TrueTEX® "Read Me" document

(Release Notes and Other
Last-Minute Information)
TrueTEX Release 4.5U – October, 2007

The LATEX document you are reading contains last-minute information regarding the TrueTEX release you have installed. This information supercedes any statements in the TrueTEX *User's Guide*.

The TrueTEX "Read Me First" icon views this document using the TrueTEX previewer. If you are unfamiliar with the previewer and haven't yet read about it in the TrueTEX *User's Guide*, you should find the basics of using the previewer very easy: just use the scroll bars to move about the page, drag the mouse to zoom in, press `Space` to zoom out, and use `PgDn` and `PgUp` to move to the next or previous page.

You can print this document by selecting `File Print` in the previewer.

Previewing this document also serves as a basic test of whether the TrueTEX installation succeeded. For example, if any of the fonts on this page appear in blue on your color display (indicating a font substitution), then there was a problem with the installation of the TrueType fonts included with TrueTEX, and you should consult the TrueTEX *User's Guide* for information on font troubleshooting.

You can get later editions of this document over the Internet to keep up-to-date on the status of TrueTEX upgrades:

`http://www.truetex.com`

See the table at the end of this document for a summary of new features and corrections implemented in the various TrueTEX releases.

Contents

1 Quick Start

To install TrueTEX, run the `setup.exe` program on the TrueTEX Setup Disc CD. Consult the TrueTEX *User's Guide* for instructions on formatting, previewing, and printing TEX and LATEX documents.

For Windows Vista compatibility, be sure to run `setup.exe` with Administrator privileges. One way to do this is to log in as a user with Administrator privileges, right-click the `setup.exe` icon, and select "Run as Administrator".

2 PDF Output Files

(First available in 4.5H:) TrueTEX now includes command-line formatting to generate PDF files directly from TEX and LATEX sources using TrueType Computer Modern fonts. The command-line invocation is easiest via the `-x` option of the previewer. For details, see the Release History section below for releases 4.5H and 4.4. You may wish to create a convenient batch file named something like `pdflatex.bat` somewhere in the console `PATH`, containing the following:

```
@echo off
c:\TrueTeX\TrueTeX\dvigdi32.exe -1 -x "LaTeX-PDF %1"
```

Which will allow you to type:

`c> pdflatex` *full-path-of-LATEX-source-file*

to create PDF files from LATEX sources.

3 The TEX Live CD

We are pleased to include a licensed copy of the TEX Live CD as a supplement to each retail copy of TrueTEX. TEX Live is not a part of TrueTEX and is not required to run TrueTEX; you do not need to install the TEX Live disc (an enormous amount of software) if you just want the essential TEX and LATEX for Windows that TrueTEX provides.

TEX live contains macros, fonts, and documentation not included in TrueTEX, as well as ready-to-run TEX executables for Linux, UNIX, Macintosh, OS/2, DOS, and WIN32 based on the Web2c-7.3 GNU code. The *TEX Live Guide* is the essential documentation, which you may find in various languages and formats by consulting the file `index.html` in the root directory of the disc.

4 Public Domain TEXware

TEX is a widely used typesetting system and many people freely share their software efforts related to TEX. Many programs, macros, and other accessories for TEX are available on the Internet via ftp to the ctan (Comprehensive TEX Archive Network) archive, and if you are a serious TEX user you will find much valuable software there:

`ftp://ctan.tug.org/tex-archive`

5 Font Reconstitution and Metafiles

Font reconstitution is a new, mostly invisible, feature which the previewer uses to avoid font encoding problems in non-Unicode Windows (95, 98, Me). Many of the TrueType fonts in Windows contain characters that are encoded "unreliably". For example, the Windows New Times Roman font has ligatures available at codes like `0xfb01`, but many legacy printer drivers cannot render a character code in that range. Such characters are encountered with `\usepackage{times}`. Whenever a TeX virtual font thus calls on the previewer to render an unreliable character code, the previewer builds and installs a temporary, reconstituted TrueType font out of the Windows font. This reconstituted font is a new symbol font which encodes the formerly unreliably-encoded characters in "safe" character codes. The previewer thereafter automatically renders unreliable codes in the old font with the safe codes in the reconstituted font. The previewer builds and installs these reconstituted fonts as needed, and deletes them when exiting. The previewer keeps the reconstituted fonts private, so that they do not show up in font selection dialogs or in the Font applet.

Reconstituted fonts may be called for in pages you copy to the clipboard with `Edit+Copy` in the previewer. If you see characters appearing as '?' (question marks) when you paste previewed pages into other applications, check that font reconstitution is turned on in the `Options+Expert` menu. When using the clipboard with the previewer, you must copy, paste and render in the target application with the previewer still active and displaying the copied document. Metafile pictures on the clipboard are not portable to another system, or even to a later time on the same system. If you exit the previewer, the application that plays the metafile will show random font substitutions and character codes for the reconstituted ones that disappeared.

6 Windows 2000 Compatibility

(First available in 4.2B:) Windows 2000 is incompatible with several features in versions before 4.2B of TrueTeX Setup and the previewer. This is due to Windows 2000 reporting a peculiar version that breaks Microsoft's `GetVersion()` parsing. Setup 4.2A and earlier will not install fonts into the Windows 2000 registry, although the font files will be installed; this does not affect a prior installation of TrueTeX on a Windows 95/98/NT system subsequently updated to Windows 2000. Windows 2000 affects various features in the previewer 4.2A and earlier, including: encoding map for font (displays only first 256 characters), no UNICODE rendering (uses 8-bit rendering), default font list is taken from `WIN.INI`, PostScript printers not detected by the `\special` handler, and font metric exports will only use the first 256 character codes.

A very effective work-around corrects this problem in older versions of the previewer: the `apcompat.exe` tool on the Microsoft Windows 2000 CD in the `Support\Tools` directory. You can run this program directly from the CD, or you can install it on your system by running setup.exe in the `Support\Tools` directory of the CD. After launching this application, use the Browse button to select the previewer executable file (typically `c:\TrueTeX\TrueTeX\DVIGDI32.EXE`), select the operating system "Windows NT 4 SP3" (any service pack version will work), and select the check box "Make the above check box settings permanent."

(First available in 4.4D:) Release 4.4D corrects a problem on Windows 2000 when Eastern Europe encoding support is installed. Installing that support causes Windows to enumerate a variety of non-ANSI-encoded fonts for a single font, which the previewer previously did not distinguish as non-ANSI. On certain documents using non-TeX fonts (such as in the `times` package) this would cause incorrect or missing-glyph characters to be displayed. Release 4.4D and later take care to only accept ANSI and symbol encoded fonts into the previewer's run-time font dictionary.

7 Universal Modern Fonts

(First available in 4.1L.) The 9 new Universal Modern fonts are Windows ANSI TrueType versions of certain Computer Modern typeface styles. We created these fonts using the same precise techniques we used to convert the Computer Modern fonts to TrueType, except that the character sets and encoding implemented in the fonts follow the ANSI standard instead of the TeX versions. This conversion involved a substantial amount of new METAFONT designs, since so many of the ANSI characters are not found in the original TeX fonts.

Because the Universal Modern fonts contain the complete ANSI character set in a Windows ANSI encoding, they are fully compatible with non-TeX Windows applications using English or Western European languages. These fonts are especially suitable:

- For use with non-TeX applications to create a graphic inserts, tables, or other items to be included in TeX documents via the TEX \special command.

- For producing documents with text having a "TEX look", but using a non-TEX application.

The fonts and their names are as follows:

Font Name	Description
umb10	Universal Modern Bold
umbx10	Universal Modern Bold Extended
umbxsl10	Universal Modern Bold Extended Slanted
umbxti10	Universal Modern Bold Extended Text Italic
umitt10	Universal Modern Typewriter Italic
umr10	Universal Modern Roman
umsltt10	Universal Modern Typewriter Slanted
umti10	Universal Modern Italic
umtt10	Universal Modern Typewriter

8 Increased Capacities

(First available in 4.1G.) Pursuant to newly increased demands for formatting capacity, release 4.1G of the formatter provides the following generous capacities:

Category	Capacity	WEB variable
Strings	30 000	max_strings
String characters	300 000	pool_size
Macro string pool	270 000	string_vacancies
Main memory (4 bytes)	262 140	mem_max
Control sequences	20 000	hash_size
Font information	300 000	font_mem_size
Number of fonts	256	font_max
Input buffer	5 000	buf_size
Save stack	10 000	save_size

9 TIFF Image Library Hooks

(First available in 4.1C.) The SPTIFF32.DLL TIFF graphics \special handler has been upgraded to call upon the LIBTIFF.DLL TIFF-graphics library for reading bit-mapped files in WIN32 environments. The SPEPSF32.DLL PostScript graphics \special handler is likewise upgraded for handling bit-mapped previews in EPSF PostScript files, which use the TIFF format (EPSI format files use ASCII, not TIFF, bit-map data).

LIBTIFF.DLL is a 32-bit DLL which supports a wide variety of TIFF file variations (including black-and-white, grayscale, and color), a wide variety of color models (including 1- to 8-bit RGB palettes, 24-bit RGB triples, CMYK, and YCBCR), and a wide variety of compression methods (including uncompressed, PackBits, CCITT, LZW, and Thunder). LIBTIFF.DLL is a new, proprietary component of TrueTEX, and is based on the excellent TIFF library software of Sam Leffler of Silicon Graphics, Inc.

With the addition of this new TIFF library, the hooks which called on the shareware SimSoft Image Library have been deleted from the TrueTEX \special handlers. While that library claimed to support a wider variety of image file formats, it was unreliable in practice, and for legal reasons could not be distributed directly with TrueTEX (although it could be downloaded readily via the Internet). An optional \special handler will be made available that provides backward compatibility for those needing SimSoft features.

The 16-bit TIFF \special handler still uses the Black Ice TIFF.DLL. (You can tell which version of the previewer is running by examining the Help About message box.)

10 Improved Installation and Un-Install for Windows

The TrueTeX Setup utility, `setup.exe`, resides on TrueTeX Setup Disk 1 and controls the automatic installation of TrueTeX on your system.

A upgraded feature in Setup provides an automatic un-installation via an icon in the TrueTeX program group, accessed from the Windows Start menu. (First available in 4.1D.) Un-installing TrueTeX will unregister the TrueTeX fonts and remove the TrueTeX application files; it will *not* remove the directory tree itself or any files added there since installation, nor will it remove the program group from the start menu. You will probably have to shut-down and re-start Windows to completely recover disk space freed by the un-installation, since the un-install process defers removal of certain files that the Windows system locks (such as fonts that have been in use, and the `setup.exe` executable itself) until the next system re-start. After the re-start, you can use the Windows Explorer to remove the TrueTeX directory tree (but be sure that you first move or back-up any important files that you might have added there).

You can remove the Start Menu "TrueTeX" group from your Windows system by right-clicking the Start button, selecting "Open", then double-clicking "Programs" in the Start Menu window, highlighting the "TrueTeX" icon with a single click, and finally pressing the [Delete] key.

After un-installing, you can run Setup again from the distribution disks to re-install TrueTeX; this should not affect any extra files you may have retained in the directory tree.

TrueTeX no longer supports automatic installation in Windows 3.1.

The previewer application still installs in two versions, a 16-bit and 32-bit version. The 16-bit version is somewhat faster than the 32-bit version and is appropriate for ordinary TeX documents, while the 32-bit version can handle special features such as Unicode fonts (described below). The formatter is only a 32-bit application.

11 Enhanced Virtual Font Handling

(First available in 4.0R.) This release improves the way the previewer creates and uses virtual fonts; accompanying the release are new virtual fonts files implementing the DC fonts, virtual fonts for the LaTeX `times` package, and virtual fonts for the MathTime Type 1 fonts.

Virtual font handling changes in the following respects:

- **Nested virtual fonts** are now recognized. Earlier releases would not recognize nested virtual fonts; only the first level of virtual character mapping would have an effect. This was not necessarily a limitation, since nested virtual font schemes can always be "flattened" into an equivalent single level, and single-level virtual fonts can be rendered much faster than nested ones. However, since certain packages (like LaTeX `times`) supply nested virtual fonts, and converting them to un-nested equivalents is difficult, direct support for nested virtual fonts is desirable.

- **Actual font naming convention:** Earlier releases used a non-standard convention that a virtual font, such as `arial.vf`, could map to a font of the same name (in this case, `arial`); it was implicit that the mapped-to font (`arial`) was the actual TrueType font and not a recursive reference to the virtual font `arial.vf` itself. This implicit convention is incompatible with support for nested virtual fonts, so the following convention now applies: A virtual font may map to a font name which includes a parenthesized suffix, and this will force the previewer to treat that mapped-to font as an actual font of the same name, less the suffix. For example, if the virtual font `arial.vf` contains a mapped-to font reference of the form:

  (MAPFONT D 0 (FONTNAME arial (Actual))(COMMENT Arial TrueType in Windows))

  then the font name "`arial (Actual)`" refers to the actual font named "`arial`"; the previewer will not attempt to find a nested virtual font of that name. In the old scheme no longer supported, `arial.vf` would have contained an apparently self-referencing mapped-to font of the form:

  (MAPFONT D 0 (FONTNAME arial)(COMMENT Arial TrueType in Windows))

  If you continue to use old virtual fonts with this obsolete convention, they will result in a warning message that `arial.vf` contains a circular self-reference; however the virtual font will still work correctly because the previewer will interrupt such circularity by converting the reference to an actual font reference.

Recursive, self-referential virtual fonts, while not necessarily erroneous (see Knuth's example), are still not supported.

In accord with this convention, the `File Export Metrics` function now names references to Windows fonts with "`(Actual)`" suffixed to the name. If you are creating virtual fonts by hand, you can use more specific suffixes as reminders of the actual font source, such as "`(ATM)`" for Adobe Type Manager. The previewer makes no use of the suffix; it is merely looking for the leading parenthesis as an actual-font indicator.

