Guide of pTEX for developers not interested in Japanese

Japanese TEX Development Community*

version p3.8.2, October 19, 2019

pTEX and its variants, upTEX, ε-pTEX and ε-upTEX, are all TEX engines with native Japanese support. Its output is always a DVI file, which can be processed by several DVI drivers with Japanese support including *dvips* and *dvipdfmx*. Formats based on LATEX is called pLATEX when running on pTEX/ε-pTEX, and called upLATEX when running on upTEX/ε-upTEX.

Purpose of this document

This document is written for developers of TEX/LATEX, who aim to support pTEX/pLATEX and its variants upTEX/upLATEX. Knowledge of the followings are assumed:

- Basic knowledge of Western TEX (Knuthian TEX, ε-TEX and pdfTEX),
- ... and its programming conventions.

Any knowledge of Japanese (characters, encodings, typesetting conventions etc.) is not assumed; some explanations are provided in this document when needed. We hope that this document helps authors of packages or classes to proceed with supporting pTEX family smoothly.

> Note: This English edition (ptex-guide-en.pdf) is *not* meant to be a complete translation of Japanese edition (ptex-manual.pdf). For example, this document does not cover the following aspects of pTEX:
>
> - Typesetting conventions of Japanese characters
> - Details of vertical writing
>
> For beginners of writing Japanese texts, please refer to the Japanese edition.

*https://texjp.org, e-mail: issue(at)texjp.org

Contents

Part I
Brief introduction

1 pTEX and its variants

There is no advantage to choose pTEX/upTEX over ε-pTEX/ε-upTEX, so we focus mainly on ε-pTEX/ε-upTEX.

2 Compatibility with Western TEX

pTEX/upTEX are almost uppercompatible with Knuthian TEX, however, they do not pass TRIP test. In pTEX/upTEX, input handling is different from Knuthian TEX; if a pair of two or more 8-bit codes matches Japanese character code, it is regarded as one Japanese character. There is no difference in handling 8-bit TFM font.

ε-pTEX/ε-upTEX are almost uppercompatible with ε-TEX, however, input handling is similar to pTEX/upTEX. It does not pass e-TRIP test. That said, please note that "raw ε-TEX" is unavailable anymore in TEX Live and derived distributions; they provide a command `etex` only as "DVI mode of pdfTEX." We should note that ε-pTEX/ε-upTEX are *not* uppercompatible with DVI mode of pdfTEX, which will be discussed later in section 6.6.

3 LATEX on pTEX/upTEX — pLATEX/upLATEX

Formats based on LATEX is called pLATEX when running on pTEX/ε-pTEX, and called upLATEX when running on upTEX/ε-upTEX. In recent versions (around 2011) of TEX Live and its derivatives, the default engines of pLATEX and upLATEX are ε-pTEX and ε-upTEX. That is, the command `platex` starts ε-pTEX (not pTEX) with preloaded format `platex.fmt`.

In the kernel level (`platex.ltx` and `uplatex.ltx`), pLATEX and upLATEX adds some additional commands related to the followings:

- Selection of Japanese fonts
- Crop marks (called "tombow") for printings
- Adjustment for mixing horizontal and vertical texts

For authors, pLATEX/upLATEX is almost uppercompatible with original LATEX, except for the followings:

- Order of float objects; in pLATEX/upLATEX, ⟨*bottom float*⟩ is placed above ⟨*footnote*⟩. That is, the complete order is ⟨*top float*⟩ → ⟨*body text*⟩ → ⟨*bottom float*⟩ → ⟨*footnote*⟩.

For developers, additional cares may be needed, for changes in the kernel macros and/or absence of pdfTEX features.

4 Eminent characteristics of pTEX/upTEX

The most important characteristics of pTEX/upTEX can be summarized as follows:

- Japanese characters are interpret and handled completely apart from Western characters.
- Texts can be aligned vertically, called "tate-gumi" (縦組). The horizontal alignment of texts is called "yoko-gumi" (横組), and both "tate-gumi" and "yoko-gumi" can be mixed even within a single document.

