_MissingFonts`.

`$dt_ErrorFile` is the name of the file that contains the error messages DirectTeX will print out for errors returned by the operating system that are not handled by DirectTeX. This file is the `SysErrs.err` file you will find in the MPW distribution. It contains the official error messages of Apple.

`$dt_ResidentFonts` is the name of a file containing a list of all fonts to be considered resident (resident meaning PostScript here). **MacDVI** scans this file to find out how to display a resident font on the various devices.

`$dt_DeviceDefs` refers to the file **MacDVI** needs to display fonts on your output devices. Please refer to the **MacDVI** section of this manual to learn more about devices, fonts and related things.

The next variables are used to access different kinds of files. The general scheme for their values is

```
'<pathname 1>,<pathname 2>,...,<pathname n>'
```

The TeX tools split such a value into the separate pathnames and work their way through the list starting with the first one in order e.g. to find a file. For example, TeX searches for its input files in the list `dt_TeXInFiles`.

Name of variable	Meaning
dt_TeXInFiles	TEX input files
dt_TeXOutFiles	TEX output files (dvi files)
dt_TeXLogFiles	TEX log files, the protocol files of TEX
dt_MFInFiles	METAFONT input files
dt_MFOutFiles	METAFONT output files (gf files)
dt_MFLogFiles	METAFONT log files, the protocol files of METAFONT
dt_ConfigFiles	Different configuration files for different tools
dt_FormatFiles	The format files, e.g. Plain.fmt
dt_HeaderFiles	Header files for use by dvips
dt_TFMFiles	The tfm files needed for input to TEX
dt_VFFiles	The vf files needed for input to TEX
dt_PKFiles	The pk files needed for input to **MacDVI**
dt_TempFiles	Temporary files
dt_ToolsPro	Search path for DirectTEX *Pro* shell tools

Please note: You can enter the separate pathnames either as complete pathnames starting with a volume name (e.g. `'Blackdog:TeX:TeX-Inputs:'`) or as incomplete pathnames starting with a colon (e.g. `':Documents:'` or even `''`). Incomplete names are automatically searched for within the current directory. The latter can be set using the menu `Directory` or the command `Directory <name>`. Opening a DirectTEX *Pro* project will automatically set the directory to the directory where the main file of the project resides.
All TEX tools build the complete name of a file from the pathname given and the filename. When `dt_TeXInFiles` is given the value

```
',Blackdog:TeX:TeX-Inputs:,Blackdog:TeX:TeX-Inputs:Documents:'
```

the command

```
TeX '&Plain Test'
```

causes TEX to search for the file `Test` according to the following scheme:

1. `Test.tex` in the current directory.
2. `Blackdog:TeX:TeX-Inputs:Test.tex`
3. `Blackdog:TeX:TeX-Inputs:Documents:Test.tex`

Of course, the file found first will be used. Furthermore, complete names will not be expanded anymore. The call

```
TeX '&Plain Blackdog:TeX:Test'
```

causes TEX to search for the file `Blackdog:TeX:Test.tex` only.
There is another feature: If a path in the list ends with `:*`, the TEX tools will automatically scan all folders contained in the given path. E.g. when `dt_TeXInFiles` is given the value

```
'Blackdog:TeX:TeX-Inputs:*'
```

TEX will search all folders contained within `Blackdog:TeX:TeX-Inputs`, but not the folders contained within these folders.
There is a simple way to turn off the path searching algorithm: All file names that begin with a colon (e.g. :Test) are searched in the current directory, and the pathnames found in the list are not searched for the file `Test`. You may also use something like `:More-Files:Foo` which will search the file `Foo` in the directory `More-Files` within the current directory.
The variable `dt_PKFiles` may contain some special characters that are replaced by the shell to construct the name of a font. Every `%m` will be replaced by the mode the font was created with (e.g. `LaserWriter`), every `%f` by the name of the font (e.g. `cmr10`), every `%d` by the resolution of the font (e.g. `329`) and every `%p` by the extension `pk`. Note that the different parts of `$dt_PKFiles` describe the name of a `pk` file, not a folder.

4.2 Managing archives of files

DirectTEX *Pro* is able to combine files into archives and to search archives for a given file. Archives may be created by using the `New...` command from the `File` menu and selecting `Archive` as the file type. Once you created an archive file you can modify its contents by opening it and using the appropriate buttons in the archive window.
Generally there is no need for archive files on the Mac because the number of files that may reside within one directory isn't limited. However, there is one good reason to use archive files: Allocating space for files on volumes is done by the MacOS only in multiples of the minimum allocation size. Even a file that contains only one byte will occupy the minimum allocation size in bytes on the volume. This minimum allocation size depends on the size of the volume. It is at least 1K and on a 2GB partition it becomes 16K !!! So every file will occupy at least 16K on such a hard disk, even if it is much smaller.
There are a lot of files used by TEX that are much smaller than 16K, e.g. the `tfm` files (≈ 1K). These files will quickly fill up your hard disk because each of them will eat 16K on a 2GB partition. This is the point where archives come into play: In archives the files only fill up the space they really need (plus some bytes additional information), and on the hard disk there is only one big file that wastes max. 16K of space. This mechanism reduces hard disk space usage a lot (even on smaller volumes). Try to use the Finder's `Get Info` command on both a folder containing a lot of small files and an archive file containing the same files.

◇ Please note that all data is stored in the resource fork of archive files and you will loose its contents if you transfer such a file on a file system that doesn't support resource forks (e.g. UNIX, MS/DOS, …

Archive files may be integrated into the path searching algorithm used by the DirectTEX *Pro* shell and the tools by adding a `@` to the beginning of the path name of the archive file and a `:` to the end. Here an example: If `Blackdog:TeX:TeX-Inputs:Foo` is an archive file, simply add a path `@Blackdog:TeX:TeX-Inputs:Foo:` to the variable `dt_TeXInFiles` to cause TEX to search for input files in this archive file.

4.3 DirectTEX memory usage

The memory management of DirectTEX is quite different from the memory management other applications use. Normally, every application on the Macintosh has its own memory partition from which it allocates memory it needs. The size of this memory partition may be adjusted using the Finder.
The DirectTEX *Pro* shell follows this conventions. If you get messages from the DirectTEX *Pro* shell that there is not enough memory than you may try increasing the partition size of the shell. However, the tools use a different memory allocation scheme:
Every tool is an application that has its own memory partition. Only small memory blocks needed by a tool are allocated from this memory partition (e.g. for opening files, …). The large memory blocks are requested as temporary memory from the operating system. That means they are taken neither from the memory partition of the DirectTEX *Pro* shell nor the memory partition of the tool, instead they are taken from the memory not occupied by any application, the free memory. So if you get a message like `Unable to allocate xxx bytes of memory.` after starting a tool like TEX you may fix the problem by closing other applications to increase the size of free memory available.

4.4 Adjusting the TEX Tools

For the present section, it is assumed that you are familiar both with DirectTEX and ResEdit, the resource editor by Apple.

The size of the memory partition every tool uses for small memory blocks should usually be sufficient. However, you may change it using ResEdit, editing the `SIZE` resource of the tool. You may not use the finder because the finder recognizes the tools as DirectTEX documents instead of applications. This is necessary because the tools may not be started as ordinary applications, but only from the DirectTEX *Pro* shell.
The amount of memory available to most of the TEX tools may be modified by editing the `CMEM` resources using ResEdit. Most of the tools contain two different `CMEM` resources: ID=128 contains the original values as they are found in the original source code, and ID=129 contains the current values which are often enlarged. Note that TEX and METAFONT contain two different `CMEM` resources that are both used: ID=129 contains the values that are used by `VirTeX` or `VirMF` and ID=130 are used by `IniTeX` or `IniMF`.
If you modify some of the values found in the `CMEM` resources be sure that all values must adhere to the range and internal consistency requirements imposed by the program you modify. In case of doubt you should refer to the original `web` files where you will find an explanation of these constants and their limitations.
You may also modify the names of the environment variables the tools use. These names are defined in the resource `ENVI`, ID=128. Note that all tools that use some environment variables contain them in such a resource. However, it is highly recommended not to change these names unless you're completely sure about what you're doing.
In TEX, BibTEX and METAFONT, there are resources called `xchr`, `xprt`, `xcls`, `xicl` and `xlcl`. These resources define some arrays that originally were hard-wired into the appropriate tools. The `xchr` resource defines the *xchr* array used in TEX, BibTEX and METAFONT to map between internal and external character sets. The `xprt` resource defines which characters (from the external character set) are unprintable and should be replaced by a control code like `^^w`. METAFONT contains the resource `xcls` which defines the character-class for each character. BibTEX contains the two resources `xicl` and `xlcl` which define the ID and lex class for every character. You should refer to the original `web` source files to learn more about these arrays.

5 Description of all DirectTEX *Pro* commands

5.1 TeX

Syntax: `TeX [-d] [-i] [-s] 'TeX commands'`
Variables: `dt_TeXInFiles`, `dt_TeXOutFiles`, `dt_TeXLogFiles`, `dt_FormatFiles`, `dt_TFMFiles`

This is the actual TEX compiler. There are two different versions of TEX merged into a single program. Usually the first one is called `IniTeX` and the second one `VirTeX`. However, in DirectTEX `IniTeX` is called by `TeX -i`, and `VirTeX` is called by `TeX`. The command line options are:

`-d` Dump memory usage

`-i` Use `IniTeX` rather than `VirTeX`

`-s` Gather statistic information

`TeX` will be the most often used program in this package. It translates TEX files. The associated call in its most simple form is

```
TeX '&<format-file> <tex-file>'
```

The menu item `TeX...` provides this call in a more efficient manner.
`TeX` exists in two versions, differing in the memory size provided: A standard one and an enlarged one, `BigTeX`. Usually you should use the standard version, since it runs faster and is 100% compatible with other TEX implementations. If during translation of a TEX file you are confronted with the message `TeX capacity exceeded...` you can either use the big version or enlarge the amount of memory for the standard version. In some rare instances you might wish to enlarge even the memory of the big one. It will be explained later how the enlargement is done.
A detailed description of TEX and LATEX is to be found in [Knu86], [Sch88], [Kop91], and [Kop90].

5.2 MF

Syntax: `MF [-d] [-i] [-s] 'MF commands'`
Variables: `dt_MFInFiles`, `dt_MFOutFiles`, `dt_MFLogFiles`, `dt_FormatFiles`

METAFONT for the Macintosh. Its usage is in complete analogy to that of TEX. You do not need to know anything about the use of METAFONT when you restrict yourself to the normal needs, i.e. generating missing fonts (`pk` files), since this can be done in a fully automated fashion using script files. In case you want to know more about METAFONT, you find a very good introduction in [Kop90].

5.3 MFToPK

Syntax: `MFToPK [-b <base>] [-m <mode>] [-x <mag>] [-c <commands>] [-d] [-i] [-s] <mf-file>`
Variables: `dt_MFInFiles`, `dt_MFOutFiles`, `dt_MFLogFiles`, `dt_FormatFiles` `dt_PKFiles`, `dt_TFMFiles`, `dt_PKFormat`

This is a combination of METAFONT and `GFToPK`. `MFtoPK` will create a `pk` file and will put this `pk` file in the location specified by `dt_PKFiles`. It will put the `tfm` file in the location specified by `dt_TFMFiles`. All needed directories are created if they do not exist. The command line options are translated into METAFONT commands as follows:

-b &<base>

-m mode=<mode>;

-x mag=<mag>;

-c <commands>

-d Dump memory usage

-i Use IniMF rather than VirMF

-s Gather statistic information

<mf-file> input <mf-file>

For example, the command

```
MFToPK -b Plain -m Standard -x '\magstep0' -c 'nonstopmode;' cmr10
```

is equivalent to using the command

```
MF &Plain \mode=Standard; mag=\magstep0; nonstopmode; input cmr10
```

and calling GFToPK after that and moving the resulting pk and tfm files to the proper locations.