Here is the precise rule used by the Previewer to detect names of actual fonts: If the font name referenced in a virtual font contains a left-parenthesis, the Previewer interprets the referenced font as an actual font with a name consisting of the referenced name truncated at the left-parenthesis. Thus the suffix "`(Actual)`" is not strictly required for you to indicate an actual font; any suffix with a left parenthesis will work, even a left parenthesis alone. Since TrueType and ATM font names do not use parentheses, this convention is orthogonally compatible with any system font names.

- **New virtual fonts: DC fonts.** A new set of virtual fonts provides access to the DC fonts using the Unicode rendering in the 32-bit previewer and the new naming convention for actual fonts. This provides correct rendering across all national versions of Windows 95 and Windows NT.

- **New virtual fonts: times package.** The LaTeX `times` package replaces the Computer Modern text fonts with the fonts Times (for Roman text), Helvetica (for sans serif text), and Courier (for typewriter text). These are implemented in the original package by three levels of nested virtual fonts, with the actual fonts being PostScript fonts rendered by the DVI-PostScript translator `dvips`. In TrueTeX, we simply replace the lowest level of these virtual fonts with new virtual fonts resolving to the TrueType fonts Times New Roman, Arial, and Courier, with appropriate character code remapping. We assume that the multi-lingual extended character sets are available in these TrueType fonts, which is the default in Windows NT and available in Windows 95 when the Windows 95 "multilanguage extensions" (see the Windows 95 help for details) are installed. If the multilanguage extensions are not installed in Windows 95, or if you are running under Windows 3.1, many accented non-English characters will not render when using `times`. Note also that the multiple levels of virtual font nesting will noticeably slow preview rendering of documents using these fonts.

- **New virtual fonts: MathTime package.** The LaTeX `mathtime` package replaces the Computer Modern math fonts with the MathTime fonts, which are Type 1 fonts stylistically compatible with the Times text font. The MathTime fonts require Adobe Type Manager (ATM) version 3.01 running on Windows 95 (ATM is not available for Windows NT as of this writing). A set of virtual fonts for TrueTeX maps the character sets properly so that the LaTeX output will render properly with the ATM font encodings. While the virtual fonts are freely available with TrueTeX, the MathTime Type 1 font files and Adobe Type Manager are proprietary products available from other vendors.

- **New encoding tables:** Several new encoding tables are now included in the dialogs for the `File Export Metrics` function; these tables represent various Unicode encodings and the PSNFSS "8r" encoding.

12 Omega Support

(First available in 4.0N.)

This release of the previewer supports all aspects of Omega, an extended version of TeX which supports enhanced multilingual capabilities and 16-bit font encodings. You can learn about the powerful capabilities of Omega, an opus of Yannis Haralambous and John Plaice, in *TUGboat* vol. 15, no. 3, September 1994, "Progress on the Omega Project". The Omega software is distributed via `ctan.org`.

Omega is a tool for the multilingual expert, and a means for TrueTeX to improve the performance of 8-bit TeX on Windows. The plain TeX and LaTeX macros are not directly usable in the Omega formatter, so the features are not immediately useful in the way an author would write conventional TeX documents.

12.1 Extensions for Omega

TrueTeX support for Omega includes the following features:

- The previewer interprets DVI files containing character codes larger than a single byte.

- The previewer loads 16-bit font metrics (Omega XFM or OFM files in addition to TeX TFM files) and virtual fonts (Omega OVF or XVF format in addition to TeX VF format).

- Virtual fonts can directly access and render all code positions in 16-bit Unicode fonts such as are available in Windows NT and Windows 95. This means you can use any national character or math symbol anywhere in a Unicode font.
- All extensions are upward compatible: the previewer still accepts the older TFM and VF files, as well as DVI files using the traditional single-byte encoding. The previewer will look first for an OFM or XFM file, and then a TFM file, when looking for a font metric file; and likewise for a OVF or XVF file, and then a VF file, when looking for a virtual font file.

12.2 Accessing Unicode Fonts

The most immediately useful aspect of the TrueTeX Omega extensions are the simple and complete access to the Unicode fonts of Windows NT and Windows 95. This means that you can avoid encoding mismatches when using non-TeX fonts in TeX documents. For example, the Unicode Times New Roman font in Windows contains all of the characters (or elements needed to compose accented characters)[1] used in the TeX Roman and LaTeX T1 encodings, and the Omega-enhanced TrueTeX previewer allows full translation between TeX and Unicode encodings via the Omega-enhanced virtual fonts. Access to Unicode allows us to finally rationalize the TeX fonts and typesetting into cross-platform standards, instead of employing ad-hoc 8-bit encodings that will not work when, for example, the Windows code page changes or when an output file is exported via Adobe Acrobat.

While both the 16-bit and 32-bit editions of the previewer support the Omega files, rendering with Unicode fonts requires the 32-bit previewer running on a Unicode-capable version of Windows (currently Windows NT and Windows 95).

12.3 Sparse Font Optimization and Unicode Virtual Packets

(First available in 4.0P.) This release adds optimizations in the Unicode edition to optimize handling of sparsely populated fonts. For example, Unicode multilingual fonts may have characters mostly in the range `0x0000`–`0x3ff`, plus a few others around `0x2000`, and around the private zone starting at `0xfb00`. The TrueTeX sparse-font optimizations store compacted tables for these fonts, instead of populating 64 K-entry arrays with mostly empty positions. This avoid large memory demands which otherwise would result.

This release of the Unicode edition also implements virtual character packets for codes above 256; previously virtual characters larger than 256 could only appear in virtual fonts as codes for actual TrueType fonts; now TrueTeX supports fully general TeX and Omega virtual fonts. TrueTeX also applies sparse table techniques to map virtual font character numbers to virtual font packets, thus avoiding inefficient use of memory.

13 Corrected and Updated Metric Export

(First available in 4.0N.)

New metric-export features allow you to access any character in any Windows font in a completely general way. (See the later sections in this document for more information on virtual fonts and the composition interpreter.)

13.1 Metric Export Formats Supported

The metric export (File-Export Metrics menu item) now supports both VPL (TeX Virtual Property List) and XVP (Omega Extended Virtual Property List) file formats when exporting font metrics. The VPL format supports 8-bit encodings (the various TeX encodings and Windows ANSI), while XVP supports not just 8-bit encodings, but also 16-bit encodings (such as Unicode).

TrueTeX includes console versions of the programs VPtoVF and XVPtoXVF translate the property list files to their respective TFM, OFM, XFM, VF, and XVF binary formats for use with TeX and Omega. When you export a metric file, TrueTeX will attempt to run the converter program in a console (Windows NT or 95) or DOS (Windows 3.1) window to automatically yield a binary metric file.

[1] Ooops, nearly all the characters. Unicode has no “polishlcross” component like Computer Modern, although it does contain the target character, “lslash”; and “dotlessj” is also not to be had.

Table 1: Ligature Rules Applied to Exported Fonts.

First	Second	Result	Description
Dashes			
hyphen	hyphen	endash	-- to endash
endash	hyphen	emdash	endash- to emdash
Shortcuts to national symbols			
comma	comma	quotedblbase	,, to quotedblbase
less	less	guillemotleft	<< to left guillemot
greater	greater	guillemotright	>> to right guillemot
exclam	quoteleft	exclamdown	!‘ to exclamdown
question	quoteleft	questiondown	?‘ to questiondown
F-ligatures			
f	f	ff	ff to ff
f	i	fi	fi to fi
f	l	fl	fl to fl
ff	i	ffi	ffi to ffi
ff	l	ffl	ffl to ffl
Paired single quotes to double quotes			
quoteleft	quoteleft	quotedblleft	‘‘ to quotedblleft
quoteright	quoteright	quotedblright	’’ to quotedblright

One important application of the XVP metric export is to create 8-bit virtual TeX fonts which use the Unicode 16-bit codes in Windows TrueType to reach accented characters and math symbols in Unicode fonts.

TrueTeX metric export also supports the obsolete PL (TeX Property List) metric file format, and the companion program PLtoTF, should you need this format for use with older TeX software. In this case you specify only an input font encoding, and the property list will reflect this encoding as applied to the TrueType font you select.

The TrueTeX *Users Guide* is incorrect in saying you must retrieve `PLTOTF` from the CTAN archive. It is now included in the TrueTeX binary directory.

13.2 Selecting Input and Output Encodings

(First available in 4.0P.) The list of encodings which TrueTeX knows is presented twice each time you export metrics, once each for the input and output encodings. Each selection corresponds to a `.cod` file in directory `\truetex\encoding`. A new entry at the end of the list:

```
[Select any encoding file from File-Open dialog]
```

allows you to select any encoding-specification file using a `File Open` common dialog. Currently such files must be in `.cod` format, which is explained in the TrueTeX *Users Guide*.

We have provided two AWK scripts, `codtoafm.awk` and `afmtocod.awk`, to convert encoding specifications from `.cod` to `.afm` and back. These scripts will work with the GNU AWK program, which you can obtain from various Internet GNU archives, or with Unix AWK.

13.3 Ligatures Supported

When you export metrics, TrueTeX puts the ligature rules of Table 1 into the TFM or XFM file, if the ligature characters exist in the font.

13.4 Getting Unicode Fonts in Windows

The Windows standard fonts (Arial, Courier, Times New Roman) can contain various sets of accented characters, depending on your version of Windows and options you have installed: Windows 3.1 will contain ANSI fonts only. Windows 95 will contain all the English and Western-European characters (Latin-1) by default; to get Eastern Europe, Greek, Cyrillic, etc., you must install the "Multilingual Options" extension (you can do this easily by consulting the Windows 95 Help topic, "multilanguage support, installing"). Windows NT automatically installs the multilanguage fonts, and also provides a Unicode demonstration font, Lucida Sans Unicode, containing 1700-odd characters.

Note that TeX text fonts like `cmr10` contain Greek capitals, so you should try to virtualize Windows fonts that contain these characters. For example, with the standard fonts like Arial, you can make a TeX virtual font that fills all the positions of fonts like `cmr10` except for the polishlcross and dotlessj characters.

You can examine the fonts on your system to verify how well-populated they are: use the TrueTeX Encoding Map on the `Text` menu (see below). This map works much better for viewing the Unicode tables than the Windows character mapper or font installer.

13.5 Limitations

Windows 95, which will display and print Unicode text from TrueTeX, unfortunately has a problem in that it will not report metrics or kerning for Unicode characters 256 and above (that is, the Microsoft GDI functions `GetGlyphOutlineW()` and `GetKerningPairsW()` do not work correctly in Windows 95). This means that you can export metrics for characters 256 and above only if you are running Windows NT. You can use the metrics TrueTeX exports under Windows NT to view and print in Windows 95, you just can't create the metrics themselves on Windows 95.

There is a work-around for this Windows 95 limitation, however. You can generate an AFM (Adobe Font Metrics) file for a TrueType font outside of Windows by using an accessory program `ttf_edit` included with TrueTeX. This program is a TrueType font encoding editor, and is documented in the file `ttf_edit.w` in directory `\truetex\bin`. For example, to create and AFM file for `arial.ttf`, use `ttf_edit` as follows:

```
C> \truetex\bin\ttf_edit arial.ttf font 3 1 afm > arial.afm
```

This will produce an AFM file giving metric information for the platform 3.1 (Windows, ANSI or Unicode) encoding of the Arial font. (warning: pending implementation as of 2 Jun 1996).

Now, using the previewer, choose `File + Export Metrics` for the Arial font, choosing the input encoding, "`Windows 3.1 ANSI or ATM (16-bit Unicode)''`. When TrueTeX encounters the Windows 95 metric export limitation at codes 256 and above, it will ask you if you want to use an AFM file instead of TrueType metrics, and you can respond by opening the AFM file `arial.afm` you have just created for that purpose.

13.6 Making 8-Bit TeX Fonts from Unicode Fonts

While exporting XVP files (Omega 16-bit font metrics) will connect the TrueTeX previewer to Windows' Unicode fonts, the TeX formatter requires TFM metric files, not Omega XFM or OFM files. If, when exporting metrics, you specify an 8-bit output encoding, the resulting virtual font will be compatible with the original TeX formatter and will not require the Omega formatter.

To create a TFM file for such a font, use the `xvptovpl.exe` filter to truncate the virtual codes in the XVP file and produce a truncated VPL file. For example, if you have exported `arial.xvp`:

```
xvptovpl < arial.xvp > arial.vpl vptovf arial
```

will create `arial.tfm` for use with TeX. You should delete the `arial.vf` file also created by these commands, since it does not properly map the 8-bit characters to Unicode.

The XFM and TFM files produced by this process will contain the same information; the XFM format is not needed in this case. The previewer will look first for an OFM or XFM file for a font in preference to a TFM file, if the OFM or XFM exists, so either format is acceptable to the previewer. The previewer does need the XVF file to map the 8-bit TeX characters to Unicode positions, however, and VF file would be incorrect.

13.7 Using Adobe Type Manager (ATM) for Type 1 Fonts with Windows 95

You can use Adobe Type Manager and Windows 95 together with TrueTeX, if you follow a few precautions. Adobe Type Manager 3.02 (the first version that worked under Windows 95) does not report the metrics and kerning for its fonts to Windows, so TrueTeX cannot learn what it needs to export TeX metrics by simply querying Windows. When you export metrics for an ATM font, TrueTeX will instead ask you for an AFM (Adobe Font Metrics) file which goes with the font you are are exporting TeX metrics for. TrueTeX will use the AFM file you provide as the source for items such as the character bounding boxes and kerning pairs. The encoding given in the AFM file is *ignored*, so do not depend on it as a means to re-encode fonts; the input and output encodings you specify to TrueTeX for the export are the ones which the export process will use.

ATM fonts are strictly 8-bit input encodings, so you should select the "`Windows 3.1 ANSI or ATM (8-bit)`" input encoding whenever you export metrics for an ATM font in Windows 3.1 or 95, and "`Windows 3.1 ANSI or ATM (16-bit Unicode)`" in Windows NT (in theory, anyway; as of this writing there is no ATM for Windows NT). The virtual font files (`.vf`, `.ovf`, or `.xvf`) will be specific to the 8-bit or Unicode environment respectively. ATM itself is lagging behind advances in Windows, notably in its lack of Unicode support and lack of a version for Windows NT.