Part II

Details

5 Output format — DVI

The output of pTEX/upTEX is always a DVI file. Its DVI format is completely compatible with Knuthian TEX, as long as the following conditions are met:

- No Japanese characters are typeset.
- There is no portion of vertical text alignment.

5.1 Extensions of DVI format

In pTEX/upTEX, some additional DVI commands, which are defined in the standard [1] but never used in TEX82, are used.

- `set2` (129), `put2` (134): Appears in both pTEX and upTEX DVI. Used to typeset a Japanese character with 2-byte code.
- `set3` (130), `put3` (135): Appears in only upTEX DVI. Used to typeset a Japanese character with 3-byte code.

When pTEX is going to typeset a Japanese character into DVI, it is encoded in JIS, which is always a 2-byte code. For this purpose, `set2` or `put2` are used. When upTEX is going to output a Japanese character into DVI, it is encoded in UTF-32. If the code is equal to or less than `U+FFFF`, the lower 16-bit is used with `set2` or `put2`. If the code is equal to or greater than `U+10000`, the lower 24-bit is used with `set3` or `put3`.

In addition, pTEX/upTEX defines one additional DVI command.

- `dir` (255): Used to change directions of text alignment.

The DVI format in the preamble is always set to 2, as with TEX82. On the other hand, the DVI ID in the postamble can be special. Normally it is set to 2, as with TEX82; however, when `dir` (255) appears at least once in a single pTEX/upTEX DVI, the `post_post` table of postamble contains ID = 3.

5.2 DVI drivers with Japanese support

There is some DVI drivers with Japanese support. The most eminent drivers are *dvips* and *dvipdfmx*. Nowadays most of casual Japanese users are using *dvipdfmx* as a DVI driver. On the other hand, users of *dvips* are unignorable, especially those working in publishing industry.

5.2.1 Using *dvipdfmx*

A DVI file which is output by pTeX can be converted directly to a PDF file using dvipdfmx.

Note for casual LaTeX users — when you choose to process the resulting DVI file with dvipdfmx after running LaTeX (command `platex` or `uplatex`), you need to pass a proper driver option `[dvipdfmx]` for all driver-dependent packages, such as `graphicx` and `color`. This is because the default for such packages is set to `dvips` mode as with the original LaTeX in DVI mode (command `latex`). For simplicity, we recommend a global driver option `[dvipdfmx]` as in the following example:

```
\documentclass[dvipdfmx,...]{article}
\usepackage{graphicx}
\usepackage{color}
```

5.2.2 Using *dvips*

A DVI file which is output by pTeX can be converted to a PostScript file using dvips.

The resulting PostScript file can then be converted to a PDF file using Ghostscript (ps2pdf) or Adobe Distiller. When using Ghostscript, a proper setup of Japanese font must be done before converting PostScript into PDF. An easy solution for the setup is a script `cjk-gs-integrate` developed by Japanese TeX Development Community.

6 Programming on pTeX family

We focus on programming aspects of pTeX and its variants.

6.1 Number of registers

pTeX and upTeX have exactly the same number (= 256) of registers as Knuthian TeX. ε-pTeX and ε-upTeX in extended mode have more registers; there are 65536, which is twice as many as 32768 of ε-TeX.

6.2 Additional primitives

Here we provide only complete lists of additional primitives of pTeX family in alphabetical order. Feature of each primitive can be found in Japanese edition.

6.2.1 SyncTeX additions (available in pTeX, upTeX, ε-pTeX, ε-upTeX)

In the standard build of TeX Live , SyncTeX extension is unavailable in Knuthian TeX; however, it is enabled in pTeX family.

