

pxrubrica パッケージ

八登 崇之 (Takayuki YATO; aka “ZR”)

v1.3e [2023/03/01]

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 p \LaTeX 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 pL^AT_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）に圏点を付す指定。
- f 指定時は、親文字列に含まれる“通常文字”の全てに圏点を付加する。F 指定時は、約物である“通常文字”には圏点を付加しない。

3.2 圏点命令の親文字列の入力規則

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

圏点命令の親文字列には以下のものを含めることができる。

- 通常文字： \LaTeX の命令や特殊文字や欧文空白でない、欧文または和文の文字を指す。通常文字には一つの圏点が付加される。
 - F オプションを指定した場合、約物（句読点等）の文字には圏点が付加されない。
 - 欧文文字に圏点を付けた場合、その文字は組版上“和文文字のように”振舞う。
- \LaTeX の命令および欧文空白：これらには圏点が付加されない。
 - 主に \backslash 、や $\backslash\text{quad}$ のような空白用の命令の使用を意図している。
 - $\backslash\text{hspace}\{1zw\}$ 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圏点が付加される。
 - japanese-otf パッケージの $\backslash\text{CID}$ 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。
- $\backslash\text{kspan}\{\text{テキスト}\}$ ：これは、出力されるテキストの幅に応じた個数の圏点が付加される。
 - 例えば、“くの字点”に圏点を付す場合に使える。
 - あるいは、(少々手抜きであるが^{*8}) $\backslash\text{kenten}\{\text{この}\backslash\text{kspan}\{\backslash\text{textgt}\{\text{文字}\}\}\text{だ}\}$ みたいな使い方も考えられる。
- $\backslash\text{kspan}*\{\text{テキスト}\}$ ：これは圏点を付さずにテキストをそのまま出力する。
- ルビ命令 ($\backslash\text{ruby}$ 等)：例えば
 - $\backslash\text{kenten}\{\text{これが}\backslash\text{ruby}[|j|]\{\text{圏点}\}\{\text{けん|てん}\}\text{です}\}$ 。
 のように、ルビ命令はそのまま書くことができる。
 - $\backslash\text{kentenrubycombination}$ の設定によっては、ルビと圏点の両方が付加される。
 - 実装上の制限^{*9}のため、圏点命令の先頭にルビ命令がある場合、ルビの前側の進入が無効になる。同様に、圏点命令の末尾にルビ命令がある場合、ルビの後側の進入が無効になる。

^{*7} 全角空白 ($\backslash\text{hspace}\{1zw\}$) や和欧文間空白 ($\backslash\text{hspace}\{\backslash\text{kanjiskip}\}$) 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、 $\backslash\text{textgt}$ の中で改めて $\backslash\text{kenten}$ を使うべきである。

^{*9} 圏点命令は常にゴースト処理を伴うため、先述の「ゴースト処理と進入は共存しない」という制限に引っかかるのである。

- 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{※}`

- `\kentenmarkinyoko{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*]
- `\kentenmarkintate{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 未実装の機能を呼び出した場合。
18 \def\pxrr@fatal@not@supported#1{%
19   \pxrr@fatal@error{Not yet supported: #1}%
20   \pxrr@eh@fatal
21 }

\pxrr@err@inv@value 引数に無効な値が指定された場合。
22 \def\pxrr@err@inv@value#1{%
23   \pxrr@error{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 (`\rubybigintrusion`/`\rubysmallintrusion`)。実数値マクロ (数字列に展開される)。

`\pxrr@small@intr`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

`\pxrr@yhtratio` 横組和文の高さの縦幅に対する割合 (`\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` 「0pt」という文字列。寸法値マクロへの代入に用いる。
159 `\def\pxrr@zeropt{0pt}`

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

`\pxrr@res` 結果を格納するマクロ。

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

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

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

`\pxrr@cond` `\pxrr@cond\ifXXX...\fi{<真>}{<偽>}` : 一般の T_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

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

の時に、`\CS` を以下の内容で置き換える。

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

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

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

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

の時に、`\pxrr@res` を以下の内容に定義する。

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

```
439 \def\pxrr@zip@single#1#2{%
440   \expandafter\pxrr@zip@single@a\expandafter#1\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` :

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

の時に、`\pxrr@res` を以下の内容に定義する。

```
\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@gttempa\@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` 系でありかつ内部漢字コードが Unicode であるか。エンジン判定は `\kchardef` のプリミティブテストで判定する。

※ テストするプリミティブを `\enablecjktoken` から `\kchardef` に変更 (`\kchardef` は実際に使っている)。

```

477 \pxrr@ifprimitive\kchardef{%
478   \ifnum\ucs"3000="3000
479   \pxrr@csletcs{ifpxrr@in@uptex}{iftrue}%
480   \else
481   \pxrr@csletcs{ifpxrr@in@uptex}{iffalse}%
482   \fi
483 }{%
484 \pxrr@csletcs{ifpxrr@in@uptex}{iffalse}%
485 }

```

`\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。

```

486 \pxrr@ifprimitive\XeTeXrevision{%
487   \pxrr@csletcs{ifpxrr@in@xetex}{iftrue}%
488 }{%
489 \pxrr@csletcs{ifpxrr@in@xetex}{iffalse}%
490 }

```

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

```

491 \@ifpackageloaded{xeCJK}{%
492   \pxrr@csletcs{ifpxrr@in@xecjk}{iftrue}%
493 }{%
494   \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。

```

495 \AtBeginDocument{%
496   \@ifpackageloaded{xeCJK}{%
497     \pxrr@warn@load@order{xeCJK}%
498   }{%
499   }%
500 }

```

`\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。

```

501 \pxrr@ifprimitive\luatexrevision{%
502   \pxrr@csletcs{ifpxrr@in@luatex}{iftrue}%
503 }{%
504   \pxrr@csletcs{ifpxrr@in@luatex}{iffalse}%
505 }

```

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。

```

506 \ifpxrr@in@luatex
507   \directlua{ pxrubtica = {} }
508 \fi

```

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

```

509 \@ifpackageloaded{luatexja-core}{%
510   \pxrr@csletcs{ifpxrr@in@luatexja}{iftrue}%
511 }{%
512   \pxrr@csletcs{ifpxrr@in@luatexja}{iffalse}%
513 \AtBeginDocument{%

```

```

514 \ifpackageloaded{luatexja-core}{%
515 \pxrr@warn@load@order{LuaTeX-ja}%
516 }{}%
517 }%
518 }

519 \ifpxrr@in@xetex
520 \else\ifpxrr@in@luatex
521 \else\ifpxrr@in@ptex
522 \else
523 \pxrr@ifprimitive\pdftexrevision{%
524 \pxrr@warn{%
525 The engine in use seems to be pdfTeX,\MessageBreak
526 so safe mode is turned on%
527 }%
528 \AtEndOfPackage{%
529 \rubysafemode
530 }%
531 }
532 \fi\fi\fi

```

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

