

pxrubrica パッケージ

八登 崇之 (Takayuki YATO; aka “ZR”)

v1.3d [2021/03/06]

概要

JIS 規格「JIS X 4051」および W3C 技術ノート「日本語組版処理の要件」で述べられているような、日本において一般的な様式に従ってルビおよび圏点を付ける機能を提供する。

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	2
2.3	ルビ命令の入力文字列の入力規則	4
2.4	ルビ文字列のグループの指定	4
2.5	ゴースト処理	5
2.6	パラメタ設定命令	6
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の入力規則	8
3.3	ゴースト処理	9
3.4	パラメタ設定命令	9
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	13
4.3.1	全般設定	13
4.3.2	呼出時パラメタ・変数	15
4.4	その他の変数	16
4.5	補助手続	17
4.5.1	雑多な定義	17

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	26
4.7	パラメタ設定公開命令	37
4.8	ルビオプション解析	40
4.9	オプション整合性検査	46
4.10	フォントサイズ	48
4.11	ルビ用均等割り	50
4.12	小書き仮名の変換	53
4.13	ブロック毎の組版	54
4.14	命令の頑強化	61
4.15	致命的エラー対策	62
4.16	先読み処理	62
4.17	進入処理	64
4.17.1	前側進入処理	65
4.17.2	後側進入処理	66
4.18	メインです	68
4.18.1	エントリーポイント	68
4.18.2	入力検査	73
4.18.3	ルビ組版処理	75
4.18.4	前処理	80
4.18.5	後処理	81
4.19	デバッグ用出力	82
5	実装 (圏点関連)	83
5.1	エラーメッセージ	83
5.2	パラメタ	84
5.2.1	全般設定	84
5.2.2	呼出時の設定	85
5.3	補助手続	85
5.3.1	\UTF 命令対応	85
5.3.2	リスト分解	85
5.4	パラメタ設定公開命令	88
5.5	圏点文字	89
5.6	圏点オプション解析	91
5.7	オプション整合性検査	93
5.8	ブロック毎の組版	93
5.9	圏点項目	94
5.9.1	\kspan 命令	98
5.10	自動抑止の検査	98

5.11	メインです	99
5.11.1	エントリーポイント	99
5.11.2	組版処理	100
5.11.3	前処理	101
5.11.4	後処理	101
5.12	デバッグ用出力	101
6	実装 (圏点ルビ同時付加)	102
6.1	呼出時パラメタ	102
6.2	その他の変数	102
6.3	オプション整合性検査	103
6.4	フォントサイズ	103
6.5	ブロック毎の組版	104
7	実装 : hyperref 対策	106

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出ること。
- 進入：ルビ文字出力が親文字に隣接する文字の領域（水平方向に見た場合）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。
- クラスタ：入力文字列中の { } で囲った部分のこと。^{*1}
- 《文字》：均等割りにおいて不可分となる単位のこと。本来の意味での文字の他、クラスタも《文字》と扱われる。
- 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。

^{*1} 本来の L^AT_EX の用語では「グループ」と呼ぶが、ここでは「グループ」が別の意味をもつので別の用語を当てた。

次の用語については、『日本語組版の要件』*2 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、圏点

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（複数指定可）。

`:` 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 `pIATeX` 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（複数指定可）。

<code>-</code>	（無指定）	<code>P</code> (<code>< primary</code>)	上側配置
<code>c</code> (<code>< center</code>)	中付き	<code>S</code> (<code>< secondary</code>)	下側配置
<code>h</code> (<code>< head</code>)	肩付き	<code>e</code> (<code>< 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>< mono</code>)	モノルビ	<code>f</code> (<code>< full-size</code>)	小書き文字変換有効
<code>g</code> (<code>< group</code>)	グループルビ	<code>F</code>	小書き文字変換無効
<code>j</code> (<code>< jukugo</code>)	熟語ルビ		
<code>M</code>	自動切替モノルビ		
<code>J</code>	自動切替熟語ルビ		

– 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。

*2 <http://www.w3.org/TR/jlreq/ja/>

- 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（あいうえおつやゆよわ、およびその片仮名）を対応の非小書き仮名に変換する。F 指定はこの機能を無効にする。
- M および J の指定は「グループルビとモノ・熟語ルビの間で自動的に切り替える」設定である。具体的には、ルビのグループが 1 つしかない場合は g、複数ある場合は m および j と等価になる。

(後補助設定) は以下の値の何れか（複数指定可）。

- : 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_{LA}T_EX の標準の動作に従うのが原則だが、直後にあるものが文字でない場合、正しく動作しない（禁則が破れる）可能性がある。従って、不適切な行分割が起こりうる場合は適宜 * を指定する必要がある（なお、段落末尾で * を指定してはならない）。
- ! 無指定の場合、段落末尾では進入が抑止される。
- ゴースト処理が有効の場合はこの設定は無視される。

(後進入設定) は以下の値。

|| 後突出禁止 > 後進入大
| 後進入無し) 後進入小

- `\jruby` [`<オプション>`] {`<親文字>`} {`<ルビ文字>`}
- `\ruby` 命令の別名。`\ruby` という命令名は他のパッケージとの衝突の可能性があるので、L_AT_EX 文書の本文開始時 (`\begin{document}`) に未定義である場合のみ定義される。これに対して `\jruby` は常に定義される。なお、`\ruby` 以外の命令 (`\jruby` を含む) が定義済であった (命令名が衝突した) 場合にはエラーとなる。
- `\aruby` [`<オプション>`] {`<親文字>`} {`<ルビ文字>`}
- 欧文ルビの命令。すなわち、欧文文字列の上側（横組）／右側（縦組）にルビを付す。欧文ルビは和文ルビと比べて以下の点が異なる。
 - 常にグループルビと扱われる。(m、g、j の指定は無効。)
 - 親文字列の均等割りは常に無効である。(e 指定は無効。)
 - ルビ付き文字と前後の文字との間の空き調整や行分割可否は両者がともに欧文であるという想定で行われる。従って、既定では空き調整量はゼロ、行分割は禁止となる。
 - 空き調整を和欧文間空白 (:) にした場合は、* が指定されるあるいは自動の禁則処理が働くのでない限り、行分割が許可される。
- `\truby` [`<オプション>`] {`<親文字>`} {`<上側ルビ文字>`} {`<下側ルビ文字>`}

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

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

※ 1.1 版以前では常にグループルビの扱いであった。旧版との互換のため、両側ルビの場合には自動切替モノルビ (M) を既定値とする。^{*3}

- `\atruby` [〈オプション〉]{親文字列}{上側ルビ文字列}{下側ルビ文字列}
- 欧文両側ルビの命令。欧文ルビであることを除き `\truby` と同じ。

2.3 ルビ命令の入力文字列の入力規則

ルビの処理では入力文字列（親文字列・ルビ文字列）を文字毎に分解する必要がある。このため、ルビ命令の入力文字列は一定の規則に従って書かれる必要がある。

ルビ命令の入力文字列には以下のものを含めることができる。

- | : グループの区切りを表す。
 - 現在の版では、親文字列でグループ区切りを利用する機能はない。^{*4}従って、親文字列中に | があると常にエラーになる。
 - ルビ文字列中では、一つのグループが一つの親文字列に対応する範囲を表す（モノルビ・熟語ルビの場合）。
- 通常文字 : \LaTeX の命令や特殊文字や欧文空白や | でない、欧文または和文の文字を指す。これは一つの《文字》と見なされる。
 - 和文ルビ命令の親文字列に欧文文字を含めた場合、その文字は組版上 “和文文字のように” 振舞う。
- クラスタ : すなわち、{ } に囲まれたテキスト。全体が一つの《文字》と見なされる。
 - クラスタの中では任意の \LaTeX の “インライン”^{*5}の命令が使える。
`\ruby[j]{\CID{7652}飾区}{かつ|し{\color{red}{か}}|く}`
 - クラスタ中の | は通常文字として扱われる。

※ 例外的に、欧文ルビの親文字列は、あたかもそれ全体が一つのクラスタであるように振舞う。つまり、任意の “インライン” の命令が使えて、全体で一つの欧文文字であるのと同様に振舞う。

2.4 ルビ文字列のグループの指定

ルビ文字列の | はグループの区切りを表す。例えば、ルビ文字列

^{*3} つまり、旧来の使用ではグループルビと扱われるため、ルビのグループは 1 つにしているはずで、これは新版でもそのままグループルビと扱われる。一方で、モノルビを使いたい場合はグループを複数にするはずで、この時は自動的にモノルビになる。なので結局、基底モード (g, m) を指定する必要は無いことになる。

^{*4} 将来の機能拡張において、親文字列が複数グループをもつような使用法が想定されている。

^{*5} 「強制改行や改段落を含まない」ということ。 \LaTeX の用語では「LR モード」と呼ぶ。

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《文字》(通常文字またはクラスタ)となる。例えば

`ベクタ{\< (-) \>}`

は1つのグループからなり、それは4つの《文字》からなる。

グループや《文字》の指定はルビの付き方に影響する。

- 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。

2.5 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補充するためだからである。

2.6 パラメタ設定命令

基本的設定。

- `\rubyssetup{<オプション>}`
オプションの既定値設定。[既定 = `|c|j|P|e|F|`]
 - これ自体の既定値は「突出許可、進入無し、中付き、熟語ルビ、上側配置、親文字均等割り有効、小書き文字変換無効」である。
 - `<前補助設定>` / `<後補助設定>` の既定値は変更できない。`\rubyssetup` でこれらのオプション文字を指定しても無視される。
 - `\rubyssetup` での設定は累積する。例えば、初期状態から、`\rubyssetup{hmf}` と `\rubyssetup{<->}` を実行した場合、既定値設定は `<hmPef>` となる。
 - この設定に関わらず、両側ルビでは「自動切替モノルビ (M)」が既定として指定される。
- `\rubyfontsetup{<命令>}`
ルビ用のフォント切替命令を設定する。例えば、ルビは必ず明朝体で出力したいという場合は、以下の命令を実行すればよい。
`\rubyfontsetup{\mcfamily}`
- `\rubymargin{<実数>}`
「大」の進入量 (ルビ全角単位)。[既定 = 1]
- `\rubysmallmargin{<実数>}`
「小」の進入量 (ルビ全角単位)。[既定 = 0.5]
- `\rubymaxmargin{<実数>}`
ルビ文字列の方が短い場合の、ルビ文字列の端の親文字列の端からの距離の上限值 (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*6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)。
<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> (<primary>)	主マーク	<code>P</code> (<primary>)	上側配置
<code>s</code> (<secondary>)	副マーク	<code>S</code> (<secondary>)	下側配置
<code>f</code> (<full>)	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。

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

- 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）に圈点を付す指定。
- f 指定時は、親文字列に含まれる“通常文字”の全てに圈点を付加する。F 指定時は、約物である“通常文字”には圈点を付加しない。

3.2 圈点命令の親文字列の入力規則

圈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、圈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。

圈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圈点が付加される。
 - F オプションを指定した場合、約物（句読点等）の文字には圈点が付加されない。
 - 欧文文字に圈点を付けた場合、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：これらには圈点が付加されない。
 - 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。^{*7}
- クラスタ：すなわち、`{ }` に囲まれた任意のテキスト。ルビ命令のクラスタと同様に、一つの《文字》として扱われ、全体に対して一つの圈点が付加される。
 - `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圈点が付加される。
 - 例えば、“くの字点”に圈点を付す場合に使える。
 - あるいは、(少々手抜きであるが^{*8}) `\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>}`：これは圈点を付さずにテキストをそのまま出力する。
- ルビ命令 (`\ruby` 等)：例えば


```
\kenten{これが\ruby[|j|]{圈点}{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、ルビと圈点の両方が付加される。
 - 実装上の制限^{*9}のため、圈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、圈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。

^{*7} 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。

^{*8} 本来は、`\textgt` の中で改めて `\kenten` を使うべきである。

^{*9} 圈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。

- 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]{\kentent{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]
 - `\kententmarkinyoko{<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。

<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>	ヽ（合成）	黒ゴマ点	<code>Circle</code>	○ 25CB	白丸
<code>sesame</code>	ヽ FE45*	黒ゴマ点	<code>bullseye</code>	◎ 25CE	二重丸
<code>Sesame</code>	ヽ 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 にないことを表す。pL^AT_EX でこれらの圏点種別を利用するためには `japanese-otf` パッケージを読み込む必要がある。
- あるいは、名前の代わりに任意の L^AT_EX のテキストを書くことができる。^{*10}

^{*10} 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

- `\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]

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 }
```

たい場合は適宜 { } を補う等の措置が必要である。

```

\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{Invalud 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     Movable 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-ja 等) はこのパッケージより前に読み込むべきだが、後で読み込まれていることが判明した場合。

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

`\pxrr@interror` 内部エラー。これが出てはいけない。:-)

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

`\ifpxrrDebug` デバッグモード指定。

```
85 \newif \ifpxrrDebug
```

4.3 パラメタ

4.3.1 全般設定

`\pxrr@ruby@font` ルビ用フォント切替命令。

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

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

`\pxrr@small@intr`

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

`\pxrr@size@ratio` ルビ文字サイズ (`\rubysize`)。実数値マクロ。

```
89 \def \pxrr@size@ratio{0.5}
```

`\pxrr@sprop@x` 伸縮配置比率 (`\rubystretchprop`)。実数値マクロ。

`\pxrr@sprop@y` 90 `\def \pxrr@sprop@x{1}`

`\pxrr@sprop@z` 91 `\def \pxrr@sprop@y{2}`

```
92 \def \pxrr@sprop@z{1}
```

`\pxrr@sprop@hy` 伸縮配置比率 (`\rubystretchprophead`)。実数値マクロ。

`\pxrr@sprop@hz` 93 `\def \pxrr@sprop@hy{1}`

```
94 \def \pxrr@sprop@hz{1}
```

`\pxrr@sprop@ex` 伸縮配置比率 (`\rubystretchpropend`)。実数値マクロ。

`\pxrr@sprop@ey` 95 `\def \pxrr@sprop@ex{1}`

```
96 \def \pxrr@sprop@ey{1}
```

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

`\pxrr@yhtratio` 横組和文の高さの縦幅に対する割合 (`\rubyyheightratio`)。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@modefalse`

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

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

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

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

4.3.2 呼出時パラメタ・変数

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

`\ifpxrr@bprotr` 突出を許すか否か。 `\ruby` の `<前設定>/<後設定>` に由来する。スイッチ。
`\ifpxrr@aprotr` 116 `\newif\ifpxrr@bprotr \pxrr@bprotrfalse`
 117 `\newif\ifpxrr@aprotr \pxrr@aprotrfalse`

`\pxrr@bintr` 進入量。 `\ruby` の `<前設定>/<後設定>` に由来する。寸法値に展開されるマクロ。
`\pxrr@aintr` 118 `\def\pxrr@bintr{}`
 119 `\def\pxrr@aintr{}`

`\pxrr@bscomp` 空き補正設定。 `\ruby` の `:` 指定に由来する。暗黙の文字トークン（無指定は `\relax`）。
`\pxrr@ascomp` ※ 既定値設定 (`\rubysetup`) でこれに対応するものはない。
 120 `\let\pxrr@bscomp\relax`
 121 `\let\pxrr@ascomp\relax`

`\ifpxrr@bnobr` ルビ付文字の直前/直後で行分割を許すか。 `\ruby` の `*` 指定に由来する。スイッチ。
`\ifpxrr@anobr` ※ 既定値設定 (`\rubysetup`) でこれに対応するものはない。
 122 `\newif\ifpxrr@bnobr \pxrr@bnobrfalse`
 123 `\newif\ifpxrr@anobr \pxrr@anobrfalse`

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

`\pxrr@athead` 肩付き/中付きの設定。 `\ruby` の `c/h/H` の設定。値の意味は `\pxrr@d@athead` と同じ。整数定数。
 126 `\chardef\pxrr@athead=0`

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

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

`\ifpxrr@mode@given` 基本モードの設定が明示的であるか。スイッチ。
129 `\newif\ifpxrr@mode@given \pxrr@mode@givenfalse`
130 `\newif\ifpxrr@afintr \pxrr@afintrfalse`

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

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

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

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

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

`\pxrr@c@ruby@font` 以下は“オプションで指定する”以外のパラメタに対応するもの。
`\pxrr@c@size@ratio` 136 `\let\pxrr@c@ruby@font\@undefined`
137 `\let\pxrr@c@size@ratio\@undefined`
`\pxrr@c@inter@gap` 138 `\let\pxrr@c@inter@gap\@undefined`

4.4 その他の変数

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

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

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

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

`\pxrr@sruby@list` 2つ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@cnta` 汎用の整数レジスタ。
151 `\newcount\pxrr@cnta`

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

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

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

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

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

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

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

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

`\pxrr@res` 結果を格納するマクロ。

```
163 \let\pxrr@res\@empty
```

`\pxrr@ifx` `\pxrr@ifx{<引数>{<真>}{<偽>}` : `\ifx<引数>` を行うテスト。

```
164 \def\pxrr@ifx#1{%
165   \ifx#1\expandafter\@firstoftwo
166   \else\expandafter\@secondoftwo
167   \fi
168 }
```

`\pxrr@cond` `\pxrr@cond\ifXXX...\fi{<真>}{<偽>}` : 一般の T_EX の if 文 `\ifXXX...` を行うテスト。
※ `\fi` を付けているのは、if-不均衡を避けるため。

```
169 \@gobbletwo\if\if \def\pxrr@cond#1\fi{%
170   #1\expandafter\@firstoftwo
171   \else\expandafter\@secondoftwo
172   \fi
173 }
```

`\pxrr@cslet` `\pxrr@cslet{NAMEa}\CSb` : `\NAMEa` に `\CSb` を `\let` する。

`\pxrr@letcs` `\pxrr@letcs\CSa{NAMEb}` : `\CSa` に `\NAMEb` を `\let` する。

`\pxrr@csletcs` `\pxrr@csletcs{NAMEa}{NAMEb}` : `\NAMEa` に `\NAMEb` を `\let` する。

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

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

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

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

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

`\pxrr@nil` ユニークトークン。

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

```
191 \def\pxrr@end{\noexpand\pxrr@end}
```

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

```

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

```

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

```

\pxrr@hbox@to 199 \def\pxrr@hbox#1{%
200   \hbox{%
201     \color@begingroup
202     #1%
203     \color@endgroup
204   }%
205 }
206 \def\pxrr@hbox@to#1#2{%
207   \pxrr@hbox@to@a{#1}%
208 }
209 \def\pxrr@hbox@to@a#1#2{%
210   \hbox to#1{%
211     \color@begingroup
212     #2%
213     \color@endgroup
214   }%
215 }

```

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

```

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

```

4.5.2 数値計算

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

```

225 \mathchardef\pxrr@invscale@ca=259
226 \def\pxrr@invscale#1#2{%
227   \begingroup
228   \@tempdima=#1\relax
229   \@tempdimb#2\p@\relax

```

```

230 \@tempcnta\@tempdima
231 \multiply\@tempcnta\@ccclvi
232 \divide\@tempcnta\@tempdimb
233 \multiply\@tempcnta\@ccclvi
234 \@tempcntb\p@
235 \divide\@tempcntb\@tempdimb
236 \advance\@tempcnta-\@tempcntb
237 \advance\@tempcnta-\tw@
238 \@tempdimb\@tempcnta\@ne
239 \advance\@tempcnta\@tempcntb
240 \advance\@tempcnta\@tempcntb
241 \advance\@tempcnta\pxrr@invscale@ca
242 \@tempdimc\@tempcnta\@ne
243 \@whiledim\@tempdimb<\@tempdimc\do{%
244 \@tempcntb\@tempdimb
245 \advance\@tempcntb\@tempdimc
246 \advance\@tempcntb\@ne
247 \divide\@tempcntb\@tw@
248 \ifdim #2\@tempcntb>\@tempdima
249 \advance\@tempcntb\m@ne
250 \@tempdimc=\@tempcntb\@ne
251 \else
252 \@tempdimb=\@tempcntb\@ne
253 \fi}%
254 \xdef\pxrr@gtempa{\the\@tempdimb}%
255 \endgroup
256 #1=\pxrr@gtempa\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:

