

# pxrubrica パッケージ

八登 崇之 (Takayuki YATO; aka “ZR”)

v1.3 [2017/04/27]

## 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	34
4.8	ルビオプション解析	37

4.9	オプション整合性検査 . . . . .	43
4.10	フォントサイズ . . . . .	45
4.11	ルビ用均等割り . . . . .	47
4.12	小書き仮名の変換 . . . . .	50
4.13	ブロック毎の組版 . . . . .	52
4.14	命令の頑強化 . . . . .	58
4.15	致命的エラー対策 . . . . .	59
4.16	先読み処理 . . . . .	59
4.17	進入処理 . . . . .	61
4.17.1	前側進入処理 . . . . .	63
4.17.2	後側進入処理 . . . . .	64
4.18	メインです . . . . .	65
4.18.1	エントリーポイント . . . . .	65
4.18.2	入力検査 . . . . .	70
4.18.3	ルビ組版処理 . . . . .	72
4.18.4	前処理 . . . . .	77
4.18.5	後処理 . . . . .	78
4.19	デバッグ用出力 . . . . .	79
5	<b>実装（圏点関連）</b> . . . . .	81
5.1	エラーメッセージ . . . . .	81
5.2	パラメタ . . . . .	81
5.2.1	全般設定 . . . . .	81
5.2.2	呼出時の設定 . . . . .	82
5.3	補助手続 . . . . .	82
5.3.1	\UTF 命令対応 . . . . .	82
5.3.2	リスト分解 . . . . .	83
5.4	パラメタ設定公開命令 . . . . .	85
5.5	圏点文字 . . . . .	86
5.6	圏点オプション解析 . . . . .	88
5.7	オプション整合性検査 . . . . .	90
5.8	ブロック毎の組版 . . . . .	90
5.9	圏点項目 . . . . .	91
5.9.1	\kspan 命令 . . . . .	95
5.10	自動抑止の検査 . . . . .	95
5.11	メインです . . . . .	96
5.11.1	エントリーポイント . . . . .	96
5.11.2	組版処理 . . . . .	97
5.11.3	前処理 . . . . .	98
5.11.4	後処理 . . . . .	98

5.12	デバッグ用出力 . . . . .	98
6	実装（圏点ルビ同時付加）	99
6.1	呼出時パラメタ . . . . .	99
6.2	その他の変数 . . . . .	99
6.3	オプション整合性検査 . . . . .	100
6.4	フォントサイズ . . . . .	100
6.5	ブロック毎の組版 . . . . .	101
7	実装：hyperref 対策	103

## 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 `pLATEX` 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> ( <code>&lt; primary</code> )	上側配置
<code>c</code> ( <code>&lt; center</code> )	中付き	<code>S</code> ( <code>&lt; secondary</code> )	下側配置
<code>h</code> ( <code>&lt; head</code> )	肩付き	<code>e</code> ( <code>&lt; even-space</code> )	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> ( <code>&lt; mono</code> )	モノルビ	<code>f</code> ( <code>&lt; full-size</code> )	小書き文字変換有効
<code>g</code> ( <code>&lt; group</code> )	グループルビ	<code>F</code>	小書き文字変換無効
<code>j</code> ( <code>&lt; jukugo</code> )	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 (あいうえおっやゅょわ、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 $\text{\LaTeX}$  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
- $\backslash\text{jruby}$  [〈オプション〉]{〈親文字〉}{〈ルビ文字〉}
- $\backslash\text{ruby}$  命令の別名。  $\backslash\text{ruby}$  という命令名は他のパッケージとの衝突の可能性があるので、 $\text{\LaTeX}$  文書の本文開始時 ( $\backslash\text{begin}\{\text{document}\}$ ) に未定義である場合にのみ定義される。これに対して  $\backslash\text{jruby}$  は常に定義される。なお、 $\backslash\text{ruby}$  以外の命令 ( $\backslash\text{jruby}$  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。
- $\backslash\text{aruby}$  [〈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。
- － 常にグルーブルビと扱われる。（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。
- $\backslash\text{truby}$  [〈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ブ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>
- $\backslash\text{atruby}$  [〈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  $\backslash\text{truby}$  と同じ。

<sup>\*1</sup> つまり、旧来の使用ではグルーブ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ブルビと扱われる。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、最初のものは 3 文字、後のものは 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{\< (一) \<}

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。

- pTeX 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。

- `\rubysetup{<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。`\rubysetup` でこれらのオプション文字を指定しても無視される。
  - － `\rubysetup` での設定は累積する。例えば、初期状態から、`\rubysetup{hmf}` と `\rubysetup{<->}` 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}`
- `\rubymarginin{<実数>}`  
「大」の進入量（ルビ全角単位）。[既定 = 1]
- `\rubymarginout{<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、強制的にグループリビに切り替わる。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[lj]{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[lj]{\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 になことを表す。pLATEX 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 LATEX 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@movable` 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@movable{%
67   \pxrr@fatal@error{%
68     Novable group ruby is not allowed here%
69   }\pxrr@eh@fatal
70 }

```

`\pxrr@fatal@na@movable` 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@movable{%
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-jp 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」の進用量 (`\rubybigintrusion`/`\rubysmallintrusion`)。実数値マクロ (数  
`\pxrr@small@intr` 字列に展開される)。

```

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}
91 \def\pxrr@sprop@y{2}
\pxrr@sprop@z 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  
`\pxrr@d@aintr` 入無し）に展開されるマクロ。  
 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@bnoobr` 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@anoobr` ※ 既定値設定 (`\rubysetup`) でこれに対応するものはない。



```

122 \newif\ifpxrr@bnoobr \pxrr@bnoobrfalse
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@reversp 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@reversp=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
\pxrr@c@inter@gap 137 \let\pxrr@c@size@ratio\@undefined
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` 「0pt」という文字列。寸法値マクロへの代入に用いる。  
159 `\def\pxrr@zeropt{0pt}`

`\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\@cc@lvi
232     \divide\@tempcnta\@tempdimb
233     \multiply\@tempcnta\@cc@lvi
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⟩}{(X1,Y1)(X2,Y2)⋯(Xn,Yn)}` : 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 :

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

この命令は \pxrr@res を以下の内容に定義する。

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

```

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。

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

```

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}\{#2\}\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}\{#2\}\{#3\}\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 }
```

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

```
501 \@ifpackageloaded{luatexja-core}{%
502   \pxrr@csletcs{ifpxrr@in@luatexja}{iftrue}%
503 }{%
504   \pxrr@csletcs{ifpxrr@in@luatexja}{iffalse}%
505   \AtBeginDocument{%
506     \@ifpackageloaded{luatexja-core}{%
507       \pxrr@warn@load@order{LuaTeX-japan}%
508     }{}%
509   }%
510 }

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

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

```
525 \ifpxrr@in@xetex
526   \pxrr@csletcs{ifpxrr@in@unicode}{iftrue}%
527 \else\ifpxrr@in@luatex
528   \pxrr@csletcs{ifpxrr@in@unicode}{iftrue}%
529 \else\ifpxrr@in@uptex
530   \pxrr@csletcs{ifpxrr@in@unicode}{iftrue}%
531 \else
532   \pxrr@csletcs{ifpxrr@in@unicode}{iffalse}%
533 \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「 $\langle$ JIS コード 16 進 4 桁 $\rangle$ : $\langle$ Unicode 16 進 4 桁 $\rangle$ 」の形式。

```
534 \def\pxrr@jc#1{%
535   \pxrr@jc@a#1\pxrr@nil
```

```

536 }
537 \ifpxrr@in@unicode
538   \def\pxrr@jc@a#1:#2\pxrr@nil{%
539     "#2\space
540   }
541 \else\ifpxrr@in@ptex
542   \def\pxrr@jc@a#1:#2\pxrr@nil{%
543     \jis"#1\space\space
544   }
545 \else
546   \def\pxrr@jc@a#1:#2\pxrr@nil{%
547     ‘?\space
548   }
549 \fi\fi

```

`\pxrr@jchardef` 和文用の `\chardef`。

```

550 \ifpxrr@in@uptex
551   \let\pxrr@jchardef\kchardef
552 \else
553   \let\pxrr@jchardef\chardef
554 \fi

```

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

```

555 \ifpxrr@in@ptex

```

pTeX 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。

```

556   \begingroup \catcode'\|=0
557   \gdef\pxrr@if@in@tate{%
558     \pxrr@cond\iftdir\fi
559   }
560   \endgroup
561 \else\ifpxrr@in@luatexja

```

LuaTeX-jā 利用の場合、`direction` パラメタを利用する。

```

562   \def\pxrr@if@in@tate{%
563     \pxrr@cond\ifnum\ltjgetparameter{direction}=\thr@@\fi
564   }
565 \else

```

それ以外は常に横組と見なす。

```

566   \let\pxrr@if@in@tate\@secondoftwo
567 \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。

```

568 \ifpxrr@in@ptex
569   \def\pxrr@get@jchar@token#1#2{%
570     \begingroup
571       \kansujichar\@ne=#2\relax
572       \xdef\pxrr@x@tempa{\kansuji\@ne}%
573     \endgroup
574     \let#1\pxrr@x@tempa
575   }

```

Unicode 対応 T<sub>E</sub>X の場合。 \lowercase トリックを利用する。

```

576 \else\ifpxrr@in@unicode
577   \def\pxrr@get@jchar@token#1#2{%
578     \begingroup
579       \lccode'\?=#2\relax
580       \lowercase{\xdef\pxrr@x@tempa{?}}%
581     \endgroup
582     \let#1\pxrr@x@tempa
583   }

```

それ以外ではダミー定義。

```

584 \else
585   \def\pxrr@get@jchar@token#1#2{%
586     \def#1{?}%
587   }
588 \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。

```

589 \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。

upT<sub>E</sub>X 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。

```

590 \ifpxrr@in@uptex
591   \kchardef\pxrr@jghost@char="113000

```

LuaT<sub>E</sub>X-j<sub>a</sub> 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。

```

592 \else\ifpxrr@in@luatexja
593   \let\pxrr@jghost@char\pxrr@zspace
594   \def\pxrr@jghost@check{%
595     \begingroup
596 %       \ltjsetParameter{jaxspmode={\pxrr@zspace,3}}%
597 %       \ltjsetParameter{xkanjiskip=\p}%

```

```

598 %      \ltjsetparameter{autoxspacing=false}%
599      \setbox\z@\hbox{\char"3001\char"3000}%
600 %      \ltjsetparameter{autoxspacing=true}%
601      \setbox\tw@\hbox{\char"3001\inhibitglue\char"3000}%
602      \ifdim\wd\tw@=\wd\z@
603          \global\chardef\pxrr@jghost@char@="00A0
604          \gdef\pxrr@jghost@char{\ltjjachar\pxrr@jghost@char@}%
605      \fi
606  \endgroup
607 }
608 \AtBeginDocument{%
609     \pxrr@jghost@check
610 }

```

それ以外の場合は（仕方が無いので）全角空白を用いる。

```

611 \else
612     \let\pxrr@jghost@char\pxrr@zspace
613 \fi\fi

```

`\pxrr@x@K` 適当な漢字（実際は〈一〉）のトークン。

```

614 \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。

```

615 \ifpxrr@in@ptex
616     \def\pxrr@get@iiskip#1{%

```

以下では `\kanjiskip` 挿入が有効であることをチェックしている。

```

617     \pxrr@x@swafalse
618     \begingroup
619         \inhibitxspcode\pxrr@x@K\thr@@
620         \kanjiskip\p@
621         \setbox\z@\hbox{\noautospacing\pxrr@x@K\pxrr@x@K}%
622         \setbox\tw@\hbox{\pxrr@x@K\pxrr@x@K}%
623         \ifdim\wd\tw@>\wd\z@
624             \aftergroup\pxrr@x@swatrue
625         \fi
626     \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。

```

627     \edef#1{%
628         \ifpxrr@x@swa \the\kanjiskip
629         \else \pxrr@zeropt
630         \fi
631     }%
632 }

```

LuaTeX-ja 使用の場合。

```

633 \else\ifpxrr@in@luatexja
634     \def\pxrr@get@iiskip#1{%

```



```

635 \edef#1{%
636 \ifnum\ltjgetparameter{autospacing}=\@ne
637 \ltjgetparameter{kanjiskip}%
638 \else \pxrr@zeropt
639 \fi
640 }%
641 }

```

それ以外の場合はゼロとする。

```

642 \else
643 \def\pxrr@get@iiskip#1{%
644 \let#1\pxrr@zeropt
645 }
646 \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。

```

647 \ifpxrr@in@ptex
648 \def\pxrr@get@iaiskip#1{%
649 \pxrr@x@swafalse
650 \begingroup
651 \inhibitxspcode\pxrr@x@K\thr@@ \xspcode'X=\thr@@
652 \xkanjiskip\p@
653 \setbox\z@\hbox{\noautoxspacing\pxrr@x@K X}%
654 \setbox\tw@\hbox{\pxrr@x@K X}%
655 \ifdim\wd\tw@>\wd\z@
656 \aftergroup\pxrr@x@swatruet
657 \fi
658 \endgroup
659 \edef#1{%
660 \ifpxrr@x@swa \the\xkanjiskip
661 \else \pxrr@zeropt
662 \fi
663 }%
664 }

```

LuaTeX-ja 使用の場合。

```

665 \else\ifpxrr@in@luatexja
666 \def\pxrr@get@iaiskip#1{%
667 \edef#1{%
668 \ifnum\ltjgetparameter{autoxspacing}=\@ne
669 \ltjgetparameter{xkanjiskip}%
670 \else \pxrr@zeropt
671 \fi
672 }%
673 }

```

それ以外の場合は実際の組版結果から判断する。

```

674 \else
675 \def\pxrr@get@iaiskip#1{%

```

```

676 \begingroup
677 \setbox\z@\hbox{M\pxrr@x@K}%
678 \setbox\tw@\hbox{M\vrule\@width\z@\relax\pxrr@x@K}%
679 \@tempdima\wd\z@ \advance\@tempdima-\wd\tw@
680 \@tempdimb\@tempdima \divide\@tempdimb\thr@@
681 \xdef\pxrr@x@gtempa{\the\@tempdima\space minus \the\@tempdimb}%
682 \endgroup
683 \let#1=\pxrr@x@gtempa
684 }%
685 \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。

```

686 \ifpxrr@in@ptex
687 \def\pxrr@get@zwidth#1{%
688 \@tempdima=1zw\relax
689 \edef#1{\the\@tempdima}%
690 }

```

\zw が定義されている場合は 1\zw とする。

```

691 \else\if\ifx\zw\@undefined T\else F\fi F% if defined
692 \def\pxrr@get@zwidth#1{%
693 \@tempdima=1\zw\relax
694 \edef#1{\the\@tempdima}%
695 }

```

\jsZw が定義されている場合は 1\jsZw とする。

```

696 \else\if\ifx\jsZw\@undefined T\else F\fi F% if defined
697 \def\pxrr@get@zwidth#1{%
698 \@tempdima=1\jsZw\relax
699 \edef#1{\the\@tempdima}%
700 }

```

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。

```

701 \else\ifnum\pxrr@x@K>\@cclv
702 \def\pxrr@get@zwidth#1{%
703 \setbox\tw@\hbox{\pxrr@x@K}%
704 \@tempdima\wd\tw@
705 \ifdim\@tempdima>\z@\else \@tempdima\f@size\p@ \fi
706 \edef#1{\the\@tempdima}%
707 }

```

それ以外の場合は要求サイズと等しいとする。

```

708 \else
709 \def\pxrr@get@zwidth#1{%
710 \@tempdima\f@size\p@\relax
711 \edef#1{\the\@tempdima}%
712 }
713 \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。

```
714 \ifpxrr@in@ptex
715   \def\pxrr@get@prebreakpenalty#1#2{%
716     #1=\prebreakpenalty#2\relax
717   }
```

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

```
718 \else\ifpxrr@in@luatexja
719   \def\pxrr@get@prebreakpenalty#1#2{%
720     #1=\ltjgetparameter{prebreakpenalty}{#2}\relax
721   }
```

それ以外の場合はゼロとして扱う。

```
722 \else
723   \def\pxrr@get@prebreakpenalty#1#2{%
724     #1=\z@
725   }
726 \fi\fi
```

`\pxrr@get@postbreakpenalty` `\pxrr@get@postbreakpenalty\CS{<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。

```
727 \ifpxrr@in@ptex
728   \def\pxrr@get@postbreakpenalty#1#2{%
729     #1=\postbreakpenalty#2\relax
730   }
```

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

```
731 \else\ifpxrr@in@luatexja
732   \def\pxrr@get@postbreakpenalty#1#2{%
733     #1=\ltjgetparameter{postbreakpenalty}{#2}\relax
734   }
```

それ以外の場合はゼロとして扱う。

```
735 \else
736   \def\pxrr@get@postbreakpenalty#1#2{%
737     #1=\z@
738   }
739 \fi\fi
```

`\pxrr@check@punct@char` `\pxrr@is@punct@char{<文字コード>}{<和文フラグ>}` : 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。

```
740 \ifpxrr@in@ptex
```

```

741 \def\pxrr@check@punct@char#1#2{%
742   \pxrr@okfalse
743   \ifcase#2\relax
744     \ifnum\xspcode#1=\thr@@\else
745       \pxrr@oktrue
746     \fi
747   \else
748     \ifnum\inhibitxspcode#1=\thr@@\else
749       \pxrr@oktrue
750     \fi
751   \fi
752 }

```

LuaTeX-j<sub>a</sub> 使用時も基本的に pT<sub>E</sub>X と同じロジックを使う。ただし LuaTeX-j<sub>a</sub> 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。

```

753 \else\ifpxrr@in@luatexja
754   \def\pxrr@check@punct@char#1#2{%
755     \ifnum\ltjgetparameter{jaxspmode}{#1}=\thr@@
756       \pxrr@okfalse
757     \else
758       \pxrr@oktrue
759     \fi
760   }

