pLaTeX 2εのフォントコマンド v1.5

Ken Nakano & Hideaki Togashi

作成日：2006/11/10

Contents

1 概要 **1**
1.1 DOCSTRIP プログラムのためのオプション 2

2 コード **2**
2.1 準備 2
2.1.1 和文フォント属性 2
2.1.2 長さ変数 3
2.1.3 一時コマンド 4
2.1.4 フォントリスト 4
2.1.5 支柱 5
2.2 コマンド 5
2.3 デフォルト設定ファイルの読み込み 21

3 デフォルト設定ファイル **21**
3.1 イタリック補正 22
3.2 テキストフォント 22
3.3 プリロードフォント 23
3.4 組版パラメータ 23

4 フォント定義ファイル **24**

索 引 **25**

1 概要

ここでは、和文書体を NFSS2 のインターフェイスで選択するためのコマンドやマクロについて説明をしています。また、フォント定義ファイルや初期設定ファイルなどの説明もしています。新しいフォント選択コマンドの使い方については、`fntguide.tex` や `usrguide.tex` を参照してください。

第 1 節 この節です。このファイルの概要と DOCSTRIP プログラムのためのオプションを示しています。

第 2 節 実際のコードの部分です。

第 3 節 プリロードフォントやエラーフォントなどの初期設定について説明をしています。

第 4 節 フォント定義ファイルについて説明をしています。

1.1 DOCSTRIP プログラムのためのオプション

DOCSTRIP プログラムのためのオプションを次に示します。

オプション	意味
plcore	`plfonts.ltx` を生成します。
trace	`ptrace.sty` を生成します。
JY1mc	横組用、明朝体のフォント定義ファイルを生成します。
JY1gt	横組用、ゴシック体のフォント定義ファイルを生成します。
JT1mc	縦組用、明朝体のフォント定義ファイルを生成します。
JT1gt	縦組用、ゴシック体のフォント定義ファイルを生成します。
pldefs	`pldefs.ltx` を生成します。次の 4 つのオプションを付加することで、プリロードするフォントを選択することができます。デフォルトは 10pt です。
xpt	10pt プリロード
xipt	11pt プリロード
xiipt	12pt プリロード
ori	`plfonts.tex` に似たプリロード

2 コード

この節で、具体的に NFSS2 を拡張するコマンドやマクロの定義を行なっています。

2.1 準備

NFSS2 を拡張するための準備です。和文フォントの属性を格納するオブジェクトや長さ変数、属性を切替える際の判断材料として使うリストなどを定義しています。

2.1.1 和文フォント属性

ここでは、和文フォントの属性を格納するためのオブジェクトについて説明をしています。

`\k@encoding`
`\ck@encoding`
`\cy@encoding`
`\ct@encoding`

和文エンコードを示すオブジェクトです。`\ck@encoding` は、最後に選択された和文エンコード名を示しています。`\cy@encoding` と `\ct@encoding` はそれぞれ、最後に選択された、横組用と縦組用の和文エンコード名を示しています。

```
1 ⟨*plcore⟩
2 \let\k@encoding\@empty
3 \let\ck@encoding\@empty
4 \def\cy@encoding{JY1}
5 \def\ct@encoding{JT1}
```

\k@family 和文書体のファミリを示すオブジェクトです。

```
\let\k@family\@empty
```

\k@series 和文書体のシリーズを示すオブジェクトです。

```
\let\k@series\@empty
```

\k@shape 和文書体のシェイプを示すオブジェクトです。

```
\let\k@shape\@empty
```

\curr@kfontshape 現在の和文フォント名を示すオブジェクトです。

```
\def\curr@kfontshape{\k@encoding/\k@family/\k@series/\k@shape}
```

\rel@fontshape 関連付けされたフォント名を示すオブジェクトです。

```
\def\rel@fontshape{\f@encoding/\f@family/\f@series/\f@shape}
```

2.1.2 長さ変数

ここでは、和文フォントの幅や高さなどを格納する変数について説明をしています。

頭文字が大文字の変数は、ノーマルサイズの書体の大きさで、基準値となります。これらは、jart10.clo などの補助クラスファイルで設定されます。

小文字だけからなる変数は、フォントが変更されたときに（\selectfont 内で）更新されます。

\Cht \cht \Cht は基準となる和文フォントの文字の高さを示します。\cht は現在の和文フォントの文字の高さを示します。なお、この “高さ” はベースラインより上の長さです。

```
\newdimen\Cht
\newdimen\cht
```

\Cdp \cdp \Cdp は基準となる和文フォントの文字の深さを示します。\cdp は現在の和文フォントの文字の深さを示します。なお、この “深さ” はベースラインより下の長さです。

```
\newdimen\Cdp
\newdimen\cdp
```

\Cwd \cwd \Cwd は基準となる和文フォントの文字の幅を示します。\cwd は現在の和文フォントの文字の幅を示します。

```
\newdimen\Cwd
\newdimen\cwd
```

\Cvs \cvs \Cvs は基準となる行送りを示します。ノーマルサイズの\baselineskip と同値です。\cvs は現在の行送りを示します。

```
\newdimen\Cvs
\newdimen\cvs
```

\Chs \chs \Chs は基準となる字送りを示します。\Cwd と同値です。\chs は現在の字送りを示します。

```
\newdimen\Chs
\newdimen\chs
```

\cHT \cHT は、現在のフォントの高さに深さを加えた長さを示します。\set@fontsize コマンド（実際は\size@update）で更新されます。

```
\newdimen\cHT
```

2.1.3 一時コマンド

\afont LaTeX 内部の\do@subst@correction マクロでは、\fontname\font で返される外部フォント名を用いて、LaTeX フォント名を定義しています。したがって、\font をそのまま使うと、和文フォント名に欧文の外部フォントが登録されたり、縦組フォント名に横組用の外部フォントが割り付けられたりしますので、\jfont か\tfont を用いるようにします。\afont は、\font コマンドの保存用です。

```
\let\afont\font
```

2.1.4 フォントリスト

ここでは、フォントのエンコードやファミリの名前を登録するリストについて説明をしています。

pLaTeX 2ε の NFSS2 では、一つのコマンドで和文か欧文のいずれか、あるいは両方を変更するため、コマンドに指定された引数が何を示すのかを判断しなくてはなりません。この判断材料として、リストを用います。

このときの具体的な判断手順については、エンコード選択コマンドやファミリ選択コマンドなどの定義を参照してください。

\inlist 次のコマンドは、エンコードやファミリのリスト内に第二引数で指定された文字列があるかどうかを調べるマクロです。

```
\def\inlist@#1#2{%
  \def\in@@##1<#1>##2##3\in@@{%
    \ifx\in@##2\in@false\else\in@true\fi}%
  \in@@#2<#1>\in@\in@@}
```

\enc@elt \fam@elt \enc@elt と\fam@elt は、登録されているエンコードに対して、なんらかの処理を逐次的に行ないたいときに使用することができます。

```
\def\fam@elt{\noexpand\fam@elt}
\def\enc@elt{\noexpand\enc@elt}
```

\fenc@list \kenc@list \kyenc@list \ktenc@list \fenc@list には、\DeclareFontEncoding コマンドで宣言されたエンコード名が格納されていきます。

\kyenc@list は、\DeclareYokoKanjiEncoding コマンドで宣言されたエンコード名が格納されていきます。\ktenc@list には、\DeclareTateKanjiEncoding コマンドで宣言されたエンコード名が格納されていきます。

ここで、これらのリストに具体的な値を入れて初期化をするのは、リストにエンコードの登録をするように\DeclareFontEncoding を再定義する前に、欧文エンコードが宣言されるため、リストに登録されないからです。

```
\def\fenc@list{\enc@elt<OML>\enc@elt<T1>\enc@elt<OT1>\enc@elt<OMS>%
               \enc@elt<OMX>\enc@elt<TS1>\enc@elt<U>}
\let\kenc@list\@empty
\let\kyenc@list\@empty
\let\ktenc@list\@empty
```

\kfam@list \ffam@list \notkfam@list \notffam@list \kfam@list には、\DeclareKanjiFamily コマンドで宣言されたファミリ名が格納されていきます。

\ffam@list には、\DeclareFontFamily コマンドで宣言されたファミリ名が格納されていきます。

\notkfam@list には、和文ファミリではないと推測されたファミリ名が格納されていきます。このリストは\fontfamily コマンドで作成されます。

\notffam@list には欧文ファミリではないと推測されたファミリ名が格納されていきます。このリストは\fontfamily コマンドで作成されます。

ここで、これらのリストに具体的な値を入れて初期化をするのは、リストにファミリの登録をするように、\DeclareFontFamily が再定義される前に、このコマンドが使用されるため、リストに登録されないからです。

```
\def\kfam@list{\fam@elt<mc>\fam@elt<gt>}
\def\ffam@list{\fam@elt<cmr>\fam@elt<cmss>\fam@elt<cmtt>%
               \fam@elt<cmm>\fam@elt<cmsy>\fam@elt<cmex>}
```

つぎの二つのリストの初期値として、上記の値を用います。これらのファミリ名は、和文でないこと、欧文でないことがはっきりしています。

```
\let\notkfam@list\ffam@list
\let\notffam@list\kfam@list
```