Whenever you export metrics, TrueTeX tries to guess some sane values for the font's `ItalicAngle`, `Quad`, `SpaceWidth`, and `XHeight`. If you find that the spacings appear incorrect, you may need to alter these values in the `.vpl` (or `.xvp`) file and convert them with `vptovf` (or `xvptoxvf`).

TrueTeX does not (in this release) forward kerning to composed ligatures (that is, those generated by the composition script), even if the components are kerned. For example, if the pair "o" followed by "f" were kerned closer in the font, and the exported font contains a composed ligature "ffi", the string "office" would not be kerned between the "o" and the "f" unless you defeated the kerning with TeX commands.

Although installed fonts are supposed to be registered in the Windows 95 registry (so that applications can enumerate fonts), ATM 3.02 does not support the Windows 95 registry. You must therefore add the file `ATM.INI` to the `Options + Expert + Font Subsitution Files(s)` path to notify TrueTeX of the list of ATM fonts. You must do this before you can use ATM fonts in TrueTeX for exporting metrics or previewing.

TrueTeX does not use the composition information present in an AFM file.

This release also corrected a possibility of exported TeX metrics having negative heights for characters which were entirely below the baseline; the corrected software ensures that such below-baseline heights are always zero.

14 Using the Windows Registry

(First available in 4.0M.) The TrueTeX 32-bit previewer (`DVIGDI32.EXE`) can now load configuration information from the Windows Registry as well as from `.INI` files. You may use the Windows Registry in a natural way for any preference item where TrueTeX formerly used an `.INI` file, such as for the font substitutions and for the `TRUETEX.INI` file.

The Windows Registry is a feature of Windows 95 and NT which was not present in Windows 3.x. Windows 95 and Windows NT both use the Registry to store configuration information that was formerly stored in system files like `WIN.INI` in Windows 3.1. If you are unfamiliar with the Windows Registry, consult the Windows 95 or Windows NT Help for an introduction to it.

Since 16-bit applications cannot access the Registry, using the 16-bit previewer (`DVIGDI16.EXE`) in Windows 95 or NT requires that you provide a `.INI` file listing the fonts installed in the system. The format of `.INI` files is described in the TrueTeX *Users Guide* and in this document.

14.1 Using the Registry for Font Substitution

Since Windows 95 and NT keep the system-wide list of fonts in the Registry instead of `WIN.INI`, it is especially important that TrueTeX be configured to access the Registry for font substitution.

In the list of font substitution files, TrueTeX considers any name starting with the following canonical Windows key names as registry keys instead of file names:

```
HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS
HKEY_CURRENT_CONFIG
HKEY_DYN_DATA
```

For example, Windows 3.0 and 3.1 typically used the `[fonts]` section of the file `C:\WINDOWS\WIN.INI` to list the fonts installed on the system. In Windows 95 and Windows NT this same list is kept in the Windows Registry; Windows 95 lists fonts under the key `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Fonts` while Windows NT lists them under `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Fonts`. So where `WIN.INI` used to appear in the preference item `Expert + Font Substitution File(s)`, we would now use (in Windows 95, for example), `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Fonts`.

If you supply a Registry key for font substitution, but TrueTeX reports that it cannot find it, check: (1) That no backslash precedes "`HKEY`..." in the name as you have entered it; (2) That you have entered only backslashes (not forward slashes) in the name; (3) That you are using the right key for Windows 95 versus Windows NT (as noted above, the two systems differ over such items as the key name for the system font list); (4) That the entry actually exists in the Registry (you can use the Windows accessory `REGEDT32` (the Windows Registry Editor) to verify existence of keys and their values).

Note: release 4.1H corrected a problem which could cause the Previewer to crash at start-up when font substitution strings from the Registry totaled more than 8,192 characters, which typically occurs at about 400 or so installed fonts.

14.2 Using the Registry for Preferences

While Windows takes care of keeping system font information in the Registry, TrueTeX uses a private profile file to store its preference information. However, you can configure Windows to map the TrueTeX private profile into the Registry. To do this, use the Registry Editor to create a key named `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\IniFileMapping\TRUETEX.INI`, where `TRUETEX.INI` is the name of the TrueTeX initialization file (you can use another subkey name for the default `TRUETEX.INI` with the `-i` command-line option).

15 Message Window

(First available in 4.0M.) The previewer now directs many messages to a `TrueTeX Messages` MDI window, which scrolls the text of informational messages from the previewer. The previewer has grown in complexity to the point where many detailed messages and conditions can appear during a previewing session; the usual dialog box is a clumsy and ineffective method to display informational messages, since it requires your input to continue. The previewer also uses the message window to display messages from a document display window that cannot be interrupted by a dialog box.

You can use scroll bars to review lengthy messages that have scrolled off the display. You can minimize the window to ignore messages, or close the window to dispose of past messages. You can use the Cliboard to transfer message texts to other applications for printing or review.

Future revisions of the previewer will use dialog boxes only for messages requiring user action.

16 Enhanced Special-Handling in 32-bit Edition

(First available in 4.0L.) We have modified and enhanced the rendering calls to the `special()` function of 32-bit special-handler DLL's. While in WIN16 applications one can pass file stream pointers to DLL's, in WIN32 this is not always possible. Thus the "`f`" parameter (a `FILE *` pointer) which `DVIGDI32.EXE` provides to `special()` handler functions is modified as follows: this pointer will be `NULL` unless the `\special` text is lengthy (currently, "lengthy" means ≥ 1024 characters), in which case the pointer will be (on auction and rendering calls) a memory-mapped pointer for the underlying DVI file.

That is, the handler function may use the pointer plus an offset between `offset` to `offset+total_len-1` to access the full text of a lengthy `\special`.

16.1 Accessing Memory-Mapped Special-Texts

Note also the following cautions:

- To make pointer of proper type, your handler must first cast the pointer `f` (which has type "`FILE *`") to type "`LPVOID`".
- This pointer value points to the beginning of the DVI file, so you must add an offset of `offset` to `(LPVOID)f` to point to the start of the `\special` text.
- The memory-mapped text is *not* null-terminated, as is the case for the parameter `text`. Handlers must take care to access the text from `(LPVOID)f+offset` to `(LPVOID)f+offset+total_len-1` only, and must not rely on a null character termination for parsing.
- `DVIGDI32.EXE` passes this memory-mapped pointer only for lengthy `\special`'s. Typical `\special`'s are shorter than 1023 characters and will result in `(LPVOID)f` being `NULL`. This avoids the overhead of memory-mapping unless necessitated by the rendering of a lengthy `\special`.
- The pointer is a read-only pointer. Attempting to write to a pointed-to location will result in a WIN32 virtual memory exception.
- Liquidation calls now always send a `NULL` f. Formerly this happened to be a valid `FILE *` stream pointer for the DVI file.
- Virtual fonts can themselves contain `\special`'s, and (unless lengthy) such `\special`'s are indistinguishable to the handler. In the case of a lengthy virtual font `\special`, `DVIGDI32.EXE` will not memory-map the underlying virtual font file (that is, `(LPVOID)f` will always be `NULL`). That is, a lengthy `\special` in a virtual font will result in a `NULL` value for `f` although `text_len<total_len`. This makes lengthy `\special`'s in virtual fonts accessible only in their first 1023 characters via the `text` parameter. The `path` and `name` in such cases refers to the DVI file, not the virtual font file, and thus is no help in locating the `\special` text.
- Special handlers must not retain pointers to data or (in WIN16) file streams between calls. These pointers are temporary and will become invalid if, for example, the underlying DVI file is reformatted by another process.
- There is a slight possibility of a race condition should the underlying DVI file change (such as being reformatted by another application from the `.tex` file) while a special handler is rendering the text of a `\special`. Memory-mapping tends to insulate the text from such races, although there is a slight possibility of a mixture of old and new text appearing in the same special region if the total text was longer than a WIN32 virtual memory page (for example, 4 KB in Windows 95). Handlers should therefore strive to handle garbaged texts gracefully, since an incomplete rendering in such circumstances is of no consequence. This avoids the need for a complex (and non-TeX-standard) file-locking and synchronization mechanism between the TeX formatter and previewer processes.
- On liquidation and closing calls the pointer "`f`" is now always `NULL`.

16.2 Messages from special handlers

Formerly, `\special` handlers have been prohibited from displaying message boxes because they are called inside a previewer `WM_PAINT`. One disadvantage to this limitation is that debugging handlers is very difficult, since handlers cannot display debugging information interactively. A new feature in this release allows `\special` handlers to pass texts to the previewer for display in a message window. The handler should `#include` the header file `regmsg.h`, and then can send informational texts for display to the previewer via a message to the document window with a statement similar to this:

```
if (hWnd) SendMessage(hWnd,IPRINTF_MESSAGE,0,
    (LPARAM)"spcolr: Hello");
```

where `hWnd` is the window handle passed to the `special()` function in the `\special` handler (that is, the second parameter to `special`). (In WIN16 compilations, you will need to follow the cast to `(LPARAM)` with a cast to `(char far *)` in small-model code.) The previewer will display the string addressed by the `LPARAM` parameter. Take care not to generate such messages on every rendering pass, since this can result in numerous messages.

17 Font Encoding Maps

(First available in 4.0K.) A new previewer menu selection, `Text + Encoding Map for a Font`, allows you to choose a Windows TrueType font which the previewer then displays in an MDI child window, showing a table of the characters in the font and the raw encoding (such as ANSI or Unicode) for that Windows font. This display is of interest to those trying to use non-TEX fonts with TEX.

In the WIN16 previewer (`DVIGDI16.EXE`) or when the operating system does not support Unicode (such as Windows 3.1 with Win32s), the previewer displays the 8-bit encoding table, namely the codes `0x00` to `0xff`. The WIN32 previewer (`DVIGDI32.EXE`) displays the Unicode encoding when running under an operating system which supports Unicode output (such as Windows NT or Windows 95). The table shows the unmodified Windows encoding; none of the virtual font or ad hoc encoding tables are applied.

If the table does not fit in the child window (such as a Unicode table having 65536 code positions), you can use the `View` menu navigation commands, shortcut keys, and scroll bars to move the window up and down the rows of the table, much like moving about a document preview.

Each row of the table shows the hexadecimal code of the first character in the row. Rows are always a power-of-two characters wide to facilitate reading the codes in the rows.

The window displays characters in the point size you select in the font choice common-dialog; you can use a small point size to fit many long rows into one view, or you can use a large point size to show details in each character. Rows and columns automatically resize if you change the size of the window. Undefined characters will display as rectangles.

While `File + Print` and `Edit + Copy` items are not available for encoding map windows, you can print portions of the tables by using `Ctrl + Print Screen` to copy the active window to the Windows Clipboard, and pasting the Clipboard into any Windows application (such as the Windows Wordpad accessory) which can print the pasted result.

18 Initial Window Sizing

(First implemented in 4.0J.) Previous versions always opened the previewer as a full-screen (maximized) window. A new command-line option, `-d` *x y w h*, lets you specify the initial size of the TrueTEX previewer window. The coordinates (*x*,*y*) specify the screen coordinates of the upper-left corner of the window, and *w* and *h* specify the width and height respectively in pixels. If *x* or *y* is negative, the initial window will be maximized and the restore-size of the window will be set from (*w*,*h*) anchored at the Windows default position. If either *w* or *h* is negative, the width and/or height of the window will be extended to the respective edge of the screen. For example, "`-d -1 -1 -1 -1`" starts a maximized window, "`-d 100 100 500 500`" a 500 × 500 window anchored at screen coordinates (100,100). and "`-d 100 100 -1 -1`" a window anchored at (100,100) and extending to the right and bottom edges of the screen.

To start a minimized (iconic) window, check the `Run Minimized` option in the `Program Item Properties` dialog of the Program Manager, or set the appropriate value in `nCmdShow` in the `WinExec` call.

19 Long File Names

(First implemented in 4.0J.) TrueTEX (32-bit edition) now supports Windows NT and Windows 95 long file names. The previewer also understands file names and other arguments containing spaces, either through file names in the File Open common dialog, or (releases 4.1E or later) through arguments which are double-quoted on the command line. (Note: TEX cannot handle file names with spaces, so they are not acceptable to the formatter.)

20 DDE Server

(First implemented in 4.0J.) Expert and OEM users can now invoke TrueTEX previews from other applications via DDE (Windows Dynamic Data Exchange) "execute strings."[2]

The DDE service name is "`TrueTeX`", the topic is "`View`", and the items are the commands listed below. The TrueTEX DDE commands follow the standard syntax of Microsoft Excel and the Windows Program Manager.

[2] See the paper "DDE Execute Strings" on the Microsoft Developer Network CD.

20.1 DDE Commands Implemented

Here are the commands implemented:

- `[Open(`*command-line*`)]` Opens DVI file previews and sets options based on the text of *command-line*. The *command-line* argument may contain one or more DVI file names (you may omit the extension `.dvi`, and TrueTEX will automatically append it) and command-line options.

  TrueTEX will interpret *command-line* in the same way as a command line given in a launch via the Windows Program Manager, Windows 95 Start arguments, or a `WinExec()` call. Exception: issuing a print command via DDE (that is, the `-w` option in an execute string) will not cause the previewer to automatically exit after printing, unlike the automatic exit which normally follows with a Start, Program Manager, or `WinExec()` launch for printing. Follow DDE printing command(s) with a DDE `Exit` when you want the previewer to exit.

  DDE client applications should specify file names in DDE execute strings as fully qualified paths, since there is no reliable way for clients to know the previewer's current directory. Names unqualified by a directory or qualified by a relative directory are potentially ambiguous.

  Although the `Open` command opens a document preview window in the previewer's MDI frame window, it will not bring the frame window to the top of the Windows desktop. Typically you will want to follow an `Open` with a `Show` or `SetPos` command to bring the document into view.

- `[Show(`*SW-COMMAND*`)]` Sets the "show state" of the previewer frame (outermost MDI) window using the Windows API `ShowWindow()` function. Each of the argument(s) should be a string giving a symbolic name of a show state for `ShowWindow()`. For example, "`[Show(SW_SHOWMINIMIZED)]`" will minimize the previewer.