```

それ以外の場合はゼロとして扱う。

```

761 \else
762   \def\pxrr@check@punct@char#1#2{%
763     \pxrr@okfalse
764   }
765 \fi\fi

```

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

```

766 \ifx\inhibitglue\undefined
767   \let\pxrr@inhibitglue\relax
768 \else
769   \let\pxrr@inhibitglue\inhibitglue
770 \fi

```

## 4.7 パラメタ設定公開命令

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

```

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

```

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

```

772 \newcommand*\rubyssetup[1]{%
773   \pxrr@in@setuptrue
774   \pxrr@fatal@errorfalse
775   \pxrr@parse@option{#1}%
776   \ifpxrr@fatal@error\else
777     \pxrr@csletcs{ifpxrr@d@bprotr}{ifpxrr@bprotr}%
778     \pxrr@csletcs{ifpxrr@d@aprotr}{ifpxrr@aprotr}%
779     \let\pxrr@d@bintr\pxrr@bintr@
780     \let\pxrr@d@aintr\pxrr@aintr@
781     \let\pxrr@d@athead\pxrr@athead
782     \let\pxrr@d@mode\pxrr@mode
783     \let\pxrr@d@side\pxrr@side
784     \let\pxrr@d@evensp\pxrr@evensp
785     \let\pxrr@d@fullsize\pxrr@fullsize
786   \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。

```

787   \pxrr@in@setupfalse
788 }

```

\rubysfontsetup 対応するパラメタを設定する。

```

789 \newcommand*\rubysfontsetup{}
790 \def\rubysfontsetup#1{%
791   \def\pxrr@ruby@font
792 }

```

\rubysbigintrusion 対応するパラメタを設定する。

```

\rubysmallintrusion 793 \newcommand*\rubysbigintrusion[1]{%
794   \edef\pxrr@big@intr{#1}%
\rubysmaxmargin 795 }
\rubyintergap 796 \newcommand*\rubysmallintrusion[1]{%
\rubysizeratio 797   \edef\pxrr@small@intr{#1}%
798 }
799 \newcommand*\rubysmaxmargin[1]{%
800   \edef\pxrr@maxmargin{#1}%
801 }
802 \newcommand*\rubyintergap[1]{%
803   \edef\pxrr@inter@gap{#1}%
804 }
805 \newcommand*\rubysizeratio[1]{%
806   \edef\pxrr@size@ratio{#1}%
807 }

```

\rubysusejghost 対応するスイッチを設定する。

```

\rubynousejghost 808 \newcommand*\rubysusejghost{%
809   \pxrr@jghosttrue
810 }
811 \newcommand*\rubynousejghost{%

```

```

812 \pxrr@jghostfalse
813 }

\rubyuseaghost 対応するスイッチを設定する。
\rubynouseaghost 814 \newcommand*\rubyuseaghost{%
815 \pxrr@aghosttrue
816 \pxrr@setup@aghost
817 }
818 \newcommand*\rubynouseaghost{%
819 \pxrr@aghostfalse
820 }

\rubyadjustatlineedge 対応するスイッチを設定する。
\rubynoadjustatlineedge 821 \newcommand*\rubyadjustatlineedge{%
822 \pxrr@edge@adjusttrue
823 }
824 \newcommand*\rubynoadjustatlineedge{%
825 \pxrr@edge@adjustfalse
826 }

\rubybreakjukugo 対応するスイッチを設定する。
\rubynobreakjukugo 827 \newcommand*\rubybreakjukugo{%
828 \pxrr@break@jukugotrue
829 }
830 \newcommand*\rubynobreakjukugo{%
831 \pxrr@break@jukugofalse
832 }

\rubysafemode 対応するスイッチを設定する。
\rubynosafemode 833 \newcommand*\rubysafemode{%
834 \pxrr@safe@modetrue
835 }
836 \newcommand*\rubynosafemode{%
837 \pxrr@safe@modefalse
838 }

\rubystretchprop 対応するパラメタを設定する。
\rubystretchprophead 839 \newcommand*\rubystretchprop[3]{%
840 \edef\pxrr@sprop@x{#1}%
\rubystretchpropend 841 \edef\pxrr@sprop@y{#2}%
842 \edef\pxrr@sprop@z{#3}%
843 }
844 \newcommand*\rubystretchprophead[2]{%
845 \edef\pxrr@sprop@hy{#1}%
846 \edef\pxrr@sprop@hz{#2}%
847 }
848 \newcommand*\rubystretchpropend[2]{%
849 \edef\pxrr@sprop@ex{#1}%
850 \edef\pxrr@sprop@ey{#2}%

```

```
851 }
```

`\rubyuseextra` 残念ながら今のところは使用不可。

```
852 \newcommand*\rubyuseextra[1]{%
853   \pxrr@cmta=#1\relax
854   \ifnum\pxrr@cmta=\z@
855     \chardef\pxrr@extra\pxrr@cmta
856   \else
857     \pxrr@err@inv@value{\the\pxrr@cmta}%
858   \fi
859 }
```

## 4.8 ルビオプシオン解析

`\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。)

```
860 \let\pxrr@bintr@\@empty
861 \let\pxrr@aintr@\@empty
```

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

```
862 \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。

```
863 \def\pxrr@parse@option#1{%
```

入力が「||」の場合は、「|-|」に置き換える。

```
864   \edef\pxrr@tempa{#1}%
865   \ifx\pxrr@tempa\pxrr@doublebar
866     \def\pxrr@tempa{|-|}%
867   \fi
```

各パラメタの値を全般設定のもので初期化する。

```
868   \pxrr@csletcs{ifpxrr@bprotr}{ifpxrr@d@bprotr}%
869   \pxrr@csletcs{ifpxrr@aprotr}{ifpxrr@d@aprotr}%
870   \let\pxrr@bintr@\pxrr@d@bintr
871   \let\pxrr@aintr@\pxrr@d@aintr
872   \let\pxrr@athead\pxrr@d@athead
873   \let\pxrr@mode\pxrr@d@mode
874   \let\pxrr@side\pxrr@d@side
875   \let\pxrr@evensp\pxrr@d@evensp
876   \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。

```
877   \let\pxrr@bscomp\relax
878   \let\pxrr@ascomp\relax
879   \pxrr@bnobrfalse
880   \pxrr@anoobrfalse
```

```
881 \pxrr@bfintrfalse
```

```
882 \pxrr@afintrfalse
```

明示フラグを偽にする。

```
883 \pxrr@mode@givenfalse
```

```
884 \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。

```
885 \ifpxrr@truby
```

```
886 \let\pxrr@mode=M%
```

```
887 \fi
```

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

```
888 \def\pxrr@po@FS{bi}%
```

```
889 \expandafter\pxrr@parse@option@loop\pxrr@tempa @\pxrr@end
```

```
890 }
```

有限状態機械のループ。

```
891 \def\pxrr@parse@option@loop#1{%
```

```
892 \ifpxrr@Debug
```

```
893 \typeout{\pxrr@po@FS/#1[\@nameuse{\pxrr@po@C@#1}]}%
```

```
894 \fi
```

```
895 \csname pxrr@po@PR@#1\endcsname
```

```
896 \expandafter\ifx\csname pxrr@po@C@#1\endcsname\relax
```

```
897 \let\pxrr@po@FS\relax
```

```
898 \else
```

```
899 \pxrr@letcs\pxrr@po@FS
```

```
900 {pxrr@po@TR@\pxrr@po@FS @\@nameuse{\pxrr@po@C@#1}]}%
```

```
901 \fi
```

```
902 \ifpxrr@Debug
```

```
903 \typeout{->\pxrr@po@FS}%
```

```
904 \fi
```

```
905 \pxrr@ifx{\pxrr@po@FS\relax}{%
```

```
906 \pxrr@fatal@unx@letter{#1}%
```

```
907 \pxrr@parse@option@exit
```

```
908 }{%
```

```
909 \pxrr@parse@option@loop
```

```
910 }%
```

```
911 }
```

後処理。

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

既定値設定 (\rubyssetup) である場合何もしない。

```
913 \ifpxrr@in@setup\else
```

両側ルビ命令の場合は、 \pxrr@side の値を変更する。

```
914 \ifpxrr@truby
```

```
915 \chardef\pxrr@side\tw@
```

```
916 \fi
```



整合性検査を行う。

```
917 \pxrr@check@option
    \pxrr@?intr の値を設定する。
918 \@tempdima=\pxrr@ruby@zw\relax
919 \@tempdimb=\pxrr@or@zero\pxrr@bintr@\@tempdima
920 \edef\pxrr@bintr{\the\@tempdimb}%
921 \@tempdimb=\pxrr@or@zero\pxrr@aintr@\@tempdima
922 \edef\pxrr@aintr{\the\@tempdimb}%
923 \fi
924 }
```

\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。

```
925 \def\pxrr@or@zero#1{%
926 \ifx#1\@empty \pxrr@zero
927 \else #1%
928 \fi
929 }
```

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。

```
930 \def\pxrr@po@C@{F}
931 \@namedef{pxrr@po@C@|}{V}
932 \@namedef{pxrr@po@C@:}{S}
933 \@namedef{pxrr@po@C@.}{S}
934 \@namedef{pxrr@po@C@*}{S}
935 \@namedef{pxrr@po@C@!}{S}
936 \@namedef{pxrr@po@C@<}{B}
937 \@namedef{pxrr@po@C@()}{B}
938 \@namedef{pxrr@po@C@>}{A}
939 \@namedef{pxrr@po@C@)}{A}
940 \@namedef{pxrr@po@C@-}{M}
941 \def\pxrr@po@C@c{M}
942 \def\pxrr@po@C@h{M}
943 \def\pxrr@po@C@H{M}
944 \def\pxrr@po@C@m{M}
945 \def\pxrr@po@C@g{M}
946 \def\pxrr@po@C@j{M}
947 \def\pxrr@po@C@M{M}
948 \def\pxrr@po@C@J{M}
949 \def\pxrr@po@C@P{M}
950 \def\pxrr@po@C@S{M}
951 \def\pxrr@po@C@e{M}
952 \def\pxrr@po@C@E{M}
953 \def\pxrr@po@C@f{M}
954 \def\pxrr@po@C@F{M}
```

機能プロセス。

```
955 \def\pxrr@po@PR@{%
```

```

956 \pxrr@parse@option@exit
957 }
958 \@namedef{pxrr@po@PR@|}{%
959 \csname pxrr@po@PRbar@\pxrr@po@FS\endcsname
960 }
961 \def\pxrr@po@PRbar@bi{%
962 \def\pxrr@bintr@{}\pxrr@bprottrue
963 }
964 \def\pxrr@po@PRbar@bb{%
965 \pxrr@bprotfalse
966 }
967 \def\pxrr@po@PRbar@bs{%
968 \def\pxrr@aintr@{}\pxrr@aprotrtrue
969 }
970 \let\pxrr@po@PRbar@mi\pxrr@po@PRbar@bs
971 \let\pxrr@po@PRbar@as\pxrr@po@PRbar@bs
972 \let\pxrr@po@PRbar@ai\pxrr@po@PRbar@bs
973 \def\pxrr@po@PRbar@ab{%
974 \pxrr@aprotrfalse
975 }
976 \@namedef{pxrr@po@PR@:}{%
977 \csname pxrr@po@PRcolon@\pxrr@po@FS\endcsname
978 }
979 \def\pxrr@po@PRcolon@bi{%
980 \let\pxrr@bscomp=: \relax
981 }
982 \let\pxrr@po@PRcolon@bb\pxrr@po@PRcolon@bi
983 \let\pxrr@po@PRcolon@bs\pxrr@po@PRcolon@bi
984 \def\pxrr@po@PRcolon@mi{%
985 \let\pxrr@ascomp=: \relax
986 }
987 \let\pxrr@po@PRcolon@as\pxrr@po@PRcolon@mi
988 \@namedef{pxrr@po@PR@.}{%
989 \csname pxrr@po@PRdot@\pxrr@po@FS\endcsname
990 }
991 \def\pxrr@po@PRdot@bi{%
992 \let\pxrr@bscomp=. \relax
993 }
994 \let\pxrr@po@PRdot@bb\pxrr@po@PRdot@bi
995 \let\pxrr@po@PRdot@bs\pxrr@po@PRdot@bi
996 \def\pxrr@po@PRdot@mi{%
997 \let\pxrr@ascomp=. \relax
998 }
999 \let\pxrr@po@PRdot@as\pxrr@po@PRdot@mi
1000 \@namedef{pxrr@po@PR@*}{%
1001 \csname pxrr@po@PRstar@\pxrr@po@FS\endcsname
1002 }
1003 \def\pxrr@po@PRstar@bi{%
1004 \pxrr@bnobrtrue

```

```

1005 }
1006 \let\pxrr@po@PRstar@bb\pxrr@po@PRstar@bi
1007 \let\pxrr@po@PRstar@bs\pxrr@po@PRstar@bi
1008 \def\pxrr@po@PRstar@mi{%
1009   \pxrr@anobrtrue
1010 }
1011 \let\pxrr@po@PRstar@as\pxrr@po@PRstar@mi
1012 \@namedef{pxrr@po@PR@!}{%
1013   \csname pxrr@po@PRbang@\pxrr@po@FS\endcsname
1014 }
1015 \def\pxrr@po@PRbang@bi{%
1016   \pxrr@bfintrtrue
1017 }
1018 \let\pxrr@po@PRbang@bb\pxrr@po@PRbang@bi
1019 \let\pxrr@po@PRbang@bs\pxrr@po@PRbang@bi
1020 \def\pxrr@po@PRbang@mi{%
1021   \pxrr@afintrtrue
1022 }
1023 \let\pxrr@po@PRbang@as\pxrr@po@PRbang@mi
1024 \@namedef{pxrr@po@PR@<}{%
1025   \def\pxrr@bintr@{\pxrr@big@intr}\pxrr@bprottrue
1026 }
1027 \@namedef{pxrr@po@PR@()}{%
1028   \def\pxrr@bintr@{\pxrr@small@intr}\pxrr@bprottrue
1029 }
1030 \@namedef{pxrr@po@PR@>}{%
1031   \def\pxrr@aintr@{\pxrr@big@intr}\pxrr@aprottrue
1032 }
1033 \@namedef{pxrr@po@PR@)}{%
1034   \def\pxrr@aintr@{\pxrr@small@intr}\pxrr@aprottrue
1035 }
1036 \def\pxrr@po@PR@c{%
1037   \chardef\pxrr@athead\z@
1038   \pxrr@athead@giventrue
1039 }
1040 \def\pxrr@po@PR@h{%
1041   \chardef\pxrr@athead\@ne
1042   \pxrr@athead@giventrue
1043 }
1044 \def\pxrr@po@PR@H{%
1045   \chardef\pxrr@athead\tw@
1046   \pxrr@athead@giventrue
1047 }
1048 \def\pxrr@po@PR@m{%
1049   \let\pxrr@mode=m%
1050   \pxrr@mode@giventrue
1051 }
1052 \def\pxrr@po@PR@g{%
1053   \let\pxrr@mode=g%

```

```

1054 \pxrr@mode@giventrue
1055 }
1056 \def\pxrr@po@PR@j{%
1057 \let\pxrr@mode=j%
1058 \pxrr@mode@giventrue
1059 }
1060 \def\pxrr@po@PR@M{%
1061 \let\pxrr@mode=M%
1062 \pxrr@mode@giventrue
1063 }
1064 \def\pxrr@po@PR@J{%
1065 \let\pxrr@mode=J%
1066 \pxrr@mode@giventrue
1067 }
1068 \def\pxrr@po@PR@P{%
1069 \chardef\pxrr@side\z@
1070 }
1071 \def\pxrr@po@PR@S{%
1072 \chardef\pxrr@side\@ne
1073 }
1074 \def\pxrr@po@PR@E{%
1075 \chardef\pxrr@evensp\z@
1076 }
1077 \def\pxrr@po@PR@e{%
1078 \chardef\pxrr@evensp\@ne
1079 }
1080 \def\pxrr@po@PR@F{%
1081 \chardef\pxrr@fullsize\z@
1082 }
1083 \def\pxrr@po@PR@f{%
1084 \chardef\pxrr@fullsize\@ne
1085 }

```

遷移表。

```

1086 \def\pxrr@po@TR@bi@F{fi}
1087 \def\pxrr@po@TR@bb@F{fi}
1088 \def\pxrr@po@TR@bs@F{fi}
1089 \def\pxrr@po@TR@mi@F{fi}
1090 \def\pxrr@po@TR@as@F{fi}
1091 \def\pxrr@po@TR@ai@F{fi}
1092 \def\pxrr@po@TR@ab@F{fi}
1093 \def\pxrr@po@TR@fi@F{fi}
1094 \def\pxrr@po@TR@bi@V{bb}
1095 \def\pxrr@po@TR@bb@V{bs}
1096 \def\pxrr@po@TR@bs@V{ab}
1097 \def\pxrr@po@TR@mi@V{ab}
1098 \def\pxrr@po@TR@as@V{ab}
1099 \def\pxrr@po@TR@ai@V{ab}
1100 \def\pxrr@po@TR@ab@V{fi}

```

```

1101 \def\pxrr@po@TR@bi@S{bs}
1102 \def\pxrr@po@TR@bb@S{bs}
1103 \def\pxrr@po@TR@bs@S{bs}
1104 \def\pxrr@po@TR@mi@S{as}
1105 \def\pxrr@po@TR@as@S{as}
1106 \def\pxrr@po@TR@bi@B{bs}
1107 \def\pxrr@po@TR@bi@M{mi}
1108 \def\pxrr@po@TR@bb@M{mi}
1109 \def\pxrr@po@TR@bs@M{mi}
1110 \def\pxrr@po@TR@mi@M{mi}
1111 \def\pxrr@po@TR@bi@A{fi}
1112 \def\pxrr@po@TR@bb@A{fi}
1113 \def\pxrr@po@TR@bs@A{fi}
1114 \def\pxrr@po@TR@mi@A{fi}
1115 \def\pxrr@po@TR@as@A{fi}
1116 \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。