2.1.5 支柱

行間の調整などに用いる支柱です。支柱のもととなるボックスの大きさは、フォントサイズが変更されるたびに、\set@fontsize コマンドによって変化します。

フォントサイズが変更されたときに、\set@fontsize コマンドで更新されます。

\tstrutbox
\zstrutbox

\tstrutbox は高さと深さが５対５、\zstrutbox は高さと深さが７対３の支柱ボックスとなります。これらは縦組ボックスの行間の調整などに使います。なお、横組ボックス用の支柱は\strutbox で、高さと深さが７対３となっています。

```
\newbox\tstrutbox
\newbox\zstrutbox
```

\strut
\tstrut
\zstrut

\strutbox は\yoko ディレクションで組まれていますので、縦組ボックス内で \unhcopy をするとエラーとなります。このマクロは ltplain.dtx で定義されています。

```
\def\strut{\relax
  \ifydir
    \ifmmode\copy\strutbox\else\unhcopy\strutbox\fi
  \else
    \ifmmode\copy\tstrutbox\else\unhcopy\tstrutbox\fi
  \fi}
\def\tstrut{\relax\hbox{\tate
   \ifmmode\copy\tstrutbox\else\unhcopy\tstrutbox\fi}}
\def\zstrut{\relax\hbox{\tate
   \ifmmode\copy\zstrutbox\else\unhcopy\zstrutbox\fi}}
```

2.2 コマンド

次のコマンドの定義をしています。

コマンド	意味
\Declare{Font|YokoKanji|TateKanji}Encoding	エンコードの宣言
\Declare{Yoko|Tate}KanjiEncodingDefaults	デフォルトの和文エンコードの宣言
\Declare{Font|Kanji}Family	ファミリの宣言
\DeclareKanjiSubstitution	和文の代用フォントの宣言
\DeclareErrorKanjiFont	和文のエラーフォントの宣言
\DeclareFixedFont	フォントの名前の宣言
\reDeclareMathAlphabet	和欧文を同時に切り替えるコマンド宣言
\{Declare|Set}RelationFont	従属書体の宣言
\userelfont	欧文書体を従属書体にする
\selectfont	フォントを切り替える
\set@fontsize	フォントサイズの変更
\adjustbaseline	ベースラインシフト量の設定
\{font|roman|kanji}encoding	エンコードの指定
\{font|roman|kanji}family	ファミリの指定
\{font|roman|kanji}series	シリーズの指定
\{font|roman|kanji}shape	シェイプの指定
\use{font|roman|kanji}	書体の切り替え
\normalfont	デフォルト値の設定に切り替える
\mcfamily,\gtfamily	和文書体を明朝体、ゴシック体にする
\textunderscore	テキストモードでの下線マクロ

\DeclareFontEncoding
\DeclareFontEncoding@

欧文エンコードを宣言するためのコマンドです。ltfssbas.dtx で定義されているものを、\fenc@list を作るように再定義をしています。

```
\def\DeclareFontEncoding{%
  \begingroup
  \nfss@catcodes
  \expandafter\endgroup
  \DeclareFontEncoding@}
%
\def\DeclareFontEncoding@#1#2#3{%
  \expandafter
  \ifx\csname T@#1\endcsname\relax
     \def\cdp@elt{\noexpand\cdp@elt}%
     \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
                    {\default@family}{\default@series}%
                    {\default@shape}}%
     \expandafter\let\csname#1-cmd\endcsname\@changed@cmd
     \def\enc@elt{\noexpand\enc@elt}%
     \xdef\fenc@list{\fenc@list\enc@elt<#1>}%
  \else
     \@font@info{Redeclaring font encoding #1}%
  \fi
  \global\@namedef{T@#1}{#2}%
  \global\@namedef{M@#1}{\default@M#3}%
  \xdef\LastDeclaredEncoding{#1}%
  }
```

\DeclareKanjiEncoding
\DeclareYokoKanjiEncoding
\DeclareYokoKanjiEncoding@
\DeclareTateKanjiEncoding
\DeclareTateKanjiEncoding@

和文エンコードの宣言をするコマンドです。

```
\def\DeclareKanjiEncoding#1{%
```

```
  \@latex@warning{%
     The \string\DeclareKanjiEncoding\space is obsoleted command.  Please use
     \MessageBreak
     the \string\DeclareTateKanjiEncoding\space for `Tate-kumi' encoding, and
     \MessageBreak
     the \string\DeclareYokoKanjiEncoding\space for `Yoko-kumi' encoding.
     \MessageBreak
     I treat the `#1' encoding as `Yoko-kumi'.}
  \DeclareYokoKanjiEncoding{#1}%
}
\def\DeclareYokoKanjiEncoding{%
  \begingroup
  \nfss@catcodes
  \expandafter\endgroup
  \DeclareYokoKanjiEncoding@}
%
\def\DeclareYokoKanjiEncoding@#1#2#3{%
  \expandafter
  \ifx\csname T@#1\endcsname\relax
    \def\cdp@elt{\noexpand\cdp@elt}%
    \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
                   {\default@k@family}{\default@k@series}%
                   {\default@k@shape}}%
    \expandafter\let\csname#1-cmd\endcsname\@changed@kcmd
    \def\enc@elt{\noexpand\enc@elt}%
    \xdef\kyenc@list{\kyenc@list\enc@elt<#1>}%
    \xdef\kenc@list{\kenc@list\enc@elt<#1>}%
  \else
    \@font@info{Redeclaring KANJI (yoko) font encoding #1}%
  \fi
  \global\@namedef{T@#1}{#2}%
  \global\@namedef{M@#1}{\default@KM#3}%
  }
%
\def\DeclareTateKanjiEncoding{%
  \begingroup
  \nfss@catcodes
  \expandafter\endgroup
  \DeclareTateKanjiEncoding@}
%
\def\DeclareTateKanjiEncoding@#1#2#3{%
  \expandafter
  \ifx\csname T@#1\endcsname\relax
    \def\cdp@elt{\noexpand\cdp@elt}%
    \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
                   {\default@k@family}{\default@k@series}%
                   {\default@k@shape}}%
    \expandafter\let\csname#1-cmd\endcsname\@changed@kcmd
    \def\enc@elt{\noexpand\enc@elt}%
    \xdef\ktenc@list{\ktenc@list\enc@elt<#1>}%
    \xdef\kenc@list{\kenc@list\enc@elt<#1>}%
  \else
    \@font@info{Redeclaring KANJI (tate) font encoding #1}%
  \fi
  \global\@namedef{T@#1}{#2}%
  \global\@namedef{M@#1}{\default@KM#3}%
  }
%
\@onlypreamble\DeclareKanjiEncoding
```

```
\@onlypreamble\DeclareYokoKanjiEncoding
\@onlypreamble\DeclareYokoKanjiEncoding@
\@onlypreamble\DeclareTateKanjiEncoding
\@onlypreamble\DeclareTateKanjiEncoding@
```

\DeclareKanjiEncodingDefaults 和文エンコードのデフォルト値を宣言するコマンドです。

```
\def\DeclareKanjiEncodingDefaults#1#2{%
  \ifx\relax#1\else
    \ifx\default@KT\@empty\else
      \@font@info{Overwriting KANJI encoding scheme text defaults}%
    \fi
    \gdef\default@KT{#1}%
  \fi
  \ifx\relax#2\else
    \ifx\default@KM\@empty\else
      \@font@info{Overwriting KANJI encoding scheme math defaults}%
    \fi
    \gdef\default@KM{#2}%
  \fi}
\let\default@KT\@empty
\let\default@KM\@empty
\@onlypreamble\DeclareKanjiEncodingDefaults
```

\DeclareFontFamily 欧文ファミリを宣言するためのコマンドです。\ffam@list を作るように再定義をします。

```
\def\DeclareFontFamily#1#2#3{%
 \@ifundefined{T@#1}%
    {\@latex@error{Encoding scheme `#1' unknown}\@eha}%
    {\edef\tmp@item{{#2}}%
     \expandafter\expandafter\expandafter
     \inlist@\expandafter\tmp@item\expandafter{\ffam@list}%
     \ifin@ \else
        \def\fam@elt{\noexpand\fam@elt}%
        \xdef\ffam@list{\ffam@list\fam@elt<#2>}%
     \fi
     \def\reserved@a{#3}%
     \global
     \expandafter\let\csname #1+#2\expandafter\endcsname
            \ifx \reserved@a\@empty
              \@empty
            \else \reserved@a
            \fi
    }%
}
```

\DeclareKanjiFamily 欧文ファミリを宣言するためのコマンドです。

```
\def\DeclareKanjiFamily#1#2#3{%
 \@ifundefined{T@#1}%
    {\@latex@error{KANJI Encoding scheme `#1' unknown}\@eha}%
    {\edef\tmp@item{{#2}}%
     \expandafter\expandafter\expandafter
     \inlist@\expandafter\tmp@item\expandafter{\kfam@list}%
     \ifin@ \else
        \def\fam@elt{\noexpand\fam@elt}%
        \xdef\kfam@list{\kfam@list\fam@elt<#2>}%
     \fi
     \def\reserved@a{#3}%
     \global
```

```
    \expandafter\let\csname #1+#2\expandafter\endcsname
          \ifx \reserved@a\@empty
            \@empty
          \else \reserved@a
          \fi
    }%
}
```