  Since the previewer examines the DVI file for changes on each repaint (if the `Re-Open DVI Files on Change` preference item is enabled), you need not close a previewed document when re-formatting. Simply ensure a repaint with a `Show` command.

- `[SetPos(`*hwndInsertAfter*`,`*x*`,`*y*`,`*cx*`,`*cy*`,`*fuFlags*`)]` Changes the size, position, Z-order, activation, etc., of the previewer frame window using the Windows API `SetWindowPos()` function. The parameters are as described in the Windows documentation for `SetWindowPos()`, except that *hwnd* is implicitly the previewer MDI frame window. Specify *hwndInsertAfter* as the string `NULL`, as a string giving one of the symbolic values such as `HWND_BOTTOM`, or as a `scanf()`-recognizable decimal or hex integer value of a window handle. The coordinates *x,y,cx,cy* must likewise be `scanf()`-recognizable decimal or hex integers. The *fuFlags* parameter should be one or more of the `SWP_...` symbolic values, with multiple values joined with a `|` operator. For example, "`[SetPos(HWND_TOP,100,100,300,300,SWP_SHOWWINDOW)]`" moves the previewer to the top of the Z-order in a 300 × 300 window at coordinate (100,100) and causes the window to be shown.

- `[Close(`*filename*`)]` Closes the first DVI child window with a file name matching the supplied *filename*. If *filename* does not contain a directory, TrueTEX will close the first child it finds with a matching name part; if *filename* is the empty string, then TrueTEX will close the first DVI child window. If you supply a file name qualified by a relative or absolute directory, the directory and file name associated with the child window must be equivalent to the path you supplied when you originally opened the preview window; however, matching will treat upper and lower case as equivalent and backslashes equivalent to forward slashes. To avoid ambiguity, use the same fully-qualified, absolute path name for both opening and closing a document. You can interactively examine the directory and file name for a given DVI child window in the previewer by selecting the `View Show DVI Info` menu item. Unlike `Open`, you must supply the extension (`.dvi`) as part of *filename*, as it is not implicit.

- `[Exit]` Exits the previewer, unless printing is active. If any print job(s) are active, the previewer will not exit, but instead will display a message box that the print job(s) must finish (or be cancelled by the user) before exiting. There is currently no way for a DDE client to inquire of or receive notice of when TrueTEX printing completes.

You may give more than one argument to `Open` or `Close`, separated by commas; the previewer will interpret each in turn. The `Show` and `SetPos` execute strings require the prescribed arguments. `Exit` must not have any arguments.

20.2 DDE Argument Syntax

TrueTEX interprets backslash (\) as an escape character in DDE execute strings to allow you to put commas, quotation marks, white space, newlines, etc., into string arguments. TrueTEX interprets the character immediately following a backslash literally, instead of using the normal syntax rules. This is similar to the C language syntax, although octal escapes and special characters (`\n`, `\r`, etc.) are not supported. You can also insert double-quotation marks by putting them in pairs, as in the style of Microsoft Excel or Visual Basic.

Note that you must use double backslashes in an execute string to indicate each single backslash in directory paths. A C program giving a literal string for a file name in an execute string therefore needs four backslashes to indicate each single directory backslash!

Double quotes around arguments are optional, but must surround any argument containing unescaped white space.

You can concatenate DDE commands into a single DDE execute string. Syntax errors (unquoted argument, incorrect number of arguments, etc.) in a given execute string will cause the DDE server to ignore the command and proceed to the next square-bracketed command (`[...]`) in the same execute string. An unknown command (for example, misspelling "`Open`") will abort execution of the rest of the execute string.

Semantic errors (files not found, malformed DVI files, missing fonts, etc.) will cause the previewer to display message boxes, as if the commands were given on the program item command line, but this will not affect the status value returned to the client. TrueTeX does not support any protocol for returning result information to the DDE client application, other than success (`bAppReturnCode = 0`) or failure (`bAppReturnCode = 0`). "Success" merely indicates that the command passed syntactical tests; it does not indicate any semantic success such as a file being successfully opened or printed.

20.3 DDE Service Enabling/Disabling

The `Options Expert DDE Server` preference item controls whether TrueTeX previewer responds to DDE commands. Disabling the service prevents the previewer from responding, which may be useful in debugging unexpected behaviors.

A DDE client will also receive no response if the TrueTeX previewer is not running. That is, the client or the user, when expecting previewer DDE service, must ensure that the previewer has already been launched, either from the Windows Program Manager or from an application (perhaps the client application itself) via a `WinExec()` call. Unlike the more advanced OLE servers, the Windows system has no registry of DDE servers to launch when a client requests a given DDE topic.

20.4 DDE Platform Interoperability

Various permutations of the Windows version versus the WIN16 or WIN32 executable versions of the program affect whether DDE execute strings work. In general, DDE will certainly work when the TrueTeX DDE server and the host system both use the same API word-length (WIN16 versus WIN32). That is, `DVIGDI32.EXE` DDE service always responds on Windows 95 or Windows NT, without regard to whether the DDE client is WIN16 or WIN32; likewise `DVIGDI16.EXE` DDE service always responds on Windows 3.1. The *exceptions* are in the following cases: `DVIGDI32.EXE` on Windows 3.1 does not respond to any DDE clients, WIN16 or WIN32; and `DVIGDI16.EXE` on Windows 95 does not respond to WIN32 DDE clients, although it does respond to WIN16 DDE clients.

The following table lists all the possible permutations:

Will Respond	Host System	DDE Client WIN16/WIN32	TrueTeX DDE Server
Yes	Windows 3.1	WIN16	`DVIGDI16.EXE`
No	Windows 3.1 + Win32s	WIN16	`DVIGDI32.EXE`
Yes	Windows 3.1 + Win32s	WIN32	`DVIGDI16.EXE`
No	Windows 3.1 + Win32s	WIN32	`DVIGDI32.EXE`
Yes	Windows 95	WIN16	`DVIGDI16.EXE`
Yes	”	WIN16	`DVIGDI32.EXE`
No	”	WIN32	`DVIGDI16.EXE`
Yes	”	WIN32	`DVIGDI32.EXE`
Yes	Windows NT	WIN16	`DVIGDI16.EXE`
Yes	”	WIN16	`DVIGDI32.EXE`
Yes	”	WIN32	`DVIGDI16.EXE`
Yes	”	WIN32	`DVIGDI32.EXE`

20.5 Programming a DDE Client

Adding DDE client features to your C application requires a small amount of new code: a few extra lines in your message-handling switch, and some short functions. TrueTeX uses “raw DDE” and your DDE client will be most economical in doing likewise, rather than using the Windows DDEML (DDE Management Library DLL) or the “standard DDE” library found on the Microsoft Develeper Network. Raw DDE is appropriate, since DDE execute strings do not involve hot links, Clipboard data passing, or other advanced DDE features.

The paper titled, “Raw DDE” on the Microsoft Developer Network is an excellent reference on this programming subject. The sample application “DDEPOP1” in Petzold’s book Programming Windows 3.1 gives a complete example of programming a DDE client. Note that this sample, however, does not provide for the WIN32 API.

Below are code excerpts from ddetst.c, a DDE client test application which compiles with Visual C++ (both WIN16 or WIN32 APIs, via a small amount of conditional compilation). This source code is available on DDETST.ZIP on the TrueTeX Setup disks (you must manually install this), or from our Web page (http://www.truetex.com). The excerpts below give only the code peculiar to raw DDE and omit the “boilerplate” Windows application code.

```
...
#include <dde.h>
...
HWND hServerWnd;                /* DDE server window                        */
...
/* We maintain a state variable to track the progress of a DDE execute      */
/* string conversation from initiation, to execution, to termination.       */
#define IDLE 1                  /* Idle, awaiting user input                */
#define INITIATING 2            /* Sent initiate, awaiting server ACK       */
#define EXECUTING 3             /* Sent execute, awaiting server ACK        */
#define TERMINATING 4           /* Got server’s exec ACK, terminating       */
int ConversationState = IDLE;


    case WM_DDE_ACK:
            /* DDE server response.  The ACK for both the        */
            /* initiate and the execute arrive here, so we       */
            /* must take care to discriminate between them       */
            /* by examining the transaction state.               */

            /* The first ACK we might get is the server’s        */
            /* response to our initiation.                       */
            if (ConversationState==INITIATING) {
                ConversationState = EXECUTING;
                hServerWnd = (HWND) wParam;
                DoDDE(
                        "[Open(\\\\kcc\\\\chk\\\\4192.dvi)]"
                        "[Open(\\\\kcc\\\\chk\\\\4196.dvi)]"
                        "[Open(\\\\kcc\\\\chk\\\\4197.dvi)]"
                        "[Open(\\\\kcc\\\\chk\\\\4198.dvi)]"
                    );
                ConversationState = TERMINATING;
                TermDDE();
                ConversationState = IDLE;
                }
            /* Otherwise we have the ACK for the execute;      */
            else if (ConversationState==EXECUTING) {
                /* Here we could examine the return status     */
                }
            /* This case is an error of synchronization        */
            else {
                }
            break;



void
IniDDE(void) {
    /* Initialize conversation with DDE server                  */
    ATOM app, topic;
    app = GlobalAddAtom("TrueTeX");
    topic = GlobalAddAtom("View");
    SendMessage(HWND_BROADCAST,WM_DDE_INITIATE,(WPARAM)hMainWnd,
            MAKELPARAM(app,topic));
    if (app) GlobalDeleteAtom(app);
    if (topic) GlobalDeleteAtom(topic);
    }

BOOL
DoDDE(char *cmds) {
    /* Send DDE commands to DDE server.  Return whether apparently */
    /* successful.                                                 */
    HGLOBAL hCmds;
    char FAR *p;
    if (hServerWnd==NULL) {
        MessageBox(hMainWnd,"Cannot connect to DDE server!",Title,MB_OK);
        return(FALSE);
        }
    hCmds = GlobalAlloc(GMEM_MOVEABLE|GMEM_DDESHARE,strlen(cmds)+1);
    if (hCmds==NULL) {
        MessageBox(hMainWnd,"No memory for DDE!",Title,MB_OK);
        return(FALSE);
        }
    p = GlobalLock(hCmds);
    if (p==NULL) {
        GlobalFree(hCmds);
        MessageBox(hMainWnd,"No memory for DDE!",Title,MB_OK);
        return(FALSE);
        }
    lstrcpy(p,cmds);
    GlobalUnlock(hCmds);  p = NULL;
    if (!PostMessage(hServerWnd,WM_DDE_EXECUTE,(WPARAM)hMainWnd,
#if _WIN16
            MAKELPARAM(0,hCmds)
#elif _WIN32
            (LPARAM)hCmds
#endif
            )) {
        GlobalFree(hCmds);
        MessageBox(hMainWnd,"Cannot connect to DDE server!",Title,MB_OK);
        }
    return(TRUE);
    }

void
TermDDE(void) {
    /* Terminate conversation with DDE server                      */
    if (hServerWnd==NULL) return;
    PostMessage(hServerWnd,WM_DDE_TERMINATE,(WPARAM)hMainWnd,0L);
    hServerWnd = NULL;
    }
```

21 Recursive Directory Searching

(Enhanced in 4.0H.) Expert users can use recursive directory specifications in TrueTeX configuration items. By appending a double backslash to any directory, you indicate that you want TrueTeX to search that directory and any subdirectories thereof. For example, you can specify the `TEXINPUTS` to include a directory like `\truetex\macros\\`, meaning you want TrueTeX to look for `.tex` input files in directory `\truetex\macros` and any subdirectories. TrueTeX searches the initial directory you specify, and thereafter in the "depth-first" order.

Use caution with this feature. If you specify recursive searching in a large directory tree or across a network, performance may suffer as TrueTeX walks the tree for each file opened. LaTeX and many other TeX macros test for non-existent files (such as LaTeX `.aux` files) by attempting a file-open, and this can result in lengthy searches with no apparent progress.

22 Using Multiple Configuration (`.INI`) Files

(Enhanced in 4.0B.) TrueTeX stores all configuration and preference information for its programs in an initialization file. By default TrueTeX names this file `TRUETEX.INI` and keeps it in the same directory as the TrueTeX executable programs. If you want to configure TrueTeX for several users or for several preference configurations, you can set up any number of configuration files.

There are two ways for you to specify to TrueTEX where it should take its configuration information: a system-wide change or a per-icon change.

First, you can make a system-wide change by editing the Windows system initialization file, `WIN.INI`, to add a section "`[TrueTeX]`" containing an entry "`INIFILE=`*Path*", where *Path* is the name you choose for the `.INI` file you wish to use.

Second, you can use the "`-i`" command-line option to specify the `.INI` file to be used by specific TrueTEX Formatter and previewer icons. This option overrides both the default `.INI` file and any system-wide file you may have specified in `WIN.INI`.

To specify an `.INI` file for an icon, select the icon in the Program Manager and then choose `File` and then `Properties`. In the "Command Line" field of the Program Item Properties dialog, append a space and the option "`-i` *Path*", where *Path* is the name you choose for the `.INI` file you wish to use. (Note that there must be a space between "-i" and "*Path*".) If you enter a *Path* which specifies a file that does not yet exist, TrueTEX will create the file with default configuration and preferences the first time you launch the icon.

You can actually specify the *Path* parameter in several forms:

Example	Description
`-i c:\mydir\myfile.ini`	A fully qualified path and file name. The file is unambiguously specified. This is the safest option to use.
`-i c:\mydir\`	A directory only, indicated by the ending backslash, and no file name. TrueTEX will use the default name `TRUETEX.INI` in the directory you thus specify.
`-i myfile.ini`	A file name only. This will use the Windows system directory to find or create the `.INI` file you have named. Use this form with caution, as it affects the Windows system directory.
`-i .\myfile.ini`	A relative path and file name. TrueTEX will use the file you have named in a directory relative to the "Working Directory" in the File Manager's "Program Item Properties." Note that the resolution of this path will change if you later change the Working Directory for the icon, so use caution with this method.