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

この命令は `\pxrr@res` を以下の内容に定義する。

```

\pxrr@pre{<X1>}{<Y1>}\pxrr@inter{<X2>}{<Y2>}\dots
\pxrr@inter{<Xn>}{<Yn>}\pxrr@post

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 :

```

\CSa = \pxrr@pre{<X1>}\pxrr@inter{<X2>}\dots\pxrr@inter{<Xn>}\pxrr@post
\CSb = \pxrr@pre{<Y1>}\pxrr@inter{<Y2>}\dots\pxrr@inter{<Yn>}\pxrr@post
\CSc = \pxrr@pre{<Z1>}\pxrr@inter{<Z2>}\dots\pxrr@inter{<Zn>}\pxrr@post

```

この命令は `\pxrr@res` を以下の内容に定義する。

```

\pxrr@pre{<X1>}{<Y1>}{<Z1>}\pxrr@inter{<X2>}{<Y2>}{<Z2>}\dots
\pxrr@inter{<Xn>}{<Yn>}{<Zn>}\pxrr@post

```

```

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

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

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

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

の時に、`\CS` を以下の内容で置き換える。

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

```
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` :

$$\backslash CSa = \langle X \rangle; \backslash CSb = \langle Y \rangle$$

の時に、`\pxrr@res` を以下の内容に定義する。

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

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

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

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

の時に、`\pxrr@res` を以下の内容に定義する。

```
\pxrr@pre{(X)}{(Y)}{(Z)}\pxrr@post
448 \def\pxrr@tzip@single#1#2#3{%
449   \expandafter\pxrr@tzip@single@a\expandafter#1\expandafter#2%
450   \expandafter{#3}%
451 }
452 \def\pxrr@tzip@single@a#1#2{%
453   \expandafter\pxrr@tzip@single@b\expandafter#1\expandafter{#2}%
454 }
455 \def\pxrr@tzip@single@b#1{%
456   \expandafter\pxrr@tzip@single@c\expandafter{#1}%
457 }
458 \def\pxrr@tzip@single@c#1#2#3{%
459   \def\pxrr@res{\pxrr@pre{#1}{#2}{#3}\pxrr@post}%
460 }
```

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。

```
461 \let\pxrr@x@tempa\@empty
462 \let\pxrr@x@tempb\@empty
463 \let\pxrr@x@gtempa\@empty
464 \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。

```
465 \def\pxrr@ifprimitive#1{%
466   \edef\pxrr@x@tempa{\string#1}%
467   \edef\pxrr@x@tempb{\meaning#1}%
468   \ifx\pxrr@x@tempa\pxrr@x@tempb \expandafter\@firstoftwo
469   \else \expandafter\@secondoftwo
470   \fi
471 }
```

`\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。

```
472 \pxrr@ifprimitive\kansuji{%
473   \pxrr@csletcs{ifpxrr@in@ptex}{iftrue}%
474 }{%
475   \pxrr@csletcs{ifpxrr@in@ptex}{iffalse}%
476 }
```

`\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。

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

```
480 \pxrr@csletcs{ifpxrr@in@uptex}{iffalse}%
481 }
```

`\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。

```
482 \pxrr@ifprimitive\XeTeXrevision{%
483 \pxrr@csletcs{ifpxrr@in@xetex}{iftrue}%
484 }{%
485 \pxrr@csletcs{ifpxrr@in@xetex}{iffalse}%
486 }
```

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

```
487 \@ifpackageloaded{xeCJK}{%
488 \pxrr@csletcs{ifpxrr@in@xecjk}{iftrue}%
489 }{%
490 \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。

```
491 \AtBeginDocument{%
492 \ifpackageloaded{xeCJK}{%
493 \pxrr@warn@load@order{xeCJK}%
494 }{%
495 }%
496 }
```

`\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。

```
497 \pxrr@ifprimitive\luatexrevision{%
498 \pxrr@csletcs{ifpxrr@in@luatex}{iftrue}%
499 }{%
500 \pxrr@csletcs{ifpxrr@in@luatex}{iffalse}%
501 }
```

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

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

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

```
505 \@ifpackageloaded{luatexja-core}{%
506 \pxrr@csletcs{ifpxrr@in@luatexja}{iftrue}%
507 }{%
508 \pxrr@csletcs{ifpxrr@in@luatexja}{iffalse}%
509 \AtBeginDocument{%
510 \ifpackageloaded{luatexja-core}{%
511 \pxrr@warn@load@order{LuaTeX-ja}%
512 }{%
513 }%
514 }
```

```
515 \ifpxrr@in@xetex
516 \else\ifpxrr@in@luatex
```

```

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

```

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

```

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

```

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

```

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

```

`\pxrr@jchardef` 和文用の `\chardef`。

```

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

```

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

```
559 \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。

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

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

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

※ 縦組対応 (`\ltj@curtfont` が定義済) でない古い LuaTeX-jā の場合は常に横組と見なす。

```
568 \ifx\ltj@curtfont\undefined
569 \let\pxrr@if@in@tate\@secondoftwo
570 \else
571 \def\pxrr@if@in@tate{%
572   \pxrr@cond{ifnum\ltjgetparameter{direction}=\thr@@\fi
573   }
574 \fi
575 \else
```

それ以外は常に横組と見なす。

```
576 \let\pxrr@if@in@tate\@secondoftwo
577 \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。

```
578 \ifpxrr@in@ptex
579 \def\pxrr@get@jchar@token#1#2{%
580   \begingroup
581     \kansujichar\@ne=#2\relax
582     \xdef\pxrr@x@gtempa{\kansuji\@ne}%
583   \endgroup
584   \let#1\pxrr@x@gtempa
585 }
```

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

```
586 \else\ifpxrr@in@unicode
587 \def\pxrr@get@jchar@token#1#2{%
```

```

588 \begingroup
589 \lccode'\?=#2\relax
590 \lowercase{\xdef\pxrr@x@gtempa{?}}%
591 \endgroup
592 \let#1\pxrr@x@gtempa
593 }

```

それ以外ではダミー定義。

```

594 \else
595 \def\pxrr@get@jchar@token#1#2{%
596 \def#1{?}%
597 }
598 \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のだから `chardef` にする。

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

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

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

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

```

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

```

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

```

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

```



```
620 }
```

それ以外の場合は（仕方が無いので）全角空白を用いる。

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

`\pxrr@x@K` 適当な漢字（実際は〈一〉）のトークン。

```
624 \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。

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

以下では `\kanjiskip` 挿入が有効であることを検査している。

```
627 \pxrr@x@swafalse
628 \begingroup
629 \inhibitxspcode\pxrr@x@K\thr@@
630 \kanjiskip\p@
631 \setbox\z@\hbox{\noautospacing\pxrr@x@K\pxrr@x@K}%
632 \setbox\tw@\hbox{\pxrr@x@K\pxrr@x@K}%
633 \ifdim\wd\tw@>\wd\z@
634 \aftergroup\pxrr@x@swatruue
635 \fi
636 \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。

```
637 \edef#1{%
638 \ifpxrr@x@swa \the\kanjiskip
639 \else \pxrr@zeropt
640 \fi
641 }%
642 }
```

LuaTeX-ja 使用の場合。

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

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

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

推定が失敗した場合。警告を（一度だけ）出した上で、値をゼロとして扱う。

```
650 \pxrr@warn@unknown@iiskip
```

```

651     \global\let\pxrr@x@tempa\pxrr@zeropt
652     \fi
653     \fi
654     \let#1\pxrr@x@tempa
655     \else
656     \let#1\pxrr@zeropt
657     \fi
658 }

```

和文間空白の推定に失敗した場合の警告。

```

659 \def\pxrr@warn@unknown@iiskip{%
660   \global\let\pxrr@warn@unknown@iiskip\relax
661   \pxrr@warn{Cannot find the kanjiskip value}%
662 }

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

ここでの n が “文字間グルー” のノードである。

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

```

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

それ以外の場合はゼロとする。

```

697 \else
698   \def\pxrr@get@iiskip#1{%
699     \let#1\pxrr@zeropt
700   }
701 \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。
p \TeX 系の場合。

```

702 \ifpxrr@in@ptex
703   \def\pxrr@get@iaiskip#1{%
704     \pxrr@x@swafalse
705     \begingroup
706       \inhibitxspcode\pxrr@x@K\thr@@ \xspcode'X=\thr@@
707       \xkanjiskip\p@
708       \setbox\z@\hbox{\noautoxspacing\pxrr@x@K X}%
709       \setbox\tw@\hbox{\pxrr@x@K X}%
710       \ifdim\wd\tw@>\wd\z@
711         \aftergroup\pxrr@x@swatrue
712       \fi
713     \endgroup
714     \edef#1{%
715       \ifpxrr@x@swa \the\xkanjiskip
716       \else \pxrr@zeropt
717     \fi
718   }%
719 }
```

Lua \TeX -ja 使用の場合。処理の流れは和文間空白の場合と同じ。

```

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

```
724 \ifdim\glueexpr\pxrr@x@gtempa=\maxdimen
```

判定用のボックスは欧文・和文の組とする。

```
725 \pxrr@get@interchar@glue{A\pxrr@x@K}%
726 \ifdim\glueexpr\pxrr@x@gtempa=\maxdimen
727 \pxrr@warn@unknown@iaiskip
728 \global\let\pxrr@x@gtempa\pxrr@zeropt
729 \fi
730 \fi
731 \let#1\pxrr@x@gtempa
732 \else
733 \let#1\pxrr@zeropt
734 \fi
735 }
```

和欧文間空白の推定に失敗した場合の警告。

```
736 \def\pxrr@warn@unknown@iaiskip{%
737 \global\let\pxrr@warn@unknown@iaiskip\relax
738 \pxrr@warn{Cannot find the xkanjiskip value}%
739 }
```

それ以外の場合は実際の組版結果から判断する。

```
740 \else
741 \def\pxrr@get@iaiskip#1{%
742 \begingroup
743 \setbox\z@\hbox{M\pxrr@x@K}%
744 \setbox\tw@\hbox{M\vrule\@width\z@\relax\pxrr@x@K}%
745 \@tempdima\wd\z@ \advance\@tempdima-\wd\tw@
746 \@tempdimb\@tempdima \divide\@tempdimb\thr@@
747 \xdef\pxrr@x@gtempa{\the\@tempdima\space minus \the\@tempdimb}%
748 \endgroup
749 \let#1=\pxrr@x@gtempa
750 }%
751 \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。

```
752 \ifpxrr@in@ptex
753 \def\pxrr@get@zwidth#1{%
754 \@tempdima=1zw\relax
755 \edef#1{\the\@tempdima}%
756 }
```

`\zw` が定義されている場合は `1\zw` とする。

```
757 \else\if\ifx\zw\undefined T\else F\fi F% if defined
758 \def\pxrr@get@zwidth#1{%
759 \@tempdima=1\zw\relax
760 \edef#1{\the\@tempdima}%
761 }
```

`\jsZw` が定義されている場合は `1\jsZw` とする。

```
762 \else\if\ifx\jsZw\undefined T\else F\fi F% if defined
763   \def\pxrr@get@zwidth#1{%
764     \@tempdima=1\jsZw\relax
765     \edef#1{\the\@tempdima}%
766   }
```

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。

```
767 \else\ifnum\pxrr@x@K>\@cclv
768   \def\pxrr@get@zwidth#1{%
769     \setbox\tw\hbox{\pxrr@x@K}%
770     \@tempdima\wd\tw@
771     \ifdim\@tempdima>z@\else \@tempdima\fontsize\p@ \fi
772     \edef#1{\the\@tempdima}%
773   }
```

それ以外の場合は要求サイズと等しいとする。

```
774 \else
775   \def\pxrr@get@zwidth#1{%
776     \@tempdima\fontsize\p@\relax
777     \edef#1{\the\@tempdima}%
778   }
779 \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。

```
780 \ifpxrr@in@ptex
781   \def\pxrr@get@prebreakpenalty#1#2{%
782     #1=\prebreakpenalty#2\relax
783   }
```

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

```
784 \else\ifpxrr@in@luatexja
785   \def\pxrr@get@prebreakpenalty#1#2{%
786     #1=\ltjgetparameter{prebreakpenalty}{#2}\relax
787   }
```

それ以外の場合はゼロとして扱う。

```
788 \else
789   \def\pxrr@get@prebreakpenalty#1#2{%
790     #1=\z@
791   }
792 \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。

```
793 \ifpxrr@in@ptex
```

```

794 \def\pxrr@get@postbreakpenalty#1#2{%
795   #1=\postbreakpenalty#2\relax
796 }

```

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

```

797 \else\ifpxrr@in@luatexja
798 \def\pxrr@get@postbreakpenalty#1#2{%
799   #1=\ltjgetparameter{postbreakpenalty}{#2}\relax
800 }

```

それ以外の場合はゼロとして扱う。

```

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

```

`\pxrr@check@punct@char` `\pxrr@check@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。

```

806 \ifpxrr@in@ptex
807 \def\pxrr@check@punct@char#1#2{%
808   \pxrr@okfalse
809   \ifcase#2\relax
810     \ifnum\xspcode#1=\thr@@\else
811       \pxrr@oktrue
812       \fi
813   \else
814     \ifnum\inhibitxspcode#1=\thr@@\else
815       \pxrr@oktrue
816       \fi
817   \fi
818 }

```

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

```

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

```

```
826 }
```

それ以外の場合は常に偽として扱う。

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

`\pxrr@force@nonpunct@achar` `\pxrr@force@nonpunct@achar{<文字コード>}`：指定の文字コードの欧文文字を“約物でない”ものと扱う。“約物である”の意味は `\pxrr@check@punct@char` の場合と同じ。
pTeX の場合。

```
832 \ifpxrr@in@ptex
833   \def\pxrr@force@nonpunct@achar#1{%
834     \global\xspcode#1=\thr@@
835   }
```

LuaTeX-ja 使用の場合。

```
836 \else\ifpxrr@in@luatexja
837   \def\pxrr@force@nonpunct@achar#1{%
838     \ltjglobalsetparameter{jaxspmde={#1,3}}%
839   }
```

それ以外の場合は何もしない。

```
840 \else
841   \def\pxrr@force@nonpunct@achar#1{}
842 \fi\fi
```

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

```
843 \ifx\inhibitglue\undefined
844   \let\pxrr@inhibitglue\relax
845 \else
846   \let\pxrr@inhibitglue\inhibitglue
847 \fi
```

4.7 パラメタ設定公開命令

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

```
848 \newif\ifpxrr@in@setup \pxrr@in@setupfalse
```

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

```
849 \newcommand*\rubyssetup[1]{%
850   \pxrr@in@setuptrue
851   \pxrr@fatal@errorfalse
852   \pxrr@parse@option{#1}%
853   \ifpxrr@fatal@error\else
854     \pxrr@csletcs{ifpxrr@d@bprotr}{ifpxrr@bprotr}%
855     \pxrr@csletcs{ifpxrr@d@aprotr}{ifpxrr@aprotr}%
```

```

856 \let\pxrr@d@bintr\pxrr@bintr@
857 \let\pxrr@d@aintr\pxrr@aintr@
858 \let\pxrr@d@athead\pxrr@athead
859 \let\pxrr@d@mode\pxrr@mode
860 \let\pxrr@d@side\pxrr@side
861 \let\pxrr@d@evensp\pxrr@evensp
862 \let\pxrr@d@fullsize\pxrr@fullsize
863 \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。

```

864 \pxrr@in@setupfalse
865 }

```

\rubyfontsetup 対応するパラメタを設定する。

```

866 \newcommand*\rubyfontsetup{}
867 \def\rubyfontsetup#{%
868 \def\pxrr@ruby@font
869 }

```

\rubybigintrusion 対応するパラメタを設定する。

```

\rubysmallintrusion 870 \newcommand*\rubybigintrusion[1]{%
\rubymaxmargin 871 \edef\pxrr@big@intr{#1}%
872 }
\rubyintergap 873 \newcommand*\rubysmallintrusion[1]{%
\rubysizeratio 874 \edef\pxrr@small@intr{#1}%
875 }
876 \newcommand*\rubymaxmargin[1]{%
877 \edef\pxrr@maxmargin{#1}%
878 }
879 \newcommand*\rubyintergap[1]{%
880 \edef\pxrr@inter@gap{#1}%
881 }
882 \newcommand*\rubysizeratio[1]{%
883 \edef\pxrr@size@ratio{#1}%
884 }

```

\rubyusejghost 対応するスイッチを設定する。

```

\rubynousejghost 885 \newcommand*\rubyusejghost{%
886 \pxrr@jghosttrue
887 }
888 \newcommand*\rubynousejghost{%
889 \pxrr@jghostfalse
890 }

```

\rubyuseaghost 対応するスイッチを設定する。

```

\rubynouseaghost 891 \newcommand*\rubyuseaghost{%
892 \pxrr@aghosttrue
893 \pxrr@setup@aghost

```



```

894 }
895 \newcommand*\rubynouseaghost{%
896   \pxrr@aghostfalse
897 }

```

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

```

\rubynoadjustatlineedge 898 \newcommand*\rubyadjustatlineedge{%
899   \pxrr@edge@adjusttrue
900 }
901 \newcommand*\rubynoadjustatlineedge{%
902   \pxrr@edge@adjustfalse
903 }

```

`\rubybreakjukugo` 対応するスイッチを設定する。

```

\rubynobreakjukugo 904 \newcommand*\rubybreakjukugo{%
905   \pxrr@break@jukugotrue
906 }
907 \newcommand*\rubynobreakjukugo{%
908   \pxrr@break@jukugofalse
909 }

```

`\rubysafemode` 対応するスイッチを設定する。

```

\rubynosafemode 910 \newcommand*\rubysafemode{%
911   \pxrr@safe@modetrue
912 }
913 \newcommand*\rubynosafemode{%
914   \pxrr@safe@modedefalse
915 }

```

`\rubystretchprop` 対応するパラメタを設定する。

```

\rubystretchprophead 916 \newcommand*\rubystretchprop[3]{%
\rubystretchpropend 917   \edef\pxrr@sprop@x{#1}%
918   \edef\pxrr@sprop@y{#2}%
919   \edef\pxrr@sprop@z{#3}%
920 }
921 \newcommand*\rubystretchprophead[2]{%
922   \edef\pxrr@sprop@hy{#1}%
923   \edef\pxrr@sprop@hz{#2}%
924 }
925 \newcommand*\rubystretchpropend[2]{%
926   \edef\pxrr@sprop@ex{#1}%
927   \edef\pxrr@sprop@ey{#2}%
928 }

```

`\rubyuseextra` 残念ながら今のところは使用不可。

```

929 \newcommand*\rubyuseextra[1]{%
930   \pxrr@ccta=#1\relax
931   \ifnum\pxrr@ccta=\z@
932     \chardef\pxrr@extra\pxrr@ccta

```

```

933 \else
934   \pxrr@err@inv@value{\the\pxrr@cmta}%
935 \fi
936 }

```

4.8 ルビオプション解析

`\pxrr@bintr@` オプション解析中にのみ使われ、進入の値を `\pxrr@d@?intr` と同じ形式で保持する。

`\pxrr@aintr@` (`\pxrr@?intr` は形式が異なることに注意。)

```

937 \let\pxrr@bintr@\empty
938 \let\pxrr@aintr@\empty

```

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

```

939 \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。

```

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

```

入力が「||」の場合は、「|-|」に置き換える。

```

941 \edef\pxrr@tempa{#1}%
942 \ifx\pxrr@tempa\pxrr@doublebar
943   \def\pxrr@tempa{|-|}%
944 \fi

```

各パラメタの値を全般設定のもので初期化する。