```

533 \ifpxrr@in@xetex
534 \pxrr@csletcs{ifpxrr@in@unicode}{iftrue}%
535 \else\ifpxrr@in@luatex
536 \pxrr@csletcs{ifpxrr@in@unicode}{iftrue}%
537 \else\ifpxrr@in@uptex
538 \pxrr@csletcs{ifpxrr@in@unicode}{iftrue}%
539 \else
540 \pxrr@csletcs{ifpxrr@in@unicode}{iffalse}%
541 \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。

```

542 \def\pxrr@jc#1{%
543 \pxrr@jc@a#1\pxrr@nil
544 }
545 \ifpxrr@in@unicode
546 \def\pxrr@jc@a#1:#2\pxrr@nil{%
547 "#2\space
548 }
549 \else\ifpxrr@in@ptex
550 \def\pxrr@jc@a#1:#2\pxrr@nil{%
551 \jis"#1\space\space
552 }
553 \else
554 \def\pxrr@jc@a#1:#2\pxrr@nil{%
555 ‘?\space
556 }

```

```
557 \fi\fi
```

`\pxrr@jchardef` 和文用の `\chardef`。

```
558 \ifpxrr@in@uptex
559 \let\pxrr@jchardef\kchardef
560 \else
561 \let\pxrr@jchardef\chardef
562 \fi
```

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

```
563 \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。)

```
564 \begingroup \catcode'\|=0
565 \gdef\pxrr@if@in@tate{%
566   \pxrr@cond{\if
567     |iftdir|ifmdir F|else T|fi|else F|fi
568   T}\fi
569 }
570 \endgroup
571 \else\ifpxrr@in@luatexja
```

LuaTeX-ja 利用の場合、`direction` パラメータを利用する。

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

```
572 \ifx\ltj@curtfnt\@undefined
573 \let\pxrr@if@in@tate\@secondoftwo
574 \else
575 \def\pxrr@if@in@tate{%
576   \pxrr@cond\ifnum\ltjgetparameter{direction}=\thr@@\fi
577 }
578 \fi
579 \else
```

それ以外は常に横組と見なす。

```
580 \let\pxrr@if@in@tate\@secondoftwo
581 \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。

```
582 \ifpxrr@in@ptex
583 \def\pxrr@get@jchar@token#1#2{%
584   \begingroup
```

```

585     \kansujichar\@ne=#2\relax
586     \xdef\pxrr@x@tempa{\kansuji\@ne}%
587     \endgroup
588     \let#1\pxrr@x@tempa
589 }

```

Unicode 対応 T_EX の場合。 \lowercase トリックを利用する。

```

590 \else\ifpxrr@in@unicode
591   \def\pxrr@get@jchar@token#1#2{%
592     \begingroup
593     \lccode'\?=#2\relax
594     \lowercase{\xdef\pxrr@x@tempa{?}}%
595     \endgroup
596     \let#1\pxrr@x@tempa
597 }

```

それ以外ではダミー定義。

```

598 \else
599   \def\pxrr@get@jchar@token#1#2{%
600     \def#1{?}%
601   }
602 \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で chardef にする。

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

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

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

upT_EX 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。

```

604 \ifpxrr@in@uptex
605   \kchardef\pxrr@jghost@char="113000

```

LuaT_EX-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。

```

606 \else\ifpxrr@in@luatexja
607   \let\pxrr@jghost@char\pxrr@zspace
608   \def\pxrr@jghost@check{%
609     \begingroup
610 %     \ltjsetparameter{jaxspmode={\pxrr@zspace,3}}%
611 %     \ltjsetparameter{xkanjiskip=\p}%
612 %     \ltjsetparameter{autoxspacing=false}%
613     \setbox\z@\hbox{\char"3001\char"3000}%
614 %     \ltjsetparameter{autoxspacing=true}%

```



```

615     \setbox\tw@\hbox{\char"3001\inhibitglue\char"3000}%
616     \ifdim\wd\tw@=\wd\z@
617         \global\chardef\pxrr@jghost@char@="00A0
618         \gdef\pxrr@jghost@char{\ltjjachar\pxrr@jghost@char}%
619     \fi
620 \endgroup
621 }
622 \AtBeginDocument{%
623     \pxrr@jghost@check
624 }

```

それ以外の場合は（仕方が無いので）全角空白を用いる。

```

625 \else
626     \let\pxrr@jghost@char\pxrr@zspace
627 \fi\fi

```

`\pxrr@x@K` 適当な漢字（実際は <一>）のトークン。

```

628 \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。

```

629 \ifpxrr@in@ptex
630     \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。

```

631     \pxrr@x@swafalse
632     \begingroup
633         \inhibitxspcode\pxrr@x@K\thr@@
634         \kanjiskip\p@
635         \setbox\z@\hbox{\noautospacing\pxrr@x@K\pxrr@x@K}%
636         \setbox\tw@\hbox{\pxrr@x@K\pxrr@x@K}%
637         \ifdim\wd\tw@>\wd\z@
638             \aftergroup\pxrr@x@swatruue
639         \fi
640     \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。

```

641     \edef#1{%
642         \ifpxrr@x@swa \the\kanjiskip
643         \else \pxrr@zeropt
644         \fi
645     }%
646 }

```

LuaTeX-ja 使用の場合。

```

647 \else\ifpxrr@in@luatexja
648     \def\pxrr@get@iiskip#1{%
649         \ifnum\ltjgetparameter{autospacing}=\@ne
650             \xdef\pxrr@x@gtempa{\ltjgetparameter{kanjiskip}}%
651             \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。)

```
652     \pxrr@get@interchar@glue{\pxrr@x@K\pxrr@x@K}%
653     \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。

```
654     \pxrr@warn@unknown@iiskip
655     \global\let\pxrr@x@gtempa\pxrr@zeropt
656     \fi
657     \fi
658     \let#1\pxrr@x@gtempa
659     \else
660     \let#1\pxrr@zeropt
661     \fi
662 }
```

和文間空白の推定に失敗した場合の警告。

```
663 \def\pxrr@warn@unknown@iiskip{%
664   \global\let\pxrr@warn@unknown@iiskip\relax
665   \pxrr@warn{Cannot find the kanjiskip value}%
666 }
```

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。

```
667 \def\pxrr@get@interchar@glue#1{%
668   \begingroup
669   \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。

```
670   \skip\z@\maxdimen\relax
671   \directlua{%
672     pcall(pxrrubrica._get_interchar_glue)
673   }%
674   \xdef\pxrr@x@gtempa{\the\skip\z@}%
675 \endgroup
676 }
677 \begingroup
678 \endlinechar=10 \directlua{%
679   local node, tex = node, tex
680   local id_glyph, id_glue = node.id("glyph"), node.id("glue")
681   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。

```
682   function pxrrubrica._get_interchar_glue()
683     local c, n = false, tex.box[0].head
```

```

684         while n do
    ※ 2014 年頃の LuaTeX-ja では文字の部分が hlist ノードになっている。
685             if n.id == id_glyph or n.id == id_hlist then
686                 c = true
687             elseif c and n.id == id_glue then

```

ここでの `n` が “文字間グルー” のノードである。

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

```

688             if n.width then
689                 tex.setglue(0, n.width, n.stretch, n.shrink,
690                     n.stretch_order, n.shrink_order)
691             elseif n.spec then
692                 tex.setskip(0, node.copy(n.spec))
693             end
694             break
695         end
696         n = n.next
697     end
698 end
699 }%
700 \endgroup%

```

それ以外の場合はゼロとする。

```

701 \else
702     \def\pxrr@get@iiskip#1{%
703         \let#1\pxrr@zeropt
704     }
705 \fi\fi