If you need help debugging the use of custom `.INI` files, the `Other Show environment` menu item in the Formatter and `View Show DVI Info` item in the previewer will reveal the fully-qualified name of the `.INI` file used by the respective TrueTEX program.

The extended-DOS version of the Formatter, `INITEX3X.EXE`, also uses the command-line option `-i` to locate its `.INI` file. This allows you to either share or separate usage, as you wish, of the preferences used by the various DOS and Windows TrueTEX programs.

23 Initial Working Directory

(New addition to 4.0B.) When you launch the TrueTEX previewer, it changes the initial working directory to that of the most-recently-used file. The "Working Directory" specified in the Program Manager icon will be used only (1) to locate the `.INI` file when you have specified a relative *Path* with the `-i` command-line option (see table above), and (2) as the default initial working directory if there is no history of viewed files in the `.INI` file.

24 Quiring

Although an elaborate quiring feature is planned for TrueTeX, for now a simple command-line option allows some primitive quiring. This option allows simple N-up printing for making labels or booklets. Enter command-line options in the form:

`-q` *Xoff Yoff Xitems Yitems*

which directs TrueTeX to step-and-repeat the contents of each page in an array of items *Xitems* wide by *Yitems* high with a distance between adjacent items of *Xoff* horizontally and *Yoff* vertically (distances in inches). You will see the effect both in previews and printed output.

For example, "`-q 4.16 3.33 2 3`" means to print an 2 by 3 array of the page image, with the array items stepped 4.16 inches horizontally and 3.33 inches vertically. This option corresponds to Avery 5164 self-adhesive label sheets for laser printers.

The print the TrueTeX *User's Guide* masters, we printed on US legal size paper ($8\frac{1}{2}$ inches by 14 inches) in full-duplex, landscape orientation, using TrueTeX option "`-q 7 0 2 1`". This prints a master book yielding two copies, which are power-cut along the midline of the sheets before binding.

You may specify a negative or zero distance for the quiring offsets to achieve certain effects. However, a zero or negative value for the *Xitems* or *Yitems* counts will cause TrueTeX to render all pages blank.

If you need a more general quiring facility, consider Rokicki's `dvidvi` program, which rearranges the pages in a DVI file into a new DVI file with quired pages. The new DVI file will preview and print appropriately with the TrueTeX previewer.

25 Virtual Fonts

TeX experts now enjoy full support in TrueTeX for virtual fonts, which are a TeX-standardized method of re-encoding and re-composing fonts. Typical applications of virtual fonts include: re-encoding TeX output characters, composing diacritical glyphs from the bare characters plus accents, simulating 8-bit encodings with 7-bit fonts, and mapping exotic glyphs to `\special` commands. You will find an explanation of the concepts in "Virtual Fonts: More Fun for Grand Wizards," by Donald Knuth, in *TUGboat*, Vol. 11, No. 1, April 1990, pages 13–24.

The explanation below assumes you understand the concepts of virtual fonts, as described in the reference above. The details are quite technical, but as we said, it is a feature for TeX experts.

Virtual fonts affect the operation of the previewer as follows (The following convention was modified in Release 4.0R; see above). If there is an "actual" font (that is, a TrueType font) for a font's external name, then the font is treated in the conventional (non-virtual) way, using the TeX or ANSI composite encoding as appropriate for the name. If there is no actual font for a given name, then the previewer expects to find a virtual font file (*fontname*`.vf`), and treats the font as a virtual font according to the instructions in the virtual font file. The previewer uses the same directory path for virtual font files as it does for TFM files, so the setting of `Options Expert Path for TFM Files` determines the directories where TrueTeX expects to find `.vf` files.

In previous TrueTeX versions which did not implement virtual fonts, all TeX font external names had to have an "actual" realization, which in the case of TrueTeX meant that they had to have a TrueType font with the same external name. With virtual fonts, a font name may refer to a virtual font file which gives instructions for rendering the font in terms of compositions from other fonts. A virtual font may also be recursive: it may refer in turn to other virtual fonts, as long as the chain ultimately leads to a glyph in an actual font.

Note that there are several details of how TrueTeX implements virtual fonts in addition to the TeX standard:

First, virtual fonts are never re-mapped. TrueTeX does not apply composite encoding to virtual fonts themselves or when rendering actual fonts used within virtual fonts. It is assumed that the virtual font mechanism will appropriately re-map the TeX font codes to the proper Windows ANSI codes. This means that if you want to make a virtual font out of several of the Computer Modern fonts, you must incorporate the "ad hoc" re-encoding table (described in the *User's Guide*) into your virtual font. By not re-encoding the fonts, TrueTeX virtual fonts allow you complete and simplified access to all glyphs in any font. Non-virtual fonts continue to be re-mapped with either the "ad hoc" or Windows ANSI tables; you can think of this behavior as if TrueTeX had a "default" virtual font (consisting of the re-mapping tables) for every top-level non-virtual font.

Second, in the case where both a virtual and actual font exist for a given external name, TrueTeX will give precedence to the virtual font at the top level. In keeping with the virtual font standard, TrueTeX prefers to treat recursive fonts within virtual fonts as actual (that is, TrueType) fonts, and treats them as virtual only if there is no TrueType version. At the top level, this bias is reversed, and TrueTeX attempts to first find a virtual font file for each top-level font in the document.

In this way we can use both a virtual and a TrueType version of a given font, without having to multiply the number of names or change the names of fonts in TeX or in the TrueType font system. For example, virtual font file `cmr10.vf` might render glyphs in terms of the font with the same name `cmr10`; TrueTeX would understand that the virtual font file is defining the virtual characters in terms of the TrueType font.

The usual virtual font scheme is to always prefer to treat font names as actual fonts, thereby precluding infinite recursion. The TrueTeX scheme is slightly different: A given font name may have both an actual and virtual realization. That is, the name may match both a TrueType font and a virtual font file, and the recursion rule sorts out which is used and when. This change presents no conflict with the usual virtual font scheme of always prefering the actual font.

We plan to extend the TrueTeX TFM export feature to also export virtual font files to facilitate your use of Windows fonts. The CTAN TeXware packages like `vptovf` and `fontinst` are good expert tools for creating virtual fonts which are compatible with TrueTeX.

26 TeX Font Names

In previous releases the previewer had a fixed list of font external names that were to be taken as TeX fonts. Now that list is made at start-up based on patterns in a preference item (`Options + Expert + TeX Font Names` ...). You may set this string to a series of patterns which the previewer will match against font names; the previewer will consider any font whose external name matches one of the patterns to be a TeX font using the “TeX ad hoc” encoding; the previewer will apply ANSI encoding to all other fonts. The default value of the string is:

```
cm*;eu*;lasy*;lcircle*;line*;msam*;msbm*;logo*
```

which replicates the behavior of the tables built into previous releases. The patterns in the default string refer respectively to: Computer Modern, Euler, LaTeX symbols, LaTeX circles, LaTeX lines, AMS Math Symbols “A” and “B”, and the METAFONT logo font. In the patterns, you may use an asterisk (`*`) as a wild card; the general pattern syntax follows that of the Unix `grep` and `sed` programs so that you may indicate more complex patterns. The preference item value is given as `TEXFONTNAMES` in the `.INI` file.

Note that you should not specify a Belleek-encoded font, such as the TeX DC fonts (`dcr10`, etc.) as “TeX fonts” in this regard, since they do not use the TeX ad hoc encoding method.

27 `\special` Handler Sources

The C source code in the `\special` handler kit has been rearranged to promote sharing of common code between the various handlers. (27 Jan 1995).

28 TEXWAKEUP Preference for initex3x.exe

You may place in the .INI file (or the environment variables) an environment variable `TEXWAKEUP`, which, if it is set to any non-empty string, will result in the DOS-extended formatter issuing a beep (ASCII 7) code to the standard output whenever TeX is expecting input. This is useful if you run TeX in a Windows iconized MS-DOS session, since you will get an audible signal when TeX is waiting for your input. (Previous editions of `initex3x.exe` always beeped; this was not always appropriate.) The Windows formatter will restore and activate the window when a TeX error occurs.

29 Programming with the Setup Utility

The TrueTeX Setup program is our own custom application, and should not be confused with the Microsoft Setup Toolkit programs, InstallShield, or other third-party installers. It is a small, quick, and versatile executive for the installation process, and supports font installation, which many install utilities do not. This section is a detailed guide to using Setup and a guide for experts programming Setup for use with other applications.

29.1 How Setup Organizes Distribution Disks

TrueTeX Setup uses a file called `manifest` on the first Setup disk as a script to guide the installation process. You can examine this text file for details on what happens during installation but is ordinarily hidden from view. You can modify the `manifest` file to create a custom Setup for networked or site-licensed copies (assuming you have paid the appropriate royalties to us to legally make internal copies of TrueTeX, of course).

29.2 Modifying Setup for Network-Master Installation

You can make a master network copy of the TrueTeX installation resources on any host file server; from this master copy any user on the network could install TrueTeX on a given system. Such a master copy simply consists of all the files on the TrueTeX setup disks, which you copy into a single directory on the file server. Networked users can then run the Setup program from the file server's copy instead of from floppy diskettes. Setup and the `manifest` file do not depend on any particular arrangement of files on the Setup disks and will automatically adapt to the situation where all the files for installation are located together in a network file server directory.

29.3 Setup command-line syntax

Setup recognizes the following command-line syntax:

`setup [opts]` Use this form for a "first pass", that is, there are no arguments, and no `-u` option. After loading the manifest (which is not acted upon in the first pass other than to check syntax and note command-line options), displays a welcome message (unless `-s` is in effect), invokes an installation destination dialog (unless `-f` given), copies itself to the destination directory, and launches the "clone" with "second pass" arguments described below. The cloning is necessary because the executable file on the floppy disk (which floppy will possibly be removed and replaced with another during the installation process) cannot be removed during installation. If `-u` is in the options, the invocation is for an un-installation (see below).

`setup [opts] src dest` This form is a "second pass," and is usually only used by the first pass to pass information to the "cloned" copy. Installs from files from the `src` directory to the `dest` directory using a manifest file (default name is "`manifest`" or other value from the `-m` option) relative to `src` directory. Not a valid form for un-installation (`-u`).

Command-line options are as follows. Certain options must precede others to take effect. The option arguments may not contain spaces; thus a there is a limitation on WIN32 long file names (they cannot contain spaces).

- `-v`: Toggles verbose mode, which is useful for debugging.
- `-s`: Toggles installation silent mode, which omits routine messages, and only displays messages when errors occur. Use with `-f` if you want absolutely no messages in the absence of errors; otherwise a target-directory selection dialog will still appear. Using `-v` will override `-s`. Silent mode does not affect un-installs very much; we feel that the user should know and confirm when things are being removed from his system.
- `-q`: Like `-s` silent mode, except for un-installation. Use with caution.
- `-p` *name*: Uses product name *name* in dialogs. Title bars in windows will show this, followed by "Setup" (or "Un-Install" if `-u` preceded). It will also be used as the "Start Menu" subdirectory for installation/un-install.

- `-d` *dest*: Uses *dest* as the default destination (target) directory for installation or un-installation. This is the initial value that will appear in the user dialog prompting for the target directory, or giving notice of a mandatory target directory, if the `-f` option is given, where the installation/un-installation will begin (the dialog or notice will not appear if `-s` is given). Not valid for 2nd pass invocation.

- `-m` *manif*: Uses *manif* as the starting manifest file name. This is valid during 1st or 2nd pass invocation or un-install. The 1st pass does not load the manifest, but will pass along the name in a `-m` argument to the 2nd pass invocation, where it will be loaded and used without further notice.

  You can specify more than one manifest file name by separating them with commas, such as "`-m execs,fonts`" specifying that files `execs` and `fonts` in the source (during install) or target (during un-install) directory will be used as starting manifests. Manifests can in turn call other manifests in a nested fashion; see the TrueTeX "`manifest`" file comments for a description of the format.

  Note that Setup will compute a full path name of the manifest file which is the concatenation of the source directory with the manifest file name (default `manifest` or `-m` value) during an installation (that is, `-u` was *not* specified), or the destination directory (that is, `-u` was specified). To specify such a destination directory during un-installation, use `-d`, and perhaps `-f`.

  It is your responsibility as the setup media creator to make sure the manifest is present in the source directory on the installation media for installation files, and that the the installation manifest installs a suitable un-installation manifest into the target directory (assuming you want to provide an un-install feature).

  Manifest script files can direct Setup to install or un-install software with a number of actions, described fully in the TrueTeX manifest file on TrueTeX Setup Disk 1. Actions can create sub-directories of the target directory, make any of those directories the current working directory, execute programs from the source media (typically self-extracting archives), copy and remove files directly by name, stream-edit text files to replace placeholder strings with the target directory (useful for creating customized `.INI` files), send DDE commands to the Windows program manager to create and populate program groups in the Start Menu, and verify that the run-time environment is a proper version of Windows, and load other manifest files.

  Setup loads the entire manifest into memory from one or more manifest files and closes the files before starting to perform the manifest actions. Besides checking the syntax of the manifest, this permits the manifest files to originate on removable media volumes that are switched during the installation process.

- `-f`: The destination directory (default or `-d` destination) is fixed. The user will not be presented a dialog offering a chance to change the default. Not valid for 2nd pass invocation.

- `-u`: Un-installs: invokes an installation destination dialog, and uses file name `manifest` in the destination directory (or another manifest name relative to the destination with the `-m` option) to un-install files and fonts. You probably should use `-d` with this to specify where the application's un-install manifest resides, since the usual convention is to name the installation (not un-install) manifest with the name "`manifest`". Use `-p`, too, before the `-u`. Do not use source and destination arguments with the un-install option; the un-install invocation should contain only dashed options.

  An un-installation manifest may be unable to directly delete certain files, such as executables files that are running (the `setup.exe` program itself, for example) or fonts that have been opened and locked by the operating system. The un-install mode of Setup will schedule such files for deletion on the next system re-start, using either the Windows 95 `WININIT` method or the Windows NT `MoveFileEx` method, as is appropriate to the run-time environment.

