

# pxrubrica パッケージ

八登 崇之 (Takayuki YATO; aka “ZR”)

v1.3a [2017/05/05]

## 目次

1	パッケージ読込	1
2	ルビ機能	1
2.1	用語集	1
2.2	ルビ用命令	1
2.3	入力文字列のグループの指定	4
2.4	ゴースト処理	4
2.5	パラメタ設定命令	5
3	圏点機能	7
3.1	圏点用命令	7
3.2	圏点命令の親文字列	7
3.3	ゴースト処理	8
3.4	パラメタ設定命令	8
4	実装 (ルビ関連)	10
4.1	前提パッケージ	10
4.2	エラーメッセージ	10
4.3	パラメタ	12
4.3.1	全般設定	12
4.3.2	呼出時パラメタ・変数	14
4.4	その他の変数	16
4.5	補助手続	16
4.5.1	雑多な定義	16
4.5.2	数値計算	19
4.5.3	リスト分解	21
4.6	エンジン依存処理	25
4.7	パラメタ設定公開命令	36
4.8	ルビオプション解析	39

4.9	オプション整合性検査 . . . . .	45
4.10	フォントサイズ . . . . .	47
4.11	ルビ用均等割り . . . . .	49
4.12	小書き仮名の変換 . . . . .	52
4.13	ブロック毎の組版 . . . . .	53
4.14	命令の頑強化 . . . . .	60
4.15	致命的エラー対策 . . . . .	61
4.16	先読み処理 . . . . .	61
4.17	進入処理 . . . . .	63
4.17.1	前側進入処理 . . . . .	64
4.17.2	後側進入処理 . . . . .	65
4.18	メインです . . . . .	67
4.18.1	エントリーポイント . . . . .	67
4.18.2	入力検査 . . . . .	72
4.18.3	ルビ組版処理 . . . . .	74
4.18.4	前処理 . . . . .	79
4.18.5	後処理 . . . . .	80
4.19	デバッグ用出力 . . . . .	81
<b>5</b>	<b>実装 (圏点関連)</b> . . . . .	<b>82</b>
5.1	エラーメッセージ . . . . .	82
5.2	パラメタ . . . . .	83
5.2.1	全般設定 . . . . .	83
5.2.2	呼出時の設定 . . . . .	84
5.3	補助手続 . . . . .	84
5.3.1	\UTF 命令対応 . . . . .	84
5.3.2	リスト分解 . . . . .	84
5.4	パラメタ設定公開命令 . . . . .	87
5.5	圏点文字 . . . . .	88
5.6	圏点オプション解析 . . . . .	90
5.7	オプション整合性検査 . . . . .	92
5.8	ブロック毎の組版 . . . . .	92
5.9	圏点項目 . . . . .	93
5.9.1	\kspan 命令 . . . . .	97
5.10	自動抑止の検査 . . . . .	97
5.11	メインです . . . . .	98
5.11.1	エントリーポイント . . . . .	98
5.11.2	組版処理 . . . . .	99
5.11.3	前処理 . . . . .	100
5.11.4	後処理 . . . . .	100

5.12	デバッグ用出力 . . . . .	100
6	実装（圏点ルビ同時付加）	101
6.1	呼出時パラメタ . . . . .	101
6.2	その他の変数 . . . . .	101
6.3	オプション整合性検査 . . . . .	102
6.4	フォントサイズ . . . . .	102
6.5	ブロック毎の組版 . . . . .	103
7	実装：hyperref 対策	105

## 1 パッケージ読込

`\usepackage` 命令を用いて読み込む。オプションは存在しない。

```
\usepackage{pxrubrica}
```

## 2 ルビ機能

### 2.1 用語集

本パッケージで独自の意味をもつ単語を挙げる。

- 突出：ルビ文字出力の端が親文字よりも外側になること。
- 進入：ルビ文字出力が親文字に隣接する文字の（水平）領域に配置されること。
- 和文ルビ：親文字が和文文字であることを想定して処理されるルビ。
- 欧文ルビ：親文字が欧文文字であることを想定して処理されるルビ。
- グループ：ユーザにより指定された、親文字列・ルビ文字列の処理単位。
- 《文字》：均等割りにおいて不可分となる単位のこと。通常は、本来の意味での文字となるが、ユーザ指定で変更できる。
- ブロック：複数の親文字・ルビ文字の集まりで、大域的な配置決定の処理の中で内部の相対位置が固定されているもの。

次の用語については、『日本語組版の要件』に従う。

ルビ、親文字、中付き、肩付き、モノルビ、グループルビ、熟語ルビ

### 2.2 ルビ用命令

- `\ruby[〈オプション〉]{〈親文字〉}{〈ルビ文字〉}`

和文ルビの命令。すなわち、和文文字列の上側（横組）／右側（縦組）にルビを付す（オプションで逆側にもできる）。

ここで、〈オプション〉は以下の形式をもつ。

〈前進入設定〉〈前補助設定〉〈モード〉〈後補助設定〉〈後進入設定〉

〈前補助設定〉・〈モード〉・〈後補助設定〉は複数指定可能で、排他的な指定が併存した場合は後のものが有効になる。また、どの要素も省略可能で、その場合は `\rubyssetup` で指定された既定値が用いられる。ただし、構文上曖昧な指定を行った場合の結果は保証されない。例えば、「前進入無し」のみ指定する場合は `|` ではなく `|-` とする必要がある。

〈前進入設定〉は以下の値の何れか。

`||` 前突出禁止      `<` 前進入大

`|` 前進入無し      `(` 前進入小

〈前補助設定〉は以下の値の何れか。

`:` 和欧文間空白挿入      `*` 行分割禁止

`.` 空白挿入なし      `!` 段落頭で進入許可

– 空白挿入量の既定値は和文間空白である。

– `*` 無指定の場合の行分割の可否は `pdfLaTeX` の標準の動作に従う。

– `!` 無指定の場合、段落冒頭では〈前進入設定〉の設定に関わらず進入が抑止される。

– ゴースト処理が有効の場合はこの設定は無視される。

〈モード〉は以下の値の何れか。

<code>-</code>	(無指定)	<code>P</code> ( <i>&lt; primary</i> )	上側配置
<code>c</code> ( <i>&lt; center</i> )	中付き	<code>S</code> ( <i>&lt; secondary</i> )	下側配置
<code>h</code> ( <i>&lt; head</i> )	肩付き	<code>e</code> ( <i>&lt; even-space</i> )	親文字均等割り有効
<code>H</code>	拡張肩付き	<code>E</code>	親文字均等割り無効
<code>m</code> ( <i>&lt; mono</i> )	モノルビ	<code>f</code> ( <i>&lt; full-size</i> )	小書き文字変換有効
<code>g</code> ( <i>&lt; group</i> )	グループルビ	<code>F</code>	小書き文字変換無効
<code>j</code> ( <i>&lt; jukugo</i> )	熟語ルビ		
<code>M</code>	自動切替モノルビ		
<code>G</code>	自動切替グループルビ		

– 肩付き (`h`) の場合、ルビが短い場合にのみ、ルビ文字列と親文字列の頭を揃えて配置される。拡張肩付き (`H`) の場合、常に頭を揃えて配置される。

– `P` は親文字列の上側 (横組) / 右側 (縦組)、`S` は親文字列の下側 (横組) / 左側 (縦組) にルビを付す指定。

– `e` 指定時は、ルビが長い場合に親文字列をルビの長さに合わせて均等割りで配置する。`E` 指定時は、空きを入れずに中央揃えで配置する。なお、ルビが短い場合のルビ文字列の均等割りは常に有効である。

– `f` 指定時は、ルビ文字列中の (`{ }` の外にある) 小書き仮名 (あいうえおっやゆよわ、およびその片仮名) を対応の非小書き仮名に変換する。`F` 指定はこの機能を無効にする。

– `M` および `J` の指定は「グループルビとモノ・熟語ルビの間で自動的に切り替える」設定である。具体的には、ルビのグループが 1 つしかない場合は `m` および `g`、複数ある場合は `g` と等価になる。

(後補助設定) は以下の値の何れか。

： 和欧文間空白挿入      \* 行分割禁止  
． 空白挿入なし            ! 段落末で進入許可

– 空白挿入量の既定値は和文間空白である。

– \* 無指定の場合の行分割の可否は p $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  の標準の動作に従うのが原則だが、直後にあるものが文字でない場合、正しく動作しない（禁則が破れる）可能性がある。従って、不適切な行分割が起こりうる場合は適宜 \* を指定する必要がある（なお、段落末尾で \* を指定してはならない）。

– ! 無指定の場合、段落末尾では進入が抑止される。

– ゴースト処理が有効の場合はこの設定は無視される。

(後進入設定) は以下の値。

|| 後突出禁止            > 後進入大  
|  後進入無し            ) 後進入小

● `\jruby` [`<オプション>`]{`<親文字>`}{`<ルビ文字>`}

`\ruby` 命令の別名。 `\ruby` という命令名は他のパッケージとの衝突の可能性が高いため、 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  文書の本文開始時 (`\begin{document}`) に未定義である場合にのみ定義される。これに対して `\jruby` は常に定義される。なお、`\ruby` 以外の命令 (`\jruby` を含む) が定義済であった (命令名の衝突) 場合にはエラーとなる。

● `\aruby` [`<オプション>`]{`<親文字>`}{`<ルビ文字>`}

欧文ルビの命令。すなわち、欧文文字列の上側 (横組) / 右側 (縦組) にルビを付す。欧文ルビは和文ルビと比べて以下の点が異なる。

– 常にグループルビと扱われる。(m、g、j の指定は無効。)

– 親文字列の均等割りには常に無効である。(e 指定は無効。)

– ルビ付き文字と前後の文字との間の空き調整や行分割可否は両者がともに欧文であるという想定で行われる。従って、既定では空き調整量はゼロ、行分割は禁止となる。

– 空き調整を和欧文間空白 (:) にした場合は、\* が指定されるあるいは自動の禁則処理が働くのでない限り、行分割が許可される。

● `\truby` [`<オプション>`]{`<親文字>`}{`<上側ルビ文字>`}{`<下側ルビ文字>`}

和文両側ルビの命令。横組の場合、親文字列の上側と下側にルビを付す。縦組の場合、親文字列の右側と左側にルビを付す。

両側ルビで熟語ルビを使うことはできない。すなわち、`<オプション>` 中で j、J は指定できない。

※ 1.1 版以前では常にグループルビの扱いであった。旧版との互換のため、両側ルビの場合には自動切替モノルビ (M) を既定値とする。<sup>\*1</sup>

● `\atruby` [`<オプション>`]{`<親文字>`}{`<上側ルビ文字>`}{`<下側ルビ文字>`}

欧文両側ルビの命令。欧文ルビであることを除き `\truby` と同じ。

<sup>\*1</sup> つまり、旧来の使用ではグループルビと扱われるため、ルビのグループは 1 つにしているはずで、これは新版でもそのままグループルビと扱われる。一方で、モノルビを使いたい場合はグループを複数にするはずで、この時は自動的にモノルビになる。なので結局、基底モード (g、m) を指定する必要は無いことになる。

## 2.3 入力文字列のグループの指定

入力文字列（親文字列\*<sup>2</sup>・ルビ文字列）の中で「|」はグループの区切りとみなされる（ただし { } の中にあるものは文字とみなされる）。

例えば、ルビ文字列

```
じゆく|ご
```

は2つのグループからなり、最初のは3文字、後のは1文字からなる。

長さを合わせるために均等割りを行う場合、その分割の単位は通常は文字であるが、{ } で囲ったものは1文字とみなされる（本文書ではこの単位のことを《文字》と記す）。例えば

```
ベクタ{\< (-) \>}
```

は1つのグループからなり、それは4つの《文字》からなる。

グループや《文字》の指定はルビの付き方に影響する。その詳細を説明する。なお、非拡張機能では親文字のグループは常に1つに限られる。

- モノルビ・熟語ルビでは親文字列の1つの《文字》にルビ文字列の1つのグループが対応する。例えば、

```
\ruby[m]{熟語}{じゆく|ご}
```

は、「熟 + じゆく」「語 + ご」の2つのブロックからなる。

- (単純) グループルビではルビ文字列のグループも1つに限られ、親文字とルビ文字の唯一のグループが対応する。例えば、

```
\ruby[g]{五月雨}{さみだれ}
```

は、「五月雨 + さみだれ」の1つのブロックからなる。

拡張機能では、親文字列が複数グループをもつような使用法が存在する予定である。

## 2.4 ゴースト処理

「和文ゴースト処理」とは以下のようなものである：

和文ルビの親文字列出力の前後に全角空白文字を挿入する（ただしその空きを打ち消すように負の空きを同時に入れる）ことで、親文字列全体が、その外側から見たときに、全角空白文字（大抵の JFM ではこれは漢字と同じ扱いになる）と同様に扱われるようにする。例えば、前に欧文文字がある場合には自動的に和欧文間空白が挿入される。

「欧文ゴースト処理」も対象が欧文であることと除いて同じである。（こちらは、「複合語記号 (compound word mark)」というゼロ幅不可視の欧文文字を用いる。ルビ付文字列全体が単一欧文文字のように扱われる。) なお、「ゴースト (ghost)」というのは Omega の用

---

\*<sup>2</sup> 後述の通り、現在の版では親文字列を複数グループにする使用法は存在しないため、親文字列中では「|」は使われない。

語で、「不可視であるが（何らかの性質において）特定の可視の文字と同等の役割をもつオブジェクト」のことである。

ゴースト処理を有効にすると次のようなメリットがある。

- 和欧文間空白が自動的に挿入される。
- 行分割禁止（禁則処理）が常に正しく機能する。
- 特殊な状況（例えば段落末）でも異常動作を起こしにくい。
- （実装が単純化され、バグ混入の余地が少なくなる。）

ただし、次のような重要なデメリットがある。

- pT<sub>E</sub>X エンジンの仕様上の制約により、ルビ出力の進入と共存できない。（従って共存するような設定を試みるとエラーになる。）

このため、既定ではゴースト処理は無効になっている。有効にするには、`\rubyusejghost`（和文）/`\rubyuseaghost`（欧文）を実行する。

なお、`<前補助設定>/<後補助設定>` で指定される機能は、ゴースト処理が有効の場合には無効化される。これらの機能の目的が自動処理が失敗するのを捕逸するためだからである。

## 2.5 パラメタ設定命令

基本的設定。

- `\rubyssetup{<オプション>}`  
オプションの既定値設定。[既定 = `|cjPeF|`]
  - これ自体の既定値は「突出許可、進入無し、中付き、熟語ルビ、上側配置、親文字均等割り有効、小書き文字変換無効」である。
  - `<前補助設定>/<後補助設定>` の既定値は変更できない。`\rubyssetup` でこれらのオプション文字を指定しても無視される。
  - `\rubyssetup` での設定は累積する。例えば、初期状態から、`\rubyssetup{hmf}` と `\rubyssetup{<->}` を実行した場合、既定値設定は `<hmPef>` となる。
  - この設定に関わらず、両側ルビでは「自動切替モノルビ (M)」が既定として指定される。
- `\rubyfontsetup{<命令>}`  
ルビ用のフォント切替命令を設定する。例えば、ルビは必ず明朝体で出力したいという場合は、以下の命令を実行すればよい。  
`\rubyfontsetup{\mcfamily}`
- `\rubymargin{<実数>}`  
「大」の進入量（ルビ全角単位）。[既定 = 1]
- `\rubysmallmargin{<実数>}`  
「小」の進入量（ルビ全角単位）。[既定 = 0.5]
- `\rubymaxmargin{<実数>}`  
ルビ文字列の方が短い場合の、ルビ文字列の端の親文字列の端からの距離の上限値

(親文字全角単位)。[既定 = 0.75]

- `\rubyintergap{<実数>}`  
ルビと親文字の間の空き (親文字全角単位)。[既定 = 0]
- `\rubyusejghost` / `\rubynousejghost`  
和文ゴースト処理を行う / 行わない。[既定 = 行わない]
- `\rubyuseaghost` / `\rubynouseaghost`  
欧文ゴースト処理を行う / 行わない。[既定 = 行わない]

詳細設定。通常はこれらの既定値を変える必要はないだろう。

- `\rubysafemode` / `\rubynosafemode`  
安全モードを有効 / 無効にする。[既定 = 無効]
  - 本パッケージがサポートするエンジンは (u)pTeX、XeTeX、LuaTeX である。「安全モード」とは、これらのエンジンを必要とする一部の機能<sup>\*3</sup>を無効化したモードである。つまり、安全モードに切り替えることで、“サポート対象”でないエンジン (pdfTeX 等) でも本パッケージの一部の機能が使える可能性がある。
  - 使用中のエンジンが pdfTeX である場合、既定で安全モードが有効になる。
- `\rubysizeratio{<実数>}`  
ルビサイズの親文字サイズに対する割合。[既定 = 0.5]
- `\rubystretchprop{<X>}{<Y>}{<Z>}`  
ルビ用均等割りの比率の指定。[既定 = 1, 2, 1]
- `\rubystretchprophead{<Y>}{<Z>}`  
前突出禁止時の均等割りの比率の指定。[既定 = 1, 1]
- `\rubystretchpropend{<X>}{<Y>}`  
後突出禁止時の均等割りの比率の指定。[既定 = 1, 1]
- `\rubyyheightratio{<実数>}`  
横組和文の高さの縦幅に対する割合。[既定 = 0.88]
- `\rubytheightratio{<実数>}`  
縦組和文の「高さ」の「縦幅」に対する割合 (pTeX の縦組では「縦」と「横」が実際の逆になる)。[既定 = 0.5]

## 3 圏点機能

### 3.1 圏点用命令

- `\kenten[<オプション>]{<親文字>}`  
和文文字列の上側 (横組) / 右側 (縦組) に圏点を付す (オプションで逆側にもできる)。

---

<sup>\*3</sup> 安全モードでは、強制的にグループルビに切り替わる。また、親文字・ルビの両方の均等割り付け、および、小書き文字自動変換が無効になる。

〈オプション〉は複数指定可能で、排他的な指定が併存した場合は後のものが有効になる。また、省略された指定については `\kentensetup` で指定された既定値が用いられる。

オプションに指定できる値は以下の通り。

<code>p</code> ( <i>&lt; primary</i> )	主マーク	<code>P</code> ( <i>&lt; primary</i> )	上側配置
<code>s</code> ( <i>&lt; secondary</i> )	副マーク	<code>S</code> ( <i>&lt; secondary</i> )	下側配置
<code>f</code> ( <i>&lt; full</i> )	全文字付加有効		
<code>F</code>	全文字付加無効		

- `p`、`s` は付加する圏点の種類を表す。横組では主マーク (`p`) は黒中点、副マーク (`s`) は黒ゴマ点が用いられ、縦組では逆に主マークが黒ゴマ点、副マークが黒中点となる。ただし設定命令により圏点の種類は変更できる。
- `P` は親文字列の上側 (横組) / 右側 (縦組)、`S` は親文字列の下側 (横組) / 左側 (縦組) に圏点を付す指定。
- `f` 指定時は、親文字列に含まれる“通常文字”の全てに圏点を付加する。`F` 指定時は、約物である“通常文字”には圏点を付加しない。

## 3.2 圏点命令の親文字列

圏点付加の処理では親文字列を文字毎に分解する必要がある。このため、圏点命令の親文字列は一定の規則に従って書かれる必要がある。

圏点命令の親文字列には以下のものを含めることができる。

- 通常文字：`LATEX` の命令や特殊文字や欧文空白でない、欧文または和文の文字を指す。通常文字には一つの圏点が付加される。
  - `F` オプションを指定した場合、約物 (句読点等) の文字には圏点が付加されない。
  - 欧文文字に圏点を付けた場合、その文字は組版上“和文文字のように”振舞う。
- `LATEX` の命令および欧文空白：これらには圏点が付加されない。
  - 主に `\`、`や` `\quad` のような空白用の命令の使用を意図している。
  - `\hspace{1zw}` のような引数を取る命令をそのまま書くことはできない。この場合は、以降に示す何れかの書式を利用する必要がある。<sup>\*4</sup>
- グループ：すなわち、`{ }` に囲まれた任意のテキスト。ルビ命令のグループと同様に、一つの《文字》として扱われ、全体に対して一つの圏点が付加される。
  - `japanese-otf` パッケージの `\CID` 命令のような、「特殊な和文文字を出力する命令」の使用を意図している。
- `\kspan{<テキスト>}`：これは、出力されるテキストの幅に応じた個数の圏点が付加される。
  - 例えば、“くの字点”に圏点を付す場合に使える。

<sup>\*4</sup> 全角空白 (`\hspace{1zw}`) や和欧文間空白 (`\hspace{\kanjiskip}`) を出力する専用のマクロを用意しておくとも便利かもしれない。

- あるいは、(少々手抜きであるが<sup>\*5</sup>) `\kenten{この\kspan{\textgt{文字}}だ}` みたいな使い方も考えられる。
- `\kspan*{<テキスト>}`: これは圏点を付さずにテキストをそのまま出力する。
- ルビ命令 (`\ruby` 等): 例えば
  - `\kenten{これが\ruby[|j|]{圏点}{けん|てん}です}`。
 のように、ルビ命令はそのまま書くことができる。
  - `\kentenrubycombination` の設定によっては、ルビと圏点の両方が付加される。
  - 実装上の制限<sup>\*6</sup>のため、圏点命令の先頭にルビ命令がある場合、ルビの前側の進入が無効になる。同様に、圏点命令の末尾にルビ命令がある場合、ルビの後側の進入が無効になる。
  - 圏点命令中のルビの処理は通常の場合と比べて“複雑”であるため、自動的な禁則処理が働かない可能性が高い。従って、必要に応じて補助設定で分割禁止 (\*) を指定する必要がある。
  - 逆にルビ命令の入力に圏点命令をそのまま書くことはできない。
    - `\ruby[|j|]{\kenten{圏点}}{けん|てん}% 不可`
 { } で囲った《文字》の中では使えるが、この場合は同時付加とは見なされず、独立に動作することになる。

### 3.3 ゴースト処理

圏点出力ではルビと異なり進入の処理が不要である。このため、現状では、圏点命令については常に和文ゴースト処理を適用する。

※ 非標準の和文メトリック (JFM) が使われている等の理由で、和文ゴースト処理が正常に機能しない場合が存在する。このため、将来的に、圏点命令についても和文ゴースト処理を行わない (ルビ命令と同様の補助設定を適用する) 設定を用意する予定である。

### 3.4 パラメタ設定命令

- `\kentensetup{<オプション>}`  
オプションの既定値設定。[既定 = pPF]
- `\kentenmarkinyoko{<名前またはテキスト>}`  
横組時の主マーク (p 指定時) として使われる圏点を指定する。[既定 = bullet\*]  
パッケージで予め用意されている圏点種別については名前で指定できる。

<sup>\*5</sup> 本来は、`\textgt` の中で改めて `\kenten` を使うべきである。

<sup>\*6</sup> 圏点命令は常にゴースト処理を伴うため、先述の「ゴースト処理と進入は共存しない」という制限に引っかかるのである。

<code>bullet*</code>	・ (合成)	黒中点	<code>triangle</code>	▲ 25B2	黒三角
<code>bullet</code>	・ 2022*	黒中点	<code>Triangle</code>	△ 25B3	白三角
<code>Bullet</code>	◦ 25E6*	白中点	<code>circle</code>	● 25CF	黒丸
<code>sesame*</code>	ゝ (合成)	黒ゴマ点	<code>Circle</code>	○ 25CB	白丸
<code>sesame</code>	ゝ FE45*	黒ゴマ点	<code>bullseye</code>	◎ 25CE	二重丸
<code>Sesame</code>	ゝ FE46*	白ゴマ点	<code>fisheye</code>	◎ 25C9*	蛇の目点

- これらの圏点種別のうち、`bullet*` は中黒 “・” (U+30FB)、`sesame*` は読点 “、” (U+3001) の字形を加工したものを利用する。これらはどんな日本語フォントでもサポートされているので、確実に使用できる。
- それ以外の圏点種別は、記載の文字コードをもつ Unicode 文字を出力する。使用するフォントによっては、字形を持っていないため何も出力されない、あるいは字形が全角幅でないため正常に出力されない、という可能性がある。
- 文字コード値に \* を付けたものは、その文字が JIS X 0208 がないことを表す。pL<sup>A</sup>T<sub>E</sub>X でこれらの圏点種別を利用するためには `japanese-otf` パッケージを読み込む必要がある。

あるいは、名前の代わりに任意の L<sup>A</sup>T<sub>E</sub>X のテキストを書くことができる。<sup>\*7</sup>

`\kentenmarkinyoko{※}`

- `\kentensubmarkinyoko{<名前またはテキスト>}`  
横組時の副マーク (s 指定時) として使われる圏点を指定する。[既定 = `sesame*`]
- `\kentenmarkintate{<名前またはテキスト>}`  
縦組時の主マーク (p 指定時) として使われる圏点を指定する。[既定 = `sesame*`]
- `\kentensubmarkintate{<名前またはテキスト>}`  
縦組時の副マーク (s 指定時) として使われる圏点を指定する。[既定 = `bullet*`]
- `\kentenfontsetup{<命令>}`  
圏点用のフォント切替命令を設定する。
- `\kentenintergap{<実数>}`  
圏点と親文字の間の空き (親文字全角単位)。[既定 = 0]
- `\kentensizeratio{<実数>}`  
圏点サイズの親文字サイズに対する割合。[既定 = 0.5]

圏点とルビの同時付加に関する設定。

- `\kentenrubycombination{<値>}` 圏点命令の親文字中でルビ命令が使われた時の挙動を指定する。[既定 = `both`]  
– `ruby`: ルビのみを出力する。  
– `both`: ルビの外側に圏点を出力する。
- `\kentenrubyintergap{<実数>}`  
圏点とルビが同じ側に付いた時の間の空き (親文字全角単位)。[既定 = 0]

<sup>\*7</sup> ただし、引数の先頭の文字が ASCII 英字である場合は名前の指定と見なされるため、テキストとして扱いたい場合は適宜 { } を補う等の措置が必要である。

## 4 実装（ルビ関連）

### 4.1 前提パッケージ

keyval を使う予定（まだ使っていない）。

```
1 \RequirePackage{keyval}
```

### 4.2 エラーメッセージ

`\pxrr@error` エラー出力命令。

```
\pxrr@warn 2 \def\pxrr@pkgname{pxrubrica}
3 \def\pxrr@error{%
4   \PackageError\pxrr@pkgname
5 }
6 \def\pxrr@warn{%
7   \PackageWarning\pxrr@pkgname
8 }
```

`\ifpxrr@fatal@error` 致命的エラーが発生したか。スイッチ。

```
9 \newif\ifpxrr@fatal@error
```

`\pxrr@fatal@error` 致命的エラーのフラグを立てて、エラーを表示する。

```
10 \def\pxrr@fatal@error{%
11   \pxrr@fatal@errortrue
12   \pxrr@error
13 }
```

`\pxrr@eh@fatal` 致命的エラーのヘルプ。

```
14 \def\pxrr@eh@fatal{%
15   The whole ruby input was ignored.\MessageBreak
16   \@ehc
17 }
```

`\pxrr@fatal@not@supported` 未実装の機能を呼び出した場合。

```
18 \def\pxrr@fatal@not@supported#1{%
19   \pxrr@fatal@error{Not yet supported: #1}%
20   \pxrr@eh@fatal
21 }
```

`\pxrr@err@inv@value` 引数に無効な値が指定された場合。

```
22 \def\pxrr@err@inv@value#1{%
23   \pxrr@error{Invalid value (#1)}%
24   \@ehc
25 }
```

`\pxrr@fatal@unx@letter` オプション中に不測の文字が現れた場合。

```

26 \def\pxrr@fatal@unx@letter#1{%
27   \pxrr@fatal@error{Unexpected letter '#1' found}%
28   \pxrr@eh@fatal
29 }

```

`\pxrr@warn@bad@athead` モノルビ以外、あるいは横組みで肩付き指定が行われた場合。強制的に中付きに変更される。

```

30 \def\pxrr@warn@bad@athead{%
31   \pxrr@warn{Position 'h' not allowed here}%
32 }

```

`\pxrr@warn@must@group` 欧文ルビでグループルビ以外の指定が行われた場合。強制的にグループルビに変更される。

```

33 \def\pxrr@warn@must@group{%
34   \pxrr@warn{Only group ruby is allowed here}%
35 }

```

`\pxrr@warn@bad@jukugo` 両側ルビで熟語ルビの指定が行われた場合。強制的に選択的モノルビ (M) に変更される。

```

36 \def\pxrr@warn@bad@jukugo{%
37   \pxrr@warn{Jukugo ruby is not allowed here}%
38 }

```

`\pxrr@fatal@bad@intr` ゴースト処理が有効で進入有りを設定した場合。(致命的エラー)。

```

39 \def\pxrr@fatal@bad@intr{%
40   \pxrr@fatal@error{%
41     Intrusion disallowed when ghost is enabled%
42   }\pxrr@eh@fatal
43 }

```

`\pxrr@fatal@bad@no@protr` 前と後の両方で突出禁止を設定した場合。(致命的エラー)。

```

44 \def\pxrr@fatal@bad@no@protr{%
45   \pxrr@fatal@error{%
46     Protrusion must be allowed for either end%
47   }\pxrr@eh@fatal
48 }

```

`\pxrr@fatal@bad@length` 親文字列とルビ文字列でグループの個数が食い違う場合。(モノルビ・熟語ルビの場合、親文字のグループ数は実際には《文字》数のこと。)

```

49 \def\pxrr@fatal@bad@length#1#2{%
50   \pxrr@fatal@error{%
51     Group count mismatch between the ruby and\MessageBreak
52     the body (#1 <> #2)%
53   }\pxrr@eh@fatal
54 }

```

`\pxrr@fatal@bad@mono` モノルビ・熟語ルビの親文字列が2つ以上のグループを持つ場合。

```

55 \def\pxrr@fatal@bad@mono{%
56   \pxrr@fatal@error{%
57     Mono-ruby body must have a single group%
58   }\pxrr@eh@fatal
59 }

```

`\pxrr@fatal@bad@switching` 選択的ルビの親文字列が2つ以上のグループを持つ場合。

```

60 \def\pxrr@fatal@bad@switching{%
61   \pxrr@fatal@error{%
62     The body of Switching-ruby (M/J) must\MessageBreak
63     have a single group%
64   }\pxrr@eh@fatal
65 }

```

`\pxrr@fatal@bad@morable` 欧文ルビ（必ずグループルビとなる）でルビ文字列が2つ以上のグループを持つ場合。

```

66 \def\pxrr@fatal@bad@morable{%
67   \pxrr@fatal@error{%
68     Novable group ruby is not allowed here%
69   }\pxrr@eh@fatal
70 }

```

`\pxrr@fatal@na@morable` グループルビでルビ文字列が2つ以上のグループを持つ（つまり可動グループルビである）が、拡張機能が無効であるため実現できない場合。

```

71 \def\pxrr@fatal@na@morable{%
72   \pxrr@fatal@error{%
73     Feature of movable group ruby is disabled%
74   }\pxrr@eh@fatal
75 }