```

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

```

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

```

```

722   }%
723 }

```

LuaTeX-ja 使用の場合。処理の流れは和文間空白の場合と同じ。

```

724 \else\ifpxrr@in@luatexja
725   \def\pxrr@get@iaiskip#1{%
726     \ifnum\ltjgetparameter{autoxspacing}=\@ne
727       \xdef\pxrr@x@tempa{\ltjgetparameter{xkanjiskip}}%
728       \ifdim\glueexpr\pxrr@x@tempa=\maxdimen

```

判定用のボックスは欧文・和文の組とする。

```

729         \pxrr@get@interchar@glue{A\pxrr@x@K}%
730         \ifdim\glueexpr\pxrr@x@tempa=\maxdimen
731           \pxrr@warn@unknown@iaiskip
732           \global\let\pxrr@x@tempa\pxrr@zeropt
733         \fi
734       \fi
735       \let#1\pxrr@x@tempa
736     \else
737       \let#1\pxrr@zeropt
738     \fi
739 }

```

和欧文間空白の推定に失敗した場合の警告。

```

740 \def\pxrr@warn@unknown@iaiskip{%
741   \global\let\pxrr@warn@unknown@iaiskip\relax
742   \pxrr@warn{Cannot find the xkanjiskip value}%
743 }

```

それ以外の場合は実際の組版結果から判断する。

```

744 \else
745   \def\pxrr@get@iaiskip#1{%
746     \begingroup
747       \setbox\z@\hbox{M\pxrr@x@K}%
748       \setbox\tw@\hbox{M\vrule\@width\z@\relax\pxrr@x@K}%
749       \@tempdima\wd\z@ \advance\@tempdima-\wd\tw@
750       \@tempdimb\@tempdima \divide\@tempdimb\thr@@
751       \xdef\pxrr@x@tempa{\the\@tempdima\space minus \the\@tempdimb}%
752     \endgroup
753     \let#1=\pxrr@x@tempa
754   }%
755 \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。

```

756 \ifpxrr@in@ptex
757   \def\pxrr@get@zwidth#1{%
758     \@tempdima=1zw\relax
759     \edef#1{\the\@tempdima}%
760 }

```

`\zw` が定義されている場合は `1\zw` とする。

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

`\jsZw` が定義されている場合は `1\jsZw` とする。

```
766 \else\if\ifx\jsZw\undefined T\else F\fi F% if defined
767   \def\pxrr@get@zwidth#1{%
768     \@tempdima=1\jsZw\relax
769     \edef#1{\the\@tempdima}%
770   }
```

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。

```
771 \else\ifnum\pxrr@x@K>\@cclv
772   \def\pxrr@get@zwidth#1{%
773     \setbox\tw@\hbox{\pxrr@x@K}%
774     \@tempdima\wd\tw@
775     \ifdim\@tempdima>z@\else \@tempdima\@size\p@ \fi
776     \edef#1{\the\@tempdima}%
777   }
```

それ以外の場合は要求サイズと等しいとする。

```
778 \else
779   \def\pxrr@get@zwidth#1{%
780     \@tempdima\@size\p@\relax
781     \edef#1{\the\@tempdima}%
782   }
783 \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。

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

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

```
788 \else\ifpxrr@in@luatexja
789   \def\pxrr@get@prebreakpenalty#1#2{%
790     #1=\ltjgetparameter{prebreakpenalty}{#2}\relax
791   }
```

それ以外の場合はゼロとして扱う。

```
792 \else
793   \def\pxrr@get@prebreakpenalty#1#2{%
794     #1=\z@
795   }
```

```
796 \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。

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

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

```
801 \else\ifpxrr@in@luatexja
802   \def\pxrr@get@postbreakpenalty#1#2{%
803     #1=\ltjgetparameter{postbreakpenalty}{#2}\relax
804   }
```

それ以外の場合はゼロとして扱う。

```
805 \else
806   \def\pxrr@get@postbreakpenalty#1#2{%
807     #1=\z@
808   }
809 \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。

```
810 \ifpxrr@in@ptex
811   \def\pxrr@check@punct@char#1#2{%
812     \pxrr@okfalse
813     \ifcase#2\relax
814       \ifnum\xspcode#1=\thr@@\else
815         \pxrr@oktrue
816       \fi
817     \else
818       \ifnum\inhibitxspcode#1=\thr@@\else
819         \pxrr@oktrue
820       \fi
821     \fi
822   }
```

LuaTeX-japan 使用時も基本的に pTeX と同じロジックを使う。ただし LuaTeX-japan 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。

```
823 \else\ifpxrr@in@luatexja
```

```

824 \def\pxrr@check@punct@char#1#2{%
825   \ifnum\ltjgetparameter{jaxspmode}{#1}=\thr@@
826     \pxrr@okfalse
827   \else
828     \pxrr@oktrue
829   \fi
830 }

```

それ以外の場合は常に偽として扱う。

```

831 \else
832 \def\pxrr@check@punct@char#1#2{%
833   \pxrr@okfalse
834 }
835 \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。

```

836 \ifpxrr@in@ptex
837 \def\pxrr@force@nonpunct@achar#1{%
838   \global\xspcode#1=\thr@@
839 }

```

LuaTeX-ja 使用の場合。

```

840 \else\ifpxrr@in@luatexja
841 \def\pxrr@force@nonpunct@achar#1{%
842   \ltjglobalsetparameter{jaxspmode={#1,3}}%
843 }

```

それ以外の場合は何もしない。

```

844 \else
845 \def\pxrr@force@nonpunct@achar#1{}
846 \fi\fi

```

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

```

847 \ifx\inhibitglue\undefined
848 \let\pxrr@inhibitglue\relax
849 \else
850 \let\pxrr@inhibitglue\inhibitglue
851 \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。

```
852 \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。

```
853 \newcommand*\rubyssetup[1]{%
```

```

854 \pxrr@in@setuptrue
855 \pxrr@fatal@errorfalse
856 \pxrr@parse@option{#1}%
857 \ifpxrr@fatal@error\else
858 \pxrr@csletcs{ifpxrr@d@bprotr}{ifpxrr@bprotr}%
859 \pxrr@csletcs{ifpxrr@d@aprotr}{ifpxrr@aprotr}%
860 \let\pxrr@d@bintr\pxrr@bintr@
861 \let\pxrr@d@aintr\pxrr@aintr@
862 \let\pxrr@d@athead\pxrr@athead
863 \let\pxrr@d@mode\pxrr@mode
864 \let\pxrr@d@side\pxrr@side
865 \let\pxrr@d@evensp\pxrr@evensp
866 \let\pxrr@d@fullsize\pxrr@fullsize
867 \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。

```

868 \pxrr@in@setupfalse
869 }

```

\rubyfontsetup 対応するパラメタを設定する。

```

870 \newcommand*\rubyfontsetup{}
871 \def\rubyfontsetup#{%
872 \def\pxrr@ruby@font
873 }