- \synctex

6.2.2 pTEX additions (available in pTEX, upTEX, ε-pTEX, ε-upTEX)

- \autospacing
- \autoxspacing
- \disinhibitglue — New primitive since p3.8.2 (TEX Live 2019)
- \dtou
- \euc
- \ifdbox — New primitive since p3.2 (TEX Live 2011)
- \ifddir — New primitive since p3.2 (TEX Live 2011)
- \ifmbox — New primitive since p3.7.1 (TEX Live 2017)
- \ifmdir
- \iftbox
- \iftdir
- \ifybox
- \ifydir
- \inhibitglue
- \inhibitxspcode
- \jcharwidowpenalty
- \jfam
- \jfont
- \jis
- \kanjiskip
- \kansuji
- \kansujichar
- \kcatcode
- \kuten
- \noautospacing
- \noautoxspacing
- \postbreakpenalty

- `\prebreakpenalty`
- `\ptexminorversion` — New primitive since p3.8.0 (TEX Live 2018)
- `\ptexrevision` — New primitive since p3.8.0 (TEX Live 2018)
- `\ptexversion` — New primitive since p3.8.0 (TEX Live 2018)
- `\scriptbaselineshiftfactor` — New primitive since p3.7 (TEX Live 2016)
- `\scriptscriptbaselineshiftfactor` — New primitive since p3.7 (TEX Live 2016)
- `\showmode`
- `\sjis`
- `\tate`
- `\tbaselineshift`
- `\textbaselineshiftfactor` — New primitive since p3.7 (TEX Live 2016)
- `\tfont`
- `\xkanjiskip`
- `\xspcode`
- `\ybaselineshift`
- `\yoko`
- `H`
- `Q`
- `zh`
- `zw`

6.2.3 upTEX additions (available in upTEX, ε-upTEX)

- `\disablecjktoken`
- `\enablecjktoken`
- `\forcecjktoken`
- `\kchar`
- `\kchardef`
- `\ucs`
- `\uptexrevision` — New primitive since u1.23 (TEX Live 2018)
- `\uptexversion` — New primitive since u1.23 (TEX Live 2018)

6.2.4 ε-pTeX additions (available in ε-pTeX, ε-upTeX)

- `\epTeXinputencoding` — New primitive since 160201 (TeX Live 2016)
- `\epTeXversion` — New primitive since 180121 (TeX Live 2018)
- `\expanded` — New primitive since 180518 (TeX Live 2019)
- `\hfi`
- `\ifincsname` — New primitive since 190709 (TeX Live 2020)
- `\ifpdfprimitive` — New primitive since 150805 (TeX Live 2016)
- `\lastnodechar` — New primitive since 141108 (TeX Live 2015)
- `\lastnodesubtype` — New primitive since 180226 (TeX Live 2018)
- `\odelcode`
- `\odelimiter`
- `\omathaccent`
- `\omathchar`
- `\omathchardef`
- `\omathcode`
- `\oradical`
- `\pagefistretch`
- `\pdfcreationdate` — New primitive since 130605 (TeX Live 2014)
- `\pdfelapsedtime` — New primitive since 161114 (TeX Live 2017)
- `\pdffiledump` — New primitive since 140506 (TeX Live 2015)
- `\pdffilemoddate` — New primitive since 130605 (TeX Live 2014)
- `\pdffilesize` — New primitive since 130605 (TeX Live 2014)
- `\pdflastxpos`
- `\pdflastypos`
- `\pdfmdfivesum` — New primitive since 150702 (TeX Live 2016)
- `\pdfnormaldeviate` — New primitive since 161114 (TeX Live 2017)
- `\pdfpageheight`
- `\pdfpagewidth`
- `\pdfprimitive` — New primitive since 150805 (TeX Live 2016)
- `\pdfrandomseed` — New primitive since 161114 (TeX Live 2017)