  The reasonable expection of a user is that an application's un-install manifest is a "mirror image" of the installation. That is, the un-install manifest should list all the files and fonts that the installation originally created.

  It is not an error, nor will a message appear, if a file to be removed by a manifest is not present or a font to be unregistered is not registered. This allows you to create a global un-installation manifest that covers the whole application, even though you may only have installed part of it. You can also create manifests which will register and un-register sets of fonts in the Windows system, without altering the underlying TrueType font files.

  Un-install manifests should unregister fonts before removing the corresponding TrueType files. This order is not important in most situations, however, as the same result will exist after a system re-start.

  The un-install mode of Setup has no provision to remove directories. This means application directories, perhaps empty, will always remain despite un-installation. You should advise the user that he should examine the vestigial directories for any critical files that might remain (such as documents that might have been created in the TrueTeX application directory tree), and that he should manually remove directories using the Windows Explorer or other method when he is satisfied that they are empty.

  Option `-u` is not valid in a 2nd-pass invocation.

29.4 Other Setup Programming Considerations

You can create various sub-manifests for components of your application, and invoke Setup with combinations of them for customized installations, with the link '`l`' manifest action.

Later versions of `setup.exe` (TrueTEX releases 4.1E and later) provide a '`u`' manifest action which will unzip archives using the `UNZIP32.DLL` library. Formerly, Setup called on self-extracting executable archives to unzip themselves; these were DOS programs and lacked some grace in running under Windows. The new, native unzipping ability in Setup means that the entire process runs in a controlled, 32-bit Windows environment.

Distribution media for use with Setup should contain not only `setup.exe` and the installation manifest (and possibly sub-manifests called by the starting manifest, depending on your design), but also `UNZIP32.DLL` if the manifest calls for unzipping ('`u`') actions. If your manifest uses "`x`" actions (now obsolete; use unzipping instead), then you must also include `execfrom.exe` and `execfrom.pif`. An un-installation manifest is strongly recommended. These recommendations imply that the installation manifest should begin with "`c`" actions for these auxiliary files (but not `setup.exe` itself, which Setup must copy itself before the manifest is loaded).

29.5 Examples of Using Setup

`a:setup`	Does an interactive install of TrueTEX.
`a:setup -p Blatz -d C:\Blatz`	Does an interactive install of "Blatz".
`a:setup -s -f -d C:\TrueTeX -m manifest`	Installs to a known directory, silently.
`c:\TrueTeX\setup -u -f -d C:\TrueTeX -m uninstal`	Un-installs from a known directory, `C:\TrueTeX`.

30 Other Network Issues

(New in 4.0B.) If you purchased a multi-user (network) copy of TrueTEX, you can install just one copy of the TrueTEX programs on a network server and then create program groups and icons on various client machines which launch the applications from the server's `.EXE` files. You can customize the icons to specify a configuration file (`.INI` file) for each client system; see the section "Using Multiple Configuration Files" above for details.

31 Character Composition in Exported Font Metrics

The font metric exporter attempts to make exact matches betweeen input and output encodings where possible. For example, the character "A" is usually available in both input and output fonts, and so the virtual font usually contains a simple re-mapping between the associated codes.

A more difficult case is where the exporter must compose characters which are not available in the input fonts. For example, if the output font contains an "A with umlaut accent" (Ä) character, the input fonts may not contain the accented character, although they may contain both an "A" and an "umlaut accent" (¨). In this case the exported virtual font should contain a virtual character packet that sets the letter A with a superimposed umlaut accent.

Composing characters requires that the exporter have some geometric sophistication, since simply overlaying accents often does not yield a visually pleasing result. Typically the bounding boxes of the marks must be aligned according to rules specific to the letter and accent, and correction for italic slant must also be taken into account.

Some compositions do not even fit the letter-plus-accent model. For example we may wish to simulate a ligature "fi" by tucking a dotless "i" closely against an "f".

In order to provide a general and extensible method for composing virtual TEX characters from deficient input fonts, the TrueTEX metric exporter contains an interpreter for a character composition language. This language generally resembles a subset of PostScript operators and stack-oriented semantics.

TrueTEX includes a base composition script that does a fair job of composing characters used by TEX, and this is the default script used to export font metrics (you can write your own scripts to cover new cases or if you want to fine-tune the placement of composed character elements). The base script attempts to compose output characters from input characters in four ways:

- Accent-plus-letter composition: for example, making an "Amacron" from a letter "A" plus an accent "macron". The script knows how to place the following accents, including skew for italic correction:

Composable Accent Characters		
Name	Position	Example
umlaut	top-center	ö
acute	top-center	ó
breve	top-center	ŏ
caron[1]	top-center	ô
cedilla	bottom-center	o̧
circumflex	top-center	ô
comma	top-right	Ľ
dieresis	top-center	ö
dotaccent	top-center	ȯ
grave	top-center	ò
hungarumlaut	top-center	ő
macron	top-center	ō
ogonek[2]	bottom-right	
period	top-center	ȯ
ring[2]	top-center	

[1] For D/d/L/l, changes to comma at top-right.

[2] Accent is not present in the Computer Modern fonts used in this portable document.

- Ligature-from-sequence composition: the script will compose ligatures from separate characters if the input font does not contain the ligated character: ff, fi, fl, ffi, ffl, IJ, ij, SS, Æ, æ, Œ, and œ.

- Synonyms: the script will recognize variations on character names:

hyphenchar	hyphen
ng	eng
Ng	Eng
Uhungarumlaut	Udblacute
uhungarumlaut	udblacute
Ohungarumlaut	Odblacute
ohungarumlaut	odblacute
Zdotaccent	Zdot
zdotaccent	zdot
hyphenchar	hyphennobreak
compwordmark	zeronobreakspace
Idotaccent	Idot
dbar	dmacron
visiblespace	spaceliteral
etc.	

- Fallbacks: the script will substitute certain closely similar characters for missing accents, such as a period for a dotaccent.

When you direct TrueTeX to export font metrics, it will call upon a set of scripts in the character composition language to compose output characters which have no exact matches in the input encodings. These scripts are ASCII files containing the PostScript-like language. TrueTeX runs the script "`compose.ps`" in the font-encoding-files path preference item. If for some reason you need to disable the composition script, you can rename or move this file so that TrueTeX cannot find it; then TrueTeX will produce a valid metric export file, although you will get a message that TrueTeX could not find the composition script, `compose.ps`.

31.1 Composition Language for Experts

If you are a real expert in TeX and PostScript, you can modify `compose.ps` for special purposes.

The language syntax, data types, and operators all closely follow a subset of PostScript. We will describe what this subset is, and refer you then to a PostScript language reference for details on the language.

Remember to exit and restart TrueTeX if you make changes to the composition script, since TrueTeX only reads `compose.ps` once per instance, when it first exports a metric file.

Here are the pre-defined PostScript-like commands available in the composition language:

```
add and array begin clear cos count currentdict cvi cvn cvr cvs cvx def dictstack dict div dup end eq
exch† exec exit false† forall getinterval get ge gt idiv ifelse if† index known length le lt mod mul
neg not or† pop put readonly roll round sin stack store† string sub tan† true† vpl_literal vpl_set
vpl_wd_ht_dp_ic where
```

(The commands marked "†" are actually macros defined at the beginning of the script `compose.ps`.)

The built-in commands above have the same effect as their PostScript counterparts. A few commands are specific to the metric exporter, and have the following function:

string	vpl_literal	–	Emits literal property-list text for *string*, indented, followed by new line.
wd ht dp ic	vpl_wd_ht_dp_ic	–	Sets width, height, depth, and italic correction of character to be output.
x y code	vpl_set	–	Emits property-list text to set input character *code* preceded by relative motion of *(x,y)*.

Since the task at hand is limited, the non-relevant portions of full PostScript are not implemented, and the function of some operators is restricted. The file scanner is very simple and requires all tokens be separated by white space; for example, procedure delimiters (curly braces) must be separated by white space from the adjacent items. The data types supported are: integer, Boolean (as integer), double, name, procedure, dictionary. There is no Boolean type as in PostScript. Instead, the semantics overload integers for logical (Boolean) values, as in the C language. That is, integer 0 is false, anything else is true. There are also no bitwise operators as in PostScript. PostScript overloads the operators "and", etc., and maintains separate integer/Boolean types; we overload the type and would require separate operators. Integers must start with a digit (subtract from zero to get a negative value), floating point numbers must start with + or -, literal names with slash; anything else will be considered a name. Literal names can be decimal number strings, like "`/48`", as we illustrate below (in standard PostScript a number cannot be used as a literal name). Certain polymorphisms and coercions provided by PostScript are not supported, notably: dictionaries can only use names as keys; operators like `roll` that need integer arguments will not coerce floats to integers.

The systemdict, concomitant with the Boolean overloading of integers, contains definitions of "true" as 1 and "false" as 0.

The current implementation does not implement the following data types and operators: bitwise arithmetic, marks, random number functions, packed arrays (unpacked arrays are implemented), virtual memory, files, resources, the "miscellaneous" category of operators in the Adobe PostScript reference, graphics-related operators (graphics state, coordinate system, matrix, path construction, painting, insideness, forms, patterns, device setup and output), character-and-font operators, interpreter parameters, and Display PostScript. The interpreter maintains separate operand, dictionary, execution, and VM stacks as in PostScript.

The TrueTeX metric-exporter is connected to the interpreter such that five phases of execution occur: (1) the exporter exec's the interpreter's `start()` function (which is analogous to the PostScript start command, but not accessible to the composition script), (2) the exporter loads the composition macro file `compose.ps`, (3) the exporter does a begin followed by the operators and data to create the font information in the interpreter environment, (4) for each "composable" character, the exporter executes the definition of `Compose` in the interpreter, pushing the output character name as an argument, and (5) the exporter exec's an end to clear the font information loaded in (3).

The exporter does phases (1) and (2) only once per invocation of the previewer. It does phases (3) through (5) for each font exported. The exporter allocates and frees VM between steps (3) and (5), so that memory is reused between each composition. Exporting a font require several hundred KB of free memory.

A "composable" character is an output character which has no exact match in any of the input fonts' encodings.

In phase (4), the `Compose` definition can rely on the following two items being defined:

`InputFont`	Dictionary of input font information
`OutputFont`	Dictionary of output font information

Each of these two dictionaries in turn contains the following item(s):

`CharCodes` (Release 4.0N and later) A dictionary giving, for each character name, the code integer. For example, for the digit zero character at code position `0x30` (decimal 48), `CharCodes` contains an entry keyed by the name `/one` with value `48`.

`CharNames` (Release 4.0N and later) A dictionary giving, for each code position (as a name consisting of the decimal representation of the code), the character name. For example, for the digit zero character at code position decimal 48, `CharNames` contains an entry keyed by the name `/48` with value "`/one`".

You can see that CharCodes and CharNames are inverse dictionaries. The size of these dictionaries is the count of the encoded characters in the relevant encoding, typically 128 or 256 for TeX encodings, and anywhere from 215 (ANSI) to 400–1700 entries for Windows Unicode fonts.

`Encoding` (Release 4.0M and prior only) An array giving character names for code numbers. The length of this array gives the encoding size, which is `CHARS_IN_FONT`, typically 256 or 128. (This table now obsoleted by `CharCodes` and `CharNames`.

These tables were upgraded beginning with release 4.0N. Also upgraded were complete search, sort, and hash internals for dictionaries and encoding tables, which make the composition interpreter run much faster.

Furthermore, the input font dictionary also contains:

`TeX_Metrics` Dictionary yielding for each character name an array:

`[ width` TFM width of character (integer)
`height` Likewise height
`depth` Likewise depth
`ic` Likewise italic correction
`llx` Lower-left X of glyph bounding box (integer)
`lly` Likewise Y
`urx` Upper-right X of glyph bounding box
`ury` Likewise Y
`]`

`ItalicAngle` Angle in degrees counter-clockwise of italic slant (float)

32 `COMMDLG.DLL` Required

TrueTeX requires that you have the Windows library `COMMDLG.DLL` installed in your Windows system directory. This has become so common that you almost certainly already have it. If you get a message that Windows cannot locate `COMMDLG` when running a TrueTeX program, you should install it using the original Microsoft Windows Setup disks.

33 Win32s

Win32s is a Microsoft add-on for Windows 3.1 that allows 32-bit Windows programs like the TrueTeX formatter to run on the 16-bit Windows 3.1 system.

You do not use Win32s on Windows 95 or Windows NT, since these systems have a native ability to run 32-bit programs. In that case, select [Cancel] if the TrueTeX Setup program warns you that Win32s is not installed.

Many users experience problems running Microsoft Win32s with Windows 3.1. If you suspect you are having troubles with Win32s, perhaps because the TrueTeX formatter won't run or runs poorly, you should try running the `FreeCell` 32-bit program (a game, actually) installed by Microsoft Win32s Setup. If you chose not to install `FreeCell` during Win32s Setup, you can run that Setup again to do so. `FreeCell` is not a demanding program like the TrueTeX formatter, but it will test the basic integrity of running 32-bit applications via Win32s on your system.

As a last resort when Win32s won't work, TrueTEX contains a DOS-extended version of TEX, `INITEX3X.EXE`, in directory `\truetex\bin`, where TrueTEX Setup also installed the Windows version (`INITEX32.EXE`) and the rest of the TrueTEX Windows executable programs. You can use this DOS version as a replacement for the Windows version of the formatter, and thereby eliminate entirely the need for Win32s to run TrueTEX, although you will be using a command-line TEX under DOS instead of the windowed style. `INITEX3X` uses the same capacities and configuration information as the Windows version, so it should work transparently as a replacement.

Note that if you only have the notorious `GROWSTUB` error trying to run Win32s programs, there is an easy fix described in the TrueTEX *User's Guide*, indexed there under `GROWSTUB`.

You can create a batch file to conveniently run the DOS formatter with preloaded macros. For example, you might make a batch file `LATEX.BAT` somewhere in the `PATH` containing:

```
c:\truetex\bin\initex3x &latex %1 %2 %3 %4 %5
```

and then you can conveniently type "`latex myfile`" to run LATEX on your document `myfile.tex`. For the Windows NT console, which considers ampersands (`&`) as command separators, you should surround the ampersand argument with double quotes:

```
c:\truetex\bin\initex3x "&latex" %1 %2 %3 %4 %5
```

The double quotes will confuse DOS, so this only works where it is needed, namely the Windows NT console.