```

\rubybigintrusion 対応するパラメタを設定する。

```

\rubysmallintrusion 874 \newcommand*\rubybigintrusion[1]{%
\rubymaxmargin 875 \edef\pxrr@big@intr{#1}%
876 }
\rubyintergap 877 \newcommand*\rubysmallintrusion[1]{%
\rubysizeratio 878 \edef\pxrr@small@intr{#1}%
879 }
880 \newcommand*\rubymaxmargin[1]{%
881 \edef\pxrr@maxmargin{#1}%
882 }
883 \newcommand*\rubyintergap[1]{%
884 \edef\pxrr@inter@gap{#1}%
885 }
886 \newcommand*\rubysizeratio[1]{%
887 \edef\pxrr@size@ratio{#1}%
888 }

```

\rubyusejghost 対応するスイッチを設定する。

```

\rubynousejghost 889 \newcommand*\rubyusejghost{%
890 \pxrr@jghosttrue
891 }
892 \newcommand*\rubynousejghost{%
893 \pxrr@jghostfalse

```


894 }

`\rubyuseaghost` 対応するスイッチを設定する。

```
\rubynouseaghost 895 \newcommand*\rubyuseaghost{%  
896   \pxrr@aghosttrue  
897   \pxrr@setup@aghost  
898 }  
899 \newcommand*\rubynouseaghost{%  
900   \pxrr@aghostfalse  
901 }
```

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

```
\rubynoadjustatlineedge 902 \newcommand*\rubyadjustatlineedge{%  
903   \pxrr@edge@adjusttrue  
904 }  
905 \newcommand*\rubynoadjustatlineedge{%  
906   \pxrr@edge@adjustfalse  
907 }
```

`\rubybreakjukugo` 対応するスイッチを設定する。

```
\rubynobreakjukugo 908 \newcommand*\rubybreakjukugo{%  
909   \pxrr@break@jukugotrue  
910 }  
911 \newcommand*\rubynobreakjukugo{%  
912   \pxrr@break@jukugofalse  
913 }
```

`\rubysafemode` 対応するスイッチを設定する。

```
\rubynosafemode 914 \newcommand*\rubysafemode{%  
915   \pxrr@safe@modetrue  
916 }  
917 \newcommand*\rubynosafemode{%  
918   \pxrr@safe@modefalse  
919 }
```

`\rubystretchprop` 対応するパラメタを設定する。

```
\rubystretchprophead 920 \newcommand*\rubystretchprop[3]{%  
921   \edef\pxrr@sprop@x{#1}%  
922   \edef\pxrr@sprop@y{#2}%  
923   \edef\pxrr@sprop@z{#3}%  
924 }  
925 \newcommand*\rubystretchprophead[2]{%  
926   \edef\pxrr@sprop@hy{#1}%  
927   \edef\pxrr@sprop@hz{#2}%  
928 }  
929 \newcommand*\rubystretchpropend[2]{%  
930   \edef\pxrr@sprop@ex{#1}%  
931   \edef\pxrr@sprop@ey{#2}%  
932 }
```

`\rubyuseextra` 残念ながら今のところは使用不可。

```
933 \newcommand*\rubyuseextra[1]{%
934   \pxrr@cmta=#1\relax
935   \ifnum\pxrr@cmta=\z@
936     \chardef\pxrr@extra\pxrr@cmta
937   \else
938     \pxrr@err@inv@value{\the\pxrr@cmta}%
939   \fi
940 }
```

4.8 ルビオブション解析

`\pxrr@bintr@` オプション解析中でのみ使われ、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。)

```
941 \let\pxrr@bintr@\@empty
942 \let\pxrr@aintr@\@empty
```

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

```
943 \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。

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

入力が「||」の場合は、「|-|」に置き換える。

```
945   \edef\pxrr@tempa{#1}%
946   \ifx\pxrr@tempa\pxrr@doublebar
947     \def\pxrr@tempa{|-|}%
948   \fi
```

各パラメタの値を全般設定のもので初期化する。

```
949   \pxrr@csletcs{ifpxrr@bprotr}{ifpxrr@d@bprotr}%
950   \pxrr@csletcs{ifpxrr@aprotr}{ifpxrr@d@aprotr}%
951   \let\pxrr@bintr@\pxrr@d@bintr
952   \let\pxrr@aintr@\pxrr@d@aintr
953   \let\pxrr@athead\pxrr@d@athead
954   \let\pxrr@mode\pxrr@d@mode
955   \let\pxrr@side\pxrr@d@side
956   \let\pxrr@evensp\pxrr@d@evensp
957   \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。

```
958   \let\pxrr@bscomp\relax
959   \let\pxrr@ascomp\relax
960   \pxrr@bnobrfalse
961   \pxrr@anobrfalse
962   \pxrr@bfintrfalse
963   \pxrr@afintrfalse
```

明示フラグを偽にする。

```
964 \pxrr@mode@givenfalse
965 \pxrr@thead@givenfalse
```

両側ルビの場合、基本モード既定値が M に固定される。

```
966 \ifpxrr@truby
967   \let\pxrr@mode=M%
968 \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。

```
969 \def\pxrr@po@FS{bi}%
970 \expandafter\pxrr@parse@option@loop\pxrr@tempa @\pxrr@end
971 }
```

有限状態機械のループ。

```
972 \def\pxrr@parse@option@loop#1{%
973 \ifpxrrDebug
974 \typeout{\pxrr@po@FS/#1[\@nameuse{pxrr@po@C@#1}]}%
975 \fi
976 \csname pxrr@po@PR@#1\endcsname
977 \expandafter\ifx\csname pxrr@po@C@#1\endcsname\relax
978   \let\pxrr@po@FS\relax
979 \else
980   \pxrr@letcs\pxrr@po@FS
981   {\pxrr@po@TR@\pxrr@po@FS \@nameuse{pxrr@po@C@#1}]}%
982 \fi
983 \ifpxrrDebug
984 \typeout{->\pxrr@po@FS}%
985 \fi
986 \pxrr@ifx{\pxrr@po@FS\relax}{-%
987   \pxrr@fatal@unx@letter{#1}%
988   \pxrr@parse@option@exit
989 }{-%
990   \pxrr@parse@option@loop
991 }%
992 }
```

後処理。

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

既定値設定 (\rubyssetup) である場合もしない。

```
994 \ifpxrr@in@setup\else
```

両側ルビ命令の場合は、 \pxrr@side の値を変更する。

```
995   \ifpxrr@truby
996     \chardef\pxrr@side\tw@
997   \fi
```

整合性検査を行う。

```
998   \pxrr@check@option
```

\pxrr@?intr の値を設定する。

```
999   \@tempdima=\pxrr@ruby@zw\relax
1000  \@tempdimb=\pxrr@or@zero\pxrr@bintr@\@tempdima
1001  \edef\pxrr@bintr{\the\@tempdimb}%
1002  \@tempdimb=\pxrr@or@zero\pxrr@aintr@\@tempdima
1003  \edef\pxrr@aintr{\the\@tempdimb}%
1004  \fi
1005 }
```

\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。

```
1006 \def\pxrr@or@zero#1{%
1007   \ifx#1\@empty \pxrr@zero
1008   \else #1%
1009   \fi
1010 }
```

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。