```

945 \pxrr@csletcs{ifpxrr@bprotr}{ifpxrr@d@bprotr}%
946 \pxrr@csletcs{ifpxrr@aprotr}{ifpxrr@d@aprotr}%
947 \let\pxrr@bintr@\pxrr@d@bintr
948 \let\pxrr@aintr@\pxrr@d@aintr
949 \let\pxrr@athead\pxrr@d@athead
950 \let\pxrr@mode\pxrr@d@mode
951 \let\pxrr@side\pxrr@d@side
952 \let\pxrr@evensp\pxrr@d@evensp
953 \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。

```

954 \let\pxrr@bscomp\relax
955 \let\pxrr@ascomp\relax
956 \pxrr@bnobrfalse
957 \pxrr@anobrfalse
958 \pxrr@bfintrfalse
959 \pxrr@afintrfalse

```

明示フラグを偽にする。

```

960 \pxrr@mode@givenfalse
961 \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。

```

962 \ifpxrr@truby
963   \let\pxrr@mode=M%
964 \fi

```

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

```

965 \def\pxrr@po@FS{bi}%
966 \expandafter\pxrr@parse@option@loop\pxrr@tempa @\pxrr@end
967 }

```

有限状態機械のループ。

```

968 \def\pxrr@parse@option@loop#1{%
969 \ifpxrrDebug
970 \typeout{\pxrr@po@FS/#1[\@nameuse{pxrr@po@C@#1}]}%
971 \fi
972 \csname pxrr@po@PR@#1\endcsname
973 \expandafter\ifx\csname pxrr@po@C@#1\endcsname\relax
974   \let\pxrr@po@FS\relax
975 \else
976   \pxrr@letcs\pxrr@po@FS
977   {\pxrr@po@TR@\pxrr@po@FS \@nameuse{pxrr@po@C@#1}]}%
978 \fi
979 \ifpxrrDebug
980 \typeout{->\pxrr@po@FS}%
981 \fi
982 \pxrr@ifx{\pxrr@po@FS\relax}{%
983   \pxrr@fatal@unx@letter{#1}%
984   \pxrr@parse@option@exit
985 }{%
986   \pxrr@parse@option@loop
987 }%
988 }

```

後処理。

```

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

```

既定値設定 (\rubyssetup) である場合もしない。

```

990 \ifpxrr@in@setup\else

```

両側ルビ命令の場合は、 \pxrr@side の値を変更する。

```

991   \ifpxrr@truby
992     \chardef\pxrr@side\tw@
993   \fi

```

整合性検査を行う。

```

994   \pxrr@check@option

```

\pxrr@?intr の値を設定する。

```

995   \@tempdima=\pxrr@ruby@zw\relax
996   \@tempdimb=\pxrr@or@zero\pxrr@bintr@\@tempdima
997   \edef\pxrr@bintr{\the\@tempdimb}%

```

```

998 \@tempdimb=\pxrr@or@zero\pxrr@aintr@\@tempdima
999 \edef\pxrr@aintr{\the\@tempdimb}%
1000 \fi
1001 }

```

\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。

```

1002 \def\pxrr@or@zero#1{%
1003 \ifx#1\@empty \pxrr@zero
1004 \else #1%
1005 \fi
1006 }

```

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。

```

1007 \def\pxrr@po@C@{F}
1008 \@namedef{pxrr@po@C@|}{V}
1009 \@namedef{pxrr@po@C@:}{S}
1010 \@namedef{pxrr@po@C@.}{S}
1011 \@namedef{pxrr@po@C@*}{S}
1012 \@namedef{pxrr@po@C@!}{S}
1013 \@namedef{pxrr@po@C@<}{B}
1014 \@namedef{pxrr@po@C@()}{B}
1015 \@namedef{pxrr@po@C@>}{A}
1016 \@namedef{pxrr@po@C@)}{A}
1017 \@namedef{pxrr@po@C@-}{M}
1018 \def\pxrr@po@C@c{M}
1019 \def\pxrr@po@C@h{M}
1020 \def\pxrr@po@C@H{M}
1021 \def\pxrr@po@C@m{M}
1022 \def\pxrr@po@C@g{M}
1023 \def\pxrr@po@C@j{M}
1024 \def\pxrr@po@C@M{M}
1025 \def\pxrr@po@C@J{M}
1026 \def\pxrr@po@C@P{M}
1027 \def\pxrr@po@C@S{M}
1028 \def\pxrr@po@C@e{M}
1029 \def\pxrr@po@C@E{M}
1030 \def\pxrr@po@C@f{M}
1031 \def\pxrr@po@C@F{M}

```

機能プロセス。

```

1032 \def\pxrr@po@PR@{F}
1033 \pxrr@parse@option@exit
1034 }
1035 \@namedef{pxrr@po@PR@|}{F}
1036 \csname pxrr@po@PRbar@\pxrr@po@FS\endcsname
1037 }
1038 \def\pxrr@po@PRbar@bi{F}
1039 \def\pxrr@bintr@{\}\pxrr@bprottrue

```

```

1040 }
1041 \def\pxrr@po@PRbar@bb{%
1042   \pxrr@bprotrfalse
1043 }
1044 \def\pxrr@po@PRbar@bs{%
1045   \def\pxrr@aintr@{}\pxrr@aprotrtrue
1046 }
1047 \let\pxrr@po@PRbar@mi\pxrr@po@PRbar@bs
1048 \let\pxrr@po@PRbar@as\pxrr@po@PRbar@bs
1049 \let\pxrr@po@PRbar@ai\pxrr@po@PRbar@bs
1050 \def\pxrr@po@PRbar@ab{%
1051   \pxrr@aprotrfalse
1052 }
1053 \@namedef{pxrr@po@PR@.}{%
1054   \csname pxrr@po@PRcolon@\pxrr@po@FS\endcsname
1055 }
1056 \def\pxrr@po@PRcolon@bi{%
1057   \let\pxrr@bscomp=\relax
1058 }
1059 \let\pxrr@po@PRcolon@bb\pxrr@po@PRcolon@bi
1060 \let\pxrr@po@PRcolon@bs\pxrr@po@PRcolon@bi
1061 \def\pxrr@po@PRcolon@mi{%
1062   \let\pxrr@ascomp=\relax
1063 }
1064 \let\pxrr@po@PRcolon@as\pxrr@po@PRcolon@mi
1065 \@namedef{pxrr@po@PR@.}{%
1066   \csname pxrr@po@PRdot@\pxrr@po@FS\endcsname
1067 }
1068 \def\pxrr@po@PRdot@bi{%
1069   \let\pxrr@bscomp=\relax
1070 }
1071 \let\pxrr@po@PRdot@bb\pxrr@po@PRdot@bi
1072 \let\pxrr@po@PRdot@bs\pxrr@po@PRdot@bi
1073 \def\pxrr@po@PRdot@mi{%
1074   \let\pxrr@ascomp=\relax
1075 }
1076 \let\pxrr@po@PRdot@as\pxrr@po@PRdot@mi
1077 \@namedef{pxrr@po@PR@*}{%
1078   \csname pxrr@po@PRstar@\pxrr@po@FS\endcsname
1079 }
1080 \def\pxrr@po@PRstar@bi{%
1081   \pxrr@bnobrtrue
1082 }
1083 \let\pxrr@po@PRstar@bb\pxrr@po@PRstar@bi
1084 \let\pxrr@po@PRstar@bs\pxrr@po@PRstar@bi
1085 \def\pxrr@po@PRstar@mi{%
1086   \pxrr@anobrtrue
1087 }
1088 \let\pxrr@po@PRstar@as\pxrr@po@PRstar@mi

```

```

1089 \@namedef{pxrr@po@PR@!}{%
1090   \csname pxrr@po@PRbang@pxrr@po@FS\endcsname
1091 }
1092 \def\pxrr@po@PRbang@bi{%
1093   \pxrr@bfintrtrue
1094 }
1095 \let\pxrr@po@PRbang@bb\pxrr@po@PRbang@bi
1096 \let\pxrr@po@PRbang@bs\pxrr@po@PRbang@bi
1097 \def\pxrr@po@PRbang@mi{%
1098   \pxrr@afintrtrue
1099 }
1100 \let\pxrr@po@PRbang@as\pxrr@po@PRbang@mi
1101 \@namedef{pxrr@po@PR@<}{%
1102   \def\pxrr@bintr@{\pxrr@big@intr}\pxrr@bprottrue
1103 }
1104 \@namedef{pxrr@po@PR@(){%
1105   \def\pxrr@bintr@{\pxrr@small@intr}\pxrr@bprottrue
1106 }
1107 \@namedef{pxrr@po@PR@>}{%
1108   \def\pxrr@aintr@{\pxrr@big@intr}\pxrr@aprottrue
1109 }
1110 \@namedef{pxrr@po@PR@}{%
1111   \def\pxrr@aintr@{\pxrr@small@intr}\pxrr@aprottrue
1112 }
1113 \def\pxrr@po@PR@c{%
1114   \chardef\pxrr@athead\z@
1115   \pxrr@athead@giventruetrue
1116 }
1117 \def\pxrr@po@PR@h{%
1118   \chardef\pxrr@athead\@ne
1119   \pxrr@athead@giventruetrue
1120 }
1121 \def\pxrr@po@PR@H{%
1122   \chardef\pxrr@athead\tw@
1123   \pxrr@athead@giventruetrue
1124 }
1125 \def\pxrr@po@PR@m{%
1126   \let\pxrr@mode=m%
1127   \pxrr@mode@giventruetrue
1128 }
1129 \def\pxrr@po@PR@g{%
1130   \let\pxrr@mode=g%
1131   \pxrr@mode@giventruetrue
1132 }
1133 \def\pxrr@po@PR@j{%
1134   \let\pxrr@mode=j%
1135   \pxrr@mode@giventruetrue
1136 }
1137 \def\pxrr@po@PR@M{%

```

```

1138 \let\pxrr@mode=M%
1139 \pxrr@mode@giventrue
1140 }
1141 \def\pxrr@po@PR@J{%
1142 \let\pxrr@mode=J%
1143 \pxrr@mode@giventrue
1144 }
1145 \def\pxrr@po@PR@P{%
1146 \chardef\pxrr@side\z@
1147 }
1148 \def\pxrr@po@PR@S{%
1149 \chardef\pxrr@side\@ne
1150 }
1151 \def\pxrr@po@PR@E{%
1152 \chardef\pxrr@evensp\z@
1153 }
1154 \def\pxrr@po@PR@e{%
1155 \chardef\pxrr@evensp\@ne
1156 }
1157 \def\pxrr@po@PR@F{%
1158 \chardef\pxrr@fullsize\z@
1159 }
1160 \def\pxrr@po@PR@f{%
1161 \chardef\pxrr@fullsize\@ne
1162 }

```

遷移表。

```

1163 \def\pxrr@po@TR@bi@F{fi}
1164 \def\pxrr@po@TR@bb@F{fi}
1165 \def\pxrr@po@TR@bs@F{fi}
1166 \def\pxrr@po@TR@mi@F{fi}
1167 \def\pxrr@po@TR@as@F{fi}
1168 \def\pxrr@po@TR@ai@F{fi}
1169 \def\pxrr@po@TR@ab@F{fi}
1170 \def\pxrr@po@TR@fi@F{fi}
1171 \def\pxrr@po@TR@bi@V{bb}
1172 \def\pxrr@po@TR@bb@V{bs}
1173 \def\pxrr@po@TR@bs@V{ab}
1174 \def\pxrr@po@TR@mi@V{ab}
1175 \def\pxrr@po@TR@as@V{ab}
1176 \def\pxrr@po@TR@ai@V{ab}
1177 \def\pxrr@po@TR@ab@V{fi}
1178 \def\pxrr@po@TR@bi@S{bs}
1179 \def\pxrr@po@TR@bb@S{bs}
1180 \def\pxrr@po@TR@bs@S{bs}
1181 \def\pxrr@po@TR@mi@S{as}
1182 \def\pxrr@po@TR@as@S{as}
1183 \def\pxrr@po@TR@bi@B{bs}
1184 \def\pxrr@po@TR@bi@M{mi}

```

```

1185 \def\pxrr@po@TR@bb@M{mi}
1186 \def\pxrr@po@TR@bs@M{mi}
1187 \def\pxrr@po@TR@mi@M{mi}
1188 \def\pxrr@po@TR@bi@A{fi}
1189 \def\pxrr@po@TR@bb@A{fi}
1190 \def\pxrr@po@TR@bs@A{fi}
1191 \def\pxrr@po@TR@mi@A{fi}
1192 \def\pxrr@po@TR@as@A{fi}
1193 \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。

```

1194 \def\pxrr@mode@grand{%
1195   \if      m\pxrr@mode m%
1196   \else\if M\pxrr@mode m%
1197   \else\if j\pxrr@mode j%
1198   \else\if J\pxrr@mode j%
1199   \else\if g\pxrr@mode g%
1200   \else ?%
1201   \fi\fi\fi\fi\fi
1202 }

```

`\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。

```

1203 \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。
1204   \ifpxrr@bprotr\else
1205     \ifpxrr@aprotr\else
1206       \pxrr@fatal@bad@no@protr
1207     \fi
1208   \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。

```

1209   \pxrr@oktrue
1210   \ifx\pxrr@bintr@\@empty\else
1211     \pxrr@okfalse
1212   \fi
1213   \ifx\pxrr@aintr@\@empty\else
1214     \pxrr@okfalse
1215   \fi
1216   \ifpxrr@ghost\else
1217     \pxrr@oktrue
1218   \fi
1219   \ifpxrr@ok\else

```



```
1220 \pxrr@fatal@bad@intr
1221 \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。

```
1222 \if g\pxrr@mode\else
1223 \ifpxrr@abody
1224 \let\pxrr@mode=g\relax
1225 \ifpxrr@mode@given
1226 \pxrr@warn@must@group
1227 \fi
1228 \fi
1229 \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。

```
1230 \if \pxrr@mode@grand j%
1231 \ifnum\pxrr@side=\tw@
1232 \let\pxrr@mode=g\relax
1233 \ifpxrr@mode@given
1234 \pxrr@warn@bad@jukugo
1235 \fi
1236 \fi
1237 \fi
```

肩付き指定 (h) に関する検査。

```
1238 \ifnum\pxrr@athead>\z@
```

横組みでは不可なので中付きに変更する。

```
1239 \pxrr@if@in@tate{ }{%else
1240 \chardef\pxrr@athead\z@
1241 }%
```

グループルビでは不可なので中付きに変更する。

```
1242 \if g\pxrr@mode
1243 \chardef\pxrr@athead\z@
1244 \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。

```
1245 \ifnum\pxrr@athead=\z@
1246 \ifpxrr@athead@given
1247 \pxrr@warn@bad@athead
1248 \fi
1249 \fi
1250 \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。

```
1251 \ifpxrr@abody
1252 \chardef\pxrr@evensp\z@
1253 \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。)

```
1254 \if g\pxrr@mode\else
1255   \chardef\pxrr@evensp\@ne
1256 \fi
```

圏点ルビ同時付加の場合の調整。

```
1257 \ifpxrr@combo
1258   \pxrr@ck@check@option
1259 \fi
1260 }
```

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。

```
1261 \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。

```
1262 \let\pxrr@body@zw\pxrr@zeropt
1263 \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。

```
1264 \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)。

```
1265 \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。

```
1266 \def\pxrr@htratio{0}
```

`\pxrr@iiskip` 和文間空白および和欧文間空白の量。

```
\pxrr@iaiskip 1267 \let\pxrr@iiskip\pxrr@zeropt
```

```
1268 \let\pxrr@iaiskip\pxrr@zeropt
```

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

```
1269 \def\pxrr@assign@fsize{%
1270   \@tempdima=\f@size\p@
1271   \@tempdima\pxrr@c@size@ratio\@tempdima
1272   \edef\pxrr@ruby@fsize{\the\@tempdima}%
1273   \pxrr@get@zwidth\pxrr@body@zw
1274   \begingroup
1275     \pxrr@use@ruby@font
1276     \pxrr@get@zwidth\pxrr@ruby@zw
```

```

1277 \global\let\pxrr@tempa\pxrr@ruby@zw
1278 \endgroup
1279 \let\pxrr@ruby@zw\pxrr@tempa
1280 \pxrr@get@iiskip\pxrr@iiskip
1281 \pxrr@get@iaiskip\pxrr@iaiskip

\pxrr@htratio の値を設定する。
1282 \pxrr@if@in@tate{%
1283 \let\pxrr@htratio\pxrr@thtratio
1284 }{%
1285 \let\pxrr@htratio\pxrr@yhtratio
1286 }%

\pxrr@ruby@raise の値を計算する。
1287 \@tempdima\pxrr@body@zw\relax
1288 \@tempdima\pxrr@htratio\@tempdima
1289 \@tempdimb\pxrr@ruby@zw\relax
1290 \advance\@tempdimb-\pxrr@htratio\@tempdimb
1291 \advance\@tempdima\@tempdimb
1292 \@tempdimb\pxrr@body@zw\relax
1293 \advance\@tempdima\pxrr@c@inter@gap\@tempdimb
1294 \edef\pxrr@ruby@raise{\the\@tempdima}%

\pxrr@ruby@lower の値を計算する。
1295 \@tempdima\pxrr@body@zw\relax
1296 \advance\@tempdima-\pxrr@htratio\@tempdima
1297 \@tempdimb\pxrr@ruby@zw\relax
1298 \@tempdimb\pxrr@htratio\@tempdimb
1299 \advance\@tempdima\@tempdimb
1300 \@tempdimb\pxrr@body@zw\relax
1301 \advance\@tempdima\pxrr@c@inter@gap\@tempdimb
1302 \edef\pxrr@ruby@lower{\the\@tempdima}%

圏点ルビ同時付加の設定。
1303 \ifpxrr@combo
1304 \pxrr@ck@assign@fsize
1305 \fi
1306 }

```

\pxrr@use@ruby@font ルビ用のフォントに切り替える。

```

1307 \def\pxrr@use@ruby@font{%
1308 \pxrr@without@macro@trace{%
1309 \let\rubyfontsize\pxrr@ruby@fsize
1310 \fontsize{\pxrr@ruby@fsize}{\z@}\selectfont
1311 \pxrr@c@ruby@font
1312 }%
1313 }

```

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 1314 \chardef\pxrr@locate@inner=1
\pxrr@locate@end 1315 \chardef\pxrr@locate@head=0
1316 \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。

```
1317 \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。

```
1318 \setbox#2\pxrr@hbox{#5}\@tempdima\wd#2%
1319 \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。

```
1320 \pxrr@decompose{#5}%
1321 \pxrr@evenspace@int{#1}{#2}{#3}{#4}%
1322 }
```

ここから実行を開始することもある。

```
1323 \def\pxrr@evenspace@int#1#2#3#4{%
```

比率パラメタの設定。

```
1324 \pxrr@save@listproc
1325 \ifcase#1%
1326 \pxrr@evenspace@param\pxrr@zero\pxrr@sprop@hy\pxrr@sprop@hz
1327 \or
1328 \pxrr@evenspace@param\pxrr@sprop@x\pxrr@sprop@y\pxrr@sprop@z
1329 \or
1330 \pxrr@evenspace@param\pxrr@sprop@ex\pxrr@sprop@ey\pxrr@zero
1331 \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。