We recommend that you leave the executable files in their original directory (typically `c:\truetex\bin`) instead of moving them to another directory that might, say, be part of the DOS `PATH`. Since TrueTEX stores the `TRUETEX.INI` configuration file in the same directory as the executable file, if you move the `INITEX3X.EXE` file to another directory, you will have separated the configuration information from the DOS version and the Windows version. That is, you will end up with more than one `TRUETEX.INI` file, which the formatters and previewer each create in their own respective directories. The next section does, however, explain how you can modify where TrueTEX looks for the `TRUETEX.INI` file, should you wish to reorganize the program files.

The 32-bit code support of Windows 95 and Windows NT eliminates these Win32s difficulties.

34 Bugs Corrected

Releases 4.1, 4.1A, and 4.1B of the formatter contained a bug which prevented "deep" TEX runs (that is, runs using a lot of TEX memory) from working (certain TEX variables had been improperly translated from Pascal to C, causing them to overflow). Symptoms included bizarre TEX errors or the formatter application hanging. Release 4.1C corrects the problem.

The `setup.exe` program could possibly be unable to copy items to the target file system, due to an incomplete set of parameters to the file open function of the C run-time library. This bug never manifested with anything but `setup32.exe` and then on only one reported occasion. Release 4.1B corrects the problem.

The addition of code to the formatter for recursive directory searching caused the reporting of file names on the terminal window and log file to be omitted. Release 4.1 corrects the problem.

Due to an undocumented Microsoft change in Windows NT and Windows 95 PostScript printing (see MS Knowledge Base item Q124135), the WIN32 previewer would not consider PostScript printers to be capable of using EPS graphics, and would always send them preview bit-maps (if available) instead. Release 4.1 corrects the problem.

Occasionally when printing from the `DVIGDI32.EXE` previewer's print dialog in Windows 95, the result would be no output at all, and possibly an error message that `StartDoc` returned an error. This was due to a Windows 95 incompatibility described in the Microsoft Knowledge Base item Q135119. The problem does not occur in any other version of Windows or with the 16-bit previewer (`DVIGDI16.EXE`). Release 4.0S corrects the problem.

TrueTEX would occasionally cancel an in-progress metric export. Release 4.0N corrects the problem.

Release 4.0L corrects an error in liquidation calls that resulted in TrueTEX passing a `NULL` value for `hWnd` to `\special` handlers. The handle value is now that of the TrueTEX MDI frame window, as stated in the `special()` documentation. Thus the parameter `hWnd` should always give a valid TrueTEX MDI-frame-window or document-window handle, suitable for message-passing.

Release 4.0L corrected a problem which caused an application error when a font substitution file in the preference item could not be found.

Release 4.0J corrects a problem where the scroll buttons in the 32-bit previewer would not position properly.

Due to handle leakage, recursive file opens would no longer work after about 65 recursively located files were opened. Release 4.0K corrects the problem.

35 TrueTeX Release Summary

The following tables summarize the TrueTeX releases and their purposes, beginning with the most recent.

TrueTeX 4.5 Release History

TrueTeX Release and Date		Enhancements Embodied and Bugs Fixed
4.5U	Oct 13 2007	Corrects PDF rendering of OpenType system fonts in Windows Vista. Integrates TeX Live 2007. Lambda format (`lambda.fmt`) for Omega is built with Babel multilingual extensions.
4.5T	May 15 2007	Corrects Times ligature rendering under Windows Vista.
4.5S	May 04 2006	The UM fonts now permit installable embedding, the least restrictive level. Corrected: "`hex:`" and "`utf-8:`" no longer truncate certain characters from the converted names.
4.5R	Jun 16 2005	The optional AMS font packages are now pre-installed, allowing use of `\usepackage{amssymb}`, `\usepackage{amsfonts}`, etc. Macros: AMS-LaTeX 2.2 updated to 16 Jun 2005. This adds support for the `amsmidx` style (multiple indexes), and various sample templates and PDF documents.
4.5P	May 11 2004	Postscript (Type 1) fonts now enumerated into font dictionary and given TeX external names based on registry filename values (Windows XP only).
4.5N	Oct 1 2003	Text menu item "Table of Fonts Installed" replaced by "Font Substitution Rules". Prefixes "`hex:`" and "`utf-8:`" added to `.ini` syntax. Added to distribution: `fontenum.exe` command-line tool to list fonts which Windows enumerates. Also, `russian.ini`, which gives an example of mapping TeX external names to non-ASCII Windows font full names. Corrected: Construction of font dictionary: The font dictionary now selects the *first* font Windows enumerates for a given full name, when Windows reports multiple national-language scripts as being ANSI-encoded (even though not ANSI). This should select the true ANSI-encoded font. Corrected: Program termination when reconstituted fonts enlarged font dictionary across a multiple of 256 entries. Only occured when font reconstitution was turned on in Preferences+Expert+Font Reconstitution.

TrueTeX 4.5 Release History (continued)

TrueTeX Release and Date	Enhancements Embodied and Bugs Fixed
4.5M 18 July 2003	Expands limits to number of lines in console windows. Corrects ariali and arialbi fonts in WinFonts.map, which had been swapped in PDF output.
4.5K 02 July 2003	Adds support for the Euro currency symbol via the LaTeX `eurosym.sty` package, including 6 new TrueType fonts named `fey[bm][lor]10`. This style, and Times math, are also now supported by `pdflatex`. Adds file `WinFonts.map` to `pdftex/config`, linking Windows system fonts for `times.sty` PDF output.
4.5J 09 May 2003	The encodings of the following cmap 3.0 (symbol) fonts are altered slightly to ensure compatibility across all Windows varieties: `cmex{7,8,9}`, `cmmi{5,6,7,8,9,10,12}`, `cmmib{5,6,7,8,9}`, `cmmr10`, `ms{a,b}m{5,6,7,8,9}`. Certain characters in these fonts would not render, or would be mismapped, in the previewer or in PDF output, especially in Windows XP. Fonts `lcircle10` and `lcirclew10` are now correctly linked to `pdftex`, which previously could not find them due to a lingering effect of the ancient DOS 8+3 naming problem affecting the files associated with these fonts (being the two fonts in standard TeX having names longer than 8 characters). TrueType versions of the AMS fonts scaled sizes `ms{a,b}m{5,6,7,8,9}` of fonts `ms{a,b}m10` are provided in diretory `TrueTeX/fonts/truetype`. This permits their use with `pdftex`. However, file `amssubs.ini` still maps these fonts to scaled sizes of `ms{a,b}m10` in the TrueTeX previewer. This avoids the TrueTeX installer having to install the 10 new fonts as Windows fonts, which would add to the system footprint. A PDF LaTeX multilingual format file is now provided, `pdflatex_ml.fmt`. Console templates for multilingual LaTeX with PDF output are also now supported. The templates are named `LaTeX-PDG`, `LaTeX-LM-PDG`, and `LaTeX-LBLM-PDG`, analogous to `LaTeX-PDF`, `LaTeX-LM-PDF`, and `LaTeX-LBLM-PDF` described below. Standard LaTeX packages for color (`\usepackage{color}`) and graphics (`\usepackage{graphics}`) now self-configure for TrueTeX previewer `\special`'s or PDF output `\special`'s as appropriate, depending on whether `pdflatex` is running or not. You can override this automatic default with options `[truetex]` or `[pdftex]`. Distribution file `special.zip` now contains the source file `spepsf.c`. The DC fonts are upgraded to support multilingual LaTeX in both `pdflatex` (`&pdflatex_ml` format) and the previewer (`&latex_ml` format).

TrueTeX 4.5 Release History (continued)

TrueTeX Release and Date		Enhancements Embodied and Bugs Fixed
4.5H	08 Feb 2003	Added `pdftex` and `pdflatex` support, which generate PDF file output directly from TeX and LaTeX documents using TrueType CM fonts; New standard templates for previewer `-x` launch include: `LaTeX-PDF`, `LaTeX-LM-PDF`, and `LaTeX-LBLM-PDF`, which are each analogous to the similarly named 1-, 2-, and 3-pass LaTeX templates described below. Added TrueType fonts `cmmib{5,6,7,8,9}` and `cmex{7,8,9}` for compatibility with `amstex`; file `amssubs.ini` updated accordingly. Added LaTeX `slides` fonts `lcmss8`, `lcmssb8`, and `lcmssi8` to setup manifest `fontreg.set`, which had been inadvertently omitted. Altered exported font metrics to maintain compatibility with the restricted parsing in the reimplemented omegafonts 2.0 tools, e.g., `ovp2ofm` (all fixword values are reals, all hex values are uppercase); reversion to omegafonts 1.8 tools are still needed since the grammatically acceptable files still crash `ovp2ofm` 2.0.
4.5G	18 Jan 2002	Like 4.5D and 4.5E except the full CD is provided.
4.5E	18 Jan 2002	Like 4.5D except formats provided for both `INITEX32.EXE` and web2c formatters. This distribution is the format files only. See 4.5G for the full CD.
4.5D	18 Jan 2002	LaTeX: preloaded format `latex_ml.fmt` is updated to have US, UK, French, and German hyphenation preloaded. (For web2c formatter only.) This distribution is the format file only. See 4.5G for the full CD.
4.5C	30 Oct 2001	Previewer: Accelerators `[Ctrl]+/` and `[Ctrl]+*` now work on the home area of the keyboard; formerly these worked only on the numeric keypad. LaTeX: PSNFSS pre-installed; the distribution files were present in 4.5B but not installed.
4.5B	24 Oct 2001	Incorporates LaTeX 01 June 2001 official release. Utility `makeindx.exe` is replaced by later version `makeindex.exe`. Utility `ovp2ofm.exe` is added to support Omega font metric export.
4.5A	15 Sep 2001	The TrueType cmap 3.0 (symbol) encoding of the `cmmi` fonts is corrected from `0x20` to `0xf020` to maintain compatibility with Windows 2000 and XP. Incorporates LaTeX 01 June 2001 pre-release. Spurious `times.ovf` (etc.) metrics removed from `timesopt.dvi` documentation directory. Decommissioned fax number removed from About dialog and `readme.tex`.
4.5	13 Sep 2001	Includes LaTeX update (June 2001, still a pre-release consisting of `base`, `graphics`, `tools`, and `cyrillic`); previous LaTeX tree in distribution CD directory `Obsolete`. Uninstall: read-only files (from BibTeX distribution) changed to read-write. Updates TeX Live 6 executables. Newly includes: Omega support files and release document; RevTeX4 BibTeX (`bst`) files. Previewer: no change from 4.4E other than marking as release 4.5. Changes to the `cmmi` fonts (noted in release 4.4 below) are not actually included until this release.

TrueTeX 4.4 Release History

TrueTeX Release and Date		Enhancements Embodied and Bugs Fixed
4.4E	30 Aug 2001	Incorporates all changes to date.
4.4D	29 Aug 2001	Previewer: font encoding maps now selectable for 8-bit (TextOutA) or 16-bit (TextOutW) rendering. Any non-Ansi and non-Symbol fonts omitted from font dictionary, correcting an encoding problem seen in Windows 2000 when, for example, viewing `timesopt.dvi` on a system where Eastern Europe support was also installed. Note: internal release only.
4.4C	19 Jul 2001	Previewer: message displays from special handlers are more guarded. Handler `specho` truncation increased to 1024 characters. The font dictionary is now dynamic; previously it was limited to 1024 fonts. Accelerator `[Ctrl]+0` corrected to select 1:1 zoom (was incorrectly fit-in-window).
4.4B	13 May 2001	TDS tree: Version 2.13 of amsmath, etc., after the amsltx2.zip distribution from ftp://ftp.ams.org current as of 5/11/2001. Added Babel multilingual hyphenation patterns. Previewer: `TRUETEX.INI` has corrected templates for `web2c` TeX console executables. Fixed process templates to allow them to be called without arguments. Lambda added to process templates. Fixed `-i` argument to not default back to module directory default. Added `latex_ml.fmt` format (preloads T1 encoding via DC fonts, and English/French/German hyphenation via Babel) for `web2c` TeX console executables. Clarification: multiple files may appear on the previewer command line; options apply only to subsequently named files.
4.4A	8 May 2001	Previewer: Clipboard handling updated to WIN32 GDI model (that is, the previewer places EMF metafiles instead of WMF metafiles on the clipboard when doing `Edit+Copy`).
4.4	3 May 2001	Previewer: New `SPECHO32.DLL` `\special` handler to echo `\special` texts for debugging. When in the list of handlers, this causes the previewer to display `\special` texts in a message window whenever a page is rendered. Item `LATEX_HELP` in section `[DVI-GDI]` in `TRUETEX.INI` points to a Windows help file (`.hlp`) which is listed as "`LaTeX Reference`" on the `Help` menu. Fonts `cmmi{5,6,7,8,9,10,12}.ttf` are updated for the "new delta" style (this change was inadvertently omitted from the distribution until release 4.5 above). Distribution includes various console templates to invoke one or more passes of `web2c` LaTeX, optionally with BibTeX and/or `MakeIndex` passes. Standard templates for use with `-x` include: `LaTeX-L`, `LaTeX-LP`, `LaTeX-LBLP`, `LaTeX-LMLP`, `LaTeX-LBLMLP`, `View`, `BibTeX`, `MakeIndex`, and `Lambda`. Multipass invocations are congolomerated by batch files in the executables directory. Substitute `T` for `L`, `TP` for `LP`, `TBTP` for `LBLP`, etc., in the above template names to invoke the multilingual LaTeX `latex_ml.fmt` format instead of standard `latex.fmt`. TeX Live executables for `web2c` implementations of LaTeX, BibTeX, and `MakeIndex` are incorporated into the distribution.

