The dvipdfmx User's Manual

The dvipdfmx project team*

May 7, 2017

1 Introduction

The dvipdfmx (formerly dvipdfm-cjk) project provides an extended version of the dvipdfm, a DVI to PDF translator developed by Mark A. Wicks.

The primary goal of this project is to support multi-byte character encodings and large character sets such as for East Asian languages. This project started as a combined work of the dvipdfm-jpn project by Shunsaku Hirata and its modified one, dvipdfm-kor, by Jin-Hwan Cho.

Extensions to dvipdfm include,

- Support for OpenType and TrueType font, including partial support for OpenType Layout for finding glyphs and vertical writing.
- Support for CJK-LaTeX and HLaTeX with Subfont Definition Files.
- Support for various legacy multi-byte encodings via PostScript CMap Resources.
- Unicode related features: Unicode as an input encoding and auto-creation of ToUnicode CMaps.
- Support for pTeX (a Japanese localized variant of TeX) including vertical writing extension.
- Some extended DVI specials.
- Reduction of output files size with on-the-fly Type1 to CFF (Type1C) conversion and PDF object stream.
- Advanced raster image support including alpha channels, embedded ICC profiles, 16-bit bit-depth colors, and so on.
- Basic PDF password security support for PDF output.

Some important features are still missing:

- Linearization.
- Color Management.

*Jin-Hwan Cho, Matthias Franz, and Shunsaku Hirata

- Resampling of images.
- Selection of compression filters.
- Variable font and OpenType 1.8.
- and many more...

dvipdfmx is now maintained as part of TEX Live. Latest source code can be found at TEX Live SVN repository. For an instruction on accessing the development sources for TEX Live, see,

http://www.tug.org/texlive/svn/

This document, "The dvipdfmx User's Manual", was originally prepared for TEX Live 2017. Current maintainer of this document is Shunsaku Hirata. Latest version and contact information can be found at:

http://github.com/shirat74/dvipdfm-x-doc

Please send quenstions or suggestions.

1.1 xdvipdfmx

xdvipdfmx is an extended version of dvipdfmx, and is now incorporated into dvipdfmx.

The xdvipdfmx extensions provides support for the Extended DVI (.xdv) format generated by XETEXwhich includes support for platform-native fonts and the XETEX graphics primitives, as well as Unicode text and OpenType font.

XETEX originally used a Mac-specific program called xdv2pdf as a backend program instead of xdvipdfmx. The xdv2pdf program supported a couple of special effects that are not yet available through xdvipdfmx: The Quartz graphics-based shadow support, AAT "variation" fonts such as Skia, transparency as an attribute of font, and so on. It would be nice if they continue to be supported. Suggestions and help are welcomed.

1.2 Legal Notice

2 Installation and Usage

Typical usage and installation steps are not different from the original dvipdfm. Please refer documents from dvipdfm distribution for detailed instruction on how to install and how to use dvipdfm. The dvipdfm manual is available from its CTAN site:

http://www.ctan.org/tex-archive/dviware/dvipdfm

Option	Description
`-C` *number*	Specify miscellaneous option flags. See, section of "Incompatible Changes" for details.
`-S`	Enable PDF encryption.
`-K` *number*	Set encryption key length. The default value is 40.
`-P` *number*	Set permission flags for PDF encryption. The *number* is a 32-bit unsigned integer representing permission flags. See, section of "Encryption Support". The default value is `0x003C`.
`-I` *number*	Life of image cache in hours. By specifying value `0` `dvipdfmx` erases cached images, and value `-1` erases all cached images and does not leave newly generated one. The default value is `-2`. (ignore image cache)
`-M`	Process METAPOST generated PostScript file.
`-E`	Always try to embed fonts *regardless of liscensing*.

Table 1: Additional command line options recognized by `dvipdfmx`.

The minimal requirements for building `dvipdfmx` is the `kpathsea` library. `zlib` for compression and `libpng` for PNG inclusion are highly recommended. Optionally, the libpaper library may be used to handle paper sizes.

This document only describes additions and modifications to `dvipdfm`. Please refer the "Dvipdfm User's Manual" available from the CTAN site mentioned above for basic usage.

Some additional command line options recognized by `dvipdfmx` are listed in Table 1. Try

```
dvipdfmx --help
```

for the list of command line options and thier explanations.

3 Quick Guide

`dvipdfmx` is supposed to be used by users of LaTeX packages for typesetting CJK languages like HLaTeX and CJK-LaTeX, and TeX variants such as XeTeX, pTeX, and upTeX. This section is intended to be a quick guide for each users.

3.1 XeTeX

For XeTeX users, most part of this document is irrelevant except section of "Graphics and Image Formats" and "DVI Specials".

3.2 pTeX

pTeX users are at least required to install several auxilially files mentioned in the section of "Auxilially Files" and to setup fontmappings. Just install the *adobemap-*

pings and *glyphlist* for auxilially files. (As TEX Live basic intallation requires them, they are probably already installed for TEX Live users.)

Setting up fontmaps can be done easily with help of the *ptex-fontmaps* package. For examples, to use with IPA fonts (contained in the *ipaex* package), run,

```
updmap ipaex
```

Alternatively, just for a quick test of installation, try the following:

```
\documentclass{article}
\begin{document}
\special{pdf:mapline rml H KozMinProVI-Regular}
...Some Japanese text goes here...
\end{document}
```

In this example, PDF viewer which can handle substitute font is required since `dvipdfmx` does not embed fonts.