```
1332 \pxrr@dima=\pxrr@cntr\p@
```

```

1333 \advance\pxrr@dima-\p@
1334 \pxrr@dima=\pxrr@sprop@y@\pxrr@dima
1335 \advance\pxrr@dima\pxrr@sprop@x@\p@
1336 \advance\pxrr@dima\pxrr@sprop@z@\p@
1337 \ifdim\pxrr@dima>\z@\else
1338   \ifnum#1>\z@
1339     \let\pxrr@sprop@x@\@ne
1340     \advance\pxrr@dima\p@
1341   \fi
1342   \ifnum#1<\tw@
1343     \let\pxrr@sprop@z@\@ne
1344     \advance\pxrr@dima\p@
1345   \fi
1346 \fi
1347 \edef\pxrr@tempa{\strip@pt\pxrr@dima}%
1348 \ifpxrrDebug
1349 \typeout{\number\pxrr@sprop@x@:\number\pxrr@sprop@z@:\pxrr@tempa}%
1350 \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。

```

1351 \def\pxrr@pre##1{\pxrr@hfilx\pxrr@sprop@x@ ##1}%
1352 \def\pxrr@inter##1{\pxrr@hfilx\pxrr@sprop@y@ ##1}%
1353 \def\pxrr@post{\pxrr@hfilx\pxrr@sprop@z@}%
1354 \def\pxrr@makebox@res{%
1355   \setbox#2=\pxrr@hbox@to#4{#3\pxrr@res}%
1356 }%
1357 \pxrr@makebox@res

```

前後の空白の量を求める。

```

1358 \pxrr@dima\wd#2%
1359 \advance\pxrr@dima-\pxrr@natwd\relax
1360 \pxrr@invscale\pxrr@dima\pxrr@tempa
1361 \@tempdima\pxrr@sprop@x@\pxrr@dima
1362 \edef\pxrr@bspace{\the\@tempdima}%
1363 \@tempdima\pxrr@sprop@z@\pxrr@dima
1364 \edef\pxrr@aspace{\the\@tempdima}%
1365 \pxrr@restore@listproc
1366 \ifpxrrDebug
1367 \typeout{\pxrr@bspace:\pxrr@aspace}%
1368 \fi
1369 }
1370 \def\pxrr@evenspace@param#1#2#3{%
1371   \let\pxrr@sprop@x@#1%
1372   \let\pxrr@sprop@y@#2%
1373   \let\pxrr@sprop@z@#3%
1374 }
1375 \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。

```
1376 \def\pxrr@adjust@margin{%
1377   \pxrr@save@listproc
1378   \@tempdima\pxrr@body@zw\relax
1379   \@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。

1380   \@tempswafalse
1381   \def\pxrr@pre##1{\pxrr@hfilx\pxrr@sprop@x@ ##1}%
1382   \def\pxrr@inter##1{\pxrr@hfilx\pxrr@sprop@y@ ##1}%
1383   \def\pxrr@post{\pxrr@hfilx\pxrr@sprop@z@}%
1384   \ifnum\pxrr@cntr>\@one
1385     \ifdim\pxrr@bspace>\@tempdima
1386       \edef\pxrr@bspace{\the\@tempdima}%
1387       \def\pxrr@pre##1{\hskip\pxrr@bspace\relax ##1}%
1388       \@tempswatrue
1389     \fi
1390     \ifdim\pxrr@aspace>\@tempdima
1391       \edef\pxrr@aspace{\the\@tempdima}%
1392       \def\pxrr@post{\hskip\pxrr@aspace\relax}%
1393       \@tempswatrue
1394     \fi
1395   \fi
```

必要に応じて再調整を行う。

```
1396   \if@tempswa
1397     \pxrr@makebox@res
1398   \fi
1399   \pxrr@restore@listproc
1400 \ifpxrrDebug
1401 \typeout{\pxrr@bspace:\pxrr@aspace}%
1402 \fi
1403 }
```

`\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。

```
1404 \def\pxrr@save@listproc{%
1405   \let\pxrr@pre@save\pxrr@pre
1406   \let\pxrr@inter@save\pxrr@inter
1407   \let\pxrr@post@save\pxrr@post
1408 }
1409 \let\pxrr@pre@save\@undefined
1410 \let\pxrr@inter@save\@undefined
1411 \let\pxrr@post@save\@undefined
```

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

```
1412 \def\pxrr@restore@listproc{%
1413   \let\pxrr@pre\pxrr@pre@save
1414   \let\pxrr@inter\pxrr@inter@save
1415   \let\pxrr@post\pxrr@post@save
1416 }
```

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。

```
1417 \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。

```
1418 \def\pxrr@transform@kana#1{%
1419   \let\pxrr@trans@res\@empty
1420   \def\pxrr@transform@kana@end\pxrr@end{%
1421     \let#1\pxrr@trans@res
1422   }%
1423   \expandafter\pxrr@transform@kana@loop@a#1\pxrr@end
1424 }
1425 \def\pxrr@transform@kana@loop@a{%
1426   \futurelet\pxrr@token\pxrr@transform@kana@loop@b
1427 }
1428 \def\pxrr@transform@kana@loop@b{%
1429   \ifx\pxrr@token\pxrr@end
1430     \let\pxrr@tempb\pxrr@transform@kana@end
1431   \else\ifx\pxrr@token\bgroup
1432     \let\pxrr@tempb\pxrr@transform@kana@loop@c
1433   \else\ifx\pxrr@token\@sptoken
1434     \let\pxrr@tempb\pxrr@transform@kana@loop@d
1435   \else
1436     \let\pxrr@tempb\pxrr@transform@kana@loop@e
1437   \fi\fi\fi
1438   \pxrr@tempb
1439 }
1440 \def\pxrr@transform@kana@loop@c#1{%
1441   \pxrr@appto\pxrr@trans@res{#{1}}%
1442   \pxrr@transform@kana@loop@a
1443 }
1444 \expandafter\def\expandafter\pxrr@transform@kana@loop@d\space{%
1445   \pxrr@appto\pxrr@trans@res{ }%
1446   \pxrr@transform@kana@loop@a
1447 }
1448 \def\pxrr@transform@kana@loop@e#1{%
1449   \expandafter\pxrr@transform@kana@loop@f\string#1\pxrr@nil#1%
1450 }
```

```

1451 \def\pxrr@transform@kana@loop@f#1#2\pxrr@nil#3{%
1452   \@tempswafalse
1453   \ifnum'#1>\@cclv
1454     \begingroup\expandafter\expandafter\expandafter\endgroup
1455     \expandafter\ifx\csname pxrr@nonsmall/#3\endcsname\relax\else
1456       \@tempswatruue
1457       \fi
1458       \fi
1459       \if@tempswa
1460         \edef\pxrr@tempa{%
1461           \noexpand\pxrr@appto\noexpand\pxrr@trans@res
1462             {\csname pxrr@nonsmall/#3\endcsname}%
1463           }%
1464         \pxrr@tempa
1465       \else
1466         \pxrr@appto\pxrr@trans@res{#3}%
1467       \fi
1468       \pxrr@transform@kana@loop@a
1469 }
1470 \def\pxrr@assign@nonsmall#1/#2\pxrr@nil{%
1471   \pxrr@get@jchar@token\pxrr@tempa{\pxrr@jc{#1}}%
1472   \pxrr@get@jchar@token\pxrr@tempb{\pxrr@jc{#2}}%
1473   \expandafter\edef\csname pxrr@nonsmall/\pxrr@tempa\endcsname
1474     {\pxrr@tempb}%
1475 }
1476 \@tfor\pxrr@tempc:=%
1477   {2421:3041/2422:3042}{2423:3043/2424:3044}%
1478   {2425:3045/2426:3046}{2427:3047/2428:3048}%
1479   {2429:3049/242A:304A}{2443:3063/2444:3064}%
1480   {2463:3083/2464:3084}{2465:3085/2466:3086}%
1481   {2467:3087/2468:3088}{246E:308E/246F:308F}%
1482   {2521:30A1/2522:30A2}{2523:30A3/2524:30A4}%
1483   {2525:30A5/2526:30A6}{2527:30A7/2528:30A8}%
1484   {2529:30A9/252A:30AA}{2543:30C3/2544:30C4}%
1485   {2563:30E3/2564:30E4}{2565:30E5/2566:30E6}%
1486   {2567:30E7/2568:30E8}{256E:30EE/256F:30EF}%
1487   \do{%
1488     \expandafter\pxrr@assign@nonsmall\pxrr@tempc\pxrr@nil
1489 }

```

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。

```
1490 \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。

```
1491 \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。

```
1492 \let\pxrr@locate@temp\relax
```

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

```
1493 \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。

```
1494 \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。

```
1495 \let\pxrr@compose@block@do\pxrr@compose@oneside@block@do
```

```
1496 \pxrr@compose@block@pre{#1}{#2}{#3}{}%
```

```
1497 }
```

こちらが本体。

```
1498 % #4 は空
```

```
1499 \def\pxrr@compose@oneside@block@do#1#2#3#4{%
```

```
1500 \setbox\pxrr@boxa\pxrr@hbox{#2}%
```

```
1501 \edef\pxrr@ck@body@natwd{\the\wd\pxrr@boxa}%
```

```
1502 \let\pxrr@ck@locate\pxrr@locate@inner
```

```
1503 \setbox\pxrr@boxr\pxrr@hbox{%
```

```
1504 \pxrr@use@ruby@font
```

```
1505 #3%
```

```
1506 }%
```

```
1507 \@tempdima\wd\pxrr@boxr
```

```
1508 \advance\@tempdima-\wd\pxrr@boxa
```

```
1509 \ifdim\pxrr@epsilon<\@tempdima
```

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

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

```
1510 \pxrr@protrtrue
```

```
1511 \let\pxrr@locate@temp#1%
```

```
1512 \ifnum\pxrr@athead>\@ne
```

```
1513 \ifnum\pxrr@locate@temp=\pxrr@locate@inner
```

```
1514 \let\pxrr@locate@temp\pxrr@locate@head
```

```
1515 \fi
```

```
1516 \fi
```

```
1517 \let\pxrr@ck@locate\pxrr@locate@temp
```

```
1518 \pxrr@decompose{#2}%
```

```
1519 \edef\pxrr@natwd{\the\wd\pxrr@boxa}%
```

```
1520 \pxrr@evenspace@int\pxrr@locate@temp\pxrr@boxa\relax
```

```

1521     {\wd\pxrr@boxr}%
1522 \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。

```

1523 \pxrr@protrfalse
1524 \ifnum\pxrr@athead=\z@
1525   \pxrr@decompose{#3}%
1526   \edef\pxrr@natwd{\the\wd\pxrr@boxr}%
1527   \pxrr@evenspace@int{#1}\pxrr@boxr
1528   \pxrr@use@ruby@font{\wd\pxrr@boxa}%
1529   \pxrr@adjust@margin
1530 \fi
1531 \let\pxrr@bspace\pxrr@zeropt
1532 \let\pxrr@aspace\pxrr@zeropt
1533 \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）。

```

1534 \pxrr@protrfalse
1535 \let\pxrr@bspace\pxrr@zeropt
1536 \let\pxrr@aspace\pxrr@zeropt
1537 \fi\fi

```

実際に組版を行う。

```

1538 \setbox\z@\hbox{%
1539   \ifnum\pxrr@side=\z@
1540     \raise\pxrr@ruby@raise\box\pxrr@boxr
1541   \else
1542     \lower\pxrr@ruby@lower\box\pxrr@boxr
1543   \fi
1544 }%
1545 \ifnum \ifpxrr@combo\pxrr@ck@ruby@combo\else\z@\fi >\z@
1546   \pxrr@ck@compose{#2}%
1547 \fi
1548 \ht\z@\z@ \dp\z@\z@
1549 \@tempdima\wd\z@
1550 \setbox\pxrr@boxr\hbox{%
1551   \box\z@
1552   \kern-\@tempdima
1553   \box\pxrr@boxa
1554 }%

```

\ifpxrr@any@protr を設定する。

```

1555 \ifpxrr@protr
1556   \pxrr@any@protrtrue
1557 \fi
1558 }

```

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

```
1559 \def\pxrr@compose@twoside@block{%
1560   \let\pxrr@compose@block@do\pxrr@compose@twoside@block@do
1561   \pxrr@compose@block@pre
1562 }
1563 \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。

```
1564   \setbox\pxrr@boxa\pxrr@hbox{#2}%
1565   \edef\pxrr@ck@body@natwd{\the\wd\pxrr@boxa}%
1566   \let\pxrr@ck@locate\pxrr@locate@inner
1567   \setbox\pxrr@boxr\pxrr@hbox{%
1568     \pxrr@use@ruby@font
1569     #3%
1570   }%
1571   \setbox\pxrr@boxb\pxrr@hbox{%
1572     \pxrr@use@ruby@font
1573     #4%
1574   }%
```

「何れかのルビが親文字列より長いか」を検査する。

```
1575   \@tempwafalse
1576   \@tempdima\wd\pxrr@boxr
1577   \advance\@tempdima-\wd\pxrr@boxa
1578   \ifdim\pxrr@epsilon<\@tempdima \@tempwatrue \fi
1579   \@tempdima\wd\pxrr@boxb
1580   \advance\@tempdima-\wd\pxrr@boxa
1581   \ifdim\pxrr@epsilon<\@tempdima \@tempwatrue \fi
```

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

```
1582   \if@tempwa
1583     \pxrr@protrtrue
```

「拡張肩付き」指定の場合、前側の突出を抑止する。

```
1584     \let\pxrr@locate@temp#1%
1585     \ifnum\pxrr@athead>\@ne
1586       \ifnum\pxrr@locate@temp=\pxrr@locate@inner
1587         \let\pxrr@locate@temp\pxrr@locate@head
1588         \fi
1589     \fi
1590     \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。

```
1591     \ifdim\wd\pxrr@boxr<\wd\pxrr@boxb
1592       \pxrr@compose@twoside@block@sub{#2}{#3}%
```

```

1593     \pxrr@boxr\pxrr@boxb
1594   \else
1595     \pxrr@compose@twoside@block@sub{#2}{#4}%
1596     \pxrr@boxb\pxrr@boxr
1597   \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。

```

1598   \else
1599   \pxrr@protrfalse

```

肩付きルビの場合は組み直しを行わない。

```

1600   \ifnum\pxrr@athead=\z@
1601     \@tempdima\wd\pxrr@boxa
1602     \advance\@tempdima-\wd\pxrr@boxr
1603     \ifdim\pxrr@epsilon<\@tempdima
1604       \pxrr@decompose{#3}%
1605       \edef\pxrr@natwd{\the\wd\pxrr@boxr}%
1606       \pxrr@evenspace@int{#1}\pxrr@boxr
1607       \pxrr@use@ruby@font{\wd\pxrr@boxa}%
1608       \pxrr@adjust@margin
1609     \fi
1610     \@tempdima\wd\pxrr@boxa
1611     \advance\@tempdima-\wd\pxrr@boxb
1612     \ifdim\pxrr@epsilon<\@tempdima
1613       \pxrr@decompose{#4}%
1614       \edef\pxrr@natwd{\the\wd\pxrr@boxb}%
1615       \pxrr@evenspace@int{#1}\pxrr@boxb
1616       \pxrr@use@ruby@font{\wd\pxrr@boxa}%
1617       \pxrr@adjust@margin
1618     \fi
1619   \fi

```

\pxrr@?space はゼロに設定する。

```

1620   \let\pxrr@bspace\pxrr@zeropt
1621   \let\pxrr@aspace\pxrr@zeropt
1622   \fi

```

実際に組版を行う。

```

1623   \setbox\z@\hbox{%
1624     \@tempdima\wd\pxrr@boxr
1625     \raise\pxrr@ruby@raise\box\pxrr@boxr
1626     \kern-\@tempdima
1627     \lower\pxrr@ruby@lower\box\pxrr@boxb
1628   }%
1629   \ifnum \ifpxrr@combo\pxrr@ck@ruby@combo\else\z@\fi >\z@
1630     \pxrr@ck@compose{#2}%
1631   \fi
1632   \ht\z@\z@ \dp\z@\z@
1633   \@tempdima\wd\z@

```

```

1634 \setbox\pxrr@boxr\hbox{%
1635   \box\z@
1636   \kern-\@tempdima
1637   \box\pxrr@boxa
1638 }%
1639 }

```

\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。

```
1640 \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。

```
1641 \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。

```

1642 \def\pxrr@compose@twoside@block@sub#1#2#3#4{%
1643   \pxrr@decompose{#1}%
1644   \edef\pxrr@natwd{\the\wd\pxrr@boxa}%
1645   \pxrr@evenspace@int\pxrr@locate@temp\pxrr@boxa\relax{\wd#4}%
1646   \@tempdima\wd#4%
1647   \advance\@tempdima-\pxrr@bspace\relax
1648   \advance\@tempdima-\pxrr@aspace\relax
1649   \edef\pxrr@body@wd{\the\@tempdima}%
1650   \advance\@tempdima-\wd#3%
1651   \ifdim\pxrr@epsilon<\@tempdima
1652     \edef\pxrr@restore@margin@values{%
1653       \edef\noexpand\pxrr@bspace{\pxrr@bspace}%
1654       \edef\noexpand\pxrr@aspace{\pxrr@aspace}%
1655     }%
1656     \pxrr@decompose{#2}%
1657     \edef\pxrr@natwd{\the\wd#3}%
1658     \pxrr@evenspace@int\pxrr@locate@temp#3%
1659     \pxrr@use@ruby@font{\pxrr@body@wd}%
1660     \pxrr@adjust@margin
1661     \pxrr@restore@margin@values
1662     \setbox#3\hbox{%
1663       \kern\pxrr@bspace\relax
1664       \box#3%
1665     }%
1666   \else
1667     \ifnum\pxrr@locate@temp=\pxrr@locate@head
1668       \@tempdima\z@
1669     \else\ifnum\pxrr@locate@temp=\pxrr@locate@inner

```

```

1670     \@tempdima.5\@tempdima
1671     \fi\fi
1672     \advance\@tempdima\pxrr@bbspace\relax
1673     \setbox#3\hbox{%
1674         \kern\@tempdima
1675         \box#3%
1676     }%
1677     \fi
1678 }
1679 %     \end{macrocode}
1680 % \end{macro}
1681 %
1682 % \begin{macro}{\pxrr@compose@block@pre}
1683 % |\pxrr@compose@block@pre{|\jmeta{パターン}}|}{|^A
1684 %r \jmeta{親文字}}|}{|\jmeta{ルビ 1}}|}{|\jmeta{ルビ 2}}|}{|\Means
1685 % 親文字列・ルビ文字列の加工を行う。
1686 % \Note 両側ルビ対応のため、ルビ用引数が 2 つある。
1687 %     \begin{macrocode}
1688 \def\pxrr@compose@block@pre{%
    f 指定時は小書き仮名の変換を施す。
1689     \pxrr@cond\ifnum\pxrr@fullsize>\z@\fi{%
1690         \pxrr@compose@block@pre@a
1691     }{%
1692         \pxrr@compose@block@pre@d
1693     }%
1694 }
1695 % {パターン}{親文字}{ルビ 1}{ルビ 2}
1696 \def\pxrr@compose@block@pre@a#1#2#3#4{%
1697     \def\pxrr@compose@block@tempa{#4}%
1698     \pxrr@transform@kana\pxrr@compose@block@tempa
1699     \expandafter\pxrr@compose@block@pre@b
1700     \expandafter{\pxrr@compose@block@tempa}{#1}{#2}{#3}%
1701 }
1702 % {ルビ 2}{パターン}{親文字}{ルビ 1}
1703 \def\pxrr@compose@block@pre@b#1#2#3#4{%
1704     \def\pxrr@compose@block@tempa{#4}%
1705     \pxrr@transform@kana\pxrr@compose@block@tempa
1706     \expandafter\pxrr@compose@block@pre@c
1707     \expandafter{\pxrr@compose@block@tempa}{#1}{#2}{#3}%
1708 }
1709 % {ルビ 1}{ルビ 2}{パターン}{親文字}
1710 \def\pxrr@compose@block@pre@c#1#2#3#4{%
1711     \pxrr@compose@block@pre@d{#3}{#4}{#1}{#2}%
1712 }
1713 \def\pxrr@compose@block@pre@d{%
1714     \pxrr@cond\ifnum\pxrr@evensp=\z@\fi{%
1715         \pxrr@compose@block@pre@e
1716     }{%

```