```

`\pxrr@warn@load@order` Unicode TeX 用の日本語組版パッケージ（LuaTeX-ja 等）はこのパッケージより前に読み込むべきだが、後で読み込まれていることが判明した場合。

```

76 \def\pxrr@warn@load@order#1{%
77   \pxrr@warn{%
78     This package should be loaded after '#1'%
79   }%
80 }

```

`\pxrr@interror` 内部エラー。これが出てはいけない。:-)

```

81 \def\pxrr@interror#1{%
82   \pxrr@fatal@error{INTERNAL ERROR (#1)}%
83   \pxrr@eh@fatal
84 }

```

`\ifpxrrDebug` デバッグモード指定。

```

85 \newif\ifpxrrDebug

```

### 4.3 パラメタ

#### 4.3.1 全般設定

`\pxrr@ruby@font` ルビ用フォント切替命令。

```

86 \let\pxrr@ruby@font\@empty

```

`\pxrr@big@intr` 「大」と「小」の進入量（`\rubybigintrusion`/`\rubysmallintrusion`）。実数値マクロ（数字列に展開される）。

`\pxrr@small@intr`

```

87 \def\pxrr@big@intr{1}
88 \def\pxrr@small@intr{0.5}

\pxrr@size@ratio ルビ文字サイズ (\rubysizeratio)。実数値マクロ。
89 \def\pxrr@size@ratio{0.5}

\pxrr@sprop@x 伸縮配置比率 (\rubystretchprop)。実数値マクロ。
\pxrr@sprop@y 90 \def\pxrr@sprop@x{1}
\pxrr@sprop@z 91 \def\pxrr@sprop@y{2}
92 \def\pxrr@sprop@z{1}

\pxrr@sprop@hy 伸縮配置比率 (\rubystretchprophead)。実数値マクロ。
\pxrr@sprop@hz 93 \def\pxrr@sprop@hy{1}
94 \def\pxrr@sprop@hz{1}

\pxrr@sprop@ex 伸縮配置比率 (\rubystretchpropend)。実数値マクロ。
\pxrr@sprop@ey 95 \def\pxrr@sprop@ex{1}
96 \def\pxrr@sprop@ey{1}

\pxrr@maxmargin ルビ文字列の最大マージン (\rubymaxmargin)。実数値マクロ。
97 \def\pxrr@maxmargin{0.75}

\pxrr@yhtratio 横組和文の高さの縦幅に対する割合 (\rubyheightratio)。実数値マクロ。
98 \def\pxrr@yhtratio{0.88}

\pxrr@thtratio 縦組和文の高さの縦幅に対する割合 (\rubytheightratio)。実数値マクロ。
99 \def\pxrr@thtratio{0.5}

\pxrr@extra 拡張機能実装方法 (\rubyuseextra)。整数定数。
100 \chardef\pxrr@extra=0

\ifpxrr@jghost 和文ゴースト処理を行うか (\ruby[no]usejghost)。スイッチ。
101 \newif\ifpxrr@jghost \pxrr@jghostfalse

\ifpxrr@aghost 欧文ゴースト処理を行うか (\ruby[no]useaghost)。スイッチ。
102 \newif\ifpxrr@aghost \pxrr@aghostfalse

\pxrr@inter@gap ルビと親文字の間の空き (\rubyintergap)。実数値マクロ。
103 \def\pxrr@inter@gap{0}

\ifpxrr@edge@adjust 行頭・行末での突出の自動補正を行うか (\ruby[no]adjustatlineedge)。スイッチ。
104 \newif\ifpxrr@edge@adjust \pxrr@edge@adjustfalse

\ifpxrr@break@jukugo 熟語ルビで中間の行分割を許すか (\ruby[no]breakjukugo)。スイッチ。
105 \newif\ifpxrr@break@jukugo \pxrr@break@jukugofalse

\ifpxrr@safe@mode 安全モードであるか。(\ruby[no]safemode)。スイッチ。
106 \newif\ifpxrr@safe@mode \pxrr@safe@modedefalse

```

`\ifpxrr@d@bprotr` 突出を許すか否か。`\rubysetup` の〈前設定〉/〈後設定〉に由来する。スイッチ。

`\ifpxrr@d@aprotr` 107 `\newif\ifpxrr@d@bprotr \pxrr@d@bprotrtrue`  
 108 `\newif\ifpxrr@d@aprotr \pxrr@d@aprotrtrue`

`\pxrr@d@bintr` 進入量。`\rubysetup` の〈前設定〉/〈後設定〉に由来する。`\pxrr@XXX@intr` または空（進入無し）に展開されるマクロ。

`\pxrr@d@aintr` 109 `\def\pxrr@d@bintr{}`  
 110 `\def\pxrr@d@aintr{}`

`\pxrr@d@athead` 肩付き/中付きの設定。`\rubysetup` の `c/h/H` の設定。0 = 中付き (c); 1 = 肩付き (h); 2 = 拡張肩付き (H)。整数定数。

111 `\chardef\pxrr@d@athead=0`

`\pxrr@d@mode` モノルビ (m)・グループルビ (g)・熟語ルビ (j) のいずれか。`\rubysetup` の設定値。オプション文字への暗黙の (`\let` された) 文字トークン。

112 `\let\pxrr@d@mode=j`

`\pxrr@d@side` ルビを親文字の上下のどちらに付すか。0 = 上側; 1 = 下側。`\rubysetup` の `P/S` の設定。整数定数。

113 `\chardef\pxrr@d@side=0`

`\pxrr@d@evensp` 親文字列均等割りの設定。0 = 無効; 1 = 有効。`\rubysetup` の `e/E` の設定。整数定数。

114 `\chardef\pxrr@d@evensp=1`

`\pxrr@d@fullsize` 小書き文字変換の設定。0 = 無効; 1 = 有効。`\rubysetup` の `f/F` の設定。整数定数。

115 `\chardef\pxrr@d@fullsize=0`

#### 4.3.2 呼出時パラメタ・変数

一般的に、特定のルビ・圏点命令の呼出に固有である（つまりその内側にネストされたルビ・圏点命令に継承すべきでない）パラメタは、呼出時の値を別に保持しておくべきである。

`\ifpxrr@bprotr` 突出を許すか否か。`\ruby` の〈前設定〉/〈後設定〉に由来する。スイッチ。

`\ifpxrr@aprotr` 116 `\newif\ifpxrr@bprotr \pxrr@bprotrfalse`  
 117 `\newif\ifpxrr@aprotr \pxrr@aprotrfalse`

`\pxrr@bintr` 進入量。`\ruby` の〈前設定〉/〈後設定〉に由来する。寸法値に展開されるマクロ。

`\pxrr@aintr` 118 `\def\pxrr@bintr{}`  
 119 `\def\pxrr@aintr{}`

`\pxrr@bscomp` 空き補正設定。`\ruby` の `:` 指定に由来する。暗黙の文字トークン（無指定は `\relax`）。

`\pxrr@ascomp` ※ 既定値設定 (`\rubysetup`) でこれに対応するものはない。

120 `\let\pxrr@bscomp\relax`  
 121 `\let\pxrr@ascomp\relax`

`\ifpxrr@bnobr` ルビ付文字の直前/直後で行分割を許すか。`\ruby` の `*` 指定に由来する。スイッチ。

`\ifpxrr@anobr` ※ 既定値設定 (`\rubysetup`) でこれに対応するものはない。

122 `\newif\ifpxrr@bnobr \pxrr@bnobrfalse`  
123 `\newif\ifpxrr@anobr \pxrr@anobrfalse`

`\ifpxrr@bfintr` 段落冒頭／末尾で進入を許可するか。`\ruby` の `!` 指定に由来する。スイッチ。  
`\ifpxrr@afintr` ※ 既定値設定 (`\rubysetup`) でこれに対応するものはない。

124 `\newif\ifpxrr@bfintr \pxrr@bfintrfalse`  
125 `\newif\ifpxrr@afintr \pxrr@afintrfalse`

`\pxrr@athead` 肩付き／中付きの設定。`\ruby` の `c/h/H` の設定。値の意味は `\pxrr@d@athead` と同じ。  
整数定数。

126 `\chardef\pxrr@athead=0`

`\ifpxrr@athead@given` 肩付き／中付きの設定が明示的であるか。スイッチ。

127 `\newif\ifpxrr@athead@given \pxrr@athead@givenfalse`

`\pxrr@mode` モノルビ (`m`)・グループルビ (`g`)・熟語ルビ (`j`) のいずれか。`\ruby` のオプションの設定  
値。オプション文字への暗黙文字トークン。

128 `\let\pxrr@mode=\@undefined`

`\ifpxrr@mode@given` 基本モードの設定が明示的であるか。スイッチ。

129 `\newif\ifpxrr@mode@given \pxrr@mode@givenfalse`  
130 `\newif\ifpxrr@afintr \pxrr@afintrfalse`

`\ifpxrr@abody` ルビが `\aruby` (欧文親文字用) であるか。スイッチ。

131 `\newif\ifpxrr@abody`

`\pxrr@side` ルビを親文字の上下のどちらに付すか。0 = 上側 ; 1 = 下側 ; 2 = 両側。`\ruby` の `P/S` が  
0/1 に対応し、`\truby` では 2 が使用される。整数定数。

132 `\chardef\pxrr@side=0`

`\pxrr@evensp` 親文字列均等割りの設定。0 = 無効 ; 1 = 有効。`\ruby` の `e/E` の設定。整数定数。

133 `\chardef\pxrr@evensp=1`

`\pxrr@revensp` ルビ文字列均等割りの設定。0 = 無効 ; 1 = 有効。整数定数。  
※ 通常は有効だが、安全モードでは無効になる。

134 `\chardef\pxrr@revensp=1`

`\pxrr@fullsize` 小書き文字変換の設定。0 = 無効 ; 1 = 有効。`\ruby` の `f/F` の設定。整数定数。

135 `\chardef\pxrr@fullsize=1`

`\pxrr@c@ruby@font` 以下は“オプションで指定する”以外のパラメタに対応するもの。

`\pxrr@c@size@ratio` 136 `\let\pxrr@c@ruby@font\@undefined`  
137 `\let\pxrr@c@size@ratio\@undefined`  
`\pxrr@c@inter@gap` 138 `\let\pxrr@c@inter@gap\@undefined`

## 4.4 その他の変数

`\pxrr@body@list` 親文字列のために使うリスト。  
139 `\let\pxrr@body@list\@undefined`

`\pxrr@body@count` `\pxrr@body@list` の長さ。整数値マクロ。  
140 `\let\pxrr@body@count\@undefined`

`\pxrr@ruby@list` ルビ文字列のために使うリスト。  
141 `\let\pxrr@ruby@list\@undefined`

`\pxrr@ruby@count` `\pxrr@ruby@list` の長さ。整数値マクロ。  
142 `\let\pxrr@ruby@count\@undefined`

`\pxrr@sruby@list` 2 目目のルビ文字列のために使うリスト。  
143 `\let\pxrr@sruby@list\@undefined`

`\pxrr@sruby@count` `\pxrr@sruby@list` の長さ。整数値マクロ。  
144 `\let\pxrr@sruby@count\@undefined`

`\pxrr@whole@list` 親文字とルビのリストを zip したリスト。  
145 `\let\pxrr@whole@list\@undefined`

`\pxrr@bspace` ルビが親文字から前側にはみだす長さ。寸法値マクロ。  
146 `\let\pxrr@bspace\@undefined`

`\pxrr@aspace` ルビが親文字から後側にはみだす長さ。寸法値マクロ。  
147 `\let\pxrr@aspace\@undefined`

`\pxrr@natwd` `\pxrr@evenspace@int` のパラメタ。寸法値マクロ。  
148 `\let\pxrr@natwd\@undefined`

`\pxrr@all@input` 両側ルビの処理で使われる一時変数。  
149 `\let\pxrr@all@input\@undefined`

## 4.5 補助手続

### 4.5.1 雑多な定義

`\ifpxrr@ok` 汎用スイッチ。  
150 `\newif\ifpxrr@ok`

`\pxrr@canta` 汎用の整数レジスタ。  
151 `\newcount\pxrr@canta`

`\pxrr@cntr` 結果を格納する整数レジスタ。  
152 `\newcount\pxrr@cntr`

`\pxrr@dima` 汎用の寸法レジスタ。  
153 `\newdimen\pxrr@dima`

`\pxrr@boxa` 汎用のボックスレジスタ。  
`\pxrr@boxb` 154 `\newbox\pxrr@boxa`  
155 `\newbox\pxrr@boxb`

`\pxrr@boxr` 結果を格納するボックスレジスタ。  
156 `\newbox\pxrr@boxr`

`\pxrr@token` `\futurelet` 用の一時変数。  
※ if-トークンなどの“危険”なトークンになりうるので使い回さない。  
157 `\let\pxrr@token\relax`

`\pxrr@zero` 整数定数のゼロ。`\z@` と異なり、「単位付寸法」の係数として使用可能。  
158 `\chardef\pxrr@zero=0`

`\pxrr@zeropt` 「Opt」という文字列。寸法値マクロへの代入に用いる。  
159 `\def\pxrr@zeropt{Opt}`

`\pxrr@hfilx` `\pxrr@hfilx{<実数>}`: 「<実数>fil」のグルーを置く。  
160 `\def\pxrr@hfilx#1{%`  
161 `\hskip\z@\@plus #1fil\relax`  
162 `}`

`\pxrr@res` 結果を格納するマクロ。  
163 `\let\pxrr@res\@empty`

`\pxrr@ifx` `\pxrr@ifx{<引数>}<真>}<偽>}`: `\ifx<引数>` を行うテスト。  
164 `\def\pxrr@ifx#1{%`  
165 `\ifx#1\expandafter\@firstoftwo`  
166 `\else\expandafter\@secondoftwo`  
167 `\fi`  
168 `}`

`\pxrr@cond` `\pxrr@cond\ifXXX...\fi{<真>}<偽>}`: 一般の T<sub>E</sub>X の if 文 `\ifXXX...` を行うテスト。  
※ `\fi` を付けているのは、if-不均衡を避けるため。  
169 `\@gobbletwo\if\if \def\pxrr@cond#1\fi{%`  
170 `#1\expandafter\@firstoftwo`  
171 `\else\expandafter\@secondoftwo`  
172 `\fi`  
173 `}`

`\pxrr@cslet` `\pxrr@cslet{NAMEa}\CSb`: `\NAMEa` に `\CSb` を `\let` する。  
`\pxrr@letcs` `\pxrr@letcs\CSa{NAMEb}`: `\CSa` に `\NAMEb` を `\let` する。  
`\pxrr@csletcs` `\pxrr@csletcs{NAMEa}{NAMEb}`: `\NAMEa` に `\NAMEb` を `\let` する。  
174 `\def\pxrr@cslet#1{%`  
175 `\expandafter\let\csname#1\endcsname`

```

176 }
177 \def\pxrr@letcs#1#2{%
178   \expandafter\let\expandafter#1\csname#2\endcsname
179 }
180 \def\pxrr@csletcs#1#2{%
181   \expandafter\let\csname#1\expandafter\endcsname
182   \csname#2\endcsname
183 }

```

`\pxrr@setok` `\pxrr@setok{<テスト>}`: テストの結果を `\ifpxrr@ok` に返す。

```

184 \def\pxrr@setok#1{%
185   #1{\pxrr@oktrue}{\pxrr@okfalse}%
186 }

```

`\pxrr@appto` `\pxrr@appto\CS{<テキスト>}`: 無引数マクロの置換テキストに追加する。

```

187 \def\pxrr@appto#1#2{%
188   \expandafter\def\expandafter#1\expandafter{#1#2}%
189 }

```

`\pxrr@nil` ユニークトークン。

```

\pxrr@end 190 \def\pxrr@nil{\noexpand\pxrr@nil}
191 \def\pxrr@end{\noexpand\pxrr@end}

```

`\pxrr@without@macro@trace` `\pxrr@without@macro@trace{<テキスト>}`: マクロ展開のトレースを無効にした状態で `<テキスト>` を実行する。

```

192 \def\pxrr@without@macro@trace#1{%
193   \chardef\pxrr@tracingmacros@save=\tracingmacros
194   \tracingmacros\z@
195   #1%
196   \tracingmacros\pxrr@tracingmacros@save
197 }
198 \chardef\pxrr@tracingmacros@save=0

```

`\pxrr@hbox` `color` パッケージ対応の `\hbox` と `\hb@xt@` (= `\hbox to`)。

```

\pxrr@hbox@to 199 \def\pxrr@hbox#1{%
200   \hbox{%
201     \color@begingroup
202     #1%
203     \color@endgroup
204   }%
205 }
206 \def\pxrr@hbox@to#1#2{%
207   \pxrr@hbox@to@a{#1}%
208 }
209 \def\pxrr@hbox@to@a#1#2{%
210   \hbox to#1{%
211     \color@begingroup
212     #2%
213     \color@endgroup

```

```

214 }%
215 }

```

color パッケージ不使用の場合は、本来の `\hbox` と `\hb@xt@` に戻しておく。これと同期して `\pxrr@takeout@any@protr` の動作も変更する。

```

216 \AtBeginDocument{%
217   \ifx\color@begingroup\relax
218     \ifx\color@endgroup\relax
219       \let\pxrr@hbox\hbox
220       \let\pxrr@hbox@to\hb@xt@
221       \let\pxrr@takeout@any@protr\pxrr@takeout@any@protr@nocolor
222     \fi
223   \fi
224 }

```

#### 4.5.2 数値計算

`\pxrr@invscale` `\pxrr@invscale{<寸法レジスタ>}{<実数>}`: 現在の `<寸法レジスタ>` の値を `<実数>` で除算した値に更新する。すなわち、`<寸法レジスタ>=<実数><寸法レジスタ>` の逆の演算を行う。

```

225 \mathchardef\pxrr@invscale@ca=259
226 \def\pxrr@invscale#1#2{%
227   \begingroup
228     \@tempdima=#1\relax
229     \@tempdimb#2\p@\relax
230     \@tempcnta\@tempdima
231     \multiply\@tempcnta\@ccclvi
232     \divide\@tempcnta\@tempdimb
233     \multiply\@tempcnta\@ccclvi
234     \@tempcntb\p@
235     \divide\@tempcntb\@tempdimb
236     \advance\@tempcnta-\@tempcntb
237     \advance\@tempcnta-\tw@
238     \@tempdimb\@tempcnta\@ne
239     \advance\@tempcnta\@tempcntb
240     \advance\@tempcnta\@tempcntb
241     \advance\@tempcnta\pxrr@invscale@ca
242     \@tempdimc\@tempcnta\@ne
243     \@whiledim\@tempdimb<\@tempdimc\do{%
244       \@tempcntb\@tempdimb
245       \advance\@tempcntb\@tempdimc
246       \advance\@tempcntb\@ne
247       \divide\@tempcntb\tw@
248     \ifdim #2\@tempcntb>\@tempdima
249       \advance\@tempcntb\m@ne
250       \@tempdimc=\@tempcntb\@ne
251     \else
252       \@tempdimb=\@tempcntb\@ne
253     \fi}%

```

```

254 \xdef\pxrr@tempa{\the\@tempdimb}%
255 \endgroup
256 #1=\pxrr@tempa\relax
257 }

```

`\pxrr@interpolate` `\pxrr@interpolate{⟨入力単位⟩}{⟨出力単位⟩}{⟨寸法レジスタ⟩}{⟨ $X_1, Y_1$ ⟩}{⟨ $X_2, Y_2$ ⟩}⋯{⟨ $X_n, Y_n$ ⟩}` : 線形補間を行う。すなわち、明示値

$$f(0 \text{ pt}) = 0 \text{ pt}, f(X_1 \text{ iu}) = Y_1 \text{ ou}, \dots, f(X_n \text{ iu}) = Y_n \text{ ou}$$

(ただし  $(0, \text{pt} < X_1 \text{ iu} < \dots < X_n \text{ iu})$  ; ここで iu は ⟨入力単位⟩、ou は ⟨出力単位⟩ に指定されたもの) を線形補間して定義される関数  $f(\cdot)$  について、 $f(\langle \text{寸法} \rangle)$  の値を ⟨寸法レジスタ⟩ に代入する。

※  $[0 \text{ pt}, X_n \text{ iu}]$  の範囲外では両端の 2 点による外挿を行う。

```

258 \def\pxrr@interpolate#1#2#3#4#5{%
259 \edef\pxrr@tempa{#1}%
260 \edef\pxrr@tempb{#2}%
261 \def\pxrr@tempd{#3}%
262 \setlength{\@tempdima}{#4}%
263 \edef\pxrr@tempc{(0,0)#5(*,*)}%
264 \expandafter\pxrr@interpolate@a\pxrr@tempc\@nil
265 }
266 \def\pxrr@interpolate@a(#1,#2)(#3,#4)(#5,#6){%
267 \if#5%
268 \def\pxrr@tempc{\pxrr@interpolate@b{#1}{#2}{#3}{#4}}%
269 \else\ifdim\@tempdima<#3\pxrr@tempa
270 \def\pxrr@tempc{\pxrr@interpolate@b{#1}{#2}{#3}{#4}}%
271 \else
272 \def\pxrr@tempc{\pxrr@interpolate@a(#3,#4)(#5,#6)}%
273 \fi\fi
274 \pxrr@tempc
275 }
276 \def\pxrr@interpolate@b#1#2#3#4#5\@nil{%
277 \@tempdimb=-#1\pxrr@tempa
278 \advance\@tempdima\@tempdimb
279 \advance\@tempdimb#3\pxrr@tempa
280 \edef\pxrr@tempc{\strip@pt\@tempdimb}%
281 \pxrr@invscale\@tempdima\pxrr@tempc
282 \edef\pxrr@tempc{\strip@pt\@tempdima}%
283 \@tempdima=#4\pxrr@tempb
284 \@tempdimb=#2\pxrr@tempb
285 \advance\@tempdima-\@tempdimb
286 \@tempdima=\pxrr@tempc\@tempdima
287 \advance\@tempdima\@tempdimb
288 \pxrr@tempd=\@tempdima
289 }

```

### 4.5.3 リスト分解

`\pxrr@decompose` `\pxrr@decompose{〈要素 1〉…〈要素 n〉}`: ここで各〈要素〉は単一トークンまたはグループ (`{…}` で囲まれたもの) とする。この場合、`\pxrr@res` を以下のトークン列に定義する。

```
\pxrr@pre{〈要素 1〉}\pxrr@inter{〈要素 2〉}…
\pxrr@inter{〈要素 n〉}\pxrr@post
```

そして、`\pxrr@cntr` を `n` に設定する。

※ 〈要素〉に含まれるグルーピングは完全に保存される (最外の `{…}` が外れたりしない)。

```
290 \def\pxrr@decompose#1{%
291   \let\pxrr@res\@empty
292   \pxrr@cntr=\z@
293   \pxrr@decompose@loopa#1\pxrr@end
294 }
295 \def\pxrr@decompose@loopa{%
296   \futurelet\pxrr@token\pxrr@decompose@loopb
297 }
298 \def\pxrr@decompose@loopb{%
299   \pxrr@ifx{\pxrr@token\pxrr@end}{%
300     \pxrr@appto\pxrr@res{\pxrr@post}%
301   }{%
302     \pxrr@setok{\pxrr@ifx{\pxrr@token\bgroup}}%
303     \pxrr@decompose@loopc
304   }%
305 }
306 \def\pxrr@decompose@loopc#1{%
307   \ifx\pxrr@res\@empty
308     \def\pxrr@res{\pxrr@pre}%
309   \else
310     \pxrr@appto\pxrr@res{\pxrr@inter}%
311   \fi
312   \ifpxrr@ok
313     \pxrr@appto\pxrr@res{{#1}}%
314   \else
315     \pxrr@appto\pxrr@res{{#1}}%
316   \fi
317   \advance\pxrr@cntr\@ne
318   \pxrr@decompose@loopa
319 }
```

`\pxrr@decompbar` `\pxrr@decompbar{〈要素 1〉|…|〈要素 n〉}`: ただし、各〈要素〉はグルーピングの外の `|` を含まないとする。入力の形式と〈要素〉の構成条件が異なることを除いて、`\pxrr@decompose` と同じ動作をする。

```
320 \def\pxrr@decompbar#1{%
321   \let\pxrr@res\@empty
322   \pxrr@cntr=\z@
```

```

323 \pxrr@decompbar@loopa\pxrr@nil#1|\pxrr@end|{%
324 }
325 \def\pxrr@decompbar@loopa#1|{%
326 \expandafter\pxrr@decompbar@loopb\expandafter{\@gobble#1}%
327 }
328 \def\pxrr@decompbar@loopb#1{%
329 \pxrr@decompbar@loopc#1\relax\pxrr@nil{#1}%
330 }
331 \def\pxrr@decompbar@loopc#1#2\pxrr@nil#3{%
332 \pxrr@ifx{#1\pxrr@end}{%
333 \pxrr@appto\pxrr@res{\pxrr@post}%
334 }{%
335 \ifx\pxrr@res\@empty
336 \def\pxrr@res{\pxrr@pre}%
337 \else
338 \pxrr@appto\pxrr@res{\pxrr@inter}%
339 \fi
340 \pxrr@appto\pxrr@res{#3}}%
341 \advance\pxrr@cntr\@ne
342 \pxrr@decompbar@loopa\pxrr@nil
343 }%
344 }

```

\pxrr@zip@list \pxrr@zip@list\CSa\CSb : \CSa と \CSb が以下のように展開されるマクロとする :

$$\begin{aligned} \backslash\text{CSa} &= \backslash\text{pxrr@pre}\langle X1\rangle\backslash\text{pxrr@inter}\langle X2\rangle\cdots\backslash\text{pxrr@inter}\langle Xn\rangle\backslash\text{pxrr@post} \\ \backslash\text{CSb} &= \backslash\text{pxrr@pre}\langle Y1\rangle\backslash\text{pxrr@inter}\langle Y2\rangle\cdots\backslash\text{pxrr@inter}\langle Yn\rangle\backslash\text{pxrr@post} \end{aligned}$$

この命令は \pxrr@res を以下の内容に定義する。

$$\begin{aligned} &\backslash\text{pxrr@pre}\langle X1\rangle\langle Y1\rangle\backslash\text{pxrr@inter}\langle X2\rangle\langle Y2\rangle\cdots \\ &\backslash\text{pxrr@inter}\langle Xn\rangle\langle Yn\rangle\backslash\text{pxrr@post} \end{aligned}$$

```

345 \def\pxrr@zip@list#1#2{%
346 \let\pxrr@res\@empty
347 \let\pxrr@post\relax
348 \let\pxrr@tempa#1\pxrr@appto\pxrr@tempa{}}%
349 \let\pxrr@tempb#2\pxrr@appto\pxrr@tempb{}}%
350 \pxrr@zip@list@loopa
351 }
352 \def\pxrr@zip@list@loopa{%
353 \expandafter\pxrr@zip@list@loopb\pxrr@tempa\pxrr@end
354 }
355 \def\pxrr@zip@list@loopb#1#2#3\pxrr@end{%
356 \pxrr@ifx{#1\relax}{%
357 \pxrr@zip@list@exit
358 }{%
359 \pxrr@appto\pxrr@res{#1{#2}}%
360 \def\pxrr@tempa{#3}%
361 \expandafter\pxrr@zip@list@loopc\pxrr@tempb\pxrr@end

```

```

362 }%
363 }
364 \def\pxrr@zip@list@loopc#1#2#3\pxrr@end{%
365   \pxrr@ifx{#1\relax}{%
366     \pxrr@interror{zip}%
367     \pxrr@appto\pxrr@res{}}%
368   \pxrr@zip@list@exit
369 }{%
370   \pxrr@appto\pxrr@res{#2}}%
371   \def\pxrr@tempb{#3}%
372   \pxrr@zip@list@loopa
373 }%
374 }
375 \def\pxrr@zip@list@exit{%
376   \pxrr@appto\pxrr@res{\pxrr@post}%
377 }

```

`\pxrr@tzip@list` `\pxrr@tzip@list\CSa\CSb\CSc` : `\CSa`、`\CSb`、`\CSc` が以下のように展開されるマクロとする :

$$\begin{aligned} \text{\CSa} &= \text{\pxrr@pre}\langle X1 \rangle \text{\pxrr@inter}\langle X2 \rangle \cdots \text{\pxrr@inter}\langle Xn \rangle \text{\pxrr@post} \\ \text{\CSb} &= \text{\pxrr@pre}\langle Y1 \rangle \text{\pxrr@inter}\langle Y2 \rangle \cdots \text{\pxrr@inter}\langle Yn \rangle \text{\pxrr@post} \\ \text{\CSc} &= \text{\pxrr@pre}\langle Z1 \rangle \text{\pxrr@inter}\langle Z2 \rangle \cdots \text{\pxrr@inter}\langle Zn \rangle \text{\pxrr@post} \end{aligned}$$

この命令は `\pxrr@res` を以下の内容に定義する。

$$\begin{aligned} &\text{\pxrr@pre}\langle X1 \rangle \text{\langle Y1 \rangle} \text{\langle Z1 \rangle} \text{\pxrr@inter}\langle X2 \rangle \text{\langle Y2 \rangle} \text{\langle Z2 \rangle} \cdots \\ &\text{\pxrr@inter}\langle Xn \rangle \text{\langle Yn \rangle} \text{\langle Zn \rangle} \text{\pxrr@post} \end{aligned}$$

```

378 \def\pxrr@tzip@list#1#2#3{%
379   \let\pxrr@res\@empty
380   \let\pxrr@post\relax
381   \let\pxrr@tempa#1\pxrr@appto\pxrr@tempa{}}%
382   \let\pxrr@tempb#2\pxrr@appto\pxrr@tempb{}}%
383   \let\pxrr@tempc#3\pxrr@appto\pxrr@tempc{}}%
384   \pxrr@tzip@list@loopa
385 }
386 \def\pxrr@tzip@list@loopa{%
387   \expandafter\pxrr@tzip@list@loopb\pxrr@tempa\pxrr@end
388 }
389 \def\pxrr@tzip@list@loopb#1#2#3\pxrr@end{%
390   \pxrr@ifx{#1\relax}{%
391     \pxrr@tzip@list@exit
392   }{%
393     \pxrr@appto\pxrr@res{#1{#2}}%
394     \def\pxrr@tempa{#3}%
395     \expandafter\pxrr@tzip@list@loopc\pxrr@tempb\pxrr@end
396   }%
397 }
398 \def\pxrr@tzip@list@loopc#1#2#3\pxrr@end{%

```

```

399 \pxrr@ifx{#1\relax}{%
400   \pxrr@interror{tzip}%
401   \pxrr@appto\pxrr@res{}}%
402   \pxrr@tzip@list@exit
403 }{%
404   \pxrr@appto\pxrr@res{#2}}%
405   \def\pxrr@tempb{#3}%
406   \expandafter\pxrr@tzip@list@loopd\pxrr@tempc\pxrr@end
407 }%
408 }
409 \def\pxrr@tzip@list@loopd#1#2#3\pxrr@end{%
410   \pxrr@ifx{#1\relax}{%
411     \pxrr@interror{tzip}%
412     \pxrr@appto\pxrr@res{}}%
413     \pxrr@tzip@list@exit
414   }{%
415     \pxrr@appto\pxrr@res{#2}}%
416     \def\pxrr@tempc{#3}%
417     \pxrr@tzip@list@loopa
418   }%
419 }
420 \def\pxrr@tzip@list@exit{%
421   \pxrr@appto\pxrr@res{\pxrr@post}}%
422 }