\DeclareKanjiSubstitution
\DeclareErrorKanjiFont

目的の和文フォントが見つからなかったときに使うフォントの宣言をするコマンドです。それぞれ、\DeclareFontSubstitution と\DeclareErrorFont に対応します。

```
\def\DeclareKanjiSubstitution#1#2#3#4{%
  \expandafter\ifx\csname T@#1\endcsname\relax
    \@latex@error{KANJI Encoding scheme `#1' unknown}\@eha
  \else
    \begingroup
       \def\reserved@a{#1}%
       \toks@{}%
       \def\cdp@elt##1##2##3##4{%
         \def\reserved@b{##1}%
         \ifx\reserved@a\reserved@b
           \addto@hook\toks@{\cdp@elt{#1}{#2}{#3}{#4}}%
         \else
           \addto@hook\toks@{\cdp@elt{##1}{##2}{##3}{##4}}%
         \fi}%
       \cdp@list
       \xdef\cdp@list{\the\toks@}%
    \endgroup
    \global\@namedef{D@#1}{\def\default@family{#2}%
                           \def\default@series{#3}%
                           \def\default@shape{#4}}%
  \fi}
%
\def\DeclareErrorKanjiFont#1#2#3#4#5{%
   \xdef\error@kfontshape{%
      \noexpand\expandafter\noexpand\split@name\noexpand\string
      \expandafter\noexpand\csname#1/#2/#3/#4/#5\endcsname
      \noexpand\@nil}%
   \gdef\default@k@family{#2}%
   \gdef\default@k@series{#3}%
   \gdef\default@k@shape{#4}%
   \global\let\k@family\default@k@family
   \global\let\k@series\default@k@series
   \global\let\k@shape\default@k@shape
   \gdef\f@size{#5}%
   \gdef\f@baselineskip{#5pt}}
%
\@onlypreamble\DeclareKanjiSubstitution
\@onlypreamble\DeclareErrorKanjiFont
```

\DeclareFixedFont

フォント名を宣言するコマンドです。

```
\def\DeclareFixedFont#1#2#3#4#5#6{%
   \begingroup
      \let\afont\font
      \math@fontsfalse
      \every@math@size{}%
      \fontsize{#6}\z@
      \edef\tmp@item{{#2}}%
      \expandafter\expandafter\expandafter
```

```
      \inlist@\expandafter\tmp@item\expandafter{\kyenc@list}%
      \ifin@
        \usekanji{#2}{#3}{#4}{#5}%
        \let\font\jfont
      \else
        \expandafter\expandafter\expandafter
        \inlist@\expandafter\tmp@item\expandafter{\ktenc@list}%
        \ifin@
          \usekanji{#2}{#3}{#4}{#5}%
          \let\font\tfont
        \else
          \useroman{#2}{#3}{#4}{#5}%
          \let\font\afont
        \fi
      \fi
      \global\expandafter\let\expandafter#1\the\font
      \let\font\afont
   \endgroup
 }
```

\reDeclareMathAlphabet 数式モード内で、数式文字用の和欧文フォントを同時に切り替えるコマンドです。

pLaTeX 2εには、本来の動作モードと 2.09 互換モードの二つがあり、両モードで数式文字を変更するコマンドや動作が異なります。本来の動作モードでは、\mathrm{...}のように\math??に引数を指定して使います。このときは引数にだけ影響します。2.09 互換モードでは、\rm のような二文字コマンドを使います。このコマンドには引数を取らず、書体はグルーピングの範囲で反映されます。二文字コマンドは、ネイティブモードでも使えるようになっていて、動作も 2.09 互換モードのコマンドと同じです。

しかし、内部的には\math??という一つのコマンドがすべての動作を受け持ち、\math??コマンドや\??コマンドから呼び出された状態に応じて、動作を変えています。したがって、欧文フォントと和文フォントの両方を一度に変更する、数式文字変更コマンドを作るとき、それぞれの状態に合った動作で動くようにフォント切り替えコマンドを実行させる必要があります。

使い方

```
usage: \reDeclareMathAlphabet{\mathAA}{\mathBB}{\mathCC}
```

欧文・和文両用の数式文字変更コマンド \mathAA を (再) 定義します。欧文用のコマンド \mathBB と、和文用の \mathCC を (p)LaTeX 標準の方法で定義しておいた後、上のように記述します。なお、{\mathBB}{\mathCC} の部分については {\@mathBB}{\@mathCC} のように @ をつけた記述をしてもかまいません (互換性のため)。上のような命令を発行すると、\mathAA が、欧文に対しては \mathBB、 和文に対しては \mathCC の意味を持つようになります。通常は、\reDeclareMathAlphabet{\mathrm}{\mathrm}{\mathmc} のようにAA=BB として用います。また、\mathrm は LaTeX kernel において標準のコマンドとして既に定義されているので、この場合は \mathrm の再定義となります。native mode での \rm のような two letter command (old font command) に対しても同様なことが引きおこります。つまり、数式モードにおいて、新たな \rm は、LaTeX original の \rm と \mc (正確に言えば \mathrm と \mathmc であるが) の意味を合わせ持つようになります。

補足

- \mathAA を再定義する他の命令 (\DeclareSymbolFontAlphabet を用いるパッケージの使用等) との衝突を避けるためには、\AtBeginDocument を併用するなどして展開位置の制御を行ってください。

- テキストモード時のエラー表示用に \mathBB のみを用いることを除いて、\mathBB と \mathCC の順は実際には意味を持ちません。和文、欧文の順に定義しても問題はありません。

- 第 2,3 引き数には {\@mathBB}{\@mathCC} のように @ をつけた記述も行えます。ただし、形式は統一してください。判断は第 2 引き数で行っているため、 {\@mathBB}{\mathCC} のような記述ではうまく動作しません。また、\makeatletter な状態で {\@mathBB }{\@mathCC } のような @ と余分なスペースをつけた場合には無限ループを引き起こすことがあります。このような記述は避けるようにして下さい。

- \reDeclareMathAlphabet を実行する際には、\mathBB, \mathCC が定義されている必要はありません。実際に \mathAA を用いる際にはこれらの\mathBB, \mathCC が (p)LaTeX 標準の方法で定義されている必要があります。

- 他の部分で \mathAA を全く定義しない場合を除き、\mathAA は\reDeclareMathAlphabet を実行する以前で (p)LaTeX 標準の方法で定義されている必要があります (\mathrm や \mathbf の標準的なコマンドは、LaTeX kernel で既に定義されています)。\DeclareMathAlphabet の場合には、\reDeclareMathAlphabet よりも前で 1 度\mathAA を定義してあれば、\reDeclareMathAlphabet の後ろで再度\DeclareMathAlphabet を用いて \mathAA の内部の定義内容を変更することには問題ありません。\DeclareSymbolFontAlphabet の場合、再定義においても \mathAA が直接定義されるので、\mathAA に対する最後の \DeclareSymbolFontAlphabet のさらに後で \reDeclareMathAlphabet を実行しなければ有効とはなりません。

- \documentstyle の互換モードの場合、\rm 等の two letter command (old font command) は、\reDeclareMathAlphabet とは関連することのない別個のコマンドとして定義されます。従って、この場合には\reDeclareMathAlphabet を用いても \rm 等は数式モードにおいて欧文･和文両用のものとはなりません。

```
\def\reDeclareMathAlphabet#1#2#3{%
  \edef#1{\noexpand\protect\expandafter\noexpand\csname%
    \expandafter\@gobble\string#1\space\space\endcsname}%
  \edef\@tempa{\expandafter\@gobble\string#2}%
  \edef\@tempb{\expandafter\@gobble\string#3}%
  \edef\@tempc{\string @\expandafter\@gobbletwo\string#2}%
  \ifx\@tempc\@tempa%
    \edef\@tempa{\expandafter\@gobbletwo\string#2}%
    \edef\@tempb{\expandafter\@gobbletwo\string#3}%
  \fi
  \expandafter\edef\csname\expandafter\@gobble\string#1\space\space\endcsname%
    {\noexpand\DualLang@mathalph@bet%
      {\expandafter\noexpand\csname\@tempa\space\endcsname}%
      {\expandafter\noexpand\csname\@tempb\space\endcsname}%
```

```
  }%
}
\@onlypreamble\reDeclareMathAlphabet
\def\DualLang@mathalph@bet#1#2{%
  \relax\ifmmode
    \ifx\math@bgroup\bgroup%     2e normal style     (\mathrm{...})
      \bgroup\let\DualLang@Mfontsw\DLMfontsw@standard
    \else
      \ifx\math@bgroup\relax%    2e two letter style (\rm->\mathrm)
        \let\DualLang@Mfontsw\DLMfontsw@oldstyle
      \else
        \ifx\math@bgroup\@empty% 2.09 oldlfont style ({\mathrm ...})
          \let\DualLang@Mfontsw\DLMfontsw@oldlfont
        \else%                   panic! assume 2e normal style
          \bgroup\let\DualLang@Mfontsw\DLMfontsw@standard
        \fi
      \fi
    \fi
  \else
    \let\DualLang@Mfontsw\@firstoftwo
  \fi
  \DualLang@Mfontsw{#1}{#2}%
}
\def\DLMfontsw@standard#1#2#3{#1{#2{#3}}\egroup}
\def\DLMfontsw@oldstyle#1#2{#1\relax\@fontswitch\relax{#2}}
\def\DLMfontsw@oldlfont#1#2{#1\relax#2\relax}
```