```

1717 \pxrr@compose@block@pre@f
1718 }%
1719 }
1720 % {パターン}{親文字}
1721 \def\pxrr@compose@block@pre@e#1#2{%
1722 \pxrr@compose@block@pre@f{#1}{#2}}%
1723 }
1724 \def\pxrr@compose@block@pre@f{%
1725 \pxrr@cond\ifnum\pxrr@reversp=\z@\fi{%
1726 \pxrr@compose@block@pre@g
1727 }{%
1728 \pxrr@compose@block@do
1729 }%
1730 }
1731 % {パターン}{親文字}{ルビ 1}{ルビ 2}
1732 \def\pxrr@compose@block@pre@g#1#2#3#4{%
1733 \pxrr@compose@block@do{#1}{#2}{#3}{#4}}%
1734 }
1735 \let\pxrr@compose@block@tempa\@undefined

```

4.14 命令の頑強化

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

※ `\CS` は「制御語」(制御記号でなく)である必要がある。

```

1736 \def\pxrr@add@protect#1{%
1737 \expandafter\pxrr@add@protect@a
1738 \csname\expandafter@gobble\string#1\space\endcsname#1%
1739 }
1740 \def\pxrr@add@protect@a#1#2{%
1741 \let#1=#2%
1742 \def#2{\pxrr@check@protect\protect#1}}%
1743 }
1744 \def\pxrr@check@protect{%
1745 \ifx\protect\@typeset@protect
1746 \expandafter@gobble
1747 \fi
1748 }

```

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。

```
1749 \let\pxrr@body@input\@empty
```

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

```
1750 \def\pxrr@prepare@fallback#1{%
1751   \pxrr@fatal@errorfalse
1752   \def\pxrr@body@input{#1}%
1753 }
```

`\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。

```
1754 \def\pxrr@fallback{%
1755   \pxrr@body@input
1756 }
```

`\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〉を展開する。

```
1757 \def\pxrr@if@alive{%
1758   \ifpxrr@fatal@error \expandafter\@gobble
1759   \else \expandafter\@firstofone
1760   \fi
1761 }
```

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。

```
1762 \def\pxrr@end@kinsoku{0}
```

`\pxrr@ruby@scan` 片側ルビ用の先読み処理。

```
1763 \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。

```
1764   \def\pxrr@tempc{%
1765     \edef\pxrr@end@kinsoku{the\pxrr@cntr}%
1766     \pxrr@do@proc{#1}{#2}%
1767   }%
1768   \pxrr@check@kinsoku\pxrr@tempc
1769 }
```


`\pxrr@truby@scan` 両側ルビ用の先読み処理。

```
1770 \def\pxrr@truby@scan#1#2#3{%
1771   \def\pxrr@tempc{%
1772     \edef\pxrr@end@kinsoku{\the\pxrr@cntr}%
1773     \pxrr@do@proc{#1}{#2}{#3}%
1774   }%
1775   \pxrr@check@kinsoku\pxrr@tempc
1776 }
```

`\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。

```
1777 \def\pxrr@check@kinsoku#1{%
1778   \let\pxrr@tempb#1%
1779   \futurelet\pxrr@token\pxrr@check@kinsoku@a
1780 }
1781 \def\pxrr@check@kinsoku@a{%
1782   \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。

```
1783   \ifpxrr@abody\else
1784     \ifnum\pxrr@cntr=\@ne
1785       \pxrr@cntr\tw@
1786     \fi
1787   \fi
1788   \ifcase\pxrr@cntr
1789     \pxrr@cntr\z@
1790     \expandafter\pxrr@tempb
1791   \or
1792     \pxrr@cntr\@MM
1793     \expandafter\pxrr@tempb
1794   \else
1795     \expandafter\pxrr@check@kinsoku@b
1796   \fi
1797 }
```

`\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。

```
1798 \def\pxrr@check@kinsoku@b#1{%
1799   \pxrr@check@kinsoku@c#1#1%
1800 }
1801 \def\pxrr@check@kinsoku@c#1{%
1802   \pxrr@get@prebreakpenalty\pxrr@cntr{‘#1}%
1803   \pxrr@tempb
1804 }
```

`\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。

```
1805 \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。)

```
1806 \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。

```
1807 \ifcat\noexpand##1\relax
1808 \pxrr@cntr\z@
1809 \else\ifcat\noexpand##1\noexpand~%
1810 \pxrr@cntr\z@
1811 \else\ifcat\noexpand##1A%
1812 \pxrr@cntr@ne
1813 \else\ifcat\noexpand##10%
1814 \pxrr@cntr@ne
1815 \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` 判定は色々と面倒な点があるので避ける。)

```
1816 \pxrr@cntr\z@
1817 \expandafter\pxrr@check@char@a\meaning##1#2\pxrr@nil
1818 \fi\fi\fi\fi
1819 }%
1820 \def\pxrr@check@char@a##1#2##2\pxrr@nil{%
1821 \ifcat @##10%
1822 \pxrr@cntr\tw@
1823 \fi
1824 }%
1825 }
```

規定の引数を用意して「定義マクロ」を呼ぶ。

```
1826 \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`。)

```
1827 \let\pxrr@auto@penalty\z@
```

`\pxrr@auto@icspace` 文字間の空き。寸法値マクロ。

```
1828 \let\pxrr@auto@icspace\pxrr@zeropt
```

`\pxrr@intr@amount` 進入の幅。寸法値マクロ。

```
1829 \let\pxrr@intr@amount\pxrr@zeropt
```

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

```
1830 \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。
1831 \ifpxrr@bnobr
1832 \let\pxrr@auto@penalty\@MM
1833 \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。
1834 \else
1835 \let\pxrr@auto@penalty\z@
1836 \if:\pxrr@bscomp
1837 \let\pxrr@auto@icspace\pxrr@iaiskip
1838 \else\if.\pxrr@bscomp
1839 \let\pxrr@auto@icspace\pxrr@zeropt
1840 \else
1841 \let\pxrr@auto@icspace\pxrr@iiskip
1842 \fi\fi
1843 \fi
1844 }
```

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

```
1845 \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。
1846 \if:\pxrr@bscomp\else
1847 \pxrr@bnobrtrue
1848 \fi
1849 \ifpxrr@bnobr
1850 \let\pxrr@auto@penalty\@MM
1851 \let\pxrr@auto@icspace\pxrr@zeropt
1852 \else
    この分岐は和欧文間空白挿入指定 (:) に限る。
1853 \let\pxrr@auto@penalty\z@
1854 \let\pxrr@auto@icspace\pxrr@iaiskip
1855 \fi
1856 }
```

4.17.1 前側進入処理

`\pxrr@intrude@head` 前側の進入処理。

```
1857 \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。)
1858 \ifpxrr@ghost\else
    実効の進入幅は \pxrr@bintr と \pxrr@bspace の小さい方。
1859 \let\pxrr@intr@amount\pxrr@bspace
```

```

1860 \ifdim\pxrr@bintr<\pxrr@intr@amount\relax
1861 \let\pxrr@intr@amount\pxrr@bintr
1862 \fi

```

\pxrr@auto@* の設定法は和文ルビと欧文ルビで処理が異なる。

```

1863 \ifpxrr@abody
1864 \pxrr@intrude@setauto@a
1865 \else
1866 \pxrr@intrude@setauto@j
1867 \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)。

```

1868 \ifpxrr@par@head
1869 \ifpxrr@bfintr
1870 \hskip-\pxrr@intr@amount\relax
1871 \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。

```

1872 \else
1873 % \penalty\pxrr@auto@penalty\relax
1874 \hskip-\pxrr@intr@amount\relax
1875 \hskip\pxrr@auto@icspace\relax
1876 \fi
1877 \fi
1878 }

```

\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。

```

1879 \def\pxrr@put@head@penalty{%
1880 \ifpxrr@ghost\else \ifpxrr@par@head\else
1881 \ifpxrr@abody
1882 \pxrr@intrude@setauto@a
1883 \else
1884 \pxrr@intrude@setauto@j
1885 \fi
1886 \ifnum\pxrr@auto@penalty=\z@\else
1887 \pxrr@canta\lastpenalty \unpenalty
1888 \advance\pxrr@canta\pxrr@auto@penalty\relax
1889 \penalty\pxrr@canta
1890 \fi
1891 \fi\fi
1892 }

```

4.17.2 後側進入処理

\pxrr@intrude@end 末尾での進入処理。

```

1893 \def\pxrr@intrude@end{%
1894   \ifpxrr@ghost\else

```

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

```

1895   \let\pxrr@intr@amount\pxrr@aspace
1896   \ifdim\pxrr@aintr<\pxrr@intr@amount\relax
1897     \let\pxrr@intr@amount\pxrr@aintr
1898   \fi

```

\pxrr@auto@* の設定法は和文ルビと欧文ルビで処理が異なる。

```

1899   \pxrr@csletcs{ifpxrr@bnobr}{ifpxrr@anobr}%
1900   \let\pxrr@bscomp\pxrr@ascomp
1901   \ifpxrr@abody
1902     \pxrr@intrude@setauto@a
1903   \else
1904     \pxrr@intrude@setauto@j
1905   \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。

```

1906   \ifnum\pxrr@auto@penalty=\z@
1907     \let\pxrr@auto@penalty\pxrr@end@kinsoku
1908   \fi
1909   \ifpxrr@afintr

```

段落末尾での進入を許す場合。

```

1910     \ifnum\pxrr@auto@penalty=\z@\else
1911       \penalty\pxrr@auto@penalty\relax
1912     \fi
1913     \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。

```

1914     \hskip\pxrr@auto@icspace\relax
1915     \ifnum\pxrr@auto@penalty=\z@\else
1916       \penalty\@MM
1917     \fi
1918   \else

```

段落末尾での進入を許さない場合。

```

1919     \@tempkipa-\pxrr@intr@amount\relax
1920     \advance\@tempkipa\pxrr@auto@icspace\relax
1921     \ifnum\pxrr@auto@penalty=\z@\else
1922       \penalty\pxrr@auto@penalty\relax
1923     \fi
1924     \hskip\@tempkipa
1925     \ifnum\pxrr@auto@penalty=\z@\else
1926       \penalty\@MM
1927     \fi

```

```

1928   \fi
1929   \fi
1930 }

```

4.18 メインです

4.18.1 エントリーポイント

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

```

1931 \AtBeginDocument{%
1932   \providecommand*\ruby{\jruby}%
1933 }
1934 \newcommand*\jruby{%
1935   \pxrr@jprologue
1936   \pxrr@trubyfalse
1937   \pxrr@ruby
1938 }

```

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

```

1939 \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。

```

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

```

`\truby` 和文両側ルビの公開命令。

```

1946 \newcommand*\truby{%
1947   \pxrr@jprologue
1948   \pxrr@trubytrue
1949   \pxrr@ruby
1950 }
1951 \pxrr@add@protect\truby

```

`\atruby` 欧文両側ルビの公開命令。

```

1952 \newcommand*\atruby{%
1953   \pxrr@aprologue
1954   \pxrr@trubytrue
1955   \pxrr@ruby
1956 }
1957 \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。

```

1958 \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 1959 \let\pxrr@option\@empty
                1960 \let\pxrr@exoption\@empty
```

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

```
\pxrr@do@scan 1961 \let\pxrr@do@proc\@empty
                1962 \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。

```
1963 \def\pxrr@ruby{%
1964   \@testopt\pxrr@ruby@a{}%
1965 }
1966 \def\pxrr@ruby@a[#1]{%
1967   \def\pxrr@option{#1}%
1968   \@testopt\pxrr@ruby@b{}%
1969 }
1970 \def\pxrr@ruby@b[#1]{%
1971   \def\pxrr@exoption{#1}%
1972   \ifpxrr@truby
1973     \let\pxrr@do@proc\pxrr@truby@proc
1974     \let\pxrr@do@scan\pxrr@truby@scan
1975   \else
1976     \let\pxrr@do@proc\pxrr@ruby@proc
1977     \let\pxrr@do@scan\pxrr@ruby@scan
1978   \fi
1979   \pxrr@ruby@c
1980 }
1981 \def\pxrr@ruby@c{%
1982   \ifpxrr@ghost
1983     \expandafter\pxrr@do@proc
1984   \else
1985     \expandafter\pxrr@do@scan
1986   \fi
1987 }
```

`\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。

```
1988 \def\pxrr@mode@is@switching{%
1989   \if M\pxrr@mode T%
1990   \else\if J\pxrr@mode T%
1991   \else F%
1992   \fi\fi T%
1993 }
```

`\pxrr@bind@param` “呼出時変数” へのコピーを行う。

```
1994 \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。

```
1995 \ifpxrr@apply@combo
1996   \pxrr@apply@combofalse
1997   \pxrr@combotrue
1998   \pxrr@ck@bind@param
1999 \else
2000   \pxrr@combofalse
2001 \fi
2002 \let\pxrr@c@ruby@font\pxrr@ruby@font
2003 \let\pxrr@c@size@ratio\pxrr@size@ratio
2004 \let\pxrr@c@inter@gap\pxrr@inter@gap
2005 }
```

`\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。

```
2006 \def\pxrr@ruby@proc#1#2{%
2007   \pxrr@prepare@fallback{#1}%
      フォントサイズの変数を設定して、
2008   \pxrr@bind@param
2009   \pxrr@assign@fsize
      オプションを解析する。
2010   \pxrr@parse@option\pxrr@option
      ルビ文字入力をグループ列に分解する。
2011   \pxrr@decompbar{#2}%
2012   \let\pxrr@ruby@list\pxrr@res
2013   \edef\pxrr@ruby@count{\the\pxrr@cntr}%
2014   \let\pxrr@sruby@list\relax
      親文字入力をグループ列に分解する。
2015   \pxrr@decompbar{#1}%
2016   \let\pxrr@body@list\pxrr@res
2017   \edef\pxrr@body@count{\the\pxrr@cntr}%
      安全モードに関する処理を行う。
2018   \ifpxrr@safe@mode
2019     \pxrr@setup@safe@mode
2020   \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。
2021   \if\pxrr@mode@is@switching
2022     \pxrr@resolve@mode
2023   \fi
2024 \ifpxrr@Debug
2025   \pxrr@debug@show@input
2026 \fi
      入力検査を行い、パスした場合は組版処理に進む。
2027   \pxrr@if@alive{%
```



```

2028 \if g\pxrr@mode
2029 \pxrr@ruby@check@g
2030 \pxrr@if@alive{%
2031 \ifnum\pxrr@body@count>\@ne
2032 \pxrr@ruby@main@mg
2033 \else
2034 \pxrr@ruby@main@g
2035 \fi
2036 }%
2037 \else
2038 \pxrr@ruby@check@m
2039 \pxrr@if@alive{\pxrr@ruby@main@m}%
2040 \fi
2041 }%

    後処理を行う。

2042 \pxrr@ruby@exit
2043 }

```

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

```

2044 \def\pxrr@truby@proc#1#2#3{%
2045 \pxrr@prepare@fallback{#1}%

    フォントサイズの変数を設定して、

2046 \pxrr@bind@param
2047 \pxrr@assign@fsize

    オプションを解析する。

2048 \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。

2049 \def\pxrr@all@input{#{1}{#2}{#3}}%

    入力文字列のグループ分解を行う。

2050 \pxrr@decompbar{#3}%
2051 \let\pxrr@sruby@list\pxrr@res
2052 \edef\pxrr@sruby@count{\the\pxrr@cntr}%
2053 \pxrr@decompbar{#2}%
2054 \let\pxrr@ruby@list\pxrr@res
2055 \edef\pxrr@ruby@count{\the\pxrr@cntr}%
2056 \pxrr@decompbar{#1}%
2057 \let\pxrr@body@list\pxrr@res
2058 \edef\pxrr@body@count{\the\pxrr@cntr}%

    安全モードに関する処理を行う。

2059 \ifpxrr@safe@mode
2060 \pxrr@setup@safe@mode
2061 \fi
2062 \if\pxrr@mode@is@switching
2063 \pxrr@resolve@mode

```

```

2064 \fi
2065 \ifpxrrDebug
2066 \pxrr@debug@show@input
2067 \fi

```

入力検査を行い、パスした場合は組版処理に進む。

```

2068 \pxrr@if@alive{%
2069 \if g\pxrr@mode
2070 \pxrr@ruby@check@tg
2071 \pxrr@if@alive{\pxrr@ruby@main@tg}%
2072 \else
2073 \pxrr@ruby@check@tm
2074 \pxrr@if@alive{\pxrr@ruby@main@tm}%
2075 \fi
2076 }%

```

後処理を行う。

```

2077 \pxrr@ruby@exit
2078 }

```

`\pxrr@setup@safe@mode` 安全モード用の設定。

```

2079 \def\pxrr@setup@safe@mode{%

```

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

```

2080 \let\pxrr@mode=g\relax
2081 \pxrr@unite@group\pxrr@body@list
2082 \def\pxrr@body@count{1}%
2083 \pxrr@unite@group\pxrr@ruby@list
2084 \def\pxrr@ruby@count{1}%
2085 \ifx\pxrr@sruby@list\relax\else
2086 \pxrr@unite@group\pxrr@sruby@list
2087 \def\pxrr@sruby@count{1}%
2088 \fi

```

“文字単位のスキャン”が必要な機能を無効にする。

```

2089 \chardef\pxrr@evensp\z@
2090 \chardef\pxrr@revensp\z@
2091 \chardef\pxrr@fullsize\z@
2092 }

```

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

```

2093 \def\pxrr@resolve@mode{%
2094 \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、

```

2095 \ifnum\pxrr@ruby@count=\@ne
2096 \let\pxrr@pre\pxrr@decompose
2097 \let\pxrr@post\relax

```

```

2098     \pxrr@body@list
2099     \ifnum\pxrr@cntr=\@ne\else
2100         \let\pxrr@mode=g%
2101     \fi
2102 \fi

```

それ以外はモノルビ・熟語ルビを選択する。

```

2103     \if M\pxrr@mode \let\pxrr@mode=m\fi
2104     \if J\pxrr@mode \let\pxrr@mode=j\fi
2105 \ifpxrrDebug
2106     \pxrr@debug@show@resolve@mode
2107 \fi

```

\pxrr@check@option で行っている調整をやり直す。

```

2108     \if g\pxrr@mode
2109         \chardef\pxrr@athead\z@
2110     \fi
2111     \if g\pxrr@mode\else
2112         \chardef\pxrr@evensp\@ne
2113     \fi
2114 \else
2115     \pxrr@fatal@bad@switching
2116 \fi
2117 }

```

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。

```

2118 \def\pxrr@ruby@check@g{%
2119     \ifnum\pxrr@body@count=\pxrr@ruby@count\relax
2120     \ifnum\pxrr@body@count=\@ne\else
2121         \ifpxrr@abody
2122             \pxrr@fatal@bad@movable
2123         \else\ifnum\pxrr@extra=\z@
2124             \pxrr@fatal@na@movable
2125         \fi\fi
2126     \fi
2127 \else
2128     \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
2129 \fi
2130 }

```

\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。

```

2131 \def\pxrr@ruby@check@m{%

```

```
2132 \ifnum\pxrr@body@count=\@ne
```

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

```
2133 \let\pxrr@pre\pxrr@decompose
2134 \let\pxrr@post\relax
2135 \pxrr@body@list
2136 \let\pxrr@body@list\pxrr@res
2137 \edef\pxrr@body@count{\the\pxrr@cntr}%
2138 \ifnum\pxrr@body@count=\pxrr@ruby@count\relax\else
2139 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
2140 \fi
2141 \else
2142 \pxrr@fatal@bad@mono
2143 \fi
2144 }
```