5.4 MacDVI

Syntax:	MacDVI [-c] [-s|-d|-p] [-v <device>] <dvi-files>
Variables:	dt_TeXOutFiles, dt_TFMFiles, dt_VFFiles, dt_PKFiles, dt_TeXInFiles, dt_ConfigFiles, dt_PKFormat, dt_DeviceDefs, dt_ResidentFonts, dt_MissingFonts

This is the interpreter for the dvi files written as output from TEX. The driver allows you to preview the dvi files on the screen and have them printed on a variety of printers. A detailed description of the many features of this TEX tool is to be found in chapter 6.

5.5 BibTeX

Syntax:	BibTeX <aux-file>
Variables:	dt_TeXInFiles, dt_TeXLogFiles

This program facilitates quasi-automatic generation of a bibliography when using LATEX. The bibliography is generated from the citations referenced in your TEX file (BibTEX reads them from the .aux file), a separate literature data base (.bib file), and a specification of the reference syntax (.bst file). BibTEX gets the names of the .bib and .bst files from the .aux file as well. Thus BibTEX is simply called by the command

```
BibTeX <file>
```

<file> has to be the name of the main TEX file without its suffix .tex. Schematically, the procedure is as follows:

- BibTEX opens the file <file>.aux in order to access the citations referenced and to read the names of the data base <base>.bib and style <style>.bst.
- BibTEX reads the file <style>.bst in order to determine the form (syntax) of the literature references and an eventual sorting order.

- BibTEX reads the complete literature references corresponding to the citations in your text from the data base `<base>.bib`.
- BibTEX creates its output file `<file>.bbl`. It contains the TEX commands necessary to write the complete bibliography for your text.

The usage of BibTEX is very easy following the items of the TEX menu: First use the menu item `Select TeX Project...` to choose your main TEX input file. For the macro package, use `LaTeX`. Translate your TEX files via `TeX...` Run BibTEX via `BibTeX...` Two further calls of TEX are necessary in order to straighten out the cross references in your text.

5.6 MakeIndex

Syntax:	`MakeIndex [option...] <idx-files>`
Variables:	`dt_TeXInFiles,dt_TeXLogFiles`

LATEX allows you to generate an index for your text from markers set at the desired places in the input file. When doing so, TEX writes a file `<file>.idx` that contains all index entries in the order of occurrence. A useful register is created from this file by sorting the entries appropriately, i.e. according to the alphabet and groups of entries. This is what `MakeIndex` does. Via

`MakeIndex <file>`

`MakeIndex` is asked to open the file `<file>.idx`, sort the entries encountered and output them to the file `<file>.ind` in a formatted fashion.
`MakeIndex` offers many more options. These — and some rules for the form of index entries — can be found in a more complete description of `MakeIndex`, as given e.g. in [Kop90]. The options in detail:

`-i` This option tells `MakeIndex` to use the standard input (i.e. the pseudo file `Dev:Stdin`) as its input. Thus you can direct the output from other tools to `MakeIndex`.

`-l` Makes the sorting procedure independent of eventual spaces occurring in the index entries.

`-r` Normally, identical index entries occurring on three or more successive pages cause one index entry of the type *from – to*. When this option is specified, all page numbers are listed explicitly.

`-q` Suppress the screen messages from `MakeIndex`.

`-c` This option is concerned with the sorting procedure: Spaces as the first and last letters of an index entry will be ignored, all other spaces are compressed, i.e. several successive spaces are output as one space only.

`-s` Changes to the shape of the index register are possible by using a style file, the option `-s` is to be followed by the name of the style file to be used.

`-o` This option allows you to specify the name of the index register file to be written. Normally, the output file will have the same name as the input file, the suffix being `.ind` instead of `.idx`. Specify `-o <name>` to get a different one.

`-t` Analogous to `-o`, but regarding the name of the log file.

`-p` This option allows you to specify the first page number of the index register. Either specify an integer number, or one of the three keywords `even`, `odd` and `any`. When one of the three latter is given, `MakeIndex` reads the last page number of your document from the `.log` file. The first page number of the index register then will be either the next `even` number, the next `odd` or the next (`any`) number.

-g Switch on German sorting. This option requires an appropriate style file, since in German TEX files umlauts are escaped by "; however, MakeIndex uses just this character as an escape sign of its own. The style file has to specify a different escape sign for MakeIndex.

5.7 DVICopy

Syntax:	DVICopy [-d] <dvi-file>
Variables:	dt_TeXOutFiles, dt_TeXOutFiles, dt_TFMFiles, dt_VFFiles

This program may be used to remove all virtual fonts from a dvi file. A backup of the original file will be saved in the same directory as the original file, the name ending in .bak instead of .dvi. Specify -d to change some options before copying. While **MacDVI** is able to process dvi files that contain virtual fonts other dvi drivers are not able to do that, so it is a good idea to "de-virtualize" a dvi file before giving it to other people.

5.8 DVIType

Syntax:	DVIType [-l <level>] [-s <start>] [-p <max>] [-r <res>] [-m <mag>] <dvi-file>
Variables:	dt_TeXOutFiles, dt_TFMFiles

This little program by DONALD E. KNUTH is the origin of all dvi drivers. Its purpose is to test dvi files for correctness and to demonstrate to programmers the basics of a dvi driver. Note that you must specify the options for displaying the dvi file on the command line, there is no dialog in which you can choose these options.

5.9 GFToDVI

Syntax:	GFToDVI [-o <dvi-file>] <gf-file>
Variables:	dt_MFOutFiles, dt_TFMFiles, dt_TeXOutFiles

This program converts the output files of METAFONT, the gf files, to dvi files which can then be printed nicely. An introduction to GFToDVI is given in [Kop90]. The impatient user might try the commands

```
MF '&Plain \mode=proof; mag=1.0; input cmr10'
GFToDVI cmr10.2602
MacDVI cmr10.2602
```

5.10 GFToPK

Syntax:	GFToPK [-o <pk-file>] <gf-file>
Variables:	dt_MFOutFiles, dt_PKFiles

This is a auxiliary program needed to generate fonts using METAFONT. The gf files are the output from METAFONT. Their principle is quite similar to that of the dvi files, i.e. they contain a number of commands to be interpreted by a type of processor that creates the pixel representation of each character. Previously, the gf files were converted to pxl files containing the pixel representation directly as a bit map, resulting in very large files. Presently, the gf files are still converted to bit maps; however, these are packed afterwards, resulting in the pk files. This is what the program GFToPK does.

Remark on the automatic generation of file names by the TEX tools METAFONT and GFToPK: The call

```
MF '&Plain \mode=<mode_def>; mag=<magnification>; input <name>'
```

will create a file `<name>.<size>.gf` in the directory `dt_MFOutFiles`. Thus the file name contains the font name as well as the font size. The latter is calculated according to

$$\textit{size} = \textit{horizontal resolution} \times \textit{magnification}$$

rounded to the next integer number. When called by

```
GFToPK <name>
```

`GFToPK` then creates the file `<name>.<size>.pk` in the directory `dt_PKFiles`. The `pk` file does not necessarily have a name in accord with the structure being requested by the environment variable `dt_PKFormat`. It is therefore highly recommended to use `MFtoPK` instead of METAFONT and `GFToPK`.

5.11 PKToGF

Syntax:	`PKToGF [-o <gf-file>] <pk-file>`
Variables:	`dt_MFOutFiles, dt_PKFiles`

This tool is the counterpart of the `GFToPK` tool. Use it to convert `pk` files back into `gf` files.

5.12 GFType

Syntax:	`GFType [-m] [-i] <gf-file>`
Variables:	`dt_MFOutFiles`

The analog of `DVIType`. It serves as a development basis for programs trying to interpret `gf` files and as a verification program for `gf` files. The options:

`-m` Switch on the output of the mnemonics found in the `gf` file.

`-i` Use this option when you want to look at the pixel images of the characters. However, the TEX tool `GFToDVI` might be a lot more suitable.

5.13 PKType

Syntax:	`PKType <pk-file>`
Variables:	`dt_PKFiles`

The analog of `DVIType`. It serves as a development basis for programs trying to interpret `pk` files and as a verification program for `pk` files.

5.14 PLToTF and TFToPL

Syntax:	`PLToTF [-o <tfm-file>] <pl-file>`
	`TFToPL [-o <pl-file>] <tfm-file>`
Variables:	`dt_TFMFiles`

These two tools serve to convert `tfm` files (the TEX font metric files) to `pl` files (property list files) and vice versa. During normal use of TEX, they are not required. Further details are to be found in the program documentation which you would have to generate from the original source files.

5.15 VFToVP and VPToVF

Syntax:	VFToVP [-o <vpl-file>] [-t <tfm-file>] <vf-file>
	VPToVF [-o <vf-file>] [-t <tfm-file>] <vpl-file>
Variables:	dt_VFFiles, dt_TFMFiles

These two tools are in complete analogy to PLToTF and TFToPL. However, they will operate on virtual fonts instead of real fonts.

5.16 MFT

Syntax:	MFT [-c <change-file>] [-s <style-file>]
	[-o <tex-file>] <mf-file>
Variables:	dt_MFInFiles, dt_TeXInFiles

Pretty-printer for mf files. It creates TeX input files from mf files (the METAFONT input files); the TeX output does look quite nice, but is unfortunately not compatible with LaTeX.

5.17 Compress

Syntax:	Compress [-e] [-q] [-d] [-t] [-n <num>] <files>
Variables:	*None*

This little program enables you to compress certain files. TeX and METAFONT are able to read compressed input files. This saves disk space because there are a lot of TeX and METAFONT input files, and METAFONT especially certainly isn't needed very often. Compress uses the compression algorithm of the UNIX program Compress. The typical amount of compression is about 50%. Note that Compress is only able to compress the data fork of a file, the resource fork will be copied unchanged unless you specify the -d option in which case the resource fork will not be copied and therefore be destroyed.

-e Expand the files rather than compressing them.

-q Don't display a progress dialog.

-d Kill the resource fork, no warnings will appear. Be careful!!

-t Ignore the type of the file. This makes it possible to compress files that are not of type TEXT and to expand files that are not of type Com+. But be careful if you use this option together with the -d option: You can easily destroy a file (e.g. specifying Compress -d -t Foo with Foo being a program containing all its code in the resource fork Foo will be killed and cannot be used again).

-n <num> Use this option to specify the compression rate of Compress. <num> may have values from 9 to 16. The bigger the value of <num> is the more files get compressed, but the more memory Compress and all tools that expand such files need to do their job.

5.18 TextTrans

Syntax:	TextTrans [-mac|-dos|-unix|-nochange]
	[-q] [-d] [-t] <text-files>
Variables:	*None*

TextTrans Usage: .

TextTrans converts text files between the various formats. The character codes used to represent the end of a line are different on different systems: UNIX uses only a linefeed (0x0A), MS-DOS uses a carriage-return-linefeed (0x0D 0x0A) and the Macintosh uses only the carriage-return (0x0D). So this little program is able to convert text files from every format to every other format. The options:

-mac Convert the text file into Macintosh format.

-dos Convert the text file into MS-DOS format.

-unix Convert the text file into UNIX format.

-nochange Copy the text file unchanged.

-q Don't display a progress dialog.

-d Kill the resource fork, no warnings will appear. Be careful!!

-t Ignore the type of the file. This makes it possible to translate files that are not of type TEXT. But be careful if you use this option together with the -d option: You can easily destroy a file (e.g. specifying TextTrans -d -t Foo with Foo being a program containing all its code in the resource fork Foo will be killed and cannot be used again). Also be sure that the data fork of the file contains text, not other information (e.g. translating a dvi file using TextTrans -t foo.dvi will probably destroy the dvi file.