\DeclareRelationFont
\SetRelationFont

和文書体に対する従属書体を宣言するコマンドです。**従属書体**とは、ある和文書体とペアになる欧文書体のことです。主に多書体パッケージskfonts を用いるための仕組みです。

\DeclareRelationFont コマンドの最初の 4 つの引数の組が和文書体の属性、その後の 4 つの引数の組が従属書体の属性です。

```
\DeclareRelationFont{JY1}{mc}{m}{n}{OT1}{cmr}{m}{n}
\DeclareRelationFont{JY1}{gt}{m}{n}{OT1}{cmr}{bx}{n}
```

上記の例は、明朝体の従属書体としてコンピュータモダンローマン、ゴシック体の従属書体としてコンピュータモダンボールドを宣言しています。カレント和文書体が\JY1/mc/m/n となると、自動的に欧文書体が\OT1/cmr/m/n になります。また、和文書体が\JY1/gt/m/n になったときは、欧文書体が\OT1/cmr/bx/n になります。

和文書体のシェイプ指定を省略するとエンコード／ファミリ／シリーズの組合せで従属書体が使われます。このときは、\selectfont が呼び出された時点でのシェイプ（\f@shape）の値が使われます。

\DeclareRelationFont の設定値はグローバルに有効です。\SetRelationFont の設定値はローカルに有効です。フォント定義ファイルで宣言をする場合は、\DeclareRelationFont を使ってください。

```
\def\all@shape{all}%
\def\DeclareRelationFont#1#2#3#4#5#6#7#8{%
  \def\rel@shape{#4}%
  \ifx\rel@shape\@empty
     \global
     \expandafter\def\csname rel@#1/#2/#3/all\endcsname{%
       \romanencoding{#5}\romanfamily{#6}%
       \romanseries{#7}}%
```

```
  \else
     \global
     \expandafter\def\csname rel@#1/#2/#3/#4\endcsname{%
       \romanencoding{#5}\romanfamily{#6}%
       \romanseries{#7}\romanshape{#8}}%
  \fi
}
\def\SetRelationFont#1#2#3#4#5#6#7#8{%
  \def\rel@shape{#4}%
  \ifx\rel@shape\@empty
     \expandafter\def\csname rel@#1/#2/#3/all\endcsname{%
       \romanencoding{#5}\romanfamily{#6}%
       \romanseries{#7}}%
  \else
     \expandafter\def\csname rel@#1/#2/#3/#4\endcsname{%
       \romanencoding{#5}\romanfamily{#6}%
       \romanseries{#7}\romanshape{#8}}%
  \fi
}
```

\if@knjcmd
\userelfont

\if@knjcmd は欧文書体を従属書体にするかどうかのフラグです。このフラグが真になると、欧文書体に従属書体が使われます。このフラグは\userelfont コマンドによって、真となります。そして\selectfont 実行後には偽に初期化されます。

```
\newif\if@knjcmd
\def\userelfont{\@knjcmdtrue}
```

\selectfont

\selectfont のオリジナルからの変更部分は、次の 3 点です。

- 和文書体を変更する部分
- 従属書体に変更する部分
- 和欧文のベースラインを調整する部分

\selectfont コマンドは、まず、和文フォントを切り替えます。

```
⟨/plcore⟩
⟨*plcore | trace⟩
\DeclareRobustCommand\selectfont{%
  \let\tmp@error@fontshape\error@fontshape
  \let\error@fontshape\error@kfontshape
  \edef\tmp@item{{\k@encoding}}%
  \expandafter\expandafter\expandafter
  \inlist@\expandafter\tmp@item\expandafter{\kyenc@list}%
  \ifin@
    \let\cy@encoding\k@encoding
    \edef\ct@encoding{\csname t@enc@\k@encoding\endcsname}%
  \else
    \expandafter\expandafter\expandafter
    \inlist@\expandafter\tmp@item\expandafter{\ktenc@list}%
    \ifin@
      \let\ct@encoding\k@encoding
      \edef\cy@encoding{\csname y@enc@\k@encoding\endcsname}%
    \else
      \@latex@error{KANJI Encoding scheme `\k@encoding' unknown}\@eha
    \fi
  \fi
  \let\font\tfont
  \let\k@encoding\ct@encoding
```

```
  \xdef\font@name{\csname\curr@kfontshape/\f@size\endcsname}%
  \pickup@font
  \font@name
  \let\font\jfont
  \let\k@encoding\cy@encoding
  \xdef\font@name{\csname\curr@kfontshape/\f@size\endcsname}%
  \pickup@font
  \font@name
  \expandafter\def\expandafter\k@encoding\tmp@item
  \kenc@update
  \let\error@fontshape\tmp@error@fontshape
```

次に、\if@knjcmd が真の場合、欧文書体を現在の和文書体に関連付けされたフォントに変えます。このフラグは\userelfont コマンドによって真となります。このフラグはここで再び、偽に設定されます。

```
  \if@knjcmd \@knjcmdfalse
    \expandafter\ifx
    \csname rel@\k@encoding/\k@family/\k@series/\k@shape\endcsname\relax
      \expandafter\ifx
         \csname rel@\k@encoding/\k@family/\k@series/all\endcsname\relax
      \else
         \csname rel@\k@encoding/\k@family/\k@series/all\endcsname
      \fi
    \else
       \csname rel@\k@encoding/\k@family/\k@series/\k@shape\endcsname
    \fi
  \fi
```

そして、欧文フォントを切り替えます。

```
  \let\font\afont
  \xdef\font@name{\csname\curr@fontshape/\f@size\endcsname}%
  \pickup@font
  \font@name
⟨trace⟩\ifnum \tracingfonts>\tw@
⟨trace⟩   \@font@info{Roman:Switching to \font@name}\fi
  \enc@update
```

最後に、サイズが変更されていれば、ベースラインの調整などを行ないます。英語版の\selectfont では最初に行なっていますが、pLaTeX 2$_\varepsilon$ではベースラインシフトの調整をするために、書体を確定しなければならないため、一番最後に行ないます

```
  \ifx\f@linespread\baselinestretch \else
    \set@fontsize\baselinestretch\f@size\f@baselineskip
  \fi
  \size@update}
```

\KanjiEncodingPair 和文の縦横のエンコーディングはそれぞれ対にして扱うため、セット化します

```
\def\KanjiEncodingPair#1#2{\@namedef{t@enc@#1}{#2}\@namedef{y@enc@#2}{#1}}
\KanjiEncodingPair{JY1}{JT1}
```

\set@fontsize \fontsize コマンドの内部形式です。ベースラインの設定と、支柱の設定を行ないます。

```
\def\set@fontsize#1#2#3{%
    \@defaultunits\@tempdimb#2pt\relax\@nnil
    \edef\f@size{\strip@pt\@tempdimb}%
    \@defaultunits\@tempskipa#3pt\relax\@nnil
    \edef\f@baselineskip{\the\@tempskipa}%
    \edef\f@linespread{#1}%
```

```
    \let\baselinestretch\f@linespread
    \def\size@update{%
      \baselineskip\f@baselineskip\relax
      \baselineskip\f@linespread\baselineskip
      \normalbaselineskip\baselineskip
```

ここで、ベースラインシフトの調整と支柱を組み立てます。

```
      \adjustbaseline
      \setbox\strutbox\hbox{\yoko
          \vrule\@width\z@
                \@height.7\baselineskip \@depth.3\baselineskip}%
      \setbox\tstrutbox\hbox{\tate
          \vrule\@width\z@
                \@height.5\baselineskip \@depth.5\baselineskip}%
      \setbox\zstrutbox\hbox{\tate
          \vrule\@width\z@
                \@height.7\baselineskip \@depth.3\baselineskip}%
```

フォントサイズとベースラインに関する診断情報を出力します。

```
⟨*trace⟩
     \ifnum \tracingfonts>\tw@
       \ifx\f@linespread\@empty
         \let\reserved@a\@empty
       \else
         \def\reserved@a{\f@linespread x}%
       \fi
       \@font@info{Changing size to\space
             \f@size/\reserved@a \f@baselineskip}%
       \aftergroup\type@restoreinfo
     \fi
⟨/trace⟩
        \let\size@update\relax}}
```

\adjustbaseline 現在の和文フォントの空白（EUC コード 0xA1A1）の中央に現在の欧文フォントの “/” の中央がくるようにベースラインシフトを設定します。

当初はまずベースラインシフト量をゼロにしていましたが、\tbaselineshift を連続して変更した後に鉤括弧類を使うと余計なアキがでる問題が起こるため、\tbaselineshift をゼロクリアする処理を削除しました。

しかし、それではベースラインシフトを調整済みの欧文ボックスと比較してしまうため、計算した値が大きくなってしまいます。そこで、このボックスの中でゼロにするようにしました。また、“/” と比較していたのを “M” にしました。

```
\newbox\adjust@box
\newdimen\adjust@dimen
\def\adjustbaseline{%
```

和文フォントの基準値を設定します。

```
    \setbox\adjust@box\hbox{\char\euc"A1A1}%
    \cht\ht\adjust@box
    \cdp\dp\adjust@box
    \cwd\wd\adjust@box
    \cvs\normalbaselineskip
    \chs\cwd
    \cHT\cht \advance\cHT\cdp
```