```

`\pxrr@concat@list` `\pxrr@concat@list\CS` : リストの要素を連結する。すなわち、`\CS` が

$$\backslash\text{CSa} = \backslash\text{pxrr@pre}\langle X1\rangle\backslash\text{pxrr@inter}\langle X2\rangle\cdots\backslash\text{pxrr@inter}\langle Xn\rangle\backslash\text{pxrr@post}$$

の時に、`\pxrr@res` を以下の内容に定義する。

$$\langle X1\rangle\langle X2\rangle\cdots\langle Xn\rangle$$

```

423 \def\pxrr@concat@list#1{%
424   \let\pxrr@res\empty
425   \def\pxrr@pre##1{%
426     \pxrr@appto\pxrr@res{##1}}%
427   }%
428   \let\pxrr@inter\pxrr@pre
429   \let\pxrr@post\relax
430   #1%
431 }

```

`\pxrr@unite@group` `\pxrr@unite@group\CS` : リストの要素を連結して 1 要素のリストに組み直す。すなわち、`\CS` が

$$\backslash\text{CS} = \backslash\text{pxrr@pre}\langle X1\rangle\backslash\text{pxrr@inter}\langle X2\rangle\cdots\backslash\text{pxrr@inter}\langle Xn\rangle\backslash\text{pxrr@post}$$

の時に、`\CS` を以下の内容で置き換える。

$$\backslash\text{pxrr@pre}\langle X1\rangle\langle X2\rangle\cdots\langle Xn\rangle\backslash\text{pxrr@post}$$

```

432 \def\pxrr@unite@group#1{%
433   \expandafter\pxrr@concat@list\expandafter{#1}%
434   \expandafter\pxrr@unite@group@a\pxrr@res\pxrr@end#1%
435 }
436 \def\pxrr@unite@group@a#1\pxrr@end#2{%
437   \def#2{\pxrr@pre{#1}\pxrr@post}%
438 }

```

`\pxrr@zip@single` `\pxrr@zip@single\CSa\CSb` :

$$\backslash\text{CSa} = \langle X \rangle; \backslash\text{CSb} = \langle Y \rangle$$

の時に、`\pxrr@res` を以下の内容に定義する。

$$\backslash\text{pxrr@pre}\langle X \rangle\{\langle Y \rangle\}\backslash\text{pxrr@post}$$

```

439 \def\pxrr@zip@single#1#2{%
440   \expandafter\pxrr@zip@single@a\expandafter#1#2\pxrr@end
441 }
442 \def\pxrr@zip@single@a#1{%
443   \expandafter\pxrr@zip@single@b#1\pxrr@end
444 }
445 \def\pxrr@zip@single@b#1\pxrr@end#2\pxrr@end{%
446   \def\pxrr@res{\pxrr@pre{#1}\pxrr@post}%
447 }

```

`\pxrr@tzip@single` `\pxrr@tzip@single\CSa\CSb\CSc` :

$$\backslash\text{CSa} = \langle X \rangle; \backslash\text{CSb} = \langle Y \rangle; \backslash\text{CSc} = \langle Z \rangle$$

の時に、`\pxrr@res` を以下の内容に定義する。

$$\backslash\text{pxrr@pre}\langle X \rangle\{\langle Y \rangle\}\{\langle Z \rangle\}\backslash\text{pxrr@post}$$

```

448 \def\pxrr@tzip@single#1#2#3{%
449   \expandafter\pxrr@tzip@single@a\expandafter#1\expandafter#2#3\pxrr@end
450 }
451 \def\pxrr@tzip@single@a#1#2{%
452   \expandafter\pxrr@tzip@single@b\expandafter#1#2\pxrr@end
453 }
454 \def\pxrr@tzip@single@b#1{%
455   \expandafter\pxrr@tzip@single@c#1\pxrr@end
456 }
457 \def\pxrr@tzip@single@c#1\pxrr@end#2\pxrr@end#3\pxrr@end{%
458   \def\pxrr@res{\pxrr@pre{#1}\pxrr@post}%
459 }

```

## 4.6 エンジン依存処理

この小節のマクロ内で使われる変数。

```
460 \let\pxrr@x@tempa\@empty
```

```

461 \let\pxrr@x@tempb\@empty
462 \let\pxrr@x@gtempa\@empty
463 \newif\ifpxrr@x@swa

```

`\pxrr@ifprimitive` `\pxrr@ifprimitive\CS{真}\偽}` : `\CS` の現在の定義が同名のプリミティブであるかをテストする。

```

464 \def\pxrr@ifprimitive#1{%
465   \edef\pxrr@x@tempa{\string#1}%
466   \edef\pxrr@x@tempb{\meaning#1}%
467   \ifx\pxrr@x@tempa\pxrr@x@tempb \expandafter\@firstoftwo
468   \else \expandafter\@secondoftwo
469   \fi
470 }

```

`\ifpxrr@in@ptex` エンジンが pTeX 系 (upTeX 系を含む) であるか。 `\kansuji` のプリミティブテストで判定する。

```

471 \pxrr@ifprimitive\kansuji{%
472   \pxrr@csletcs{ifpxrr@in@ptex}{iftrue}%
473 }{%
474   \pxrr@csletcs{ifpxrr@in@ptex}{iffalse}%
475 }

```

`\ifpxrr@in@uptex` エンジンが upTeX 系であるか。 `\enablecjktoken` のプリミティブテストで判定する。

```

476 \pxrr@ifprimitive\enablecjktoken{%
477   \pxrr@csletcs{ifpxrr@in@uptex}{iftrue}%
478 }{%
479   \pxrr@csletcs{ifpxrr@in@uptex}{iffalse}%
480 }

```

`\ifpxrr@in@xetex` エンジンが XeTeX 系であるか。 `\XeTeXrevision` のプリミティブテストで判定する。

```

481 \pxrr@ifprimitive\XeTeXrevision{%
482   \pxrr@csletcs{ifpxrr@in@xetex}{iftrue}%
483 }{%
484   \pxrr@csletcs{ifpxrr@in@xetex}{iffalse}%
485 }

```

`\ifpxrr@in@xecjk` xeCJK パッケージが使用されているか。

```

486 \@ifpackageloaded{xeCJK}{%
487   \pxrr@csletcs{ifpxrr@in@xecjk}{iftrue}%
488 }{%
489   \pxrr@csletcs{ifpxrr@in@xecjk}{iffalse}%

```

ここで未読込でかつプリアンブル末尾で読み込まれている場合は警告する。

```

490   \AtBeginDocument{%
491     \@ifpackageloaded{xeCJK}{%
492       \pxrr@warn@load@order{xeCJK}%
493     }{}%
494   }%
495 }

```

`\ifpxrr@in@luatex` エンジンが LuaTeX 系であるか。 `\luatexrevision` のプリミティブテストで判定する。

```
496 \pxrr@ifprimitive\luatexrevision{%
497   \pxrr@csletcs{ifpxrr@in@luatex}{iftrue}%
498 }{%
499   \pxrr@csletcs{ifpxrr@in@luatex}{iffalse}%
500 }
```

LuaTeX エンジンの場合、本パッケージ用の Lua モジュール `pxrubtica` を作成しておく。

```
501 \ifpxrr@in@luatex
502   \directlua{ pxrubtica = {} }
503 \fi
```

`\ifpxrr@in@luatexja` LuaTeX-ja パッケージが使用されているか。

```
504 \@ifpackageloaded{luatexja-core}{%
505   \pxrr@csletcs{ifpxrr@in@luatexja}{iftrue}%
506 }{%
507   \pxrr@csletcs{ifpxrr@in@luatexja}{iffalse}%
508   \AtBeginDocument{%
509     \@ifpackageloaded{luatexja-core}{%
510       \pxrr@warn@load@order{LuaTeX-ja}%
511     }{%
512     }%
513   }

514 \ifpxrr@in@xetex
515 \else\ifpxrr@in@luatex
516 \else\ifpxrr@in@ptex
517 \else
518   \pxrr@ifprimitive\pdftexrevision{%
519     \pxrr@warn{%
520       The engine in use seems to be pdfTeX,\MessageBreak
521       so safe mode is turned on%
522     }%
523     \AtEndOfPackage{%
524       \rubysafemode
525     }%
526   }
527 \fi\fi\fi
```

`\ifpxrr@in@unicode` 「和文」内部コードが Unicode であるか。

```
528 \ifpxrr@in@xetex
529   \pxrr@csletcs{ifpxrr@in@unicode}{iftrue}%
530 \else\ifpxrr@in@luatex
531   \pxrr@csletcs{ifpxrr@in@unicode}{iftrue}%
532 \else\ifpxrr@in@uptex
533   \pxrr@csletcs{ifpxrr@in@unicode}{iftrue}%
534 \else
535   \pxrr@csletcs{ifpxrr@in@unicode}{iffalse}%
536 \fi\fi\fi
```

`\pxrr@jc` 和文の「複合コード」を内部コードに変換する（展開可能）。「複合コード」は「〈JIS コード 16 進 4 桁〉:〈Unicode 16 進 4 桁〉」の形式。

```
537 \def\pxrr@jc#1{%
538   \pxrr@jc@a#1\pxrr@nil
539 }
540 \ifpxrr@in@unicode
541   \def\pxrr@jc@a#1:#2\pxrr@nil{%
542     "#2\space
543   }
544 \else\ifpxrr@in@ptex
545   \def\pxrr@jc@a#1:#2\pxrr@nil{%
546     \jis"#1\space\space
547   }
548 \else
549   \def\pxrr@jc@a#1:#2\pxrr@nil{%
550     '?\space
551   }
552 \fi\fi
```

`\pxrr@jchardef` 和文用の `\chardef`。

```
553 \ifpxrr@in@uptex
554   \let\pxrr@jchardef\kchardef
555 \else
556   \let\pxrr@jchardef\chardef
557 \fi
```

`\pxrr@if@in@tate` `\pxrr@if@in@tate{〈真〉}{〈偽〉}`：縦組であるか。

```
558 \ifpxrr@in@ptex
```

p<sub>T</sub>E<sub>X</sub> 系の場合、`\iftdir` プリミティブを利用する。

※ `\iftdir` が未定義のときに `if` が不均衡になるのを防ぐ。

※ 本パッケージの処理の範囲では、縦数式組方向は単に「縦組でない」と判定する。（`\ifmdir` は数式組方向を判定するプリミティブ。）

```
559   \begingroup \catcode'\|=0
560   \gdef\pxrr@if@in@tate{%
561     \pxrr@cond{\if
562       |iftdir|ifmdir F|else T|fi|else F|fi
563       T}\fi
564   }
565   \endgroup
566 \else\ifpxrr@in@luatexja
```

Lua<sub>T</sub>E<sub>X</sub>-ja 利用の場合、`direction` パラメタを利用する。

※ 縦組対応（`\ltj@curtfnt` が定義済）でない古い Lua<sub>T</sub>E<sub>X</sub>-ja の場合は常に横組と見なす。

```
567   \ifx\ltj@curtfnt\@undefined
568     \let\pxrr@if@in@tate\@secondoftwo
569   \else
570     \def\pxrr@if@in@tate{%
```

```

571     \pxrr@cond\ifnum\ltjgetparameter{direction}=\thr@@\fi
572   }
573   \fi
574 \else

それ以外は常に横組と見なす。

575   \let\pxrr@if@in@tate\@secondoftwo
576 \fi\fi

```

`\pxrr@get@jchar@token` `\pxrr@get@jchar@token\CS{⟨整数⟩}`: 内部文字コードが⟨整数⟩である和文文字のトークンを得る。

※ `.sty` ファイルは完全に ASCII 文字だけにする方針のため、和文文字が必要な場合はこの補助マクロや `\pxrr@jchardef` を利用して複合コード値から作り出すことになる。

pTeX 系の場合、`\kansuji` トリックを利用する。

```

577 \ifpxrr@in@ptex
578   \def\pxrr@get@jchar@token#1#2{%
579     \begingroup
580     \kansujichar\@ne=#2\relax
581     \xdef\pxrr@x@gtempa{\kansuji\@ne}%
582   \endgroup
583   \let#1\pxrr@x@gtempa
584 }

```

Unicode 対応 TeX の場合、`\lowercase` トリックを利用する。

```

585 \else\ifpxrr@in@unicode
586   \def\pxrr@get@jchar@token#1#2{%
587     \begingroup
588     \lccode'\?=#2\relax
589     \lowercase{\xdef\pxrr@x@gtempa{?}}%
590   \endgroup
591   \let#1\pxrr@x@gtempa
592 }

```

それ以外ではダミー定義。

```

593 \else
594   \def\pxrr@get@jchar@token#1#2{%
595     \def#1{?}%
596   }
597 \fi\fi

```

`\pxrr@zspace` 全角空白文字。文字そのものをファイルに含ませたくないなので `chardef` にする。

```

598 \pxrr@jchardef\pxrr@zspace=\pxrr@jc{2121:3000}

```

`\pxrr@jghost@char` 和文ゴースト処理に利用する文字。字形が空であり、かつ一般の漢字と同じ挙動を示す必要がある。実際のゴースト処理では字幅を相殺する処理を入れる為、字幅がゼロである必要はない。

ほとんどの場合、全角空白文字で構わないが、全角空白文字が文字タイプ 0 でない JFM が使われている場合は問題になる。

upTeX の場合、“拡張符号空間”の文字コードを使う。すなわち、文字コード "113000 の文字は DVI では文字コード "3000 と扱われるが、“BMP 外”にあるため必ず文字タイプ 0 になる。

```
599 \ifpxrr@in@uptex
600 \kchardef\pxrr@jghost@char="113000
```

LuaTeX-ja の場合。文書先頭で“全角空白文字が使えるか”を検査して、失敗した場合は「和文の U+00A0」を代わりに利用することにする。

```
601 \else\ifpxrr@in@luatexja
602 \let\pxrr@jghost@char\pxrr@zspace
603 \def\pxrr@jghost@check{%
604   \begingroup
605 %     \ltjsetparameter{jaxspmode={\pxrr@zspace,3}}%
606 %     \ltjsetparameter{xkanjiskip=\p@}%
607 %     \ltjsetparameter{autoxspacing=false}%
608     \setbox\z@\hbox{\char"3001\char"3000}%
609 %     \ltjsetparameter{autoxspacing=true}%
610     \setbox\tw@\hbox{\char"3001\inhibitglue\char"3000}%
611     \ifdim\wd\tw@=\wd\z@
612       \global\chardef\pxrr@jghost@char="00A0
613       \gdef\pxrr@jghost@char{\ltjjachar\pxrr@jghost@char}%
614     \fi
615   \endgroup
616 }
617 \AtBeginDocument{%
618   \pxrr@jghost@check
619 }
```

それ以外の場合は（仕方が無いので）全角空白を用いる。

```
620 \else
621 \let\pxrr@jghost@char\pxrr@zspace
622 \fi\fi
```

`\pxrr@x@K` 適当な漢字（実際は <一>）のトークン。

```
623 \pxrr@jchardef\pxrr@x@K=\pxrr@jc{306C:4E00}
```

`\pxrr@get@iiskip` `\pxrr@get@iiskip\CS`：現在の実効の和文間空白の量を取得する。  
pTeX 系の場合。

```
624 \ifpxrr@in@ptex
625 \def\pxrr@get@iiskip#1{%
```

以下では `\kanjiskip` 挿入が有効であるかを検査している。

```
626   \pxrr@x@swafalse
627   \begingroup
628     \inhibitxspcode\pxrr@x@K\thr@@
629     \kanjiskip\p@
630     \setbox\z@\hbox{\noautospace\pxrr@x@K\pxrr@x@K}%
631     \setbox\tw@\hbox{\pxrr@x@K\pxrr@x@K}%
632     \ifdim\wd\tw@>\wd\z@
```

```

633     \aftergroup\pxrr@x@swatruue
634     \fi
635     \endgroup

```

以下では `\kanjiskip` 挿入が有効ならば `\kanjiskip` の値、無効ならばゼロを返す。

```

636     \edef#1{%
637         \ifpxrr@x@swa \the\kanjiskip
638         \else \pxrr@zeropt
639         \fi
640     }%
641 }

```

LuaTeX-ja 使用の場合。

```

642 \else\ifpxrr@in@luatexja
643     \def\pxrr@get@iiskip#1{%
644         \ifnum\ltjgetparameter{autospacing}=\@ne
645             \xdef\pxrr@x@gtempa{\ltjgetparameter{kanjiskip}}%
646             \ifdim\glueexpr\pxrr@x@gtempa=\maxdimen

```

`kanjiskip` パラメタの値が `\maxdimen` の場合、JFM のパラメタにより和欧文間空白の量が決定される。この値を読み出す公式のインタフェースは存在しないため、実際の組版結果から推定する。(値は `\pxrr@x@gtempa` に返る。)

```

647         \pxrr@get@interchar@glue{\pxrr@x@K\pxrr@x@K}%
648         \ifdim\glueexpr\pxrr@x@gtempa=\maxdimen

```

推定が失敗した場合。警告を (一度だけ) 出した上で、値をゼロとして扱う。

```

649         \pxrr@warn@unknown@iiskip
650         \global\let\pxrr@x@gtempa\pxrr@zeropt
651         \fi
652         \fi
653         \let#1\pxrr@x@gtempa
654     \else
655         \let#1\pxrr@zeropt
656     \fi
657 }

```

和文間空白の推定に失敗した場合の警告。

```

658     \def\pxrr@warn@unknown@iiskip{%
659         \global\let\pxrr@warn@unknown@iiskip\relax
660         \pxrr@warn{Cannot find the kanjiskip value}%
661     }

```

テキスト #1 を組版した水平ボックスの中にある、“文字間グルー”の値を `\pxrr@g@tempa` に返す。

```

662     \def\pxrr@get@interchar@glue#1{%
663         \begingroup
664         \setbox\z@\hbox{#1}%

```

Lua の補助関数は所望の値を `\skip0` に返す。失敗時の検出のため、このレジスタを `\maxdimen` で初期化する。

```

665     \skip\z@\maxdimen\relax
666     \directlua{%
667         pcall(pxrubrica._get_interchar_glue)
668     }%
669     \xdef\pxrr@x@gtempa{\the\skip\z@}%
670 \endgroup
671 }
672 \begingroup
673 \endlinechar=10 \directlua{%
674     local node, tex = node, tex
675     local id_glyph, id_glue = node.id("glyph"), node.id("glue")
676     local id_hlist = node.id("hlist")

```

`_get_interchar_glue()` は `\box0` の“文字間グルー”の量を取得し、`\skip0` に代入する。実際には、「最初の `glyph` ノードの後にある最初の `glue` ノードを“文字間グルー”と判断し、その量を読み出す。

```

677     function pxrubrica._get_interchar_glue()
678         local c, n = false, tex.box[0].head
679         while n do

```

※ 2014 年頃の LuaTeX-ja では文字の部分が `hlist` ノードになっている。

```

680         if n.id == id_glyph or n.id == id_hlist then
681             c = true
682         elseif c and n.id == id_glue then

```

ここでの `n` が“文字間グルー”のノードである。

※ 0.85 版以降の LuaTeX では、`glue` ノードに直接値 (`n.width` 等) が入っている。それより古い版では、`glue_spec` データを介したインタフェースになっている。

```

683         if n.width then
684             tex.setglue(0, n.width, n.stretch, n.shrink,
685                 n.stretch_order, n.shrink_order)
686         elseif n.spec then
687             tex.setskip(0, node.copy(n.spec))
688         end
689         break
690     end
691     n = n.next
692 end
693 end
694 }%
695 \endgroup%

```

それ以外の場合はゼロとする。

```

696 \else
697 \def\pxrr@get@iiskip#1{%
698     \let#1\pxrr@zeropt
699 }
700 \fi\fi

```

`\pxrr@get@iaiskip` `\pxrr@get@iaiskip\CS` : 現在の実効の和欧文間空白の量を取得する。

pTeX 系の場合。

```
701 \ifpxrr@in@ptex
702   \def\pxrr@get@iaiskip#1{%
703     \pxrr@x@swafalse
704     \begingroup
705       \inhibitxspcode\pxrr@x@K\thr@@ \xspcode'X=\thr@@
706       \xkanjiskip\p@
707       \setbox\z@\hbox{\noautoxspacing\pxrr@x@K X}%
708       \setbox\tw@\hbox{\pxrr@x@K X}%
709       \ifdim\wd\tw@>\wd\z@
710         \aftergroup\pxrr@x@swatruue
711       \fi
712     \endgroup
713   \edef#1{%
714     \ifpxrr@x@swa \the\xkanjiskip
715     \else \pxrr@zeropt
716     \fi
717   }%
718 }
```

LuaTeX-ja 使用の場合。処理の流れは和文間空白の場合と同じ。

```
719 \else\ifpxrr@in@luatexja
720   \def\pxrr@get@iaiskip#1{%
721     \ifnum\ltjgetparameter{autoxspacing}=\@ne
722       \xdef\pxrr@x@gtempa{\ltjgetparameter{xkanjiskip}}%
723       \ifdim\glueexpr\pxrr@x@gtempa=\maxdimen
```

判定用のボックスは欧文・和文の組とする。

```
724       \pxrr@get@interchar@glue{A\pxrr@x@K}%
725       \ifdim\glueexpr\pxrr@x@gtempa=\maxdimen
726         \pxrr@warn@unknown@iaiskip
727         \global\let\pxrr@x@gtempa\pxrr@zeropt
728       \fi
729     \fi
730     \let#1\pxrr@x@gtempa
731   \else
732     \let#1\pxrr@zeropt
733   \fi
734 }
```

和欧文間空白の推定に失敗した場合の警告。

```
735   \def\pxrr@warn@unknown@iaiskip{%
736     \global\let\pxrr@warn@unknown@iaiskip\relax
737     \pxrr@warn{Cannot find the xkanjiskip value}%
738   }
```

それ以外の場合は実際の組版結果から判断する。

```
739 \else
740   \def\pxrr@get@iaiskip#1{%
741     \begingroup
```

```

742     \setbox\z@\hbox{M\pxrr@x@K}%
743     \setbox\tw@\hbox{M\vrule\@width\z@\relax\pxrr@x@K}%
744     \@tempdima\wd\z@ \advance\@tempdima-\wd\tw@
745     \@tempdimb\@tempdima \divide\@tempdimb\thr@@
746     \xdef\pxrr@x@gtempa{\the\@tempdima\space minus \the\@tempdimb}%
747   \endgroup
748   \let#1=\pxrr@x@gtempa
749 }%
750 \fi\fi

```

`\pxrr@get@zwidth` `\pxrr@get@zwidth\CS` : 現在の和文フォントの全角幅を取得する。  
`pTeX` の場合、`1zw` でよい。

```

751 \ifpxrr@in@ptex
752   \def\pxrr@get@zwidth#1{%
753     \@tempdima=1zw\relax
754     \edef#1{\the\@tempdima}%
755   }

```

`\zw` が定義されている場合は `1\zw` とする。

```

756 \else\if\ifx\zw\@undefined T\else F\fi F% if defined
757   \def\pxrr@get@zwidth#1{%
758     \@tempdima=1\zw\relax
759     \edef#1{\the\@tempdima}%
760   }

```

`\jsZw` が定義されている場合は `1\jsZw` とする。

```

761 \else\if\ifx\jsZw\@undefined T\else F\fi F% if defined
762   \def\pxrr@get@zwidth#1{%
763     \@tempdima=1\jsZw\relax
764     \edef#1{\the\@tempdima}%
765   }

```

それ以外で、`\pxrr@x@K` が有効な場合は実際の組版結果から判断する。

```

766 \else\ifnum\pxrr@x@K>\@cclv
767   \def\pxrr@get@zwidth#1{%
768     \setbox\tw@\hbox{\pxrr@x@K}%
769     \@tempdima\wd\tw@
770     \ifdim\@tempdima>\z@\else \@tempdima\@size\p@ \fi
771     \edef#1{\the\@tempdima}%
772   }

```

それ以外の場合は要求サイズと等しいとする。

```

773 \else
774   \def\pxrr@get@zwidth#1{%
775     \@tempdima\@size\p@\relax
776     \edef#1{\the\@tempdima}%
777   }
778 \fi\fi\fi\fi

```

`\pxrr@get@prebreakpenalty` `\pxrr@get@prebreakpenalty\CS{(文字コード)}` : 文字の後禁則ペナルティ値を整数レジスタに代入する。

pTeX の場合、`\prebreakpenalty` を使う。

```
779 \ifpxrr@in@ptex
780   \def\pxrr@get@prebreakpenalty#1#2{%
781     #1=\prebreakpenalty#2\relax
782   }
```

LuaTeX-japan 使用時は、`prebreakpenalty` プロパティを読み出す。

```
783 \else\ifpxrr@in@luatexja
784   \def\pxrr@get@prebreakpenalty#1#2{%
785     #1=\ltjgetparameter{prebreakpenalty}{#2}\relax
786   }
```

それ以外の場合はゼロとして扱う。

```
787 \else
788   \def\pxrr@get@prebreakpenalty#1#2{%
789     #1=\z@
790   }
791 \fi\fi
```

`\pxrr@get@postbreakpenalty <文字コード>` : 文字の前禁則ペナルティ値を整数レジスタに代入する。

pTeX の場合、`\postbreakpenalty` を使う。

```
792 \ifpxrr@in@ptex
793   \def\pxrr@get@postbreakpenalty#1#2{%
794     #1=\postbreakpenalty#2\relax
795   }
```

LuaTeX-japan 使用時は、`postbreakpenalty` プロパティを読み出す。

```
796 \else\ifpxrr@in@luatexja
797   \def\pxrr@get@postbreakpenalty#1#2{%
798     #1=\ltjgetparameter{postbreakpenalty}{#2}\relax
799   }
```

それ以外の場合はゼロとして扱う。

```
800 \else
801   \def\pxrr@get@postbreakpenalty#1#2{%
802     #1=\z@
803   }
804 \fi\fi
```

`\pxrr@check@punct@char <文字コード>` : 指定の文字コードの文字が“約物であるか”を調べて、結果を `\ifpxrr@ok` に返す。(和文フラグ)は“対象が pTeX の和文である”場合に 1、それ以外は 0。

pTeX の場合、欧文なら `\xspcode`、和文なら `\inhibitxspcode` の値を見て、それが 3 以外なら約物と見なす。

```
805 \ifpxrr@in@ptex
806   \def\pxrr@check@punct@char#1#2{%
807     \pxrr@okfalse
808     \ifcase#2\relax
```

```

809     \ifnum\xspcode#1=\thr@@\else
810     \pxrr@oktrue
811     \fi
812 \else
813     \ifnum\inhibitxspcode#1=\thr@@\else
814     \pxrr@oktrue
815     \fi
816 \fi
817 }

```

LuaTeX-ja 使用時も基本的に pTeX と同じロジックを使う。ただし LuaTeX-ja では「文字トークンの和文と欧文の区別」という概念が存在しないため、〈和文フラグ〉は必ず 0 となる。そして、`\xspcode`/`\inhibitxspcode` に相当するパラメタとしては、欧文用の `alxspmode` と和文用の `jaxspmode` が一応あるが、実際には和文と欧文の区別はなくこの両者は同義になっている。従って、「`jaxspmode` が 3 以外か」を調べることにする。

```

818 \else\ifpxrr@in@luatexja
819   \def\pxrr@check@punct@char#1#2{%
820     \ifnum\ltjgetparameter{jaxspmode}{#1}=\thr@@
821     \pxrr@okfalse
822   \else
823     \pxrr@oktrue
824   \fi
825 }

```

それ以外の場合はゼロとして扱う。

```

826 \else
827   \def\pxrr@check@punct@char#1#2{%
828     \pxrr@okfalse
829   }
830 \fi\fi

```

`\pxrr@inhibitglue` `\inhibitglue` が定義されているなら実行する。

```

831 \ifx\inhibitglue\undefined
832   \let\pxrr@inhibitglue\relax
833 \else
834   \let\pxrr@inhibitglue\inhibitglue
835 \fi

```

#### 4.7 パラメタ設定公開命令

`\ifpxrr@in@setup` `\pxrr@parse@option` が `\rubyssetup` の中で呼ばれたか。真の場合は警告処理を行わない。

```

836 \newif\ifpxrr@in@setup \pxrr@in@setupfalse

```

`\rubyssetup` `\pxrr@parse@option` で解析した後、設定値を全般設定にコピーする。

```

837 \newcommand*\rubyssetup[1]{%
838   \pxrr@in@setuptrue
839   \pxrr@fatal@errorfalse
840   \pxrr@parse@option{#1}%

```

```

841 \ifpxrr@fatal@error\else
842 \pxrr@csletcs{ifpxrr@d@bprotr}{ifpxrr@bprotr}%
843 \pxrr@csletcs{ifpxrr@d@aprotr}{ifpxrr@aprotr}%
844 \let\pxrr@d@bintr\pxrr@bintr@
845 \let\pxrr@d@aintr\pxrr@aintr@
846 \let\pxrr@d@athead\pxrr@athead
847 \let\pxrr@d@mode\pxrr@mode
848 \let\pxrr@d@side\pxrr@side
849 \let\pxrr@d@evensp\pxrr@evensp
850 \let\pxrr@d@fullsize\pxrr@fullsize
851 \fi

```

\ifpxrr@in@setup を偽に戻す。ただし \ifpxrr@fatal@error は書き換えられたままであることに注意。

```

852 \pxrr@in@setupfalse
853 }

```

\rubyfontsetup 対応するパラメタを設定する。

```

854 \newcommand*\rubyfontsetup{}
855 \def\rubyfontsetup#1{%
856 \def\pxrr@ruby@font
857 }

```

\rubybigintrusion 対応するパラメタを設定する。

```

\rubysmallintrusion 858 \newcommand*\rubybigintrusion[1]{%
\rubymaxmargin 859 \edef\pxrr@big@intr{#1}%
860 }
\rubyintergap 861 \newcommand*\rubysmallintrusion[1]{%
\rubysizeratio 862 \edef\pxrr@small@intr{#1}%
863 }
864 \newcommand*\rubymaxmargin[1]{%
865 \edef\pxrr@maxmargin{#1}%
866 }
867 \newcommand*\rubyintergap[1]{%
868 \edef\pxrr@inter@gap{#1}%
869 }
870 \newcommand*\rubysizeratio[1]{%
871 \edef\pxrr@size@ratio{#1}%
872 }

```

\rubyusejghost 対応するスイッチを設定する。

```

\rubynousejghost 873 \newcommand*\rubyusejghost{%
874 \pxrr@jghosttrue
875 }
876 \newcommand*\rubynousejghost{%
877 \pxrr@jghostfalse
878 }