\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。

```
2145 \def\pxrr@ruby@check@tg{%
2146 \ifnum\pxrr@body@count=\pxrr@ruby@count\relax\else
2147 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
2148 \fi
2149 \ifnum\pxrr@body@count=\pxrr@sruby@count\relax\else
2150 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@sruby@count
2151 \fi
2152 \pxrr@if@alive{%
2153 \ifnum\pxrr@body@count=\@ne\else
2154 \ifpxrr@abody
2155 \pxrr@fatal@bad@movable
2156 \else\ifnum\pxrr@extra=\z@
2157 \pxrr@fatal@na@movable
2158 \fi\fi
2159 \fi
2160 }%
2161 }
```

\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。

```
2162 \def\pxrr@ruby@check@tm{%
2163 \ifnum\pxrr@body@count=\@ne
2164 \let\pxrr@pre\pxrr@decompose
2165 \let\pxrr@post\relax
2166 \pxrr@body@list
2167 \let\pxrr@body@list\pxrr@res
2168 \edef\pxrr@body@count{\the\pxrr@cntr}%
2169 \ifnum\pxrr@body@count=\pxrr@ruby@count\relax\else
2170 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
2171 \fi
2172 \ifnum\pxrr@body@count=\pxrr@sruby@count\relax\else
2173 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@sruby@count
2174 \fi
```

```

2175 \else
2176 \pxrr@fatal@bad@mono
2177 \fi
2178 }

```

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。

```

2179 \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。

```

2180 \def\pxrr@check@par@head{%
2181 \ifvmode
2182 \pxrr@par@headtrue
2183 \else
2184 \pxrr@par@headfalse
2185 \fi
2186 }

```

`\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。

```

2187 \def\pxrr@if@last#1#2#3{%
2188 \ifx#3\pxrr@post #1%
2189 \else #2%
2190 \fi
2191 #3%
2192 }

```

`\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。

```

2193 \def\pxrr@inter@mono{%
2194 \hskip\pxrr@iiskip\relax
2195 }

```

`\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。

```

2196 \def\pxrr@takeout@any@protr@nocolor{%
2197 \ifpxrr@any@protr
2198 \aftergroup\pxrr@any@protrtrue
2199 \fi
2200 }

```

`color` 使用時の定義。

```

2201 \def\pxrr@takeout@any@protr{%
2202   \ifpxrr@any@protr
2203     \aftergroup\pxrr@takeout@any@protr@a
2204   \fi
2205 }
2206 \def\pxrr@takeout@any@protr@a{%
2207   \aftergroup\pxrr@any@protrtrue
2208 }

```

\pxrr@ruby@main@m モノルビ。

```

2209 \def\pxrr@ruby@main@m{%
2210   \pxrr@zip@list\pxrr@body@list\pxrr@ruby@list
2211   \let\pxrr@whole@list\pxrr@res
2212   \pxrr@check@par@head
2213   \pxrr@put@head@penalty
2214   \pxrr@any@protrfalse
2215 \ifpxrr@Debug
2216 \pxrr@debug@show@recomp
2217 \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。

```

2218 \let\pxrr@locate@head@\pxrr@locate@inner
2219 \let\pxrr@locate@end@\pxrr@locate@inner
2220 \let\pxrr@locate@sing@\pxrr@locate@inner
2221 \ifpxrr@aprotr\else
2222   \let\pxrr@locate@end@\pxrr@locate@end
2223   \let\pxrr@locate@sing@\pxrr@locate@end
2224 \fi
2225 \ifpxrr@bprotr\else
2226   \let\pxrr@locate@head@\pxrr@locate@head
2227   \let\pxrr@locate@sing@\pxrr@locate@head
2228 \fi
2229 \def\pxrr@pre##1##2{%
2230   \pxrr@if@last{%

```

単独ブロックの場合。

```

2231   \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
2232   \pxrr@intrude@head
2233   \unhbox\pxrr@boxr
2234   \pxrr@intrude@end
2235   \pxrr@takeout@any@protr
2236   }{%

```

先頭ブロックの場合。

```

2237   \pxrr@compose@block\pxrr@locate@head@{##1}{##2}%
2238   \pxrr@intrude@head
2239   \unhbox\pxrr@boxr
2240   }%
2241   }%

```

```
2242 \def\pxrr@inter##1##2{%
2243   \pxrr@if@last{%
```

末尾ブロックの場合。

```
2244   \pxrr@compose@block\pxrr@locate@end@{##1}{##2}%
2245   \pxrr@inter@mono
2246   \unhbox\pxrr@boxr
2247   \pxrr@intrude@end
2248   \pxrr@takeout@any@protr
2249   }{%
```

中間ブロックの場合。

```
2250   \pxrr@compose@block\pxrr@locate@inner{##1}{##2}%
2251   \pxrr@inter@mono
2252   \unhbox\pxrr@boxr
2253   }%
2254 }%
2255 \let\pxrr@post\@empty
2256 \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。

```
2257 \if j\pxrr@mode
2258   \ifpxrr@any@protr
2259     \pxrr@ruby@redo@j
2260   \fi
2261 \fi
2262 \unhbox\pxrr@boxr
2263 }
```

\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。

```
2264 \def\pxrr@ruby@redo@j{%
2265   \pxrr@concat@list\pxrr@body@list
2266   \let\pxrr@body@list\pxrr@res
2267   \pxrr@concat@list\pxrr@ruby@list
2268   \let\pxrr@ruby@list\pxrr@res
2269   \pxrr@zip@single\pxrr@body@list\pxrr@ruby@list
2270   \let\pxrr@whole@list\pxrr@res
2271 \ifpxrr@Debug
2272 \pxrr@debug@show@concat
2273 \fi
2274 \let\pxrr@locate@sing@\pxrr@locate@inner
2275 \ifpxrr@aprotr\else
2276   \let\pxrr@locate@sing@\pxrr@locate@end
2277 \fi
2278 \ifpxrr@bprotr\else
2279   \let\pxrr@locate@sing@\pxrr@locate@head
2280 \fi
2281 \def\pxrr@pre##1##2{%
2282   \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
```

```

2283 \pxrr@intrude@head
2284 \unhbox\pxrr@boxr
2285 \pxrr@intrude@end
2286 }%
2287 \let\pxrr@inter\@undefined
2288 \let\pxrr@post\@empty
2289 \setbox\pxrr@boxr\pxrr@hbox{\pxrr@whole@list}%
2290 }

```

\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。

```

2291 \def\pxrr@ruby@main@g{%
2292 \pxrr@zip@list\pxrr@body@list\pxrr@ruby@list
2293 \let\pxrr@whole@list\pxrr@res
2294 \pxrr@check@par@head
2295 \pxrr@put@head@penalty
2296 \ifpxrr@Debug
2297 \pxrr@debug@show@recomp
2298 \fi
2299 \let\pxrr@locate@sing@\pxrr@locate@inner
2300 \ifpxrr@aprotr\else
2301 \let\pxrr@locate@sing@\pxrr@locate@end
2302 \fi
2303 \ifpxrr@bprotr\else
2304 \let\pxrr@locate@sing@\pxrr@locate@head
2305 \fi
2306 \def\pxrr@pre##1##2{%
2307 \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
2308 \pxrr@intrude@head
2309 \unhbox\pxrr@boxr
2310 \pxrr@intrude@end
2311 }%
2312 \let\pxrr@inter\@undefined
2313 \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。

```

2314 \pxrr@whole@list
2315 }

```

\pxrr@ruby@main@tm 両側のモノルビの場合。

```

2316 \def\pxrr@ruby@main@tm{%
2317 \pxrr@tzip@list\pxrr@body@list\pxrr@ruby@list\pxrr@sruby@list
2318 \let\pxrr@whole@list\pxrr@res
2319 \pxrr@check@par@head
2320 \pxrr@any@protrfalse
2321 \ifpxrr@Debug
2322 \pxrr@debug@show@recomp
2323 \fi

```



```

2324 \let\pxrr@locate@head@\pxrr@locate@inner
2325 \let\pxrr@locate@end@\pxrr@locate@inner
2326 \let\pxrr@locate@sing@\pxrr@locate@inner
2327 \ifpxrr@aprotr\else
2328   \let\pxrr@locate@end@\pxrr@locate@end
2329   \let\pxrr@locate@sing@\pxrr@locate@end
2330 \fi
2331 \ifpxrr@bprotr\else
2332   \let\pxrr@locate@head@\pxrr@locate@head
2333   \let\pxrr@locate@sing@\pxrr@locate@head
2334 \fi
2335 \def\pxrr@pre##1##2##3{%
2336   \pxrr@if@last{%
2337     \pxrr@compose@twoside@block\pxrr@locate@sing@
2338     {##1}{##2}{##3}%
2339     \pxrr@intrude@head
2340     \unhbox\pxrr@boxr
2341     \pxrr@intrude@end
2342     \pxrr@takeout@any@protr
2343   }{%
2344     \pxrr@compose@twoside@block\pxrr@locate@head@
2345     {##1}{##2}{##3}%
2346     \pxrr@intrude@head
2347     \unhbox\pxrr@boxr
2348   }%
2349 }%
2350 \def\pxrr@inter##1##2##3{%
2351   \pxrr@if@last{%
2352     \pxrr@compose@twoside@block\pxrr@locate@end@
2353     {##1}{##2}{##3}%
2354     \pxrr@inter@mono
2355     \unhbox\pxrr@boxr
2356     \pxrr@intrude@end
2357     \pxrr@takeout@any@protr
2358   }{%
2359     \pxrr@compose@twoside@block\pxrr@locate@inner
2360     {##1}{##2}{##3}%
2361     \pxrr@inter@mono
2362     \unhbox\pxrr@boxr
2363   }%
2364 }%
2365 \let\pxrr@post\@empty
2366 \setbox\pxrr@boxr\pxrr@hbox{\pxrr@whole@list}%
2367 \unhbox\pxrr@boxr
2368 }

```

\pxrr@ruby@main@tg 両側の単純グループビの場合。

```

2369 \def\pxrr@ruby@main@tg{%
2370   \pxrr@check@par@head

```

```

2371 \pxrr@put@head@penalty
2372 \let\pxrr@locate@sing@\pxrr@locate@inner
2373 \ifpxrr@aprotr\else
2374   \let\pxrr@locate@sing@\pxrr@locate@end
2375 \fi
2376 \ifpxrr@bprotr\else
2377   \let\pxrr@locate@sing@\pxrr@locate@head
2378 \fi
2379 \expandafter\pxrr@compose@twoside@block\expandafter\pxrr@locate@sing@
2380 \pxrr@all@input
2381 \pxrr@intrude@head
2382 \unhbox\pxrr@boxr
2383 \pxrr@intrude@end
2384 }

```

`\pxrr@ruby@main@mg` 未実装（呼出もない）。

```
2385 \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。

```
2386 \newif\ifpxrr@ghost
```

`\pxrr@jprologue` 和文ルビ用の開始処理。

```
2387 \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。

```

2388 \ifpxrr@jghost
2389   \pxrr@jghost@char
2390   \pxrr@inhibitglue
2391 \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。

```

2392 \begingroup
2393   \pxrr@abodyfalse
2394   \pxrr@csletcs{\ifpxrr@ghost}{\ifpxrr@jghost}%

```

出力した全角空白の幅だけ戻しておく。

```

2395 \ifpxrr@jghost
2396   \setbox\pxrr@boxa\hbox{\pxrr@jghost@char}%
2397   \kern-\wd\pxrr@boxa
2398 \fi
2399 }

```

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

いる。極小のサイズにしているのは、合成語記号の高さが影響する可能性を避けるためである。LM フォントの TeX フォント名は版により異なるようなので、NFSS を通じ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。)

```

2400 \chardef\pxrr@aghostchar=23 % compwordmark
2401 \let\pxrr@aghost\relax
2402 \let\pxrr@aghostfont\relax
2403 \def\pxrr@setup@aghost{%
2404   \global\let\pxrr@setup@aghost\relax
2405   \IfFileExists{t1lmr.fd}{%
2406     \begingroup
2407       \fontsize{2.5}{0}\usefont{T1}{lmr}{m}{n}%
2408     \endgroup
2409     \global\pxrr@letcs\pxrr@aghostfont{T1/lmr/m/n/2.5}%
2410     \gdef\pxrr@aghost{\pxrr@aghostfont\pxrr@aghostchar}%
2411     \pxrr@force@nonpunct@achar{\pxrr@aghostchar}%
2412   }{%else
2413     \pxrr@warn{Ghost embedding for \string\aruby\space
2414       is disabled,\MessageBreak
2415       since package lmodern is missing}%
2416     \global\pxrr@aghostfalse
2417     \global\let\pxrr@aghosttrue\relax
2418   }%
2419 }

```

`\pxrr@aprologue` 欧文ルビ用の開始処理。

```

2420 \def\pxrr@aprologue{%
2421   \ifpxrr@aghost
2422     \pxrr@aghost
2423     \fi
2424   \begingroup
2425     \pxrr@abodytrue
2426     \pxrr@csletcs{ifpxrr@ghost}{ifpxrr@aghost}%
2427 }

```

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。

```

2428 \def\pxrr@ruby@exit{%
2429   \ifpxrr@fatal@error
2430     \pxrr@fallback
2431     \fi
2432   \ifpxrr@abody
2433     \expandafter\pxrr@aepilogue
2434   \else

```

```

2435 \expandafter\pxrr@jepilogue
2436 \fi
2437 }

```

`\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。

```

2438 \def\pxrr@jepilogue{%
2439 \ifpxrr@jghost
2440 \setbox\pxrr@boxa\hbox{\pxrr@jghost@char}%
2441 \kern-\wd\pxrr@boxa
2442 \fi
2443 \endgroup
2444 \ifpxrr@jghost
2445 \pxrr@inhibitglue
2446 \pxrr@jghost@char
2447 \fi
2448 }

```

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

`\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。

```

2449 \def\pxrr@aepilogue{%
2450 \endgroup
2451 \ifpxrr@aghost
2452 \pxrr@aghost
2453 \fi
2454 }

```

4.19 デバッグ用出力

```

2455 \def\pxrr@debug@show@input{%
2456 \typeout{---\pxrr@pkgname\space input:^^J%
2457 \ifpxrr@abody = \meaning\ifpxrr@abody^^J%
2458 \ifpxrr@truby = \meaning\ifpxrr@truby^^J%
2459 pxrr@ruby@fsize = \pxrr@ruby@fsize^^J%
2460 pxrr@body@zw = \pxrr@body@zw^^J%
2461 pxrr@ruby@zw = \pxrr@ruby@zw^^J%
2462 pxrr@iiskip = \pxrr@iiskip^^J%
2463 pxrr@iaiskip = \pxrr@iaiskip^^J%
2464 pxrr@htratio = \pxrr@htratio^^J%
2465 pxrr@ruby@raise = \pxrr@ruby@raise^^J%
2466 pxrr@ruby@lower = \pxrr@ruby@lower^^J%
2467 \ifpxrr@bprotr = \meaning\ifpxrr@bprotr^^J%
2468 \ifpxrr@aprotr = \meaning\ifpxrr@aprotr^^J%
2469 pxrr@side = \the\pxrr@side^^J%
2470 pxrr@evensp = \the\pxrr@evensp^^J%
2471 pxrr@fullsize = \the\pxrr@fullsize^^J%
2472 pxrr@bscomp = \meaning\pxrr@bscomp^^J%
2473 pxrr@ascomp = \meaning\pxrr@ascomp^^J%
2474 \ifpxrr@bnoobr = \meaning\ifpxrr@bnoobr^^J%

```

```

2475 ifpxrr@anobr = \meaning\ifpxrr@anobr^^J%
2476 ifpxrr@bfintr = \meaning\ifpxrr@bfintr^^J%
2477 ifpxrr@afintr = \meaning\ifpxrr@afintr^^J%
2478 pxrr@bintr = \pxrr@bintr^^J%
2479 pxrr@aintr = \pxrr@aintr^^J%
2480 pxrr@ahead = \the\pxrr@ahead^^J%
2481 pxrr@mode = \meaning\pxrr@mode^^J%
2482 ifpxrr@ahead@given = \meaning\ifpxrr@ahead@given^^J%
2483 ifpxrr@mode@given = \meaning\ifpxrr@mode@given^^J%
2484 pxrr@body@list = \meaning\pxrr@body@list^^J%
2485 pxrr@body@count = \@nameuse{pxrr@body@count}^^J%
2486 pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
2487 pxrr@ruby@count = \@nameuse{pxrr@ruby@count}^^J%
2488 pxrr@end@kinsoku = \pxrr@end@kinsoku^^J%
2489 ----
2490 }%
2491 }
2492 \def\pxrr@debug@show@recomp{%
2493 \typeout{----\pxrr@pkgname\space recomp:^^J%
2494 pxrr@body@list = \meaning\pxrr@body@list^^J%
2495 pxrr@body@count = \pxrr@body@count^^J%
2496 pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
2497 pxrr@ruby@count = \pxrr@ruby@count^^J%
2498 pxrr@res = \meaning\pxrr@res^^J%
2499 ----
2500 }%
2501 }
2502 \def\pxrr@debug@show@concat{%
2503 \typeout{----\pxrr@pkgname\space concat:^^J%
2504 pxrr@body@list = \meaning\pxrr@body@list^^J%
2505 pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
2506 pxrr@whole@list = \meaning\pxrr@whole@list^^J%
2507 ----
2508 }%
2509 }
2510 \def\pxrr@debug@show@resolve@mode{%
2511 \typeout{----\pxrr@pkgname\space resolve-mode:
2512 \meaning\pxrr@mode}%
2513 }

```

5 実装（圈点関連）

5.1 エラーメッセージ

指定の名前の圈点文字が未登録の場合。

```

2514 \def\pxrr@warn@na@kmark#1{%
2515 \pxrr@warn{Unavailable kenten mark '#1'}%
2516 }

```

パラメタ設定命令で無効な値が指定された場合。

```
2517 \def\pxrr@err@invalid@value#1{%
2518   \pxrr@error{Invalid value '#1'}%
2519   {\@eha}%
2520 }
```

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。

```
2521 \let\pxrr@k@ymark\@undefined
```

`\pxrr@k@ysmark` 横組の副の圏点マークのコード。

```
2522 \let\pxrr@k@ysmark\@undefined
```

`\pxrr@k@tmark` 縦組の主の圏点マークのコード。

```
2523 \let\pxrr@k@tmark\@undefined
```

`\pxrr@k@tsmark` 縦組の副の圏点マークのコード。

```
2524 \let\pxrr@k@tsmark\@undefined
```

圏点マークの初期値の設定。

```
2525 \AtEndOfPackage{%
2526   \pxrr@k@get@mark\pxrr@k@ymark{bullet*}%
2527   \pxrr@k@get@mark\pxrr@k@ysmark{sesame*}%
2528   \pxrr@k@get@mark\pxrr@k@tmark{sesame*}%
2529   \pxrr@k@get@mark\pxrr@k@tsmark{bullet*}%
2530 }
```

`\pxrr@k@ruby@font` 圏点用フォント切替命令。

```
2531 \let\pxrr@k@ruby@font\@empty
```

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

```
2532 \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。

```
2533 \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。

```
2534 \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。

```
2535 \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。

```
2536 \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。

```
2537 \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。

```
2538 \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。

```
2539 \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。

```
2540 \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。

```
2541 \chardef\pxrr@k@full=0
```

`\pxrr@k@the@mark` 適用される圏点マークの命令。

```
2542 \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。

```
2543 \newif\ifpxrr@avail@UTF
```

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