基準となる欧文フォントの文字を含んだボックスを作成し、ベースラインシフト量の計算を行ないます。計算式は次のとおりです。

$$\text{ベースラインシフト量} \quad = \quad \{(\text{全角空白の深さ}) - (/\text{の深さ})\} \\ \_\frac{(\text{全角空白の高さ} + \text{深さ}) - (/\text{の高さ} + \text{深さ})}{2}$$

```
  \iftdir
    \setbox\adjust@box\hbox{\tbaselineshift\z@ M}%
    \adjust@dimen\ht\adjust@box
    \advance\adjust@dimen\dp\adjust@box
    \advance\adjust@dimen-\cHT
    \divide\adjust@dimen\tw@
    \advance\adjust@dimen\cdp
    \advance\adjust@dimen-\dp\adjust@box
    \tbaselineshift\adjust@dimen
⟨trace⟩   \ifnum \tracingfonts>\tw@
⟨trace⟩     \typeout{baselineshift:\the\tbaselineshift}
⟨trace⟩   \fi
  \fi}
⟨/plcore | trace⟩
⟨*plcore⟩
```

\romanencoding
\kanjiencoding
\fontencoding

書体のエンコードを指定するコマンドです。\fontencoding コマンドは和欧文のどちらかに影響します。\DeclareKanjiEncoding で指定されたエンコードは和文エンコードとして、\DeclareFontEncoding で指定されたエンコードは欧文エンコードとして認識されます。

\kanjiencoding と\romanencoding は与えられた引数が、エンコードとして登録されているかどうかだけを確認し、それが和文か欧文かのチェックは行なっていません。そのため、高速に動作をしますが、\kanjiencoding に欧文エンコードを指定したり、逆に\romanencoding に和文エンコードを指定した場合はエラーとなります。

```
\DeclareRobustCommand\romanencoding[1]{%
    \expandafter\ifx\csname T@#1\endcsname\relax
      \@latex@error{Encoding scheme ‘#1’ unknown}\@eha
    \else
      \edef\f@encoding{#1}%
      \ifx\cf@encoding\f@encoding
        \let\enc@update\relax
      \else
        \let\enc@update\@@enc@update
      \fi
    \fi
}
\DeclareRobustCommand\kanjiencoding[1]{%
    \expandafter\ifx\csname T@#1\endcsname\relax
      \@latex@error{KANJI Encoding scheme ‘#1’ unknown}\@eha
    \else
      \edef\k@encoding{#1}%
      \ifx\ck@encoding\k@encoding
         \let\kenc@update\relax
      \else
         \let\kenc@update\@@kenc@update
      \fi
    \fi
}
\DeclareRobustCommand\fontencoding[1]{%
```

```
469   \edef\tmp@item{{#1}}%
470   \expandafter\expandafter\expandafter
471   \inlist@\expandafter\tmp@item\expandafter{\kenc@list}%
472   \ifin@ \kanjiencoding{#1}\else\romanencoding{#1}\fi}
```

\@@kenc@update

\kanjiencoding コマンドのコードからもわかるように、\ck@encoding と\k@encoding が異なる場合、\kenc@update コマンドは\@@kenc@update コマンドと等しくなります。

\@@kenc@update コマンドは、そのエンコードでのデフォルト値を設定するためのコマンドです。欧文用の\@@enc@update コマンドでは、474 行目と 475 行目のような代入もしていますが、和文用にはコメントにしてあります。これらは\DeclareTextCommand や\ProvideTextCommand などでエンコードごとに設定されるコマンドを使うための仕組みです。しかし、和文エンコードに依存するようなコマンドやマクロを作成することは、現時点では、ないと思います。

```
473 \def\@@kenc@update{%
474 %  \expandafter\let\csname\ck@encoding -cmd\endcsname\@changed@kcmd
475 %  \expandafter\let\csname\k@encoding-cmd\endcsname\@current@cmd
476   \default@KT
477   \csname T@\k@encoding\endcsname
478   \csname D@\k@encoding\endcsname
479   \let\kenc@update\relax
480   \let\ck@encoding\k@encoding
481   \edef\tmp@item{{\k@encoding}}%
482   \expandafter\expandafter\expandafter
483   \inlist@\expandafter\tmp@item\expandafter{\kyenc@list}%
484   \ifin@ \let\cy@encoding\k@encoding
485   \else
486     \expandafter\expandafter\expandafter
487     \inlist@\expandafter\tmp@item\expandafter{\ktenc@list}%
488     \ifin@ \let\ct@encoding\k@encoding
489     \else
490       \@latex@error{KANJI Encoding scheme `\k@encoding' unknown}\@eha
491     \fi
492   \fi
493 }
494 \let\kenc@update\relax
```

\@changed@cmd の和文エンコーディングバージョン。

```
495 \def\@changed@kcmd#1#2{%
496    \ifx\protect\@typeset@protect
497       \@inmathwarn#1%
498       \expandafter\ifx\csname\ck@encoding\string#1\endcsname\relax
499          \expandafter\ifx\csname ?\string#1\endcsname\relax
500             \expandafter\def\csname ?\string#1\endcsname{%
501                \TextSymbolUnavailable#1%
502             }%
503          \fi
504          \global\expandafter\let
505                \csname\cf@encoding \string#1\expandafter\endcsname
506                \csname ?\string#1\endcsname
507       \fi
508       \csname\ck@encoding\string#1%
509          \expandafter\endcsname
510    \else
511       \noexpand#1%
512    \fi}
```

\@notkfam
\@notffam

\fontfamily コマンド内で使用するフラグです。@notkfam フラグは和文ファミリでなかったことを、@notffam フラグは欧文ファミリでなかったことを示します。

```
\newif\if@notkfam
\newif\if@notffam

\newif\if@tempswz
```

\romanfamily
\kanjifamily
\fontfamily

書体のファミリを指定するコマンドです。

\kanjifamily と\romanfamily は与えられた引数が、和文あるいは欧文のファミリとして正しいかのチェックは行なっていません。そのため、高速に動作をしますが、\kanjifamily に欧文ファミリを指定したり、逆に\romanfamily に和文ファミリを指定した場合は、エラーとなり、代用フォントかエラーフォントが使われます。

```
\DeclareRobustCommand\romanfamily[1]{\edef\f@family{#1}}
\DeclareRobustCommand\kanjifamily[1]{\edef\k@family{#1}}
```

\fontfamily は、指定された値によって、和文ファミリか欧文ファミリ、**あるいは両方**のファミリを切り替えます。和欧文ともに無効なファミリ名が指定された場合は、和欧文ともに代替書体が使用されます。

引数が\rmfamily のような名前で与えられる可能性があるため、まず、これを展開したものを作ります。

また、和文ファミリと欧文ファミリのそれぞれになかったことを示すフラグを偽にセットします。

```
\DeclareRobustCommand\fontfamily[1]{%
  \edef\tmp@item{{#1}}%
  \@notkfamfalse
  \@notffamfalse
```

次に、この引数が\kfam@list に登録されているかどうかを調べます。登録されていれば、\k@family にその値を入れます。

```
  \expandafter\expandafter\expandafter
  \inlist@\expandafter\tmp@item\expandafter{\kfam@list}%
  \ifin@ \edef\k@family{#1}%
```

そうでないときは、\notkfam@list に登録されているかどうかを調べます。登録されていれば、この引数は和文ファミリではありませんので、\@notkfam フラグを真にして、欧文ファミリのルーチンに移ります。

このとき、\efam@list を調べるのではないことに注意をしてください。\efam@list を調べ、これにないファミリを和文ファミリであるとすると、たとえば、欧文ナールファミリが定義されているけれども、和文ナールファミリが未定義の場合、\fontfamily{nar}という指定は、nar が\efam@list にだけ、登録されているため、和文書体をナールにすることができません。

逆に、\kfam@list に登録されていないからといって、\k@family にnar を設定すると、cmr のようなファミリも\k@family に設定される可能性があります。したがって、「欧文でない」を明示的に示す\notkfam@list を見る必要があります。

```
  \else
    \expandafter\expandafter\expandafter
    \inlist@\expandafter\tmp@item\expandafter{\notkfam@list}%
    \ifin@ \@notkfamtrue
```

\notkfam@list に登録されていない場合は、フォント定義ファイルが存在するかどうかを調べます。ファイルが存在する場合は、\k@family を変更します。ファイルが存在しない場合は、\notkfam@list に登録します。

\kenc@list に登録されているエンコードと、指定された和文ファミリの組合せのフォント定義ファイルが存在する場合は、\k@family に指定された値を入れます。

```
    \else
      \@tempswzfalse
      \def\fam@elt{\noexpand\fam@elt}%
      \message{(I search kanjifont definition file:}%
      \def\enc@elt<##1>{\message{.}%
        \edef\reserved@a{\lowercase{\noexpand\IfFileExists{##1#1.fd}}}%
        \reserved@a{\@tempswztrue}{}\relax}%
      \kenc@list
      \message{)}%
      \if@tempswz
        \edef\k@family{#1}%
```

つぎの部分が実行されるのは、和文ファミリとして認識できなかった場合です。この場合は、\@notkfam フラグを真にして、\notkfam@list に登録します。

```
      \else
        \@notkfamtrue
        \xdef\notkfam@list{\notkfam@list\fam@elt<#1>}%
      \fi
```