```

\rubyuseaghost 対応するスイッチを設定する。

\rubynouseaghost

```

879 \newcommand*\rubyuseaghost{%
880   \pxrr@aghosttrue
881   \pxrr@setup@aghost
882 }
883 \newcommand*\rubynouseaghost{%
884   \pxrr@aghostfalse
885 }

```

`\rubyadjustatlineedge` 対応するスイッチを設定する。

```

\rubynoadjustatlineedge 886 \newcommand*\rubyadjustatlineedge{%
887   \pxrr@edge@adjusttrue
888 }
889 \newcommand*\rubynoadjustatlineedge{%
890   \pxrr@edge@adjustfalse
891 }

```

`\rubybreakjukugo` 対応するスイッチを設定する。

```

\rubynobreakjukugo 892 \newcommand*\rubybreakjukugo{%
893   \pxrr@break@jukugotrue
894 }
895 \newcommand*\rubynobreakjukugo{%
896   \pxrr@break@jukugofalse
897 }

```

`\rubysafemode` 対応するスイッチを設定する。

```

\rubynosafemode 898 \newcommand*\rubysafemode{%
899   \pxrr@safe@modetrue
900 }
901 \newcommand*\rubynosafemode{%
902   \pxrr@safe@modefalse
903 }

```

`\rubystretchprop` 対応するパラメタを設定する。

```

\rubystretchprophead 904 \newcommand*\rubystretchprop[3]{%
\rubystretchpropend 905   \edef\pxrr@sprop@x{#1}%
906   \edef\pxrr@sprop@y{#2}%
907   \edef\pxrr@sprop@z{#3}%
908 }
909 \newcommand*\rubystretchprophead[2]{%
910   \edef\pxrr@sprop@hy{#1}%
911   \edef\pxrr@sprop@hz{#2}%
912 }
913 \newcommand*\rubystretchpropend[2]{%
914   \edef\pxrr@sprop@ex{#1}%
915   \edef\pxrr@sprop@ey{#2}%
916 }

```

`\rubyuseextra` 残念ながら今のところは使用不可。

```

917 \newcommand*\rubyuseextra[1]{%

```

```

918 \pxrr@cнта=#1\relax
919 \ifnum\pxrr@cнта=\z@
920   \chardef\pxrr@extra\pxrr@cнта
921 \else
922   \pxrr@err@inv@value{\the\pxrr@cнта}%
923 \fi
924 }

```

## 4.8 ルビオプション解析

`\pxrr@bintr@` オプション解析中にのみ使われ、進入の値を `\pxrr@d@?intr` と同じ形式で保持する。  
`\pxrr@aintr@` (`\pxrr@?intr` は形式が異なることに注意。)

```

925 \let\pxrr@bintr@\@empty
926 \let\pxrr@aintr@\@empty

```

`\pxrr@doublebar` `\pxrr@parse@option` 中で使用される。

```

927 \def\pxrr@doublebar{||}

```

`\pxrr@parse@option` `\pxrr@parse@option{〈オプション〉}`: 〈オプション〉を解析し、`\pxrr@athead` や `\pxrr@mode` 等のパラメタを設定する。

```

928 \def\pxrr@parse@option#1{%

```

入力が「||」の場合は、「|-|」に置き換える。

```

929 \edef\pxrr@tempa{#1}%
930 \ifx\pxrr@tempa\pxrr@doublebar
931   \def\pxrr@tempa{|-|}%
932 \fi

```

各パラメタの値を全般設定のもので初期化する。

```

933 \pxrr@csletcs{ifpxrr@bprotr}{ifpxrr@d@bprotr}%
934 \pxrr@csletcs{ifpxrr@aprotr}{ifpxrr@d@aprotr}%
935 \let\pxrr@bintr@\pxrr@d@bintr
936 \let\pxrr@aintr@\pxrr@d@aintr
937 \let\pxrr@athead\pxrr@d@athead
938 \let\pxrr@mode\pxrr@d@mode
939 \let\pxrr@side\pxrr@d@side
940 \let\pxrr@evensp\pxrr@d@evensp
941 \let\pxrr@fullsize\pxrr@d@fullsize

```

以下のパラメタの既定値は固定されている。

```

942 \let\pxrr@bscomp\relax
943 \let\pxrr@ascomp\relax
944 \pxrr@bnobrfalse
945 \pxrr@anobrfalse
946 \pxrr@bfintrfalse
947 \pxrr@afintrfalse

```

明示フラグを偽にする。

```

948 \pxrr@mode@givenfalse

```

```
949 \pxrr@athead@givenfalse
```

両側ルビの場合、基本モード既定値が M に固定される。

```
950 \ifpxrr@truby
951 \let\pxrr@mode=M%
952 \fi
```

有限状態機械を開始させる。入力の末尾に @ を加えている。 \pxrr@end はエラー時の脱出に用いる。

```
953 \def\pxrr@po@FS{bi}%
954 \expandafter\pxrr@parse@option@loop\pxrr@tempa @\pxrr@end
955 }
```

有限状態機械のループ。

```
956 \def\pxrr@parse@option@loop#1{%
957 \ifpxrrDebug
958 \typeout{\pxrr@po@FS/#1[\@nameuse{pxrr@po@C@#1}]}%
959 \fi
960 \csname pxrr@po@PR@#1\endcsname
961 \expandafter\ifx\csname pxrr@po@C@#1\endcsname\relax
962 \let\pxrr@po@FS\relax
963 \else
964 \pxrr@letcs\pxrr@po@FS
965 {\pxrr@po@TR@\pxrr@po@FS @\@nameuse{pxrr@po@C@#1}]}%
966 \fi
967 \ifpxrrDebug
968 \typeout{->\pxrr@po@FS}%
969 \fi
970 \pxrr@ifx{\pxrr@po@FS\relax}{-%
971 \pxrr@fatal@unx@letter{#1}%
972 \pxrr@parse@option@exit
973 }{-%
974 \pxrr@parse@option@loop
975 }%
976 }
```

後処理。

```
977 \def\pxrr@parse@option@exit#1\pxrr@end{%
```

既定値設定 (\rubyssetup) である場合もしない。

```
978 \ifpxrr@in@setup\else
```

両側ルビ命令の場合は、 \pxrr@side の値を変更する。

```
979 \ifpxrr@truby
980 \chardef\pxrr@side\tw@
981 \fi
```

整合性検査を行う。

```
982 \pxrr@check@option
```

\pxrr@?intr の値を設定する。

```

983 \tempdima=\pxrr@ruby@zw\relax
984 \tempdimb=\pxrr@or@zero\pxrr@bintr@\tempdima
985 \edef\pxrr@bintr{\the\tempdimb}%
986 \tempdimb=\pxrr@or@zero\pxrr@aintr@\tempdima
987 \edef\pxrr@aintr{\the\tempdimb}%
988 \fi
989 }

```

\pxrr@or@zero \pxrr@or@zero\pxrr@?intr@ とすると、\pxrr@?intr@ が空の時に代わりにゼロと扱う。

```

990 \def\pxrr@or@zero#1{%
991 \ifx#1\@empty \pxrr@zero
992 \else #1%
993 \fi
994 }

```

以下はオプション解析の有限状態機械の定義。

記号のクラスの設定。

```

995 \def\pxrr@po@C@{F}
996 \@namedef{pxrr@po@C@|}{V}
997 \@namedef{pxrr@po@C@:}{S}
998 \@namedef{pxrr@po@C@.}{S}
999 \@namedef{pxrr@po@C@*}{S}
1000 \@namedef{pxrr@po@C@!}{S}
1001 \@namedef{pxrr@po@C@<}{B}
1002 \@namedef{pxrr@po@C@()}{B}
1003 \@namedef{pxrr@po@C@>}{A}
1004 \@namedef{pxrr@po@C@)}{A}
1005 \@namedef{pxrr@po@C@-}{M}
1006 \def\pxrr@po@C@c{M}
1007 \def\pxrr@po@C@h{M}
1008 \def\pxrr@po@C@H{M}
1009 \def\pxrr@po@C@m{M}
1010 \def\pxrr@po@C@g{M}
1011 \def\pxrr@po@C@j{M}
1012 \def\pxrr@po@C@M{M}
1013 \def\pxrr@po@C@J{M}
1014 \def\pxrr@po@C@P{M}
1015 \def\pxrr@po@C@S{M}
1016 \def\pxrr@po@C@e{M}
1017 \def\pxrr@po@C@E{M}
1018 \def\pxrr@po@C@f{M}
1019 \def\pxrr@po@C@F{M}

```

機能プロセス。

```

1020 \def\pxrr@po@PR@{f%
1021 \pxrr@parse@option@exit
1022 }
1023 \@namedef{pxrr@po@PR@|}{f%
1024 \csname pxrr@po@PRbar@\pxrr@po@FS\endcsname

```

```

1025 }
1026 \def\pxrr@po@PRbar@bi{%
1027   \def\pxrr@bintr@{}\pxrr@bprottrue
1028 }
1029 \def\pxrr@po@PRbar@bb{%
1030   \pxrr@bprotrfalse
1031 }
1032 \def\pxrr@po@PRbar@bs{%
1033   \def\pxrr@aintr@{}\pxrr@aprotrtrue
1034 }
1035 \let\pxrr@po@PRbar@mi\pxrr@po@PRbar@bs
1036 \let\pxrr@po@PRbar@as\pxrr@po@PRbar@bs
1037 \let\pxrr@po@PRbar@ai\pxrr@po@PRbar@bs
1038 \def\pxrr@po@PRbar@ab{%
1039   \pxrr@aprotrfalse
1040 }
1041 \@namedef{pxrr@po@PR@:}{%
1042   \csname pxrr@po@PRcolon@\pxrr@po@FS\endcsname
1043 }
1044 \def\pxrr@po@PRcolon@bi{%
1045   \let\pxrr@bscomp=: \relax
1046 }
1047 \let\pxrr@po@PRcolon@bb\pxrr@po@PRcolon@bi
1048 \let\pxrr@po@PRcolon@bs\pxrr@po@PRcolon@bi
1049 \def\pxrr@po@PRcolon@mi{%
1050   \let\pxrr@ascomp=: \relax
1051 }
1052 \let\pxrr@po@PRcolon@as\pxrr@po@PRcolon@mi
1053 \@namedef{pxrr@po@PR@.}{%
1054   \csname pxrr@po@PRdot@\pxrr@po@FS\endcsname
1055 }
1056 \def\pxrr@po@PRdot@bi{%
1057   \let\pxrr@bscomp=. \relax
1058 }
1059 \let\pxrr@po@PRdot@bb\pxrr@po@PRdot@bi
1060 \let\pxrr@po@PRdot@bs\pxrr@po@PRdot@bi
1061 \def\pxrr@po@PRdot@mi{%
1062   \let\pxrr@ascomp=. \relax
1063 }
1064 \let\pxrr@po@PRdot@as\pxrr@po@PRdot@mi
1065 \@namedef{pxrr@po@PR@*}{%
1066   \csname pxrr@po@PRstar@\pxrr@po@FS\endcsname
1067 }
1068 \def\pxrr@po@PRstar@bi{%
1069   \pxrr@bnobrtrue
1070 }
1071 \let\pxrr@po@PRstar@bb\pxrr@po@PRstar@bi
1072 \let\pxrr@po@PRstar@bs\pxrr@po@PRstar@bi
1073 \def\pxrr@po@PRstar@mi{%

```

```

1074 \pxrr@anobrtrue
1075 }
1076 \let\pxrr@po@PRstar@as\pxrr@po@PRstar@mi
1077 \@namedef{pxrr@po@PR@!}{%
1078 \csname pxrr@po@PRbang@\pxrr@po@FS\endcsname
1079 }
1080 \def\pxrr@po@PRbang@bi{%
1081 \pxrr@bfintrtrue
1082 }
1083 \let\pxrr@po@PRbang@bb\pxrr@po@PRbang@bi
1084 \let\pxrr@po@PRbang@bs\pxrr@po@PRbang@bi
1085 \def\pxrr@po@PRbang@mi{%
1086 \pxrr@afintrtrue
1087 }
1088 \let\pxrr@po@PRbang@as\pxrr@po@PRbang@mi
1089 \@namedef{pxrr@po@PR@<}{%
1090 \def\pxrr@bintr@{\pxrr@big@intr}\pxrr@bprottrue
1091 }
1092 \@namedef{pxrr@po@PR@()}{%
1093 \def\pxrr@bintr@{\pxrr@small@intr}\pxrr@bprottrue
1094 }
1095 \@namedef{pxrr@po@PR@>}{%
1096 \def\pxrr@aintr@{\pxrr@big@intr}\pxrr@aprottrue
1097 }
1098 \@namedef{pxrr@po@PR@)}{%
1099 \def\pxrr@aintr@{\pxrr@small@intr}\pxrr@aprottrue
1100 }
1101 \def\pxrr@po@PR@c{%
1102 \chardef\pxrr@athead\z@
1103 \pxrr@athead@giventtrue
1104 }
1105 \def\pxrr@po@PR@h{%
1106 \chardef\pxrr@athead\@ne
1107 \pxrr@athead@giventtrue
1108 }
1109 \def\pxrr@po@PR@H{%
1110 \chardef\pxrr@athead\tw@
1111 \pxrr@athead@giventtrue
1112 }
1113 \def\pxrr@po@PR@m{%
1114 \let\pxrr@mode=m%
1115 \pxrr@mode@giventtrue
1116 }
1117 \def\pxrr@po@PR@g{%
1118 \let\pxrr@mode=g%
1119 \pxrr@mode@giventtrue
1120 }
1121 \def\pxrr@po@PR@j{%
1122 \let\pxrr@mode=j%

```

```

1123 \pxrr@mode@giventruе
1124 }
1125 \def\pxrr@po@PR@M{%
1126 \let\pxrr@mode=M%
1127 \pxrr@mode@giventruе
1128 }
1129 \def\pxrr@po@PR@J{%
1130 \let\pxrr@mode=J%
1131 \pxrr@mode@giventruе
1132 }
1133 \def\pxrr@po@PR@P{%
1134 \chardef\pxrr@side\z@
1135 }
1136 \def\pxrr@po@PR@S{%
1137 \chardef\pxrr@side\@ne
1138 }
1139 \def\pxrr@po@PR@E{%
1140 \chardef\pxrr@evensp\z@
1141 }
1142 \def\pxrr@po@PR@e{%
1143 \chardef\pxrr@evensp\@ne
1144 }
1145 \def\pxrr@po@PR@F{%
1146 \chardef\pxrr@fullsize\z@
1147 }
1148 \def\pxrr@po@PR@f{%
1149 \chardef\pxrr@fullsize\@ne
1150 }

```

遷移表。

```

1151 \def\pxrr@po@TR@bi@F{fi}
1152 \def\pxrr@po@TR@bb@F{fi}
1153 \def\pxrr@po@TR@bs@F{fi}
1154 \def\pxrr@po@TR@mi@F{fi}
1155 \def\pxrr@po@TR@as@F{fi}
1156 \def\pxrr@po@TR@ai@F{fi}
1157 \def\pxrr@po@TR@ab@F{fi}
1158 \def\pxrr@po@TR@fi@F{fi}
1159 \def\pxrr@po@TR@bi@V{bb}
1160 \def\pxrr@po@TR@bb@V{bs}
1161 \def\pxrr@po@TR@bs@V{ab}
1162 \def\pxrr@po@TR@mi@V{ab}
1163 \def\pxrr@po@TR@as@V{ab}
1164 \def\pxrr@po@TR@ai@V{ab}
1165 \def\pxrr@po@TR@ab@V{fi}
1166 \def\pxrr@po@TR@bi@S{bs}
1167 \def\pxrr@po@TR@bb@S{bs}
1168 \def\pxrr@po@TR@bs@S{bs}
1169 \def\pxrr@po@TR@mi@S{as}

```

```

1170 \def\pxrr@po@TR@as@S{as}
1171 \def\pxrr@po@TR@bi@B{bs}
1172 \def\pxrr@po@TR@bi@M{mi}
1173 \def\pxrr@po@TR@bb@M{mi}
1174 \def\pxrr@po@TR@bs@M{mi}
1175 \def\pxrr@po@TR@mi@M{mi}
1176 \def\pxrr@po@TR@bi@A{fi}
1177 \def\pxrr@po@TR@bb@A{fi}
1178 \def\pxrr@po@TR@bs@A{fi}
1179 \def\pxrr@po@TR@mi@A{fi}
1180 \def\pxrr@po@TR@as@A{fi}
1181 \def\pxrr@po@TR@ai@A{fi}

```

#### 4.9 オプション整合性検査

`\pxrr@mode@grand` 基本モードの“大分類”。モノ (m)・熟語 (j)・グループ (g) の何れか。つまり“選択的”設定の M・J を m・j に寄せる。

※ 完全展開可能であるが、“先頭完全展開可能”でないことに注意。

```

1182 \def\pxrr@mode@grand{%
1183   \if      m\pxrr@mode m%
1184   \else\if M\pxrr@mode m%
1185   \else\if j\pxrr@mode j%
1186   \else\if J\pxrr@mode j%
1187   \else\if g\pxrr@mode g%
1188   \else ?%
1189   \fi\fi\fi\fi\fi
1190 }

```

`\pxrr@check@option` `\pxrr@parse@option` の結果であるオプション設定値の整合性を検査し、必要に応じて、致命的エラーを出したり、警告を出して適切な値に変更したりする。

```

1191 \def\pxrr@check@option{%
    前と後の両方で突出が禁止された場合は致命的エラーとする。
1192   \ifpxrr@bprotr\else
1193   \ifpxrr@aprotr\else
1194     \pxrr@fatal@bad@no@protr
1195   \fi
1196 \fi

```

ゴースト処理有効で進入有りの場合は致命的エラーとする。

```

1197   \pxrr@oktrue
1198   \ifx\pxrr@bintr@\@empty\else
1199     \pxrr@okfalse
1200   \fi
1201   \ifx\pxrr@aintr@\@empty\else
1202     \pxrr@okfalse
1203   \fi
1204   \ifpxrr@ghost\else

```

```

1205     \pxrr@oktrue
1206     \fi
1207     \ifpxrr@ok\else
1208     \pxrr@fatal@bad@intr
1209     \fi

```

欧文ルビではモノルビ (m)・熟語ルビ (j) は指定不可なので、グループルビに変更する。この時に明示指定である場合は警告を出す。

```

1210     \if g\pxrr@mode\else
1211     \ifpxrr@abody
1212     \let\pxrr@mode=g\relax
1213     \ifpxrr@mode@given
1214     \pxrr@warn@must@group
1215     \fi
1216     \fi
1217     \fi

```

両側ルビでは熟語ルビ (j) は指定不可なので、グループルビに変更する。この時に明示指定である場合は警告を出す。

```

1218     \if \pxrr@mode@grand j%
1219     \ifnum\pxrr@side=\tw@
1220     \let\pxrr@mode=g\relax
1221     \ifpxrr@mode@given
1222     \pxrr@warn@bad@jukugo
1223     \fi
1224     \fi
1225     \fi

```

肩付き指定 (h) に関する検査。

```

1226     \ifnum\pxrr@athead>\z@

```

横組みでは不可なので中付きに変更する。

```

1227     \pxrr@if@in@tate@{ }{%else
1228     \chardef\pxrr@athead\z@
1229     }%

```

グループルビでは不可なので中付きに変更する。

```

1230     \if g\pxrr@mode
1231     \chardef\pxrr@athead\z@
1232     \fi

```

以上の 2 つの場合について、明示指定であれば警告を出す。

```

1233     \ifnum\pxrr@athead=\z@
1234     \ifpxrr@athead@given
1235     \pxrr@warn@bad@athead
1236     \fi
1237     \fi
1238     \fi

```

親文字列均等割り抑止 (E) の再設定 (エラー・警告なし)。

欧文ルビの場合は、均等割りを常に無効にする。

```

1239 \ifpxrr@abody
1240   \chardef\pxrr@evensp\z@
1241   \fi

```

グループルビ以外では、均等割りを有効にする。(この場合、親文字列は一文字毎に分解されるので、意味はもたない。均等割り抑止の方が特殊な処理なので、通常の処理に合わせる。)

```

1242 \if g\pxrr@mode\else
1243   \chardef\pxrr@evensp\@ne
1244   \fi

```

圏点ルビ同時付加の場合の調整。

```

1245   \ifpxrr@combo
1246     \pxrr@ck@check@option
1247   \fi
1248 }

```

## 4.10 フォントサイズ

`\pxrr@ruby@fsize` ルビ文字の公称サイズ。寸法値マクロ。ルビ命令呼出時に `\f@size` (親文字の公称サイズ) の `\pxrr@size@ratio` 倍に設定される。

```
1249 \let\pxrr@ruby@fsize\pxrr@zeropt
```

`\pxrr@body@zw` それぞれ、親文字とルビ文字の全角幅 (実際の `1zw` の寸法)。寸法値マクロ。pTeX では和文と欧文のバランスを整えるために和文を縮小することが多く、その場合「全角幅」は「公称サイズ」より小さくなる。なお、このパッケージでは漢字の幅が `1zw` であることを想定する。これらもルビ命令呼出時に正しい値に設定される。

```

1250 \let\pxrr@body@zw\pxrr@zeropt
1251 \let\pxrr@ruby@zw\pxrr@zeropt

```

`\pxrr@ruby@raise` ルビ文字に対する垂直方向の移動量。

```
1252 \let\pxrr@ruby@raise\pxrr@zeropt
```

`\pxrr@ruby@lower` ルビ文字に対する垂直方向の移動量 (下側ルビ)。

```
1253 \let\pxrr@ruby@lower\pxrr@zeropt
```

`\pxrr@htratio` 現在の組方向により、`\pxrr@yhtratio` と `\pxrr@thtratio` のいずれか一方に設定される。

```
1254 \def\pxrr@htratio{0}
```

`\pxrr@iiskip` 和文間空白および和欧文間空白の量。

```

\pxrr@iaiskip 1255 \let\pxrr@iiskip\pxrr@zeropt
1256 \let\pxrr@iaiskip\pxrr@zeropt

```

`\pxrr@assign@fsize` 上記の変数 (マクロ) を設定する。

```

1257 \def\pxrr@assign@fsize{%
1258   \@tempdima=\f@size\p@
1259   \@tempdima\pxrr@c@size@ratio\@tempdima
1260   \edef\pxrr@ruby@fsize{\the\@tempdima}%

```

```

1261 \pxrr@get@zwidth\pxrr@body@zw
1262 \begingroup
1263   \pxrr@use@ruby@font
1264   \pxrr@get@zwidth\pxrr@ruby@zw
1265   \global\let\pxrr@gtempa\pxrr@ruby@zw
1266 \endgroup
1267 \let\pxrr@ruby@zw\pxrr@gtempa
1268 \pxrr@get@iiskip\pxrr@iiskip
1269 \pxrr@get@iaiskip\pxrr@iaiskip

  \pxrr@htratio の値を設定する。
1270 \pxrr@if@in@tate{%
1271   \let\pxrr@htratio\pxrr@thtratio
1272 }{%
1273   \let\pxrr@htratio\pxrr@yhtratio
1274 }%

  \pxrr@ruby@raise の値を計算する。
1275 \@tempdima\pxrr@body@zw\relax
1276 \@tempdima\pxrr@htratio\@tempdima
1277 \@tempdimb\pxrr@ruby@zw\relax
1278 \advance\@tempdimb-\pxrr@htratio\@tempdimb
1279 \advance\@tempdima\@tempdimb
1280 \@tempdimb\pxrr@body@zw\relax
1281 \advance\@tempdima\pxrr@c@inter@gap\@tempdimb
1282 \edef\pxrr@ruby@raise{\the\@tempdima}%

  \pxrr@ruby@lower の値を計算する。
1283 \@tempdima\pxrr@body@zw\relax
1284 \advance\@tempdima-\pxrr@htratio\@tempdima
1285 \@tempdimb\pxrr@ruby@zw\relax
1286 \@tempdimb\pxrr@htratio\@tempdimb
1287 \advance\@tempdima\@tempdimb
1288 \@tempdimb\pxrr@body@zw\relax
1289 \advance\@tempdima\pxrr@c@inter@gap\@tempdimb
1290 \edef\pxrr@ruby@lower{\the\@tempdima}%

  圏点ルビ同時付加の設定。
1291 \ifpxrr@combo
1292   \pxrr@ck@assign@fsize
1293   \fi
1294 }

```

`\pxrr@use@ruby@font` ルビ用のフォントに切り替える。

```

1295 \def\pxrr@use@ruby@font{%
1296   \pxrr@without@macro@trace{%
1297     \let\rubyfontsize\pxrr@ruby@fsize
1298     \fontsize{\pxrr@ruby@fsize}{\z@}\selectfont
1299     \pxrr@c@ruby@font
1300   }%
1301 }

```

## 4.11 ルビ用均等割り

`\pxrr@locate@inner` ルビ配置パターン（行頭／行中／行末）を表す定数。

```
\pxrr@locate@head 1302 \chardef\pxrr@locate@inner=1
\pxrr@locate@end 1303 \chardef\pxrr@locate@head=0
1304 \chardef\pxrr@locate@end=2
```

`\pxrr@evenspace` `\pxrr@evenspace{<パターン>}\CS{<フォント>}{<幅>}{<テキスト>}`：<テキスト>を指定の<幅>に対する<パターン>（行頭／行中／行末）の「行中ルビ用均等割り」で配置し、結果をボックスレジスタ `\CS` に代入する。均等割りの要素分割は `\pxrr@decompose` を用いて行われるので、要素数が `\pxrr@cntr` に返る。また、先頭と末尾の空きの量をそれぞれ `\pxrr@bspace` と `\pxrr@aspace` に代入する。

`\pxrr@evenspace@int{<パターン>}\CS{<フォント>}{<幅>}`： `\pxrr@evenspace` の実行を、

`\pxrr@res` と `\pxrr@cntr` にテキストの `\pxrr@decompose` の結果が入っていて、  
テキストの自然長がマクロ `\pxrr@natwd` に入っている

という状態で、途中から開始する。

```
1305 \def\pxrr@evenspace#1#2#3#4#5{%
```

<テキスト>の自然長を計測し、`\pxrr@natwd` に格納する。

```
1306 \setbox#2\pxrr@hbox{#5}\@tempdima\wd#2%
1307 \edef\pxrr@natwd{\the\@tempdima}%
```

<テキスト>をリスト解析する（`\pxrr@cntr` に要素数が入る）。`\pxrr@evenspace@int` に引き継ぐ。

```
1308 \pxrr@decompose{#5}%
1309 \pxrr@evenspace@int{#1}{#2}{#3}{#4}%
1310 }
```

ここから実行を開始することもある。

```
1311 \def\pxrr@evenspace@int#1#2#3#4{%
```

比率パラメタの設定。

```
1312 \pxrr@save@listproc
1313 \ifcase#1%
1314 \pxrr@evenspace@param\pxrr@zero\pxrr@sprop@hy\pxrr@sprop@hz
1315 \or
1316 \pxrr@evenspace@param\pxrr@sprop@x\pxrr@sprop@y\pxrr@sprop@z
1317 \or
1318 \pxrr@evenspace@param\pxrr@sprop@ex\pxrr@sprop@ey\pxrr@zero
1319 \fi
```

挿入される `fil` の係数を求め、これがゼロの場合（この時  $X = Z = 0$  である）は、アンダーフル防止のため、 $X = Z = 1$  に変更する。

```
1320 \pxrr@dima=\pxrr@cntr\p@
```

```

1321 \advance\pxrr@dima-\p@
1322 \pxrr@dima=\pxrr@sprop@y@\pxrr@dima
1323 \advance\pxrr@dima\pxrr@sprop@x@\p@
1324 \advance\pxrr@dima\pxrr@sprop@z@\p@
1325 \ifdim\pxrr@dima>\z@\else
1326   \ifnum#1>\z@
1327     \let\pxrr@sprop@x@\@ne
1328     \advance\pxrr@dima\p@
1329   \fi
1330   \ifnum#1<\tw@
1331     \let\pxrr@sprop@z@\@ne
1332     \advance\pxrr@dima\p@
1333   \fi
1334 \fi
1335 \edef\pxrr@tempa{\strip@pt\pxrr@dima}%
1336 \ifpxrrDebug
1337 \typeout{\number\pxrr@sprop@x@:\number\pxrr@sprop@z@:\pxrr@tempa}%
1338 \fi

```

\pxrr@pre/inter/post にグルーを設定して、\pxrr@res を組版する。なお、\setbox... を一旦マクロ \pxrr@makebox@res に定義しているのは、後で \pxrr@adjust@margin で再度呼び出せるようにするため。

```

1339 \def\pxrr@pre##1{\pxrr@hfilx\pxrr@sprop@x@ ##1}%
1340 \def\pxrr@inter##1{\pxrr@hfilx\pxrr@sprop@y@ ##1}%
1341 \def\pxrr@post{\pxrr@hfilx\pxrr@sprop@z@}%
1342 \def\pxrr@makebox@res{%
1343   \setbox#2=\pxrr@hbox@to#4{#3\pxrr@res}%
1344 }%
1345 \pxrr@makebox@res

```

前後の空白の量を求める。

```

1346 \pxrr@dima\wd#2%
1347 \advance\pxrr@dima-\pxrr@natwd\relax
1348 \pxrr@invscale\pxrr@dima\pxrr@tempa
1349 \@tempdima\pxrr@sprop@x@\pxrr@dima
1350 \edef\pxrr@bspace{\the\@tempdima}%
1351 \@tempdima\pxrr@sprop@z@\pxrr@dima
1352 \edef\pxrr@aspace{\the\@tempdima}%
1353 \pxrr@restore@listproc
1354 \ifpxrrDebug
1355 \typeout{\pxrr@bspace:\pxrr@aspace}%
1356 \fi
1357 }
1358 \def\pxrr@evenspace@param#1#2#3{%
1359   \let\pxrr@sprop@x@#1%
1360   \let\pxrr@sprop@y@#2%
1361   \let\pxrr@sprop@z@#3%
1362 }
1363 \let\pxrr@makebox@res\undefined

```

`\pxrr@adjust@margin` `\pxrr@adjust@margin`: `\pxrr@evenspace(@int)` を呼び出した直後に呼ぶ必要がある。  
先頭と末尾の各々について、空きの量が `\pxrr@maxmargin` により決まる上限値を超える場合に、空きを上限値に抑えるように再調整する。

```

1364 \def\pxrr@adjust@margin{%
1365   \pxrr@save@listproc
1366   \@tempdima\pxrr@body@zw\relax
1367   \@tempdima\pxrr@maxmargin\@tempdima

  再調整が必要かを \if@tempswa に記録する。1文字しかない場合は調整不能だから検査を飛ばす。

1368   \@tempswafalse
1369   \def\pxrr@pre##1{\pxrr@hfilx\pxrr@sprop@x@ ##1}%
1370   \def\pxrr@inter##1{\pxrr@hfilx\pxrr@sprop@y@ ##1}%
1371   \def\pxrr@post{\pxrr@hfilx\pxrr@sprop@z@}%
1372   \ifnum\pxrr@cntr>\@ne
1373     \ifdim\pxrr@bspace>\@tempdima
1374       \edef\pxrr@bspace{\the\@tempdima}%
1375       \def\pxrr@pre##1{\hskip\pxrr@bspace\relax ##1}%
1376       \@tempswatrue
1377     \fi
1378     \ifdim\pxrr@aspace>\@tempdima
1379       \edef\pxrr@aspace{\the\@tempdima}%
1380       \def\pxrr@post{\hskip\pxrr@aspace\relax}%
1381       \@tempswatrue
1382     \fi
1383   \fi