```
1011 \def\pxrr@po@C@{F}
1012 \@namedef{pxrr@po@C@|}{V}
1013 \@namedef{pxrr@po@C@:}{S}
1014 \@namedef{pxrr@po@C@.}{S}
1015 \@namedef{pxrr@po@C@*}{S}
1016 \@namedef{pxrr@po@C@!}{S}
1017 \@namedef{pxrr@po@C@<}{B}
1018 \@namedef{pxrr@po@C@()}{B}
1019 \@namedef{pxrr@po@C@>}{A}
1020 \@namedef{pxrr@po@C@)}{A}
1021 \@namedef{pxrr@po@C@-}{M}
1022 \def\pxrr@po@C@c{M}
1023 \def\pxrr@po@C@h{M}
1024 \def\pxrr@po@C@H{M}
1025 \def\pxrr@po@C@m{M}
1026 \def\pxrr@po@C@g{M}
1027 \def\pxrr@po@C@j{M}
1028 \def\pxrr@po@C@M{M}
1029 \def\pxrr@po@C@J{M}
1030 \def\pxrr@po@C@P{M}
1031 \def\pxrr@po@C@S{M}
1032 \def\pxrr@po@C@e{M}
1033 \def\pxrr@po@C@E{M}
1034 \def\pxrr@po@C@f{M}
1035 \def\pxrr@po@C@F{M}
```

機能プロセス。

```
1036 \def\pxrr@po@PR@{,%
1037   \pxrr@parse@option@exit
1038 }
1039 \@namedef{pxrr@po@PR@|}{,%
```

```

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

```

```

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

```

```

1138 \let\pxrr@mode=j%
1139 \pxrr@mode@giventrue
1140 }
1141 \def\pxrr@po@PR@M{%
1142 \let\pxrr@mode=M%
1143 \pxrr@mode@giventrue
1144 }
1145 \def\pxrr@po@PR@J{%
1146 \let\pxrr@mode=J%
1147 \pxrr@mode@giventrue
1148 }
1149 \def\pxrr@po@PR@P{%
1150 \chardef\pxrr@side\z@
1151 }
1152 \def\pxrr@po@PR@S{%
1153 \chardef\pxrr@side\@ne
1154 }
1155 \def\pxrr@po@PR@E{%
1156 \chardef\pxrr@evensp\z@
1157 }
1158 \def\pxrr@po@PR@e{%
1159 \chardef\pxrr@evensp\@ne
1160 }
1161 \def\pxrr@po@PR@F{%
1162 \chardef\pxrr@fullsize\z@
1163 }
1164 \def\pxrr@po@PR@f{%
1165 \chardef\pxrr@fullsize\@ne
1166 }

```

遷移表。

```

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

```

```

1185 \def\pxrr@po@TR@mi@S{as}
1186 \def\pxrr@po@TR@as@S{as}
1187 \def\pxrr@po@TR@bi@B{bs}
1188 \def\pxrr@po@TR@bi@M{mi}
1189 \def\pxrr@po@TR@bb@M{mi}
1190 \def\pxrr@po@TR@bs@M{mi}
1191 \def\pxrr@po@TR@mi@M{mi}
1192 \def\pxrr@po@TR@bi@A{fi}
1193 \def\pxrr@po@TR@bb@A{fi}
1194 \def\pxrr@po@TR@bs@A{fi}
1195 \def\pxrr@po@TR@mi@A{fi}
1196 \def\pxrr@po@TR@as@A{fi}
1197 \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。

```

1198 \def\pxrr@mode@grand{%
1199   \if      m\pxrr@mode m%
1200   \else\if M\pxrr@mode m%
1201   \else\if j\pxrr@mode j%
1202   \else\if J\pxrr@mode j%
1203   \else\if g\pxrr@mode g%
1204   \else ?%
1205   \fi\fi\fi\fi\fi
1206 }

```

`\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。

```

1207 \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。
1208   \ifpxrr@bprotr\else
1209     \ifpxrr@aprotr\else
1210       \pxrr@fatal@bad@no@protr
1211     \fi
1212   \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。

```

1213   \pxrr@oktrue
1214   \ifx\pxrr@bintr@\@empty\else
1215     \pxrr@okfalse
1216   \fi
1217   \ifx\pxrr@aintr@\@empty\else
1218     \pxrr@okfalse
1219   \fi

```



```

1220 \ifpxrr@ghost\else
1221   \pxrr@oktrue
1222 \fi
1223 \ifpxrr@ok\else
1224   \pxrr@fatal@bad@intr
1225 \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。

```

1226 \if g\pxrr@mode\else
1227   \ifpxrr@abody
1228     \let\pxrr@mode=g\relax
1229     \ifpxrr@mode@given
1230       \pxrr@warn@must@group
1231     \fi
1232   \fi
1233 \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。

```

1234 \if \pxrr@mode@grand j%
1235   \ifnum\pxrr@side=\tw@
1236     \let\pxrr@mode=g\relax
1237     \ifpxrr@mode@given
1238       \pxrr@warn@bad@jukugo
1239     \fi
1240   \fi
1241 \fi

```

肩付き指定 (h) に関する検査。

```

1242 \ifnum\pxrr@athead>\z@

```

横組みでは不可なので中付きに変更する。

```

1243 \pxrr@if@in@tate{ }{\%else
1244   \chardef\pxrr@athead\z@
1245 }%

```

グループルビでは不可なので中付きに変更する。

```

1246 \if g\pxrr@mode
1247   \chardef\pxrr@athead\z@
1248 \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。

```

1249 \ifnum\pxrr@athead=\z@
1250   \ifpxrr@athead@given
1251     \pxrr@warn@bad@athead
1252   \fi
1253 \fi
1254 \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。

```
1255 \ifpxrr@abody
1256   \chardef\pxrr@evensp\z@
1257 \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。）

```
1258 \if g\pxrr@mode\else
1259   \chardef\pxrr@evensp\@ne
1260 \fi
```

圏点ルビ同時付加の場合の調整。

```
1261 \ifpxrr@combo
1262   \pxrr@ck@check@option
1263 \fi
1264 }
```

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。

```
1265 \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。

```
1266 \let\pxrr@body@zw\pxrr@zeropt
1267 \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。

```
1268 \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）。

```
1269 \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。

```
1270 \def\pxrr@htratio{0}
```

`\pxrr@iiskip` 和文間空白および和欧文間空白の量。

```
\pxrr@iaiskip 1271 \let\pxrr@iiskip\pxrr@zeropt
1272 \let\pxrr@iaiskip\pxrr@zeropt
```