\kfam@list と\notkfam@list に登録されているかどうかを調べた\ifin@を閉じます。

```
  \fi\fi
```

欧文ファミリの場合も、和文ファミリと同様の方法で確認をします。

```
  \expandafter\expandafter\expandafter
  \inlist@\expandafter\tmp@item\expandafter{\ffam@list}%
  \ifin@ \edef\f@family{#1}\else
    \expandafter\expandafter\expandafter
    \inlist@\expandafter\tmp@item\expandafter{\notffam@list}%
    \ifin@ \@notffamtrue \else
      \@tempswzfalse
      \def\fam@elt{\noexpand\fam@elt}%
      \message{(I search font definition file:}%
      \def\enc@elt<##1>{\message{.}%
        \edef\reserved@a{\lowercase{\noexpand\IfFileExists{##1#1.fd}}}%
        \reserved@a{\@tempswztrue}{}\relax}%
      \fenc@list
      \message{)}%
      \if@tempswz
        \edef\f@family{#1}%
      \else
        \@notffamtrue
        \xdef\notffam@list{\notffam@list\fam@elt<#1>}%
      \fi
  \fi\fi
```

最後に、指定された文字列が、和文ファミリと欧文ファミリのいずれか、あるいは両方として認識されたかどうかを確認します。

どちらとも認識されていない場合は、ファミリの指定ミスですので、代用フォントを使うために、故意に指定された文字列をファミリに入れます。

```
  \if@notkfam\if@notffam
```

```
      \edef\k@family{#1}\edef\f@family{#1}%
  \fi\fi}
```

\romanseries
\kanjiseries
\fontseries

書体のシリーズを指定するコマンドです。\fontseries コマンドは和欧文の両方に影響します。

```
\DeclareRobustCommand\romanseries[1]{\edef\f@series{#1}}
\DeclareRobustCommand\kanjiseries[1]{\edef\k@series{#1}}
\DeclareRobustCommand\fontseries[1]{\kanjiseries{#1}\romanseries{#1}}
```

\romanshape
\kanjishape
\fontshape

書体のシェイプを指定するコマンドです。\fontshape コマンドは和欧文の両方に影響します。

```
\DeclareRobustCommand\romanshape[1]{\edef\f@shape{#1}}
\DeclareRobustCommand\kanjishape[1]{\edef\k@shape{#1}}
\DeclareRobustCommand\fontshape[1]{\kanjishape{#1}\romanshape{#1}}
```

\usekanji
\useroman
\usefont

書体属性を一度に指定するコマンドです。和文書体には\usekanji を、欧文書体には\useroman を指定してください。

\usefont コマンドは、第一引数で指定されるエンコードによって、和文または欧文フォントを切り替えます。

```
\def\usekanji#1#2#3#4{%
    \kanjiencoding{#1}\kanjifamily{#2}\kanjiseries{#3}\kanjishape{#4}%
    \selectfont\ignorespaces}
\def\useroman#1#2#3#4{%
    \romanencoding{#1}\romanfamily{#2}\romanseries{#3}\romanshape{#4}%
    \selectfont\ignorespaces}
\def\usefont#1#2#3#4{%
  \edef\tmp@item{{#1}}%
  \expandafter\expandafter\expandafter
  \inlist@\expandafter\tmp@item\expandafter{\kenc@list}%
  \ifin@ \usekanji{#1}{#2}{#3}{#4}%
  \else\useroman{#1}{#2}{#3}{#4}%
  \fi}
```

\normalfont

書体をデフォルト値にするコマンドです。和文書体もデフォルト値になるように再定義しています。ただし高速化のため、\usekanji と\useroman を展開し、\selectfont を一度しか呼び出さないようにしています。

```
\DeclareRobustCommand\normalfont{%
    \kanjiencoding{\kanjiencodingdefault}%
    \kanjifamily{\kanjifamilydefault}%
    \kanjiseries{\kanjiseriesdefault}%
    \kanjishape{\kanjishapedefault}%
    \romanencoding{\encodingdefault}%
    \romanfamily{\familydefault}%
    \romanseries{\seriesdefault}%
    \romanshape{\shapedefault}%
    \selectfont\ignorespaces}
\adjustbaseline
\let\reset@font\normalfont
```

\mcfamily
\gtfamily

和文書体を明朝体にする\mcfamily とゴシック体にする\gtfamily を定義します。これらは、\rmfamily などに対応します。\mathmc と\mathgt は数式内で用いるときのコマンド名です。

```
\DeclareRobustCommand\mcfamily
        {\not@math@alphabet\mcfamily\mathmc
```

```
602          \kanjifamily\mcdefault\selectfont}
603 \DeclareRobustCommand\gtfamily
604         {\not@math@alphabet\gtfamily\mathgt
605          \kanjifamily\gtdefault\selectfont}
```

\romanprocess@table
\kanjiprocess@table
\process@table

文書の先頭で、和文デフォルトフォントの変更が反映されないのを修正します。

```
606 \let\romanprocess@table\process@table
607 \def\kanjiprocess@table{%
608   \kanjiencoding{\kanjiencodingdefault}%
609   \kanjifamily{\kanjifamilydefault}%
610   \kanjiseries{\kanjiseriesdefault}%
611   \kanjishape{\kanjishapedefault}%
612 }
613 \def\process@table{%
614   \romanprocess@table
615   \kanjiprocess@table
616 }
617 \@onlypreamble\romanprocess@table
618 \@onlypreamble\kanjiprocess@table
```

\textunderscore

このコマンドはテキストモードで指定された_の内部コマンドです。縦組での位置を調整するように再定義をします。もとは ltoutenc.dtx で定義されています。

なお、_を数式モードで使うと\mathunderscore が実行されます。

```
619 \DeclareTextCommandDefault{\textunderscore}{%
620   \leavevmode\kern.06em
621   \iftdir\raise-\tbaselineshift\fi
622   \vbox{\hrule\@width.3em}}
```

2.3 デフォルト設定ファイルの読み込み

最後に、デフォルト設定ファイルである、pldefs.ltx を読み込みます。このファイルについての詳細は、第 3 節を参照してください。TEX の入力ファイル検索パスに設定されているディレクトリに pldefs.cfg ファイルがある場合は、そのファイルを使います。

```
623 \InputIfFileExists{pldefs.cfg}
624           {\typeout{*************************************^^J%
625                     * Local config file pldefs.cfg used^^J%
626                     *************************************}}%
627           {\input{pldefs.ltx}}
628 ⟨/plcore⟩
```

3 デフォルト設定ファイル

ここでは、フォーマットファイルに読み込まれるデフォルト値を設定しています。この節での内容は pldefs.ltx に出力されます。このファイルの内容を plcore.ltx に含めてもよいのですが、デフォルトの設定を参照しやすいように、別ファイルにしてあります。pldefs.ltx は plcore.ltx から読み込まれます。

プリロードサイズは、DOCSTRIP プログラムのオプションで変更することができます。これ以外の設定を変更したい場合は、pldefs.ltx を直接、修正するのではなく、このファイルを pldefs.cfg という名前でコピーをして、そのファイルに対して修正を加えるようにしてください。

```
629 ⟨*pldefs⟩
```

```
\ProvidesFile{pldefs.ltx}
      [2000/07/13 v1.2 pLaTeX Kernel (Default settings)]
```

3.1 イタリック補正

\check@nocorr@ 「あ \texttt{abc}い」としたとき、書体の変更を指定された欧文の左側に和欧文間スペースが入らないのを修正します。

```
\def \check@nocorr@ #1#2\nocorr#3\@nil {%
  \let \check@icl \relax% \maybe@ic から変更
  \def \check@icr {\ifvmode \else \aftergroup \maybe@ic \fi}%
  \def \reserved@a {\nocorr}%
  \def \reserved@b {#1}%
  \def \reserved@c {#3}%
  \ifx \reserved@a \reserved@b
    \ifx \reserved@c \@empty
      \let \check@icl \@empty
    \else
      \let \check@icl \@empty
      \let \check@icr \@empty
    \fi
  \else
    \ifx \reserved@c \@empty
    \else
      \let \check@icr \@empty
    \fi
  \fi
}
```

3.2 テキストフォント

テキストフォントのための属性やエラー書体などの宣言です。

縦横エンコード共通：

```
\DeclareKanjiEncodingDefaults{}{}
\DeclareErrorKanjiFont{JY1}{mc}{m}{n}{10}
```

横組エンコード：

```
\DeclareYokoKanjiEncoding{JY1}{}{}
\DeclareKanjiSubstitution{JY1}{mc}{m}{n}
```

縦組エンコード：

```
\DeclareTateKanjiEncoding{JT1}{}{}
\DeclareKanjiSubstitution{JT1}{mc}{m}{n}
```

フォント属性のデフォルト値：

```
\newcommand\mcdefault{mc}
\newcommand\gtdefault{gt}
\newcommand\kanjiencodingdefault{JY1}
\newcommand\kanjifamilydefault{\mcdefault}
\newcommand\kanjiseriesdefault{\mddefault}
\newcommand\kanjishapedefault{\updefault}
```

和文エンコードの指定：

```
\kanjiencoding{JY1}
```

フォント定義：これらの具体的な内容は第 4 節を参照してください。

```
\input{jy1mc.fd}
\input{jy1gt.fd}
\input{jt1mc.fd}
```

```
\input{jt1gt.fd}
```