For using Japanese text in PDF document information and annotations, put the following `special` command,

```
\AtBeginDocument{\special{pdf:tounicode 90ms-RKSJ-UCS2}}
```

in the preamble. The above `special` command instructs `dvipdfmx` to convert text encoded in Shift-JIS to Unicode. For EUC-JP, replace 90ms-RKJK-UCS2 with EUC-UCS2.

3.3 upTEX

upTEX users are basically the same as pTEX users but there are two choices for setting fontmaps. Setup fontmaps as mentioned in the above for pTEX, or use keyword `unicode` in the encoding field of fontmap files.

The former case might be easier as auto-creation of fontmap files can be done with the `updmap` program and the *ptex-fontmaps* package. But in this method there are some difficulties when using fonts which employ character collection (glyph repertoire) other than Adobe-Japan1 in the case of PostScript flvavored OpenType fonts. In the later case, the *adobemappings* package is not reuiqred and newer PostScript flavored OpenType fonts which do not employ Adobe-Japan1 can be used too.

Using `unicode` is more simpler and intuitive thus it is recommended to use this method.[1] Typical example fontmap entries for using Adobe's SouceHan fonts look like:

```
urml    unicode  SourceHanSerifJP-Light.otf
```

[1] For TEX Live 2017. Earlier versions have buggy support.

```
urmlv   unicode  SourceHanSerifJP-Light.otf  -w 1
ugbm    unicode  SourceHanSansJP-Medium.otf
ugbmv   unicode  SourceHanSansJP-Medium.otf  -w 1
```

As in pTeX, the following `special` instruction is necessary to PDF document information and annotations to be shown correctly:

```
\AtBeginDocument{\special{pdf:tounicode UTF8-UCS2}}
```

Here, input encodig is assumed to be UTF-8.

3.4 CJK-LaTeX

CJK-LaTeX users are required to have Subfont Definition Files to be installed. They are available as part of the *ttfutils* package.

To use TrueType Ariphic fonts provided by the *arphic-ttf* package:

```
\documentclass{article}
\usepackage{CJKutf8}
...Other packages loaded here...
\AtBeginDocument{%
  \special{pdf:tounicode UTF8-USC2}%
  \special{pdf:mapline bsmiu@Unicode@ unicode bsmi00lp.ttf}%
 }
\begin{document}
\begin{CJK}{UTF8}{bsmi}
...some Chinese text goes here...
\end{CJK}
\end{document}
```

Here, `pdf:mapline` special is used to setup fontmappings.

4 Auxiliary Files

This section is mostly for supporting legacy encodings and legacy font format such as PostScript Type1 font. XeTeX users may skip this section.

`dvipdfmx` has a capability to handle various input encodings from 7-bit encodings to variable-width multi-byte encodings. It also has some sort of Unicode support. Several auxiliary files which are not common to TeX users are needed to enable those features. This section shortly describes about them.

4.1 PostScript CMap Resources

PostScript CMap Resources[2] are required for supporting lagacy encodings such as Shift-JIS, EUC-JP, Big5, and other East Asian encodings. `dvipdfmx` internally identifies glyphs with identifiers (CIDs[3]) represented as an integer ranging from 0 to 65535 in the CID-based glyph access. PostScript CMap Resources describes the mapping between sequences of input character codes and CIDs. `dvipdfmx` has an extensible support for multi-byte encodings via PostScript CMap Resources.

CMap files for standard East Asian encodings, for use with Adobe's character collections, are included in the *adobemapping* package. The latest version of those CMap files maintained by Adobe can be found at Adobe's GitHub Project page:

`http://github.com/adobe-type-tools/cmap-resources`

Those files are mandatory for supporting pTEX. upTEX users may also want to install them but they are not required.

4.2 Subfont Definition Files

CJK fonts usually contain several thousands of glyphs. For using such fonts with (original) TEX, which can only handle 8-bit encodings, it is necessary to split fonts into several "subfonts". Subfont Definition File (SFD) specify the way how those fonts are split into subfonts. `dvipdfmx` uses SFD files to convert subfonts back to a single font.

SFD files are not required for use with TEX variants which can handle multi-byte character encodings and large character sets such as pTEX, upTEX, XETEX, and Omega. HLATEX and CJK-LATEX users are required to have those files to be installed. SFD files are available as a part of the *ttfutils* package for TEX Live users.

4.3 The Adobe Glyph List and ToUnicode Mappings

The Adobe Glyph List[4] (AGL) describes correspondence between PostScript glyph names (e.g., `AE`, `Aacute`,...) and Unicode character sequences representing them. Some features described in the section "Unicode Support" requires AGL file.

`dvipdfmx` looks for the file `glyphlist.txt` when conversion from PostScript glyph names to Unicode sequences is necessary. This conversion is done in various situations; when creating ToUnicode CMaps for 8-bit encoding fonts, finding glyph descriptions from TrueType and OpenType fonts when the font itself does not provide a mapping from PostScript glyph names to glyph indices (version 2.0 "post" table), and when the encoding `unicode` is specified for Type1 font.