```

必要に応じて再調整を行う。

```

1384   \if@tempswa
1385     \pxrr@makebox@res
1386   \fi
1387   \pxrr@restore@listproc
1388 \ifpxrrDebug
1389 \typeout{\pxrr@bspace:\pxrr@aspace}%
1390 \fi
1391 }

```

`\pxrr@save@listproc` `\pxrr@pre/inter/post` の定義を退避する。

※ 退避のネストはできない。

```

1392 \def\pxrr@save@listproc{%
1393   \let\pxrr@pre@save\pxrr@pre
1394   \let\pxrr@inter@save\pxrr@inter
1395   \let\pxrr@post@save\pxrr@post
1396 }
1397 \let\pxrr@pre@save\@undefined
1398 \let\pxrr@inter@save\@undefined
1399 \let\pxrr@post@save\@undefined

```

`\pxrr@restore@listproc` `\pxrr@pre/inter/post` の定義を復帰する。

```
1400 \def\pxrr@restore@listproc{%
1401   \let\pxrr@pre\pxrr@pre@save
1402   \let\pxrr@inter\pxrr@inter@save
1403   \let\pxrr@post\pxrr@post@save
1404 }
```

## 4.12 小書き仮名の変換

`\pxrr@trans@res` `\pxrr@transform@kana` 内で変換結果を保持するマクロ。

```
1405 \let\pxrr@trans@res\@empty
```

`\pxrr@transform@kana` `\pxrr@transform@kana\CS` : マクロ `\CS` の展開テキストの中でグループに含まれない小書き仮名を対応する非小書き仮名に変換し、`\CS` を上書きする。

```
1406 \def\pxrr@transform@kana#1{%
1407   \let\pxrr@trans@res\@empty
1408   \def\pxrr@transform@kana@end\pxrr@end{%
1409     \let#1\pxrr@trans@res
1410   }%
1411   \expandafter\pxrr@transform@kana@loop@a#1\pxrr@end
1412 }
1413 \def\pxrr@transform@kana@loop@a{%
1414   \futurelet\pxrr@token\pxrr@transform@kana@loop@b
1415 }
1416 \def\pxrr@transform@kana@loop@b{%
1417   \ifx\pxrr@token\pxrr@end
1418     \let\pxrr@tempb\pxrr@transform@kana@end
1419   \else\ifx\pxrr@token\bgroup
1420     \let\pxrr@tempb\pxrr@transform@kana@loop@c
1421   \else\ifx\pxrr@token\@sptoken
1422     \let\pxrr@tempb\pxrr@transform@kana@loop@d
1423   \else
1424     \let\pxrr@tempb\pxrr@transform@kana@loop@e
1425   \fi\fi\fi
1426   \pxrr@tempb
1427 }
1428 \def\pxrr@transform@kana@loop@c#1{%
1429   \pxrr@appto\pxrr@trans@res{{#1}}%
1430   \pxrr@transform@kana@loop@a
1431 }
1432 \expandafter\def\expandafter\pxrr@transform@kana@loop@d\space{%
1433   \pxrr@appto\pxrr@trans@res{ }%
1434   \pxrr@transform@kana@loop@a
1435 }
1436 \def\pxrr@transform@kana@loop@e#1{%
1437   \expandafter\pxrr@transform@kana@loop@f\string#1\pxrr@nil#1%
1438 }
```

```

1439 \def\pxrr@transform@kana@loop@f#1#2\pxrr@nil#3{%
1440   \@tempswafalse
1441   \ifnum'#1>\@cclv
1442     \begingroup\expandafter\expandafter\expandafter\endgroup
1443     \expandafter\ifx\csname pxrr@nonsmall/#3\endcsname\relax\else
1444       \@tempswatruue
1445       \fi
1446       \fi
1447       \if@tempswa
1448         \edef\pxrr@tempa{%
1449           \noexpand\pxrr@appto\noexpand\pxrr@trans@res
1450           {\csname pxrr@nonsmall/#3\endcsname}%
1451         }%
1452         \pxrr@tempa
1453       \else
1454         \pxrr@appto\pxrr@trans@res{#3}%
1455       \fi
1456       \pxrr@transform@kana@loop@a
1457 }
1458 \def\pxrr@assign@nonsmall#1/#2\pxrr@nil{%
1459   \pxrr@get@jchar@token\pxrr@tempa{\pxrr@jc{#1}}%
1460   \pxrr@get@jchar@token\pxrr@tempb{\pxrr@jc{#2}}%
1461   \expandafter\edef\csname pxrr@nonsmall/\pxrr@tempa\endcsname
1462   {\pxrr@tempb}%
1463 }
1464 \@tfor\pxrr@tempc:=%
1465   {2421:3041/2422:3042}{2423:3043/2424:3044}%
1466   {2425:3045/2426:3046}{2427:3047/2428:3048}%
1467   {2429:3049/242A:304A}{2443:3063/2444:3064}%
1468   {2463:3083/2464:3084}{2465:3085/2466:3086}%
1469   {2467:3087/2468:3088}{246E:308E/246F:308F}%
1470   {2521:30A1/2522:30A2}{2523:30A3/2524:30A4}%
1471   {2525:30A5/2526:30A6}{2527:30A7/2528:30A8}%
1472   {2529:30A9/252A:30AA}{2543:30C3/2544:30C4}%
1473   {2563:30E3/2564:30E4}{2565:30E5/2566:30E6}%
1474   {2567:30E7/2568:30E8}{256E:30EE/256F:30EF}%
1475   \do{%
1476     \expandafter\pxrr@assign@nonsmall\pxrr@tempc\pxrr@nil
1477 }

```

#### 4.13 ブロック毎の組版

\ifpxrr@protr ルビ文字列の突出があるか。スイッチ。

```
1478 \newif\ifpxrr@protr
```

\ifpxrr@any@protr 複数ブロックの処理で、いずれかのブロックにルビ文字列の突出があるか。スイッチ。

```
1479 \newif\ifpxrr@any@protr
```

`\pxrr@locate@temp` `\pxrr@compose*side@block@do` で使われる一時変数。整数定数。

```
1480 \let\pxrr@locate@temp\relax
```

`\pxrr@epsilon` ルビ文字列と親文字列の自然長の差がこの値以下の場合、差はないものとみなす（演算誤差対策）。

```
1481 \def\pxrr@epsilon{0.01pt}
```

`\pxrr@compose@block` `\pxrr@compose@block{<パターン>}{<親文字ブロック>}{<ルビ文字ブロック>}`：1つのブロックの組版処理。`<パターン>`は`\pxrr@evenspace`と同じ意味。突出があるかを`\ifpxrr@protr`に返し、前と後の突出の量をそれぞれ`\pxrr@bspace`と`\pxrr@aspace`に返す。

```
1482 \def\pxrr@compose@block#1#2#3{%
```

本体の前に加工処理を介入させる。

※`\pxrr@compose@block@pre`は2つのルビ引数を取る。`\pxrr@compose@block@do`に本体マクロを`\let`する。

```
1483 \let\pxrr@compose@block@do\pxrr@compose@oneside@block@do
```

```
1484 \pxrr@compose@block@pre{#1}{#2}{#3}{}%
```

```
1485 }
```

こちらが本体。

```
1486 % #4 は空
```

```
1487 \def\pxrr@compose@oneside@block@do#1#2#3#4{%
```

```
1488 \setbox\pxrr@boxa\pxrr@hbox{#2}%
```

```
1489 \edef\pxrr@ck@body@natwd{\the\wd\pxrr@boxa}%
```

```
1490 \let\pxrr@ck@locate\pxrr@locate@inner
```

```
1491 \setbox\pxrr@boxr\pxrr@hbox{%
```

```
1492 \pxrr@use@ruby@font
```

```
1493 #3%
```

```
1494 }%
```

```
1495 \@tempdima\wd\pxrr@boxr
```

```
1496 \advance\@tempdima-\wd\pxrr@boxa
```

```
1497 \ifdim\pxrr@epsilon<\@tempdima
```

ルビ文字列の方が長い場合。親文字列をルビ文字列の長さに合わせて均等割りて組み直す。

`\pxrr@?space`は`\pxrr@evenspace@int`が返す値のままよい。「拡張肩付き」指定の場合、前側の突出を抑制する。

```
1498 \pxrr@protrtrue
```

```
1499 \let\pxrr@locate@temp#1%
```

```
1500 \ifnum\pxrr@athead>\@ne
```

```
1501 \ifnum\pxrr@locate@temp=\pxrr@locate@inner
```

```
1502 \let\pxrr@locate@temp\pxrr@locate@head
```

```
1503 \fi
```

```
1504 \fi
```

```
1505 \let\pxrr@ck@locate\pxrr@locate@temp
```

```
1506 \pxrr@decompose{#2}%
```

```
1507 \edef\pxrr@natwd{\the\wd\pxrr@boxa}%
```

```
1508 \pxrr@evenspace@int\pxrr@locate@temp\pxrr@boxa\relax
```

```

1509     {\wd\pxrr@boxr}%
1510 \else\ifdim-\pxrr@epsilon>\@tempdima

```

ルビ文字列の方が短い場合。ルビ文字列を親文字列の長さに合わせて均等割りで組み直す。  
 この場合、`\pxrr@maxmargin` を考慮する必要がある。ただし肩付きルビの場合は組み直し  
 を行わない。`\pxrr@?space` はゼロに設定する。

```

1511 \pxrr@protrfalse
1512 \ifnum\pxrr@athead=\z@
1513 \pxrr@decompose{#3}%
1514 \edef\pxrr@natwd{\the\wd\pxrr@boxr}%
1515 \pxrr@evenspace@int{#1}\pxrr@boxr
1516 \pxrr@use@ruby@font{\wd\pxrr@boxa}%
1517 \pxrr@adjust@margin
1518 \fi
1519 \let\pxrr@bspace\pxrr@zeropt
1520 \let\pxrr@aspace\pxrr@zeropt
1521 \else

```

両者の長さが等しい（とみなす）場合。突出フラグは常に偽にする（実際にはルビの方が僅  
 かだけ長いかも知れないが）。

```

1522 \pxrr@protrfalse
1523 \let\pxrr@bspace\pxrr@zeropt
1524 \let\pxrr@aspace\pxrr@zeropt
1525 \fi\fi

```

実際に組版を行う。

```

1526 \setbox\z@\hbox{%
1527 \ifnum\pxrr@side=\z@
1528 \raise\pxrr@ruby@raise\box\pxrr@boxr
1529 \else
1530 \lower\pxrr@ruby@lower\box\pxrr@boxr
1531 \fi
1532 }%
1533 \ifnum \ifpxrr@combo\pxrr@ck@ruby@combo\else\z@\fi >\z@
1534 \pxrr@ck@compose{#2}%
1535 \fi
1536 \ht\z@\z@ \dp\z@\z@
1537 \@tempdima\wd\z@
1538 \setbox\pxrr@boxr\hbox{%
1539 \box\z@
1540 \kern-\@tempdima
1541 \box\pxrr@boxa
1542 }%

```

`\ifpxrr@any@protr` を設定する。

```

1543 \ifpxrr@protr
1544 \pxrr@any@protrtrue
1545 \fi
1546 }

```

`\pxrr@compose@twoside@block` 両側ルビ用のブロック構成。

```
1547 \def\pxrr@compose@twoside@block{%
1548   \let\pxrr@compose@block@do\pxrr@compose@twoside@block@do
1549   \pxrr@compose@block@pre
1550 }
1551 \def\pxrr@compose@twoside@block@do#1#2#3#4{%
```

`\pxrr@boxa` に親文字、`\pxrr@boxr` に上側ルビ、`\pxrr@boxb` に下側ルビの出力を保持する。

```
1552   \setbox\pxrr@boxa\pxrr@hbox{#2}%
1553   \edef\pxrr@ck@body@natwd{\the\wd\pxrr@boxa}%
1554   \let\pxrr@ck@locate\pxrr@locate@inner
1555   \setbox\pxrr@boxr\pxrr@hbox{%
1556     \pxrr@use@ruby@font
1557     #3%
1558   }%
1559   \setbox\pxrr@boxb\pxrr@hbox{%
1560     \pxrr@use@ruby@font
1561     #4%
1562   }%
```

「何れかのルビが親文字列より長いか」を検査する。

```
1563   \@tempwafalse
1564   \@tempdima\wd\pxrr@boxr
1565   \advance\@tempdima-\wd\pxrr@boxa
1566   \ifdim\pxrr@epsilon<\@tempdima \@tempwatrue \fi
1567   \@tempdima\wd\pxrr@boxb
1568   \advance\@tempdima-\wd\pxrr@boxa
1569   \ifdim\pxrr@epsilon<\@tempdima \@tempwatrue \fi
```

親文字より長いルビが存在する場合。長い方のルビ文字列の長さに合わせて、親文字列と他方のルビ文字列を組み直す。(実際の処理は `\pxrr@compose@twoside@block@sub` で行う。)

```
1570   \if@tempwa
1571     \pxrr@protrtrue
```

「拡張肩付き」指定の場合、前側の突出を抑止する。

```
1572     \let\pxrr@locate@temp#1%
1573     \ifnum\pxrr@athead>\@ne
1574       \ifnum\pxrr@locate@temp=\pxrr@locate@inner
1575         \let\pxrr@locate@temp\pxrr@locate@head
1576         \fi
1577       \fi
1578     \let\pxrr@ck@locate\pxrr@locate@temp
```

上側と下側のどちらのルビが長いかに応じて引数を変えて、`\pxrr@compose@twoside@block@sub` を呼び出す。

```
1579     \ifdim\wd\pxrr@boxr<\wd\pxrr@boxb
1580       \pxrr@compose@twoside@block@sub{#2}{#3}%
```

```

1581     \pxrr@boxr\pxrr@boxb
1582   \else
1583     \pxrr@compose@twoside@block@sub{#2}{#4}%
1584     \pxrr@boxb\pxrr@boxr
1585   \fi

```

親文字の方が長い場合。親文字列の長さに合わせて、両方のルビを（片側の場合と同様の）均等割りで組み直す。

```

1586   \else
1587     \pxrr@protrfalse

```

肩付きルビの場合は組み直しを行わない。

```

1588   \ifnum\pxrr@athead=\z@
1589     \@tempdima\wd\pxrr@boxa
1590     \advance\@tempdima-\wd\pxrr@boxr
1591     \ifdim\pxrr@epsilon<\@tempdima
1592       \pxrr@decompose{#3}%
1593       \edef\pxrr@natwd{\the\wd\pxrr@boxr}%
1594       \pxrr@evenspace@int{#1}\pxrr@boxr
1595       \pxrr@use@ruby@font{\wd\pxrr@boxa}%
1596       \pxrr@adjust@margin
1597     \fi
1598     \@tempdima\wd\pxrr@boxa
1599     \advance\@tempdima-\wd\pxrr@boxb
1600     \ifdim\pxrr@epsilon<\@tempdima
1601       \pxrr@decompose{#4}%
1602       \edef\pxrr@natwd{\the\wd\pxrr@boxb}%
1603       \pxrr@evenspace@int{#1}\pxrr@boxb
1604       \pxrr@use@ruby@font{\wd\pxrr@boxa}%
1605       \pxrr@adjust@margin
1606     \fi
1607   \fi

```

\pxrr@?space はゼロに設定する。

```

1608     \let\pxrr@bspace\pxrr@zeropt
1609     \let\pxrr@aspace\pxrr@zeropt
1610   \fi

```

実際に組版を行う。

```

1611   \setbox\z@\hbox{%
1612     \@tempdima\wd\pxrr@boxr
1613     \raise\pxrr@ruby@raise\box\pxrr@boxr
1614     \kern-\@tempdima
1615     \lower\pxrr@ruby@lower\box\pxrr@boxb
1616   }%
1617   \ifnum \ifpxrr@combo\pxrr@ck@ruby@combo\else\z@\fi >\z@
1618     \pxrr@ck@compose{#2}%
1619   \fi
1620   \ht\z@\z@ \dp\z@\z@
1621   \@tempdima\wd\z@

```

```

1622 \setbox\pxrr@boxr\hbox{%
1623   \box\z@
1624   \kern-\@tempdima
1625   \box\pxrr@boxa
1626 }%
1627 }

```

\pxrr@body@wd \pxrr@compose@twoside@block@sub の内部で用いられる変数で、“親文字列の実際の長さ”（均等割りで入った中間の空きを入れるが両端の空きを入れない）を表す。寸法値マクロ。

```
1628 \let\pxrr@body@wd\relax
```

\pxrr@compose@twoside@block@sub \pxrr@compose@twoside@block@sub の内部で用いられるマクロ。

```
1629 \let\pxrr@restore@margin@values\relax
```

\pxrr@compose@twoside@block@sub \pxrr@compose@twoside@block@sub{親文字}{短い方のルビ文字}\CSa\CSb：両側ルビで親文字列より長いルビ文字列が存在する場合の組み直しの処理を行う。このマクロの呼出時、上側ルビの出力結果が \pxrr@boxr、下側ルビの出力結果が \pxrr@boxb に入っているが、この2つのボックスのうち、短いルビの方が \CSa、長いルビの方が \CSb として渡されている。

```

1630 \def\pxrr@compose@twoside@block@sub#1#2#3#4{%
1631   \pxrr@decompose{#1}%
1632   \edef\pxrr@natwd{\the\wd\pxrr@boxa}%
1633   \pxrr@evenspace@int\pxrr@locate@temp\pxrr@boxa\relax{\wd#4}%
1634   \@tempdima\wd#4%
1635   \advance\@tempdima-\pxrr@bspace\relax
1636   \advance\@tempdima-\pxrr@aspace\relax
1637   \edef\pxrr@body@wd{\the\@tempdima}%
1638   \advance\@tempdima-\wd#3%
1639   \ifdim\pxrr@epsilon<\@tempdima
1640     \edef\pxrr@restore@margin@values{%
1641       \edef\noexpand\pxrr@bspace{\pxrr@bspace}%
1642       \edef\noexpand\pxrr@aspace{\pxrr@aspace}%
1643     }%
1644     \pxrr@decompose{#2}%
1645     \edef\pxrr@natwd{\the\wd#3}%
1646     \pxrr@evenspace@int\pxrr@locate@temp#3%
1647     \pxrr@use@ruby@font{\pxrr@body@wd}%
1648     \pxrr@adjust@margin
1649     \pxrr@restore@margin@values
1650     \setbox#3\hbox{%
1651       \kern\pxrr@bspace\relax
1652       \box#3%
1653     }%
1654   \else
1655     \ifnum\pxrr@locate@temp=\pxrr@locate@head
1656       \@tempdima\z@
1657     \else\ifnum\pxrr@locate@temp=\pxrr@locate@inner

```

```

1658     \@tempdima.5\@tempdima
1659     \fi\fi
1660     \advance\@tempdima\pxrr@bspace\relax
1661     \setbox#3\hbox{%
1662         \kern\@tempdima
1663         \box#3%
1664     }%
1665     \fi
1666 }
1667     \end{macrocode}
1668 % \end{macro}
1669 %
1670 % \begin{macro}{\pxrr@compose@block@pre}
1671 % |\pxrr@compose@block@pre{|\jmeta{パターン}}|}{|^A
1672 %r \jmeta{親文字}}|}{|\jmeta{ルビ 1}}|}{|\jmeta{ルビ 2}}|}{\Means
1673 % 親文字列・ルビ文字列の加工を行う。
1674 % \Note 両側ルビ対応のため、ルビ用引数が2つある。
1675 %     \begin{macrocode}
1676 \def\pxrr@compose@block@pre{%

    f 指定時は小書き仮名の変換を施す。

1677     \pxrr@cond\ifnum\pxrr@fullsize>\z@\fi{%
1678         \pxrr@compose@block@pre@a
1679     }{%
1680         \pxrr@compose@block@pre@d
1681     }%
1682 }
1683 % {パターン}{親文字}{ルビ 1}{ルビ 2}
1684 \def\pxrr@compose@block@pre@a#1#2#3#4{%
1685     \def\pxrr@compose@block@tempa{#4}%
1686     \pxrr@transform@kana\pxrr@compose@block@tempa
1687     \expandafter\pxrr@compose@block@pre@b
1688     \expandafter{\pxrr@compose@block@tempa}{#1}{#2}{#3}%
1689 }
1690 % {ルビ 2}{パターン}{親文字}{ルビ 1}
1691 \def\pxrr@compose@block@pre@b#1#2#3#4{%
1692     \def\pxrr@compose@block@tempa{#4}%
1693     \pxrr@transform@kana\pxrr@compose@block@tempa
1694     \expandafter\pxrr@compose@block@pre@c
1695     \expandafter{\pxrr@compose@block@tempa}{#1}{#2}{#3}%
1696 }
1697 % {ルビ 1}{ルビ 2}{パターン}{親文字}
1698 \def\pxrr@compose@block@pre@c#1#2#3#4{%
1699     \pxrr@compose@block@pre@d{#3}{#4}{#1}{#2}%
1700 }
1701 \def\pxrr@compose@block@pre@d{%
1702     \pxrr@cond\ifnum\pxrr@evensp=\z@\fi{%
1703         \pxrr@compose@block@pre@e
1704     }{%

```

```

1705 \pxrr@compose@block@pre@f
1706 }%
1707 }
1708 % {パターン}{親文字}
1709 \def\pxrr@compose@block@pre@e#1#2{%
1710 \pxrr@compose@block@pre@f{#1}{#2}}%
1711 }
1712 \def\pxrr@compose@block@pre@f{%
1713 \pxrr@cond\ifnum\pxrr@reversp=\z@\fi{%
1714 \pxrr@compose@block@pre@g
1715 }{%
1716 \pxrr@compose@block@do
1717 }%
1718 }
1719 % {パターン}{親文字}{ルビ 1}{ルビ 2}
1720 \def\pxrr@compose@block@pre@g#1#2#3#4{%
1721 \pxrr@compose@block@do{#1}{#2}{#3}{#4}}%
1722 }
1723 \let\pxrr@compose@block@tempa\@undefined

```

#### 4.14 命令の頑強化

`\pxrr@add@protect` `\pxrr@add@protect\CS`: 命令 `\CS` に `\protect` を施して頑強なものに変える。`\CS` は最初から `\DeclareRobustCommand` で定義された頑強な命令とほぼ同じように振舞う——例えば、`\CS` の定義の本体は `\CS□` という制御綴に移される。唯一の相違点は、「組版中」(すなわち `\protect = \@typeset@protect`) の場合は、`\CS` は `\protect\CS□` ではなく、単なる `\CS□` に展開されることである。組版中は `\protect` は結局 `\relax` であるので、`\DeclareRobustCommand` 定義の命令の場合、`\relax` が「実行」されることになるが、`pTeX` ではこれがメトリックグループの挿入に干渉するので、このパッケージの目的に沿わないのである。

※ `\CS` は「制御語」(制御記号でなく)である必要がある。

```

1724 \def\pxrr@add@protect#1{%
1725 \expandafter\pxrr@add@protect@a
1726 \csname\expandafter@gobble\string#1\space\endcsname#1%
1727 }
1728 \def\pxrr@add@protect@a#1#2{%
1729 \let#1=#2%
1730 \def#2{\pxrr@check@protect\protect#1}}%
1731 }
1732 \def\pxrr@check@protect{%
1733 \ifx\protect\@typeset@protect
1734 \expandafter@gobble
1735 \fi
1736 }

```

## 4.15 致命的エラー対策

致命的エラーが起こった場合は、ルビ入力を放棄して単に親文字列を出力することにする。

`\pxrr@body@input` 入力された親文字列。

```
1737 \let\pxrr@body@input\@empty
```

`\pxrr@prepare@fallback` `\pxrr@prepare@fallback{<親文字列>}` :

```
1738 \def\pxrr@prepare@fallback#1{%
1739   \pxrr@fatal@errorfalse
1740   \def\pxrr@body@input{#1}%
1741 }
```

`\pxrr@fallback` 致命的エラー時に出力となるもの。単に親文字列を出力することにする。

```
1742 \def\pxrr@fallback{%
1743   \pxrr@body@input
1744 }
```

`\pxrr@if@alive` `\pxrr@if@alive{<コード>}` : 致命的エラーが未発生の場合に限り、<コード>に展開する。

```
1745 \def\pxrr@if@alive{%
1746   \ifpxrr@fatal@error \expandafter\@gobble
1747   \else \expandafter\@firstofone
1748   \fi
1749 }
```

## 4.16 先読み処理

ゴースト処理が無効の場合に後ろ側の禁則処理を行うため、ルビ命令の直後に続くトークンを取得して、その前禁則ペナルティ (`\prebreakpenalty`) の値を保存する。信頼性の低い方法なので、ゴースト処理が可能な場合はそちらを利用するべきである。

`\pxrr@end@kinsoku` ルビ命令直後の文字の前禁則ペナルティ値とみなす値。

```
1750 \def\pxrr@end@kinsoku{0}
```

`\pxrr@ruby@scan` 片側ルビ用の先読み処理。

```
1751 \def\pxrr@ruby@scan#1#2{%
```

`\pxrr@check@kinsoku` の続きの処理。`\pxrr@cntr` の値を `\pxrr@end@kinsoku` に保存して、ルビ処理本体を呼び出す。

```
1752   \def\pxrr@tempc{%
1753     \edef\pxrr@end@kinsoku{the\pxrr@cntr}%
1754     \pxrr@do@proc{#1}{#2}%
1755   }%
1756   \pxrr@check@kinsoku\pxrr@tempc
1757 }
```

`\pxrr@truby@scan` 両側ルビ用の先読み処理。

```
1758 \def\pxrr@truby@scan#1#2#3{%
1759   \def\pxrr@tempc{%
1760     \edef\pxrr@end@kinsoku{\the\pxrr@cntr}%
1761     \pxrr@do@proc{#1}{#2}{#3}%
1762   }%
1763   \pxrr@check@kinsoku\pxrr@tempc
1764 }
```

`\pxrr@check@kinsoku` `\pxrr@check@kinsoku\CS` : `\CS` の直後に続くトークンについて、それが「通常文字」（和文文字トークンまたはカテゴリコード 11、12 の欧文文字トークン）である場合にはその前禁則ペナルティ（`\prebreakpenalty`）の値を、そうでない場合はゼロを `\pxrr@cntr` に代入する。その後、`\CS` を実行（展開）する。

※ ただし、欧文ルビの場合、欧文文字の前禁則ペナルティは 20000 として扱う。

```
1765 \def\pxrr@check@kinsoku#1{%
1766   \let\pxrr@tempb#1%
1767   \futurelet\pxrr@token\pxrr@check@kinsoku@a
1768 }
1769 \def\pxrr@check@kinsoku@a{%
1770   \pxrr@check@char\pxrr@token
```

和文ルビの場合は、欧文通常文字も和文通常文字と同じ扱いにする。

```
1771   \ifpxrr@abody\else
1772     \ifnum\pxrr@cntr=\@ne
1773       \pxrr@cntr\tw@
1774     \fi
1775   \fi
1776   \ifcase\pxrr@cntr
1777     \pxrr@cntr\z@
1778     \expandafter\pxrr@tempb
1779   \or
1780     \pxrr@cntr\@MM
1781     \expandafter\pxrr@tempb
1782   \else
1783     \expandafter\pxrr@check@kinsoku@b
1784   \fi
1785 }
```

`\let` されたトークンのままでは符号位置を得ることができないため、改めてマクロの引数として受け取り、複製した上で片方を後の処理に使う。既に後続トークンは「通常文字」である（つまり空白や `{` ではない）ことが判明していることに注意。

```
1786 \def\pxrr@check@kinsoku@b#1{%
1787   \pxrr@check@kinsoku@c#1#1%
1788 }
1789 \def\pxrr@check@kinsoku@c#1{%
1790   \pxrr@get@prebreakpenalty\pxrr@cntr{‘#1}%
1791   \pxrr@tempb
1792 }
```

`\pxrr@check@char` `\pxrr@check@char\CS` : トークン `\CS` が「通常文字」であるかを調べ、以下の値を `\pxrr@cntr` に返す: 0 = 通常文字でない; 1 = 欧文通常文字; 2 = 和文通常文字。  
 定義本体の中でカテゴリコード 12 の `kanji` というトークン列が必要なので、少々特殊な処置をしている。まず `\pxrr@check@char` を定義するためのマクロを用意する。

```
1793 \def\pxrr@tempa#1#2\pxrr@nil{%
```

実際に呼び出される時には #2 はカテゴリコード 12 の `kanji` に置き換わる。(不要な `\` を #1 に受け取らせている。)

```
1794 \def\pxrr@check@char##1{%
```

まず制御綴とカテゴリコード 11、12、13 を手早く `\ifcat` で判定する。

```
1795 \ifcat\noexpand##1\relax
1796 \pxrr@cntr\z@
1797 \else\ifcat\noexpand##1\noexpand~%
1798 \pxrr@cntr\z@
1799 \else\ifcat\noexpand##1A%
1800 \pxrr@cntr@ne
1801 \else\ifcat\noexpand##10%
1802 \pxrr@cntr@ne
1803 \else
```

それ以外の場合、和文文字トークンであるかを `\meaning` テストで調べる。(和文文字の `\ifcat` 判定は色々面倒な点があるので避ける。)

```
1804 \pxrr@cntr\z@
1805 \expandafter\pxrr@check@char@a\meaning##1#2\pxrr@nil
1806 \fi\fi\fi\fi
1807 }%
1808 \def\pxrr@check@char@a##1#2##2\pxrr@nil{%
1809 \ifcat @##10%
1810 \pxrr@cntr\tw@
1811 \fi
1812 }%
1813 }
```

規定の引数を用意して「定義マクロ」を呼ぶ。

```
1814 \expandafter\pxrr@tempa\string\kanji\pxrr@nil
```

## 4.17 進入処理

`\pxrr@auto@penalty` 自動挿入されるペナルティ。(整数定数への `\let`。)

```
1815 \let\pxrr@auto@penalty\z@
```

`\pxrr@auto@icspace` 文字間の空き。寸法値マクロ。

```
1816 \let\pxrr@auto@icspace\pxrr@zeropt
```

`\pxrr@intr@amount` 進入の幅。寸法値マクロ。

```
1817 \let\pxrr@intr@amount\pxrr@zeropt
```

`\pxrr@intrude@setauto@j` 和文の場合の `\pxrr@auto@*` の設定。

```
1818 \def\pxrr@intrude@setauto@j{%
    行分割禁止 (*) の場合、ペナルティを 20000 とし、字間空きはゼロにする。
1819 \ifpxrr@bnobr
1820 \let\pxrr@auto@penalty\@MM
1821 \let\pxrr@auto@icspace\pxrr@zeropt
    それ以外の場合は、ペナルティはゼロで、\pxrr@bspace の設定を活かす。
1822 \else
1823 \let\pxrr@auto@penalty\z@
1824 \if:\pxrr@bscomp
1825 \let\pxrr@auto@icspace\pxrr@iaiskip
1826 \else\if.\pxrr@bscomp
1827 \let\pxrr@auto@icspace\pxrr@zeropt
1828 \else
1829 \let\pxrr@auto@icspace\pxrr@iiskip
1830 \fi\fi
1831 \fi
1832 }
```