TrueTeX 4.3 Release History

TrueTeX Release and Date		Enhancements Embodied and Bugs Fixed
4.3E	21 April 2001	Previewer: Window menu had lost child selection ability.
4.3D	13 April 2001	Previewer: Improvements to `-x` facility: Environment value `PREVIEW` causes automatic preview window after `EXE` process finishes (but only when a full-path `-x` argument is present). Absent `EXE` value omits the process, and goes directly to `PREVIEW` (if any). Should be used only for WIN32 console executables, not for 16-bit `.EXE`'s. Environment value `ARRANGE` may be `maximize`, `minimize`, `cascade`, `tilehorizontal`, or `tilevertical` to control the arrangement of MDI children arising from `-x` processes.
4.3C	09 April 2001	Previewer: Improvements to `-x` facility. CWD for `-x` process is implied from the single argument, which must specify an absolute path, as must `EXE`; this is consistent with the formatter behavior.
4.3B	21 December 2000	Previewer: When font reconstitution is off, the previewer renders characters with 8-bit TextOutA() calls versus 16-bit TextOutW() call, depending on whether the 16-bit character codes have an upper byte of zero; this works around a strange Windows GDI bug that magically re-maps code 0xb7 to 0x2219 in TextOutW(), which goes back to a Microsoft kludge for Microsoft Word under Windows 3.1.
4.3A	05 June 2000	Previewer: Reconstituted fonts now have a unique id in their name tables, consisting of the product name ("`TrueTeX`") prefixed to the full name, for compliance with an undocumented requirement of Windows 2000. Miscellaneous: Added `bin2text`, `text2bin`, and `*.tab` files to the distribution to support the `ttf_edit` utility as described in `ttf_edit.htm`.
4.3	22 Mar 2000	Previewer: `File+Export Metrics` extended to allow user to choose the TeX external name, to export Omega `.ovp`, `.ovf`, and `.ofm` metric files, to convert Omega metrics with `ovp2ofm`, and to allow the user to optionally bypass the re-encoding in favor of 1:1 encoding of all characters in the font. Added Web link in Help menu. `Window+Tile` menu item expanded to `Window+Tile Horizontally` and `Window+Tile Vertically`.

TrueTeX 4.2 Release History

TrueTeX Release and Date		Enhancements Embodied and Bugs Fixed
4.2F	2 Mar 00	Previewer: Signals `-x` completion to other processes via a named event. Handles `[Ctrl+Break]` interrupts in the format tab of console windows. Does not lock up closing a console window when the console executable is missing or otherwise fails to start. Console window titles now show the `-x` arg, not the expanded command line.
4.2E	29 Feb 00	Previewer: The restored size of an instance launched with command-line option ”-d -1 -1 -1 -1” (maximized size) is now the Windows default size rather than a zero size. New “`-x` *console-process*” option to launch console processes. New “`-1`” option to force single-instance behavior. New “`-z`” option to close the application when the option is encountered; used after `-w` print requests or to shut down a single-instance mentor; may appear anywhere on the command line, but only takes effect after command line has been processed. Logic for printing from previewer command-lines has changed a bit: The -w (print from command-line) option is modified to apply only to .dvi files appearing *subsequently* on the command line (which presumably was the only way it was used);
4.2D	24 Feb 00	Previewer: Added console child windows for external console processes via ”-x ¡command line¿” option; these will evolve into document-centric tabbed project windows, the separate formatter will become obsolete, and the previewer will become the single TrueTeX application window. View menu item “Show DVI Info” changed to ”Toggle DVI Info”. Accelerator keys changed to avoid conflicts with typing text into console windows (printing characters changed to Ctrl+key, etc.). “About Author” photo updated to 12/1999.
4.2C	03 Feb 00	Macros: AMS-LaTeX 2.0 updated to 2 Feb 2000. Includes LaTeX version 1999/12/01.
4.2B	14 Jan 00	Setup: Windows 2000 run-time detection. Previewer: Windows 2000 run-time detection. Font reconstitution toggle expert preference works correctly. Macros: Unpacked Babel.
4.2A	16 Dec 99	Formatter: Moved old WIN16 previewer and DLL’s to “obsolete” sub-directory. Macros: includes AMS-LaTeX 2.0. LaTeX pre-loaded format and font definitions configured for “old DC” fonts.

TrueTeX 4.2 Release History–Continued

TrueTeX Release and Date	Enhancements Embodied and Bugs Fixed
4.2 17 Nov 99	Previewer: Extensive reworking and simplification of how TeX external font names are mapped to Windows fonts. Windows fonts are now universally identified by their full names, not their face names and/or styles. The maximum number of characters per font which can be exported as metrics is increased from 2048 to 32768 (certain Asian fonts exceed the old limit). Page selection in the dialog now works correctly when first printing a page range and then printing all pages. Font loading in the previewer: The previewer now accepts the Omega 1.5 `.ofm` (font metric level 0) files. It also recognizes, but does not accept, level 1 `.ofm` files. The older `.xfm` format is still supported, and TrueTeX tools and fonts are still supplied using that format. When loading font metrics, the previewer will first search for metric files ending in `.ofm`, then `.xfm` and finally `.tfm`. Likewise for virtual fonts, the previewer will search in the order `.ovf`, then `.xvf`, and finally `.vf`. A bug preventing virtual font packets beyond the first 256 from working was fixed. There have been no such virtual fonts published until now, so this problem has not been apparent. Belleek fonts: virtual font blmi (`blmi.vf` and `blmi.xvf`) is remapped to use higher Unicode codes instead of 0x0-0x20 and 0x7f. TrueType fonts `blex.ttf`, `blsy.ttf`, and `rblmi.ttf` are updated to reflect later standards and marked internally as version 1.1 of the fonts. Macros: includes LaTeX 1999/06/01

TrueTeX 4.1 Release History

TrueTeX Release and Date		Enhancements Embodied and Bugs Fixed
4.1L	19 Feb 99	Reorganized the distribution medium for a single volume instead of a collection of floppies concatenated on a single volume. Added 9 Universal Modern fonts. Converted the Times option to be a regular part of the installation. Freshened LaTeX distribution. Added all optional LaTeX packages from CTAN to distribution (not all packages are unpacked, however, although all of AMS-LaTeX, `graphics`, and the `tools` set are). Removed the old 16-bit Previewer from the distribution. Added the SliTeX fonts `lcmss8`, `lcmssb8`, and `lcmssi8`. `TRUETEX.INI` now includes the pattern `lcm*` as a TeX font name. Previewer: Added "Unknown encoding (8-bit)" to VPL export encodings, which provides a set of hexadecimal names for each character.
4.1K	06 Nov 98	Previewer: Improved missing font handling default preference to display a warning message, instead of showing blue characters; Windows forces certain font substitutions (such as the PostScript 35 names) that are not necessarily in error.
4.1J		(Releases I and J were skipped.)
4.1H	29 Oct 97	Previewer: Fixed possibility of crash at start-up when font substitution strings taken from the Windows registry exceeded a total length of 8192 (about 400 or so typical font names).
4.1G	24 Sep 97	Latest versions of `plain.tex`, `testfont.tex`, `epsf.{sty,tex}`. WIN32 `makeindx.exe` from web2c version. Formatter: increased TeX capacities to "generous" AMS standards (*Note*: old `.fmt` formats will be incompatible and must be regenerated); increased maximum length for any one environment string to 1300 characters (5*`MAX_PATH`); fixed recursive file opening to not be inhibited by paths starting with WIN32 UNC's (same problem in previewer also fixed); fixed Other+Show-environment menu item to not crash for long environment strings. Previewer: corrected file opens to maximize windows if the active child is maximized, or if no active child exists; enlarged font-substitution capacity from 64 KB to 1024 KB; scroll buttons enhanced to scale to zoomed page size.

TrueTeX 4.1 Release History–Continued

TrueTeX Release and Date		Enhancements Embodied and Bugs Fixed
4.1E	25 Aug 97	Corrected color-rendering in 24-bit BMP `\special`'s. Corrected previewer's slight possibility of crash when an unknown font appears in a document. Enhanced previewer's command-line parsing to understand double-quoted arguments. Converted `setup.exe` to use INFO-ZIP's `UNZIP32.DLL` instead of PKWARE's self-extracting ZIP archives; added 'u' action for setup manifests; added `-q` option for un-install quietness.
4.1D	17 Jul 97	Reconfigured installation disks to install all TrueTeX files in fully TDS-compliant directory tree. Added `null.tex` to the plain macros. Corrected `setup.exe` to treat blank lines in manifest as comments. Formatter no longer uses environmental paths to search for file names starting with the current directory ("`./`" or "`.\`"). Previewer unchanged from 4.1C. Installation program `setup.exe` upgraded, including an uninstall icon in the program group; Windows 3.1 installation no longer supported.
4.1C	16 May 97	Corrected formatter problem which caused TeX errors, or which caused the application to hang, on "deep" TeX runs. Correct previewer problem where long, unknown command-line options could overflow an 80-character buffer. Converted `SPTIFF32.DLL` and `SPEPSF32.DLL` to use `LIBTIFF.DLL` for TIFF graphics, and added `LIBTIFF.DLL` to the distribution.
4.1B	7 Apr 97	Corrected default rule interior color to be black instead of white. Corrected `setup32.exe` to send mode parameter to C run-time file open function.
4.1A	27 Jan 97	Enhanced color `\special` to affect rules as well as text.
4.1	22 Nov 96	Wide release; floppy distribution grows from three to four 1.44 MB disks. Executables recompiled with Microsoft Visual C++ 4.2. SimSoft Image Library hooked in `SPTIFF32.DLL` (but not provided in distribution). Corrected: EPS special handler now recognizes PostScript printers under WIN32; formatter no longer omits opened file names from terminal and log file output.

TrueTeX 4.0 Release History

TrueTeX Release and Date	Enhancements Embodied and Bugs Fixed
4.0T 24 Oct 96	Formatter: Any Windows command-line argument starting with ‘`&`’ will be taken as a format file specification, and overrides any `.INI` file history, as is the case in the DOS-extended version (use forward slashes for file path separators). Added debugging messages for command-line options when verbose option (`-v`) is active (you can specify `-v` as the first option to see the effects of later options). Verbose option `-v` now toggles if you give it more than once on the command line (allowing you to turn it on while the formatter parses other options, and then turn it off before the previewer starts formatting). Navigating in the `Format` dialog no longer changes the current directory of any later `File+Open` dialog. Previewer: PSNFSS Times-text support; Times-styled math fonts. To mark an optional font substitution item (no message issued if not found), begin it with ’?’. CRC’s for version check of preference table are now (finally) equal between WIN16 and WIN32 previewers, eliminating false-alarms when switching between editions. Corrected unregistered TeX fonts not getting TeX remapping. Corrected kerns in exported TTF metrics being twice their correct values. TIFF `\special` handler warns if subsidiary DLL’s cannot be loaded (for example, `sptiff16.dll`). Setup: WIN16 vs. WIN32 discrimination. The utility `ttf_edit`.
4.0S 18 Oct 96	Non-starting Win95 printing corrected.
4.0R 9 Oct 96	Recursive virtual fonts.
4.0P	Unicode virtual fonts; sparse font optimization.
4.0N	Omega extensions and AFM metrics for virtual fonts; general `.cod` encoding selection; 8-bit subset fonts from Unicode fonts; extended support for ATM fonts. Corrected spurious cancelling of metric exports.
4.0M	Windows registry for initialization; message windows.
4.0L	Memory-mapped `\special` texts; `\special`-handler messages. Corrected liquidation calls to `\special`-handlers. Corrected crash from missing `.INI` files.
4.0K	Font encoding map displays. Corrected handle leakage limiting recursive file opens to 65.
4.0J	Previewer: window sizing command-line options; long file names; DDE server. Formatter: stdio window class name to ProgramName, added support for WIN32 long file names. Corrected scroll button positioning in WIN32 previewer.
4.0H	TDS-standard recursive directory searching.
4.0D	Corrected `.INI` file handle leakage introduced in 4.0B.
4.0C	Corrected preload dialog in the formatter.
4.0B	Multiple `.INI` files; initial working directory; quiring; virtual fonts; TeX font names; `\special`-handler sources; `TEXWAKEUP`; manifest-driven `setup`; network compatibility; virtual font composition engine; DC fonts.

36 Regarding Copyrights and Other Legal Matters

36.1 LaTeX 2ε Copyright Notice

The principal authors of the LaTeX 2ε project have made their works freely available, but have nevertheless asserted a copyright on LaTeX as a means of controlling distribution and avoiding incompatibilities that troubled the old LaTeX 2.09. Accordingly, we reproduce the following notice, which applies only to the LaTeX macros, and not to TrueTeX itself:

TrueTeX provides the genuine LaTeX, direct from the authors' distribution, free of any modifications or incompatibilities. The LaTeX authors express a valid concern that their work not be published in a modified form, unless the modified files are given new names. This prevents the proliferation of non-standard versions that might be confused with the genuine LaTeX autograph. Should you wish to modify any of the LaTeX distribution files, you must take care to observe the authors' restrictions on doing so.

TrueTeX does not provide every file for every package and tool available with LaTeX, which now totals over 100 megabytes. If you wish to complete the partial distribution provided with TrueTeX, you can copy it from the CTAN FTP archive described above, or if you are unable to do so, request a copy of the current full LaTeX distribution from us.

36.2 TIFF Library Copyright Notice

The following notice applies only to the `libtiff` code, which constitutes a portion of `LIBTIFF.DLL`, and not to TrueTeX itself:

36.3 Info-Zip Disclaimer

> The TrueTeX Setup program may incorporate UnZip code to uncompress zip archives from the distribution media. Info-ZIP's software (Zip, UnZip and related utilities) is free and can be obtained as source code or executables from various bulletin board services and Internet/WWW sites, including `ftp://ftp.cdrom.com/pub/infozip`.

37 Technical Support

You can reach TrueTeX technical support from 10 a.m. to 4:30 p.m. Eastern time:

Richard J. Kinch, Ph.D.
6994 Pebble Beach Ct
Lake Worth FL 33467
Tel (561) 966-8400
`kinch@truetex.com`
`http://www.truetex.com`

This document was formatted on October 15, 2007.