AGL file is included in the *glyphlist* package. The latest version can be found at Adobe's GitHub site:

`http://github.com/adobe-type-tools/agl-aglfn`

ToUnicode Mappings are similar to AGL but they describe correspondence between CID numbers (instead of glyph names) and Unicode values. The content of those files are the same as CMap Resources. They are required when using

[2]See, "Adobe CMap and CIDFont Files Specification"
[3]PostScript terminology "Character IDentifier".
[4]See, "Adobe Glyph List Specification"

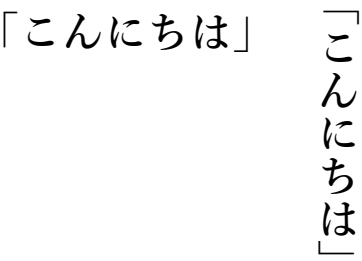

Figure 1: An example of horizontal and vertical text; left and right corner brackets are replaced with their vertical counterparts.

TrueType fonts emulated as CID-keyed fonts. They should be found in the same directory as ordinary CMap files.

ToUnicode Mapping files are included in the *adobemapping* package. Those files are not required for XeTeX users.

5 Overview of Extensions

This section gives a quick overview of `dvipdfmx`'s extended capabilities.

5.1 CJK Support

There are many extensions made for supporting CJK languages. Features described here is mainly for CJK languages but their use is actually not limited to it. Those features are implemented in a generic way so that it can be beneficial to users who are not involved in CJK languages.

5.1.1 Legacy Multi-byte Encodings

`dvipdfmx` has an extensible support for encodings by means of PostScript CMap Resources. Just like `enc` files are written for 8-bit encodings, one can write their own CMap files to support custom encodings. See, Adobe's technical notes for details on PostScript CMap Resources.

5.1.2 Vertical Writing

`dvipdfmx` supports vertical writing extension used by pTeX and upTeX. A DVI instruction to set writing mode is supported. OpenType Layout GSUB Feature is supported for selecting vertical version of glyphs.

5.2 Unicode Support

Unicode support here consists of two parts: Supporting Unicode as input encodings and making output PDF files "Unicode aware".

5.2.1 Unicode as Input Encoding

dvipdfmx recognizes an additional keyword unicode in fontmap files to declare that Unicode values are used in input DVI files. Unicode support is basically limited to the Basic Multilingual Plane (BMP) since there are no support for code ranges that requires more than three bytes in TFM and extended TFM formats.

5.2.2 ToUnicode CMap Support

In PDF,it is often the case that text is not encoded in Unicode. However, modern applications usually want them represented in Unicode to make it usable. ToUnicode CMap is a bridge between PDF text string encodings and Unicode encodings, and they makes it possible to extract text in PDF as Unicode encoded strings. It is important to make resulting PDF search-able and copy-and-past-able. Dvipdfmx supports the auto-creation of ToUnicode CMaps.

It will not work properly for multiply encoded glyphs due to fundamental limitations of Unicode conversion mechanism with ToUnicode CMaps.

5.3 Other Extensions

dvipdfmx can generate encrypted PDF documents to protect its contents from unauthorized access. It is limited to passorwd-based authentication, and public-key based authentication is not supported. The 256-bit AES encryption is also supported for PDF version 1.7 setting although it may not be supported by PDF viewers.

There are various other improvements over dvipdfm. The most notable one is more improved PDF input and output support: The cross-reference stream and object stream introduced in PDF-1.5 are also supported.

6 Graphics and Image Format

Graphics support was mostly rewritten. Support for BMP and JPEG2000 is added. An effort to preserve more information originally found in included images, e.g., embedded ICC Profiles and XMP Metadata, was made.

However, dvipdfmx does not support various features common to graphics manipulation programs such as resampling, color conversion, and selection of compression filters. Thus, it is recommended to use other programs specialized in image manipulation for preparation of images.

6.1 Supported Graphics File Formats

Supported formats are, PNG, JPEG, JPEG2000, BMP, PDF, and METAPOST generated EPS. All other format images, such as SVG and PostScript, must be converted to other format supported by dvipdfmx before inclusion. The '-D' option, as in dvipdfm, can be used for filtering images.

6.2 PNG and JPEG

PNG and JPEG are supported as in dvipdfm but it is mostly rewritten.

PNG support includes most of important features of PNG such as color palette, transparency, 16bit bit-depth color, embedded ICC Profiles, and calibrated colors specified by `gAMA` and `cHRM` chunks. XMP Metadata is also supported. Predictor filter may be applied for Flate compression which result in better compression for larger images. Application of the predictor filter sometimes makes compression speed very slow. Please try '`-C 0x20`' (disable predictor filters) command line option to check if slowness is due to the predictor filter.

JPEG is relatively well supported. `dvipdfmx` supports embedded ICC Profiles and CMYK color. Embedded XMP metadata is also supported. `dvipdfmx` uses JFIF or Exif data to determin image's physical size.

6.3 PDF and "MPS"

PDF inclusion is supported as in `dvipdfm`, with various important enhancement over `dvipdfm` for robust inclusion. `dvipdfmx` can handle cross-reference streams and object streams introduced in PDF-1.5. `dvipdfmx` also supports inclusion of PDF pages other than the first page. However, tagged PDF may cause problems and annotations are not kept.

For METAPOST generated EPS files, multi-byte encoding support is added. `dvipdfmx` also supports "METAPOST mode": When `dvipdfmx` is invoked with '`-M`' option, it enters in METAPOST mode and processes a METAPOST generated EPS file as input.[5]

6.4 Additional Supported Formats

BMP is supported but limited to uncompressed or RLE-compressed raster images. Extensions are not (won't be) supported.