A hint: Use TextTrans -nochange -d foo.tex to strip the resource fork from the file foo.tex. This will save disk space because MPW always saves the window position in the resource fork, and this information isn't needed for a lot of files, e.g. all METAFONT input files.

5.19 ClipToRez

Syntax:	ClipToRez [-e] [-n] <pict-file>
Variables:	*None*

This program saves a PICT from the clipboard in the file ¡output-file¿, putting the PICT into the resource fork and an appropriate %%BoundingBox comment into the data fork. This is the format required by **MacDVI** and the epsf.sty macro package to include pictures into documents (see **MacDVI** manual for more details). This command is also able to convert the PICT to (simple) PostScript code, thus producing full EPSF files. The correct suffix .epsf or .rsrc is automatically appended unless you specify the -n option. The options:

-e Use this option to enable the PICT to PostScript conversion.

-n Use this option to prevent ClipToRez from adding an extension like .epsf or .rsrc to the output file name.

5.20 `Tangle`, `CTangle`, `MTangle`, `Weave`, `CWeave` and `MWeave`

Syntax:
```
Tangle [-c <change-file>] [-p <pool-file>]
[-o <pascal-file>] <web-file>
CTangle [-s] [-c <change-file>]
[-o <c-file>] <cweb-file>
MTangle [-c <change-file>]
[-o <c-file>] <mweb-file>
Weave [-x] [-c <change-file>]
[-o <tex-file>] <web-file>
CWeave [-s] [-x] [-f] [-c <change-file>]
[-o <tex-file>] <cweb-file>
MWeave [-x] [-c <change-file>]
[-o <tex-file>] <mweb-file>
```
Variables: *None*

These are the TEX tools needed for using the programming language `web`. All standard TEX tools are written in this language, which basically combines source code, a simple change mechanism and associated documentation. Generation of the Pascal code from the `web` code is done by `Tangle`, `Weave` generates a TEX file containing the documentation. The corresponding tools operating on `cweb` files are `CTangle` and `CWeave` and on `mweb` files are `MTangle` and `MWeave`.

5.21 `PatGen`

Syntax:
```
PatGen [-p <pattern-file>] [-t <translation-file>]
[-o <output-file>] <dictionary-file>
```
Variables: `dt_TeXInFiles`

This program may be used to generate hyphenation patterns for use with TEX out of a dictionary of hyphenated words. Most standard hyphenation patterns have been built with this program. The options:

`-p` This option allows you to read in hyphenation patterns from a file before generating new ones.

`-t` This option allows you to specify the name of a translation file. This file should contain all language-specific character translations and the correct values for `\lefthyphenmin` and `\righthyphenmin`.

`-o` This option allows you to specify the name of the output file. In this file you will find the patterns read in by the `-p` option and the new patterns generated from the dictionary file.

If you are planning to use `PatGen` for some new language, you are advised to contact also the Technical Working Group on Multiple Language Coordination; they will be able to help you, and perhaps have done those patterns already (you contact the chair of the group at `Yannis.Haralambous@univ-lille1.fr`).

6 Description of MacDVI

Sorry, but the information in this chapter isn't up-to-date. It still describes the old MPW version of DirectTEX. Please wait for the final manual to appear.

6.1 General Information about MacDVI

The purpose of any `dvi` driver is to interpret the device-independent files written by TEX and make them visible on a number of output devices including the screen and printers. Thus this program is the most computer system-dependent of all TEX tools. **MacDVI** provides facilities for output to the Macintosh screen, to any printer that is compatible with the Macintosh (i.e. QuickDraw printers, PostScript printers, Fax machines, …), and to disk-resident files (PostScript files).
MacDVI supports three different font formats, the `pk` (packed pixel) fonts generated by `MFtoPK` (or METAFONT and `GFToPK`), PostScript or TrueType fonts, and — being the most important but presently also most unknown concept — the virtual fonts. The former two of them need not be explained any more, while virtual fonts deserve (at least) one sentence of explanation: This concept provides much more flexibility in constructing letters and fonts from various other fonts (either existing physically or being virtual fonts themselves) and even entire sequences of TEX commands.
TEX originally is a type-setting program, not a graphics program, although it offers some rather limited graphical capabilities. Thus many TEX drivers use the `\special` command to provide enhanced graphics. **MacDVI** is no exception in that it allows you to include any PostScript code (e.g. graphics, but also anything else) or to include QuickDraw `PICT` files (representing pictures only).

- ◇ **MacDVI** is implemented as tool running under MPW and as tool build into the DirectTEX *Pro* shell. In fact, **MacDVI** is an integral part of the DirectTEX *Pro* shell.

6.2 Basic Concepts of MacDVI

6.2.1 The Device Concept

MacDVI knows logical and physical devices. The logical devices **MacDVI** knows are `Preview`, `Printer` and `DiskFile`. They only tell **MacDVI** where to redirect its output. The physical devices **MacDVI** supports are declared in so-called *device descriptions*. These descriptions contain a lot of information about the device, e.g. its resolution. The device descriptions and the remapping between logical and physical devices is to be found in the device description file.
You tell **MacDVI** which logical device to use by specifying it on the command line. The three devices known to **MacDVI** and the appropriate command line options are:

Name of device	Command line option
Preview	`-s` or none
Printer	`-p`
DiskFile	`-d`

The `Preview` device is used to display `dvi` files on the Macintosh screen within a standard Macintosh window. You may scroll the display, change the magnification, select any page of the `dvi` file and get some information about the `dvi` file.
The `Printer` device sends the output to a printer connected to your Macintosh, either via a network like AppleTalk or one of the serial ports. **MacDVI** supports PostScript and QuickDraw printers.
The `DiskFile` device should be used when you want the output of **MacDVI** to be written to a file. **MacDVI** will create a PostScript file with this device.

You may specify -c with any of these options to tell **MacDVI** to check only for the presence of fonts for the particular device, not to display the dvi file on the device. **MacDVI** will always output the commands needed to create a missing font, so you may use this option to output the commands needed for all missing fonts in your document.

◇ Using `Open...` from the `File` menu in the DirectTEX *Pro* shell and selecting a `dvi` file will invoke **MacDVI** to preview the file, without producing any output to the console.

6.2.2 The Device Description File

MacDVI needs a file that contains different things, namely

- The remapping between logical and physical devices.
- A list of `mode_def` names and resolutions that are known to METAFONT.
- The description of the physical devices.

MacDVI will use the file that is referenced by the shell variable `dt_DeviceDefs`. Use the command `Open $dt_DeviceDefs` to open this file and see which devices are defined. The syntax of this device description file is as follows:

- Remapping between logical and physical devices:

```
Preview := <name> | default;
DiskFile := <name> | default;
Printer := <name> | default;
```

  You may tell **MacDVI** an explicit physical device to use for a given logical device by specifying the name of the physical device, or you may tell **MacDVI** to use the name of the currently selected printer as the name of the physical device by specifiying `default`. Note that you may change the currently selected printer by using the *Chooser* which is part of your system software.

- **MacDVI** needs to know which `mode_defs` are known to METAFONT. METAFONT generates fonts using a `mode_def` which tells METAFONT the resolution of the font and some other things that are uninteresting to **MacDVI**. However, **MacDVI** has to know which `mode_defs` METAFONT knows and which resolution a given `mode_def` represents. Here is how to tell **MacDVI** about these `mode_defs`:

```
mode_defs =
    <name>:=(<h-res>,<v-res>);
        ...
    <name>:=(<h-res>,<v-res>);
enddef;
```

- Now we come to the syntax of the device descriptions. They must contain some data common to all devices and may contain additional data only needed by specific devices. The general syntax of a device description is

```
device_def <name> =
    <identifier>:=<value>;
        ...
    <identifier>:=<value>;
enddef;
```

The following identifier/value combinations are allowed:

```
device_type := QuickDraw | PostScript | default;
mode_def := <name>;
shift_origin := (<h-shift>,<v-shift>);
only_bitmaps := true | false;
conserve_vm := true | false;
res_check := true | false;
actual_size := <x>:<y> | default;
page_order := <order>;
```

The mode_def identifier is required for all physical devices. You must tell **MacDVI** about the mode_def connected to the physical device. The device_type identifier is optional, if you omit it (or specify default) **MacDVI** will generate both the PostScript and QuickDraw representation of your dvi file when printing the file on that physical device. The other identifiers have the following meaning:

- shift_origin may be used to shift the origin of the TEX page on the physical page. Normally the origin is exactly 1″ from the top and 1″ from the left edge of the paper. However, the printer driver you are using may be slightly inexact, so you may use this option to shift the origin to the exact place. Note that h-shift and v-shift must be specified in printer pixels.
  - ◇ The preview device simply ignores the shift_origin command because this device always produces exact output.
- only_bitmaps may be used to force **MacDVI** to use bitmap-only printing. In this mode the whole page is created as a bitmap in memory, and this bitmap will be send to the printer via the current printer driver. For documents that do only use pk fonts this option is obsolete because these documents always will be printed as bitmaps. However, for documents that use PostScript fonts this option may be useful for some printer drivers that have speed problems when printing such documents.
- conserve_vm is used only for PostScript devices. If your PostScript printer has not enough memory the PostScript code normally generated by **MacDVI** may overflow your printer's memory. In such a case you should specify conserve_vm := true to tell **MacDVI** to modify the PostScript code so that the memory of the printer is released after each page. If this fails, too, you should use dvips to generate the PostScript code for your printer.
- res_check should be used with caution: By default **MacDVI** will check that the printer is able to print at the resolution that has been specified by the mode_def. However, certain printer drivers (especially older ones) do not support this check, in such a case you may use res_check := false to prevent the check. However, it may be that the driver uses the wrong resolution in such a case, so be prepared to get strange results.
- actual_size is used only for previewing dvi files. It tells **MacDVI** which magnification to use as default magnification. By default **MacDVI** will try to view the dvi file in original size on the screen. You may override this by specifying one of the Options/Magnification menu items (e.g. 1:3 or 4:1).
- page_order may be used to specify how double-sided printouts should be made: <order> must be a string that looks like 1f2r or 2f1f. This string tells **MacDVI** how to print odd pages (1) and even pages (2), namely f(orward) or r(everse), and which pages should be printed first.

6.2.3 Resident Fonts

MacDVI supports different font formats: The usual `pk` fonts, virtual fonts and PostScript or TrueType fonts. Using `pk` fonts and virtual fonts is straightforward, no additional information is needed for **MacDVI**. However, when using PostScript or TrueType fonts and a few virtual fonts, **MacDVI** needs some more information to display them correctly on the different devices. TeX himself doesn't make this difference, because TeX only needs information about the *metrics* of the font, not the actual images. Therefore TeX only needs the `tfm` (*TeX font metrics*) files.
Which files **MacDVI** needs depends on the type of font: For `pk` fonts, only the `pk` files in the proper sizes are needed. For a virtual font, **MacDVI** only needs the `vf` file, which contains the information on how to build up characters out of TeX commands and other fonts. For PostScript fonts, **MacDVI** needs the `tfm` file to position the characters correctly on the page and some information about how to display the characters on different devices. This information is stored in the file referenced by the shell variable `dt_ResidentFonts`. Every font listed in this file is considered to be a resident font. The information listed in the `$dt_ResidentFonts` file is:

Title	Meaning
Name	The name used for the font in the `dvi` file
PS Name	The name to use for PostScript output
Mac Name	The name to use for QuickDraw output
Flags	Some flags needed by **MacDVI**
PS Code	Additional PS Code for PostScript output

The first three entries specify the remapping between the `dvi` file name and the PostScript and QuickDraw name of the font.
The following flags must be specified:

- **MacDVI** is able to translate the character codes found in a `dvi` file to other character codes by using an encoding vector. This encoding vector must be used when displaying `dvi` files on QuickDraw devices. The reason is that the Macintosh screen fonts use a different encoding than the Adobe PostScript fonts, and all `vf` files supplied with DirectTeX map the TeX encoding to the Adobe encoding. The encoding scheme may presently be either `MAC` to map to Mac encoding or `CMR` to map to the encoding used for the Blue Sky CMR PostScript fonts. If you use any other name, **MacDVI** will use a standard encoding vector without any translation. You should use `nil` if no encoding should be used.
- The QuickDraw style: Here you tell **MacDVI** which style to use for QuickDraw display of the font. The following styles are allowed: P (Plain), B (Bold), I (Italic), U (Underline), O (Outline), S (Shadow), C (Condensed), E (Extended).
- Downloading of fonts: You may specify wether the font should be downloaded to the printer (specify T (true)) or not (specify F (false)). Note that **MacDVI** uses the standard mechanism of the printer driver to download the font, so you should have either a TrueType or PostScript Type 1 file of the font installed in your system folder.
  - ◇ **MacDVI** is only able to automatically load fonts into the printer that are correctly installed in your system. To check wether a given font is installed in your system you may use e.g. the desk accessory `Key Caps`. Simply check wether the font may be displayed or not.