```
2544 \def\pxrr@decide@avail@UTF{%
2545   \global\let\pxrr@decide@avail@UTF\relax
2546   \ifx\UTF\undefined \global\pxrr@avail@UTFfalse
2547   \else \global\pxrr@avail@UTFtrue
2548   \fi
2549 }
```

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
```

```
2550 \def\pxrr@k@decompose#1{%
```

```
2551   \let\pxrr@res\@empty
```

```

2552 \pxrr@cntr=\z@
2553 \pxrr@k@decompose@loopa#1\pxrr@end
2554 }
2555 \def\pxrr@k@decompose@loopa{%
2556 \futurelet\pxrr@token\pxrr@k@decompose@loopb
2557 }
2558 \def\pxrr@k@decompose@loopb{%
2559 \pxrr@cond\ifx\pxrr@token\pxrr@end\fi{%
2560 \pxrr@appto\pxrr@res{\pxrr@post}%
2561 }{\pxrr@if@kspan@cmd\pxrr@token{%
2562 \pxrr@k@decompose@special\pxrr@k@decompose@kspan
2563 }{\pxrr@if@ruby@cmd\pxrr@token{%
2564 \pxrr@k@decompose@special\pxrr@k@decompose@ruby
2565 }{\pxrr@if@truby@cmd\pxrr@token{%
2566 \pxrr@k@decompose@special\pxrr@k@decompose@truby
2567 }{\pxrr@if@kenten@cmd\pxrr@token{%
2568 \pxrr@k@decompose@special\pxrr@k@decompose@kenten
2569 }{\pxrr@cond\ifx\pxrr@token\@sptoken\fi{%
2570 \pxrr@k@decompose@loope
2571 }{%
2572 \pxrr@setok{\pxrr@ifx{\pxrr@token\bgroup}}}%
2573 \pxrr@k@decompose@loopc
2574 }}}}]}%
2575 }
2576 \def\pxrr@k@decompose@loopc#1{%
2577 \pxrr@appto\pxrr@res{\pxrr@entry}%
2578 \ifpxrr@ok
2579 \pxrr@appto\pxrr@res{{{#1}}}%
2580 \else
2581 \pxrr@appto\pxrr@res{#{#1}}%
2582 \fi
2583 \pxrr@k@decompose@loopd
2584 }
2585 \def\pxrr@k@decompose@loopd{%
2586 \advance\pxrr@cntr\@ne
2587 \pxrr@k@decompose@loopa
2588 }
2589 \expandafter\def\expandafter\pxrr@k@decompose@loope\space{%
2590 \pxrr@okfalse
2591 \pxrr@k@decompose@loopc{ }%
2592 }
2593 \def\pxrr@k@decompose@special#1#2#3{%
2594 #1{#2}%
2595 }
2596 \def\pxrr@k@decompose@kspan#1#2{%
2597 \pxrr@appto\pxrr@res{\pxrr@entry@kspan{#1{#2}}}%
2598 \pxrr@k@decompose@loopd
2599 }
2600 \def\pxrr@k@decompose@ruby#1#2#3{%

```



```

2601 \pxrr@appto\pxrr@res{\pxrr@entry@ruby{#1{#2}{#3}}}%
2602 \pxrr@k@decompose@loopd
2603 }
2604 \def\pxrr@k@decompose@truby#1#2#3#4{%
2605 \pxrr@appto\pxrr@res{\pxrr@entry@ruby{#1{#2}{#3}{#4}}}%
2606 \pxrr@k@decompose@loopd
2607 }
2608 \def\pxrr@k@decompose@kenten#1#2{%
2609 \pxrr@appto\pxrr@res{\pxrr@entry@kenten{#1{#2}}}%
2610 \pxrr@k@decompose@loopd
2611 }
2612 \def\pxrr@cmd@ruby{\jruby}
2613 \def\pxrr@cmd@kenten{jkenten}
2614 \def\pxrr@if@ruby@cmd#1{%
2615 \if \ifcat\noexpand#1\relax
2616 \ifx#1\pxrr@cmd@ruby T%
2617 \else\ifx#1\jruby T%
2618 \else\ifx#1\aruby T%
2619 \else F%
2620 \fi\fi\fi
2621 \else F%
2622 \fi T\expandafter\@firstoftwo
2623 \else \expandafter\@secondoftwo
2624 \fi
2625 }
2626 \def\pxrr@if@truby@cmd#1{%
2627 \if \ifcat\noexpand#1\relax
2628 \ifx#1\truby T%
2629 \else\ifx#1\atruby T%
2630 \else F%
2631 \fi\fi
2632 \else F%
2633 \fi T\expandafter\@firstoftwo
2634 \else \expandafter\@secondoftwo
2635 \fi
2636 }
2637 \def\pxrr@if@kspan@cmd#1{%
2638 \pxrr@cond\ifx#1\kspan\fi
2639 }
2640 \def\pxrr@if@kenten@cmd#1{%
2641 \if \ifcat\noexpand#1\relax
2642 \ifx#1\pxrr@cmd@kenten T%
2643 \else\ifx#1\jkenten T%
2644 \else F%
2645 \fi\fi
2646 \else F%
2647 \fi T\expandafter\@firstoftwo
2648 \else \expandafter\@secondoftwo
2649 \fi

```

2650 }

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。

```
2651 \newcommand*\kentensetup[1]{%
2652   \pxrr@in@setuptrue
2653   \pxrr@fatal@errorfalse
2654   \pxrr@k@parse@option{#1}%
2655   \ifpxrr@fatal@error\else
2656     \let\pxrr@k@d@side\pxrr@side
2657     \let\pxrr@k@d@mark\pxrr@k@mark
2658     \let\pxrr@k@d@full\pxrr@k@full
2659   \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。

```
2660   \pxrr@in@setupfalse
2661 }
```

`\kentenfontsetup` 対応するパラメタを設定する。

```
2662 \newcommand*\kentenfontsetup{}
2663 \def\kentenfontsetup#{%
2664   \def\pxrr@k@ruby@font
2665 }
```

`\kentensizeratio` 対応するパラメタを設定する。

```
2666 \newcommand*\kentensizeratio[1]{%
2667   \edef\pxrr@k@size@ratio{#1}%
2668 }
```

`\kentenintergap` 対応するパラメタを設定する。

```
2669 \newcommand*\kentenintergap[1]{%
2670   \edef\pxrr@k@inter@gap{#1}%
2671 }
```

`\kentenrubyintergap` 対応するパラメタを設定する。

```
2672 \newcommand*\kentenrubyintergap[1]{%
2673   \edef\pxrr@k@ruby@inter@gap{#1}%
2674 }
```

`\kentenmarkinyoko` 対応するパラメタを設定する。

```
\kentenmarkinyoko 2675 \newcommand*\kentenmarkinyoko[1]{%
\kentenmarkintate 2676   \pxrr@k@get@mark\pxrr@k@ymark{#1}%
2677 }
\kentenmarkintate 2678 \newcommand*\kentenmarkintate[1]{%
2679   \pxrr@k@get@mark\pxrr@k@ysmark{#1}%
2680 }
```

```

2681 \newcommand*\kentenmarkintate[1]{%
2682   \pxrr@k@get@mark\pxrr@k@tmark{#1}%
2683 }
2684 \newcommand*\kentensubmarkintate[1]{%
2685   \pxrr@k@get@mark\pxrr@k@tsmark{#1}%
2686 }

```

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

```

2687 \chardef\pxrr@k@ruby@combo@ruby=0
2688 \chardef\pxrr@k@ruby@combo@both=1
2689 \newcommand*\kentenrubycombination[1]{%
2690   \pxrr@letcs\pxrr@tempa{\pxrr@k@ruby@combo@#1}%
2691   \ifx\pxrr@tempa\relax
2692     \pxrr@err@invalid@value{#1}%
2693   \else
2694     \let\pxrr@k@ruby@combo\pxrr@tempa
2695   \fi
2696 }

```

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。

```

2697 \def\pxrr@k@declare@mark#1{%
2698   \global\@namedef{\pxrr@k@mark@#1}%
2699 }

```

`\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。

```

2700 \def\pxrr@k@let@mark#1{%
2701   \global\pxrr@cslet{\pxrr@k@mark@#1}%
2702 }

```

`\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。

```

2703 \def\pxrr@k@get@mark#1#2{%
2704   \futurelet\pxrr@token\pxrr@k@get@mark@a#2\pxrr@nil#1%
2705 }
2706 \def\pxrr@k@get@mark@a{%
2707   \pxrr@cond@ifcat A\noexpand\pxrr@token\fi{%
2708     \pxrr@k@get@mark@c
2709   }{%else
2710     \pxrr@k@get@mark@b
2711   }%
2712 }
2713 \def\pxrr@k@get@mark@b#1\pxrr@nil#2{%
2714   \def#2{#1}%
2715 }

```

```

2716 \def\pxrr@k@get@mark@c#1#2\pxrr@nil#3{%
2717   \ifnum'#1<128
2718     \pxrr@letcs\pxrr@tempa{\pxrr@k@mark@c#1#2}%
2719     \ifx\pxrr@tempa\relax
2720       \pxrr@warn@na@kmark{#1#2}%
2721     \else
2722       \let#3\pxrr@tempa
2723     \fi
2724 \else
2725   \pxrr@k@get@mark@b#1#2\pxrr@nil#3%
2726 \fi
2727 }

```

`\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。

```

2728 \def\pxrr@k@declare@mark@char#1#2{%
2729   \pxrr@k@declare@mark@char@a{#1}#2\pxrr@end
2730 }
2731 \def\pxrr@k@declare@mark@char@a#1#2:#3\pxrr@end{%
2732   \pxrr@jchardef\pxrr@tempa\pxrr@jc{#2:#3}%
2733   \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。

```

2734     \pxrr@k@declare@mark{#1}{\pxrr@k@char@UTF{#1}{#3}}%
2735   \else
2736     \pxrr@k@let@mark{#1}\pxrr@tempa
2737   \fi
2738 }

```

`\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。

```

2739 \def\pxrr@k@char@UTF#1#2{%
2740   \pxrr@decide@avail@UTF
2741   \ifpxrr@avail@UTF
2742     \pxrr@k@declare@mark{#1}{\UTF{#2}}%
2743     \UTF{#2}%
2744   \else
2745     \pxrr@k@let@mark{#1}\@empty
2746     \pxrr@warn@na@kmark{#1}%
2747   \fi
2748 }

```

標準サポートの圏点マークの定義。

```

2749 \pxrr@k@declare@mark@char{bullet} {2121:2022}
2750 \pxrr@k@declare@mark@char{triangle}{2225:25B2}
2751 \pxrr@k@declare@mark@char{Triangle}{2224:25B3}
2752 \pxrr@k@declare@mark@char{fisheye} {2121:25C9}
2753 \pxrr@k@declare@mark@char{Circle} {217B:25CB}

```

```

2754 \pxrr@k@declare@mark@char{bullseye}{217D:25CE}
2755 \pxrr@k@declare@mark@char{circle} {217C:25CF}
2756 \pxrr@k@declare@mark@char{Bullet} {2121:25E6}
2757 \pxrr@k@declare@mark@char{sesame} {2121:FE45}
2758 \pxrr@k@declare@mark@char{Sesame} {2121:FE46}
2759 \pxrr@jchardef\pxrr@ja@dot=\pxrr@jc{2126:30FB}
2760 \pxrr@jchardef\pxrr@ja@comma=\pxrr@jc{2122:3001}
2761 \pxrr@k@declare@mark{bullet*}{%
2762   \pxrr@dima=\pxrr@ruby@zw\relax
2763   \hb@xt@\pxrr@dima{%
2764     \kern-.5\pxrr@dima
2765     \pxrr@if@in@tate{ }\lower.38\pxrr@dima}%
2766     \hb@xt@2\pxrr@dima{%
2767       \pxrr@dima=\f@size\p@
2768       \fontsize{2\pxrr@dima}{\z@}\selectfont
2769       \hss
2770       \pxrr@ja@dot
2771       \hss
2772     }%
2773     \hss
2774   }%
2775 }
2776 \pxrr@k@declare@mark{sesame*}{%
2777   \pxrr@dima=\pxrr@ruby@zw\relax
2778   \hb@xt@\pxrr@dima{%
2779     \pxrr@if@in@tate{\kern.1\pxrr@dima}{\kern.05\pxrr@dima}%
2780     \pxrr@if@in@tate{\lower.85\pxrr@dima}{\raise.3\pxrr@dima}%
2781     \hbox{%
2782       \pxrr@dima=\f@size\p@
2783       \fontsize{2.4\pxrr@dima}{\z@}\selectfont
2784       \pxrr@ja@comma
2785     }%
2786     \hss
2787   }%
2788 }

```

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。

```

2789 \def\pxrr@k@parse@option#1{%
2790   \edef\pxrr@tempa{#1}%
2791   \let\pxrr@side\pxrr@k@d@side
2792   \let\pxrr@k@mark\pxrr@k@d@mark
2793   \let\pxrr@k@full\pxrr@k@d@full
2794   \expandafter\pxrr@k@parse@option@loop\pxrr@tempa @\pxrr@end
2795 }
2796 \def\pxrr@k@parse@option@loop#1{%

```

圏点オプションの解析器は“有限状態”を持たないので非常に単純である。

```
2797 \pxrr@letcs\pxrr@tempa{pxrr@k@po@PR@#1}%
2798 \pxrr@cond\ifx\pxrr@tempa\relax\fi{%
2799   \pxrr@fatal@knx@letter{#1}%
2800   \pxrr@k@parse@option@exit
2801 }{%
2802   \pxrr@tempa
2803   \pxrr@k@parse@option@loop
2804 }%
2805 }
2806 \def\pxrr@k@parse@option@exit#1\pxrr@end{%
2807   \ifpxrr@in@setup\else
2808     \pxrr@k@check@option
```

ここで `\pxrr@k@the@mark` を適切に定義する。

```
2809   \pxrr@if@in@tate{%
2810     \ifcase\pxrr@k@mark \let\pxrr@k@the@mark\pxrr@k@tmark
2811     \or \let\pxrr@k@the@mark\pxrr@k@tsmark
2812     \fi
2813   }{%
2814     \ifcase\pxrr@k@mark \let\pxrr@k@the@mark\pxrr@k@ymark
2815     \or \let\pxrr@k@the@mark\pxrr@k@ysmark
2816     \fi
2817   }%
2818 \fi
2819 }
2820 \def\pxrr@k@po@PR@{%
2821   \pxrr@k@parse@option@exit
2822 }
2823 \def\pxrr@k@po@PR@P{%
2824   \chardef\pxrr@side\z@
2825 }
2826 \def\pxrr@k@po@PR@S{%
2827   \chardef\pxrr@side\@ne
2828 }
2829 \def\pxrr@k@po@PR@p{%
2830   \chardef\pxrr@k@mark\z@
2831 }
2832 \def\pxrr@k@po@PR@s{%
2833   \chardef\pxrr@k@mark\@ne
2834 }
2835 \def\pxrr@k@po@PR@F{%
2836   \chardef\pxrr@k@full\z@
2837 }
2838 \def\pxrr@k@po@PR@f{%
2839   \chardef\pxrr@k@full\@ne
2840 }
```

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。

```
2841 \def\pxrr@k@check@option{%
2842 }
```

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。

```
2843 \def\pxrr@k@compose@block#1#2{%
2844   \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。

```
2845   \pxrr@k@make@rep@list{\copy\pxrr@boxb}{#2}%
2846   \let\pxrr@natwd\pxrr@zeropt
2847   \pxrr@evenspace@int\pxrr@locate@inner\pxrr@boxr
2848     \relax{\wd\pxrr@boxa}%
2849   \setbox\z@\hbox{%
2850     \ifnum\pxrr@side=\z@
2851       \raise\pxrr@ruby@raise\box\pxrr@boxr
2852     \else
2853       \lower\pxrr@ruby@lower\box\pxrr@boxr
2854     \fi
2855   }%
2856   \ht\z@\z@ \dp\z@\z@
2857   \@tempdima\wd\z@
2858   \setbox\pxrr@boxr\hbox{%
2859     \box\z@
2860     \kern-\@tempdima
2861     \box\pxrr@boxa
2862   }%
2863 }
```

`\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。

```
2864 \def\pxrr@k@make@rep@list#1#2{%
2865   \def\pxrr@res{\pxrr@pre{#1}}%
2866   \pxrr@cntr=#2\relax
2867   \ifnum\pxrr@cntr>\@ne
```

```

2868 \@tempcnta\pxrr@cntr \advance\@tempcnta\m@ne
2869 \@whilenum{\@tempcnta>\z}\do{%
2870 \pxrr@appto\pxrr@res{\pxrr@inter{#1}}%
2871 \advance\@tempcnta\m@ne
2872 }%
2873 \fi
2874 \pxrr@appto\pxrr@res{\pxrr@post}%
2875 }

```

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@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。
2876 \newif\ifpxrr@k@first@entry

```

```

\ifpxrr@k@last@entry  末尾の項目であるか。
2877 \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。
2878 \newif\ifpxrr@k@prev@is@block

```

```

\pxrr@k@accum@res  累積の直接出力。
2879 \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。
2880 \mathchardef\pxrr@k@prebreakpenalty\z@

```

```

\pxrr@k@postbreakpenalty  圏点項目の後禁則ペナルティ。
2881 \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。
2882 \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。

```
2883 \def\pxrr@k@list@pre{%
2884   \pxrr@k@first@entrytrue
2885   \pxrr@k@last@entryfalse
2886   \pxrr@k@prev@is@blockfalse
2887   \let\pxrr@k@accum@res\@empty
2888   \chardef\pxrr@k@block@seq@state\z@
2889 }
```

`\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。

```
2890 \def\pxrr@k@entry@with#1#2{%
2891   \pxrr@if@last{%
2892     \pxrr@k@last@entrytrue
2893     \pxrr@k@entry@with@a#1{#2}%
2894   }{%
2895     \pxrr@k@entry@with@a#1{#2}%
2896   }%
2897 }
2898 \def\pxrr@k@entry@with@a#1#2{%
2899   \mathchardef\pxrr@k@prebreakpenalty\z@
2900   \mathchardef\pxrr@k@postbreakpenalty\z@
```

下請けマクロを実行して結果を得る。

```
2901   #1{#2}%
2902   \%typeout{%
2903   %first=\meaning\ifpxrr@k@first@entry^^J%
2904   %last=\meaning\ifpxrr@k@last@entry^^J%
2905   %prev=\meaning\ifpxrr@k@prev@is@block^^J%
2906   %res=\meaning\pxrr@res^^J%
2907   %type=\meaning\pxrr@k@entry@res@type^^J%
2908   %prepen=\the\pxrr@k@prebreakpenalty^^J%
2909   %postpen=\the\pxrr@k@postbreakpenalty}%
```

累積直接出力の処理。

```
2910   \ifnum\pxrr@k@entry@res@type=\z@
2911     \expandafter\pxrr@appto\expandafter\pxrr@k@accum@res
2912     \expandafter{\pxrr@res}%
2913   \else
2914     \pxrr@k@accum@res
2915     \let\pxrr@k@accum@res\@empty
2916   \fi
```

前禁則ペナルティを入れる。

```
2917   \ifnum\pxrr@k@prebreakpenalty>\z@
2918     \@tempcntb\lastpenalty \unpenalty
2919     \advance\@tempcntb\pxrr@k@prebreakpenalty
2920     \penalty\@tempcntb
2921   \fi
```

圏点ブロックが連続する場合は和文間空白を入れる。

```
2922 \ifnum\pxrr@k@entry@res@type=\tw@
2923   \ifpxrr@k@prev@is@block
2924     \pxrr@inter@mono
2925   \fi
2926   \pxrr@k@prev@is@blocktrue
2927 \else
2928   \pxrr@k@prev@is@blockfalse
2929 \fi
```

ボックスの結果を実際に出力する。

```
2930 \ifnum\pxrr@k@entry@res@type>\z@
2931   \unhbox\pxrr@boxr
2932 \fi
```

後禁則ペナルティを入れる。

```
2933 \ifnum\pxrr@k@postbreakpenalty>\z@
2934   \penalty\pxrr@k@postbreakpenalty
2935 \fi
```

次の項目に進む。

```
2936 \pxrr@k@first@entryfalse
2937 }
```

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