JPEG2000 is also supported. It is restricted to JP2 and JPX baseline subset as required by PDF spec. It is not well supported and still in an experimental stage. J2C format and transparency are not supported.

6.5 Image Cache

Caching of images generated via filtering command specified with '`-D`' option is supported. It solves the problems that image inclusion becomes very slow when external filtering program such as GhostScript is invoked each time images are included.

Use '`-I`' option to enable this feature:

```
-I 24
```

where the integer represents the life of cache files, 24 hours in this example.

[5] `prologue` must be set to 2.

7 DVI Specials

dvipdfmx is mostly compatible to dvipdfm. Several DVI special commnads are added for more flexible PDF generation: Creation of arbitrary stream object, controlling dvipdfmx behavior, and some specials which might be useful for graphics drawing packages.

7.1 Additions to dvipdfm's pdf: Special

The pdf:fstream special is added, which enables creation of PDF stream object from an existing file.

```
pdf:fstream @identifier (filename) <<dictionary>>
```

where identifier and filename (specified as a PDF string object) are mandatory and a dictionary object which is to be added to the stream dictionary following the filename is optional.

For examples, to incorporate a XMP Metadata,

```
\special{pdf:fstream @xmp (test.xmp) <<
    /Type /Metadata
    /Subtype /XML
>>}
\special{pdf:put @catalog << /Metadata @xmp >>}
```

Similary, pdf:stream special can be used to create a PDF stream object from a PDF string instead of a file:

```
\special{pdf:stream @MyPattern
    (0.16 0 0 0.16 0 0 cm 4 w
     50 0 m 50 28 28 50 0 50 c S 100 50
     m 72 50 50 28 50 0 c S
     50 100 m 50 72 72 50 100 50 c S
     0 50 m 28 50 50 72 50 100 c S
     100 50 m 100 78 78 100 50 100 c 22 100 0 78 0 50 c
     0 22 22 0 50 0 c 78 0 100 22 100 50 c S
     0 0 m 20 10 25 5 25 0 c f 0 0 m 10 20 5 25 0 25 c f
     100 0 m 80 10 75 5 75 0 c f
     100 0 m 90 20 95 25 100 25 c f
     100 100 m 80 90 75 95 75 100 c f
     100 100 m 90 80 95 75 100 75 c f
     0 100 m 20 90 25 95 25 100 c f
     0 100 m 10 80 5 75 0 75 c f
     50 50 m 70 60 75 55 75 50 c 75 45 70 40 50 50 c f
     50 50 m 60 70 55 75 50 75 c 45 75 40 70 50 50 c f
     50 50 m 30 60 25 55 25 50 c
```

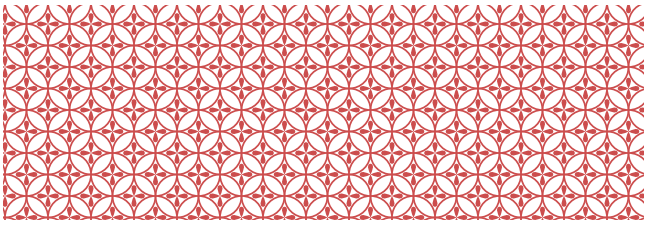

Figure 2: An example of tiling pattern.

```
     25 45 30 40 50 50 c f
     50 50 m 60 30 55 25 50 25 c 45 25 40 30 50 50 c f)
    <<
        /BBox [0 0 16 16]
        /PaintType 2
        /PatternType 1
        /Resources <<
            /ProcSet [/PDF]
        >>
        /TilingType 3
        /Type /Pattern
        /XStep 16
        /YStep 16
    >>
}
```

The above example defines a tiling pattern. With the following code:

```
\special{pdf:put @resources
    <<
        /ColorSpace << /CS01 [/Pattern /DeviceRGB] >>
        /Pattern << /PT01 @MyPattern >>
    >>
}
\special{pdf:content
    q 0.8 0.3 0.3 RG /CS01 cs 0.8 0.3 0.3 /PT01 scn
    0 0 240 80 re f
}
```

it draws a box filled with a tiling pattern as shown in Figure 2.

`pdf:mapline` and `pdf:mapfile` specials can be used to append a fontmap entry or to load a fontmap file:

```
pdf:mapline foo unicode bar
pdf:mapfile foo.map
```

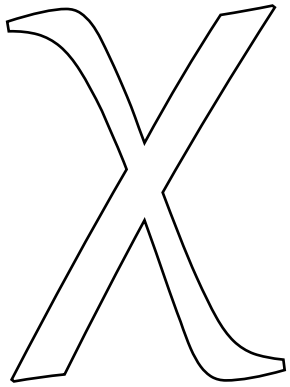

Figure 3: A character drawn in the PDF text rendering mode 1.

`pdf:majorversion` and `pdf:minorversion` specials can be used to specify major and minor version of output PDF.

```
pdf:minorversion 3
```

To protect output PDF with encryption, use `pdf:encrypt` special

```
pdf:encrypt userpw (foo) ownerpw (bar) length 128 perm 20
```

where user-password and owner-password must be specified as PDF string objects. (which can be empty) Numbers specifying key-length and permission flags here must be decimal numbers. See, section of "Encryption Support" for a brief description of permission flags.