`\pxrr@intrude@setauto@a` 欧文の場合の `\pxrr@auto@*` の設定。

```
1833 \def\pxrr@intrude@setauto@a{%
    欧文の場合、和欧文間空白挿入指定 (:) でない場合は、(欧文同士と見做して) 行分割禁止
    にする。
1834 \if:\pxrr@bscomp\else
1835 \pxrr@bnobrtrue
1836 \fi
1837 \ifpxrr@bnobr
1838 \let\pxrr@auto@penalty\@MM
1839 \let\pxrr@auto@icspace\pxrr@zeropt
1840 \else
    この分岐は和欧文間空白挿入指定 (:) に限る。
1841 \let\pxrr@auto@penalty\z@
1842 \let\pxrr@auto@icspace\pxrr@iaiskip
1843 \fi
1844 }
```

#### 4.17.1 前側進入処理

`\pxrr@intrude@head` 前側の進入処理。

```
1845 \def\pxrr@intrude@head{%
    ゴースト処理が有効な場合は進入処理を行わない。(だから進入が扱えない。)
1846 \ifpxrr@ghost\else
    実効の進入幅は \pxrr@bintr と \pxrr@bspace の小さい方。
1847 \let\pxrr@intr@amount\pxrr@bspace
```

```

1848 \ifdim\pxrr@bintr<\pxrr@intr@amount\relax
1849 \let\pxrr@intr@amount\pxrr@bintr
1850 \fi

```

\pxrr@auto@\* の設定法は和文ルビと欧文ルビで処理が異なる。

```

1851 \ifpxrr@abody
1852 \pxrr@intrude@setauto@a
1853 \else
1854 \pxrr@intrude@setauto@j
1855 \fi

```

実際に項目の出力を行う。

段落冒頭の場合、! 指定 (pxrr@bfintr が真) ならば進入のための負のグルーを入れる (他の項目は入れない)。

```

1856 \ifpxrr@par@head
1857 \ifpxrr@bfintr
1858 \hskip-\pxrr@intr@amount\relax
1859 \fi

```

段落冒頭でない場合、字間空きのグルー、進入用のグルーを順番に入れる。

※ ペナルティは \pxrr@put@head@penalty で既に入れている。

```

1860 \else
1861 % \penalty\pxrr@auto@penalty\relax
1862 \hskip-\pxrr@intr@amount\relax
1863 \hskip\pxrr@auto@icspace\relax
1864 \fi
1865 \fi
1866 }

```

\pxrr@put@head@penalty 前側に補助指定で定められた値のペナルティを置く。現在位置に既にペナルティがある場合は合算する。

```

1867 \def\pxrr@put@head@penalty{%
1868 \ifpxrr@ghost\else \ifpxrr@par@head\else
1869 \ifpxrr@abody
1870 \pxrr@intrude@setauto@a
1871 \else
1872 \pxrr@intrude@setauto@j
1873 \fi
1874 \ifnum\pxrr@auto@penalty=\z@\else
1875 \pxrr@canta\lastpenalty \unpenalty
1876 \advance\pxrr@canta\pxrr@auto@penalty\relax
1877 \penalty\pxrr@canta
1878 \fi
1879 \fi\fi
1880 }

```

#### 4.17.2 後側進入処理

\pxrr@intrude@end 末尾での進入処理。

```
1881 \def\pxrr@intrude@end{%
1882   \ifpxrr@ghost\else
```

実効の進入幅は \pxrr@aintr と \pxrr@aspace の小さい方。

```
1883   \let\pxrr@intr@amount\pxrr@aspace
1884   \ifdim\pxrr@aintr<\pxrr@intr@amount\relax
1885     \let\pxrr@intr@amount\pxrr@aintr
1886   \fi
```

\pxrr@auto@\* の設定法は和文ルビと欧文ルビで処理が異なる。

```
1887   \pxrr@csletcs{ifpxrr@bnober}{ifpxrr@anober}%
1888   \let\pxrr@bscomp\pxrr@ascomp
1889   \ifpxrr@abody
1890     \pxrr@intrude@setauto@a
1891   \else
1892     \pxrr@intrude@setauto@j
1893   \fi
```

直後の文字の前禁則ペナルティが、挿入されるグルーの前に入るようにする。

```
1894   \ifnum\pxrr@auto@penalty=\z@
1895     \let\pxrr@auto@penalty\pxrr@end@kinsoku
1896   \fi
1897   \ifpxrr@afintr
```

段落末尾での進入を許す場合。

```
1898     \ifnum\pxrr@auto@penalty=\z@\else
1899       \penalty\pxrr@auto@penalty\relax
1900     \fi
1901     \kern-\pxrr@intr@amount\relax
```

段落末尾では次のグルーを消滅させる（前のカーンは残る）。そのため、禁則ペナルティがある（段落末尾ではあり得ない）場合にのみその次のペナルティ 20000 を置く。本物の禁則ペナルティはこれに加算されるが、合計値は 10000 以上になるのでこの位置での行分割が禁止される。

```
1902     \hskip\pxrr@auto@icspace\relax
1903     \ifnum\pxrr@auto@penalty=\z@\else
1904       \penalty\@MM
1905     \fi
1906   \else
```

段落末尾での進入を許さない場合。

```
1907     \@tempkipa-\pxrr@intr@amount\relax
1908     \advance\@tempkipa\pxrr@auto@icspace\relax
1909     \ifnum\pxrr@auto@penalty=\z@\else
1910       \penalty\pxrr@auto@penalty\relax
1911     \fi
1912     \hskip\@tempkipa
1913     \ifnum\pxrr@auto@penalty=\z@\else
1914       \penalty\@MM
1915     \fi
```

```

1916     \fi
1917     \fi
1918 }

```

## 4.18 メインです

### 4.18.1 エントリーポイント

`\ruby` 和文ルビの公開命令。`\jruby` を頑強な命令として定義した上で、`\ruby` はそれに展開されるマクロに（未定義ならば）定義する。

```

1919 \AtBeginDocument{%
1920   \providecommand*\ruby{\jruby}%
1921 }
1922 \newcommand*\jruby{%
1923   \pxrr@jprologue
1924   \pxrr@trubyfalse
1925   \pxrr@ruby
1926 }

```

頑強にするために、先に定義した `\pxrr@add@protect` を用いる。

```

1927 \pxrr@add@protect\jruby

```

`\aruby` 欧文ルビの公開命令。こちらも頑強な命令にする。

```

1928 \newcommand*\aruby{%
1929   \pxrr@aprologue
1930   \pxrr@trubyfalse
1931   \pxrr@ruby
1932 }
1933 \pxrr@add@protect\aruby

```

`\truby` 和文両側ルビの公開命令。

```

1934 \newcommand*\truby{%
1935   \pxrr@jprologue
1936   \pxrr@trubytrue
1937   \pxrr@ruby
1938 }
1939 \pxrr@add@protect\truby

```

`\atruby` 欧文両側ルビの公開命令。

```

1940 \newcommand*\atruby{%
1941   \pxrr@aprologue
1942   \pxrr@trubytrue
1943   \pxrr@ruby
1944 }
1945 \pxrr@add@protect\atruby

```

`\ifpxrr@truby` 両側ルビであるか。スイッチ。`\pxrr@parse@option` で `\pxrr@side` を適切に設定するために使われる。

```

1946 \newif\ifpxrr@truby

```

`\pxrr@option` オプションおよび第 2 オプションを格納するマクロ。

```
\pxrr@exoption 1947 \let\pxrr@option\@empty
                1948 \let\pxrr@exoption\@empty
```

`\pxrr@do@proc` `\pxrr@ruby` の処理中に使われる。

```
\pxrr@do@scan 1949 \let\pxrr@do@proc\@empty
                1950 \let\pxrr@do@scan\@empty
```

`\pxrr@ruby` `\ruby` および `\aruby` の共通の下請け。オプションの処理を行う。  
オプションを読みマクロに格納する。

```
1951 \def\pxrr@ruby{%
1952   \@testopt\pxrr@ruby@a{}%
1953 }
1954 \def\pxrr@ruby@a[#1]{%
1955   \def\pxrr@option{#1}%
1956   \@testopt\pxrr@ruby@b{}%
1957 }
1958 \def\pxrr@ruby@b[#1]{%
1959   \def\pxrr@exoption{#1}%
1960   \ifpxrr@truby
1961     \let\pxrr@do@proc\pxrr@truby@proc
1962     \let\pxrr@do@scan\pxrr@truby@scan
1963   \else
1964     \let\pxrr@do@proc\pxrr@ruby@proc
1965     \let\pxrr@do@scan\pxrr@ruby@scan
1966   \fi
1967   \pxrr@ruby@c
1968 }
1969 \def\pxrr@ruby@c{%
1970   \ifpxrr@ghost
1971     \expandafter\pxrr@do@proc
1972   \else
1973     \expandafter\pxrr@do@scan
1974   \fi
1975 }
```

`\pxrr@mode@is@switching` `\if\pxrr@mode@is@switching{<基本モード>}` の形の if 文として使う。モードが“選択的” (M・J) であるか。

```
1976 \def\pxrr@mode@is@switching{%
1977   \if M\pxrr@mode T%
1978   \else\if J\pxrr@mode T%
1979   \else F%
1980   \fi\fi T%
1981 }
```

`\pxrr@bind@param` “呼出時変数” へのコピーを行う。

```
1982 \def\pxrr@bind@param{%
```

圏点ルビ同時付加フラグの処理。圏点側が指定した `apply@combo` の値を“呼出時パラメタ”の `pxrr@combo` に移動させる。

```
1983 \ifpxrr@apply@combo
1984   \pxrr@apply@combofalse
1985   \pxrr@combotrue
1986   \pxrr@ck@bind@param
1987 \else
1988   \pxrr@combofalse
1989 \fi
1990 \let\pxrr@c@ruby@font\pxrr@ruby@font
1991 \let\pxrr@c@size@ratio\pxrr@size@ratio
1992 \let\pxrr@c@inter@gap\pxrr@inter@gap
1993 }
```

`\pxrr@ruby@proc` `\pxrr@ruby@proc{<親文字列>}{<ルビ文字列>}` : これが手続の本体となる。

```
1994 \def\pxrr@ruby@proc#1#2{%
1995   \pxrr@prepare@fallback{#1}%
      フォントサイズの変数を設定して、
1996   \pxrr@bind@param
1997   \pxrr@assign@fsize
      オプションを解析する。
1998   \pxrr@parse@option\pxrr@option
      ルビ文字入力をグループ列に分解する。
1999   \pxrr@decompbar{#2}%
2000   \let\pxrr@ruby@list\pxrr@res
2001   \edef\pxrr@ruby@count{\the\pxrr@cntr}%
2002   \let\pxrr@sruby@list\relax
      親文字入力をグループ列に分解する。
2003   \pxrr@decompbar{#1}%
2004   \let\pxrr@body@list\pxrr@res
2005   \edef\pxrr@body@count{\the\pxrr@cntr}%
      安全モードに関する処理を行う。
2006   \ifpxrr@safe@mode
2007     \pxrr@setup@safe@mode
2008   \fi
      モードが“選択的”である場合、“普通の”モード (m・j・g) に帰着させる。
2009   \if\pxrr@mode@is@switching
2010     \pxrr@resolve@mode
2011   \fi
2012 \ifpxrr@Debug
2013   \pxrr@debug@show@input
2014 \fi
      入力検査を行い、パスした場合は組版処理に進む。
2015   \pxrr@if@alive{%
```

```

2016 \if g\pxrr@mode
2017 \pxrr@ruby@check@g
2018 \pxrr@if@alive{%
2019 \ifnum\pxrr@body@count>\@ne
2020 \pxrr@ruby@main@mg
2021 \else
2022 \pxrr@ruby@main@g
2023 \fi
2024 }%
2025 \else
2026 \pxrr@ruby@check@m
2027 \pxrr@if@alive{\pxrr@ruby@main@m}%
2028 \fi
2029 }%

    後処理を行う。

2030 \pxrr@ruby@exit
2031 }

```

`\pxrr@truby@proc` `\pxrr@ruby@proc{<親文字列>}{<上側ルビ文字列>}{<下側ルビ文字列>}` : 両側ルビの場合の  
 手続の本体。

```

2032 \def\pxrr@truby@proc#1#2#3{%
2033 \pxrr@prepare@fallback{#1}%

    フォントサイズの変数を設定して、

2034 \pxrr@bind@param
2035 \pxrr@assign@fsize

    オプションを解析する。

2036 \pxrr@parse@option\pxrr@option

    両側のグループルビでは pxrr@all@input を利用するので、入力文字列を設定する。

2037 \def\pxrr@all@input{#{1}{#2}{#3}}%

    入力文字列のグループ分解を行う。

2038 \pxrr@decompbar{#3}%
2039 \let\pxrr@sruby@list\pxrr@res
2040 \edef\pxrr@sruby@count{\the\pxrr@cntr}%
2041 \pxrr@decompbar{#2}%
2042 \let\pxrr@ruby@list\pxrr@res
2043 \edef\pxrr@ruby@count{\the\pxrr@cntr}%
2044 \pxrr@decompbar{#1}%
2045 \let\pxrr@body@list\pxrr@res
2046 \edef\pxrr@body@count{\the\pxrr@cntr}%

    安全モードに関する処理を行う。

2047 \ifpxrr@safe@mode
2048 \pxrr@setup@safe@mode
2049 \fi
2050 \if\pxrr@mode@is@switching
2051 \pxrr@resolve@mode

```

```

2052 \fi
2053 \ifpxrrDebug
2054 \pxrr@debug@show@input
2055 \fi

```

入力検査を行い、パスした場合は組版処理に進む。

```

2056 \pxrr@if@alive{%
2057   \if g\pxrr@mode
2058     \pxrr@ruby@check@tg
2059     \pxrr@if@alive{\pxrr@ruby@main@tg}%
2060   \else
2061     \pxrr@ruby@check@tm
2062     \pxrr@if@alive{\pxrr@ruby@main@tm}%
2063   \fi
2064 }%

```

後処理を行う。

```

2065 \pxrr@ruby@exit
2066 }

```

`\pxrr@setup@safe@mode` 安全モード用の設定。

```

2067 \def\pxrr@setup@safe@mode{%

```

単純グループルビに強制的に変更する。これに応じて、親文字列とルビ文字列のグループを1つに集成する。

```

2068 \let\pxrr@mode=g\relax
2069 \pxrr@unite@group\pxrr@body@list
2070 \def\pxrr@body@count{1}%
2071 \pxrr@unite@group\pxrr@ruby@list
2072 \def\pxrr@ruby@count{1}%
2073 \ifx\pxrr@sruby@list\relax\else
2074   \pxrr@unite@group\pxrr@sruby@list
2075   \def\pxrr@sruby@count{1}%
2076 \fi

```

“文字単位のスキャン”が必要な機能を無効にする。

```

2077 \chardef\pxrr@evensp\z@
2078 \chardef\pxrr@revensp\z@
2079 \chardef\pxrr@fullsize\z@
2080 }

```

`\pxrr@resolve@mode` 基本モードが“選択的”(M・J)である場合に、状況に応じて適切な通常モードに切り替える。

```

2081 \def\pxrr@resolve@mode{%
2082   \ifnum\pxrr@body@count=\@ne

```

ルビグループが1つで親文字が複数ある場合にはグループルビを選択し、

```

2083   \ifnum\pxrr@ruby@count=\@ne
2084     \let\pxrr@pre\pxrr@decompose
2085     \let\pxrr@post\relax

```

```

2086     \pxrr@body@list
2087     \ifnum\pxrr@cntr=\@ne\else
2088         \let\pxrr@mode=g%
2089     \fi
2090 \fi

```

それ以外はモノルビ・熟語ルビを選択する。

```

2091     \if M\pxrr@mode \let\pxrr@mode=m\fi
2092     \if J\pxrr@mode \let\pxrr@mode=j\fi
2093 \ifpxrrDebug
2094     \pxrr@debug@show@resolve@mode
2095 \fi

```

\pxrr@check@option で行っている調整をやり直す。

```

2096     \if g\pxrr@mode
2097         \chardef\pxrr@athead\z@
2098     \fi
2099     \if g\pxrr@mode\else
2100         \chardef\pxrr@evensp\@ne
2101     \fi
2102 \else
2103     \pxrr@fatal@bad@switching
2104 \fi
2105 }

```

#### 4.18.2 入力検査

グループ・文字の個数の検査を行う手続。

\pxrr@ruby@check@g グループルビの場合、ルビ文字グループと親文字グループの個数が一致する必要がある。さらに、グループが複数（可動グループルビ）にできるのは、和文ルビであり、しかも拡張機能が有効である場合に限られる。

```

2106 \def\pxrr@ruby@check@g{%
2107     \ifnum\pxrr@body@count=\pxrr@ruby@count\relax
2108     \ifnum\pxrr@body@count=\@ne\else
2109         \ifpxrr@abody
2110             \pxrr@fatal@bad@movable
2111         \else\ifnum\pxrr@extra=\z@
2112             \pxrr@fatal@na@movable
2113         \fi\fi
2114     \fi
2115 \else
2116     \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
2117 \fi
2118 }

```

\pxrr@ruby@check@m モノルビ・熟語ルビの場合、親文字列は単一のグループからなる必要がある。さらに、親文字列の《文字》の個数とルビ文字列のグループの個数が一致する必要がある。

```

2119 \def\pxrr@ruby@check@m{%

```

```
2120 \ifnum\pxrr@body@count=\@ne
```

ここで \pxrr@body@list/count を文字ごとの分解に置き換える。

```
2121 \let\pxrr@pre\pxrr@decompose
2122 \let\pxrr@post\relax
2123 \pxrr@body@list
2124 \let\pxrr@body@list\pxrr@res
2125 \edef\pxrr@body@count{\the\pxrr@cntr}%
2126 \ifnum\pxrr@body@count=\pxrr@ruby@count\relax\else
2127 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
2128 \fi
2129 \else
2130 \pxrr@fatal@bad@mono
2131 \fi
2132 }
```

\pxrr@ruby@check@tg 両側のグループルビの場合。ルビが2つあることを除き、片側の場合と同じ。

```
2133 \def\pxrr@ruby@check@tg{%
2134 \ifnum\pxrr@body@count=\pxrr@ruby@count\relax\else
2135 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
2136 \fi
2137 \ifnum\pxrr@body@count=\pxrr@sruby@count\relax\else
2138 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@sruby@count
2139 \fi
2140 \pxrr@if@alive{%
2141 \ifnum\pxrr@body@count=\@ne\else
2142 \ifpxrr@abody
2143 \pxrr@fatal@bad@movable
2144 \else\ifnum\pxrr@extra=\z@
2145 \pxrr@fatal@na@movable
2146 \fi\fi
2147 \fi
2148 }%
2149 }
```

\pxrr@ruby@check@tm 両側のモノルビの場合。ルビが2つあることを除き、片側の場合と同じ。

```
2150 \def\pxrr@ruby@check@tm{%
2151 \ifnum\pxrr@body@count=\@ne
2152 \let\pxrr@pre\pxrr@decompose
2153 \let\pxrr@post\relax
2154 \pxrr@body@list
2155 \let\pxrr@body@list\pxrr@res
2156 \edef\pxrr@body@count{\the\pxrr@cntr}%
2157 \ifnum\pxrr@body@count=\pxrr@ruby@count\relax\else
2158 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
2159 \fi
2160 \ifnum\pxrr@body@count=\pxrr@sruby@count\relax\else
2161 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@sruby@count
2162 \fi
```

```

2163 \else
2164 \pxrr@fatal@bad@mono
2165 \fi
2166 }

```

#### 4.18.3 ルビ組版処理

`\ifpxrr@par@head` ルビ付文字列の出力位置が段落の先頭であるか。

```

2167 \newif\ifpxrr@par@head

```

`\pxrr@check@par@head` 現在の位置に基づいて `\ifpxrr@par@head` の値を設定する。当然、何らかの出力を行う前に呼ぶ必要がある。

```

2168 \def\pxrr@check@par@head{%
2169 \ifvmode
2170 \pxrr@par@headtrue
2171 \else
2172 \pxrr@par@headfalse
2173 \fi
2174 }

```

`\pxrr@if@last` `\pxrr@if@last{⟨真⟩}{⟨偽⟩}`: `\pxrr@pre/inter` の本体として使い、それが最後の `\pxrr@pre/inter` である (`\pxrr@post` の直前にある) 場合に `⟨真⟩`、ない場合に `⟨偽⟩` に展開される。このマクロの呼出は `\pxrr@preinterpre` の本体の末尾でなければならない。

```

2175 \def\pxrr@if@last#1#2#3{%
2176 \ifx#3\pxrr@post #1%
2177 \else #2%
2178 \fi
2179 #3%
2180 }

```

`\pxrr@inter@mono` モノルビのブロック間に挿入される空き。和文間空白とする。

```

2181 \def\pxrr@inter@mono{%
2182 \hskip\pxrr@iiskip\relax
2183 }

```

`\pxrr@takeout@any@protr` `\ifpxrr@any@protr` の値を `\pxrr@hbox` の外に出す。

※ `color` 不使用時は `\hbox` による 1 段のグループだけ処理すればよいが、`color` 使用時は `\color@begingroup`~`\color@endgroup` によるグループが生じるので、2 段分の処理が必要。

`color` 不使用時の定義。

```

2184 \def\pxrr@takeout@any@protr@nocolor{%
2185 \ifpxrr@any@protr
2186 \aftergroup\pxrr@any@protrtrue
2187 \fi
2188 }

```

`color` 使用時の定義。

```

2189 \def\pxrr@takeout@any@protr{%
2190   \ifpxrr@any@protr
2191     \aftergroup\pxrr@takeout@any@protr@a
2192   \fi
2193 }
2194 \def\pxrr@takeout@any@protr@a{%
2195   \aftergroup\pxrr@any@protrtrue
2196 }

```

\pxrr@ruby@main@m モノルビ。

```

2197 \def\pxrr@ruby@main@m{%
2198   \pxrr@zip@list\pxrr@body@list\pxrr@ruby@list
2199   \let\pxrr@whole@list\pxrr@res
2200   \pxrr@check@par@head
2201   \pxrr@put@head@penalty
2202   \pxrr@any@protrfalse
2203 \ifpxrrDebug
2204 \pxrr@debug@show@recomp
2205 \fi

```

\ifpxrr@?intr の値に応じて \pxrr@locate@\*% の値を決定する。なお、両側で突出を禁止するのは不可であることに注意。

```

2206 \let\pxrr@locate@head@\pxrr@locate@inner
2207 \let\pxrr@locate@end@\pxrr@locate@inner
2208 \let\pxrr@locate@sing@\pxrr@locate@inner
2209 \ifpxrr@aprotr\else
2210   \let\pxrr@locate@end@\pxrr@locate@end
2211   \let\pxrr@locate@sing@\pxrr@locate@end
2212 \fi
2213 \ifpxrr@bprotr\else
2214   \let\pxrr@locate@head@\pxrr@locate@head
2215   \let\pxrr@locate@sing@\pxrr@locate@head
2216 \fi
2217 \def\pxrr@pre##1##2{%
2218   \pxrr@if@last{%

```

単独ブロックの場合。

```

2219   \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
2220   \pxrr@intrude@head
2221   \unhbox\pxrr@boxr
2222   \pxrr@intrude@end
2223   \pxrr@takeout@any@protr
2224   }{%

```

先頭ブロックの場合。

```

2225   \pxrr@compose@block\pxrr@locate@head@{##1}{##2}%
2226   \pxrr@intrude@head
2227   \unhbox\pxrr@boxr
2228   }%
2229   }%

```

```
2230 \def\pxrr@inter##1##2{%
```

```
2231 \pxrr@if@last{%
```

末尾ブロックの場合。

```
2232 \pxrr@compose@block\pxrr@locate@end@{##1}{##2}%
```

```
2233 \pxrr@inter@mono
```

```
2234 \unhbox\pxrr@boxr
```

```
2235 \pxrr@intrude@end
```

```
2236 \pxrr@takeout@any@protr
```

```
2237 }{%
```

中間ブロックの場合。

```
2238 \pxrr@compose@block\pxrr@locate@inner{##1}{##2}%
```

```
2239 \pxrr@inter@mono
```

```
2240 \unhbox\pxrr@boxr
```

```
2241 }%
```

```
2242 }%
```

```
2243 \let\pxrr@post\@empty
```

```
2244 \setbox\pxrr@boxr\pxrr@hbox{\pxrr@whole@list}%
```

熟語ルビ指定の場合、`\ifpxrr@any@protr` が真である場合は再調整する。

```
2245 \if j\pxrr@mode
```

```
2246 \ifpxrr@any@protr
```

```
2247 \pxrr@ruby@redo@j
```

```
2248 \fi
```

```
2249 \fi
```

```
2250 \unhbox\pxrr@boxr
```

```
2251 }
```

`\pxrr@ruby@redo@j` モノルビ処理できない（ルビが長くなるブロックがある）熟語ルビを適切に組みなおす。現状では、単純にグループルビの組み方にする。

```
2252 \def\pxrr@ruby@redo@j{%
```

```
2253 \pxrr@concat@list\pxrr@body@list
```

```
2254 \let\pxrr@body@list\pxrr@res
```

```
2255 \pxrr@concat@list\pxrr@ruby@list
```

```
2256 \let\pxrr@ruby@list\pxrr@res
```

```
2257 \pxrr@zip@single\pxrr@body@list\pxrr@ruby@list
```

```
2258 \let\pxrr@whole@list\pxrr@res
```

```
2259 \ifpxrr@Debug
```

```
2260 \pxrr@debug@show@concat
```

```
2261 \fi
```

```
2262 \let\pxrr@locate@sing@\pxrr@locate@inner
```

```
2263 \ifpxrr@aprotr\else
```

```
2264 \let\pxrr@locate@sing@\pxrr@locate@end
```

```
2265 \fi
```

```
2266 \ifpxrr@bprotr\else
```

```
2267 \let\pxrr@locate@sing@\pxrr@locate@head
```

```
2268 \fi
```

```
2269 \def\pxrr@pre##1##2{%
```

```
2270 \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
```

```

2271 \pxrr@intrude@head
2272 \unhbox\pxrr@boxr
2273 \pxrr@intrude@end
2274 }%
2275 \let\pxrr@inter\@undefined
2276 \let\pxrr@post\@empty
2277 \setbox\pxrr@boxr\pxrr@hbox{\pxrr@whole@list}%
2278 }

```

\pxrr@ruby@main@g 単純グループルビの場合。

グループが1つしかない前提なので多少冗長となるが、基本的に \pxrr@ruby@main@m の処理を踏襲する。

```

2279 \def\pxrr@ruby@main@g{%
2280 \pxrr@zip@list\pxrr@body@list\pxrr@ruby@list
2281 \let\pxrr@whole@list\pxrr@res
2282 \pxrr@check@par@head
2283 \pxrr@put@head@penalty
2284 \ifpxrr@Debug
2285 \pxrr@debug@show@recomp
2286 \fi
2287 \let\pxrr@locate@sing@\pxrr@locate@inner
2288 \ifpxrr@aprotr\else
2289 \let\pxrr@locate@sing@\pxrr@locate@end
2290 \fi
2291 \ifpxrr@bprotr\else
2292 \let\pxrr@locate@sing@\pxrr@locate@head
2293 \fi
2294 \def\pxrr@pre##1##2{%
2295 \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
2296 \pxrr@intrude@head
2297 \unhbox\pxrr@boxr
2298 \pxrr@intrude@end
2299 }%
2300 \let\pxrr@inter\@undefined
2301 \let\pxrr@post\@empty

```

グループルビは \ifpxrr@any@protr の判定が不要なので直接出力する。

```

2302 \pxrr@whole@list
2303 }

```

\pxrr@ruby@main@tm 両側のモノルビの場合。

```

2304 \def\pxrr@ruby@main@tm{%
2305 \pxrr@tzip@list\pxrr@body@list\pxrr@ruby@list\pxrr@sruby@list
2306 \let\pxrr@whole@list\pxrr@res
2307 \pxrr@check@par@head
2308 \pxrr@any@protrfalse
2309 \ifpxrr@Debug
2310 \pxrr@debug@show@recomp
2311 \fi

```

```

2312 \let\pxrr@locate@head@\pxrr@locate@inner
2313 \let\pxrr@locate@end@\pxrr@locate@inner
2314 \let\pxrr@locate@sing@\pxrr@locate@inner
2315 \ifpxrr@aprotr\else
2316   \let\pxrr@locate@end@\pxrr@locate@end
2317   \let\pxrr@locate@sing@\pxrr@locate@end
2318 \fi
2319 \ifpxrr@bprotr\else
2320   \let\pxrr@locate@head@\pxrr@locate@head
2321   \let\pxrr@locate@sing@\pxrr@locate@head
2322 \fi
2323 \def\pxrr@pre##1##2##3{%
2324   \pxrr@if@last{%
2325     \pxrr@compose@twoside@block\pxrr@locate@sing@
2326     {##1}{##2}{##3}%
2327     \pxrr@intrude@head
2328     \unhbox\pxrr@boxr
2329     \pxrr@intrude@end
2330     \pxrr@takeout@any@protr
2331   }{%
2332     \pxrr@compose@twoside@block\pxrr@locate@head@
2333     {##1}{##2}{##3}%
2334     \pxrr@intrude@head
2335     \unhbox\pxrr@boxr
2336   }%
2337 }%
2338 \def\pxrr@inter##1##2##3{%
2339   \pxrr@if@last{%
2340     \pxrr@compose@twoside@block\pxrr@locate@end@
2341     {##1}{##2}{##3}%
2342     \pxrr@inter@mono
2343     \unhbox\pxrr@boxr
2344     \pxrr@intrude@end
2345     \pxrr@takeout@any@protr
2346   }{%
2347     \pxrr@compose@twoside@block\pxrr@locate@inner
2348     {##1}{##2}{##3}%
2349     \pxrr@inter@mono
2350     \unhbox\pxrr@boxr
2351   }%
2352 }%
2353 \let\pxrr@post\@empty
2354 \setbox\pxrr@boxr\pxrr@hbox{\pxrr@whole@list}%
2355 \unhbox\pxrr@boxr
2356 }

```

\pxrr@ruby@main@tg 両側の単純グループビの場合。

```

2357 \def\pxrr@ruby@main@tg{%
2358   \pxrr@check@par@head

```

```

2359 \pxrr@put@head@penalty
2360 \let\pxrr@locate@sing@\pxrr@locate@inner
2361 \ifpxrr@aprotr\else
2362   \let\pxrr@locate@sing@\pxrr@locate@end
2363 \fi
2364 \ifpxrr@bprotr\else
2365   \let\pxrr@locate@sing@\pxrr@locate@head
2366 \fi
2367 \expandafter\pxrr@compose@twoside@block\expandafter\pxrr@locate@sing@
2368 \pxrr@all@input
2369 \pxrr@intrude@head
2370 \unhbox\pxrr@boxr
2371 \pxrr@intrude@end
2372 }