The PostScript code is optional, you may use some PostScript code here to generate fonts by modifying some other fonts. See the file `$dt_ResidentFonts` for some examples.

◇ The `$dt_ResidentFonts` file shipped with this installation contains all entries needed for the standard PostScript fonts (the fonts found in every Apple LaserWriter). You shouldn't alter the file unless you are sure about what you're doing.

6.3 MacDVI Windows

MacDVI displays the `dvi` files on the Macintosh screen within standard Macintosh windows. Note that the number of open windows is only limited by the amount of memory you have. You may enlarge the partition size of DirectTEX *Pro* (or MPW) by using the Finder's `Get Info` menu item.
The **MacDVI** window contains the usual scroll bars to scroll the window contents, an info bar displaying some information like current page number and cursor position and two additional controls to select any `dvi` page you like. The additional window controls and their meanings are (see also figure 7):

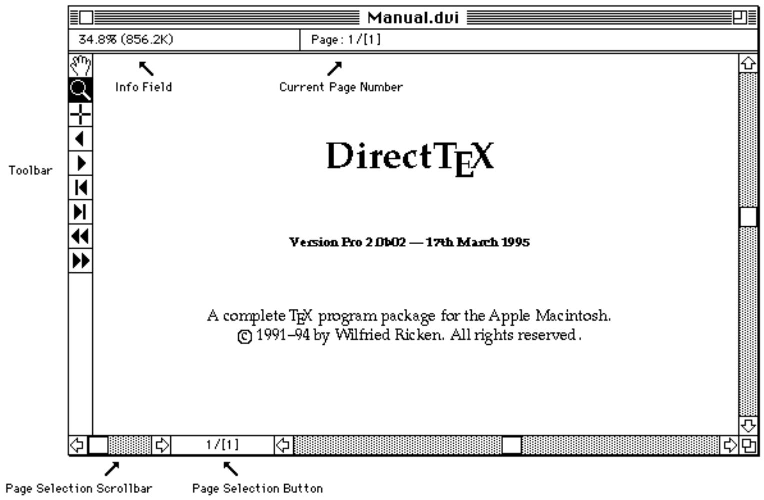


Figure 7: The **MacDVI** Window

- Info Field: In this field **MacDVI** will display some information. You may select the type of information to display by clicking into this field (a pop-up menu will appear).
- Current Page Number: Here the page number of the current page is displayed. Note that the first number is the `dvi` page number and the next number(s) within the brackets is (are) the TEX page number(s).
- Page Selection Scroll Bar: You may use this scroll bar to quickly select a `dvi` page for preview. Use the scroll bar until the desired `dvi` page is displayed to the left of the scroll bar.
- Page Selection Button: Click on this button to switch to the `dvi` page displayed within this button. Pressing the Return or Enter key is equivalent to clicking on this button.

MacDVI offers you different tools to manipulate the window display and to give you some info about a `dvi` file. The tools may be selected from the toolbar on the left side of the window. Additionally, the toolbar contains some buttons to navigate through your document. The tools and buttons from top down:

- Hand Tool: Move the page around without using the scroll bars.
- Magnifiyng Glass Tool: Magnify the portion of the page under the cursor. Use the option key to increase magnification instead and the shift key to decrease magnification instead.
- Cross Tool: Display information about characters and rules.

- Left Arrow: Display previous page.
- Right Arrow: Display next page.
- Left Arrow Bar: Display first page.
- Right Arrow Bar: Display last page.
- Leftleft Arrow: Step display history backwards.
- Rightright Arrow: Step display history forward.

MacDVI supports different keyboard shortcuts to help you navigating through your document. You may use the arrow keys to scroll through your document. Use the arrow keys in combination with the shift key to speed up scrolling. Note that **MacDVI** will automatically select the next (previous) page if you scroll down (up) using the arrow keys and you reached the bottom (top) edge of the page. This allows you to read a whole document on the screen without using the mouse. **MacDVI** also supports the enhanced keyboard keys like `home`, `end`, `pageup`, `pagedown`.

You may use the keys `1` to `9` to move the window display. Look at the keypad to understand how the display will be affected (7 displays the top-left corner, 6 displays the right edge, centered vertically, ...). Other keys on the keyboard have other meanings: Use `+` to select the next page, `-` to select the previous page, `*` to zoom in, `/` to zoom out, `0` to switch to magnification 1:9 and `.` to switch to magnification 4:1.

◇ If you're using the DirectTEX *Pro* shell, you may configure the keyboard by using the `SetKey` command. Use `SetKey` also to display the current settings.

◇ The default keyboard remapping is defined in a resource of type `DKEY`.

You may copy the displayed page to the clipboard using the menu command `Copy` or the appropriate function key on the enhanced keyboard. Note that the page is copied in the current magnification.

6.4 Printing a Document

6.4.1 The Page Range Concept

To give you a great flexibility in printing your document, it may be split up into *page ranges*. A single page range simply is a part of your document with some printing characteristics.

First, however, a short remark about the page numbers: There are two different kinds of page numbers, the absolute page numbers and the TEX page numbers. Because the `dvi` file contains a sequence of complete TEX pages, every page has a defined position within the `dvi` file and therefore a defined absolute page number. If the `dvi` file contains n pages, these numbers will be in the range $1 \ldots n$. However, each page contains a TEX page number. Most commonly, this is the number that will appear on the top or bottom of your page. These are the numbers TEX displays on the screen within square brackets while translating your document. In general, each page in the `dvi` file may have any TEX page number. Therefore all page references **MacDVI** will accept must be absolute page numbers and not TEX page numbers. However, in certain instances **MacDVI** will display something like `3/[5]` as a page number. The number before the slash is the absolute page number, the number in brackets is the TEX page number. For those of you who are interested: The TEX page numbers are the values of the registers `\count0` to `\count9`. The usual macro packages will set `\count1` to `\count9` to zero and `\count0` to the current page number. TEX displays these ten count registers in square brackets on your terminal when processing a document.

Warning: **MacDVI** has an option to sort the pages by TEX page numbers. Note that the absolute page numbers will change, i.e. the page with the smallest TEX page number will now have the absolute page number 1, regardless of the position it has in the `dvi` file.
Now back to page ranges. They are described by the following parameters:

Parameter	Meaning
`firstPage <n>`	Start printing with absolute page *n*
`lastPage <n>`	Stop printing with absolute page *n*
`offPage <n>`	Print only every *n*th page
`reversePages`	Reverse the output of the pages
`pauseAfterPrint`	Pause printing before printing the next page range
`ejectAddPage`	Eject an additional empty page at the end

Page ranges are displayed by **MacDVI** in the following format:

```
<first>..<last> by <offset>; <options>
```

Options contain a list of characters for the additional page range parameters:

Character	Parameter
r	`reversePages`
p	`pauseAfterPrint`
e	`ejectAddPage`

6.4.2 The Print Job Dialog

If you print a single document (not two or more documents opened either from the MPW shell or the Finder) the standard print job dialog will be modified to give you more control over printing (see figure 8). These additional controls allow you to specify any number of page ranges to print.

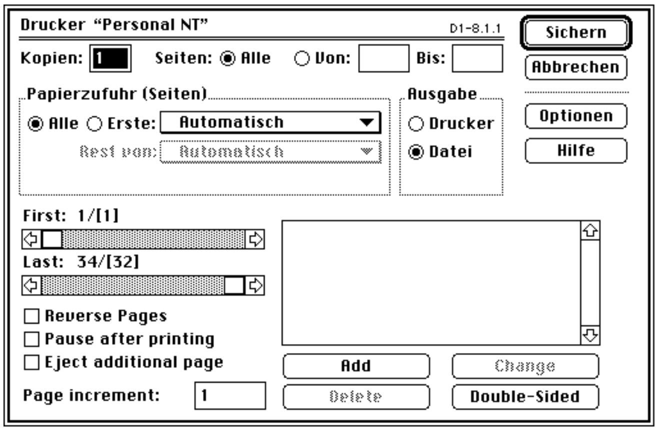


Figure 8: Print Job Dialog

The usage of these additional items is simple: On the left side you will find some controls to specify a single page range, and on the right side you will find a list of page ranges that will be printed and some controls to add, delete or change some page ranges from the list. You may double-click on an item in the list to copy its settings to the controls on the left side.
If you didn't specify any page range (the list is empty) **MacDVI** will use the normal settings of the print job dialog that you specify with the controls at the top of the dialog.

One note about the `Eject additional page` item: If checked **MacDVI** will eject an additional page at the end for normal printouts and at the beginning for reverted printouts.
One note about double-sided printouts: **MacDVI** simply will print the first, third, ... page of the page on the front side and the second, fourth, ... page on the back side of the paper, regardless which TEX page numbers they have. You should select the proper first and last page with the scroll bars on the left side and then click on `Double-Sided` to add two different page ranges to the list. **MacDVI** will use some settings from the file `$dt_DeviceDefs` to decide how double-sided printouts are done on your particular printer (which side to print first: front or back? print front sides normal or reverted? print back sides normal or reverted, ...). See chapter 6.2.2 for more details. **MacDVI** will also automatically eject an additional page if needed. If the settings in the `$dt_DeviceDefs` file are correct the only thing you have to do is to reinsert the printed pages back into your printer and tell **MacDVI** to continue printing.

- ⋄ Note that the file `$dt_DeviceDefs` might not contain correct settings for your printer because I simply have a limited amount of testing capabilities. Please send me a corrected `device_def` of your printer if you found the correct settings !!!

7 Including Graphics into dvi Files

7.1 A Sample Graphic

Including graphics into TEX documents is a breeze with DirectTEX — if the graphics file is correctly formed. But first we will give an example of an included graphic — a complex graphic created with Mathematica™.