Other notable extensions are `code`, `bcontent`, and `econtent`. The `code` special can be used to insert raw PDF graphics instructions into output page content stream. It is different from `dvipdfm`'s `content` special in that it does not enclose contents with a `q` and `Q` (save-restore of graphics state) pair. A typical usage of this special is:

```
\special{pdf:code q 1 Tr}
...some text goes here...
\special{pdf:code Q}
```

which changes text rendering mode to 1, as shown in Figure 3 for example.

Be careful on using this special as it is very easy to generate broken PDF files. The `bcontent` and `econtent` pair is fragile and often incompatible to other groups of special commands. It is not always guaranteed to work as 'expected'.

7.2 Dvipdfmx Extensions

Addition in TeX Live 2016

A new special `dvipdfmx:config` was introduced in TeXLive 2016 which makes it possible to invoke a command line option. Several single letter command line options except 'D' are supported. For examples,

```
dvipdfmx:config C 0x10
```

sets the `dvipdfmx`'s compatibility flag.

7.3 DVI Special Examples

The following code reads TeX source file and creates a stream object named as `SourceFile`, then creates a file attachment annotation.

```
\special{pdf:fstream @SourceFile (\jobname.tex)}%
\special{pdf:ann width 10bp height 20bp
   << /Type /Annot
      /Subtype /FileAttachment
      /FS <<
            /Type /Filespec
            /F    (\jobname.tex)
            /EF   << /F @SourceFile >>
          >>
      /Name     /PushPin
      /C        [0.8 0.2 0.2]
      /T        (The dvipdfmx project team)
      /Subj     (The Dvipdfmx User's Manual)
      /Contents (XeLaTeX source file of the manual.)
   >>
}
```

Viewer support is required for annotation to be viewed correctly. A push-pin image must be shown in the margin if viewer supports this kind of annotation. PDF viewer applications are required to provide predefined icon appearances at least for the following standard icons: `Graph`, `PushPin`, `PaperClip`, and `Tag`.

Various color spaces can be used. For examples, Separation color space:

Orange and Green

```
\special{pdf:stream @TintTransform1
  ({0 exch dup 0.62 mul exch 0})
  <<  /FunctionType 4
      /Domain [ 0.0 1.0 ]
```

```
      /Range  [ 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 ]
  >>
}
\special{pdf:stream @TintTransform2
  ({dup 0.78 mul exch dup 0.05 mul exch 0.71 mul 0})
  <<  /FunctionType 4
      /Domain [ 0.0 1.0 ]
      /Range  [ 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 ]
  >>
}
\special{pdf:obj @Orange [
    /Separation /Orange /DeviceCMYK @TintTransform1
  ]
}
\special{pdf:obj @Green [
    /Separation /Green /DeviceCMYK @TintTransform2
  ]
}
\mbox{%
  \special{pdf:put @resources <<
      /ColorSpace << /CS01 @Orange /CS02 @Green >>
    >>
  }%
  \special{pdf:code q /CS01 cs 1.0 scn}
  Orange
  \special{pdf:code Q}
  and
  \special{pdf:code q /CS02 cs 1.0 scn}
  Green
  \special{pdf:code Q}
}
```

`TintTransforms` defined here are functions for transforming *tint* values into approximate colors in the *alternate color space* (`DeviceCMYK` in this example). PostScript calculator functions are used for converting a single component value representing "Orange" or "Green" into a four component CMYK value to approximate those colors.

The `cs` operator for selecting color space and the `scn` operator for color values are used. Be sure that `pdf:put` command, which puts `ColorSpace` resources into the current page's resource dictionary, goes into the same page as subsequent drawing commands.

`dvipdfmx` currently does not have an easy interface for using various color space families such as CIE-Based color space (e.g., calibrated colors and color space with an ICC profile) and special color space (e.g., indexed, spot color, and shading and patterns).

XeTeX's transparency feature is currently lost in `xdvipdfmx`, but the same effect can be achieved by setting graphics state with `ExtGState` and `gs` operator. Here is a simple transparency example:

```
\special{pdf:obj @GS01 <<
  /Type /ExtGState /CA 0.5 /ca 0.5 /AIS false
>>}%
\mbox{%
  \special{pdf:put @resources <<
    /ExtGState << /GS01 @GS01 >>
  >>}%
  \special{pdf:code q /GS01 gs 1.0 0.8 0.2 rg}%
  α%
  \special{pdf:code 0.4 0.8 0.4 rg}%
  \hspace{-0.3emβ}%
  \hspace{-0.3em}\raisebox{0.5ex}{%
    \special{pdf:code 0.4 0.4 0.8 rg}%
    π%
  }%
  \special{pdf:code 1.0 0.2 0.4 rg}%
  \hspace{-0.2em}%
  γ%
  \special{pdf:code Q}%
}
```

where `CA` and `ca` represent opacity of stroke and fill color individually. Again, `pdf:put` command must go into the same page as subsequent grahics and text drawing operators.

8 Font Mapping

Syntax of fontmap file is basically the same as `dvipdfm`. There are few extensions in `dvipdfmx`. In addition to 8-bit `enc` files and keyword `builtin` and `none`, `dvipdfmx` accepts CMap name and the keyword `unicode` in the encoding field.

This section is completely irrevant to XeTeX users.

8.1 Extended Syntax and Options

Few options are available in `dvipdfmx` in addition to the original dvipdfm's one. All options and features that makes `dvipdfmx` to use unembedded fonts are deprecated, as by using them makes `dvipdfmx` to create PDF files which can be non-compliant to the ISO standards.