```

1117 \def\pxrr@mode@grand{%
1118   \if      m\pxrr@mode m%
1119   \else\if M\pxrr@mode m%
1120   \else\if j\pxrr@mode j%
1121   \else\if J\pxrr@mode j%
1122   \else\if g\pxrr@mode g%
1123   \else ?%
1124   \fi\fi\fi\fi\fi
1125 }

```

`\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。

```

1126 \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。
1127   \ifpxrr@bprotr\else
1128     \ifpxrr@aprotr\else
1129       \pxrr@fatal@bad@no@protr
1130     \fi
1131   \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。

```

1132   \pxrr@oktrue
1133   \ifx\pxrr@bintr@\@empty\else
1134     \pxrr@okfalse
1135   \fi

```

```

1136 \ifx\pxrr@aintr@\@empty\else
1137   \pxrr@okfalse
1138 \fi
1139 \ifpxrr@ghost\else
1140   \pxrr@oktrue
1141 \fi
1142 \ifpxrr@ok\else
1143   \pxrr@fatal@bad@intr
1144 \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。

```

1145 \if g\pxrr@mode\else
1146   \ifpxrr@abody
1147     \let\pxrr@mode=g\relax
1148     \ifpxrr@mode@given
1149       \pxrr@warn@must@group
1150     \fi
1151   \fi
1152 \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。

```

1153 \if \pxrr@mode@grand j%
1154   \ifnum\pxrr@side=\tw@
1155     \let\pxrr@mode=g\relax
1156     \ifpxrr@mode@given
1157       \pxrr@warn@bad@jukugo
1158     \fi
1159   \fi
1160 \fi

```

肩付き指定 (h) に関する検査。

```

1161 \ifnum\pxrr@athead>\z@

```

横組みでは不可なので中付きに変更する。

```

1162   \pxrr@if@in@tate{}\{%else
1163     \chardef\pxrr@athead\z@
1164   }%

```

グループルビでは不可なので中付きに変更する。

```

1165   \if g\pxrr@mode
1166     \chardef\pxrr@athead\z@
1167   \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。

```

1168   \ifnum\pxrr@athead=\z@
1169     \ifpxrr@athead@given
1170       \pxrr@warn@bad@athead
1171     \fi

```

```

1172     \fi
1173     \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。

```

1174     \ifpxrr@abody
1175         \chardef\pxrr@evensp\z@
1176     \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。)

```

1177     \if g\pxrr@mode\else
1178         \chardef\pxrr@evensp\@ne
1179     \fi

```

圏点ルビ同時付加の場合の調整。

```

1180     \ifpxrr@combo
1181         \pxrr@ck@check@option
1182     \fi
1183 }

```

## 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。

```

1184 \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。

```

1185 \let\pxrr@body@zw\pxrr@zeropt
1186 \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。

```

1187 \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)。

```

1188 \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。

```

1189 \def\pxrr@htratio{0}

```

`\pxrr@iiskip` 和文間空白および和欧文間空白の量。

```

\pxrr@iaiskip 1190 \let\pxrr@iiskip\pxrr@zeropt
1191 \let\pxrr@iaiskip\pxrr@zeropt

```

`\pxrr@assign@fsize` 上記の変数（マクロ）を設定する。

```
1192 \def\pxrr@assign@fsize{%
1193   \@tempdima=\f@size\p@
1194   \@tempdima\pxrr@c@size@ratio\@tempdima
1195   \edef\pxrr@ruby@fsize{\the\@tempdima}%
1196   \pxrr@get@zwidth\pxrr@body@zw
1197   \begingroup
1198     \pxrr@use@ruby@font
1199     \pxrr@get@zwidth\pxrr@ruby@zw
1200     \global\let\pxrr@gtempa\pxrr@ruby@zw
1201   \endgroup
1202   \let\pxrr@ruby@zw\pxrr@gtempa
1203   \pxrr@get@iiskip\pxrr@iiskip
1204   \pxrr@get@iaiskip\pxrr@iaiskip
```

`\pxrr@htratio` の値を設定する。

```
1205 \pxrr@if@in@tate{%
1206   \let\pxrr@htratio\pxrr@thtratio
1207 }{%
1208   \let\pxrr@htratio\pxrr@yhtratio
1209 }%
```

`\pxrr@ruby@raise` の値を計算する。

```
1210 \@tempdima\pxrr@body@zw\relax
1211 \@tempdima\pxrr@htratio\@tempdima
1212 \@tempdimb\pxrr@ruby@zw\relax
1213 \advance\@tempdimb-\pxrr@htratio\@tempdimb
1214 \advance\@tempdima\@tempdimb
1215 \@tempdimb\pxrr@body@zw\relax
1216 \advance\@tempdima\pxrr@c@inter@gap\@tempdimb
1217 \edef\pxrr@ruby@raise{\the\@tempdima}%
```

`\pxrr@ruby@lower` の値を計算する。

```
1218 \@tempdima\pxrr@body@zw\relax
1219 \advance\@tempdima-\pxrr@htratio\@tempdima
1220 \@tempdimb\pxrr@ruby@zw\relax
1221 \@tempdimb\pxrr@htratio\@tempdimb
1222 \advance\@tempdima\@tempdimb
1223 \@tempdimb\pxrr@body@zw\relax
1224 \advance\@tempdima\pxrr@c@inter@gap\@tempdimb
1225 \edef\pxrr@ruby@lower{\the\@tempdima}%
```

圏点ルビ同時付加の設定。

```
1226 \ifpxrr@combo
1227   \pxrr@ck@assign@fsize
1228 \fi
1229 }
```

`\pxrr@use@ruby@font` ルビ用のフォントに切り替える。

```
1230 \def\pxrr@use@ruby@font{%
```



```

1231 \pxrr@without@macro@trace{%
1232 \let\rubyfontsize\pxrr@ruby@fsize
1233 \fontsize{\pxrr@ruby@fsize}{\z@}\selectfont
1234 \pxrr@c@ruby@font
1235 }%
1236 }

```

#### 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 1237 \chardef\pxrr@locate@inner=1
\pxrr@locate@end 1238 \chardef\pxrr@locate@head=0
1239 \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。

```
1240 \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。

```

1241 \setbox#2\pxrr@hbox{#5}\@tempdima\wd#2%
1242 \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。

```

1243 \pxrr@decompose{#5}%
1244 \pxrr@evenspace@int{#1}{#2}{#3}{#4}%
1245 }

```

ここから実行を開始することもある。

```
1246 \def\pxrr@evenspace@int#1#2#3#4{%
```

比率パラメタの設定。

```

1247 \pxrr@save@listproc
1248 \ifcase#1%
1249 \pxrr@evenspace@param\pxrr@zero\pxrr@sprop@hy\pxrr@sprop@hz
1250 \or
1251 \pxrr@evenspace@param\pxrr@sprop@x\pxrr@sprop@y\pxrr@sprop@z
1252 \or

```

```

1253 \pxrr@evenspace@param\pxrr@sprop@ex\pxrr@sprop@ey\pxrr@zero
1254 \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。

```

1255 \pxrr@dima=\pxrr@cntr\p@
1256 \advance\pxrr@dima-\p@
1257 \pxrr@dima=\pxrr@sprop@y@\pxrr@dima
1258 \advance\pxrr@dima\pxrr@sprop@x@\p@
1259 \advance\pxrr@dima\pxrr@sprop@z@\p@
1260 \ifdim\pxrr@dima>z@\else
1261 \ifnum#1>z@
1262 \let\pxrr@sprop@x@\@ne
1263 \advance\pxrr@dima\p@
1264 \fi
1265 \ifnum#1<\tw@
1266 \let\pxrr@sprop@z@\@ne
1267 \advance\pxrr@dima\p@
1268 \fi
1269 \fi
1270 \edef\pxrr@tempa{\strip@pt\pxrr@dima}%
1271 \ifpxrr@Debug
1272 \typeout{\number\pxrr@sprop@x@:\number\pxrr@sprop@z@:\pxrr@tempa}%
1273 \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。

```

1274 \def\pxrr@pre##1{\pxrr@hfilx\pxrr@sprop@x@ ##1}%
1275 \def\pxrr@inter##1{\pxrr@hfilx\pxrr@sprop@y@ ##1}%
1276 \def\pxrr@post{\pxrr@hfilx\pxrr@sprop@z@}%
1277 \def\pxrr@makebox@res{%
1278 \setbox#2=\pxrr@hbox@to#4{#3\pxrr@res}%
1279 }%
1280 \pxrr@makebox@res

```

前後の空白の量を求める。

```

1281 \pxrr@dima\wd#2%
1282 \advance\pxrr@dima-\pxrr@natwd\relax
1283 \pxrr@invscale\pxrr@dima\pxrr@tempa
1284 \@tempdima\pxrr@sprop@x@\pxrr@dima
1285 \edef\pxrr@bspace{\the\@tempdima}%
1286 \@tempdima\pxrr@sprop@z@\pxrr@dima
1287 \edef\pxrr@aspace{\the\@tempdima}%
1288 \pxrr@restore@listproc
1289 \ifpxrr@Debug
1290 \typeout{\pxrr@bspace:\pxrr@aspace}%
1291 \fi
1292 }

```

```

1293 \def\pxrr@evenspace@param#1#2#3{%
1294   \let\pxrr@sprop@x@#1%
1295   \let\pxrr@sprop@y@#2%
1296   \let\pxrr@sprop@z@#3%
1297 }
1298 \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。

```

1299 \def\pxrr@adjust@margin{%
1300   \pxrr@save@listproc
1301   \@tempdima\pxrr@body@zw\relax
1302   \@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。

```

1303   \@tempswafalse
1304   \def\pxrr@pre##1{\pxrr@hfilx\pxrr@sprop@x@ ##1}%
1305   \def\pxrr@inter##1{\pxrr@hfilx\pxrr@sprop@y@ ##1}%
1306   \def\pxrr@post{\pxrr@hfilx\pxrr@sprop@z@}%
1307   \ifnum\pxrr@cntr>\@ne
1308     \ifdim\pxrr@bspace>\@tempdima
1309       \edef\pxrr@bspace{\the\@tempdima}%
1310       \def\pxrr@pre##1{\hskip\pxrr@bspace\relax ##1}%
1311       \@tempswatrue
1312     \fi
1313     \ifdim\pxrr@aspace>\@tempdima
1314       \edef\pxrr@aspace{\the\@tempdima}%
1315       \def\pxrr@post{\hskip\pxrr@aspace\relax}%
1316       \@tempswatrue
1317     \fi
1318   \fi

```

必要に応じて再調整を行う。

```

1319   \if@tempswa
1320     \pxrr@makebox@res
1321   \fi
1322   \pxrr@restore@listproc
1323   \ifpxrr@Debug
1324     \typeout{\pxrr@bspace:\pxrr@aspace}%
1325   \fi
1326 }

```

`\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。

```

1327 \def\pxrr@save@listproc{%
1328   \let\pxrr@pre@save\pxrr@pre
1329   \let\pxrr@inter@save\pxrr@inter

```

```

1330 \let\pxrr@post@save\pxrr@post
1331 }
1332 \let\pxrr@pre@save\@undefined
1333 \let\pxrr@inter@save\@undefined
1334 \let\pxrr@post@save\@undefined

```

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

```

1335 \def\pxrr@restore@listproc{%
1336 \let\pxrr@pre\pxrr@pre@save
1337 \let\pxrr@inter\pxrr@inter@save
1338 \let\pxrr@post\pxrr@post@save
1339 }

```

## 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。

```

1340 \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。

```

1341 \def\pxrr@transform@kana#1{%
1342 \let\pxrr@trans@res\@empty
1343 \def\pxrr@transform@kana@end\pxrr@end{%
1344 \let#1\pxrr@trans@res
1345 }%
1346 \expandafter\pxrr@transform@kana@loop@a#1\pxrr@end
1347 }
1348 \def\pxrr@transform@kana@loop@a{%
1349 \futurelet\pxrr@token\pxrr@transform@kana@loop@b
1350 }
1351 \def\pxrr@transform@kana@loop@b{%
1352 \ifx\pxrr@token\pxrr@end
1353 \let\pxrr@tempb\pxrr@transform@kana@end
1354 \else\ifx\pxrr@token\bgroup
1355 \let\pxrr@tempb\pxrr@transform@kana@loop@c
1356 \else\ifx\pxrr@token\@sptoken
1357 \let\pxrr@tempb\pxrr@transform@kana@loop@d
1358 \else
1359 \let\pxrr@tempb\pxrr@transform@kana@loop@e
1360 \fi\fi\fi
1361 \pxrr@tempb
1362 }
1363 \def\pxrr@transform@kana@loop@c#1{%
1364 \pxrr@appto\pxrr@trans@res{{#1}}%
1365 \pxrr@transform@kana@loop@a
1366 }
1367 \expandafter\def\expandafter\pxrr@transform@kana@loop@d\space{%
1368 \pxrr@appto\pxrr@trans@res{ }%

```

```

1369 \pxrr@transform@kana@loop@a
1370 }
1371 \def\pxrr@transform@kana@loop@e#1{%
1372 \expandafter\pxrr@transform@kana@loop@f\string#1\pxrr@nil#1%
1373 }
1374 \def\pxrr@transform@kana@loop@f#1#2\pxrr@nil#3{%
1375 \@tempswafalse
1376 \ifnum'#1>\@cclv
1377 \begingroup\expandafter\expandafter\expandafter\endgroup
1378 \expandafter\ifx\csname pxrr@nonsmall/#3\endcsname\relax\else
1379 \@tempwattrue
1380 \fi
1381 \fi
1382 \if@tempswa
1383 \edef\pxrr@tempa{%
1384 \noexpand\pxrr@appto\noexpand\pxrr@trans@res
1385 {\csname pxrr@nonsmall/#3\endcsname}%
1386 }%
1387 \pxrr@tempa
1388 \else
1389 \pxrr@appto\pxrr@trans@res{#3}%
1390 \fi
1391 \pxrr@transform@kana@loop@a
1392 }
1393 \def\pxrr@assign@nonsmall#1/#2\pxrr@nil{%
1394 \pxrr@get@jchar@token\pxrr@tempa{\pxrr@jc{#1}}%
1395 \pxrr@get@jchar@token\pxrr@tempb{\pxrr@jc{#2}}%
1396 \expandafter\edef\csname pxrr@nonsmall/\pxrr@tempa\endcsname
1397 {\pxrr@tempb}%
1398 }
1399 \@tfor\pxrr@tempc:=%
1400 {2421:3041/2422:3042}{2423:3043/2424:3044}%
1401 {2425:3045/2426:3046}{2427:3047/2428:3048}%
1402 {2429:3049/242A:304A}{2443:3063/2444:3064}%
1403 {2463:3083/2464:3084}{2465:3085/2466:3086}%
1404 {2467:3087/2468:3088}{246E:308E/246F:308F}%
1405 {2521:30A1/2522:30A2}{2523:30A3/2524:30A4}%
1406 {2525:30A5/2526:30A6}{2527:30A7/2528:30A8}%
1407 {2529:30A9/252A:30AA}{2543:30C3/2544:30C4}%
1408 {2563:30E3/2564:30E4}{2565:30E5/2566:30E6}%
1409 {2567:30E7/2568:30E8}{256E:30EE/256F:30EF}%
1410 \do{%
1411 \expandafter\pxrr@assign@nonsmall\pxrr@tempc\pxrr@nil
1412 }

```

#### 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。

```
1413 \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。

```
1414 \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。

```
1415 \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は、差はないものとみなす（演算誤差対策）。

```
1416 \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。

```
1417 \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。

```
1418 \let\pxrr@compose@block@do\pxrr@compose@oneside@block@do
```

```
1419 \pxrr@compose@block@pre{#1}{#2}{#3}{}%
```

```
1420 }
```

こちらが本体。

```
1421 % #4 は空
```

```
1422 \def\pxrr@compose@oneside@block@do#1#2#3#4{%
```

```
1423 \setbox\pxrr@boxa\pxrr@hbox{#2}%
```

```
1424 \edef\pxrr@ck@body@natwd{\the\wd\pxrr@boxa}%
```

```
1425 \let\pxrr@ck@locate\pxrr@locate@inner
```

```
1426 \setbox\pxrr@boxr\pxrr@hbox{%
```

```
1427 \pxrr@use@ruby@font
```

```
1428 #3%
```

```
1429 }%
```

```
1430 \@tempdima\wd\pxrr@boxr
```

```
1431 \advance\@tempdima-\wd\pxrr@boxa
```

```
1432 \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で組み直す。

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

```
1433 \pxrr@protrtrue
```

```
1434 \let\pxrr@locate@temp#1%
```

```

1435 \ifnum\pxrr@athead>\@ne
1436 \ifnum\pxrr@locate@temp=\pxrr@locate@inner
1437 \let\pxrr@locate@temp\pxrr@locate@head
1438 \fi
1439 \fi
1440 \let\pxrr@ck@locate\pxrr@locate@temp
1441 \pxrr@decompose{#2}%
1442 \edef\pxrr@natwd{\the\wd\pxrr@boxa}%
1443 \pxrr@evenspace@int\pxrr@locate@temp\pxrr@boxa\relax
1444 {\wd\pxrr@boxr}%
1445 \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。

```

1446 \pxrr@protrfalse
1447 \ifnum\pxrr@athead=\z@
1448 \pxrr@decompose{#3}%
1449 \edef\pxrr@natwd{\the\wd\pxrr@boxr}%
1450 \pxrr@evenspace@int{#1}\pxrr@boxr
1451 \pxrr@use@ruby@font{\wd\pxrr@boxa}%
1452 \pxrr@adjust@margin
1453 \fi
1454 \let\pxrr@bspace\pxrr@zeropt
1455 \let\pxrr@aspace\pxrr@zeropt
1456 \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）。

```

1457 \pxrr@protrfalse
1458 \let\pxrr@bspace\pxrr@zeropt
1459 \let\pxrr@aspace\pxrr@zeropt
1460 \fi\fi