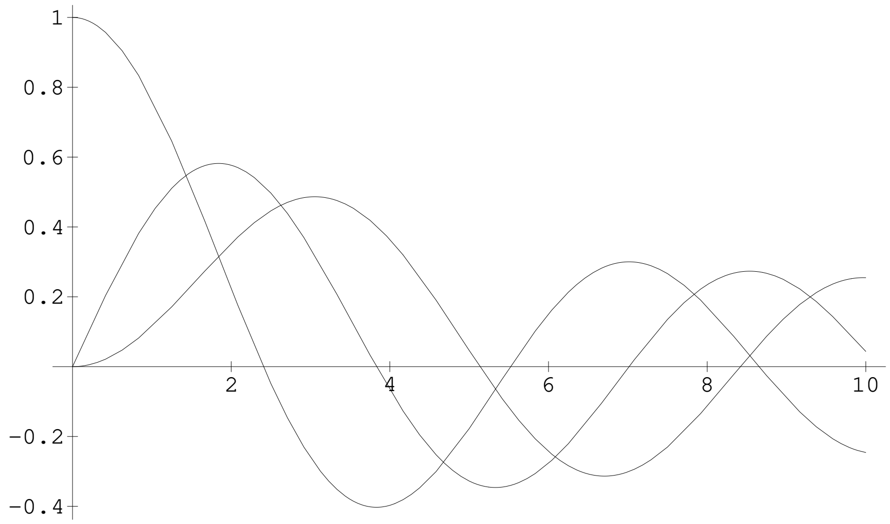


Figure 9: A sample Mathematica™ graphic

This picture has been included by using the following commands in the `tex` file:

```
\begin{figure}[htb]
\def\epsfsize#1#2{0.8\textwidth}
\def\MM{{\rm Mathematica$^{\hbox{\tiny TM}}$}}
$$\epsffile{Mathematica.epsf}$$
\caption{A sample \protect\MM\ graphic}
\end{figure}
```

You may take a look at the file `:Include-Files:DirectTeX:Mathematica.epsf` to see how a good graphics file is made. Be sure to look at the resource and the data fork of this file. The data fork contains some Postscript code, the resource fork contains a `PICT` resource which in turn contains the QuickDraw representation of the graphic.
MacDVI is also able to include QuickDraw-only graphics. However, it is highly recommended to use graphic files that contain the PostScript and QuickDraw representation of the graphic.
The following sections of this chapter are taken from the manual for `dvips`, a PostScript driver for `dvi` files by Thomas Rokicki, since PostScript graphics inclusion into `dvi` files works similarly in **MacDVI** and Rokicki's `dvi` file driver `dvips`. Note that `dvips` is also included in the DirectTEX package.

7.2 The Bounding Box Comment

Every well-formed PostScript file has a comment describing where on the page the graphic is located, and how big that graphic is. This information is given in terms of the lower left and upper right corners of a box just enclosing the graphic, and is thus referred to as a bounding

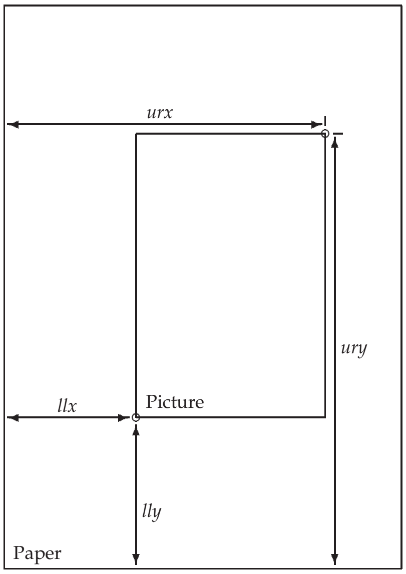
box. These coordinates are given in PostScript units (there are precisely 72 PostScript units to the inch) with respect to the lower left corner of a sheet of paper.
To find out wether a PostScript file has a bounding box comment, just look at the first few lines of the file. PostScript is standard ASCII, so you can use any text editor to do this. If within the first few dozen lines there is a line of the form

```
%%BoundingBox: 0 0 396 396
```

(with any numbers), chances are very good that the file is Encapsulated PostScript and will work easily with **MacDVI**. If the file instead contains a line such as

```
%%BoundingBox: (atend)
```

the file is still probably Encapsulated PostScript, but the bounding box (which is needed by the `epsf.sty` and **MacDVI** in order to position the graphic) is at the end of the file and should be moved to the position of the line above. This can be done with your text editor.
If the document lacks a bounding box altogether, one can easily be added. Simply print the file. Now, take a ruler, and make the following measurements. All measurements should be in PostScript units, so measure it in inches and multiply by 72. The distance from the left edge of the paper to the leftmost mark on the paper is *llx*, the first number. The distance from the bottom edge of the paper to the bottommost mark on the paper is *lly*, the second number. The distance from the left edge of the paper to the rightmost mark on the paper is *urx*, the third number. The fourth and final number, *ury*, is the distance from the bottom of the page to the uppermost mark on the paper. Refer to the figure at the right. Now, add a comment of the form `%%BoundingBox:` *llx lly urx ury* as the second line of the document, the first line should be a line starting with the two characters '`%!`'; if it doesn't, the file probably isn't PostScript. Or, if you don't want to modify the file, you can simply write these numbers down in a convenient place and use them when you import the graphic. If the document does not have such a bounding box, or if the bounding box is given at the end of the document, please complain to the authors of the software package that generated the file; without such a line, including PostScript graphics can be tedious.



7.3 Using the `epsf` Macros

Now you are ready to include the graphic into a TEX file. Simply add to the top of your TEX file a line like `\input epsf` (or, if your document is in LATEX or SliTEX, add the `epsf` style option, as was done to the following line).

```
\documentstyle[12pt,epsf]{article}
```

This needs to be done only once, no matter how many figures you plan to include. Now, at the place where you want to include the file, enter a line such as

```
\epsffile{foo.ps}
```

If you are using LATEX or SliTEX, you may need to add a `\leavevmode` command immediately before the `\epsffile` command to get certain environments to work correctly. If your file did not (or does not currently) have a bounding box comment, you should supply those numbers you wrote down as in the following example:

```
\epsffile[100 100 500 500]{foo.ps}
```

(in the same order they would have been in a normal bounding box comment). Now, save your changes and run TeX and **MacDVI**; the output should have your graphic positioned at precisely the point you indicated, with the proper amount of space reserved.

Note that you *must* supply the bounding box numbers to the `epsffile` macro in case the graphics file does not contain any PostScript code in the data fork. This prevents the `epsffile` macro from opening the data fork and trying to find the `%%BoundingBox` comment. If you omit this and the data fork of the file contains the `PICT` then you will get some unpredictable results.

The effect of the `epsffile` macro is to typeset the figure as a TeX `\vbox` at the point of the page where the command is executed. By default, the graphic will have its 'natural' width (the width of its bounding box). The TeX box will have depth zero and a 'natural' height. The graphic will be scaled by any `dvi` magnification in effect at the time.

Any PostScript graphics included by any method in this document (except everything that is included using the `bop-hook`) are scaled by the current `dvi` magnification. For graphics included with `\epsffile` where the size is given in TeX dimensions, this scaling will produce the correct, or expected, results. For compatibility with old PostScript drivers, it is possible to turn this scaling off with the following TeX command:

```
\special{!/magscale false def}
```

Use of this command is not recommended because it will make the `\epsffile` graphics the wrong size if global magnification is used in a `dvi` document, and it will cause any PostScript graphics to appear improperly scaled and out of position if a `dvi` to `dvi` program is used to scale or otherwise modify the document.

You can enlarge or reduce the figure by putting

```
\epsfxsize=<dimen>
```

right before the call to `\epsffile`. Then the width of the TeX box will be `<dimen>` and its height will be scaled proportionately. Alternatively you can force the vertical size to a particular size with

```
\epsfysize=<dimen>
```

in which case the height will be set and the width will be scaled proportionally. If you set both, the `\epsfysize` will be ignored; there is currently no way to change the aspect ratio of the image.

A more general facility for sizing is available by defining the `\epsfsize` macro. You can redefine this macro to do almost anything. This TeX macro is passed two parameters by `\epsffile`. The first parameter is the natural horizontal size of the PostScript graphic, and the second parameter is the natural vertical size. This macro is responsible for returning the desired horizontal size of the graph (the same as assigning `\epsfxsize` above). In the definitions given below, only the body is given; it should be inserted in

`\def\epsfsize#1#2{`*body*`}`

Some common definitions are:

- `\epsfxsize`: This definition (the default) enables the default features listed above, by setting `\epsfxsize` to the same value it had before the macro was called.

- `0pt`: This definition forces natural sizes for all graphics by setting the width to zero, which turns on horizontal scaling.

- `#1`: This forces natural sizes too, only by just returning the first parameter (which is the natural width) and setting the width to that.
- `\hsize`: This forces all graphics to be scaled so they are as wide as the current horizontal size (in LaTeX, use `\textwidth` instead of `\hsize`).
- `0.5#1`: This scales all figures to half of their natural size.
- `\ifnum#1>\hsize\hsize\else#1\fi`: This keeps graphics at their natural size, unless the width would be wider than the current `\hsize`, in which case the graphic is scaled down to `\hsize`.

If you want TeX to report the size of the figure as a message on your terminal when it processes each figure, give the command `\epsfverbosetrue`.

7.4 Using QuickDraw-only Graphics and `ClipToRez`

You may use QuickDraw-only graphics. However, you should be aware of the following: If you are using the `epsffile` macro you must supply the bounding box values as the optional parameters to the macro. Otherwise, the `epsffile` tries to open the data fork of your graphics file and to find the `%%BoundingBox` comment (which is missing because you are using a QuickDraw-only graphic), which will simply fail.

MacDVI supports different types of QuickDraw graphics: The file may be a normal `PICT` file generated e.g. by MacDraw™. Such files contain the QuickDraw commands in the data fork and nothing in the resource fork. The other type of file **MacDVI** supports contains the QuickDraw commands in a resource of type `PICT` in the resource fork, the contents of the data fork may be undefined (or PostScript code).

Extracting the bounding box values from a `PICT` file may be very tedious, so there is an easier way: The `ClipToRez` tool allows you to save the clipboard to a file, which contains the `PICT` from the clipboard in a resource and in the data fork the appropriate `%%BoundingBox` comment.

You should use the tool `ClipToRez` if you plan to include QuickDraw-only graphics. Simply copy the picture you want to include to the clipboard, and then choose the menu item `Convert Clipboard...` from the `TeX` menu. A file selector box will appear which lets you choose the name of the output file. Alternatively, you may execute the `ClipToRez` command from the shell.

Note that `ClipToRez` does not translate the `PICT` to PostScript code. However, it generates a PostScript header that contains the `%%BoundingBox` comment. **MacDVI** itself is able to include QuickDraw-only files into PostScript code by converting the `PICT` into a bitmap with printer resolution and converting the bitmap into PostScript code. However, `ClipToRez` may do the same job as **MacDVI**, but you should be aware that the resulting PostScript code is device-dependent because it is dependent on the resolution of the printer. You may use the command-line option `-e` to tell `ClipToRez` to generate a PostScript-bitmap of your `PICT` .

7.5 Header Files

Quite often, in order to get a particular graphic file to work, it might be necessary to send a certain header first. Sometimes this is even desirable, since the size of the header macros can dominate the size of certain PostScript graphics files. **MacDVI** provides support for this with three different mechanisms:

- Name your file `global.ps` and put it into the current directory. This method should be used if you want to include the header file only into a specific document.

- Put your file into the directory `:Include-Files:Automatic:`. Now **MacDVI** will include this file for every document.
- Put the file into a `PSHD` resource within **MacDVI**. Note that for this method the PostScript code of the header must have an `initialize` and a `terminate` macro defined because they are automatically called for every header included by this way. **MacDVI** will include such headers for every document. The standard headers that are needed for converting `dvi` files to PostScript code are included by this method.