```

`\pxrr@ruby@main@mg` 未実装（呼出もない）。

```
2373 \let\pxrr@ruby@main@mg\undefined
```

#### 4.18.4 前処理

ゴースト処理する。そのため、展開不能命令が…。

`\ifpxrr@ghost` 実行中のルビ命令でゴースト処理が有効か。

```
2374 \newif\ifpxrr@ghost
```

`\pxrr@jprologue` 和文ルビ用の開始処理。

```
2375 \def\pxrr@jprologue{%
```

ゴースト処理を行う場合、一番最初に現れる展開不能トークンがゴースト文字（全角空白）であることが肝要である。

```

2376 \ifpxrr@jghost
2377   \pxrr@jghost@char
2378   \pxrr@inhibitglue
2379 \fi

```

ルビの処理の本体は全てこのグループの中で行われる。

```

2380 \begingroup
2381   \pxrr@abodyfalse
2382   \pxrr@csletcs{ifpxrr@ghost}{ifpxrr@jghost}%

```

出力した全角空白の幅だけ戻しておく。

```

2383 \ifpxrr@jghost
2384   \setbox\pxrr@boxa\hbox{\pxrr@jghost@char}%
2385   \kern-\wd\pxrr@boxa
2386 \fi
2387 }

```

`\pxrr@aghost` 欧文用のゴースト文字の定義。合成語記号は T1 エンコーディングの位置 23 にある。従って、T1 のフォントが必要になるが、ここでは Latin Modern Roman を 2.5 pt のサイズで用

いる。極小のサイズにしているのは、合成語記号の高さが影響する可能性を避けるためである。LM フォントの TeX フォント名は版により異なるようなので、NFSS を通して目的のフォントの fontdef を得ている。(グループ内で `\usefont{T1}{lmr}{m}{n}` を呼んでおくと、大域的に `\T1/lmr/m/n/2.5` が定義される。)

```

2388 \chardef\pxrr@aghostchar=23 % compwordmark
2389 \let\pxrr@aghost\relax
2390 \let\pxrr@aghostfont\relax
2391 \def\pxrr@setup@aghost{%
2392   \global\let\pxrr@setup@aghost\relax
2393   \IfFileExists{t1lmr.fd}{%
2394     \begingroup
2395       \fontsize{2.5}{0}\usefont{T1}{lmr}{m}{n}%
2396     \endgroup
2397     \global\pxrr@letcs\pxrr@aghostfont{T1/lmr/m/n/2.5}%
2398     \gdef\pxrr@aghost{\pxrr@aghostfont\pxrr@aghostchar}%
2399     \global\xspcode\pxrr@aghostchar=3 %
2400   }{%else
2401     \pxrr@warn{Ghost embedding for \string\aruby\space
2402       is disabled,\MessageBreak
2403       since package lmodern is missing}%
2404     \global\pxrr@aghostfalse
2405     \global\let\pxrr@aghosttrue\relax
2406   }%
2407 }

```

`\pxrr@aprologue` 欧文ルビ用の開始処理。

```

2408 \def\pxrr@aprologue{%
2409   \ifpxrr@aghost
2410     \pxrr@aghost
2411     \fi
2412   \begingroup
2413     \pxrr@abodytrue
2414     \pxrr@csletcs{ifpxrr@ghost}{ifpxrr@aghost}%
2415 }

```

#### 4.18.5 後処理

ゴースト処理する。

`\pxrr@ruby@exit` 出力を終えて、最後に呼ばれるマクロ。致命的エラーが起こった場合はフォールバック処理を行う。その後は、和文ルビと欧文ルビで処理が異なる。

```

2416 \def\pxrr@ruby@exit{%
2417   \ifpxrr@fatal@error
2418     \pxrr@fallback
2419     \fi
2420   \ifpxrr@abody
2421     \expandafter\pxrr@aepilogue
2422   \else

```

```

2423 \expandafter\pxrr@jepilogue
2424 \fi
2425 }

```

`\pxrr@jepilogue` 和文の場合の終了処理。開始処理と同様、全角空白をゴースト文字に用いる。

```

2426 \def\pxrr@jepilogue{%
2427 \ifpxrr@jghost
2428 \setbox\pxrr@boxa\hbox{\pxrr@jghost@char}%
2429 \kern-\wd\pxrr@boxa
2430 \fi

```

`\pxrr@?prologue` の中の `\begingroup` で始まるグループを閉じる。

```

2431 \endgroup
2432 \ifpxrr@jghost
2433 \pxrr@inhibitglue
2434 \pxrr@jghost@char
2435 \fi
2436 }

```

`\pxrr@aepilogue` 欧文の場合の終了処理。合成語記号をゴースト文字に用いる。

```

2437 \def\pxrr@aepilogue{%
2438 \endgroup
2439 \ifpxrr@aghost
2440 \pxrr@aghost
2441 \fi
2442 }

```

#### 4.19 デバッグ用出力

```

2443 \def\pxrr@debug@show@input{%
2444 \typeout{---\pxrr@pkgname\space input:^^J%
2445 \ifpxrr@abody = \meaning\ifpxrr@abody^^J%
2446 \ifpxrr@truby = \meaning\ifpxrr@truby^^J%
2447 pxrr@ruby@fsize = \pxrr@ruby@fsize^^J%
2448 pxrr@body@zw = \pxrr@body@zw^^J%
2449 pxrr@ruby@zw = \pxrr@ruby@zw^^J%
2450 pxrr@iiskip = \pxrr@iiskip^^J%
2451 pxrr@iaiskip = \pxrr@iaiskip^^J%
2452 pxrr@htratio = \pxrr@htratio^^J%
2453 pxrr@ruby@raise = \pxrr@ruby@raise^^J%
2454 pxrr@ruby@lower = \pxrr@ruby@lower^^J%
2455 \ifpxrr@bprotr = \meaning\ifpxrr@bprotr^^J%
2456 \ifpxrr@aprotr = \meaning\ifpxrr@aprotr^^J%
2457 pxrr@side = \the\pxrr@side^^J%
2458 pxrr@evensp = \the\pxrr@evensp^^J%
2459 pxrr@fullsize = \the\pxrr@fullsize^^J%
2460 pxrr@bscomp = \meaning\pxrr@bscomp^^J%
2461 pxrr@ascomp = \meaning\pxrr@ascomp^^J%
2462 \ifpxrr@bnoobr = \meaning\ifpxrr@bnoobr^^J%

```

```

2463 ifpxrr@anobr = \meaning\ifpxrr@anobr^^J%
2464 ifpxrr@bfintr = \meaning\ifpxrr@bfintr^^J%
2465 ifpxrr@afintr = \meaning\ifpxrr@afintr^^J%
2466 pxrr@bintr = \pxrr@bintr^^J%
2467 pxrr@aintr = \pxrr@aintr^^J%
2468 pxrr@ahead = \the\pxrr@ahead^^J%
2469 pxrr@mode = \meaning\pxrr@mode^^J%
2470 ifpxrr@ahead@given = \meaning\ifpxrr@ahead@given^^J%
2471 ifpxrr@mode@given = \meaning\ifpxrr@mode@given^^J%
2472 pxrr@body@list = \meaning\pxrr@body@list^^J%
2473 pxrr@body@count = \@nameuse{pxrr@body@count}^^J%
2474 pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
2475 pxrr@ruby@count = \@nameuse{pxrr@ruby@count}^^J%
2476 pxrr@end@kinsoku = \pxrr@end@kinsoku^^J%
2477 ----
2478 }%
2479 }
2480 \def\pxrr@debug@show@recomp{%
2481 \typeout{----\pxrr@pkgname\space recomp:^^J%
2482 pxrr@body@list = \meaning\pxrr@body@list^^J%
2483 pxrr@body@count = \pxrr@body@count^^J%
2484 pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
2485 pxrr@ruby@count = \pxrr@ruby@count^^J%
2486 pxrr@res = \meaning\pxrr@res^^J%
2487 ----
2488 }%
2489 }
2490 \def\pxrr@debug@show@concat{%
2491 \typeout{----\pxrr@pkgname\space concat:^^J%
2492 pxrr@body@list = \meaning\pxrr@body@list^^J%
2493 pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
2494 pxrr@whole@list = \meaning\pxrr@whole@list^^J%
2495 ----
2496 }%
2497 }
2498 \def\pxrr@debug@show@resolve@mode{%
2499 \typeout{----\pxrr@pkgname\space resolve-mode:
2500 \meaning\pxrr@mode}%
2501 }

```

## 5 実装（圏点関連）

### 5.1 エラーメッセージ

指定の名前の圏点文字が未登録の場合。

```

2502 \def\pxrr@warn@na@kmark#1{%
2503 \pxrr@warn{Unavailable kenten mark '#1'}%
2504 }

```

パラメタ設定命令で無効な値が指定された場合。

```
2505 \def\pxrr@err@invalid@value#1{%
2506   \pxrr@error{Invalid value '#1'}%
2507   {\@eha}%
2508 }
```

## 5.2 パラメタ

### 5.2.1 全般設定

`\pxrr@k@ymark` 横組の主の圏点マークのコード。

```
2509 \let\pxrr@k@ymark\@undefined
```

`\pxrr@k@ysmark` 横組の副の圏点マークのコード。

```
2510 \let\pxrr@k@ysmark\@undefined
```

`\pxrr@k@tmark` 縦組の主の圏点マークのコード。

```
2511 \let\pxrr@k@tmark\@undefined
```

`\pxrr@k@tsmark` 縦組の副の圏点マークのコード。

```
2512 \let\pxrr@k@tsmark\@undefined
```

圏点マークの初期値の設定。

```
2513 \AtEndOfPackage{%
2514   \pxrr@k@get@mark\pxrr@k@ymark{bullet*}%
2515   \pxrr@k@get@mark\pxrr@k@ysmark{sesame*}%
2516   \pxrr@k@get@mark\pxrr@k@tmark{sesame*}%
2517   \pxrr@k@get@mark\pxrr@k@tsmark{bullet*}%
2518 }
```

`\pxrr@k@ruby@font` 圏点用フォント切替命令。

```
2519 \let\pxrr@k@ruby@font\@empty
```

`\pxrr@k@size@ratio` 圏点文字サイズ。 (`\kentensizeratio`)。実数値マクロ。

```
2520 \def\pxrr@k@size@ratio{0.5}
```

`\ifpxrr@k@ghost` ゴースト処理を行うか。スイッチ。

※ 圏点では和文ゴースト処理を必ず行う。

```
2521 \newif\ifpxrr@k@ghost \pxrr@k@ghosttrue
```

`\pxrr@k@inter@gap` 圏点と親文字の間の空き (`\kentenintergap`)。実数値マクロ。

```
2522 \def\pxrr@k@inter@gap{0}
```

`\pxrr@k@ruby@inter@gap` 圏点とルビの間の空き (`\kentenrubyintergap`)。実数値マクロ。

```
2523 \def\pxrr@k@ruby@inter@gap{0}
```

`\pxrr@k@d@side` 圏点を親文字の上下のどちらに付すか。0 = 上側 ; 1 = 下側。 `\kentensetup` の P/S の設定。整数定数。

```
2524 \chardef\pxrr@k@d@side=0
```

`\pxrr@k@d@mark` 圏点マークの種類。0 = 主 ; 1 = 副。 `\kentensetup` の `p/s` の設定。整数定数。

```
2525 \chardef\pxrr@k@d@mark=0
```

`\pxrr@k@ruby@combo` ルビと圏点が同時に適用された場合の挙動。0 = ルビだけ出力 ; 1 = ルビの上に圏点 (同時付加)。 `\kentenrubycombination` の設定値に対応する。整数定数。

```
2526 \chardef\pxrr@k@ruby@combo=1
```

`\pxrr@k@d@full` 約物にも圏点を付加するか。0 = 無効 ; 1 = 有効。 `\kentensetup` の `f/F` の設定。整数定数。

```
2527 \chardef\pxrr@k@d@full=0
```

### 5.2.2 呼出時の設定

`\kenten` の `P/S` の設定は、 `\pxrr@side` をルビと共用する。

`\pxrr@k@mark` 圏点マークの種類。0 = 主 ; 1 = 副。 `\kenten` の `p/s` の設定。整数定数。

```
2528 \chardef\pxrr@k@mark=0
```

`\pxrr@k@full` 約物にも圏点を付加するか。0 = 無効 ; 1 = 有効。 `\kenten` の `f/F` の設定。整数定数。

```
2529 \chardef\pxrr@k@full=0
```

`\pxrr@k@the@mark` 適用される圏点マークの命令。

```
2530 \let\pxrr@k@the@mark\relax
```

## 5.3 補助手続

### 5.3.1 \UTF 命令対応

`\ifpxrr@avail@UTF` \UTF 命令が利用できるか。スイッチ。

```
2531 \newif\ifpxrr@avail@UTF
```

`\pxrr@decide@avail@UTF` `\ifpxrr@avail@UTF` の値を確定させる。

```
2532 \def\pxrr@decide@avail@UTF{%
```

```
2533   \global\let\pxrr@decide@avail@UTF\relax
```

```
2534   \ifx\UTF\@undefined \global\pxrr@avail@UTFfalse
```

```
2535   \else \global\pxrr@avail@UTFtrue
```

```
2536   \fi
```

```
2537 }
```

### 5.3.2 リスト分解

`\pxrr@k@decompose` `\pxrr@k@decompose{<テキスト>}` : テキスト (圏点命令の引数) を分解した結果の圏点項目リストを `\pxrr@res` に返す。

※ 圏点項目リストの形式 :

```
\pxrr@entry[@XXX]{<引数>}……\pxrr@entry[@XXX]{<引数>}\pxrr@post
```

```
2538 \def\pxrr@k@decompose#1{%
```

```
2539   \let\pxrr@res\@empty
```

```

2540 \pxrr@cntr=\z@
2541 \pxrr@k@decompose@loopa#1\pxrr@end
2542 }
2543 \def\pxrr@k@decompose@loopa{%
2544 \futurelet\pxrr@token\pxrr@k@decompose@loopb
2545 }
2546 \def\pxrr@k@decompose@loopb{%
2547 \pxrr@cond\ifx\pxrr@token\pxrr@end\fi{%
2548 \pxrr@appto\pxrr@res{\pxrr@post}%
2549 }{\pxrr@if@kspan@cmd\pxrr@token{%
2550 \pxrr@k@decompose@special\pxrr@k@decompose@kspan
2551 }{\pxrr@if@ruby@cmd\pxrr@token{%
2552 \pxrr@k@decompose@special\pxrr@k@decompose@ruby
2553 }{\pxrr@if@truby@cmd\pxrr@token{%
2554 \pxrr@k@decompose@special\pxrr@k@decompose@truby
2555 }{\pxrr@if@kenten@cmd\pxrr@token{%
2556 \pxrr@k@decompose@special\pxrr@k@decompose@kenten
2557 }{\pxrr@cond\ifx\pxrr@token\@sptoken\fi{%
2558 \pxrr@k@decompose@loope
2559 }{%
2560 \pxrr@setok{\pxrr@ifx{\pxrr@token\bgroup}}}%
2561 \pxrr@k@decompose@loopc
2562 }}}}]}%
2563 }
2564 \def\pxrr@k@decompose@loopc#1{%
2565 \pxrr@appto\pxrr@res{\pxrr@entry}%
2566 \ifpxrr@ok
2567 \pxrr@appto\pxrr@res{{{#1}}}%
2568 \else
2569 \pxrr@appto\pxrr@res{#{#1}}%
2570 \fi
2571 \pxrr@k@decompose@loopd
2572 }
2573 \def\pxrr@k@decompose@loopd{%
2574 \advance\pxrr@cntr\@ne
2575 \pxrr@k@decompose@loopa
2576 }
2577 \expandafter\def\expandafter\pxrr@k@decompose@loope\space{%
2578 \pxrr@okfalse
2579 \pxrr@k@decompose@loopc{ }%
2580 }
2581 \def\pxrr@k@decompose@special#1#2#3{%
2582 #1{#2}%
2583 }
2584 \def\pxrr@k@decompose@kspan#1#2{%
2585 \pxrr@appto\pxrr@res{\pxrr@entry@kspan{#1{#2}}}%
2586 \pxrr@k@decompose@loopd
2587 }
2588 \def\pxrr@k@decompose@ruby#1#2#3{%

```

```

2589 \pxrr@appto\pxrr@res{\pxrr@entry@ruby{#1{#2}{#3}}}%
2590 \pxrr@k@decompose@loopd
2591 }
2592 \def\pxrr@k@decompose@truby#1#2#3#4{%
2593 \pxrr@appto\pxrr@res{\pxrr@entry@ruby{#1{#2}{#3}{#4}}}%
2594 \pxrr@k@decompose@loopd
2595 }
2596 \def\pxrr@k@decompose@kenten#1#2{%
2597 \pxrr@appto\pxrr@res{\pxrr@entry@kenten{#1{#2}}}%
2598 \pxrr@k@decompose@loopd
2599 }
2600 \def\pxrr@cmd@ruby{\jruby}
2601 \def\pxrr@cmd@kenten{jkenten}
2602 \def\pxrr@if@ruby@cmd#1{%
2603 \if \ifcat\noexpand#1\relax
2604 \ifx#1\pxrr@cmd@ruby T%
2605 \else\ifx#1\jruby T%
2606 \else\ifx#1\aruby T%
2607 \else F%
2608 \fi\fi\fi
2609 \else F%
2610 \fi T\expandafter\@firstoftwo
2611 \else \expandafter\@secondoftwo
2612 \fi
2613 }
2614 \def\pxrr@if@truby@cmd#1{%
2615 \if \ifcat\noexpand#1\relax
2616 \ifx#1\truby T%
2617 \else\ifx#1\atruby T%
2618 \else F%
2619 \fi\fi
2620 \else F%
2621 \fi T\expandafter\@firstoftwo
2622 \else \expandafter\@secondoftwo
2623 \fi
2624 }
2625 \def\pxrr@if@kspan@cmd#1{%
2626 \pxrr@cond\ifx#1\kspan\fi
2627 }
2628 \def\pxrr@if@kenten@cmd#1{%
2629 \if \ifcat\noexpand#1\relax
2630 \ifx#1\pxrr@cmd@kenten T%
2631 \else\ifx#1\jkenten T%
2632 \else F%
2633 \fi\fi
2634 \else F%
2635 \fi T\expandafter\@firstoftwo
2636 \else \expandafter\@secondoftwo
2637 \fi

```

2638 }

## 5.4 パラメタ設定公開命令

`\kentensetup` `\pxrr@k@parse@option` で解析した後、設定値を全般設定にコピーする。

```
2639 \newcommand*\kentensetup[1]{%
2640   \pxrr@in@setuptrue
2641   \pxrr@fatal@errorfalse
2642   \pxrr@k@parse@option{#1}%
2643   \ifpxrr@fatal@error\else
2644     \let\pxrr@k@d@side\pxrr@side
2645     \let\pxrr@k@d@mark\pxrr@k@mark
2646     \let\pxrr@k@d@full\pxrr@k@full
2647   \fi
```

`\ifpxrr@in@setup` を偽に戻す。ただし `\ifpxrr@fatal@error` は書き換えられたままであることに注意。

```
2648   \pxrr@in@setupfalse
2649 }
```

`\kentenfontsetup` 対応するパラメタを設定する。

```
2650 \newcommand*\kentenfontsetup{}
2651 \def\kentenfontsetup#{%
2652   \def\pxrr@k@ruby@font
2653 }
```

`\kentensizeratio` 対応するパラメタを設定する。

```
2654 \newcommand*\kentensizeratio[1]{%
2655   \edef\pxrr@k@size@ratio{#1}%
2656 }
```

`\kentenintergap` 対応するパラメタを設定する。

```
2657 \newcommand*\kentenintergap[1]{%
2658   \edef\pxrr@k@inter@gap{#1}%
2659 }
```

`\kentenrubyintergap` 対応するパラメタを設定する。

```
2660 \newcommand*\kentenrubyintergap[1]{%
2661   \edef\pxrr@k@ruby@inter@gap{#1}%
2662 }
```

`\kentenmarkinyoko` 対応するパラメタを設定する。

```
\kentenmarkinyoko 2663 \newcommand*\kentenmarkinyoko[1]{%
\kentenmarkintate 2664   \pxrr@k@get@mark\pxrr@k@ymark{#1}%
2665 }
\kentenmarkintate 2666 \newcommand*\kentenmarkintate[1]{%
2667   \pxrr@k@get@mark\pxrr@k@ysmark{#1}%
2668 }
```

```

2669 \newcommand*\kentenmarkintate[1]{%
2670   \pxrr@k@get@mark\pxrr@k@tmark{#1}%
2671 }
2672 \newcommand*\kentensubmarkintate[1]{%
2673   \pxrr@k@get@mark\pxrr@k@tsmark{#1}%
2674 }

```

`\kentenrubycombination` 対応するパラメタを設定する。

```

2675 \chardef\pxrr@k@ruby@combo@ruby=0
2676 \chardef\pxrr@k@ruby@combo@both=1
2677 \newcommand*\kentenrubycombination[1]{%
2678   \pxrr@letcs\pxrr@tempa{\pxrr@k@ruby@combo@#1}%
2679   \ifx\pxrr@tempa\relax
2680     \pxrr@err@invalid@value{#1}%
2681   \else
2682     \let\pxrr@k@ruby@combo\pxrr@tempa
2683   \fi
2684 }

```

## 5.5 圏点文字

`\pxrr@k@declare@mark` `\pxrr@k@declare@mark{<名前>}{<本体>}` : 圏点マーク命令を定義する。

```

2685 \def\pxrr@k@declare@mark#1{%
2686   \global\@namedef{\pxrr@k@mark@#1}%
2687 }

```

`\pxrr@k@let@mark` `\pxrr@k@declare@mark{<名前>}\CS` : 圏点マーク命令を `\let` で定義する。

```

2688 \def\pxrr@k@let@mark#1{%
2689   \global\pxrr@cslet{\pxrr@k@mark@#1}%
2690 }

```

`\pxrr@k@get@mark` `\pxrr@k@get@mark\CS{<名前または定義本体>}` : 指定の圏点マーク命令を `\CS` に代入する。第 2 引数の先頭トークンが ASCII 英字の場合は名前と見なし、それ以外は定義本体のコードと見なす。

```

2691 \def\pxrr@k@get@mark#1#2{%
2692   \futurelet\pxrr@token\pxrr@k@get@mark@a#2\pxrr@nil#1%
2693 }
2694 \def\pxrr@k@get@mark@a{%
2695   \pxrr@cond@ifcat A\noexpand\pxrr@token\fi{%
2696     \pxrr@k@get@mark@c
2697   }{%else
2698     \pxrr@k@get@mark@b
2699   }%
2700 }
2701 \def\pxrr@k@get@mark@b#1\pxrr@nil#2{%
2702   \def#2{#1}%
2703 }

```

```

2704 \def\pxrr@k@get@mark@c#1#2\pxrr@nil#3{%
2705   \ifnum'#1<128
2706     \pxrr@letcs\pxrr@tempa{pxrr@k@mark@c#1#2}%
2707     \ifx\pxrr@tempa\relax
2708       \pxrr@warn@na@kmark{#1#2}%
2709     \else
2710       \let#3\pxrr@tempa
2711     \fi
2712 \else
2713   \pxrr@k@get@mark@b#1#2\pxrr@nil#3%
2714 \fi
2715 }

```

`\pxrr@k@declare@mark@char` `\pxrr@k@declare@mark@char\CS{⟨二重コード⟩}`: 指定のコード値の文字の(和文)chardefを`\CS`に代入する。ただし`pTeX`でJISに無い文字(便宜的に和文空白のJISコード値2121で表す)の場合は代わりに`\pxrr@k@char@UTF`を利用する。

```

2716 \def\pxrr@k@declare@mark@char#1#2{%
2717   \pxrr@k@declare@mark@char@a{#1}#2\pxrr@end
2718 }
2719 \def\pxrr@k@declare@mark@char@a#1#2:#3\pxrr@end{%
2720   \pxrr@jchardef\pxrr@tempa\pxrr@jc{#2:#3}%
2721   \ifnum\pxrr@tempa=\pxrr@zspace

```

エンジンが`pTeX`でかつJISに無い文字である場合。

```

2722     \pxrr@k@declare@mark{#1}{\pxrr@k@char@UTF{#1}{#3}}%
2723   \else
2724     \pxrr@k@let@mark{#1}\pxrr@tempa
2725   \fi
2726 }

```

`\pxrr@k@char@UTF` `\pxrr@k@char@UTF{⟨名前⟩}{⟨Unicode値⟩}`: `\UTF{⟨Unicode値⟩}`を実行するが、`\UTF`が利用不可の場合は、(最初の1回だけ)警告した上で何も出力しない。

```

2727 \def\pxrr@k@char@UTF#1#2{%
2728   \pxrr@decide@avail@UTF
2729   \ifpxrr@avail@UTF
2730     \pxrr@k@declare@mark{#1}{\UTF{#2}}%
2731     \UTF{#2}%
2732   \else
2733     \pxrr@k@let@mark{#1}\@empty
2734     \pxrr@warn@na@kmark{#1}%
2735   \fi
2736 }

```

標準サポートの圏点マークの定義。

```

2737 \pxrr@k@declare@mark@char{bullet} {2121:2022}
2738 \pxrr@k@declare@mark@char{triangle}{2225:25B2}
2739 \pxrr@k@declare@mark@char{Triangle}{2224:25B3}
2740 \pxrr@k@declare@mark@char{fisheye} {2121:25C9}
2741 \pxrr@k@declare@mark@char{Circle} {217B:25CB}

```

```

2742 \pxrr@k@declare@mark@char{bullseye}{217D:25CE}
2743 \pxrr@k@declare@mark@char{circle} {217C:25CF}
2744 \pxrr@k@declare@mark@char{Bullet} {2121:25E6}
2745 \pxrr@k@declare@mark@char{sesame} {2121:FE45}
2746 \pxrr@k@declare@mark@char{Sesame} {2121:FE46}
2747 \pxrr@jchardef\pxrr@ja@dot=\pxrr@jc{2126:30FB}
2748 \pxrr@jchardef\pxrr@ja@comma=\pxrr@jc{2122:3001}
2749 \pxrr@k@declare@mark{bullet*}{%
2750 \pxrr@dima=\pxrr@ruby@zw\relax
2751 \hb@xt@\pxrr@dima{%
2752 \kern-.5\pxrr@dima
2753 \pxrr@if@in@tate{|\lower.38\pxrr@dima}%
2754 \hb@xt@2\pxrr@dima{%
2755 \pxrr@dima=\f@size\p@
2756 \fontsize{2\pxrr@dima}{\z@}\selectfont
2757 \hss
2758 \pxrr@ja@dot
2759 \hss
2760 }%
2761 \hss
2762 }%
2763 }
2764 \pxrr@k@declare@mark{sesame*}{%
2765 \pxrr@dima=\pxrr@ruby@zw\relax
2766 \hb@xt@\pxrr@dima{%
2767 \pxrr@if@in@tate{\kern.1\pxrr@dima}{\kern.05\pxrr@dima}%
2768 \pxrr@if@in@tate{\lower.85\pxrr@dima}{\raise.3\pxrr@dima}%
2769 \hbox{%
2770 \pxrr@dima=\f@size\p@
2771 \fontsize{2.4\pxrr@dima}{\z@}\selectfont
2772 \pxrr@ja@comma
2773 }%
2774 \hss
2775 }%
2776 }

```

## 5.6 圏点オプション解析

`\pxrr@k@parse@option` `\pxrr@k@parse@option{オプション}`: `<オプション>` を解析し、`\pxrr@side` や `\pxrr@k@mark` 等のパラメタを設定する。

```

2777 \def\pxrr@k@parse@option#1{%
2778 \edef\pxrr@tempa{#1}%
2779 \let\pxrr@side\pxrr@k@d@side
2780 \let\pxrr@k@mark\pxrr@k@d@mark
2781 \let\pxrr@k@full\pxrr@k@d@full
2782 \expandafter\pxrr@k@parse@option@loop\pxrr@tempa @\pxrr@end
2783 }
2784 \def\pxrr@k@parse@option@loop#1{%

```

圏点オプションの解析器は“有限状態”を持たないので非常に単純である。

```
2785 \pxrr@letcs\pxrr@tempa{pxrr@k@po@PR@#1}%
2786 \pxrr@cond\ifx\pxrr@tempa\relax\fi{%
2787   \pxrr@fatal@knx@letter{#1}%
2788   \pxrr@k@parse@option@exit
2789 }{%
2790   \pxrr@tempa
2791   \pxrr@k@parse@option@loop
2792 }%
2793 }
2794 \def\pxrr@k@parse@option@exit#1\pxrr@end{%
2795   \ifpxrr@in@setup\else
2796     \pxrr@k@check@option
```

ここで `\pxrr@k@the@mark` を適切に定義する。

```
2797   \pxrr@if@in@tate{%
2798     \ifcase\pxrr@k@mark \let\pxrr@k@the@mark\pxrr@k@tmark
2799     \or \let\pxrr@k@the@mark\pxrr@k@tsmark
2800     \fi
2801   }{%
2802     \ifcase\pxrr@k@mark \let\pxrr@k@the@mark\pxrr@k@ymark
2803     \or \let\pxrr@k@the@mark\pxrr@k@ysmark
2804     \fi
2805   }%
2806 \fi
2807 }
2808 \def\pxrr@k@po@PR@{%
2809   \pxrr@k@parse@option@exit
2810 }
2811 \def\pxrr@k@po@PR@P{%
2812   \chardef\pxrr@side\z@
2813 }
2814 \def\pxrr@k@po@PR@S{%
2815   \chardef\pxrr@side\@ne
2816 }
2817 \def\pxrr@k@po@PR@p{%
2818   \chardef\pxrr@k@mark\z@
2819 }
2820 \def\pxrr@k@po@PR@s{%
2821   \chardef\pxrr@k@mark\@ne
2822 }
2823 \def\pxrr@k@po@PR@F{%
2824   \chardef\pxrr@k@full\z@
2825 }
2826 \def\pxrr@k@po@PR@f{%
2827   \chardef\pxrr@k@full\@ne
2828 }
```

## 5.7 オプション整合性検査

今のところ検査すべき点がない。

```
2829 \def\pxrr@k@check@option{%
2830 }
```

## 5.8 ブロック毎の組版

`\pxrr@k@compose@block` `\pxrr@k@compose@block{<親文字ブロック>}{<圏点の個数>}`: 1つのブロックの組版処理。ボックス `\pxrr@boxb` に圏点1つを組版したものが入っている必要がある。なお、圏点はゼロ幅に潰した形で扱う前提のため、`\pxrr@boxb` の幅はゼロでないといけない。基本的に、ルビ用の `\pxrr@compose@oneside@block` を非常に簡略化した処理になっている。

```
2831 \def\pxrr@k@compose@block#1#2{%
2832   \setbox\pxrr@boxa\pxrr@hbox{#1}%
```