- `\pdfresettimer` — New primitive since 161114 (TeX Live 2017)
- `\pdfsavepos`
- `\pdfsetrandomseed` — New primitive since 161114 (TeX Live 2017)
- `\pdfshellescape` — New primitive since 141108 (TeX Live 2015)
- `\pdfstrcmp`
- `\pdfuniformdeviate` — New primitive since 161114 (TeX Live 2017)
- `\readpapersizespecial` — New primitive since 180901 (TeX Live 2019)
- `\vfi`
- `fi`

6.3 Omitted primitives and unsupported features

Compared to Knuthian TeX and ε-TeX, some primitives are omitted due to conflict with Japanese handling. One is encTeX extension, such as `\mubyte`. The MLTeX extension, such as `\charsubdef`, is included but not well-tested.

6.4 Behavior of Western TeX primitives

Here we provide some notes on behavior of Knuthian TeX and ε-TeX primitives when used within pTeX family.

6.4.1 Primitives with limitations in handling Japanese

Each of the following primitives allows only character codes 0–255; other codes will give an error "! Bad character code."

`\catcode`, `\sfcode`, `\mathcode`, `\delcode`, `\mathchardef`, `\lccode`, `\uccode`.

Each of the following primivies has `\...char` in its name, however, the effective values are restricted to 0–255.

`\endlinechar`, `\newlinechar`, `\escapechar`, `\defaulthyphenchar`, `\defaultskewchar`.

6.4.2 Primitives capable of handling Japanese

The following primitives are extended to support Japanese characters:

- `\char` ⟨*character code*⟩, `\chardef` ⟨*control sequence*⟩=⟨*character code*⟩

  In addition to 0–255, internal codes of Japanese characters are allowed. For putting Japanese characters, a Japanese font is chosen.

▶ `\font`, `\fontname`, `\fontdimen`

▶ `\accent` ⟨*character code*⟩=⟨*character*⟩

▶ `\if` $\langle \mathit{token}_1 \rangle$ $\langle \mathit{token}_2 \rangle$, `\ifcat` $\langle \mathit{token}_1 \rangle$ $\langle \mathit{token}_2 \rangle$

Japanese character token is also allowed. In that case,

- `\if` tests the internal character code of the Japanese character.
- `\ifcat` tests the `\kcatcode` of the Japanese character.

TeXbook describes the behavior of `\if` and `\ifcat` as follows;

> If either token is a control sequence, TeX considers it to have character code 256 and category code 16, unless the current equivalent of that control sequence has been `\let` equal to a non-active character token.

However, this includes a lie; in the real implementation of tex.web, a control sequence is considered to have a category code 0.

6.5 Case study

Based on the above knowledge, we provide some code examples which may be useful for package developers.

6.5.1 Detecting pTeX

Since the primitive `\ptexversion` is rather new (added in 2018), the safer solution for detecting pTeX is to test if a primitive `\kanjiskip` is defined.

```
\ifx\kanjiskip\undefined
  % not pTeX, upTeX, e-pTeX and e-upTeX
\else
  % pTeX, upTeX, e-pTeX and e-upTeX
\fi
```

6.6 Difference from pdfTeX in DVI mode

As stated above, ε-pTeX/ε-upTeX are *not* uppercompatible with DVI mode of pdfTeX.

6.7 Recommendation for file encoding

6.8 Input handling

For simplicity, first we introduce of input handling of ε-upTeX.

6.9 Japanese tokens

7 Basic introduction to Japanese typesetting

This section does not aim to explain Japanese typesetting completely; here we provide a minimum requirement for “getting away” with Japanese.

7.1 Automatic insertion of glues and penalties

Sometimes pTEX family automatically inserts glues and penalties between characters.

7.2 Japanese fonts

For typesetting Japanese characters, a JFM (Japanese TEX font metric) format is used. It is a modified version of TEX TFM.

References

[1] TUG DVI Standards Working Group, *The DVI Driver Standard, Level 0*.
https://ctan.org/pkg/dvistd

Index