`\pxrr@assign@fsize` 上記の変数（マクロ）を設定する。

```
1273 \def\pxrr@assign@fsize{%
1274   \@tempdima=\f@size\p@
1275   \@tempdima\pxrr@c@size@ratio\@tempdima
```

```

1276 \edef\pxrr@ruby@fsize{\the\@tempdima}%
1277 \pxrr@get@zwidth\pxrr@body@zw
1278 \begingroup
1279   \pxrr@use@ruby@font
1280   \pxrr@get@zwidth\pxrr@ruby@zw
1281   \global\let\pxrr@tempa\pxrr@ruby@zw
1282 \endgroup
1283 \let\pxrr@ruby@zw\pxrr@tempa
1284 \pxrr@get@iiskip\pxrr@iiskip
1285 \pxrr@get@iaiskip\pxrr@iaiskip

\pxrr@htratio の値を設定する。
1286 \pxrr@if@in@tate{%
1287   \let\pxrr@htratio\pxrr@thtratio
1288 }{%
1289   \let\pxrr@htratio\pxrr@yhtratio
1290 }%

\pxrr@ruby@raise の値を計算する。
1291 \@tempdima\pxrr@body@zw\relax
1292 \@tempdima\pxrr@htratio\@tempdima
1293 \@tempdimb\pxrr@ruby@zw\relax
1294 \advance\@tempdimb-\pxrr@htratio\@tempdimb
1295 \advance\@tempdima\@tempdimb
1296 \@tempdimb\pxrr@body@zw\relax
1297 \advance\@tempdima\pxrr@c@inter@gap\@tempdimb
1298 \edef\pxrr@ruby@raise{\the\@tempdima}%

\pxrr@ruby@lower の値を計算する。
1299 \@tempdima\pxrr@body@zw\relax
1300 \advance\@tempdima-\pxrr@htratio\@tempdima
1301 \@tempdimb\pxrr@ruby@zw\relax
1302 \@tempdimb\pxrr@htratio\@tempdimb
1303 \advance\@tempdima\@tempdimb
1304 \@tempdimb\pxrr@body@zw\relax
1305 \advance\@tempdima\pxrr@c@inter@gap\@tempdimb
1306 \edef\pxrr@ruby@lower{\the\@tempdima}%

圏点ルビ同時付加の設定。
1307 \ifpxrr@combo
1308   \pxrr@ck@assign@fsize
1309   \fi
1310 }

```

`\pxrr@use@ruby@font` ルビ用のフォントに切り替える。

```

1311 \def\pxrr@use@ruby@font{%
1312   \pxrr@without@macro@trace{%
1313     \let\rubyfontsize\pxrr@ruby@fsize
1314     \fontsize{\pxrr@ruby@fsize}{\z@}\selectfont
1315     \pxrr@c@ruby@font

```

```

1316 }%
1317 }

```

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 1318 \chardef\pxrr@locate@inner=1
\pxrr@locate@end 1319 \chardef\pxrr@locate@head=0
1320 \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。

```

1321 \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。

```

1322 \setbox#2\pxrr@hbox{#5}\@tempdima\wd#2%
1323 \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。

```

1324 \pxrr@decompose{#5}%
1325 \pxrr@evenspace@int{#1}{#2}{#3}{#4}%
1326 }

```

ここから実行を開始することもある。

```

1327 \def\pxrr@evenspace@int#1#2#3#4{%

```

比率パラメタの設定。

```

1328 \pxrr@save@listproc
1329 \ifcase#1%
1330 \pxrr@evenspace@param\pxrr@zero\pxrr@sprop@hy\pxrr@sprop@hz
1331 \or
1332 \pxrr@evenspace@param\pxrr@sprop@x\pxrr@sprop@y\pxrr@sprop@z
1333 \or
1334 \pxrr@evenspace@param\pxrr@sprop@ex\pxrr@sprop@ey\pxrr@zero
1335 \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。

```

1336 \pxrr@dima=\pxrr@cntr\p@
1337 \advance\pxrr@dima-\p@
1338 \pxrr@dima=\pxrr@sprop@y@\pxrr@dima
1339 \advance\pxrr@dima\pxrr@sprop@x@\p@
1340 \advance\pxrr@dima\pxrr@sprop@z@\p@
1341 \ifdim\pxrr@dima>\z@\else
1342   \ifnum#1>\z@
1343     \let\pxrr@sprop@x@\@ne
1344     \advance\pxrr@dima\p@
1345   \fi
1346   \ifnum#1<\tw@
1347     \let\pxrr@sprop@z@\@ne
1348     \advance\pxrr@dima\p@
1349   \fi
1350 \fi
1351 \edef\pxrr@tempa{\strip@pt\pxrr@dima}%
1352 \ifpxrr@Debug
1353 \typeout{\number\pxrr@sprop@x@:\number\pxrr@sprop@z@:\pxrr@tempa}%
1354 \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。

```

1355 \def\pxrr@pre##1{\pxrr@hfilx\pxrr@sprop@x@ ##1}%
1356 \def\pxrr@inter##1{\pxrr@hfilx\pxrr@sprop@y@ ##1}%
1357 \def\pxrr@post{\pxrr@hfilx\pxrr@sprop@z@}%
1358 \def\pxrr@makebox@res{%
1359   \setbox#2=\pxrr@hbox@to#4{#3\pxrr@res}%
1360 }%
1361 \pxrr@makebox@res

```

前後の空白の量を求める。

```

1362 \pxrr@dima\wd#2%
1363 \advance\pxrr@dima-\pxrr@natwd\relax
1364 \pxrr@invscale\pxrr@dima\pxrr@tempa
1365 \@tempdima\pxrr@sprop@x@\pxrr@dima
1366 \edef\pxrr@bspace{\the\@tempdima}%
1367 \@tempdima\pxrr@sprop@z@\pxrr@dima
1368 \edef\pxrr@aspace{\the\@tempdima}%
1369 \pxrr@restore@listproc
1370 \ifpxrr@Debug
1371 \typeout{\pxrr@bspace:\pxrr@aspace}%
1372 \fi
1373 }
1374 \def\pxrr@evenspace@param#1#2#3{%
1375   \let\pxrr@sprop@x@#1%
1376   \let\pxrr@sprop@y@#2%

```

```

1377 \let\pxrr@sprop@z@#3%
1378 }
1379 \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。

```

1380 \def\pxrr@adjust@margin{%
1381 \pxrr@save@listproc
1382 \@tempdima\pxrr@body@zw\relax
1383 \@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。

```

1384 \@tempswafalse
1385 \def\pxrr@pre##1{\pxrr@hfilx\pxrr@sprop@x@ ##1}%
1386 \def\pxrr@inter##1{\pxrr@hfilx\pxrr@sprop@y@ ##1}%
1387 \def\pxrr@post{\pxrr@hfilx\pxrr@sprop@z@}%
1388 \ifnum\pxrr@cntr>\@ne
1389 \ifdim\pxrr@bspace>\@tempdima
1390 \edef\pxrr@bspace{\the\@tempdima}%
1391 \def\pxrr@pre##1{\hskip\pxrr@bspace\relax ##1}%
1392 \@tempswatrue
1393 \fi
1394 \ifdim\pxrr@aspace>\@tempdima
1395 \edef\pxrr@aspace{\the\@tempdima}%
1396 \def\pxrr@post{\hskip\pxrr@aspace\relax}%
1397 \@tempswatrue
1398 \fi
1399 \fi

```

必要に応じて再調整を行う。

```

1400 \if@tempswa
1401 \pxrr@makebox@res
1402 \fi
1403 \pxrr@restore@listproc
1404 \ifpxrr@Debug
1405 \typeout{\pxrr@bspace:\pxrr@aspace}%
1406 \fi
1407 }

```

`\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。

```

1408 \def\pxrr@save@listproc{%
1409 \let\pxrr@pre@save\pxrr@pre
1410 \let\pxrr@inter@save\pxrr@inter
1411 \let\pxrr@post@save\pxrr@post
1412 }
1413 \let\pxrr@pre@save\@undefined

```

```

1414 \let\pxrr@inter@save\@undefined
1415 \let\pxrr@post@save\@undefined

```

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

```

1416 \def\pxrr@restore@listproc{%
1417   \let\pxrr@pre\pxrr@pre@save
1418   \let\pxrr@inter\pxrr@inter@save
1419   \let\pxrr@post\pxrr@post@save
1420 }

```

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。

```

1421 \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。