The PostScript dictionary stack will be at the `userdict` level when header files are included. The PostScript code **MacDVI** generates checks for the existence of four hooks at the `userdict` level. These hooks are:

Hook	Time called
`start-hook`	Start of document
`bop-hook`	Start of every page
`eop-hook`	End of every page
`end-hook`	End of document

You may define some of these hooks from within a header file. E.g. you may use a header file that has the following structure:

```
userdict begin /bop-hook
{ /mysave save def gsave initgraphics
  /Helvetica-Bold findfont 144 scalefont setfont
  45 rotate 310 80 moveto 0.95 setgray
  (DRAFT) show
  grestore mysave restore
} def end
```

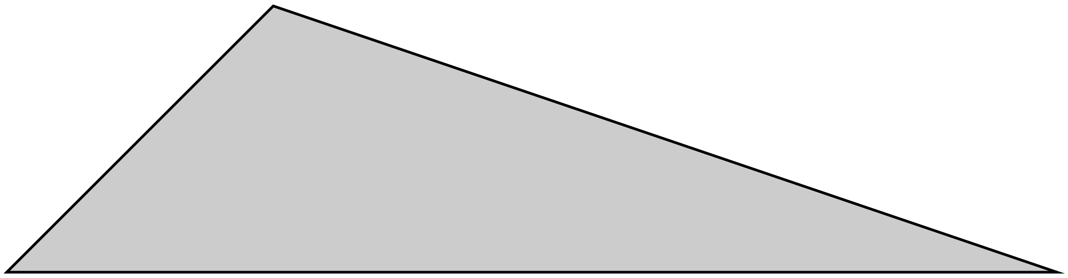
However, you should be aware that this might interfere with the surrounding PostScript code: When printing to a PostScript printer the PostScript code generated by **MacDVI** is embedded into some PostScript code generated by the printer driver, and this might cause some problems. You should save the `dvi` file to disk and then download the resulting PostScript file to your printer to circumvent such problems.

7.6 Using the PostScript `\special` Commands

There are many different `\special` commands **MacDVI** understands. They will be explained in the next chapters. However, a general hint for using any `\special` command must be given: **MacDVI** doesn't scan the whole `dvi` file before starting to print a page. Therefore, `\special` commands are interpreted only if the page they are contained in is interpreted. This may be a problem to those `\special` commands that do not produce a picture, but redefine the behavior of the PostScript code generated. e.g., if you define some `\special` commands on page 7 that define some PostScript macros, these macros will be unknown for all pages printed *before* page 7. There is a general solution to this problem: Define your PostScript macros in a separate file and include that file by any of the methods **MacDVI** supplies.

7.6.1 Literal PostScript

For simple graphics, or just for experimentation, literal PostScript graphics can be included. Simply use a `\special` command that starts with a double quote `"`. For instance, the following (simple) graphic:



was created by typing:

```
\vbox to 100bp{\vss % a bp is the same as a PostScript unit
\special{" newpath 0 0 moveto 100 100 lineto 394 0 lineto
closepath gsave 0.8 setgray fill grestore stroke}}
```

Note that you are responsible for leaving space for such literal graphics.

7.6.2 Literal Headers

Similarly, you can define your own macros for use in such literal graphics through the use of literal macros. Literal macros are defined just like literal graphics, only you begin the special with an exclamation mark instead of a double quote. These literal macros are included as part of the header material in a special dictionary called `SDict`. This dictionary is the first one on the PostScript dictionary stack when any PostScript graphic is included, whether by literal inclusion or through the `epsffile` macros. The previously mentioned `\special` command

```
\special{!/magscale false def}
```

is an example of such a literal macro. However, the note from above applies to this case.

7.6.3 Other Graphics Support

There are other ways to include graphics with **MacDVI**. One is to use an existing package, such as `psfig`, which **MacDVI** supports.
Other facilities are available for historical reasons, but their use is discouraged, in hope that some 'sane' form of PostScript inclusion shall become standard. Note that the main advantage of the `epsffile` macros is that they can be adapted to whatever form of `\special` eventually becomes standard, and thus only minor modifications to that one file need to be made, rather than revising an entire library of TEX documents.
Most of these specials use a flexible key and value scheme:

```
\special{psfile=filename.ps [ key=value]*}
```

This will download the PostScript file called `filename.ps` such that the current point will be the origin of the PostScript coordinate system. The optional key/value assignments allow you to specify transformations of the PostScript. The possible keys are:

Key	Meaning
hoffset	The horizontal offset (default 0)
voffset	The vertical offset (default 0)
hsize	The horizontal clipping size (default 612)
vsize	The vertical clipping size (default 792)
hscale	The horizontal scaling factor (default 100)
vscale	The vertical scaling factor (default 100)
angle	The rotation (default 0)

The dimension parameters are all given in PostScript units (`bp`). The `hscale` and `vscale` are given in non-dimensioned percentage units, and the rotation value is specified in degrees. Thus

```
\special{psfile=foo.ps hoffset=72 hscale=90 vscale=90}
```

will shift the graphics produced by file `foo.ps` right by one inch and will draw it at 0.9 times normal size. Offsets are given relative to the point of the special command, and are unaffected by scaling or rotation. Rotation is counterclockwise about the origin. The order of operations is to rotate the figure, then scale it, then offset it.
For compatibility with older PostScript drivers, it is possible to change the units that `hscale` and `vscale` are given in. This can be done by redefining `@scaleunit` in `SDict` by a TeX command such as

```
\special{!/@scaleunit 1 def}
```

The `@scaleunit` variable, which defaults to 100, is what `hscale` and `vscale` are divided by to yield an absolute scale factor.
All of the methods for including graphics we have described so far enclose the graphic in a PostScript save/restore pair, which guarantee that the figure will have no effect on the rest of the document.
Another type of special command allows literal PostScript instructions to be inserted without enclosing them in this protective shield; users of this feature are supposed to understand what they are doing (and they shouldn't change the PostScript graphics state unless they are willing to take the consequences). This command can take many forms, because it has had a tortuous history; any one of the following will work:

`special{ps:`*text*`}`
`special{ps::`*text*`}`
`special{ps::[begin]`*text*`}`
`special{ps::[end]`*text*`}`

(with longer forms taking precedence over shorter forms, when they are used). An exception is the command

```
\special{ps:plotfile filename}
```

which will copy the commands from `filename` verbatim into the output. An example of the proper use of literal specials can be found in the file `Rotate.tex` which makes it easy to typeset text turned by 90 degrees. To finish off this section, the following example is presented without explanation:

```
{\def\rotninety{\special{ps:currentpoint currentpoint %
translate 90 rotate neg exch neg exch translate}}%
\setbox0=\hbox to0pt{\huge\bf A\hss}%
\vskip16truept\centerline{\copy0\special{ps:gsave}%
\rotninety\copy0\rotninety\copy0\rotninety\box0%
\special{ps:grestore}}\vskip16truept}%
\smallskip%
```

Some caveats are in order when using the above forms. Make sure that each `gsave` on a page is matched with a `grestore` on the same page. Do not use `save` or `restore`. Use of these macros can interact with the PostScript generated by **MacDVI** if care is not taken; try to understand what the above macros are doing before writing your own. The `\rotninety` macro especially contains a useful trick that appears again and again.

8 Changes since v1.0

In this section I will give you an idea about how much work has been done since the release of DirectTEX v1.0 on 1/23/1992. This section will contain a copy of my personal `ReleaseLog` file which contains almost all changes I've made since 1/23/1992. Please note that I didn't remove the bugs in my English in the following chapter.

01/23/92 Pre-Release DirectTEX v1.0

02/05/92 Release DirectTEX v1.0. Added Prerelease of `Pict2Rez`. Added New `LaTeX` / `SliTeX`. Slight modifications of `DVIReader`

02/15/92 Updated the following tools (`web` files copied from `hadron.tp2`):

`GFType` 3.0/1.0 → 3.1/1.1
`PLToTF` 3.3/1.0 → 3.4/1.1
`Tangle` 4.1/1.0 → 4.2/1.1
`TeX` 3.0/1.0 → 3.14/1.1
`VPToVF` 1.2/1.0 → 1.3/1.1
`Weave` 4.1/1.0 → 4.2/1.1
Some `web` source files changed (only typos corrected, no code change, they are not listed here).

02/16/92 Implemented the following new tools:

`PatGen` v1.0/1.0

02/24/92 Renamed `Pict2Rez` into `ClipToRez` and released it.

02/25/92 Removed the shell variable `TeXMemSize`, now the name of the format file is used to determine the version of TEX needed. Renamed shell variable `TeXFormatFiles` into `FormatFiles` and removed the shell variable `MFBaseFiles`. Now TEX and METAFONT search for their format (base) files within `{FormatFiles}`. The TEX menu layout has changed: The format files now are at the end of the menu. To build up the list of available format files the folder `{FormatFiles}` is scanned now.

02/25/92 Revised the tool `GFToPK` and released the following new tool:

`PKToGF` v1.1/1.0

02/27/92 Removed unneeded `xchr` / `xord` arrays from the following tools: `DVIType`, `GFToDVI`, `GFToPK`, `GFType`, `MFT`, `PKToGF`. Slight changes to `PoolType`.

03/01/92 Implemented the following new tools:

`PKType` v2.3/1.0

03/02/92 Checked and revised the following tools, introducing the new `search_file` procedure:

`DVIType`, `GFToDVI`, `GFToPK`, `GFType`, `MFT`, `PatGen`, `PKToGF`, `PKType`, `PLToTF`, `PoolType`, `TFToPL`.

03/19/92 Slight modifications to `DVIReader`. Fixed problem with keyboard layout change caused by the command-option-space key combination. Called `DVIReader` now v1.1. Changed the version number of the whole DirectTEX package to v1.1. Adjusted the `AboutTeX` dialog. This release is the first release that is to be found on the ftp server in Stuttgart.

03/31/92 Minor change to `DVIReader`: Introduced the `blueSkyEncoding` for Mac Screen Fonts to enable the `DVIReader` using Blue Sky PS Fonts. Didn't change the version number. Didn't like the results.

04/04/92 Implemented `TeX-XeT` and a pre-release of `ivd2dvi`. The concept isn't clear at all.

04/06/92 Revised the TEX Scripts, fixed some bugs. TEX and METAFONT changed to new versions (TEX 1.2 / METAFONT 1.1) because full batch mode support has been included (look for `{TeXBatchMode}` and don't display file dialogs and don't exit with -9 if `{TeXBatchMode} = 1`.

04/16/92 Fixed some bugs in `MFWare` / `TeXWare` tools and `WebLib`. Changed most Mac version numbers to 1.2. Changed the `TeX-XeT` implementation slightly.

04/17/92 Implemented `PatGen` v2.0/1.1. That was the first time I wrote a change file completely from scratch. Because it was so easy I started the implementation of the `mweb` package (oh Yannis, this will NOT do the things you would like to see...). Decided not to implement `ivd2dvi` within the `DVIReader`, but to write a new `IVDtoDVI web` program (derived from `DVIType`). This should be a standard on all systems using `TeX-XeT`.

05/01/92 Implemented several new versions (TEX 3.141, METAFONT 2.71, `PatGen` 2.1, `Weave` 4.4). Major change: Adapted the `TeX--XeT` change file by Peter Breitenlohner. This version of `TeX--XeT` produces `dvi` files directly, so a `IVDToDVI` is never used (whow !!!). Bye bye `IVDToDVI`, and welcome to the pleasuredome. Implemented a new `InitFormats` script that uses some script files to initialize new format / base files.

05/07/92 Fixed code in Libraries and `DVIReader` that required ColorQD and didn't run on Macintosh without ColorQD (`GetMCInfo`, `SetMCInfo`, ...). Simply removed it (lets look if it works...)

05/07/92 Added a new item `Check for fonts` to the TEX Menu. This will check the current projects `dvi` file for all fonts used.

06/12/92 Started to release v1.2 of DirectTEX because people are asking for a bug fix for non-ColorQD. Decided to include bigger changes in v2.0. Removed the HD-disk-version of DirectTEX. Now only a DD-disk version will be released with 7 DD-disks.

06/26/92 Bug fix in `DVIReader` which caused an address error on 68000 machines. Must be tested !!!