8.1.1 SFD Specification

For bundling up fonts split into multiple subfonts via SFD back into a single font, dvipdfmx supports an extended sytax of the form

```
tfm_name@SFD@  encoding  filename  options
```

A typical example looks like:

```
gbsn@EUC@  GB-EUC-H  gbsn00lp
```

where TFMs `gbsn00`, `gbsn01`, `gbsn02`... are mapped into a single font named `gbsn00lp` via the rule described in the SFD file `EUC`.

8.1.2 TrueType Collection Index

TrueType Collection index number can be specified with `:n:` in front of TrueType font name:

```
min10  H  :1:mincho
```

In this example, the option `:1:` tells `dvipdfmx` to select first TrueType font from the TTC font `mincho.ttc`. Alternatively, the '`-i`' option can be used in the option field to specify TTC index:

```
min10  H  mincho  -i 1
```

8.1.3 Non-embedding Switch

The character '`!`' in front of the font name can be used to indicate that the font shall not be embedded. This feature greatly reduces the size of the final PDF output, but the PDF file may not be viewed exactly in other systems on which appropriate fonts are not installed.

Use of this option is deprecated.

Character Collection	Font Family	Description
Adobe-Japan1	Ryumin-Light	PS printers
	GothicBBB-Medium	
Adobe-CNS1	MHei-Medium-Acro	Acrobat Reader 4
	MSung-Light-Acro	
Adobe-GB1	STSong-Light-Acro	
	STHeiti-Regular-Acro	
Adobe-Japan1	HeiseiMin-W3-Acro	
	HeiseiKakuGO-W5-Acro	
Adobe-Korea1	HYGoThic-Medium-Acro	
	HYSMyeongJo-Medium-Acro	
Adobe-CNS1	MSungStd-Light-Acro	Acrobat Reader 5
Adobe-GB1	STSongStd-Light-Acro	
Adobe-Korea1	HYSMyeongJoStd-Medium-Acro	
Adobe-CNS1	AdobeMingStd-Light-Acro	Adobe Reader 6
Adobe-GB1	AdobeSongStd-Light-Acro	
Adobe-Japan1	KozMinPro-Regular-Acro	
	KozGoPro-Medium-Acro	
Adobe-Korea1	AdobeMyungjoStd-Medium-Acro	
Adobe-CNS1	AdobeHeitiStd-Regular	Adobe Reader 7
Adobe-Japan1	KozMinProVI-Regular	Adobe Reader 8

Table 2: List of available 'Standard' CJK font. Most of them are available as a part of Adobe Asian Font Packs for each versions of Adobe or Acrobat Reader.

NOTE: `dvipdfmx` always converts input encodings to CIDs and then uses Identity CMaps[6] in the output PDF. However, ISO 32000-1:2008 describes as

> *The Identity-H and Identity-V CMaps shall not be used with a non-embedded font. Only standardized character sets may be used.*

which had never appeared in Adobe's PDF References. This makes all PDF files generated by `dvipdfmx` with non-embedded CID-keyed fonts non-compliant to the ISO standards.

8.1.4 'Standard' CJK Fonts

This feature is deprecated.

Use of this feature shall be avoided for new documents. It is described here since it might still be useful for some situations.

`dvipdfmx` recognizes several 'Standard' CJK fonts although there are no such notion in PDF. In older days where there were not so many freely available CJK fonts, it was sometimes useful to create PDF files without embedded fonts and let PDF viewers or printers to use substitute fonts (tend to be higher quality) installed in their systems. `dvipdfmx` 'knows' several fonts which might be available in PostScript printers and PDF applications such as Acrobat Reader, and uses them without actually having it. See, Table 2, for the list of available 'Standard' CJK fonts.

Only fixed-pitch glyphs (i.e., quarter, third, half, and full widths) are supported for those fonts.

[6] Predefined CMaps `Identity-H` and `Identity-V` for the identity mapping.

Figure 4: JIS2004 vs. JIS1990 form.

8.1.5 Stylistic Variants

Keywords `,Bold`, `,Italic`, and `,BoldItalic` can be used to create synthetic bold, italic, and bolditalic style variants from other font using PDF viewer's (or OS's) function.

Use of this option is deprecated.

```
jbtmo@UKS@  UniKSCms-UCS2-H  :0:!batang,Italic
```

Availability of this feature highly depends on the implementation of PDF viewers. This feature is usually not supported for embedded fonts. Notice that this option automatically disables font embedding thus use of it is deprecated.

8.2 Specifying Unicode Plane

As there are no existing 3-bytes or 4-bytes TFM formats, the only way to use Unicode characters other than the BMP is to map code range 0-65535 to different planes via (e.g., to plane 1) '`-p 1`' fontmap option. This option is available only when `unicode` is specified in the encoding field.

8.3 OpentType Layout Feature

OpenType Layout Feature fontmap options mentioned below are only meaningful when `unicode` is specified in the encoding field.

With the '`-w`' option, writing mode can be specified. '`-w 1`' denotes the font is for vertical writing. It automatically enables an OpentType Layout Feature related to vertical writing, namely, `vert` or `vrt2`, to choose proper glyphs for vertical text.

Addition in TEX Live 2017.

The '`-l`' (lower case el) option can be used to enable various OpenType Layout GSUB Features. For examples, '`-l jp04`' enables `jp04` feature to select JIS2004 forms for Kanjis. Features can be specified as a ":" separated list of OpenType Layout Feature tags like '`-l vkna:jp04`'. Script and language may be additionally specified as '`-l kana.JAN.ruby`'.