```

実際に組版を行う。

```

1461 \setbox\z@\hbox{%
1462 \ifnum\pxrr@side=\z@
1463 \raise\pxrr@ruby@raise\box\pxrr@boxr
1464 \else
1465 \lower\pxrr@ruby@lower\box\pxrr@boxr
1466 \fi
1467 }%
1468 \ifnum \ifpxrr@combo\pxrr@ck@ruby@combo\else\z@\fi >\z@
1469 \pxrr@ck@compose{#2}%
1470 \fi
1471 \ht\z@\z@ \dp\z@\z@
1472 \@tempdima\wd\z@
1473 \setbox\pxrr@boxr\hbox{%
1474 \box\z@

```

```

1475     \kern-\@tempdima
1476     \box\pxrr@boxa
1477 }%

\ifpxrr@any@protr を設定する。
1478 \ifpxrr@protr
1479     \pxrr@any@protrtrue
1480 \fi
1481 }

```

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

```

1482 \def\pxrr@compose@twoside@block{%
1483     \let\pxrr@compose@block@do\pxrr@compose@twoside@block@do
1484     \pxrr@compose@block@pre
1485 }
1486 \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。

```

1487     \setbox\pxrr@boxa\pxrr@hbox{#2}%
1488     \edef\pxrr@ck@body@natwd{\the\wd\pxrr@boxa}%
1489     \let\pxrr@ck@locate\pxrr@locate@inner
1490     \setbox\pxrr@boxr\pxrr@hbox{%
1491         \pxrr@use@ruby@font
1492         #3%
1493     }%
1494     \setbox\pxrr@boxb\pxrr@hbox{%
1495         \pxrr@use@ruby@font
1496         #4%
1497     }%

```

「何れかのルビが親文字列より長いか」を検査する。

```

1498 \@tempswafalse
1499 \@tempdima\wd\pxrr@boxr
1500 \advance\@tempdima-\wd\pxrr@boxa
1501 \ifdim\pxrr@epsilon<\@tempdima \@tempwattrue \fi
1502 \@tempdima\wd\pxrr@boxb
1503 \advance\@tempdima-\wd\pxrr@boxa
1504 \ifdim\pxrr@epsilon<\@tempdima \@tempwattrue \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。)

```

1505 \if@tempwa
1506     \pxrr@protrtrue

```

「拡張肩付き」指定の場合、前側の突出を抑止する。

```

1507     \let\pxrr@locate@temp#1%
1508     \ifnum\pxrr@athead>\@ne
1509         \ifnum\pxrr@locate@temp=\pxrr@locate@inner

```



```

1510      \let\pxrr@locate@temp\pxrr@locate@head
1511      \fi
1512      \fi
1513      \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。

```

1514      \ifdim\wd\pxrr@boxr<\wd\pxrr@boxb
1515          \pxrr@compose@twoside@block@sub{#2}{#3}%
1516          \pxrr@boxr\pxrr@boxb
1517      \else
1518          \pxrr@compose@twoside@block@sub{#2}{#4}%
1519          \pxrr@boxb\pxrr@boxr
1520      \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。

```

1521      \else
1522          \pxrr@protrfalse

```

肩付きルビの場合は組み直しを行わない。

```

1523      \ifnum\pxrr@athead=\z@
1524          \@tempdima\wd\pxrr@boxa
1525          \advance\@tempdima-\wd\pxrr@boxr
1526          \ifdim\pxrr@epsilon<\@tempdima
1527              \pxrr@decompose{#3}%
1528              \edef\pxrr@natwd{\the\wd\pxrr@boxr}%
1529              \pxrr@evenspace@int{#1}\pxrr@boxr
1530              \pxrr@use@ruby@font{\wd\pxrr@boxa}%
1531              \pxrr@adjust@margin
1532          \fi
1533          \@tempdima\wd\pxrr@boxa
1534          \advance\@tempdima-\wd\pxrr@boxb
1535          \ifdim\pxrr@epsilon<\@tempdima
1536              \pxrr@decompose{#4}%
1537              \edef\pxrr@natwd{\the\wd\pxrr@boxb}%
1538              \pxrr@evenspace@int{#1}\pxrr@boxb
1539              \pxrr@use@ruby@font{\wd\pxrr@boxa}%
1540              \pxrr@adjust@margin
1541          \fi
1542      \fi

```

\pxrr@?space はゼロに設定する。

```

1543      \let\pxrr@bspace\pxrr@zeropt
1544      \let\pxrr@aspace\pxrr@zeropt
1545      \fi

```

実際に組版を行う。

```

1546      \setbox\z@\hbox{%
1547          \@tempdima\wd\pxrr@boxr

```

```

1548 \raise\pxrr@ruby@raise\box\pxrr@boxr
1549 \kern-\@tempdima
1550 \lower\pxrr@ruby@lower\box\pxrr@boxb
1551 }%
1552 \ifnum \ifpxrr@combo\pxrr@ck@ruby@combo\else\z@\fi >\z@
1553 \pxrr@ck@compose{#2}%
1554 \fi
1555 \ht\z@\z@ \dp\z@\z@
1556 \@tempdima\wd\z@
1557 \setbox\pxrr@boxr\hbox{%
1558 \box\z@
1559 \kern-\@tempdima
1560 \box\pxrr@boxa
1561 }%
1562 }

```

\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。

```

1563 \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。

```

1564 \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。

```

1565 \def\pxrr@compose@twoside@block@sub#1#2#3#4{%
1566 \pxrr@decompose{#1}%
1567 \edef\pxrr@natwd{\the\wd\pxrr@boxa}%
1568 \pxrr@evenspace@int\pxrr@locate@temp\pxrr@boxa\relax{\wd#4}%
1569 \@tempdima\wd#4%
1570 \advance\@tempdima-\pxrr@bspace\relax
1571 \advance\@tempdima-\pxrr@aspace\relax
1572 \edef\pxrr@body@wd{\the\@tempdima}%
1573 \advance\@tempdima-\wd#3%
1574 \ifdim\pxrr@epsilon<\@tempdima
1575 \edef\pxrr@restore@margin@values{%
1576 \edef\noexpand\pxrr@bspace{\pxrr@bspace}%
1577 \edef\noexpand\pxrr@aspace{\pxrr@aspace}%
1578 }%
1579 \pxrr@decompose{#2}%
1580 \edef\pxrr@natwd{\the\wd#3}%
1581 \pxrr@evenspace@int\pxrr@locate@temp#3%
1582 \pxrr@use@ruby@font{\pxrr@body@wd}%
1583 \pxrr@adjust@margin

```

```

1584 \pxrr@restore@margin@values
1585 \setbox#3\hbox{%
1586 \kern\pxrr@bspace\relax
1587 \box#3%
1588 }%
1589 \else
1590 \ifnum\pxrr@locate@temp=\pxrr@locate@head
1591 \@tempdima\z@
1592 \else\ifnum\pxrr@locate@temp=\pxrr@locate@inner
1593 \@tempdima.5\@tempdima
1594 \fi\fi
1595 \advance\@tempdima\pxrr@bspace\relax
1596 \setbox#3\hbox{%
1597 \kern\@tempdima
1598 \box#3%
1599 }%
1600 \fi
1601 }
1602 % \end{macrocode}
1603 \end{macro}
1604 %
1605 % \begin{macro}{\pxrr@compose@block@pre}
1606 % |\pxrr@compose@block@pre{|\jmeta{パターン}}|{|^A
1607 %r \jmeta{親文字}}|{|\jmeta{ルビ 1}}|{|\jmeta{ルビ 2}}|}|\Means
1608 % 親文字列・ルビ文字列の加工を行う。
1609 % \Note 両側ルビ対応のため、ルビ用引数が 2 つある。
1610 % \begin{macrocode}
1611 \def\pxrr@compose@block@pre{%
1612
1613     f 指定時は小書き仮名の変換を施す。
1614
1615     \pxrr@cond\ifnum\pxrr@fullsize>\z@\fi{%
1616         \pxrr@compose@block@pre@a
1617     }{%
1618         \pxrr@compose@block@pre@d
1619     }%
1620
1621     {パターン}{親文字}{ルビ 1}{ルビ 2}
1622     \def\pxrr@compose@block@pre@a#1#2#3#4{%
1623         \def\pxrr@compose@block@tempa{#4}%
1624         \pxrr@transform@kana\pxrr@compose@block@tempa
1625         \expandafter\pxrr@compose@block@pre@b
1626         \expandafter{\pxrr@compose@block@tempa}{#1}{#2}{#3}%
1627     }
1628     {ルビ 2}{パターン}{親文字}{ルビ 1}
1629     \def\pxrr@compose@block@pre@d#1#2#3#4{%
1630         \def\pxrr@compose@block@tempa{#4}%
1631         \pxrr@transform@kana\pxrr@compose@block@tempa
1632         \expandafter\pxrr@compose@block@pre@c
1633         \expandafter{\pxrr@compose@block@tempa}{#1}{#2}{#3}%

```

```

1631 }
1632 % {ルビ 1}{ルビ 2}{パターン}{親文字}
1633 \def\pxrr@compose@block@pre@c#1#2#3#4{%
1634   \pxrr@compose@block@pre@d{#3}{#4}{#1}{#2}%
1635 }
1636 \def\pxrr@compose@block@pre@d{%
1637   \pxrr@cond\ifnum\pxrr@evensp=\z@\fi{%
1638     \pxrr@compose@block@pre@e
1639   }{%
1640     \pxrr@compose@block@pre@f
1641   }%
1642 }
1643 % {パターン}{親文字}
1644 \def\pxrr@compose@block@pre@e#1#2{%
1645   \pxrr@compose@block@pre@f{#1}{#2}%
1646 }
1647 \def\pxrr@compose@block@pre@f{%
1648   \pxrr@cond\ifnum\pxrr@evensp=\z@\fi{%
1649     \pxrr@compose@block@pre@g
1650   }{%
1651     \pxrr@compose@block@do
1652   }%
1653 }
1654 % {パターン}{親文字}{ルビ 1}{ルビ 2}
1655 \def\pxrr@compose@block@pre@g#1#2#3#4{%
1656   \pxrr@compose@block@do{#1}{#2}{#3}{#4}%
1657 }
1658 \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。

```

1659 \def\pxrr@add@protect#1{%
1660   \expandafter\pxrr@add@protect@a
1661   \csname\expandafter\@gobble\string#1\space\endcsname#1%
1662 }
1663 \def\pxrr@add@protect@a#1#2{%
1664   \let#1=#2%

```

```

1665 \def#2{\pxrr@check@protect\protect#1}%
1666 }
1667 \def\pxrr@check@protect{%
1668 \ifx\protect\@typeset@protect
1669 \expandafter\@gobble
1670 \fi
1671 }

```

## 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。

```

1672 \let\pxrr@body@input\@empty

```

`\pxrr@prepare@fallback` `\pxrr@prepare@fallback{〈親文字列〉}` :

```

1673 \def\pxrr@prepare@fallback#1{%
1674 \pxrr@fatal@errorfalse
1675 \def\pxrr@body@input{#1}%
1676 }

```

`\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。

```

1677 \def\pxrr@fallback{%
1678 \pxrr@body@input
1679 }

```

`\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。

```

1680 \def\pxrr@if@alive{%
1681 \ifpxrr@fatal@error \expandafter\@gobble
1682 \else \expandafter\@firstofone
1683 \fi
1684 }

```

## 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。

```

1685 \def\pxrr@end@kinsoku{0}

```

`\pxrr@ruby@scan` 片側ルビ用の先読み処理。

```

1686 \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。

```

1687 \def\pxrr@tempc{%
1688     \edef\pxrr@end@kinsoku{\the\pxrr@cntr}%
1689     \pxrr@do@proc{#1}{#2}%
1690 }%
1691 \pxrr@check@kinsoku\pxrr@tempc
1692 }

```

\pxrr@truby@scan 両側ルビ用の先読み処理。

```

1693 \def\pxrr@truby@scan#1#2#3{%
1694     \def\pxrr@tempc{%
1695         \edef\pxrr@end@kinsoku{\the\pxrr@cntr}%
1696         \pxrr@do@proc{#1}{#2}{#3}%
1697     }%
1698     \pxrr@check@kinsoku\pxrr@tempc
1699 }

```

\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。

```

1700 \def\pxrr@check@kinsoku#1{%
1701     \let\pxrr@tempb#1%
1702     \futurelet\pxrr@token\pxrr@check@kinsoku@a
1703 }
1704 \def\pxrr@check@kinsoku@a{%
1705     \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。

```

1706     \ifpxrr@abody\else
1707         \ifnum\pxrr@cntr=\@ne
1708             \pxrr@cntr\tw@
1709         \fi
1710     \fi
1711     \ifcase\pxrr@cntr
1712         \pxrr@cntr\z@
1713         \expandafter\pxrr@tempb
1714     \or
1715         \pxrr@cntr\@MM
1716         \expandafter\pxrr@tempb
1717     \else
1718         \expandafter\pxrr@check@kinsoku@b
1719     \fi
1720 }

```

\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。

```

1721 \def\pxrr@check@kinsoku@b#1{%
1722   \pxrr@check@kinsoku@c#1#1%
1723 }
1724 \def\pxrr@check@kinsoku@c#1{%
1725   \pxrr@get@prebreakpenalty\pxrr@cntr{'#1}%
1726   \pxrr@tempb
1727 }

```

\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。

```

1728 \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。)

```

1729   \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。

```

1730     \ifcat\noexpand##1\relax
1731       \pxrr@cntr\z@
1732     \else\ifcat\noexpand##1\noexpand~%
1733       \pxrr@cntr\z@
1734     \else\ifcat\noexpand##1A%
1735       \pxrr@cntr\@ne
1736     \else\ifcat\noexpand##10%
1737       \pxrr@cntr\@ne
1738     \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。)

```

1739       \pxrr@cntr\z@
1740       \expandafter\pxrr@check@char@a\meaning##1#2\pxrr@nil
1741       \fi\fi\fi\fi
1742     }%
1743   \def\pxrr@check@char@a##1#2##2\pxrr@nil{%
1744     \ifcat @##10%
1745       \pxrr@cntr\tw@
1746     \fi
1747   }%
1748 }

```

規定の引数を用意して「定義マクロ」を呼ぶ。

```

1749 \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。)

```

1750 \let\pxrr@auto@penalty\z@

\pxrr@auto@icspace 文字間の空き。寸法値マクロ。
1751 \let\pxrr@auto@icspace\pxrr@zeropt

\pxrr@intr@amount 進入の幅。寸法値マクロ。
1752 \let\pxrr@intr@amount\pxrr@zeropt

\pxrr@intrude@setauto@j 和文の場合の \pxrr@auto@* の設定。
1753 \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。
1754 \ifpxrr@bnoobr
1755     \let\pxrr@auto@penalty\@MM
1756     \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。
1757 \else
1758     \let\pxrr@auto@penalty\z@
1759     \if : \pxrr@bscomp
1760         \let\pxrr@auto@icspace\pxrr@iaiskip
1761     \else\if . \pxrr@bscomp
1762         \let\pxrr@auto@icspace\pxrr@zeropt
1763     \else
1764         \let\pxrr@auto@icspace\pxrr@iiskip
1765     \fi\fi
1766 \fi
1767 }

\pxrr@intrude@setauto@a 欧文の場合の \pxrr@auto@* の設定。
1768 \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。
1769 \if : \pxrr@bscomp\else
1770     \pxrr@bnoobrtrue
1771 \fi
1772 \ifpxrr@bnoobr
1773     \let\pxrr@auto@penalty\@MM
1774     \let\pxrr@auto@icspace\pxrr@zeropt
1775 \else

    この分岐は和欧文間空白挿入指定（:）に限る。
1776     \let\pxrr@auto@penalty\z@
1777     \let\pxrr@auto@icspace\pxrr@iaiskip
1778 \fi
1779 }

```



#### 4.17.1 前側進入処理

`\pxrr@intrude@head` 前側の進入処理。

```
1780 \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。)

```
1781 \ifpxrr@ghost\else
```

実効の進入幅は `\pxrr@bintr` と `\pxrr@bspace` の小さい方。

```
1782 \let\pxrr@intr@amount\pxrr@bspace
```

```
1783 \ifdim\pxrr@bintr<\pxrr@intr@amount\relax
```

```
1784 \let\pxrr@intr@amount\pxrr@bintr
```

```
1785 \fi
```

`\pxrr@auto@*` の設定法は和文ルビと欧文ルビで処理が異なる。

```
1786 \ifpxrr@abody
```

```
1787 \pxrr@intrude@setauto@a
```

```
1788 \else
```

```
1789 \pxrr@intrude@setauto@j
```

```
1790 \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)。

```
1791 \ifpxrr@par@head
```

```
1792 \ifpxrr@bfintr
```

```
1793 \hskip-\pxrr@intr@amount\relax
```

```
1794 \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。

```
1795 \else
```

```
1796 % \penalty\pxrr@auto@penalty\relax
```

```
1797 \hskip-\pxrr@intr@amount\relax
```

```
1798 \hskip\pxrr@auto@icspace\relax
```

```
1799 \fi
```

```
1800 \fi
```

```
1801 }
```

`\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。

```
1802 \def\pxrr@put@head@penalty{%
```

```
1803 \ifpxrr@ghost\else \ifpxrr@par@head\else
```

```
1804 \ifpxrr@abody
```

```
1805 \pxrr@intrude@setauto@a
```

```
1806 \else
```

```
1807 \pxrr@intrude@setauto@j
```

```
1808 \fi
```

```
1809 \ifnum\pxrr@auto@penalty=z\else
```

```

1810      \pxrr@canta\lastpenalty \unpenalty
1811      \advance\pxrr@canta\pxrr@auto@penalty\relax
1812      \penalty\pxrr@canta
1813      \fi
1814      \fi\fi
1815 }