`\pxrr@evenspace@int` を使うために辻褄を合わせる。すなわち、`\copy\pxrr@boxb` を圏点個数分だけ反復したリストを `\pxrr@res` に入れて、“圏点の自然長”に当たる `\pxrr@natwd` をゼロとする。

```
2833   \pxrr@k@make@rep@list{\copy\pxrr@boxb}{#2}%
2834   \let\pxrr@natwd\pxrr@zeropt
2835   \pxrr@evenspace@int\pxrr@locate@inner\pxrr@boxr
2836     \relax{\wd\pxrr@boxa}%
2837   \setbox\z@\hbox{%
2838     \ifnum\pxrr@side=\z@
2839       \raise\pxrr@ruby@raise\box\pxrr@boxr
2840     \else
2841       \lower\pxrr@ruby@lower\box\pxrr@boxr
2842     \fi
2843   }%
2844   \ht\z@\z@ \dp\z@\z@
2845   \@tempdima\wd\z@
2846   \setbox\pxrr@boxr\hbox{%
2847     \box\z@
2848     \kern-\@tempdima
2849     \box\pxrr@boxa
2850   }%
2851 }
```

`\pxrr@k@make@rep@list` `\pxrr@k@make@rep@list{<要素>}{<回数>}`: 要素を指定の回数だけ反復したリストを `\pxrr@res` に代入する。

```
2852 \def\pxrr@k@make@rep@list#1#2{%
2853   \def\pxrr@res{\pxrr@pre{#1}}%
2854   \pxrr@cntr=#2\relax
2855   \ifnum\pxrr@cntr>\@ne
```

```

2856 \@tempcnta\pxrr@cntr \advance\@tempcnta\m@ne
2857 \@whilenum{\@tempcnta>\z}\do{%
2858 \pxrr@appto\pxrr@res{\pxrr@inter{#1}}%
2859 \advance\@tempcnta\m@ne
2860 }%
2861 \fi
2862 \pxrr@appto\pxrr@res{\pxrr@post}%
2863 }

```

## 5.9 圏点項目

- 圏点項目リスト： テキストを `\pxrr@k@decompose` で分解した結果のリスト。
- 圏点項目： 圏点リストに含まれる `\pxrr@entry[XXX]{...}` という形式のこと。圏点項目は直接に実行する（出力する）ことができる。
- 圏点ブロック： 一つの《文字》に圏点を付加して出力したもの。
- 参照文字コード： 圏点項目の出力の前後の禁則ペナルティの扱いにおいて、「ある文字と同等」と扱う場合の、その文字の文字コード。

※現状では、まず `\pxrr@k@nten@entry@XXX` というマクロを定義して圏点命令の実行時にそれを `\pxrr@entry@XXX` にコピーする、という手続きを採っている。（ただそうする意味が全く無い気がする。）

```

\ifpxrr@k@first@entry  先頭の項目であるか。
2864 \newif\ifpxrr@k@first@entry

```

```

\ifpxrr@k@last@entry  末尾の項目であるか。
2865 \newif\ifpxrr@k@last@entry

```

```

\ifpxrr@k@prev@is@block  直前の項目の結果が圏点ブロックであったか。
2866 \newif\ifpxrr@k@prev@is@block

```

```

\pxrr@k@accum@res  累積の直接出力。
2867 \let\pxrr@k@accum@res\relax

```

以下の 3 つの変数は“項目の下請けマクロ”が値を返すべきもの。これらに加えて、`\pxrr@res` と `\pxrr@boxr` の一方に（組版の）結果を返す必要がある。

```

\pxrr@k@prebreakpenalty  圏点項目の前禁則ペナルティ。
2868 \mathchardef\pxrr@k@prebreakpenalty\z@

```

```

\pxrr@k@postbreakpenalty  圏点項目の後禁則ペナルティ。
2869 \mathchardef\pxrr@k@postbreakpenalty\z@

```

```

\pxrr@k@entry@res@type  項目の出力のタイプ。0=直接出力；1=ボックス出力；2=圏点ブロック。0の場合、出力は
\pxrr@res にあり、それ以外は、出力は \pxrr@boxr にある。
2870 \chardef\pxrr@k@entry@res@type\z@

```

`\pxrr@k@list@pre` 圏点項目リストの出力の開始時に行う処理。

```
2871 \def\pxrr@k@list@pre{%
2872   \pxrr@k@first@entrytrue
2873   \pxrr@k@last@entryfalse
2874   \pxrr@k@prev@is@blockfalse
2875   \let\pxrr@k@accum@res\@empty
2876   \chardef\pxrr@k@block@seq@state\z@
2877 }
```

`\pxrr@k@entry@with` 補助マクロ。各種圏点項目の共通の処理を行う。

※ #1 は各圏点項目命令の下請けのマクロで、#2 は圏点項目の引数。

```
2878 \def\pxrr@k@entry@with#1#2{%
2879   \pxrr@if@last{%
2880     \pxrr@k@last@entrytrue
2881     \pxrr@k@entry@with@a#1{#2}%
2882   }{%
2883     \pxrr@k@entry@with@a#1{#2}%
2884   }%
2885 }
2886 \def\pxrr@k@entry@with@a#1#2{%
2887   \mathchardef\pxrr@k@prebreakpenalty\z@
2888   \mathchardef\pxrr@k@postbreakpenalty\z@
```

下請けマクロを実行して結果を得る。

```
2889   #1{#2}%
2890   \%typeout{%
2891   %first=\meaning\ifpxrr@k@first@entry^^J%
2892   %last=\meaning\ifpxrr@k@last@entry^^J%
2893   %prev=\meaning\ifpxrr@k@prev@is@block^^J%
2894   %res=\meaning\pxrr@res^^J%
2895   %type=\meaning\pxrr@k@entry@res@type^^J%
2896   %prepen=\the\pxrr@k@prebreakpenalty^^J%
2897   %postpen=\the\pxrr@k@postbreakpenalty}%
```

累積直接出力の処理。

```
2898   \ifnum\pxrr@k@entry@res@type=\z@
2899     \expandafter\pxrr@appto\expandafter\pxrr@k@accum@res
2900     \expandafter{\pxrr@res}%
2901   \else
2902     \pxrr@k@accum@res
2903     \let\pxrr@k@accum@res\@empty
2904   \fi
```

前禁則ペナルティを入れる。

```
2905   \ifnum\pxrr@k@prebreakpenalty>\z@
2906     \@tempcntb\lastpenalty \unpenalty
2907     \advance\@tempcntb\pxrr@k@prebreakpenalty
2908     \penalty\@tempcntb
2909   \fi
```

圏点ブロックが連続する場合は和文間空白を入れる。

```
2910 \ifnum\pxrr@k@entry@res@type=\tw@
2911   \ifpxrr@k@prev@is@block
2912     \pxrr@inter@mono
2913   \fi
2914   \pxrr@k@prev@is@blocktrue
2915 \else
2916   \pxrr@k@prev@is@blockfalse
2917 \fi
```

ボックスの結果を実際に出力する。

```
2918 \ifnum\pxrr@k@entry@res@type>\z@
2919   \unhbox\pxrr@boxr
2920 \fi
```

後禁則ペナルティを入れる。

```
2921 \ifnum\pxrr@k@postbreakpenalty>\z@
2922   \penalty\pxrr@k@postbreakpenalty
2923 \fi
```

次の項目に進む。

```
2924 \pxrr@k@first@entryfalse
2925 }
```

`\pxrr@k@list@post` 圏点項目リストの出力の最後に行う処理。

```
2926 \def\pxrr@k@list@post{%
2927   \pxrr@k@accum@res
2928   \let\pxrr@k@accum@res\@empty
2929 }
```

`\pxrr@k@kenten@entry` 一般の《文字》を表す圏点項目 `\pxrr@entry{⟨文字⟩}` の処理。圏点を1つ付けて出力する。

```
2930 \def\pxrr@k@kenten@entry{%
2931   \pxrr@k@entry@with\pxrr@k@kenten@entry@
2932 }
2933 \def\pxrr@k@kenten@entry@#1{%
2934   \pxrr@k@check@char{#1}%
2935   \ifpxrr@ok
2936     \pxrr@k@compose@block{#1}\@ne
2937     \chardef\pxrr@k@entry@res@type=\tw@
2938   \else
2939     \def\pxrr@res{#1}%
2940     \chardef\pxrr@k@entry@res@type=\z@
2941   \fi
2942 }
```

`\pxrr@k@kenten@entry@kspan` `\kspan` 命令を表す圏点項目 `\pxrr@entry@kspan{⟨kspan{⟨テキスト⟩}}}` の処理。テキストの幅が“およそ  $n$  全角”である場合に、 $n$  個の圏点をルビ均等割りで配置して出力する。

```
2943 \def\pxrr@k@kenten@entry@kspan{%
2944   \pxrr@k@entry@with\pxrr@k@kenten@entry@kspan@
```

```

2945 }
2946 \def\pxrr@kenten@entry@kspan@#1{%
2947   \pxrr@kenten@entry@kspan@a#1%
2948 }
2949 \def\pxrr@kenten@entry@kspan@a#1{%
    \kspan (= #1) が * 付かを調べる。
2950   \@ifstar{%
2951     \@testopt\pxrr@kenten@entry@kspan@c{%}
2952   }{%
2953     \@testopt\pxrr@kenten@entry@kspan@b{%}
2954   }%
2955 }
2956 \def\pxrr@kenten@entry@kspan@b[#1]#2{%
    ( $n - 1/4$ )zw 以上 ( $n + 3/4$ )zw 未満の時に “およそ  $n$  全角” と見なす。
2957   \setbox\z@\pxrr@hbox{#2}%
2958   \@tempdima\pxrr@body@zw\relax
2959   \@tempdimb\wd\z@ \advance\@tempdimb.25\@tempdima
2960   \divide\@tempdimb\@tempdima
2961   \edef\pxrr@kenten@entry@tempa{\number\@tempdimb}%
2962   \pxrr@k@compose@block{#2}\pxrr@kenten@entry@tempa
2963   \chardef\pxrr@k@entry@res@type=\tw@
2964 }
2965 \def\pxrr@kenten@entry@kspan@c[#1]#2{%
    \kspan* となっている場合。この時は圈点を付加せず直接出力する。
2966   \def\pxrr@res{#2}%
2967   \chardef\pxrr@k@entry@res@type=\z@
2968 }

```

`\pxrr@kenten@entry@kenten` ネストした `\kenten` 命令の圈点項目。単純にその `\kenten` を実行したものを出力とする。  
すなわち、内側の圈点の設定のみが生きる。

```

2969 \def\pxrr@kenten@entry@kenten{%
2970   \pxrr@k@entry@with\pxrr@kenten@entry@kenten@
2971 }
2972 \def\pxrr@kenten@entry@kenten@#1{%
    この場合は圈点ブロックとは見なさないことに注意。
2973   \setbox\pxrr@boxr\hbox{#1}%
2974   \chardef\pxrr@k@entry@res@type=\@ne
2975 }

```

`\pxrr@kenten@entry@ruby` ルビ命令の圈点項目。

```

2976 \def\pxrr@kenten@entry@ruby{%
2977   \pxrr@k@entry@with\pxrr@kenten@entry@ruby@
2978 }
2979 \def\pxrr@kenten@entry@ruby@#1{%
2980   \pxrr@apply@combotrue
2981   \setbox\pxrr@boxr\hbox{#1}%

```

```
2982 \chardef\pxrr@k@entry@res@type=\@ne
2983 }
```

### 5.9.1 \kspan 命令

`\kspan` テキストの幅に相応した個数の圏点を付ける命令。`\kenten` の引数のテキストの中で使う。`\kenten` の外で使われた場合は単純に引数を出力するだけ。  
 ※ 処理の都合上、オプション引数を持たせているが、実際には（現在は）これは使われない。

```
2984 \newcommand*\kspan{%
2985   \@ifstar{%
2986     \@testopt\pxrr@kspan@a{}}%
2987   }{%
2988     \@testopt\pxrr@kspan@a{}}%
2989   }%
2990 }
2991 \pxrr@add@protect\kspan
2992 \def\pxrr@kspan@a[#1]#2{%
2993   \begingroup
2994     #2%
2995   \endgroup
2996 }
```

## 5.10 自動抑止の検査

`\pxrr@k@check@char` 通常項目 (`\pxrr@entry`) の引数を検査して、圏点を付加すべきか否かをスイッチ `pxrr@ok` に返す。また、項目の前禁則・後禁則ペナルティを設定する。  
 引数が（単一の）通常文字である時はその文字、引数がグループの場合は和文空白の内部文字コードを `\pxrr@cntr` に返す（禁則ペナルティを後で見られるように）。

```
2997 \def\pxrr@k@check@char#1{%
2998   \futurelet\pxrr@token\pxrr@k@check@char@a#1\pxrr@end
2999 }
3000 \def\pxrr@k@check@char@a#1\pxrr@end{%
3001   \pxrr@cond\ifx\pxrr@token\bgroup\fi%
```

グループには圏点を付ける。

```
3002   \pxrr@oktrue
3003 }{\pxrr@cond\ifx\pxrr@token\@sptoken\fi%
```

欧文空白には圏点を付けない。

```
3004   \pxrr@okfalse
3005 }{%
3006   \pxrr@check@char\pxrr@token
3007   \ifcase\pxrr@cntr
```

通常文字でないので圏点を付けない。

```
3008   \pxrr@okfalse
3009   \or
```

欧文の通常文字。圏点を付ける。

```
3010 \pxrr@oktrue
3011 \chardef\pxrr@check@char@temp\z@
3012 \or
```

和文の通常文字。圏点を付ける。

```
3013 \pxrr@oktrue
3014 \chardef\pxrr@check@char@temp\@ne
3015 \fi
```

約物の圏点付加が無効の場合は、引数の文字が約物であるか検査し、そうである場合は圏点を付けない。

```
3016 \ifnum\pxrr@k@full=\z@\ifpxrr@ok
3017 \pxrr@check@punct@char{'#1}\pxrr@check@char@temp
3018 \ifpxrr@ok \pxrr@okfalse
3019 \else \pxrr@oktrue
3020 \fi
3021 \fi\fi
3022 \ifpxrr@ok
3023 \pxrr@get@prebreakpenalty\@tempcnta{'#1}%
3024 \mathchardef\pxrr@k@prebreakpenalty\@tempcnta
3025 \pxrr@get@postbreakpenalty\@tempcnta{'#1}%
3026 \mathchardef\pxrr@k@postbreakpenalty\@tempcnta
3027 \fi
3028 }}%
3029 }
```

## 5.11 メインです

### 5.11.1 エントリーポイント

`\kenten` 圏点の公開命令。`\jkenten` を頑強な命令として定義した上で、`\kenten` はそれに展開されるマクロに（未定義ならば）定義する。

```
3030 \AtBeginDocument{%
3031 \providecommand*\kenten{\jkenten}%
3032 }
3033 \newcommand*\jkenten{%
3034 \pxrr@k@prologue
3035 \pxrr@kenten
3036 }
3037 \pxrr@add@protect\jkenten
```

`\pxrr@kenten` オプションの処理を行う。

```
3038 \def\pxrr@kenten{%
3039 \@testopt\pxrr@kenten@a{%}
3040 }
3041 \def\pxrr@kenten@a[#1]{%
3042 \def\pxrr@option{#1}%
3043 \ifpxrr@safe@mode
```

安全モードでは圏点機能は無効なので、フォールバックとして引数のテキストをそのまま出力する。

```
3044 \expandafter\@firstofone
3045 \else
3046 \expandafter\pxrr@kenten@proc
3047 \fi
3048 }
```

\pxrr@k@bind@param “呼出時変数” へのコピーを行う。

```
3049 \def\pxrr@k@bind@param{%
3050 \let\pxrr@c@ruby@font\pxrr@k@ruby@font
3051 \let\pxrr@c@size@ratio\pxrr@k@size@ratio
3052 \let\pxrr@c@inter@gap\pxrr@k@inter@gap
3053 }
```

\pxrr@kenten@proc \pxrr@kenten@proc{(親文字列)}：これが手続の本体となる。

```
3054 \def\pxrr@kenten@proc#1{%
3055 \pxrr@prepare@fallback{#1}%
3056 \pxrr@k@bind@param
3057 \pxrr@assign@fsize
3058 \pxrr@k@parse@option\pxrr@option
3059 \pxrr@if@alive{%
3060 \pxrr@k@decompose{#1}%
3061 \let\pxrr@body@list\pxrr@res
3062 \pxrr@kenten@main
3063 }%
3064 \pxrr@kenten@exit
3065 }
```

### 5.11.2 組版処理

\pxrr@kenten@main 圏点の組版処理。

```
3066 \def\pxrr@kenten@main{%
3067 \setbox\pxrr@boxb\pxrr@hbox@to\z@f%
3068 \pxrr@use@ruby@font
3069 \hss\pxrr@k@the@mark\hss
3070 }%
3071 \let\pxrr@entry\pxrr@kenten@entry
3072 \let\pxrr@entry@kspan\pxrr@kenten@entry@kspan
3073 \let\pxrr@entry@ruby\pxrr@kenten@entry@ruby
3074 \let\pxrr@entry@kenten\pxrr@kenten@entry@kenten
3075 \let\pxrr@post\pxrr@k@list@post
3076 \pxrr@k@list@pre
3077 \pxrr@body@list
3078 }
```

### 5.11.3 前処理

`\pxrr@jprologue` 圏点用の開始処理。

```
3079 \def\pxrr@k@prologue{%
3080   \ifpxrr@k@ghost
3081     \pxrr@jghost@char
3082     \pxrr@inhibitglue
3083   \fi
3084   \begingroup
3085     \ifpxrr@k@ghost
3086       \setbox\pxrr@boxa\hbox{\pxrr@jghost@char}%
3087       \kern-\wd\pxrr@boxa
3088     \fi
3089 }
```

### 5.11.4 後処理

`\pxrr@kenten@exit` 出力を終えて、最後に呼ばれるマクロ。

```
3090 \def\pxrr@kenten@exit{%
3091   \ifpxrr@fatal@error
3092     \pxrr@fallback
3093   \fi
3094   \pxrr@k@epilogue
3095 }
```

`\pxrr@jepilogue` 終了処理。

```
3096 \def\pxrr@k@epilogue{%
3097   \ifpxrr@k@ghost
3098     \setbox\pxrr@boxa\hbox{\pxrr@jghost@char}%
3099     \kern-\wd\pxrr@boxa
3100   \fi
3101   \endgroup
3102   \ifpxrr@k@ghost
3103     \pxrr@inhibitglue
3104     \pxrr@jghost@char
3105   \fi
3106 }
```

## 5.12 デバッグ用出力

```
3107 \def\pxrr@debug@show@kenten@input{%
3108   \typeout{%
3109     pxrr@k@the@mark=\meaning\pxrr@k@the@mark^^J%
3110     pxrr@side=\meaning\pxrr@side^^J%
3111     pxrr@body@list=\meaning\pxrr@body@list^^J%
3112   }%
3113 }
```

## 6 実装（圏点ルビ同時付加）

コンボ！

### 6.1 呼出時パラメタ

```
\ifpxrr@apply@combo 直後に実行するルビ命令について同時付加を行うか。スイッチ。
3114 \newif\ifpxrr@apply@combo

\ifpxrr@combo 現在実行中のルビ命令について同時付加を行うか。スイッチ。
3115 \newif\ifpxrr@combo

\pxrr@ck@ruby@font 同時付加時の圏点側の呼出時パラメタの値。
\pxrr@ck@size@ratio 3116 \let\pxrr@ck@ruby@font\relax
\pxrr@ck@inter@gap 3117 \let\pxrr@ck@size@ratio\relax
3118 \let\pxrr@ck@inter@gap\relax
\pxrr@ck@ruby@inter@gap 3119 \let\pxrr@ck@ruby@inter@gap\relax
\pxrr@ck@side 3120 \let\pxrr@ck@side\relax
\pxrr@ck@the@mark 3121 \let\pxrr@ck@the@mark\relax
3122 \let\pxrr@ck@ruby@combo\relax
\pxrr@ck@ruby@combo

\ifpxrr@ck@kenten@head 当該のルビ命令が、圏点命令の引数の先頭にあるか。
3123 \newif\ifpxrr@ck@kenten@head

\ifpxrr@ck@kenten@end 当該のルビ命令が、圏点命令の引数の先頭にあるか。
3124 \newif\ifpxrr@ck@kenten@end

\pxrr@ck@bind@param “呼出時変数” へのコピーを行う。
3125 \def\pxrr@ck@bind@param{%
3126 \let\pxrr@ck@ruby@font\pxrr@c@ruby@font
3127 \let\pxrr@ck@size@ratio\pxrr@c@size@ratio
3128 \let\pxrr@ck@inter@gap\pxrr@c@inter@gap
3129 \let\pxrr@ck@ruby@inter@gap\pxrr@k@ruby@inter@gap
3130 \let\pxrr@ck@side\pxrr@side
3131 \let\pxrr@ck@the@mark\pxrr@k@the@mark
3132 \let\pxrr@ck@ruby@combo\pxrr@k@ruby@combo
3133 \pxrr@csletcs{ifpxrr@ck@kenten@head}{ifpxrr@k@first@entry}%
3134 \pxrr@csletcs{ifpxrr@ck@kenten@end}{ifpxrr@k@last@entry}%
3135 }
```

### 6.2 その他の変数

```
\pxrr@ck@zw 圏点の全角幅。
3136 \let\pxrr@ck@zw\relax

\pxrr@ck@raise@P ルビ側が P である場合の、圏点の垂直方向の移動量。
※ 圏点側が S である場合は負値になる。
3137 \let\pxrr@ck@raise@P\relax
```

`\pxrr@ck@raise@S` ルビ側が S である場合の、圏点の垂直方向の移動量。

```
3138 \let\pxrr@ck@raise@S\relax
```

`\pxrr@ck@raise@t` ルビ側が両側ルビである場合の、圏点の垂直方向の移動量。

```
3139 \let\pxrr@ck@raise@t\relax
```

### 6.3 オプション整合性検査

`\pxrr@ck@check@option` 同時付加のための呼出時パラメタの調整。

```
3140 \def\pxrr@ck@check@option{%
3141   \ifpxrr@ck@kenten@head
3142     \let\pxrr@bintr@\@empty
3143     \let\pxrr@bscomp=. \relax
3144     \pxrr@bnobrtrue
3145   \fi
3146   \ifpxrr@ck@kenten@end
3147     \let\pxrr@aintr@\@empty
3148     \let\pxrr@ascomp=. \relax
3149     \pxrr@anobrtrue
3150   \fi
3151 }
```

### 6.4 フォントサイズ

`\pxrr@ck@assign@fsize` フォントに関連する設定。

```
3152 \def\pxrr@ck@assign@fsize{%
  \pxrr@ck@zw の値を求める。
3153   \begingroup
3154     \@tempdima=\f@size\p@
3155     \@tempdima\pxrr@ck@size@ratio\@tempdima
3156     \edef\pxrr@ruby@fsize{\the\@tempdima}%
3157     \let\pxrr@c@ruby@font\pxrr@ck@ruby@font
3158     \pxrr@use@ruby@font
3159     \pxrr@get@zwidth\pxrr@ck@zw
3160     \global\let\pxrr@gtempa\pxrr@ck@zw
3161   \endgroup
3162   \let\pxrr@ck@zw\pxrr@gtempa

  \pxrr@ck@raise@P、\pxrr@ck@raise@S の値を計算する。
3163   \ifcase\pxrr@ck@side
    圏点側が P の場合。
3164     \@tempdimc\pxrr@ck@zw
3165     \advance\@tempdimc-\pxrr@htratio\@tempdimc
3166     \@tempdima\pxrr@ruby@raise\relax
3167     \@tempdimb\pxrr@ruby@zw\relax
```

```

3168 \advance\@tempdima\pxrr@htratio\@tempdimb
3169 \@tempdimb\pxrr@body@zw\relax
3170 \advance\@tempdima\pxrr@ck@ruby@inter@gap\@tempdimb
3171 \advance\@tempdima\@tempdimc
3172 \edef\pxrr@ck@raise@P{\the\@tempdima}%
3173 \@tempdima\pxrr@body@zw\relax
3174 \@tempdima\pxrr@htratio\@tempdima
3175 \@tempdimb\pxrr@body@zw\relax
3176 \advance\@tempdima\pxrr@ck@inter@gap\@tempdimb
3177 \advance\@tempdima\@tempdimc
3178 \edef\pxrr@ck@raise@S{\the\@tempdima}%
3179 \let\pxrr@ck@raise@t\pxrr@ck@raise@P
3180 \or

```

圏点側が S の場合。

```

3181 \@tempdimc\pxrr@ck@zw
3182 \@tempdimc\pxrr@htratio\@tempdimc
3183 \@tempdima-\pxrr@ruby@lower\relax
3184 \@tempdimb\pxrr@ruby@zw\relax
3185 \advance\@tempdimb-\pxrr@htratio\@tempdimb
3186 \advance\@tempdima-\@tempdimb
3187 \@tempdimb\pxrr@body@zw\relax
3188 \advance\@tempdima-\pxrr@ck@ruby@inter@gap\@tempdimb
3189 \advance\@tempdima-\@tempdimc
3190 \edef\pxrr@ck@raise@S{\the\@tempdima}%
3191 \@tempdima-\pxrr@body@zw\relax
3192 \advance\@tempdima-\pxrr@htratio\@tempdima
3193 \@tempdimb\pxrr@body@zw\relax
3194 \advance\@tempdima-\pxrr@ck@inter@gap\@tempdimb
3195 \advance\@tempdima-\@tempdimc
3196 \edef\pxrr@ck@raise@P{\the\@tempdima}%
3197 \let\pxrr@ck@raise@t\pxrr@ck@raise@S
3198 \fi
3199 }

```

## 6.5 ブロック毎の組版

`\pxrr@ck@body@natwd` 親文字列の自然長。

```
3200 \let\pxrr@ck@body@natwd\relax
```

`\pxrr@ck@locate` 圏点列のパターン指定。

```
3201 \let\pxrr@ck@locate\relax
```

`\pxrr@ck@kenten@list` 圏点列のリスト。

```
3202 \let\pxrr@ck@kenten@list\relax
```

`\pxrr@ck@compose` #1 に親文字テキスト、`\pxrr@ck@body@natwd` に親文字の自然長、ボックス 0 にルビ出力、`\pxrr@boxa` に親文字出力、`\pxrr@ck@locate` にパターンが入っている前提で、ボックス

0 に圏点を追加する。

```
3203 \def\pxrr@ck@compose#1{%
```

圏点を組んだボックスを作る。

```
3204 \setbox\tw@\pxrr@hbox@to\z@{%
3205 \@tempdima=\f@size\p@
3206 \@tempdima\pxrr@ck@size@ratio\@tempdima
3207 \edef\pxrr@ruby@fsize{\the\@tempdima}%
3208 \let\pxrr@c@ruby@font\pxrr@ck@ruby@font
3209 \pxrr@use@ruby@font
3210 \hss\pxrr@ck@the@mark\hss
3211 }%
```

親文字テキストを分解した後、リスト `\pxrr@res` を圏点のリストに置き換える。

```
3212 \pxrr@save@listproc
3213 \pxrr@decompose{#1}%
3214 \def\pxrr@pre{%
3215 \let\pxrr@res\@empty
3216 \pxrr@ck@compose@entry\pxrr@pre
3217 }%
3218 \def\pxrr@inter{%
3219 \pxrr@ck@compose@entry\pxrr@inter
3220 }%
3221 \def\pxrr@post{%
3222 \pxrr@appto\pxrr@res{\pxrr@post}%
3223 }%
3224 \pxrr@res
3225 \pxrr@restore@listproc
3226 \let\pxrr@natwd\pxrr@ck@body@natwd
```

圏点リストを均等配置する。

```
3227 \pxrr@evenspace@int\pxrr@ck@locate\pxrr@boxb\relax
3228 {\wd\pxrr@boxa}%
```

合成処理。

```
3229 \setbox\z@\hbox{%
3230 \unhcopy\z@
3231 \kern-\wd\z@
3232 \ifcase\pxrr@side
3233 \raise\pxrr@ck@raise@P
3234 \or
3235 \raise\pxrr@ck@raise@S
3236 \or
3237 \raise\pxrr@ck@raise@t
3238 \fi
3239 \hb@xt@\wd\pxrr@boxa{\hss\copy\pxrr@boxb\hss}%
3240 }%
3241 }
3242 \def\pxrr@ck@compose@entry#1#2{%
3243 \setbox\pxrr@boxb\pxrr@hbox{#2}%
```

```

3244 \edef\pxrr@tempa{%
3245   \noexpand\pxrr@appto\noexpand\pxrr@res{\noexpand#1{%
3246     \hb@xt@\the\wd\pxrr@boxb{\hss\copy\tw@\hss}}}%
3247   }\pxrr@tempa
3248 }

```

## 7 実装：hyperref 対策

PDF 文字列中ではルビ命令や圏点命令が“無難な出力”をするようにする。現状では、ルビ・圏点ともに親文字のみを出力することにする。

`\pxrr@dumb@sub` オプション部分を読み飛ばす補助マクロ。

```
3249 \def\pxrr@dumb@sub#1#2#1#1}
```

`\pxrr@dumb@ruby` 無難なルビ命令。

```

3250 \def\pxrr@dumb@ruby{%
3251   \pxrr@dumb@sub\pxrr@dumb@ruby@
3252 }
3253 \def\pxrr@dumb@ruby@#1#2#1#1}

```

`\pxrr@dumb@truby` 無難な両側ルビ命令。

```

3254 \def\pxrr@dumb@truby{%
3255   \pxrr@dumb@sub\pxrr@dumb@truby@
3256 }
3257 \def\pxrr@dumb@truby@#1#2#3#1#1}

```

`\pxrr@dumb@tkenten` 無難な圏点命令。

※ `\kspan` もこの定義を利用する。

```

3258 \def\pxrr@dumb@kenten{%
3259   \pxrr@dumb@sub\pxrr@dumb@kenten@
3260 }
3261 \def\pxrr@dumb@kenten@#1#1#1}

```

hyperref の `\pdfstringdef` 用のフック `\pdfstringdefPreHook` に上書き処理を追記する。

```

3262 \providecommand*\pdfstringdefPreHook{}
3263 \g@addto@macro\pdfstringdefPreHook{%

```

`\ruby` と `\kenten` は「本パッケージの命令であるか」の検査が必要。

```

3264   \ifx\pxrr@cmd@ruby\ruby
3265     \let\ruby\pxrr@dumb@ruby
3266   \fi
3267   \let\jruby\pxrr@dumb@ruby
3268   \let\aruby\pxrr@dumb@ruby
3269   \let\truby\pxrr@dumb@truby
3270   \let\atruby\pxrr@dumb@truby
3271   \ifx\pxrr@cmd@kenten\kenten
3272     \let\kenten\pxrr@dumb@kenten
3273   \fi

```

```
3274 \let\kspan\pxrr@dumb@kenten  
3275 }
```