フォントを有効にする

```
\fontencoding{JT1}\selectfont
\fontencoding{JY1}\selectfont
```

\textmc
\textgt

テキストファミリを切り替えるためのコマンドです。ltfntcmd.dtx で定義されている\textrm などに対応します。

```
\DeclareTextFontCommand{\textmc}{\mcfamily}
\DeclareTextFontCommand{\textgt}{\gtfamily}
```

\em
\emph

従来は\em, \emph で和文フォントの切り替えは行っていませんでしたが、和文フォントも\gtfamily に切り替えるようにしました。

```
\DeclareRobustCommand\em
        {\@nomath\em \ifdim \fontdimen\@ne\font >\z@
                        \mcfamily \upshape \else \gtfamily \itshape \fi}
```

3.3 プリロードフォント

あらかじめフォーマットファイルにロードされるフォントの宣言です。DOCSTRIP プログラムのオプションでロードされるフォントのサイズを変更することができます。platex.ins ではxpt を指定しています。

```
⟨*xpt⟩
\DeclarePreloadSizes{JY1}{mc}{m}{n}{5,7,10,12}
\DeclarePreloadSizes{JY1}{gt}{m}{n}{5,7,10,12}
\DeclarePreloadSizes{JT1}{mc}{m}{n}{5,7,10,12}
\DeclarePreloadSizes{JT1}{gt}{m}{n}{5,7,10,12}
⟨/xpt⟩
⟨*xipt⟩
\DeclarePreloadSizes{JY1}{mc}{m}{n}{5,7,10.95,12}
\DeclarePreloadSizes{JY1}{gt}{m}{n}{5,7,10.95,12}
\DeclarePreloadSizes{JT1}{mc}{m}{n}{5,7,10.95,12}
\DeclarePreloadSizes{JT1}{gt}{m}{n}{5,7,10.95,12}
⟨/xipt⟩
⟨*xiipt⟩
\DeclarePreloadSizes{JY1}{mc}{m}{n}{7,9,12,14.4}
\DeclarePreloadSizes{JY1}{gt}{m}{n}{7,9,12,14.4}
\DeclarePreloadSizes{JT1}{mc}{m}{n}{7,9,12,14.4}
\DeclarePreloadSizes{JT1}{gt}{m}{n}{7,9,12,14.4}
⟨/xiipt⟩
⟨*ori⟩
\DeclarePreloadSizes{JY1}{mc}{m}{n}
        {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74,24.88}
\DeclarePreloadSizes{JY1}{gt}{m}{n}
        {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74,24.88}
\DeclarePreloadSizes{JT1}{mc}{m}{n}
        {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74,24.88}
\DeclarePreloadSizes{JT1}{gt}{m}{n}
        {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74,24.88}
⟨/ori⟩
```

3.4 組版パラメータ

禁則パラメータや文字間へ挿入するスペースの設定などです。実際の各文字への禁則パラメータおよびスペースの挿入の許可設定などは、kinsoku.tex で行なってい

ます。具体的な設定については、kinsoku.dtx を参照してください。

```
\InputIfFileExists{kinsoku.tex}%
  {\message{Loading kinsoku patterns for japanese.}}
  {\errhelp{The configuration for kinsoku is incorrectly installed.^^J%
            If you don't understand this error message you need
            to seek^^Jexpert advice.}%
   \errmessage{OOPS! I can't find any kinsoku patterns for japanese^^J%
               \space Think of getting some or the
               platex2e setup will never succeed}\@@end}
```

組版パラメータの設定をします。\kanjiskip は、漢字と漢字の間に挿入されるグルーです。\noautospacing で、挿入を中止することができます。デフォルトは\autospacing です。

```
\kanjiskip=0pt plus .4pt minus .5pt
\autospacing
```

\xkanjiskip は、和欧文間に自動的に挿入されるグルーです。\noautoxspacing で、挿入を中止することができます。デフォルトは\autoxspacing です。

```
\xkanjiskip=.25zw plus1pt minus1pt
\autoxspacing
```

\jcharwidowpenalty は、パラグラフに対する禁則です。パラグラフの最後の行が 1 文字だけにならないように調整するために使われます。

```
\jcharwidowpenalty=500
```

最後に、\inhibitglue の簡略形を定義します。このコマンドは、和文フォントのメトリック情報から、自動的に挿入されるグルーの挿入を禁止します。

```
\def\<{\inhibitglue}
```

ここまでが、pldefs.ltx の内容です。

⟨/pldefs⟩

4 フォント定義ファイル

ここでは、フォント定義ファイルの設定をしています。フォント定義ファイルは、LaTeX のフォント属性を TeX フォントに置き換えるためのファイルです。記述方法についての詳細は、fntguide.tex を参照してください。

欧文書体の設定については、cmfonts.fdd や slides.fdd などを参照してください。skfonts.fdd には、写研代用書体を使うためのパッケージとフォント定義が記述されています。

```
⟨JY1mc⟩\ProvidesFile{jy1mc.fd}
⟨JY1gt⟩\ProvidesFile{jy1gt.fd}
⟨JT1mc⟩\ProvidesFile{jt1mc.fd}
⟨JT1gt⟩\ProvidesFile{jt1gt.fd}
⟨JY1mc, JY1gt, JT1mc, JT1gt⟩        [1997/01/24 v1.3 KANJI font defines]
```

横組用、縦組用ともに、明朝体のシリーズbx がゴシック体となるように宣言しています。

```
⟨*JY1mc⟩
\DeclareKanjiFamily{JY1}{mc}{}
\DeclareRelationFont{JY1}{mc}{m}{}{OT1}{cmr}{m}{}
\DeclareRelationFont{JY1}{mc}{bx}{}{OT1}{cmr}{bx}{}
\DeclareFontShape{JY1}{mc}{m}{n}{<5> <6> <7> <8> <9> <10> sgen*min
    <10.95><12><14.4><17.28><20.74><24.88> min10
```

```
730     <-> min10
731     }{}
732 \DeclareFontShape{JY1}{mc}{bx}{n}{<->ssub*gt/m/n}{}
733 ⟨/JY1mc⟩
734 ⟨*JT1mc⟩
735 \DeclareKanjiFamily{JT1}{mc}{}
736 \DeclareRelationFont{JT1}{mc}{m}{}{OT1}{cmr}{m}{}
737 \DeclareRelationFont{JT1}{mc}{bx}{}{OT1}{cmr}{bx}{}
738 \DeclareFontShape{JT1}{mc}{m}{n}{<5> <6> <7> <8> <9> <10> sgen*tmin
739     <10.95><12><14.4><17.28><20.74><24.88> tmin10
740     <-> tmin10
741     }{}
742 \DeclareFontShape{JT1}{mc}{bx}{n}{<->ssub*gt/m/n}{}
743 ⟨/JT1mc⟩
744 ⟨*JY1gt⟩
745 \DeclareKanjiFamily{JY1}{gt}{}
746 \DeclareRelationFont{JY1}{gt}{m}{}{OT1}{cmr}{bx}{}
747 \DeclareFontShape{JY1}{gt}{m}{n}{<5> <6> <7> <8> <9> <10> sgen*goth
748     <10.95><12><14.4><17.28><20.74><24.88> goth10
749     <-> goth10
750     }{}
751 \DeclareFontShape{JY1}{gt}{bx}{n}{<->ssub*gt/m/n}{}
752 ⟨/JY1gt⟩
753 ⟨*JT1gt⟩
754 \DeclareKanjiFamily{JT1}{gt}{}
755 \DeclareRelationFont{JT1}{gt}{m}{}{OT1}{cmr}{bx}{}
756 \DeclareFontShape{JT1}{gt}{m}{n}{<5> <6> <7> <8> <9> <10> sgen*tgoth
757     <10.95><12><14.4><17.28><20.74><24.88> tgoth10
758     <-> tgoth10
759     }{}
760 \DeclareFontShape{JT1}{gt}{bx}{n}{<->ssub*gt/m/n}{}
761 ⟨/JT1gt⟩
```

索 引

イタリック体の数字は、その項目が説明されているページを示しています。下線の引かれた数字は、定義されているページを示しています。その他の数字は、その項目が使われているページを示しています。

Symbols

\< 717
\@@enc@update 452
\@@end 711
\@@kenc@update 464, 473
\@changed@cmd 64
\@changed@kcmd 98, 122, 474, 495
\@current@cmd 475
\@defaultunits 386, 388
\@depth 399, 402, 405
\@eha . 156, 175, 194, 344, 446, 458, 490
\@firstoftwo 290
\@font@info 68, 103, 127, 141, 147, 377, 413
\@fontswitch 295
\@gobble 259, 260, 261, 267
\@gobbletwo 262, 264, 265
\@height 399, 402, 405
\@ifundefined 155, 174
\@inmathwarn 497
\@knjcmdfalse 360
\@knjcmdtrue 325
\@latex@error 156, 175, 194, 344, 446, 458, 490
\@latex@warning 75
\@namedef 70, 71, 105, 106, 129, 130, 209, 383
\@nil 218, 632
\@nnil 386, 388
\@nomath 674
\@notffam 513
\@notffamfalse 521
\@notffamtrue 550, 562
\@notkfam 513

\@notkfamfalse 520
\@notkfamtrue 528, 541
\@onlypreamble . 133, 134, 135, 136, 137, 153, 228, 229, 273, 617, 618
\@tempa 260, 263, 264, 269
\@tempb 261, 265, 270
\@tempc 262, 263
\@tempdimb 386, 387
\@tempskipa 388, 389
\@tempswzfalse 530, 551
\@tempswztrue 535, 556
\@typeset@protect 496
\@width 398, 401, 404, 622