```

1422 \def\pxrr@transform@kana#1{%
1423   \let\pxrr@trans@res\@empty
1424   \def\pxrr@transform@kana@end\pxrr@end{%
1425     \let#1\pxrr@trans@res
1426   }%
1427   \expandafter\pxrr@transform@kana@loop@a#1\pxrr@end
1428 }
1429 \def\pxrr@transform@kana@loop@a{%
1430   \futurelet\pxrr@token\pxrr@transform@kana@loop@b
1431 }
1432 \def\pxrr@transform@kana@loop@b{%
1433   \ifx\pxrr@token\pxrr@end
1434     \let\pxrr@tempb\pxrr@transform@kana@end
1435   \else\ifx\pxrr@token\bgroup
1436     \let\pxrr@tempb\pxrr@transform@kana@loop@c
1437   \else\ifx\pxrr@token\@sptoken
1438     \let\pxrr@tempb\pxrr@transform@kana@loop@d
1439   \else
1440     \let\pxrr@tempb\pxrr@transform@kana@loop@e
1441   \fi\fi\fi
1442   \pxrr@tempb
1443 }
1444 \def\pxrr@transform@kana@loop@c#1{%
1445   \pxrr@appto\pxrr@trans@res{#{#1}}%
1446   \pxrr@transform@kana@loop@a
1447 }
1448 \expandafter\def\expandafter\pxrr@transform@kana@loop@d\space{%
1449   \pxrr@appto\pxrr@trans@res{ }%
1450   \pxrr@transform@kana@loop@a
1451 }
1452 \def\pxrr@transform@kana@loop@e#1{%

```

```

1453 \expandafter\pxrr@transform@kana@loop@f\string#1\pxrr@nil#1%
1454 }
1455 \def\pxrr@transform@kana@loop@f#1#2\pxrr@nil#3{%
1456 \@tempswafalse
1457 \ifnum'#1>\@cclv
1458 \begingroup\expandafter\expandafter\expandafter\endgroup
1459 \expandafter\ifx\csname pxrr@nonsmall/#3\endcsname\relax\else
1460 \@tempswatruue
1461 \fi
1462 \fi
1463 \if@tempswa
1464 \edef\pxrr@tempa{%
1465 \noexpand\pxrr@appto\noexpand\pxrr@trans@res
1466 {\csname pxrr@nonsmall/#3\endcsname}%
1467 }%
1468 \pxrr@tempa
1469 \else
1470 \pxrr@appto\pxrr@trans@res{#3}%
1471 \fi
1472 \pxrr@transform@kana@loop@a
1473 }
1474 \def\pxrr@assign@nonsmall#1/#2\pxrr@nil{%
1475 \pxrr@get@jchar@token\pxrr@tempa{\pxrr@jc{#1}}%
1476 \pxrr@get@jchar@token\pxrr@tempb{\pxrr@jc{#2}}%
1477 \expandafter\edef\csname pxrr@nonsmall/\pxrr@tempa\endcsname
1478 {\pxrr@tempb}%
1479 }
1480 \@tfor\pxrr@tempc:=%
1481 {2421:3041/2422:3042}{2423:3043/2424:3044}%
1482 {2425:3045/2426:3046}{2427:3047/2428:3048}%
1483 {2429:3049/242A:304A}{2443:3063/2444:3064}%
1484 {2463:3083/2464:3084}{2465:3085/2466:3086}%
1485 {2467:3087/2468:3088}{246E:308E/246F:308F}%
1486 {2521:30A1/2522:30A2}{2523:30A3/2524:30A4}%
1487 {2525:30A5/2526:30A6}{2527:30A7/2528:30A8}%
1488 {2529:30A9/252A:30AA}{2543:30C3/2544:30C4}%
1489 {2563:30E3/2564:30E4}{2565:30E5/2566:30E6}%
1490 {2567:30E7/2568:30E8}{256E:30EE/256F:30EF}%
1491 \do{%
1492 \expandafter\pxrr@assign@nonsmall\pxrr@tempc\pxrr@nil
1493 }

```

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。

```

1494 \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。

```
1495 \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。

```
1496 \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）。

```
1497 \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。

```
1498 \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。
  1499 \let\pxrr@compose@block@do\pxrr@compose@oneside@block@do
  1500 \pxrr@compose@block@pre{#1}{#2}{#3}{}%
  1501 }
```

こちらが本体。

```
1502 % #4 は空
1503 \def\pxrr@compose@oneside@block@do#1#2#3#4{%
1504 \setbox\pxrr@boxa\pxrr@hbox{#2}%
1505 \edef\pxrr@ck@body@natwd{\the\wd\pxrr@boxa}%
1506 \let\pxrr@ck@locate\pxrr@locate@inner
1507 \setbox\pxrr@boxr\pxrr@hbox{%
1508 \pxrr@use@ruby@font
1509 #3%
1510 }%
1511 \@tempdima\wd\pxrr@boxr
1512 \advance\@tempdima-\wd\pxrr@boxa
1513 \ifdim\pxrr@epsilon<\@tempdima
```

ルビ文字列の方が長い場合、親文字列をルビ文字列の長さに合わせて均等割りで組み直す。`\pxrr@?space` は `\pxrr@evenspace@int` が返す値のままよい。「拡張肩付き」指定の場合、前側の突出を抑止する。

```
1514 \pxrr@protrtrue
1515 \let\pxrr@locate@temp#1%
1516 \ifnum\pxrr@athead>\@ne
1517 \ifnum\pxrr@locate@temp=\pxrr@locate@inner
1518 \let\pxrr@locate@temp\pxrr@locate@head
1519 \fi
1520 \fi
1521 \let\pxrr@ck@locate\pxrr@locate@temp
```

```

1522 \pxrr@decompose{#2}%
1523 \edef\pxrr@natwd{\the\wd\pxrr@boxa}%
1524 \pxrr@evenspace@int\pxrr@locate@temp\pxrr@boxa\relax
1525 {\wd\pxrr@boxr}%
1526 \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。

```

1527 \pxrr@protrfalse
1528 \ifnum\pxrr@athead=\z@
1529 \pxrr@decompose{#3}%
1530 \edef\pxrr@natwd{\the\wd\pxrr@boxr}%
1531 \pxrr@evenspace@int{#1}\pxrr@boxr
1532 \pxrr@use@ruby@font{\wd\pxrr@boxa}%
1533 \pxrr@adjust@margin
1534 \fi
1535 \let\pxrr@bspace\pxrr@zeropt
1536 \let\pxrr@aspace\pxrr@zeropt
1537 \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）。

```

1538 \pxrr@protrfalse
1539 \let\pxrr@bspace\pxrr@zeropt
1540 \let\pxrr@aspace\pxrr@zeropt
1541 \fi\fi

```

実際に組版を行う。

```

1542 \setbox\z@\hbox{%
1543 \ifnum\pxrr@side=\z@
1544 \raise\pxrr@ruby@raise\box\pxrr@boxr
1545 \else
1546 \lower\pxrr@ruby@lower\box\pxrr@boxr
1547 \fi
1548 }%
1549 \ifnum \ifpxrr@combo\pxrr@ck@ruby@combo\else\z@\fi >\z@
1550 \pxrr@ck@compose{#2}%
1551 \fi
1552 \ht\z@\z@ \dp\z@\z@
1553 \@tempdima\wd\z@
1554 \setbox\pxrr@boxr\hbox{%
1555 \box\z@
1556 \kern-\@tempdima
1557 \box\pxrr@boxa
1558 }%

```

\ifpxrr@any@protr を設定する。