```

#### 4.17.2 後側進入処理

`\pxrr@intrude@end` 末尾での進入処理。

```

1816 \def\pxrr@intrude@end{%
1817   \ifpxrr@ghost\else

```

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

```

1818     \let\pxrr@intr@amount\pxrr@aspace
1819     \ifdim\pxrr@aintr<\pxrr@intr@amount\relax
1820       \let\pxrr@intr@amount\pxrr@aintr
1821     \fi

```

`\pxrr@auto@*` の設定法は和文ルビと欧文ルビで処理が異なる。

```

1822     \pxrr@csletcs{ifpxrr@bnobr}{ifpxrr@anobr}%
1823     \let\pxrr@bscomp\pxrr@ascomp
1824     \ifpxrr@abody
1825       \pxrr@intrude@setauto@a
1826     \else
1827       \pxrr@intrude@setauto@j
1828     \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。

```

1829     \ifnum\pxrr@auto@penalty=\z@
1830       \let\pxrr@auto@penalty\pxrr@end@kinsoku
1831     \fi
1832     \ifpxrr@afintr

```

段落末尾での進入を許す場合。

```

1833     \ifnum\pxrr@auto@penalty=\z@\else
1834       \penalty\pxrr@auto@penalty\relax
1835     \fi
1836     \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。

```

1837     \hskip\pxrr@auto@icspace\relax
1838     \ifnum\pxrr@auto@penalty=\z@\else
1839       \penalty\@MM
1840     \fi
1841   \else

```

段落末尾での進入を許さない場合。

```
1842 \@tempskipa-\pxrr@intr@amount\relax
1843 \advance\@tempskipa\pxrr@auto@icspace\relax
1844 \ifnum\pxrr@auto@penalty=\z@\else
1845 \penalty\pxrr@auto@penalty\relax
1846 \fi
1847 \hskip\@tempskipa
1848 \ifnum\pxrr@auto@penalty=\z@\else
1849 \penalty\@MM
1850 \fi
1851 \fi
1852 \fi
1853 }
```

## 4.18 メインです

### 4.18.1 エントリーポイント

`\ruby` と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。

```
1854 \AtBeginDocument{%
1855 \providecommand*\ruby{\jruby}%
1856 }
1857 \newcommand*\jruby{%
1858 \pxrr@jprologue
1859 \pxrr@trubyfalse
1860 \pxrr@ruby
1861 }
```

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

```
1862 \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。

```
1863 \newcommand*\aruby{%
1864 \pxrr@aprologue
1865 \pxrr@trubyfalse
1866 \pxrr@ruby
1867 }
1868 \pxrr@add@protect\aruby
```

`\truby` と和文両側ルビの公開命令。

```
1869 \newcommand*\truby{%
1870 \pxrr@jprologue
1871 \pxrr@trubytrue
1872 \pxrr@ruby
1873 }
1874 \pxrr@add@protect\truby
```

`\atruby` 欧文両側ルビの公開命令。

```

1875 \newcommand*{\atruby}{%
1876   \pxrr@aprologue
1877   \pxrr@trubytrue
1878   \pxrr@ruby
1879 }
1880 \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。

```

1881 \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 1882 \let\pxrr@option\@empty
1883 \let\pxrr@exoption\@empty

```

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

```

\pxrr@do@scan 1884 \let\pxrr@do@proc\@empty
1885 \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。

```

1886 \def\pxrr@ruby{%
1887   \@testopt\pxrr@ruby@a{}}%
1888 }
1889 \def\pxrr@ruby@a[#1]{%
1890   \def\pxrr@option{#1}%
1891   \@testopt\pxrr@ruby@b{}}%
1892 }
1893 \def\pxrr@ruby@b[#1]{%
1894   \def\pxrr@exoption{#1}%
1895   \ifpxrr@truby
1896     \let\pxrr@do@proc\pxrr@truby@proc
1897     \let\pxrr@do@scan\pxrr@truby@scan
1898   \else
1899     \let\pxrr@do@proc\pxrr@ruby@proc
1900     \let\pxrr@do@scan\pxrr@ruby@scan
1901   \fi
1902   \pxrr@ruby@c
1903 }
1904 \def\pxrr@ruby@c{%
1905   \ifpxrr@ghost
1906     \expandafter\pxrr@do@proc
1907   \else
1908     \expandafter\pxrr@do@scan
1909   \fi
1910 }

```

`\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。

```

1911 \def\pxrr@mode@is@switching{%
1912   \if      M\pxrr@mode T%
1913   \else\if J\pxrr@mode T%
1914   \else F%
1915   \fi\fi T%
1916 }

```

\pxrr@bind@param “呼出時変数” へのコピーを行う。

```

1917 \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。
1918   \ifpxrr@apply@combo
1919     \pxrr@apply@combofalse
1920     \pxrr@combotrue
1921     \pxrr@ck@bind@param
1922   \else
1923     \pxrr@combofalse
1924   \fi
1925   \let\pxrr@c@ruby@font\pxrr@ruby@font
1926   \let\pxrr@c@size@ratio\pxrr@size@ratio
1927   \let\pxrr@c@inter@gap\pxrr@inter@gap
1928 }

```

\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。

```

1929 \def\pxrr@ruby@proc#1#2{%
1930   \pxrr@prepare@fallback{#1}%
    フォントサイズの変数を設定して、
1931   \pxrr@bind@param
1932   \pxrr@assign@fsize
    オプションを解析する。
1933   \pxrr@parse@option\pxrr@option
    ルビ文字入力をグループ列に分解する。
1934   \pxrr@decompbar{#2}%
1935   \let\pxrr@ruby@list\pxrr@res
1936   \edef\pxrr@ruby@count{\the\pxrr@cntr}%
1937   \let\pxrr@sruby@list\relax
    親文字入力をグループ列に分解する。
1938   \pxrr@decompbar{#1}%
1939   \let\pxrr@body@list\pxrr@res
1940   \edef\pxrr@body@count{\the\pxrr@cntr}%
    安全モードに関する処理を行う。
1941   \ifpxrr@safe@mode
1942     \pxrr@setup@safe@mode
1943   \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。

```
1944 \if\pxrr@mode@is@switching
1945     \pxrr@resolve@mode
1946 \fi
1947 \ifpxrr@Debug
1948     \pxrr@debug@show@input
1949 \fi
```

入力検査を行い、パスした場合は組版処理に進む。

```
1950 \pxrr@if@alive{%
1951     \if g\pxrr@mode
1952         \pxrr@ruby@check@g
1953         \pxrr@if@alive{%
1954             \ifnum\pxrr@body@count>\@ne
1955                 \pxrr@ruby@main@mg
1956             \else
1957                 \pxrr@ruby@main@g
1958             \fi
1959         }%
1960     \else
1961         \pxrr@ruby@check@m
1962         \pxrr@if@alive{\pxrr@ruby@main@m}%
1963     \fi
1964 }%
```

後処理を行う。

```
1965 \pxrr@ruby@exit
1966 }
```

`\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  
の手續の本体。

```
1967 \def\pxrr@truby@proc#1#2#3{%
1968     \pxrr@prepare@fallback{#1}%
```

フォントサイズの変数を設定して、

```
1969 \pxrr@bind@param
1970 \pxrr@assign@fsize
```

オプションを解析する。

```
1971 \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。

```
1972 \def\pxrr@all@input{#{1}{#2}{#3}}%
```

入力文字列のグループ分解を行う。

```
1973 \pxrr@decompbar{#3}%
1974 \let\pxrr@sruby@list\pxrr@res
1975 \edef\pxrr@sruby@count{\the\pxrr@cntr}%
1976 \pxrr@decompbar{#2}%
1977 \let\pxrr@ruby@list\pxrr@res
```

```

1978 \edef\pxrr@ruby@count{\the\pxrr@cntr}%
1979 \pxrr@decompbar{#1}%
1980 \let\pxrr@body@list\pxrr@res
1981 \edef\pxrr@body@count{\the\pxrr@cntr}%

```

安全モードに関する処理を行う。

```

1982 \ifpxrr@safe@mode
1983   \pxrr@setup@safe@mode
1984 \fi
1985 \if\pxrr@mode@is@switching
1986   \pxrr@resolve@mode
1987 \fi
1988 \ifpxrr@Debug
1989   \pxrr@debug@show@input
1990 \fi

```

入力検査を行い、パスした場合は組版処理に進む。

```

1991 \pxrr@if@alive{%
1992   \if g\pxrr@mode
1993     \pxrr@ruby@check@tg
1994     \pxrr@if@alive{\pxrr@ruby@main@tg}%
1995   \else
1996     \pxrr@ruby@check@tm
1997     \pxrr@if@alive{\pxrr@ruby@main@tm}%
1998   \fi
1999 }%

```

後処理を行う。

```

2000 \pxrr@ruby@exit
2001 }

```

`\pxrr@setup@safe@mode` 安全モード用の設定。

```

2002 \def\pxrr@setup@safe@mode{%

```

単純グループ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。

```

2003 \let\pxrr@mode=g\relax
2004 \pxrr@unite@group\pxrr@body@list
2005 \def\pxrr@body@count{1}%
2006 \pxrr@unite@group\pxrr@ruby@list
2007 \def\pxrr@ruby@count{1}%
2008 \ifx\pxrr@sruby@list\relax\else
2009   \pxrr@unite@group\pxrr@sruby@list
2010   \def\pxrr@sruby@count{1}%
2011 \fi

```

“文字単位のスキャン”が必要な機能を無効にする。

```

2012 \chardef\pxrr@evensp\z@
2013 \chardef\pxrr@reversp\z@
2014 \chardef\pxrr@fullsize\z@
2015 }

```

`\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のモードに切り替える。

```
2016 \def\pxrr@resolve@mode{%
2017   \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、
2018   \ifnum\pxrr@ruby@count=\@ne
2019     \let\pxrr@pre\pxrr@decompose
2020     \let\pxrr@post\relax
2021     \pxrr@body@list
2022     \ifnum\pxrr@cntr=\@ne\else
2023       \let\pxrr@mode=g%
2024     \fi
2025   \fi
```

それ以外はモノルビ・熟語ルビを選択する。

```
2026   \if M\pxrr@mode \let\pxrr@mode=m\fi
2027   \if J\pxrr@mode \let\pxrr@mode=j\fi
2028 \ifpxrr@Debug
2029   \pxrr@debug@show@resolve@mode
2030 \fi
```

`\pxrr@check@option` で行っている調整をやり直す。

```
2031   \if g\pxrr@mode
2032     \chardef\pxrr@athead\z@
2033   \fi
2034   \if g\pxrr@mode\else
2035     \chardef\pxrr@evensp\@ne
2036   \fi
2037 \else
2038   \pxrr@fatal@bad@switching
2039 \fi
2040 }
```

#### 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。

```
2041 \def\pxrr@ruby@check@g{%
2042   \ifnum\pxrr@body@count=\pxrr@ruby@count\relax
2043   \ifnum\pxrr@body@count=\@ne\else
2044     \ifpxrr@abody
2045       \pxrr@fatal@bad@movable
2046     \else\ifnum\pxrr@extra=\z@
2047       \pxrr@fatal@na@movable
2048     \fi\fi
```



```

2049 \fi
2050 \else
2051 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
2052 \fi
2053 }

```

`\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。

```

2054 \def\pxrr@ruby@check@m{%
2055 \ifnum\pxrr@body@count=\@ne
2056 \let\pxrr@pre\pxrr@decompose
2057 \let\pxrr@post\relax
2058 \pxrr@body@list
2059 \let\pxrr@body@list\pxrr@res
2060 \edef\pxrr@body@count{\the\pxrr@cntr}%
2061 \ifnum\pxrr@body@count=\pxrr@ruby@count\relax\else
2062 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
2063 \fi
2064 \else
2065 \pxrr@fatal@bad@mono
2066 \fi
2067 }

```

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

`\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。

```

2068 \def\pxrr@ruby@check@tg{%
2069 \ifnum\pxrr@body@count=\pxrr@ruby@count\relax\else
2070 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
2071 \fi
2072 \ifnum\pxrr@body@count=\pxrr@sruby@count\relax\else
2073 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@sruby@count
2074 \fi
2075 \pxrr@if@alive{%
2076 \ifnum\pxrr@body@count=\@ne\else
2077 \ifpxrr@abody
2078 \pxrr@fatal@bad@movable
2079 \else\ifnum\pxrr@extra=\z@
2080 \pxrr@fatal@na@movable
2081 \fi\fi
2082 \fi
2083 }%
2084 }

```

`\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。

```

2085 \def\pxrr@ruby@check@tm{%
2086 \ifnum\pxrr@body@count=\@ne
2087 \let\pxrr@pre\pxrr@decompose
2088 \let\pxrr@post\relax

```

```

2089 \pxrr@body@list
2090 \let\pxrr@body@list\pxrr@res
2091 \edef\pxrr@body@count{\the\pxrr@cntr}%
2092 \ifnum\pxrr@body@count=\pxrr@ruby@count\relax\else
2093 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
2094 \fi
2095 \ifnum\pxrr@body@count=\pxrr@sruby@count\relax\else
2096 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@sruby@count
2097 \fi
2098 \else
2099 \pxrr@fatal@bad@mono
2100 \fi
2101 }

```

#### 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。

```
2102 \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。

```

2103 \def\pxrr@check@par@head{%
2104 \ifvmode
2105 \pxrr@par@headtrue
2106 \else
2107 \pxrr@par@headfalse
2108 \fi
2109 }

```

`\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。

```

2110 \def\pxrr@if@last#1#2#3{%
2111 \ifx#3\pxrr@post #1%
2112 \else #2%
2113 \fi
2114 #3%
2115 }

```

`\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。

```

2116 \def\pxrr@inter@mono{%
2117 \hskip\pxrr@iiskip\relax
2118 }

```

`\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。

```
2119 \def\pxrr@takeout@any@protr@nocolor{%
2120   \ifpxrr@any@protr
2121     \aftergroup\pxrr@any@protrtrue
2122   \fi
2123 }
```

color 使用時の定義。

```
2124 \def\pxrr@takeout@any@protr{%
2125   \ifpxrr@any@protr
2126     \aftergroup\pxrr@takeout@any@protr@a
2127   \fi
2128 }
2129 \def\pxrr@takeout@any@protr@a{%
2130   \aftergroup\pxrr@any@protrtrue
2131 }
```

\pxrr@ruby@main@m モノルビ。

```
2132 \def\pxrr@ruby@main@m{%
2133   \pxrr@zip@list\pxrr@body@list\pxrr@ruby@list
2134   \let\pxrr@whole@list\pxrr@res
2135   \pxrr@check@par@head
2136   \pxrr@put@head@penalty
2137   \pxrr@any@protrfalse
2138   \ifpxrr@Debug
2139     \pxrr@debug@show@recomp
2140   \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。

```
2141   \let\pxrr@locate@head@\pxrr@locate@inner
2142   \let\pxrr@locate@end@\pxrr@locate@inner
2143   \let\pxrr@locate@sing@\pxrr@locate@inner
2144   \ifpxrr@aprotr\else
2145     \let\pxrr@locate@end@\pxrr@locate@end
2146     \let\pxrr@locate@sing@\pxrr@locate@end
2147   \fi
2148   \ifpxrr@bprotr\else
2149     \let\pxrr@locate@head@\pxrr@locate@head
2150     \let\pxrr@locate@sing@\pxrr@locate@head
2151   \fi
2152   \def\pxrr@pre##1##2{%
2153     \pxrr@if@last{%
```

単独ブロックの場合。

```
2154     \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
2155     \pxrr@intrude@head
2156     \unhbox\pxrr@boxr
2157     \pxrr@intrude@end
```

```

2158     \pxrr@takeout@any@protr
2159     }{%

```

先頭ブロックの場合。

```

2160     \pxrr@compose@block\pxrr@locate@head@{##1}{##2}%
2161     \pxrr@intrude@head
2162     \unhbox\pxrr@boxr
2163     }%
2164     }%
2165     \def\pxrr@inter##1##2{%
2166     \pxrr@if@last{%

```

末尾ブロックの場合。

```

2167     \pxrr@compose@block\pxrr@locate@end@{##1}{##2}%
2168     \pxrr@inter@mono
2169     \unhbox\pxrr@boxr
2170     \pxrr@intrude@end
2171     \pxrr@takeout@any@protr
2172     }{%

```

中間ブロックの場合。

```

2173     \pxrr@compose@block\pxrr@locate@inner{##1}{##2}%
2174     \pxrr@inter@mono
2175     \unhbox\pxrr@boxr
2176     }%
2177     }%
2178     \let\pxrr@post\@empty
2179     \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。

```

2180     \if j\pxrr@mode
2181     \ifpxrr@any@protr
2182     \pxrr@ruby@redo@j
2183     \fi
2184     \fi
2185     \unhbox\pxrr@boxr
2186     }

```

\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。

```

2187 \def\pxrr@ruby@redo@j{%
2188   \pxrr@concat@list\pxrr@body@list
2189   \let\pxrr@body@list\pxrr@res
2190   \pxrr@concat@list\pxrr@ruby@list
2191   \let\pxrr@ruby@list\pxrr@res
2192   \pxrr@zip@single\pxrr@body@list\pxrr@ruby@list
2193   \let\pxrr@whole@list\pxrr@res
2194 \ifpxrr@Debug
2195 \pxrr@debug@show@concat
2196 \fi

```

```

2197 \let\pxrr@locate@sing@\pxrr@locate@inner
2198 \ifpxrr@aprotr\else
2199   \let\pxrr@locate@sing@\pxrr@locate@end
2200 \fi
2201 \ifpxrr@bprotr\else
2202   \let\pxrr@locate@sing@\pxrr@locate@head
2203 \fi
2204 \def\pxrr@pre##1##2{%
2205   \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
2206   \pxrr@intrude@head
2207   \unhbox\pxrr@boxr
2208   \pxrr@intrude@end
2209 }%
2210 \let\pxrr@inter\@undefined
2211 \let\pxrr@post\@empty
2212 \setbox\pxrr@boxr\pxrr@hbox{\pxrr@whole@list}%
2213 }