An example can be

```
uprml-v  unicode  SourceHanSerifJP-Light.otf  -w 1 -l jp90
```

which declares that font should be treated as for vertical writing and use JIS1990 form for Kanjis. (See, Figure 4 for an example)

This feature is limited to the single substitution, there are no way to select a glyph from multiple candidates, such as in `aalt`, and specifying general many-to-many glyph substitutions does not take effect.

9 Encryption Support

dvipdfmx offers basic PDF password security support including 256-bits AES encryption. Only "Standard" security handler is supported and public-key security handlers are currently not supported. Encryption is enabled by '-S' command line option.

When encrypting the document, up to two passwords can be specified – an owner password and a user password. If a user attempts to open an encrypted document that has a user password, PDF applications prompt for a password. Correctly supplying either password enables the user to open the document to display or to access to the contents. Depending on which password (user or owner) was supplied, additional operations allowed for an opened document is determined; full access for users who opened the document with the correct owner password or additional operations controlled by permission flags for users who opened the document with the correct user password.

Access permision flags can be specified via the '-P' option. Each bits of the (32-bit unsigned) integer number given to this option represents user access permissions; e.g., bit position 3 for allowing "print", 4 for "modify", 5 for "copy or extract", and so on. See, Table 3. For examples,

```
dvipdfmx -S -P 0x34 foo.dvi
```

allows printing, copying and extraction of text, and adding and modifying text annotation and filling in interactive form fields (but disallows modification of the contents of the document).

The '-K' option can be used to specify the encryption key length. The key length must be multiple of 8 in the range 40 to 128, or 256 (for PDF version 1.7 plus Adobe Extension or forthcoming PDF version 2.0). Please note that key length 256 requires Adobe's Extension to PDF-1.7 and hence PDF applications may not support it.

Password will be asked when encryption is enabled. It may not work well on Windows platforms. Use the pdf:encrypt special instead of command line options in this case.

The default values for 'K' is 40 and for '-P' is 0x003C0 (all bits from bit-position 3 to 6 set).

Bit Position	Meaning
3	Print the document.
4	Modify the contents of the document.
5	Copy or extract text and graphics from the document.
6	Add or modify text annotations, fill in interactive form fields. Creation and modification of interactive form field is also allowed if bit 4 is set.

Table 3: Flag bits and their short explaination for the Revision 2 Standard Security Handler.

10 Other Improvements

This section briefly describes other improvements made for dvipdfmx. There is an extension to glyph name handling in enc files for seamless support of both PostScript Type1 and TrueType fonts. PostScript Type1 font support is enhanced although this format might be considered obsolete. There are several changes in PostScript special support. Support for PostScript special is somewhat buggy and fragile.

10.1 Extended Glyph Name Syntax

dvipdfmx accepts the following syntax for glyph names in enc files: uni0130, zero.onum and T_h.liga. Each represents a glyph accessed with Unicode value U+0130, oldstyle number for zero and "Th" ligature accessed via OpenType Layout GSUB Feature onum and liga respectively. Note that dvipdfmx does not understand glyph names which directly use glyph indices such as index0102 or gid2104, since those indices are private to each font.

When dvipdfmx encounters a glyph name, e.g., T_h.liga, it first looks for OpenType post table if such glyph exists; if it exists, then dvipdfmx simply uses post talbe for mapping glyph name to glyph index; if not, dvipdfmx tries to convert T_h to Unicode sequence (U+0054 U+0068 in this example) via the AGL mapping; then, OpenType cmap table is used to further converting resulting Unicode sequence to a sequence of glyph indices; finally, OpenType Layout Feature liga is applied to get desired glyph.

Glyph name of the form a.swsh2 can be specified to denote 2nd swash variant form of letter 'a'.

10.2 CFF Conversion

dvipdfmx supports on-the-fly PostScript Type1 to CFF (Type1C) conversion which greatly reduces size of resulting PDF files when using Type1 fonts. Conversion is essentially 'lossless' and there should not be any quality loss. However, due to differences in the ability of rasterizers, there might be noticeable differences on the rendering result.

When using (older) Type1 fonts, dvipdfmx may give the following warning:

```
Obsolete four arguments of "endchar" will be used for Type1
"seac" operator.
```

This happens when there is an accented character made from composite glyphs. This warning message is given as Adobe made the use of endchar operator for composite glyphs deprecated.

However, as mentioned in "Appendix C Compatibility and Deprecated Operators" of Adobe Technical Note #5177, "Type 2 Charstring Format", PDF applications are supposed to support this operator. Hence, this warning message can be ignored. Please note that deprecated use of endchar operator for compositing glyphs is not allowed in ordinary OpenType fonts. So, dvipdfmx stops with an error if such font is encountered.

Classification	Operators
Arithmetic & Math	`add sub mul div neg truncate`
Stack Operation	`clear pop exch`
Graphis State	`gsave grestore setlinewidth setdash setlinecap setlinejoin setmiterlimit setgray setrgbcolor setcmykcolor`
Coordinate System	`concat scale translate rotate idtransform dtransform`
Path Construction	`currentpoint newpath closepath moveto rmoveto lineto rlineto curveto rcurveto arc arcn clip eoclip`
Painting	`stroke fill`
Glyph & Font	`show findfont scalefont setfont currentfont stringwidth`

Table 4: PostScript operators recognized by `dvipdfmx`.