A

\addto@hook 202, 204
\adjust@box 419, 422, 423, 424, 425, 430, 431, 432, 436
\adjust@dimen 420, 431, 432, 433, 434, 435, 436, 437
\adjustbaseline 396, 419, 598
\afont 22, 232, 250, 254, 372
\aftergroup 415, 634
\all@shape 297
\autospacing 713
\autoxspacing 715

B

\baselineskip 393, 394, 395, 399, 402, 405
\baselinestretch 379, 380, 391

C

\Cdp 13
\cdp 13, 424, 428, 435
\cdp@elt 60, 61, 94, 95, 118, 119, 199, 202, 204
\cdp@list 61, 95, 119, 206, 207
\cf@encoding 449, 505
\char 422
\check@icl 633, 640, 642
\check@icr 634, 643, 648
\check@nocorr@ 632
\Chs 19
\chs 19, 427
\Cht 11
\cHT 21, 428, 433
\cht 11, 423, 428
\ck@encoding . 1, 461, 474, 480, 498, 508
\ct@encoding 1, 336, 341, 348, 488
\curr@fontshape 373
\curr@kfontshape 9, 349, 354
\Cvs 17
\cvs 17, 426
\Cwd 15
\cwd 15, 425, 427
\cy@encoding 1, 335, 342, 353, 484

D

\DeclareErrorKanjiFont 192, 653
\DeclareFixedFont 230
\DeclareFontEncoding 51
\DeclareFontEncoding@ 51
\DeclareFontFamily 154
\DeclareFontShape 728, 732, 738, 742, 747, 751, 756, 760
\DeclareKanjiEncoding 74
\DeclareKanjiEncodingDefaults 138, 652
\DeclareKanjiFamily 173, 725, 735, 745, 754
\DeclareKanjiSubstitution 192, 655, 657
\DeclarePreloadSizes 677, 678, 679, 680, 683, 684, 685, 686, 689, 690, 691, 692, 695, 697, 699, 701
\DeclareRelationFont 297, 726, 727, 736, 737, 746, 755
\DeclareRobustCommand 328, 444, 456, 468, 516, 517, 518, 569, 570, 571, 572, 573, 574, 588, 600, 603, 673
\DeclareTateKanjiEncoding . . . 74, 656
\DeclareTateKanjiEncoding@ 74
\DeclareTextCommandDefault 619
\DeclareTextFontCommand . . . 671, 672
\DeclareYokoKanjiEncoding . . . 74, 654
\DeclareYokoKanjiEncoding@ 74
\default@family 62, 209
\default@k@family . . 96, 120, 219, 222
\default@k@series . . 96, 120, 220, 223
\default@k@shape . . . 97, 121, 221, 224
\default@KM . . . 106, 130, 146, 149, 152
\default@KT 140, 143, 151, 476
\default@M 71
\default@series 62, 210
\default@shape 63, 211
\DLMfontsw@oldlfont 283, 296
\DLMfontsw@oldstyle 280, 295
\DLMfontsw@standard . . . 277, 285, 294
\DualLang@mathalph@bet 268, 274
\DualLang@Mfontsw 277, 280, 283, 285, 290, 292

E

\em 673
\emph 673
\enc@elt 27, 29, 30, 65, 66, 99, 100, 101, 123, 124, 125, 533, 554
\enc@update 378, 450, 452
\encodingdefault 593
\errhelp 706
\errmessage 709
\error@fontshape 329, 330, 359
\error@kfontshape 215, 330
\euc 422
\every@math@size 234

F

\f@baselineskip 226, 380, 389, 393, 414

\f@encoding 10, 448, 449
\f@family 10, 516, 547, 560, 567
\f@linespread 379, 390, 391, 394, 408, 411
\f@series 10, 569
\f@shape 10, 572
\f@size 225, 349, 354, 373, 380, 387, 414
\fam@elt 27, 34, 35, 36, 161, 162, 180, 181, 531, 542, 552, 563
\familydefault 594
\fenc@list 29, 66, 557
\ffam@list 34, 159, 162, 546
\font 22, 232, 241, 247, 250, 253, 254, 347, 352, 372, 674
\font@name 349, 351, 354, 356, 373, 375, 377
\fontdimen 674
\fontencoding 444, 669, 670
\fontfamily 516
\fontseries 569
\fontshape 572
\fontsize 235

G

\gtdefault 605, 659
\gtfamily 600, 672, 675

H

\hrule 622

I

\if@knjcmd 324, 360
\if@notffam 514, 566
\if@notkfam 513, 566
\if@tempswz 515, 538, 559
\IfFileExists 534, 555
\ifin@ 160, 179, 239, 245, 334, 340, 472, 484, 488, 524, 528, 547, 550, 585
\iftdir 429, 621
\ifydir 42
\ignorespaces 577, 580, 597
\in@ 25, 26
\in@@ 24, 26
\in@false 25
\in@true 25
\inhibitglue 717
\inlist 23
\inlist@ 23, 159, 178, 238, 244, 333, 339, 471, 483, 487, 523, 527, 546, 549, 584
\input 627, 665, 666, 667, 668
\InputIfFileExists 623, 704
\itshape 675

J

\jcharwidowpenalty 716
\jfont 241, 352

K

\k@encoding 1, 9, 331, 335, 336, 341, 342, 344, 348, 353, 357, 362, 364, 366, 369, 460, 461, 475, 477, 478, 480, 481, 484, 488, 490
\k@family 6, 9, 222, 362, 364, 366, 369, 517, 524, 539, 567
\k@series 7, 9, 223, 362, 364, 366, 369, 570
\k@shape 8, 9, 224, 362, 369, 573
\kanjiencoding 444, 576, 589, 608, 664
\kanjiencodingdefault . 589, 608, 660
\KanjiEncodingPair 383
\kanjifamily 516, 576, 590, 602, 605, 609
\kanjifamilydefault . . . 590, 609, 661
\kanjiprocess@table 606
\kanjiseries 569, 576, 591, 610
\kanjiseriesdefault . . . 591, 610, 662
\kanjishape 572, 576, 592, 611
\kanjishapedefault 592, 611, 663
\kanjiskip 712
\kenc@list . 29, 101, 125, 471, 536, 584
\kenc@update . . 358, 462, 464, 479, 494
\kfam@list 34, 178, 181, 523
\ktenc@list 29, 124, 244, 339, 487
\kyenc@list 29, 100, 238, 333, 483

L

\LastDeclaredEncoding 72
\leavevmode 620
\lowercase 534, 555

M

\math@bgroup 276, 279, 282
\math@fontsfalse 233
\mathgt 604
\mathmc 601
\mathrm 276, 279, 282
\maybe@ic 633, 634
\mcdefault 602, 658, 661
\mcfamily 600, 671, 675
\mddefault 662
\MessageBreak 77, 79, 81

N

\newbox 39, 40, 419
\newdimen 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 420
\newif 324, 513, 514, 515
\nfss@catcodes 53, 87, 111
\nocorr 632, 635
\normalbaselineskip 395, 426
\normalfont 588
\not@math@alphabet 601, 604
\notffam@list 34, 549, 563
\notkfam@list 34, 527, 542

P

\pickup@font 350, 355, 374
\process@table 606

\protect 258, 496
\ProvidesFile . 630, 719, 720, 721, 722

R

\raise 621
\reDeclareMathAlphabet 257
\rel@fontshape 10
\rel@shape 299, 300, 313, 314
\reserved@a 164, 167, 169, 183, 186, 188, 197, 201, 409, 411, 414, 534, 535, 555, 556, 635, 638
\reserved@b 200, 201, 636, 638
\reserved@c 637, 639, 646
\reset@font 599
\rm 279
\romanencoding 303, 308, 316, 320, 444, 579, 593
\romanfamily 303, 308, 316, 320, 516, 579, 594
\romanprocess@table 606
\romanseries 304, 309, 317, 321, 569, 579, 595
\romanshape . . . 309, 321, 572, 579, 596

S

\selectfont 326, 577, 580, 597, 602, 605, 669, 670
\seriesdefault 595
\set@fontsize 380, 385
\SetRelationFont 297
\shapedefault 596
\size@update 382, 392, 418
\split@name 216
\strip@pt 387
\strut 41
\strutbox 43, 397

T

\tate 47, 49, 400, 403
\tbaselineshift . . . 430, 437, 439, 621
\textgt 671
\textmc 671
\TextSymbolUnavailable 501
\textunderscore 619
\tfont 247, 347
\tmp@error@fontshape 329, 359
\tmp@item 157, 159, 176, 178, 236, 238, 244, 331, 333, 339, 357, 469, 471, 481, 483, 487, 519, 523, 527, 546, 549, 582, 584
\toks@ 198, 202, 204, 207
\tracingfonts 376, 407, 438
\tstrut 41
\tstrutbox 39, 45, 48, 400
\type@restoreinfo 415
\typeout 439, 624

U

\unhcopy 43, 45, 48, 50
\updefault 663
\upshape 675
\usefont 575
\usekanji 240, 246, 575
\userelfont 324
\useroman 249, 575

V

\vrule 398, 401, 404

X

\xkanjiskip 714

Y

\yoko 397

Z

\zstrut 41
\zstrutbox 39, 50, 403