```

\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。

```

2214 \def\pxrr@ruby@main@g{%
2215   \pxrr@zip@list\pxrr@body@list\pxrr@ruby@list
2216   \let\pxrr@whole@list\pxrr@res
2217   \pxrr@check@par@head
2218   \pxrr@put@head@penalty
2219 \ifpxrr@Debug
2220 \pxrr@debug@show@recomp
2221 \fi
2222 \let\pxrr@locate@sing@\pxrr@locate@inner
2223 \ifpxrr@aprotr\else
2224   \let\pxrr@locate@sing@\pxrr@locate@end
2225 \fi
2226 \ifpxrr@bprotr\else
2227   \let\pxrr@locate@sing@\pxrr@locate@head
2228 \fi
2229 \def\pxrr@pre##1##2{%
2230   \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
2231   \pxrr@intrude@head
2232   \unhbox\pxrr@boxr
2233   \pxrr@intrude@end
2234 }%
2235 \let\pxrr@inter\@undefined
2236 \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。

```

2237 \pxrr@whole@list
2238 }

```

\pxrr@ruby@main@tm 両側のモノルビの場合。

```
2239 \def\pxrr@ruby@main@tm{%
2240   \pxrr@tzip@list\pxrr@body@list\pxrr@ruby@list\pxrr@sruby@list
2241   \let\pxrr@whole@list\pxrr@res
2242   \pxrr@check@par@head
2243   \pxrr@any@protrfalse
2244   \ifpxrr@Debug
2245   \pxrr@debug@show@recomp
2246 \fi
2247   \let\pxrr@locate@head@\pxrr@locate@inner
2248   \let\pxrr@locate@end@\pxrr@locate@inner
2249   \let\pxrr@locate@sing@\pxrr@locate@inner
2250   \ifpxrr@aprotr\else
2251     \let\pxrr@locate@end@\pxrr@locate@end
2252     \let\pxrr@locate@sing@\pxrr@locate@end
2253   \fi
2254   \ifpxrr@bprotr\else
2255     \let\pxrr@locate@head@\pxrr@locate@head
2256     \let\pxrr@locate@sing@\pxrr@locate@head
2257   \fi
2258   \def\pxrr@pre##1##2##3{%
2259     \pxrr@if@last{%
2260       \pxrr@compose@twoside@block\pxrr@locate@sing@
2261       {##1}{##2}{##3}%
2262       \pxrr@intrude@head
2263       \unhbox\pxrr@boxr
2264       \pxrr@intrude@end
2265       \pxrr@takeout@any@protr
2266     }{%
2267       \pxrr@compose@twoside@block\pxrr@locate@head@
2268       {##1}{##2}{##3}%
2269       \pxrr@intrude@head
2270       \unhbox\pxrr@boxr
2271     }%
2272   }%
2273   \def\pxrr@inter##1##2##3{%
2274     \pxrr@if@last{%
2275       \pxrr@compose@twoside@block\pxrr@locate@end@
2276       {##1}{##2}{##3}%
2277       \pxrr@inter@mono
2278       \unhbox\pxrr@boxr
2279       \pxrr@intrude@end
2280       \pxrr@takeout@any@protr
2281     }{%
2282       \pxrr@compose@twoside@block\pxrr@locate@inner
2283       {##1}{##2}{##3}%
2284       \pxrr@inter@mono
2285       \unhbox\pxrr@boxr
```

```

2286     }%
2287 }%
2288 \let\pxrr@post\@empty
2289 \setbox\pxrr@boxr\pxrr@hbox{\pxrr@whole@list}%
2290 \unhbox\pxrr@boxr
2291 }

```

\pxrr@ruby@main@tg 両側の単純グループビの場合。

```

2292 \def\pxrr@ruby@main@tg{%
2293   \pxrr@check@par@head
2294   \pxrr@put@head@penalty
2295   \let\pxrr@locate@sing@\pxrr@locate@inner
2296   \ifpxrr@aprotr\else
2297     \let\pxrr@locate@sing@\pxrr@locate@end
2298   \fi
2299   \ifpxrr@bprotr\else
2300     \let\pxrr@locate@sing@\pxrr@locate@head
2301   \fi
2302   \expandafter\pxrr@compose@twoside@block\expandafter\pxrr@locate@sing@
2303   \pxrr@all@input
2304   \pxrr@intrude@head
2305   \unhbox\pxrr@boxr
2306   \pxrr@intrude@end
2307 }

```

\pxrr@ruby@main@mg 未実装（呼出もない）。

```

2308 \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。

```

2309 \newif\ifpxrr@ghost

```

\pxrr@jprologue 和文ルビ用の開始処理。

```

2310 \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。

```

2311   \ifpxrr@jghost
2312     \pxrr@jghost@char
2313     \pxrr@inhibitglue
2314   \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。

```

2315   \begingroup
2316     \pxrr@abodyfalse
2317     \pxrr@csletcs{\ifpxrr@ghost}{\ifpxrr@jghost}%

```

出力した全角空白の幅だけ戻しておく。

```
2318 \ifpxrr@jghost
2319 \setbox\pxrr@boxa\hbox{\pxrr@jghost@char}%
2320 \kern-\wd\pxrr@boxa
2321 \fi
2322 }
```

`\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 2.5pt 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 T<sub>E</sub>X 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。)

```
2323 \chardef\pxrr@aghostchar=23 % compwordmark
2324 \let\pxrr@aghost\relax
2325 \let\pxrr@aghostfont\relax
2326 \def\pxrr@setup@aghost{%
2327 \global\let\pxrr@setup@aghost\relax
2328 \IfFileExists{t1lmr.fd}{%
2329 \begingroup
2330 \fontsize{2.5}{0}\usefont{T1}{lmr}{m}{n}%
2331 \endgroup
2332 \global\pxrr@letcs\pxrr@aghostfont{T1/lmr/m/n/2.5}%
2333 \gdef\pxrr@aghost{\pxrr@aghostfont\pxrr@aghostchar}}%
2334 \global\xspcode\pxrr@aghostchar=3 %
2335 }{%else
2336 \pxrr@warn{Ghost embedding for \string\aruby\space
2337 is disabled,\MessageBreak
2338 since package lmodern is missing}%
2339 \global\pxrr@aghostfalse
2340 \global\let\pxrr@aghosttrue\relax
2341 }%
2342 }
```

`\pxrr@aprologue` 欧文ルビ用の開始処理。

```
2343 \def\pxrr@aprologue{%
2344 \ifpxrr@aghost
2345 \pxrr@aghost
2346 \fi
2347 \begingroup
2348 \pxrr@abodytrue
2349 \pxrr@csletcs{ifpxrr@ghost}{ifpxrr@aghost}%
2350 }
```

#### 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。

```
2351 \def\pxrr@ruby@exit{%
2352   \ifpxrr@fatal@error
2353     \pxrr@fallback
2354   \fi
2355   \ifpxrr@abody
2356     \expandafter\pxrr@aepilogue
2357   \else
2358     \expandafter\pxrr@jepilogue
2359   \fi
2360 }
```

`\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。

```
2361 \def\pxrr@jepilogue{%
2362   \ifpxrr@jghost
2363     \setbox\pxrr@boxa\hbox{\pxrr@jghost@char}%
2364     \kern-\wd\pxrr@boxa
2365   \fi

   \pxrr@?prologue の中の \begingroup で始まるグループを閉じる。
2366   \endgroup
2367   \ifpxrr@jghost
2368     \pxrr@inhibitglue
2369     \pxrr@jghost@char
2370   \fi
2371 }
```

`\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。

```
2372 \def\pxrr@aepilogue{%
2373   \endgroup
2374   \ifpxrr@aghost
2375     \pxrr@aghost
2376   \fi
2377 }
```

## 4.19 デバッグ用出力

```
2378 \def\pxrr@debug@show@input{%
2379   \typeout{---\pxrr@pkgname\space input:^^J%
2380   \ifpxrr@abody = \meaning\ifpxrr@abody^^J%
2381   \ifpxrr@truby = \meaning\ifpxrr@truby^^J%
2382   pxrr@ruby@fsize = \pxrr@ruby@fsize^^J%
2383   pxrr@body@zw = \pxrr@body@zw^^J%
2384   pxrr@ruby@zw = \pxrr@ruby@zw^^J%
2385   pxrr@iiskip = \pxrr@iiskip^^J%
2386   pxrr@iaiskip = \pxrr@iaiskip^^J%
2387   pxrr@htratio = \pxrr@htratio^^J%
```

```

2388 pxrr@ruby@raise = \pxrr@ruby@raise^^J%
2389 pxrr@ruby@lower = \pxrr@ruby@lower^^J%
2390 ifpxrr@bprotr = \meaning\ifpxrr@bprotr^^J%
2391 ifpxrr@aprotr = \meaning\ifpxrr@aprotr^^J%
2392 pxrr@side = \the\pxrr@side^^J%
2393 pxrr@evensp = \the\pxrr@evensp^^J%
2394 pxrr@fullsize = \the\pxrr@fullsize^^J%
2395 pxrr@bscomp = \meaning\pxrr@bscomp^^J%
2396 pxrr@ascomp = \meaning\pxrr@ascomp^^J%
2397 ifpxrr@bnoBr = \meaning\ifpxrr@bnoBr^^J%
2398 ifpxrr@anoBr = \meaning\ifpxrr@anoBr^^J%
2399 ifpxrr@bfintr = \meaning\ifpxrr@bfintr^^J%
2400 ifpxrr@afintr = \meaning\ifpxrr@afintr^^J%
2401 pxrr@bintr = \pxrr@bintr^^J%
2402 pxrr@aintr = \pxrr@aintr^^J%
2403 pxrr@athead = \the\pxrr@athead^^J%
2404 pxrr@mode = \meaning\pxrr@mode^^J%
2405 ifpxrr@athead@given = \meaning\ifpxrr@athead@given^^J%
2406 ifpxrr@mode@given = \meaning\ifpxrr@mode@given^^J%
2407 pxrr@body@list = \meaning\pxrr@body@list^^J%
2408 pxrr@body@count = \@nameuse{pxrr@body@count}^^J%
2409 pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
2410 pxrr@ruby@count = \@nameuse{pxrr@ruby@count}^^J%
2411 pxrr@end@kinsoku = \pxrr@end@kinsoku^^J%
2412 ----
2413 }%
2414 }
2415 \def\pxrr@debug@show@recomp{%
2416 \typeout{----\pxrr@pkgname\space recomp:^^J%
2417 pxrr@body@list = \meaning\pxrr@body@list^^J%
2418 pxrr@body@count = \pxrr@body@count^^J%
2419 pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
2420 pxrr@ruby@count = \pxrr@ruby@count^^J%
2421 pxrr@res = \meaning\pxrr@res^^J%
2422 ----
2423 }%
2424 }
2425 \def\pxrr@debug@show@concat{%
2426 \typeout{----\pxrr@pkgname\space concat:^^J%
2427 pxrr@body@list = \meaning\pxrr@body@list^^J%
2428 pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
2429 pxrr@whole@list = \meaning\pxrr@whole@list^^J%
2430 ----
2431 }%
2432 }
2433 \def\pxrr@debug@show@resolve@mode{%
2434 \typeout{----\pxrr@pkgname\space resolve-mode:
2435 \meaning\pxrr@mode}%
2436 }

```

## 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。

```
2437 \def\pxrr@warn@na@kmark#1{%  
2438   \pxrr@warn{Unavailable kenten mark '#1'}%  
2439 }
```

パラメタ設定命令で無効な値が指定された場合。

```
2440 \def\pxrr@err@invalid@value#1{%  
2441   \pxrr@error{Invalid value '#1'}%  
2442     {\@eha}%  
2443 }
```

### 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。

```
2444 \let\pxrr@k@ymark\@undefined
```

`\pxrr@k@ysmark` 横組の副の圏点マークのコード。

```
2445 \let\pxrr@k@ysmark\@undefined
```

`\pxrr@k@tmark` 縦組の主の圏点マークのコード。

```
2446 \let\pxrr@k@tmark\@undefined
```

`\pxrr@k@tsmark` 縦組の服の圏点マークのコード。

```
2447 \let\pxrr@k@tsmark\@undefined
```

圏点マークの初期値の設定。

```
2448 \AtEndOfPackage{%  
2449   \pxrr@k@get@mark\pxrr@k@ymark{bullet*}%  
2450   \pxrr@k@get@mark\pxrr@k@ysmark{sesame*}%  
2451   \pxrr@k@get@mark\pxrr@k@tmark{sesame*}%  
2452   \pxrr@k@get@mark\pxrr@k@tsmark{bullet*}%  
2453 }
```

`\pxrr@k@ruby@font` 圏点用フォント切替命令。

```
2454 \let\pxrr@k@ruby@font\@empty
```

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

```
2455 \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。

```
2456 \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。

```
2457 \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。

```
2458 \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。

```
2459 \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。

```
2460 \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。

```
2461 \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。

```
2462 \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。

```
2463 \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。

```
2464 \chardef\pxrr@k@full=0
```

`\pxrr@k@the@mark` 適用される圏点マークの命令。

```
2465 \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。

```
2466 \newif\ifpxrr@avail@UTF
```

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

```
2467 \def\pxrr@decide@avail@UTF{%
```

```
2468   \global\let\pxrr@decide@avail@UTF\relax
```

```
2469   \ifx\UTF\@undefined \global\pxrr@avail@UTFfalse
```

```
2470   \else \global\pxrr@avail@UTFtrue
```

```
2471   \fi
```

```
2472 }
```

### 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`

```
2473 \def\pxrr@k@decompose#1{%
2474   \let\pxrr@res\@empty
2475   \pxrr@cntr=\z@
2476   \pxrr@k@decompose@loopa#1\pxrr@end
2477 }
2478 \def\pxrr@k@decompose@loopa{%
2479   \futurelet\pxrr@token\pxrr@k@decompose@loopb
2480 }
2481 \def\pxrr@k@decompose@loopb{%
2482   \pxrr@cond\ifx\pxrr@token\pxrr@end\fi{%
2483     \pxrr@appto\pxrr@res{\pxrr@post}%
2484   }\pxrr@if@kspan@cmd\pxrr@token{%
2485     \pxrr@k@decompose@special\pxrr@k@decompose@kspan
2486   }\pxrr@if@ruby@cmd\pxrr@token{%
2487     \pxrr@k@decompose@special\pxrr@k@decompose@ruby
2488   }\pxrr@if@truby@cmd\pxrr@token{%
2489     \pxrr@k@decompose@special\pxrr@k@decompose@truby
2490   }\pxrr@if@kenten@cmd\pxrr@token{%
2491     \pxrr@k@decompose@special\pxrr@k@decompose@kenten
2492   }\pxrr@cond\ifx\pxrr@token\@sptoken\fi{%
2493     \pxrr@k@decompose@loope
2494   }{%
2495     \pxrr@setok{\pxrr@ifx{\pxrr@token\bgroup}}%
2496     \pxrr@k@decompose@loopc
2497   }}}}}%
2498 }
2499 \def\pxrr@k@decompose@loopc#1{%
2500   \pxrr@appto\pxrr@res{\pxrr@entry}%
2501   \ifpxrr@ok
2502     \pxrr@appto\pxrr@res{{\#1}}}%
2503   \else
2504     \pxrr@appto\pxrr@res{{\#1}}}%
2505   \fi
2506   \pxrr@k@decompose@loopd
2507 }
2508 \def\pxrr@k@decompose@loopd{%
2509   \advance\pxrr@cntr\@ne
2510   \pxrr@k@decompose@loopa
2511 }
2512 \expandafter\def\expandafter\pxrr@k@decompose@loope\space{%
2513   \pxrr@okfalse
```

```

2514 \pxrr@k@decompose@loopc{ }%
2515 }
2516 \def\pxrr@k@decompose@special#1#2#{%
2517   #1{#2}%
2518 }
2519 \def\pxrr@k@decompose@kspan#1#2{%
2520   \pxrr@appto\pxrr@res{\pxrr@entry@kspan{#1{#2}}}%
2521   \pxrr@k@decompose@loopd
2522 }
2523 \def\pxrr@k@decompose@ruby#1#2#3{%
2524   \pxrr@appto\pxrr@res{\pxrr@entry@ruby{#1{#2}{#3}}}%
2525   \pxrr@k@decompose@loopd
2526 }
2527 \def\pxrr@k@decompose@truby#1#2#3#4{%
2528   \pxrr@appto\pxrr@res{\pxrr@entry@ruby{#1{#2}{#3}{#4}}}%
2529   \pxrr@k@decompose@loopd
2530 }
2531 \def\pxrr@k@decompose@kenten#1#2{%
2532   \pxrr@appto\pxrr@res{\pxrr@entry@kenten{#1{#2}}}%
2533   \pxrr@k@decompose@loopd
2534 }
2535 \def\pxrr@cmd@ruby{\jruby}
2536 \def\pxrr@cmd@kenten{\jkenten}
2537 \def\pxrr@if@ruby@cmd#1{%
2538   \if \ifcat\noexpand#1\relax
2539     \ifx#1\pxrr@cmd@ruby T%
2540     \else\ifx#1\jruby T%
2541     \else\ifx#1\aruby T%
2542     \else F%
2543     \fi\fi\fi
2544   \else F%
2545   \fi T\expandafter\@firstoftwo
2546 \else \expandafter\@secondoftwo
2547 \fi
2548 }
2549 \def\pxrr@if@truby@cmd#1{%
2550   \if \ifcat\noexpand#1\relax
2551     \ifx#1\truby T%
2552     \else\ifx#1\atruby T%
2553     \else F%
2554     \fi\fi
2555   \else F%
2556   \fi T\expandafter\@firstoftwo
2557 \else \expandafter\@secondoftwo
2558 \fi
2559 }
2560 \def\pxrr@if@kspan@cmd#1{%
2561   \pxrr@cond\ifx#1\kspan\fi
2562 }