07/06/92 All bugs related to MC68000 processor problems in `DVIReader` disappeared. Now it runs smoothly (but slow) on all 68000 machines...

07/26/92 Another bug in `DVIReader`: Handling long `\special` commands was not possible (overflow in `outputBuffer...`). Fixed.

09/02/92 Released a beta version of DirectTEX v1.2. Didn't include the latest version of the DC fonts and full DC font support, but that seems to be not so bad.

09/22/92 Fixed some bugs in PostScript output of the `DVIReader`. Now printing of `EPSF` seems to be ok.

09/26/92 Implemented new versions of `CTangle` and `CWeave` from scratch (that means, I wrote the whole change file by my own). They work... Included `MTangle` and `MWeave` (not complete at all) again (after implementing them a while before).

10/01/92 Fixed a bug in the `DVIReader` that made the PostScript code unusable on some printers. Yannis reported this bug to me. I will never use the phrase "All bugs in `DVIReader` disappeared" again.

10/05/92 The bug IS fixed. Fixed another bug that made the previewer incompatible with `BoxedEPS` by Laurent Siebenmann. Now both `epsf.tex` and `boxedeps.tex` are supported by the `DVIReader`. I must send a mail to Laurent that he should add DirectTEX support to both `BoxedEPS` and `BoxedArt`.

10/08/92 Changed some script files. Renamed `TeXProject` / `TeXDirectory` to `TeXProjectName`, `TeXProjectDir` and `TeXProjectExt`. This allows to run files like `array.doc` through the project mechanism.

10/08/92 More changes to `DVIReader` to make it compatible with `BoxedEPS`. Now the PostScript output for `PICT` files should be fixed. But now the bad news: The bug reported by Yannis isn't fixed at all. Waiting for some code demonstrating this.

10/13/92 More changes to `DVIReader`. Now supports `OzTeX \special` and `DVIView \special` (although `DVIView` seems to be rather exotic...) In the future we will add `PNTG` (MacPaint) file support (maybe...) Changed the way the `DVIReader` handles `PICT` files within PostScript code (lowered resolution to 72dpi). This works much faster for all files, and the preferred `PICT` files will be only bitmaps, this means screwing up the resolution doesn't make sense. Yannis reported another bug to me. I don't know what the problem is...

10/14/92 Changed the way DirectTEX remembers its preferences. We now store them in a file `DirectTeX Preferences` in the system folder (or, for system 7, in the preferences folder). Changed METAFONT output window: We now allocate two offscreen bitmaps (if possible) so the update of the window is always clean (and not disturbed by drawing in the buffer). Decided not to add `PNTG` support to the `DVIReader` because this format seems to be rather old.

10/14/92 To do: Fix section numbers in `MWeave.ch`. Fixed a bug in METAFONT that didn't update the window for a second `showit;` call. Now METAFONT seems to work nicely. Many thanks to Scott Petrack for his useful information. He is the only person known to me who tried to do all the exercises in the METAFONT book chapter 5. I didn't do that for a long time, but now I decided to do that (not for me, but as a test of DirectTEX).

11/15/92 General revision of all make files, the TEX library, the linking process, the compilation options, etc etc. Called this version now v2.0 because v1.2beta never explicitly showed up the beta. Don't release v2.0 before it is tested !!! Included compressed file support into TEX and METAFONT. Wrote a new utility `Compress` that enables the user to compress / decompress the TEX and METAFONT input files. Added menu item `Recompress files...` that processes list output from `Compress`.

11/17/92 Fixed bug in Gnu C 1.37.1.r7 that caused some tools to compile wrong. Told the author of Gnu C about this bug. He told me that he is working on v2.0 of Gnu C. Maybe DirectTEX will get lightning fast when compiled with this version. Released the `Compress` tool. Working on the `TextTrans` tool.

11/20/92 `TextTrans` is ready. Must go through beta test stage. Things to do: Check all resources if they can be made purgeable, and check all places where resources are loaded if they should be released later on. Some resources should be locked down before use !!!

12/08/92 Major revision of all tools, now everything is going straight forward to v2.0. Tested `MathTime` fonts with DirectTEX, no problems found. Must tell Michael Spivak how to support DirectTEX. Must convert the `$\approx$.ps` files that contain the fonts into Adobe type 1 fonts and TrueType fonts by using the tools `Post2Hex`,... from Yannis.

12/09/92 Enhanced the speed of `DVIReader` a second time (don't remember when I did it the first time). Now it should be a factor 8 faster than v1.2 (wow...)

12/11/92 Revised all script files. Now they should be OK. Introduced a new mechanism for including header files into the `DVIReader`: Just search a folder called `Automatic` along the `{PicFiles}` search path and include all files found inside that folder as header files (in the order they appear there). TEX and METAFONT now do an exit(history) at the end, but translate `fatal_error_stop = 3` into `fatal_error_stop = -9`. `calledit` will leave with `exit(4)`, all script files calling TEX or METAFONT can now decide when to execute the output of TEX and METAFONT on `dev:stdout`. Revised `WebLib.c` and `WebLib.h`, introduced some new register variables, deleted unneeded parts, shortened filenames to 256 bytes max., included a general check that strings do not exceed 256 characters (including the length byte for p-strings at the beginning and the `'\0'` for c-strings at the end). Maybe sometimes I will introduce a general mixing of p- and c-strings by always creating strings with a length byte and a trailing `'\0'`.

12/14/92 Merged `IniTeX` / `VirTeX` and `IniMF` / `VirMF` into a single program. This required some changes to `Web2C` (handle `sendtocpp()` accordingly). Now `IniTeX` is called by `TeX -i`, and `VirTeX` is called by `TeX` (same for METAFONT with `Ini` / `VirMF` → `MF`). Additionally, there is another command line option `-s` which turns statistics on (previously, this had to be done at compile time...). Changed the way TEX and METAFONT handle their pool file: Instead of reading an ASCII text file the string pool is read from a resource included into the tools.

12/14/92 The `TextTrans` and `Compress` tool seem to work very stable. The only thing that I don't like is the following: Because we work on a temporary file and delete the original file if we are ready, the filename of the file may get changed (but only with respect to lowercase/uppercase letters), e.g. doing something like `TextTrans foo` will create a file `foo`, even if it originally was called `Foo` or `fOO` or `FOO` or... Is there any way to convert a partial pathname into the correct naming of the file?

12/15/92 The answer to the question above is: YES. The solution is a two-way solution: Under System 7, we use `PBExchangeFiles()` to exchange the file and the temporary file, and this approach conserves the case of the input file name. If we are not running under System 7, we use a routine that converts a given filename into the correct name by the following approach: Convert the folder the file resides in into a `dirID`, and convert that `dirID` back into a path name. This path name now spells correctly (b.t.w., this will change something like ::...::... into something without a :: !!!). In the directory the file is in we are simply searching it by `PBGetCatInfo(1..numOfFiles)`. That's all (but it is slow if the directory contains a huge amount of files...)

12/15/92 Killed the `PoolType` tool. Why should we have a `PoolType` if there are no more `pool` files around? And for verifying a pool file we don't need it because `Tangle` will output correct `pool` files.

12/16/92 Removed the annoying dialog from `DVIType`, replacing it by command line options. This is much better !!!

12/16/92 Revised the DirectTEX manual, included this file. Things to do: Revise the `DVIReader` manual, and recompile the whole package (for all of you who want to know: This takes more than 2 hours on my Quadra 700...)

12/18/92 Decided to include a serial-number mechanism into DirectTEX because I think that there are a lot of people around who use DirectTEX but do not pay the ShareWare fee. Now DirectTEX may be used without serial number, but every time you start a tool you will be asked to personalize the copy of DirectTEX. You may press `OK` even if the input in the dialog is invalid, but this dialog will appear three times before it gives up and allows you to use the tool you just started. I hope this is boring enough to make people think about paying the ShareWare fee to get a personal registration code, in which case you can enter it and will never see the dialog again (unless you delete the preferences file in which all tools will look for a registration code). Oh, let me tell you that the serial number is simply computed out of the name and organization inputs… Exercise 1: Try to figure out the algorithm that computes a valid serial number for given user and organization strings.

12/20/92 Slight bug fixes in the `WebLib`. The Gnu C compiler has still some bugs inside. Must tell the author of Gnu C about problems with floating-point math in the optimizing 68000-compiler part of Gnu C. I hope Gnu C v2.0 won't have these bugs. At this time I compile the whole package using the `-mc68020` and `-mc68881` options.

12/30/92 Changed the way TEX, METAFONT and BibTEX ask for files. Now the message they put on the screen is displayed in the standard file dialog box. Now the user can see which file is missing (before this change, the message was obscured by the dialog box !). Wrote a new script named `ShowMissing` which displays the names, sizes,… of the fonts currently recorded in the `{dt_MissingFonts}` file.

01/16/93 Introduced a new memory management to TEX and METAFONT: Now `mem_max` is not longer specified in the `CMEM` resource, but is computed at startup by the following scheme: If `IniTeX` or `IniMF` is running, `mem_max` is set to `mem_top` because this is required by TEX and METAFONT. If `VirTeX` or `VirMF` is running: `mem_max` is set to `(max_block-mem_save)/sizeof(memory_word)`. Now this value is forced to be in the range `mem_top`…`max_halfword`-1. `max_block` is the size of the largest available memory block in bytes. `mem_save` is a new variable introduced into the `CMEM` resource. The size should be something like `max_inopen*bytes_needed`, with `bytes_needed` being the space in bytes that the decompression algorithm needs (see next paragraph).

Tested the compression routine with respect to efficiency and memory requirements. Got the following results (by compressing the `TeX-Inputs` folder):

`n_bits`	`bytes_needed` (Bytes)	Disk Space (Bytes)	Compression (%)
9	9456	1522415	15.42
10	10992	1190786	33.84
11	14064	1008908	43.95
12	20208	893344	50.37
13	32496	845717	53.01
14	57072	824028	54.22
15	106224	814537	54.74
16	204528	810467	54.97

I changed also the way the `Compress` tool works: The value for `n_bits` that should be used for compression is now stored in a `CMEM` resource. By this way the user has full flexibility when adjusting the memory settings of TEX and METAFONT. Note that there is a relationship between `n_bits` in the `Compress` tool and `max_inopen` and `mem_save` in TEX and METAFONT.

04/01/93 I haven't written my comments to this file for a very long time, so now I will update it as much as possible. The following things happened up to this day:

- I rewrote the `DVIReader` completely, calling it `MacDVI` now and made it a stand-alone version. For people with enough memory this has great advantages: They are able to look at the `dvi` file and to look at the source code at the same time. However, for people with little memory this is really bad. Therefore I decided to rewrite `MacDVI` someday again and make it compilable either as a stand-alone or `MPW` version. However, at this moment, `MacDVI` isn't ready at all. I have to fix some real bad memory management problems.
- I started implementing some stand-alone versions. Now I can compile every tool as a stand-alone version which accepts input from files opened by the Finder and from the console. However, at the moment I'm using the `SIOW` library from Apple, and this library has indeed very poor abilities (and a few bugs).
- The great news: I wrote `MFToPK`, which is something like a METAFONT with a post-processor (here: `GFToPK`) included. `MFToPK` will do the following: It will produce a font, convert it into a `pk` file, and save this `pk` file where the environment variables `PKFiles` and `PKFormat` want it. It will create every folder needed for this task if it is not there. Therefore `MFToPK` will shorten the script file `MakeMissing` a lot.
- This `MFToPK` is the main target for my stand-alone libraries: The stand-alone version will be available for members of the french Gutenberg group.

05/09/93 Some days ago I "released" a developer release of DirectTEX v2.0d. Now I have a lot of bug reports:

- `MFToPK` and `MacDVI` do not read properly the `PKFormat` environment variable. They are using a default instead. Fixed.
- `CWeave` has a real bad bug inside: It doesn't produce any TEX-Code, except for the index and the title-page. Fixed.
- A lot of other things...