```
2938 \def\pxrr@k@list@post{%
2939   \pxrr@k@accum@res
2940   \let\pxrr@k@accum@res\@empty
2941 }
```

`\pxrr@k@kenten@entry` 一般の《文字》を表す圏点項目 `\pxrr@entry{⟨文字⟩}` の処理。圏点を1つ付けて出力する。

```
2942 \def\pxrr@k@kenten@entry{%
2943   \pxrr@k@entry@with\pxrr@k@kenten@entry@
2944 }
2945 \def\pxrr@k@kenten@entry@#1{%
2946   \pxrr@k@check@char{#1}%
2947   \ifpxrr@ok
2948     \pxrr@k@compose@block{#1}\@ne
2949     \chardef\pxrr@k@entry@res@type=\tw@
2950   \else
2951     \def\pxrr@res{#1}%
2952     \chardef\pxrr@k@entry@res@type=\z@
2953   \fi
2954 }
```

`\pxrr@k@kenten@entry@kspan` `\kspan` 命令を表す圏点項目 `\pxrr@entry@kspan{⟨kspan{⟨テキスト⟩}}}` の処理。テキストの幅が“およそ n 全角”である場合に、 n 個の圏点をルビ均等割りで配置して出力する。

```
2955 \def\pxrr@k@kenten@entry@kspan{%
2956   \pxrr@k@entry@with\pxrr@k@kenten@entry@kspan@
```

```

2957 }
2958 \def\pxrr@kenten@entry@kspan@#1{%
2959   \pxrr@kenten@entry@kspan@a#1%
2960 }
2961 \def\pxrr@kenten@entry@kspan@a#1{%
   \kspan (= #1) が * 付かを調べる。
2962   \@ifstar{%
2963     \@testopt\pxrr@kenten@entry@kspan@c{%}
2964   }{%
2965     \@testopt\pxrr@kenten@entry@kspan@b{%}
2966   }%
2967 }
2968 \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。
2969   \setbox\z@\pxrr@hbox{#2}%
2970   \@tempdima\pxrr@body@zw\relax
2971   \@tempdimb\wd\z@ \advance\@tempdimb.25\@tempdima
2972   \divide\@tempdimb\@tempdima
2973   \edef\pxrr@kenten@entry@tempa{\number\@tempdimb}%
2974   \pxrr@k@compose@block{#2}\pxrr@kenten@entry@tempa
2975   \chardef\pxrr@k@entry@res@type=\tw@
2976 }
2977 \def\pxrr@kenten@entry@kspan@c[#1]#2{%
   \kspan* となっている場合。この時は圈点を付加せず直接出力する。
2978   \def\pxrr@res{#2}%
2979   \chardef\pxrr@k@entry@res@type=\z@
2980 }

```

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

```

2981 \def\pxrr@kenten@entry@kenten{%
2982   \pxrr@k@entry@with\pxrr@kenten@entry@kenten@
2983 }
2984 \def\pxrr@kenten@entry@kenten@#1{%
   この場合は圈点ブロックとは見なさないことに注意。
2985   \setbox\pxrr@boxr\hbox{#1}%
2986   \chardef\pxrr@k@entry@res@type=\@ne
2987 }

```

`\pxrr@kenten@entry@ruby` ルビ命令の圈点項目。

```

2988 \def\pxrr@kenten@entry@ruby{%
2989   \pxrr@k@entry@with\pxrr@kenten@entry@ruby@
2990 }
2991 \def\pxrr@kenten@entry@ruby@#1{%
2992   \pxrr@apply@combotrue
2993   \setbox\pxrr@boxr\hbox{#1}%

```

```
2994 \chardef\pxrr@k@entry@res@type=\@ne
2995 }
```

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。

```
2996 \newcommand*\kspan{%
2997   \@ifstar{%
2998     \@testopt\pxrr@kspan@a{}%
2999   }{%
3000     \@testopt\pxrr@kspan@a{}%
3001   }%
3002 }
3003 \pxrr@add@protect\kspan
3004 \def\pxrr@kspan@a[#1]#2{%
3005   \begingroup
3006     #2%
3007   \endgroup
3008 }
```

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）。

```
3009 \def\pxrr@k@check@char#1{%
3010   \futurelet\pxrr@token\pxrr@k@check@char@a#1\pxrr@end
3011 }
3012 \def\pxrr@k@check@char@a#1\pxrr@end{%
3013   \pxrr@cond\ifx\pxrr@token\bgroup\fi%
```

グループには圏点を付ける。

```
3014   \pxrr@oktrue
3015 }{\pxrr@cond\ifx\pxrr@token\@sptoken\fi%
```

欧文空白には圏点を付けない。

```
3016   \pxrr@okfalse
3017 }{%
3018   \pxrr@check@char\pxrr@token
3019   \ifcase\pxrr@cntr
```

通常文字でないので圏点を付けない。

```
3020   \pxrr@okfalse
3021   \or
```

欧文の通常文字。圏点を付ける。

```
3022     \pxrr@oktrue
3023     \chardef\pxrr@check@char@temp\z@
3024     \or
```

和文の通常文字。圏点を付ける。

```
3025     \pxrr@oktrue
3026     \chardef\pxrr@check@char@temp\@ne
3027     \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。

```
3028     \ifnum\pxrr@k@full=\z@\ifpxrr@ok
3029     \pxrr@check@punct@char{'#1}\pxrr@check@char@temp
3030     \ifpxrr@ok \pxrr@okfalse
3031     \else \pxrr@oktrue
3032     \fi
3033     \fi\fi
3034     \ifpxrr@ok
3035     \pxrr@get@prebreakpenalty\@tempcnta{'#1}%
3036     \mathchardef\pxrr@k@prebreakpenalty\@tempcnta
3037     \pxrr@get@postbreakpenalty\@tempcnta{'#1}%
3038     \mathchardef\pxrr@k@postbreakpenalty\@tempcnta
3039     \fi
3040     }}%
3041 }
```

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。

```
3042 \AtBeginDocument{%
3043   \providecommand*\kenten{\jkenten}%
3044 }
3045 \newcommand*\jkenten{%
3046   \pxrr@k@prologue
3047   \pxrr@kenten
3048 }
3049 \pxrr@add@protect\jkenten
```

`\pxrr@kenten` オプションの処理を行う。

```
3050 \def\pxrr@kenten{%
3051   \@testopt\pxrr@kenten@a{%}
3052 }
3053 \def\pxrr@kenten@a[#1]{%
3054   \def\pxrr@option{#1}%
3055   \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。

```
3056 \expandafter\@firstofone
3057 \else
3058 \expandafter\pxrr@kenten@proc
3059 \fi
3060 }
```

\pxrr@k@bind@param “呼出時変数” へのコピーを行う。

```
3061 \def\pxrr@k@bind@param{%
3062 \let\pxrr@c@ruby@font\pxrr@k@ruby@font
3063 \let\pxrr@c@size@ratio\pxrr@k@size@ratio
3064 \let\pxrr@c@inter@gap\pxrr@k@inter@gap
3065 }
```

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

```
3066 \def\pxrr@kenten@proc#1{%
3067 \pxrr@prepare@fallback{#1}%
3068 \pxrr@k@bind@param
3069 \pxrr@assign@fsize
3070 \pxrr@k@parse@option\pxrr@option
3071 \pxrr@if@alive{%
3072 \pxrr@k@decompose{#1}%
3073 \let\pxrr@body@list\pxrr@res
3074 \pxrr@kenten@main
3075 }%
3076 \pxrr@kenten@exit
3077 }
```

5.11.2 組版処理

\pxrr@kenten@main 圏点の組版処理。

```
3078 \def\pxrr@kenten@main{%
3079 \setbox\pxrr@boxb\pxrr@hbox@to\z@f%
3080 \pxrr@use@ruby@font
3081 \hss\pxrr@k@the@mark\hss
3082 }%
3083 \let\pxrr@entry\pxrr@kenten@entry
3084 \let\pxrr@entry@kspan\pxrr@kenten@entry@kspan
3085 \let\pxrr@entry@ruby\pxrr@kenten@entry@ruby
3086 \let\pxrr@entry@kenten\pxrr@kenten@entry@kenten
3087 \let\pxrr@post\pxrr@k@list@post
3088 \pxrr@k@list@pre
3089 \pxrr@body@list
3090 }
```

5.11.3 前処理

`\pxrr@jprologue` 圏点用の開始処理。

```
3091 \def\pxrr@k@prologue{%
3092   \ifpxrr@k@ghost
3093     \pxrr@jghost@char
3094     \pxrr@inhibitglue
3095   \fi
3096   \begingroup
3097   \ifpxrr@k@ghost
3098     \setbox\pxrr@boxa\hbox{\pxrr@jghost@char}%
3099     \kern-\wd\pxrr@boxa
3100   \fi
3101 }
```

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。

```
3102 \def\pxrr@kenten@exit{%
3103   \ifpxrr@fatal@error
3104     \pxrr@fallback
3105   \fi
3106   \pxrr@k@epilogue
3107 }
```

`\pxrr@jepilogue` 終了処理。

```
3108 \def\pxrr@k@epilogue{%
3109   \ifpxrr@k@ghost
3110     \setbox\pxrr@boxa\hbox{\pxrr@jghost@char}%
3111     \kern-\wd\pxrr@boxa
3112   \fi
3113   \endgroup
3114   \ifpxrr@k@ghost
3115     \pxrr@inhibitglue
3116     \pxrr@jghost@char
3117   \fi
3118 }
```

5.12 デバッグ用出力

```
3119 \def\pxrr@debug@show@kenten@input{%
3120   \typeout{%
3121     pxrr@k@the@mark=\meaning\pxrr@k@the@mark^^J%
3122     pxrr@side=\meaning\pxrr@side^^J%
3123     pxrr@body@list=\meaning\pxrr@body@list^^J%
3124   }%
3125 }
```

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。
3126 \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。
3127 \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 3128 \let\pxrr@ck@ruby@font\relax
3129 \let\pxrr@ck@size@ratio\relax
\pxrr@ck@inter@gap 3130 \let\pxrr@ck@inter@gap\relax
3131 \let\pxrr@ck@inter@gap\relax
\pxrr@ck@ruby@inter@gap 3131 \let\pxrr@ck@ruby@inter@gap\relax
\pxrr@ck@side 3132 \let\pxrr@ck@side\relax
\pxrr@ck@the@mark 3133 \let\pxrr@ck@the@mark\relax
3134 \let\pxrr@ck@ruby@combo\relax
\pxrr@ck@ruby@combo
\ifpxrr@ck@kenten@head 当該のルビ命令が、圏点命令の引数の先頭にあるか。
3135 \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。
3136 \newif\ifpxrr@ck@kenten@end

\pxrr@ck@bind@param “呼出時変数” へのコピーを行う。
3137 \def\pxrr@ck@bind@param{%
3138 \let\pxrr@ck@ruby@font\pxrr@c@ruby@font
3139 \let\pxrr@ck@size@ratio\pxrr@c@size@ratio
3140 \let\pxrr@ck@inter@gap\pxrr@c@inter@gap
3141 \let\pxrr@ck@ruby@inter@gap\pxrr@k@ruby@inter@gap
3142 \let\pxrr@ck@side\pxrr@side
3143 \let\pxrr@ck@the@mark\pxrr@k@the@mark
3144 \let\pxrr@ck@ruby@combo\pxrr@k@ruby@combo
3145 \pxrr@csletcs{ifpxrr@ck@kenten@head}{ifpxrr@k@first@entry}%
3146 \pxrr@csletcs{ifpxrr@ck@kenten@end}{ifpxrr@k@last@entry}%
3147 }
```

6.2 その他の変数

```
\pxrr@ck@zw 圏点の全角幅。
3148 \let\pxrr@ck@zw\relax

\pxrr@ck@raise@P ルビ側が P である場合の、圏点の垂直方向の移動量。
※ 圏点側が S である場合は負値になる。
3149 \let\pxrr@ck@raise@P\relax
```


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

```
3150 \let\pxrr@ck@raise@S\relax
```

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

```
3151 \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。

```
3152 \def\pxrr@ck@check@option{%
3153   \ifpxrr@ck@kenten@head
3154     \let\pxrr@bintr@\@empty
3155     \let\pxrr@bscomp=. \relax
3156     \pxrr@bnobrtrue
3157   \fi
3158   \ifpxrr@ck@kenten@end
3159     \let\pxrr@aintr@\@empty
3160     \let\pxrr@ascomp=. \relax
3161     \pxrr@anobrtrue
3162   \fi
3163 }
```

6.4 フォントサイズ

`\pxrr@ck@assign@fsize` フォントに関連する設定。

```
3164 \def\pxrr@ck@assign@fsize{%
  \pxrr@ck@zw の値を求める。
3165   \begingroup
3166     \@tempdima=\f@size\p@
3167     \@tempdima\pxrr@ck@size@ratio\@tempdima
3168     \edef\pxrr@ruby@fsize{\the\@tempdima}%
3169     \let\pxrr@c@ruby@font\pxrr@ck@ruby@font
3170     \pxrr@use@ruby@font
3171     \pxrr@get@zwidth\pxrr@ck@zw
3172     \global\let\pxrr@gtempa\pxrr@ck@zw
3173   \endgroup
3174   \let\pxrr@ck@zw\pxrr@gtempa

  \pxrr@ck@raise@P、\pxrr@ck@raise@S の値を計算する。
3175   \ifcase\pxrr@ck@side
    圏点側が P の場合。
3176     \@tempdimc\pxrr@ck@zw
3177     \advance\@tempdimc-\pxrr@htratio\@tempdimc
3178     \@tempdima\pxrr@ruby@raise\relax
3179     \@tempdimb\pxrr@ruby@zw\relax
```

```

3180 \advance\@tempdima\pxrr@htratio\@tempdimb
3181 \@tempdimb\pxrr@body@zw\relax
3182 \advance\@tempdima\pxrr@ck@ruby@inter@gap\@tempdimb
3183 \advance\@tempdima\@tempdimc
3184 \edef\pxrr@ck@raise@P{\the\@tempdima}%
3185 \@tempdima\pxrr@body@zw\relax
3186 \@tempdima\pxrr@htratio\@tempdima
3187 \@tempdimb\pxrr@body@zw\relax
3188 \advance\@tempdima\pxrr@ck@inter@gap\@tempdimb
3189 \advance\@tempdima\@tempdimc
3190 \edef\pxrr@ck@raise@S{\the\@tempdima}%
3191 \let\pxrr@ck@raise@t\pxrr@ck@raise@P
3192 \or

```

圏点側が S の場合。

```

3193 \@tempdimc\pxrr@ck@zw
3194 \@tempdimc\pxrr@htratio\@tempdimc
3195 \@tempdima-\pxrr@ruby@lower\relax
3196 \@tempdimb\pxrr@ruby@zw\relax
3197 \advance\@tempdimb-\pxrr@htratio\@tempdimb
3198 \advance\@tempdima-\@tempdimb
3199 \@tempdimb\pxrr@body@zw\relax
3200 \advance\@tempdima-\pxrr@ck@ruby@inter@gap\@tempdimb
3201 \advance\@tempdima-\@tempdimc
3202 \edef\pxrr@ck@raise@S{\the\@tempdima}%
3203 \@tempdima-\pxrr@body@zw\relax
3204 \advance\@tempdima-\pxrr@htratio\@tempdima
3205 \@tempdimb\pxrr@body@zw\relax
3206 \advance\@tempdima-\pxrr@ck@inter@gap\@tempdimb
3207 \advance\@tempdima-\@tempdimc
3208 \edef\pxrr@ck@raise@P{\the\@tempdima}%
3209 \let\pxrr@ck@raise@t\pxrr@ck@raise@S
3210 \fi
3211 }

```

6.5 ブロック毎の組版

`\pxrr@ck@body@natwd` 親文字列の自然長。

```
3212 \let\pxrr@ck@body@natwd\relax
```

`\pxrr@ck@locate` 圏点列のパターン指定。

```
3213 \let\pxrr@ck@locate\relax
```

`\pxrr@ck@kenten@list` 圏点列のリスト。

```
3214 \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。

```
3215 \def\pxrr@ck@compose#1{%
```

圏点を組んだボックスを作る。

```
3216 \setbox\tw@\pxrr@hbox@to\z@{%
3217 \@tempdima=\f@size\p@
3218 \@tempdima\pxrr@ck@size@ratio\@tempdima
3219 \edef\pxrr@ruby@fsize{\the\@tempdima}%
3220 \let\pxrr@c@ruby@font\pxrr@ck@ruby@font
3221 \pxrr@use@ruby@font
3222 \hss\pxrr@ck@the@mark\hss
3223 }%
```

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。

```
3224 \pxrr@save@listproc
3225 \pxrr@decompose{#1}%
3226 \def\pxrr@pre{%
3227 \let\pxrr@res\@empty
3228 \pxrr@ck@compose@entry\pxrr@pre
3229 }%
3230 \def\pxrr@inter{%
3231 \pxrr@ck@compose@entry\pxrr@inter
3232 }%
3233 \def\pxrr@post{%
3234 \pxrr@appto\pxrr@res{\pxrr@post}%
3235 }%
3236 \pxrr@res
3237 \pxrr@restore@listproc
3238 \let\pxrr@natwd\pxrr@ck@body@natwd
```

圏点リストを均等配置する。

```
3239 \pxrr@evenspace@int\pxrr@ck@locate\pxrr@boxb\relax
3240 {\wd\pxrr@boxa}%
```

合成処理。

```
3241 \setbox\z@\hbox{%
3242 \unhcopy\z@
3243 \kern-\wd\z@
3244 \ifcase\pxrr@side
3245 \raise\pxrr@ck@raise@P
3246 \or
3247 \raise\pxrr@ck@raise@S
3248 \or
3249 \raise\pxrr@ck@raise@t
3250 \fi
3251 \hb@xt@\wd\pxrr@boxa{\hss\copy\pxrr@boxb\hss}%
3252 }%
3253 }
3254 \def\pxrr@ck@compose@entry#1#2{%
3255 \setbox\pxrr@boxb\pxrr@hbox{#2}%
```

```

3256 \edef\pxrr@tempa{%
3257   \noexpand\pxrr@appto\noexpand\pxrr@res{\noexpand#1{%
3258     \hb@xt@\the\wd\pxrr@boxb{\hss\copy\tw@\hss}}}%
3259 } \pxrr@tempa
3260 }

```

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。

```
3261 \def\pxrr@dumb@sub#1#2#3{#1}
```

`\pxrr@dumb@ruby` 無難なルビ命令。

```

3262 \def\pxrr@dumb@ruby{%
3263   \pxrr@dumb@sub\pxrr@dumb@ruby@
3264 }
3265 \def\pxrr@dumb@ruby@#1#2#3{#1}

```

`\pxrr@dumb@truby` 無難な両側ルビ命令。

```

3266 \def\pxrr@dumb@truby{%
3267   \pxrr@dumb@sub\pxrr@dumb@truby@
3268 }
3269 \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。

```

3270 \def\pxrr@dumb@kenten{%
3271   \pxrr@dumb@sub\pxrr@dumb@kenten@
3272 }
3273 \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。

```

3274 \providecommand*\pdfstringdefPreHook{}
3275 \g@addto@macro\pdfstringdefPreHook{%

```

`\ruby` と `\kenten` は「本パッケージの命令であるか」の検査が必要。

```

3276   \ifx\pxrr@cmd@ruby\ruby
3277     \let\ruby\pxrr@dumb@ruby
3278   \fi
3279   \let\jruby\pxrr@dumb@ruby
3280   \let\aruby\pxrr@dumb@ruby
3281   \let\truby\pxrr@dumb@truby
3282   \let\atruby\pxrr@dumb@truby
3283   \ifx\pxrr@cmd@kenten\kenten
3284     \let\kenten\pxrr@dumb@kenten
3285   \fi

```

```
3286 \let\kspan\pxrr@dumb@kenten  
3287 }
```