```

```

2563 \def\pxrr@if@kenten@cmd#1{%
2564   \if \ifcat\noexpand#1\relax
2565     \ifx#1\pxrr@cmd@kenten T%
2566     \else\ifx#1\j@kenten T%
2567     \else F%
2568     \fi\fi
2569   \else F%
2570   \fi T\expandafter\@firstoftwo
2571 \else \expandafter\@secondoftwo
2572 \fi
2573 }

```

## 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。

```

2574 \newcommand*\kentensetup[1]{%
2575   \pxrr@in@setuptrue
2576   \pxrr@fatal@errorfalse
2577   \pxrr@k@parse@option{#1}%
2578   \ifpxrr@fatal@error\else
2579     \let\pxrr@k@d@side\pxrr@side
2580     \let\pxrr@k@d@mark\pxrr@k@mark
2581     \let\pxrr@k@d@full\pxrr@k@full
2582   \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。

```

2583   \pxrr@in@setupfalse
2584 }

```

`\kentenfontsetup` 対応するパラメタを設定する。

```

2585 \newcommand*\kentenfontsetup{}
2586 \def\kentenfontsetup#1{%
2587   \def\pxrr@k@ruby@font
2588 }

```

`\kentensizeratio` 対応するパラメタを設定する。

```

2589 \newcommand*\kentensizeratio[1]{%
2590   \edef\pxrr@k@size@ratio{#1}%
2591 }

```

`\kentenintergap` 対応するパラメタを設定する。

```

2592 \newcommand*\kentenintergap[1]{%
2593   \edef\pxrr@k@inter@gap{#1}%
2594 }

```

`\kentenrubyintergap` 対応するパラメタを設定する。

```

2595 \newcommand*\kentenrubyintergap[1]{%

```

```

2596 \edef\pxrr@k@ruby@inter@gap{#1}%
2597 }

```

\kentenmarkinyoko 対応するパラメタを設定する。

```

\kentensubmarkinyoko 2598 \newcommand*\kentenmarkinyoko[1]{%
\kentenmarkintate 2599 \pxrr@k@get@mark\pxrr@k@ymark{#1}%
2600 }
\kentensubmarkintate 2601 \newcommand*\kentensubmarkinyoko[1]{%
2602 \pxrr@k@get@mark\pxrr@k@ysmark{#1}%
2603 }
2604 \newcommand*\kentenmarkintate[1]{%
2605 \pxrr@k@get@mark\pxrr@k@tmark{#1}%
2606 }
2607 \newcommand*\kentensubmarkintate[1]{%
2608 \pxrr@k@get@mark\pxrr@k@tsmark{#1}%
2609 }

```

\kentenrubycombination 対応するパラメタを設定する。

```

2610 \chardef\pxrr@k@ruby@combo@ruby=0
2611 \chardef\pxrr@k@ruby@combo@both=1
2612 \newcommand*\kentenrubycombination[1]{%
2613 \pxrr@letcs\pxrr@tempa{\pxrr@k@ruby@combo@#1}%
2614 \ifx\pxrr@tempa\relax
2615 \pxrr@err@invalid@value{#1}%
2616 \else
2617 \let\pxrr@k@ruby@combo\pxrr@tempa
2618 \fi
2619 }

```

## 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。

```

2620 \def\pxrr@k@declare@mark#1{%
2621 \global\@namedef{pxrr@k@mark@#1}%
2622 }

```

\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。

```

2623 \def\pxrr@k@let@mark#1{%
2624 \global\pxrr@cslet{pxrr@k@mark@#1}%
2625 }

```

\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。

```

2626 \def\pxrr@k@get@mark#1#2{%
2627 \futurelet\pxrr@token\pxrr@k@get@mark@a#2\pxrr@nil#1%
2628 }

```



```

2629 \def\pxrr@k@get@mark@a{%
2630   \pxrr@cond\ifcat A\noexpand\pxrr@token\fi{%
2631     \pxrr@k@get@mark@c
2632   }{%else
2633     \pxrr@k@get@mark@b
2634   }%
2635 }
2636 \def\pxrr@k@get@mark@b#1\pxrr@nil#2{%
2637   \def#2{#1}%
2638 }
2639 \def\pxrr@k@get@mark@c#1#2\pxrr@nil#3{%
2640   \ifnum'#1<128
2641     \pxrr@letcs\pxrr@tempa{\pxrr@k@mark@@#1#2}%
2642     \ifx\pxrr@tempa\relax
2643       \pxrr@warn@na@kmark{#1#2}%
2644     \else
2645       \let#3\pxrr@tempa
2646     \fi
2647   \else
2648     \pxrr@k@get@mark@b#1#2\pxrr@nil#3%
2649   \fi
2650 }

```

`\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。

```

2651 \def\pxrr@k@declare@mark@char#1#2{%
2652   \pxrr@k@declare@mark@char@a{#1}#2\pxrr@end
2653 }
2654 \def\pxrr@k@declare@mark@char@a#1#2:#3\pxrr@end{%
2655   \pxrr@jchardef\pxrr@tempa\pxrr@jc{#2:#3}%
2656   \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。

```

2657     \pxrr@k@declare@mark{#1}{\pxrr@k@char@UTF{#1}{#3}}%
2658   \else
2659     \pxrr@k@let@mark{#1}\pxrr@tempa
2660   \fi
2661 }

```

`\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。

```

2662 \def\pxrr@k@char@UTF#1#2{%
2663   \pxrr@decide@avail@UTF
2664   \ifpxrr@avail@UTF
2665     \pxrr@k@declare@mark{#1}{\UTF{#2}}%
2666     \UTF{#2}%
2667   \else
2668     \pxrr@k@let@mark{#1}\@empty

```

```

2669 \pxrr@warn@na@kmark{#1}%
2670 \fi
2671 }

標準サポートの圏点マークの定義。

2672 \pxrr@k@declare@mark@char{bullet} {2121:2022}
2673 \pxrr@k@declare@mark@char{triangle}{2225:25B2}
2674 \pxrr@k@declare@mark@char{Triangle}{2224:25B3}
2675 \pxrr@k@declare@mark@char{fisheye} {2121:25C9}
2676 \pxrr@k@declare@mark@char{Circle} {217B:25CB}
2677 \pxrr@k@declare@mark@char{bullseye}{217D:25CE}
2678 \pxrr@k@declare@mark@char{circle} {217C:25CF}
2679 \pxrr@k@declare@mark@char{Bullet} {2121:25E6}
2680 \pxrr@k@declare@mark@char{sesame} {2121:FE45}
2681 \pxrr@k@declare@mark@char{Sesame} {2121:FE46}
2682 \pxrr@jchardef\pxrr@ja@dot=\pxrr@jc{2126:30FB}
2683 \pxrr@jchardef\pxrr@ja@comma=\pxrr@jc{2122:3001}
2684 \pxrr@k@declare@mark{bullet*}-{%
2685 \pxrr@dima=\pxrr@ruby@zw\relax
2686 \hb@xt@\pxrr@dima{%
2687 \kern-.5\pxrr@dima
2688 \pxrr@if@in@tate{}{\lower.38\pxrr@dima}%
2689 \hbox{%
2690 \pxrr@dima=\f@size\p@
2691 \fontsize{2\pxrr@dima}{\z@}\selectfont
2692 \pxrr@ja@dot
2693 }%
2694 \hss
2695 }%
2696 }
2697 \pxrr@k@declare@mark{sesame*}-{%
2698 \pxrr@dima=\pxrr@ruby@zw\relax
2699 \hb@xt@\pxrr@dima{%
2700 \pxrr@if@in@tate{\kern.1\pxrr@dima}{\kern.05\pxrr@dima}%
2701 \pxrr@if@in@tate{\lower.85\pxrr@dima}{\raise.3\pxrr@dima}%
2702 \hbox{%
2703 \pxrr@dima=\f@size\p@
2704 \fontsize{2.4\pxrr@dima}{\z@}\selectfont
2705 \pxrr@ja@comma
2706 }%
2707 \hss
2708 }%
2709 }

```

## 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。

```

2710 \def\pxrr@k@parse@option#1{%
2711   \edef\pxrr@tempa{#1}%
2712   \let\pxrr@side\pxrr@k@d@side
2713   \let\pxrr@k@mark\pxrr@k@d@mark
2714   \let\pxrr@k@full\pxrr@k@d@full
2715   \expandafter\pxrr@k@parse@option@loop\pxrr@tempa @\pxrr@end
2716 }
2717 \def\pxrr@k@parse@option@loop#1{%

```

圈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。

```

2718   \pxrr@letcs\pxrr@tempa{pxrr@k@po@PR@#1}%
2719   \pxrr@cond\ifx\pxrr@tempa\relax\fi{%
2720     \pxrr@fatal@knx@letter{#1}%
2721     \pxrr@k@parse@option@exit
2722   }{%
2723     \pxrr@tempa
2724     \pxrr@k@parse@option@loop
2725   }%
2726 }
2727 \def\pxrr@k@parse@option@exit#1\pxrr@end{%
2728   \ifpxrr@in@setup\else
2729     \pxrr@k@check@option

```

ここで \pxrr@k@the@mark を適切に定義する。

```

2730   \pxrr@if@in@tate{%
2731     \ifcase\pxrr@k@mark \let\pxrr@k@the@mark\pxrr@k@tmark
2732     \or \let\pxrr@k@the@mark\pxrr@k@tsmark
2733     \fi
2734   }{%
2735     \ifcase\pxrr@k@mark \let\pxrr@k@the@mark\pxrr@k@ymark
2736     \or \let\pxrr@k@the@mark\pxrr@k@ysmark
2737     \fi
2738   }%
2739   \fi
2740 }
2741 \def\pxrr@k@po@PR@{%
2742   \pxrr@k@parse@option@exit
2743 }
2744 \def\pxrr@k@po@PR@P{%
2745   \chardef\pxrr@side\z@
2746 }
2747 \def\pxrr@k@po@PR@S{%
2748   \chardef\pxrr@side\@ne
2749 }
2750 \def\pxrr@k@po@PR@p{%
2751   \chardef\pxrr@k@mark\z@
2752 }
2753 \def\pxrr@k@po@PR@s{%
2754   \chardef\pxrr@k@mark\@ne

```

```

2755 }
2756 \def\pxrr@k@po@PR@F{%
2757   \chardef\pxrr@k@full\z@
2758 }
2759 \def\pxrr@k@po@PR@f{%
2760   \chardef\pxrr@k@full\@ne
2761 }

```

## 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。

```

2762 \def\pxrr@k@check@option{%
2763 }

```

## 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。

```

2764 \def\pxrr@k@compose@block#1#2{%
2765   \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。

```

2766   \pxrr@k@make@rep@list{\copy\pxrr@boxb}{#2}%
2767   \let\pxrr@natwd\pxrr@zeropt
2768   \pxrr@evenspace@int\pxrr@locate@inner\pxrr@boxr
2769     \relax{\wd\pxrr@boxa}%
2770   \setbox\z@\hbox{%
2771     \ifnum\pxrr@side=\z@
2772       \raise\pxrr@ruby@raise\box\pxrr@boxr
2773     \else
2774       \lower\pxrr@ruby@lower\box\pxrr@boxr
2775     \fi
2776   }%
2777   \ht\z@\z@ \dp\z@\z@
2778   \@tempdima\wd\z@
2779   \setbox\pxrr@boxr\hbox{%
2780     \box\z@
2781     \kern-\@tempdima
2782     \box\pxrr@boxa
2783   }%
2784 }

```

`\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。

```

2785 \def\pxrr@k@make@rep@list#1#2{%
2786   \def\pxrr@res{\pxrr@pre{#1}}%
2787   \pxrr@cntr=#2\relax
2788   \ifnum\pxrr@cntr>\@ne
2789     \@tempcnta\pxrr@cntr \advance\@tempcnta\m@ne
2790     \@whilenum{\@tempcnta>\z@}\do{%
2791       \pxrr@appto\pxrr@res{\pxrr@inter{#1}}%
2792       \advance\@tempcnta\m@ne
2793     }%
2794   \fi
2795   \pxrr@appto\pxrr@res{\pxrr@post}%
2796 }
```

## 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。

```
2797 \newif\ifpxrr@k@first@entry
```

`\ifpxrr@k@last@entry` 末尾の項目であるか。

```
2798 \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。

```
2799 \newif\ifpxrr@k@prev@is@block
```

`\pxrr@k@accum@res` 累積の直接出力。

```
2800 \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。

```
2801 \mathchardef\pxrr@k@prebreakpenalty\z@
```

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

```
2802 \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。

```
2803 \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。

```
2804 \def\pxrr@k@list@pref{%
2805   \pxrr@k@first@entrytrue
2806   \pxrr@k@last@entryfalse
2807   \pxrr@k@prev@is@blockfalse
2808   \let\pxrr@k@accum@res\@empty
2809   \chardef\pxrr@k@block@seq@state\z@
2810 }
```

`\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。

```
2811 \def\pxrr@k@entry@with#1#2{%
2812   \pxrr@if@last{%
2813     \pxrr@k@last@entrytrue
2814     \pxrr@k@entry@with@a#1{#2}%
2815   }{%
2816     \pxrr@k@entry@with@a#1{#2}%
2817   }%
2818 }
2819 \def\pxrr@k@entry@with@a#1#2{%
2820   \mathchardef\pxrr@k@prebreakpenalty\z@
2821   \mathchardef\pxrr@k@postbreakpenalty\z@
```

下請けマクロを実行して結果を得る。

```
2822   #1{#2}%
2823   %\typeout{%
2824   %first=\meaning\ifpxrr@k@first@entry^^J%
2825   %last=\meaning\ifpxrr@k@last@entry^^J%
2826   %prev=\meaning\ifpxrr@k@prev@is@block^^J%
2827   %res=\meaning\pxrr@res^^J%
2828   %type=\meaning\pxrr@k@entry@res@type^^J%
2829   %prepen=\the\pxrr@k@prebreakpenalty^^J%
2830   %postpen=\the\pxrr@k@postbreakpenalty}%
```

累積直接出力の処理。

```
2831   \ifnum\pxrr@k@entry@res@type=\z@
2832     \expandafter\pxrr@appto\expandafter\pxrr@k@accum@res
2833     \expandafter{\pxrr@res}%
2834   \else
2835     \pxrr@k@accum@res
2836     \let\pxrr@k@accum@res\@empty
2837   \fi
```

前禁則ペナルティを入れる。

```
2838 \ifnum\pxrr@k@prebreakpenalty>\z@
2839 \@tempcntb\lastpenalty \unpenalty
2840 \advance\@tempcntb\pxrr@k@prebreakpenalty
2841 \penalty\@tempcntb
2842 \fi
```

圏点ブロックが連続する場合は和文間空白を入れる。

```
2843 \ifnum\pxrr@k@entry@res@type=\tw@
2844 \ifpxrr@k@prev@is@block
2845 \pxrr@inter@mono
2846 \fi
2847 \pxrr@k@prev@is@blocktrue
2848 \else
2849 \pxrr@k@prev@is@blockfalse
2850 \fi
```

ボックスの結果を実際に出力する。

```
2851 \ifnum\pxrr@k@entry@res@type>\z@
2852 \unhbox\pxrr@boxr
2853 \fi
```

後禁則ペナルティを入れる。

```
2854 \ifnum\pxrr@k@postbreakpenalty>\z@
2855 \penalty\pxrr@k@postbreakpenalty
2856 \fi
```

次の項目に進む。

```
2857 \pxrr@k@first@entryfalse
2858 }
```

`\pxrr@k@list@post` 圏点項目リストの出力の最後に行う処理。

```
2859 \def\pxrr@k@list@post{%
2860 \pxrr@k@accum@res
2861 \let\pxrr@k@accum@res\@empty
2862 }
```

`\pxrr@k@enten@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。

```
2863 \def\pxrr@k@enten@entry{%
2864 \pxrr@k@entry@with\pxrr@k@enten@entry@
2865 }
2866 \def\pxrr@k@enten@entry@#1{%
2867 \pxrr@k@check@char{#1}%
2868 \ifpxrr@ok
2869 \pxrr@k@compose@block{#1}\@ne
2870 \chardef\pxrr@k@entry@res@type=\tw@
2871 \else
2872 \def\pxrr@res{#1}%
2873 \chardef\pxrr@k@entry@res@type=\z@
2874 \fi
```

2875 }

`\pxrr@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。

```
2876 \def\pxrr@kenten@entry@kspan{%
2877   \pxrr@k@entry@with\pxrr@kenten@entry@kspan@
2878 }
2879 \def\pxrr@kenten@entry@kspan@#1{%
2880   \pxrr@kenten@entry@kspan@a#1%
2881 }
2882 \def\pxrr@kenten@entry@kspan@a#1{%
```

`\kspan (= #1)` が \* 付かを調べる。

```
2883   \@ifstar{%
2884     \@testopt\pxrr@kenten@entry@kspan@c{}%
2885   }{%
2886     \@testopt\pxrr@kenten@entry@kspan@b{}%
2887   }%
2888 }
2889 \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。

```
2890   \setbox\z@\pxrr@hbox{#2}%
2891   \@tempdima\pxrr@body@zw\relax
2892   \@tempdimb\wd\z@ \advance\@tempdimb.25\@tempdima
2893   \divide\@tempdimb\@tempdima
2894   \edef\pxrr@kenten@entry@tempa{\number\@tempdimb}%
2895   \pxrr@k@compose@block{#2}\pxrr@kenten@entry@tempa
2896   \chardef\pxrr@k@entry@res@type=\tw@
2897 }
2898 \def\pxrr@kenten@entry@kspan@c[#1]#2{%
```

`\kspan*` となっている場合。この時は圏点を付加せず直接出力する。

```
2899   \def\pxrr@res{#2}%
2900   \chardef\pxrr@k@entry@res@type=\z@
2901 }
```