07/15/93 Created a `cmdo` resource for every tool, now they all have a commando interface and are pretty MPW-like. Changed `Web2C` to automatically create the needed resources for changeable constants (the `CMEM` resource). Now every tool that has some constants that define some memory settings has a `CMEM` resource and therefore has adjustable memory settings. Pretty nice.

Did a lot of small changes the last two month, and did a lot of fighting against buggy C-compilers. Gnu C 1.37.1.r15 still has some bugs, I reported them again to the author — nothing happens. But the bugs occur in quite simple situations, so I wonder that nobody did see them before. Still waiting for Gnu C 2.3.3. At the moment I'm using MPW C again, it does a quite good job.

09/16/93 Created the β-Release of DirectTEX v2.0. Maybe I do some changes in the next few days, and then I put everything out on the server. People are waiting for it. There is still one big piece of work left: The revision of the manual. Unfortunately I simply do not like that. That's all. One thing changed: I included `TeX--XeT` v1.1 into TeX v3.141. In the next days I hope I can switch to `TeX--XeT` 1.1 together with TeX 3.1415. This should be included into DirectTEX v2.0.

09/27/93 Released DirectTEX v2.0b01. Released `MFToPK` v2.0b01. Put everything on the server. Creating all the installation images and files took me exactly two days !!!

09/29/93 Bug report by Yannis: Although it is possible to use relative pathnames for directories, `MFToPK` seems to have some problems with searching files within relative directories. Verified that: Searching something like `:MF-Inputs:$\approx$1` doesn't produce correct results. Searched, found and killed this bug. Maybe meanwhile two other bugs were born. That's life...

OK, in the next days I will release v2.0b02. But I will wait a little.

10/27/93 Started editing the manual for v2.0. Decided to include some more chapters explaining the usage of DirectTEX with a simple example. v2.0b02 is ready, I will put it on our server as soon as the server is working again. At the moment `hadron` is down, so it doesn't make sense to put something on `hadron`. Maybe I have enough time to finish the manual.

11/16/93 v2.0b02 isn't out at all. I did a lot of work the last two weeks, and now everything should go straight to v2.0b02. The manual is getting bigger and bigger… Fixed a bug in `DVIType`: `DVIType` did not close the input files, so if a `dvi` file contains a lot of fonts the maximum number of open files will be reached, and opening the next one will fail. Somebody should tell D.E. Knuth that most operating systems have some limits on the number of open files, so it is a good idea to include some code that closes files if they are not used anymore. Checked all other tools for such situations, didn't find any other problem.

11/18/93 Installed `NFSS2`. It seems to work. Removed the obsolete `NFSS` package. Now I only have to add some support to **MacDVI** for the Cork encoding. Updated all TEX input files.

12/07/93 Fixed a bug in **MacDVI**, changed the libraries slightly. Waiting for MPW 3.3... Changed all environment variables to start with `dt_` so that they may be easily recognized. Changed `UserStartup•TeX` into `UserStartup•DirectTeX`. Yannis supposed to do the latter two changes.

12/15/93 Got MPW 3.3. Now fixed some bugs that are introduced by using MPW 3.3.

12/27/93 Fixed bugs in all tools that have their own menu bar. This killed the MPW help menu (this bug showed up under MPW 3.3).

01/12/94 Updated `CTangle` and `CWeave` to v3.1.

01/18/94 Revised the manual. Everything is going straight to v2.0b03.

01/20/94 Made **MacDVI** `RShell` compatible. Now something like `RShell -r MacDVI 'MacDVI -check foo'` works. The output of **MacDVI** is sent back to the MPW Shell. Very nice.

02/15/94 I wrote a MPW replacement and included it into **MacDVI**, calling the result DirectTEX. At this moment it is still unfinished, but in some time we don't need MPW anymore. However, the MPW version of DirectTEX will still be available and fully supported (because I like it very much, and I like MPW ;-)...).

I rewrote all my libraries to fully support system 7, but now the "bad´´ news: They now only support system 7, you may not use system 6.x, ... Bit I think this isn't a great disadvantage. For all you want to know: Writing code for systems 6.x and system 7 at the same time is very bad, you always have to check that some toolbox routines are supported, and you always have to use two different concepts. Code size grows, and readability tends towards zero because there are a lot of things like
`if (TrapAvailable (...)) { ... } else { ... }` or
`Gestalt ( .. ); if ( ... ) { ... } else { ... }`.

So I will put DirectTEX v2.0b05 onto the server in the next days, it will be a system 7 only, mc68020 or higher only version of DirectTEX.

02/17/94 Released DirectTEX v2.0b05, send a letter either per e-mail or ordinary mail to all registered users, telling them their personal registration code. Put the complete installation on the ftp server `hadron`.

02/18/94 Bug report by Yannis: `CTangle` is able to open different named output files, but my version didn't open anything — I simply got the wrong file name for the `fwriteopen ()` call. Fixed.

02/19/94 Bug fix in **MacDVI**: Calling `PicComment` unlocks the handle, and I didn't notice that. This caused **MacDVI** to crash under certain low memory conditions when printing to a PostScript printer.

03/23/94 In the last four weeks, I went from v2.0b05 to v2.0b10, and I fixed a lot of minor bugs. Furthermore, I introduced a lot of new features into **MacDVI**, and I work hard on DirectTEX Pro, the Standalone-version of DirectTEX. I will send out the first beta release within the next weeks to a few beta testers.

04/27/94 Another four weeks passed, in principle nothing happened. v2.0b12 is ready, and I'm thinking about calling it v2.0 and releasing it... I changed ALL file types because I wanted to register them with Apple and there are some conflicts... The new Standalone-DirectTEX is in principle ready, it's creator type is `'TeX+'`, therefore all file types will end with a `'+'` and start with a capital letter. This conforms to Apple's guidelines about file/creator types.

Rewrote the mechanism to initialize format files, it is now compatible with the new Standalone-DirectTEX (which I will call DirectTEX-Pro...). Thinking about including LaTeX2e within DirectTEX v2.0.

07/30/94 In the last months, I worked mainly on DirectTEX *Pro* the new Stand-Alone shell that should replace MPW in the future. I have done a lot of things, but now it looks like getting ready in a finite amount of time.

??? I can't remember what I did the rest of the year...

1/11/95 DirectTEX *Pro* v2.0b01 is available !!! I called it v2.0b01 because it is the first release of the DirectTEX *Pro* package. The former v2.0b12 of the DirectTEX package is now out of date.

1/15/95 `TeX--XeT` 3.1415 is available, **MacDVI** has been slightly revised, `dvihps` (a special version of `dvips` that supports HyperTEX links) is ready, and **MacDVI** now supports HyperTEX too. The Installation has been revised, some menu items have been renamed.

1/15/95 The LATEX 2$_\varepsilon$ version that was available at this date has been integrated into DirectTEX *Pro*.

1/25/95 Fixed some bugs in DirectTEX *Pro*. Added some features to some tools. Added the ability to typeset files without creating a project. Modified the Alpha and MPW Support.

3/26/95 Released DirectTEX *Pro* v2.0.

4/01/95 Fixed a bug in DirectTEX *Pro*: When using "Use Front Window" one couldn't change the TEX format to use. A workaround was to use `Set dt_TeXFormat <Format-Name>` directly from the shell window. This has been fixed.

4/03/95 Fixed a bug in **MacDVI**: Also **MacDVI** doesn't check the registration, it deleted the registration under circumstances. This has been fixed.

4/03/95 Made it possible to modify the `TeX` and `Formats` menu by introducing a `-r` option to `AddItem` and installing some `CUST` resources that are read by this `-r` option. However, one should be aware that the project manager doesn't check wether there are enough items in the `TeX` menu or not, so one shouldn't remove any items. This option is mainly for people that would like to modify the command key equivalents.

4/07/95 Fixed a bug that made it impossible to specify the default magnification to use for a new `dvi` file. Made some more steps towards the PowerPC, maybe I'm ready to port DirectTEX *Pro* to the PowerPC. This depends heavily on Developer Technical Support at Apple Computer...

4/10/95 Implemented a printer device for use with the DirectTEX *Pro* shell. Now one can do things like `catenate foo > dev:printer`, and the file `foo` will be downloaded to the currently selected printer. This is much better than using the little utility `Drop.PS`.

5/09/95 Added support for Archive files – they contain the data fork of some files in a resource of type `FILE`. This mechanism reduces the large amount of files that normally comes with a DirectTEX *Pro* distribution because now the standard files (TEX input files, METAFONT input files, `tfm` files, `vf` files, …) are contained within archive files.

Some notes: To identify a file within an archive, the name of the file must have the following syntax:

- It must start with a `@`.
- It must contain at least one colon `:`.

The part of the name before the last colon without the `@` is taken as a path to the library file. This path may NOT contain aliases. The part of the name after the last colon is taken as the name of the file, a resource with type `FILE` and that name is searched within the archive file.

The only supported mode for such files is reading. For creating and managing of archive files there will be a new commando.

5/09/95 The variables `dt_AuxFiles`, `dt_PICFiles` and `dt_PKFormat` have been removed. `dt_PKFormat` now is contained within `dt_PKFiles`, which allows to create and use the archives even for `pk` files. Auxiliary and picture files are now searched within the normal TEX input path.

5/09/95 The `LaTeX-2.09` folder has been removed, it's contents has been moved to an archive inside the normal `TeX-Inputs` folder.

5/27/95 Added the ability to use DirectTEX *Pro*-interna while jobs are running. This allows the user to open documents, … while e.g. TEX is running.However, this feature has to bee tested !!!

6/18/95 Updated TEX and METAFONT.

8/23/95 The support for the MPW version of DirectTEX has been removed. But the good news: DirectTEX has been ported to the PowerPC. I only have to test it on a PowerPC now.

11/07/95 The PowerPC version is ready and can now be tested. Thanks to Yannis for supplying the PowerPC Card. It helped me a lot.

2/11/96 DirectTEX *Pro* v2.1 will be ready in a few days. Fixed some bugs I encountered during testing, made some performance enhancements, and tested the PowerPC code. Seems to work, let people tell me the truth…

Installed the latest release of LaTeX2e, and changed the directory structure for it. For this reason DirectTeX *Pro* will no longer be delivered with an installer script, but simply as an archive file. Reason: Changes are much more transparent for ME.

3/10/96 Changed the path searching algorithm again to support recursive search. You may specify the depth to search using a construct like `...:*<n>`. `<n>` must be replaced by the number of directory levels to search.

3/10/96 Released DirectTeX *Pro* v2.1. Still left to do: Update of the manual.

3/16/97 v2.1.1 incorporated minor bug fixes and one major bug fix: The incompatibility between DirectTeX and System 7.6 has been removed. v2.1.2 has another few bug fixes. Left to do: The update of the LaTeX installation.

9 Appendix

References

[Knu86] Donald E. Knuth. *Computers and Typesetting*, volume A-E. Addison-Wesley Co., Reading, MA, 1984-1986. 1.1, 5.1

[Kop90] Helmut Kopka. *LaTeX – Erweiterungsmöglichkeiten*. Addison-Wesley (Deutschland) GmbH, Bonn, 2nd revised and enlarged edition, 1990. 1.1, 5.1, 5.2, 5.6, 5.9

[Kop91] Helmut Kopka. *LaTeX – Eine Einführung*. Addison-Wesley (Deutschland) GmbH, Bonn, 3rd revised and enlarged edition, 1991. 1.1, 5.1

[Sch88] Norbert Schwarz. *Einführung in TeX*. Addison-Wesley (Deutschland) GmbH, Bonn, 2nd revised edition, 1988. 1.1, 5.1