Use of Type1 font should be avoided as much as possible. Please consider using OpenType version of font whenever possible.

10.3 PostScript Special

Text handling in PostScript special is extended to support CJK text. The following code draws Japanese text like shown in Figure 1:

```
\special{pdf:mapline uprml UniJIS-UTF8-H yumindb.ttf}
\special{ps: uprml findfont 16 scalefont setfont
  currentpoint moveto
  (...some Japanese text goes here...) show
}
```

Please note that support for PostScript operators in `dvipdfmx` is very limited. It is just enough for supporting METAPOST output. Only basic set of operators for arithmetic and math, stack operation and manupilation, graphics state, path construction and painting, glyph and font, are supported. See, Table 4 for the list of recognized PostScript operators.

It might be enough for the purpose of basic graphics drawing but as there are no support for conditionals and controls it is not enough for complicated tasks.

11 Incompatible Changes

There are various minor incompatible changes to `dvipdfm`.

The '`-C`' command line option may be used for compatibility to `dvipdfm` or older versions of `dvipdfmx`. The '`-C`' option takes flags meaning

- bit position 2: Use semi-transparent filling for tpic shading command, instead of opaque gray color. (requires PDF 1.4)

- bit position 3: Treat all CID-keyed font as fixed-pitch font. This is only for compatibility.
- bit position 4: Do not replace duplicate fontmap entries. `dvipdfm` behavior.
- bit position 5: Do not optimize PDF destinations. Use this if you want to refer from other files to destinations in the current file.
- bit position 6: Do not use predictor filter for Flate compression.
- bit position 7: Do not use object stream.

The remap option '`-r`' in fontmaps is no longer supported and is silently ignored. The command line option '`-e`' to disable partial (subset) font embedding is not supported.

12 Font Licensing and Embedding

In OpenType format, information regarding how a font should be treated when creating documents can be recorded.[7] `dvipdfmx` uses this information to decide whether embedding font is permitted.

This font embedding information is indicated by a flag called `fsType`; each bit representing different restrictions on font embedding. If multiple flag bits are set in `fsType`, the least restrictive license granted takes precedence in `dvipdfmx`. The `fsType` flag bits recognized by `dvipdfmx` is as follows:

- Installable embedding
- Editable embedding
- Embedding for Preview & Print only

`dvipdfmx` give the following warning message for fonts with 'Preview & Print only' setting:

```
This document contains 'Preview & Print' only licensed font
```

For fonts with this type of licensing, font embedding is allowed solely for the purpose of (on-screen) viewing and/or printing; further editing of the document or extracting embedded font data for other purposes are not allowed. One way to ensure this condition is to protect your document with a non-empty password.

All other flags are treated as more restrictive license than any of the above flags and treated as "No embedding allowed"; e.g., if both of the editable-embedding flag and unrecognized license flag is set, the font is treated as editable-embedding allowed, however, if only unrecognized flags are set, the font is not embedded.

Embedding flags are preserved in embedded font if the font is embedded as a TrueType font or a CIDFontType2 CID-keyed font. For all font embedded as a PostScript font (Type1C and CIDFontType0 CID-keyed font), they are not preserved. Only `Copyright` and `Notice` in the FontInfo dictionary are preserved in this case.

[7] See, " OpenType Sepcification: OS/2 – OS/2 and Windows Metrics Table".

Some font vendors put different embedding restrictions for different condition; e.g., font embedding might be not permitted for commercial materials unless you acquire "commercial license" separately. Please read EULA carefully before making decision on font usage.

See, for examples, Adobe's site on font embedding permissions for fonts in the Adobe Type Library. Microsoft also has a FAQ page on Font Redistribution.

For Japanese font in general, embedding permission tend to be somewhat restrictive. Japanese users should read the statement regarding font embedding from Japan Typography Association (in Japanese):

`http://www.typography.or.jp/act/morals/moral4.html`

`dvipdfmx` does not support full embedding. Only subset embedding is supported.

13 Important Changes

Here is a list of important changes since the TeX Live 2016 release:

- Changes to make PDF/A creation easier: Always write CIDSet and CharSet for embedded fonts. Do not compress XMP metadata.
- Merge from libdpx for pTeX-ng by Clerk Ma.
- Addition of `STHeiti-Regular-Acro` for CJK 'Standard' fonts.
- Command line option '`-p`' takes precedence over `papersize` and `pagesize` specials.
- Fixed serious bugs in supporting '`unicode`' encoding: OpenType Layout Feature `vert` and `vrt2` was not enabled. Support for format 2 CFF charsets was broken.
- Added simplified version of OpenType Layout support: The "`-l`" option in fontmaps.

The full `ChangeLog` entries can be viewed via the web interface of TeX Live SVN repository:

`http://www.tug.org/svn/texlive/trunk/Build/source/texk/`

There was an undocumented feature for supporting OpenType Layout but it was dropped. Simplified support for OpenType Layout was intorduced instead.

Further Reading

[1] Adobe's PDF References and a free copy of ISO 32000-1:2008 standard are available from "PDF Technology Center" on Adobe Developer Connection.

[2] The OpenType Specification is available from Microsoft's site: "OpenType Specification".

[3] An article regarding DVI specials: Jin-Hwan Cho, "DVI specials for PDF generation", TUGboat, 30(1):6-11, 2009.

GNU Free Documentation License