`\pxrr@kenten@entry@kenten` ネストした `\kenten` 命令の圏点項目。単純にその `\kenten` を実行したものを出力とする。すなわち、内側の圏点の設定のみが生きる。

```
2902 \def\pxrr@kenten@entry@kenten{%
2903   \pxrr@k@entry@with\pxrr@kenten@entry@kenten@
2904 }
2905 \def\pxrr@kenten@entry@kenten@#1{%
```

この場合は圏点ブロックとは見なさないことに注意。

```
2906   \setbox\pxrr@boxr\hbox{#1}%
2907   \chardef\pxrr@k@entry@res@type=\@ne
2908 }
```

`\pxrr@kenten@entry@ruby` ルビ命令の圏点項目。



```

2909 \def\pxrr@kenten@entry@ruby{%
2910   \pxrr@k@entry@with\pxrr@kenten@entry@ruby@
2911 }
2912 \def\pxrr@kenten@entry@ruby@#1{%
2913   \pxrr@apply@combotrue
2914   \setbox\pxrr@boxr\hbox{#1}%
2915   \chardef\pxrr@k@entry@res@type=\@ne
2916 }

```

### 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。

```

2917 \newcommand*\kspan{%
2918   \@ifstar{%
2919     \@testopt\pxrr@kspan@a{}}%
2920   }{%
2921     \@testopt\pxrr@kspan@a{}}%
2922   }%
2923 }
2924 \pxrr@add@protect\kspan
2925 \def\pxrr@kspan@a[#1]#2{%
2926   \begingroup
2927     #2%
2928   \endgroup
2929 }

```

## 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）。

```

2930 \def\pxrr@k@check@char#1{%
2931   \futurelet\pxrr@token\pxrr@k@check@char@a#1\pxrr@end
2932 }
2933 \def\pxrr@k@check@char@a#1\pxrr@end{%
2934   \pxrr@cond\ifx\pxrr@token\bgroup\fi{%

```

グループには圏点を付ける。

```

2935     \pxrr@oktrue
2936   }{\pxrr@cond\ifx\pxrr@token\@sptoken\fi{%

```

欧文空白には圏点を付けない。

```

2937     \pxrr@okfalse
2938   }{%

```

```

2939 \pxrr@check@char\pxrr@token
2940 \ifcase\pxrr@cctr

```

通常文字でないので圏点を付けない。

```

2941 \pxrr@okfalse
2942 \or

```

欧文の通常文字。圏点を付ける。

```

2943 \pxrr@oktrue
2944 \chardef\pxrr@check@char@temp\z@
2945 \or

```

和文の通常文字。圏点を付ける。

```

2946 \pxrr@oktrue
2947 \chardef\pxrr@check@char@temp\@ne
2948 \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。

```

2949 \ifnum\pxrr@k@full=\z@\ifpxrr@ok
2950 \pxrr@check@punct@char{'#1}\pxrr@check@char@temp
2951 \ifpxrr@ok \pxrr@okfalse
2952 \else \pxrr@oktrue
2953 \fi
2954 \fi\fi
2955 \ifpxrr@ok
2956 \pxrr@get@prebreakpenalty\@tempcnta{'#1}%
2957 \mathchardef\pxrr@k@prebreakpenalty\@tempcnta
2958 \pxrr@get@postbreakpenalty\@tempcnta{'#1}%
2959 \mathchardef\pxrr@k@postbreakpenalty\@tempcnta
2960 \fi
2961 }}%
2962 }

```

## 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。

```

2963 \AtBeginDocument{%
2964 \providecommand*\kenten{}\jkenten}%
2965 }
2966 \newcommand*\jkenten{%
2967 \pxrr@k@prologue
2968 \pxrr@kenten
2969 }
2970 \pxrr@add@protect\jkenten

```

`\pxrr@kenten` オプションの処理を行う。

```

2971 \def\pxrr@kenten{%
2972   \@testopt\pxrr@kenten@a{}%
2973 }
2974 \def\pxrr@kenten@a[#1]{%
2975   \def\pxrr@option{#1}%
2976   \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。

```

2977     \expandafter\@firstofone
2978   \else
2979     \expandafter\pxrr@kenten@proc
2980   \fi
2981 }

```

`\pxrr@k@bind@param` “呼出時変数” へのコピーを行う。

```

2982 \def\pxrr@k@bind@param{%
2983   \let\pxrr@c@ruby@font\pxrr@k@ruby@font
2984   \let\pxrr@c@size@ratio\pxrr@k@size@ratio
2985   \let\pxrr@c@inter@gap\pxrr@k@inter@gap
2986 }

```

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

```

2987 \def\pxrr@kenten@proc#1{%
2988   \pxrr@prepare@fallback{#1}%
2989   \pxrr@k@bind@param
2990   \pxrr@assign@fsize
2991   \pxrr@k@parse@option\pxrr@option
2992   \pxrr@if@alive{%
2993     \pxrr@k@decompose{#1}%
2994     \let\pxrr@body@list\pxrr@res
2995     \pxrr@kenten@main
2996   }%
2997   \pxrr@kenten@exit
2998 }

```

### 5.11.2 組版処理

`\pxrr@kenten@main` 圏点の組版処理。

```

2999 \def\pxrr@kenten@main{%
3000   \setbox\pxrr@boxb\pxrr@hbox@to\z@{%
3001     \pxrr@use@ruby@font
3002     \hss\pxrr@k@the@mark\hss
3003   }%
3004   \let\pxrr@entry\pxrr@kenten@entry
3005   \let\pxrr@entry@kspan\pxrr@kenten@entry@kspan
3006   \let\pxrr@entry@ruby\pxrr@kenten@entry@ruby
3007   \let\pxrr@entry@kenten\pxrr@kenten@entry@kenten
3008   \let\pxrr@post\pxrr@k@list@post

```

```

3009 \pxrr@k@list@pre
3010 \pxrr@body@list
3011 }

```

### 5.11.3 前処理

\pxrr@jprologue 圏点用の開始処理。

```

3012 \def\pxrr@k@prologue{%
3013 \ifpxrr@k@ghost
3014 \pxrr@jghost@char
3015 \pxrr@inhibitglue
3016 \fi
3017 \begingroup
3018 \ifpxrr@k@ghost
3019 \setbox\pxrr@boxa\hbox{\pxrr@jghost@char}%
3020 \kern-\wd\pxrr@boxa
3021 \fi
3022 }

```

### 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。

```

3023 \def\pxrr@kenten@exit{%
3024 \ifpxrr@fatal@error
3025 \pxrr@fallback
3026 \fi
3027 \pxrr@k@epilogue
3028 }

```

\pxrr@jepilogue 終了処理。

```

3029 \def\pxrr@k@epilogue{%
3030 \ifpxrr@k@ghost
3031 \setbox\pxrr@boxa\hbox{\pxrr@jghost@char}%
3032 \kern-\wd\pxrr@boxa
3033 \fi
3034 \endgroup
3035 \ifpxrr@k@ghost
3036 \pxrr@inhibitglue
3037 \pxrr@jghost@char
3038 \fi
3039 }

```

## 5.12 デバッグ用出力

```

3040 \def\pxrr@debug@show@kenten@input{%
3041 \typeout{%
3042 pxrr@k@the@mark=\meaning\pxrr@k@the@mark^^J%

```

```

3043 pxrr@side=\meaning\pxrr@side^^J%
3044 pxrr@body@list=\meaning\pxrr@body@list^^J%
3045 }%
3046 }

```

## 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。
3047 \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。
3048 \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 3049 \let\pxrr@ck@ruby@font\relax
\pxrr@ck@inter@gap 3050 \let\pxrr@ck@size@ratio\relax
\pxrr@ck@ruby@inter@gap 3051 \let\pxrr@ck@inter@gap\relax
\pxrr@ck@side 3052 \let\pxrr@ck@ruby@inter@gap\relax
\pxrr@ck@the@mark 3053 \let\pxrr@ck@side\relax
\pxrr@ck@ruby@combo 3054 \let\pxrr@ck@the@mark\relax
\ifpxrr@ck@kenten@head 3055 \let\pxrr@ck@ruby@combo\relax
\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。
3056 \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。
3057 \newif\ifpxrr@ck@kenten@end

\pxrr@ck@bind@param “呼出時変数” へのコピーを行う。
3058 \def\pxrr@ck@bind@param{%
3059 \let\pxrr@ck@ruby@font\pxrr@c@ruby@font
3060 \let\pxrr@ck@size@ratio\pxrr@c@size@ratio
3061 \let\pxrr@ck@inter@gap\pxrr@c@inter@gap
3062 \let\pxrr@ck@ruby@inter@gap\pxrr@k@ruby@inter@gap
3063 \let\pxrr@ck@side\pxrr@side
3064 \let\pxrr@ck@the@mark\pxrr@k@the@mark
3065 \let\pxrr@ck@ruby@combo\pxrr@k@ruby@combo
3066 \pxrr@csletcs{ifpxrr@ck@kenten@head}{ifpxrr@k@first@entry}%
3067 \pxrr@csletcs{ifpxrr@ck@kenten@end}{ifpxrr@k@last@entry}%
3068 }

```

### 6.2 その他の変数

\pxrr@ck@zw 圏点の全角幅。

```
3069 \let\pxrr@ck@zw\relax
```

`\pxrr@ck@raise@P` ルビ側が P である場合の、圈点の垂直方向の移動量。

※ 圈点側が S である場合は負値になる。

```
3070 \let\pxrr@ck@raise@P\relax
```

`\pxrr@ck@raise@S` ルビ側が S である場合の、圈点の垂直方向の移動量。

```
3071 \let\pxrr@ck@raise@S\relax
```

`\pxrr@ck@raise@t` ルビ側が両側ルビである場合の、圈点の垂直方向の移動量。

```
3072 \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。

```
3073 \def\pxrr@ck@check@option{%
3074   \ifpxrr@ck@kenten@head
3075     \let\pxrr@bintr@\@empty
3076     \let\pxrr@bscomp=. \relax
3077     \pxrr@bno brtrue
3078   \fi
3079   \ifpxrr@ck@kenten@end
3080     \let\pxrr@aintr@\@empty
3081     \let\pxrr@ascomp=. \relax
3082     \pxrr@ano brtrue
3083   \fi
3084 }
```

### 6.4 フォントサイズ

`\pxrr@ck@assign@fsize` フォントに関連する設定。

```
3085 \def\pxrr@ck@assign@fsize{%
  \pxrr@ck@zw の値を求める。
3086   \begingroup
3087     \@tempdima=\f@size\p@
3088     \@tempdima\pxrr@ck@size@ratio\@tempdima
3089     \edef\pxrr@ruby@fsize{\the\@tempdima}%
3090     \let\pxrr@c@ruby@font\pxrr@ck@ruby@font
3091     \pxrr@use@ruby@font
3092     \pxrr@get@zwidth\pxrr@ck@zw
3093     \global\let\pxrr@gtempa\pxrr@ck@zw
3094   \endgroup
3095   \let\pxrr@ck@zw\pxrr@gtempa
  \pxrr@ck@raise@P、\pxrr@ck@raise@S の値を計算する。
3096   \ifcase\pxrr@ck@side
```

圏点側が P の場合。

```
3097 \@tempdimc\pxrr@ck@zw
3098 \advance\@tempdimc-\pxrr@htratio\@tempdimc
3099 \@tempdima\pxrr@ruby@raise\relax
3100 \@tempdimb\pxrr@ruby@zw\relax
3101 \advance\@tempdima\pxrr@htratio\@tempdimb
3102 \@tempdimb\pxrr@body@zw\relax
3103 \advance\@tempdima\pxrr@ck@ruby@inter@gap\@tempdimb
3104 \advance\@tempdima\@tempdimc
3105 \edef\pxrr@ck@raise@P{\the\@tempdima}%
3106 \@tempdima\pxrr@body@zw\relax
3107 \@tempdima\pxrr@htratio\@tempdima
3108 \@tempdimb\pxrr@body@zw\relax
3109 \advance\@tempdima\pxrr@ck@inter@gap\@tempdimb
3110 \advance\@tempdima\@tempdimc
3111 \edef\pxrr@ck@raise@S{\the\@tempdima}%
3112 \let\pxrr@ck@raise@t\pxrr@ck@raise@P
3113 \or
```

圏点側が S の場合。

```
3114 \@tempdimc\pxrr@ck@zw
3115 \@tempdimc\pxrr@htratio\@tempdimc
3116 \@tempdima-\pxrr@ruby@lower\relax
3117 \@tempdimb\pxrr@ruby@zw\relax
3118 \advance\@tempdimb-\pxrr@htratio\@tempdimb
3119 \advance\@tempdima-\@tempdimb
3120 \@tempdimb\pxrr@body@zw\relax
3121 \advance\@tempdima-\pxrr@ck@ruby@inter@gap\@tempdimb
3122 \advance\@tempdima-\@tempdimc
3123 \edef\pxrr@ck@raise@S{\the\@tempdima}%
3124 \@tempdima-\pxrr@body@zw\relax
3125 \advance\@tempdima-\pxrr@htratio\@tempdima
3126 \@tempdimb\pxrr@body@zw\relax
3127 \advance\@tempdima-\pxrr@ck@inter@gap\@tempdimb
3128 \advance\@tempdima-\@tempdimc
3129 \edef\pxrr@ck@raise@P{\the\@tempdima}%
3130 \let\pxrr@ck@raise@t\pxrr@ck@raise@S
3131 \fi
3132 }
```

## 6.5 ブロック毎の組版

\pxrr@ck@body@natwd 親文字列の自然長。

```
3133 \let\pxrr@ck@body@natwd\relax
```

\pxrr@ck@locate 圏点列のパターン指定。

```
3134 \let\pxrr@ck@locate\relax
```

`\pxrr@ck@kenten@list` 圏点列のリスト。

```
3135 \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。

```
3136 \def\pxrr@ck@compose#1{%
```

圏点を組んだボックスを作る。

```
3137 \setbox\tw@\pxrr@hbox@to\z{%
3138 \tempdima=f@size\p@
3139 \tempdima\pxrr@ck@size@ratio\tempdima
3140 \edef\pxrr@ruby@fsize{\the\tempdima}%
3141 \let\pxrr@c@ruby@font\pxrr@ck@ruby@font
3142 \pxrr@use@ruby@font
3143 \hss\pxrr@ck@the@mark\hss
3144 }%
```

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。

```
3145 \pxrr@save@listproc
3146 \pxrr@decompose{#1}%
3147 \def\pxrr@pre{%
3148 \let\pxrr@res\@empty
3149 \pxrr@ck@compose@entry\pxrr@pre
3150 }%
3151 \def\pxrr@inter{%
3152 \pxrr@ck@compose@entry\pxrr@inter
3153 }%
3154 \def\pxrr@post{%
3155 \pxrr@appto\pxrr@res{\pxrr@post}%
3156 }%
3157 \pxrr@res
3158 \pxrr@restore@listproc
3159 \let\pxrr@natwd\pxrr@ck@body@natwd
```

圏点リストを均等配置する。

```
3160 \pxrr@evenspace@int\pxrr@ck@locate\pxrr@boxb\relax
3161 {\wd\pxrr@boxa}%
```

合成処理。

```
3162 \setbox\z@\hbox{%
3163 \unhcopy\z@
3164 \kern-\wd\z@
3165 \ifcase\pxrr@side
3166 \raise\pxrr@ck@raise@P
3167 \or
3168 \raise\pxrr@ck@raise@S
3169 \or
3170 \raise\pxrr@ck@raise@t
3171 \fi
```



```

3172 \hb@xt@\wd\pxrr@boxa{\hss\copy\pxrr@boxb\hss}%
3173 }%
3174 }
3175 \def\pxrr@ck@compose@entry#1#2{%
3176 \setbox\pxrr@boxb\pxrr@hbox{#2}%
3177 \edef\pxrr@tempa{%
3178 \noexpand\pxrr@appto\noexpand\pxrr@res{\noexpand#1%
3179 \hb@xt@\the\wd\pxrr@boxb{\hss\copy\tw@hss}}}%
3180 }\pxrr@tempa
3181 }

```

## 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。

```

3182 \def\pxrr@dumb@sub#1#2#{#1}

```

\pxrr@dumb@ruby 無難なルビ命令。

```

3183 \def\pxrr@dumb@ruby{%
3184 \pxrr@dumb@sub\pxrr@dumb@ruby@
3185 }
3186 \def\pxrr@dumb@ruby@#1#2#{#1}

```

\pxrr@dumb@truby 無難な両側ルビ命令。

```

3187 \def\pxrr@dumb@truby{%
3188 \pxrr@dumb@sub\pxrr@dumb@truby@
3189 }
3190 \def\pxrr@dumb@truby@#1#2#3#{#1}

```

\pxrr@dumb@tkenten 無難な圏点命令。

※ \kspan もこの定義を利用する。

```

3191 \def\pxrr@dumb@kenten{%
3192 \pxrr@dumb@sub\pxrr@dumb@kenten@
3193 }
3194 \def\pxrr@dumb@kenten@#1{#1}

```

hyperref の \pdfstringdef 用のフック \pdfstringdefPreHook に上書き処理を追記する。

```

3195 \providecommand*\pdfstringdefPreHook{}
3196 \g@addto@macro\pdfstringdefPreHook{%

```

\ruby と \kenten は「本パッケージの命令であるか」のチェックが必要。

```

3197 \ifx\pxrr@cmd@ruby\ruby
3198 \let\ruby\pxrr@dumb@ruby
3199 \fi
3200 \let\jruby\pxrr@dumb@ruby
3201 \let\aruby\pxrr@dumb@ruby

```

```
3202 \let\truby\pxrr@dumb@truby
3203 \let\atruby\pxrr@dumb@truby
3204 \ifx\pxrr@cmd@kenten\kenten
3205   \let\kenten\pxrr@dumb@kenten
3206 \fi
3207 \let\kspan\pxrr@dumb@kenten
3208 }
```