```

1559 \ifpxrr@protr
1560 \pxrr@any@protrtrue

```

```
1561 \fi
1562 }
```

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

```
1563 \def\pxrr@compose@twoside@block{%
1564 \let\pxrr@compose@block@do\pxrr@compose@twoside@block@do
1565 \pxrr@compose@block@pre
1566 }
1567 \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。

```
1568 \setbox\pxrr@boxa\pxrr@hbox{#2}%
1569 \edef\pxrr@ck@body@natwd{\the\wd\pxrr@boxa}%
1570 \let\pxrr@ck@locate\pxrr@locate@inner
1571 \setbox\pxrr@boxr\pxrr@hbox{%
1572 \pxrr@use@ruby@font
1573 #3%
1574 }%
1575 \setbox\pxrr@boxb\pxrr@hbox{%
1576 \pxrr@use@ruby@font
1577 #4%
1578 }%
```

「何れかのルビが親文字列より長いか」を検査する。

```
1579 \@tempwafalse
1580 \@tempdima\wd\pxrr@boxr
1581 \advance\@tempdima-\wd\pxrr@boxa
1582 \ifdim\pxrr@epsilon<\@tempdima \@tempwatrue \fi
1583 \@tempdima\wd\pxrr@boxb
1584 \advance\@tempdima-\wd\pxrr@boxa
1585 \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。)

```
1586 \if@tempwa
1587 \pxrr@protrtrue
```

「拡張肩付き」指定の場合、前側の突出を抑止する。

```
1588 \let\pxrr@locate@temp#1%
1589 \ifnum\pxrr@athead>\@ne
1590 \ifnum\pxrr@locate@temp=\pxrr@locate@inner
1591 \let\pxrr@locate@temp\pxrr@locate@head
1592 \fi
1593 \fi
1594 \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。

```

1595 \ifdim\wd\pxrr@boxr<\wd\pxrr@boxb
1596 \pxrr@compose@twoside@block@sub{#2}{#3}%
1597 \pxrr@boxr\pxrr@boxb
1598 \else
1599 \pxrr@compose@twoside@block@sub{#2}{#4}%
1600 \pxrr@boxb\pxrr@boxr
1601 \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。

```

1602 \else
1603 \pxrr@protrfalse

```

肩付きルビの場合は組み直しを行わない。

```

1604 \ifnum\pxrr@athead=\z@
1605 \@tempdima\wd\pxrr@boxa
1606 \advance\@tempdima-\wd\pxrr@boxr
1607 \ifdim\pxrr@epsilon<\@tempdima
1608 \pxrr@decompose{#3}%
1609 \edef\pxrr@natwd{\the\wd\pxrr@boxr}%
1610 \pxrr@evenspace@int{#1}\pxrr@boxr
1611 \pxrr@use@ruby@font{\wd\pxrr@boxa}%
1612 \pxrr@adjust@margin
1613 \fi
1614 \@tempdima\wd\pxrr@boxa
1615 \advance\@tempdima-\wd\pxrr@boxb
1616 \ifdim\pxrr@epsilon<\@tempdima
1617 \pxrr@decompose{#4}%
1618 \edef\pxrr@natwd{\the\wd\pxrr@boxb}%
1619 \pxrr@evenspace@int{#1}\pxrr@boxb
1620 \pxrr@use@ruby@font{\wd\pxrr@boxa}%
1621 \pxrr@adjust@margin
1622 \fi
1623 \fi

```

\pxrr@?space はゼロに設定する。

```

1624 \let\pxrr@bspace\pxrr@zeropt
1625 \let\pxrr@aspace\pxrr@zeropt
1626 \fi

```

実際に組版を行う。

```

1627 \setbox\z@\hbox{%
1628 \@tempdima\wd\pxrr@boxr
1629 \raise\pxrr@ruby@raise\box\pxrr@boxr
1630 \kern-\@tempdima
1631 \lower\pxrr@ruby@lower\box\pxrr@boxb
1632 }%
1633 \ifnum \ifpxrr@combo\pxrr@ck@ruby@combo\else\z@\fi >\z@
1634 \pxrr@ck@compose{#2}%
1635 \fi

```

```

1636 \ht\z@\z@ \dp\z@\z@
1637 \@tempdima\wd\z@
1638 \setbox\pxrr@boxr\hbox{%
1639   \box\z@
1640   \kern-\@tempdima
1641   \box\pxrr@boxa
1642 }%
1643 }

```

`\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。

```
1644 \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。

```
1645 \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。

```

1646 \def\pxrr@compose@twoside@block@sub#1#2#3#4{%
1647   \pxrr@decompose{#1}%
1648   \edef\pxrr@natwd{\the\wd\pxrr@boxa}%
1649   \pxrr@evenspace@int\pxrr@locate@temp\pxrr@boxa\relax{\wd#4}%
1650   \@tempdima\wd#4%
1651   \advance\@tempdima-\pxrr@bspace\relax
1652   \advance\@tempdima-\pxrr@aspace\relax
1653   \edef\pxrr@body@wd{\the\@tempdima}%
1654   \advance\@tempdima-\wd#3%
1655   \ifdim\pxrr@epsilon<\@tempdima
1656     \edef\pxrr@restore@margin@values{%
1657       \edef\noexpand\pxrr@bspace{\pxrr@bspace}%
1658       \edef\noexpand\pxrr@aspace{\pxrr@aspace}%
1659     }%
1660     \pxrr@decompose{#2}%
1661     \edef\pxrr@natwd{\the\wd#3}%
1662     \pxrr@evenspace@int\pxrr@locate@temp#3%
1663     \pxrr@use@ruby@font{\pxrr@body@wd}%
1664     \pxrr@adjust@margin
1665     \pxrr@restore@margin@values
1666     \setbox#3\hbox{%
1667       \kern\pxrr@bspace\relax
1668       \box#3%
1669     }%
1670   \else
1671     \ifnum\pxrr@locate@temp=\pxrr@locate@head

```

```

1672     \@tempdima\z@
1673     \else\ifnum\pxrr@locate@temp=\pxrr@locate@inner
1674     \@tempdima.5\@tempdima
1675     \fi\fi
1676     \advance\@tempdima\pxrr@ospace\relax
1677     \setbox#3\hbox{%
1678     \kern\@tempdima
1679     \box#3%
1680     }%
1681     \fi
1682 }
1683 %     \end{macrocode}
1684 % \end{macro}
1685 %
1686 % \begin{macro}{\pxrr@compose@block@pre}
1687 % |\pxrr@compose@block@pre{|\jmeta{パターン}}|{|^A
1688 %r \jmeta{親文字}}{|\jmeta{ルビ 1}}{|\jmeta{ルビ 2}}|\Means
1689 % 親文字列・ルビ文字列の加工を行う。
1690 % \Note 両側ルビ対応のため、ルビ用引数が 2 つある。
1691 %     \begin{macrocode}
1692 \def\pxrr@compose@block@pre{%
    f 指定時は小書き仮名の変換を施す。
1693     \pxrr@cond\ifnum\pxrr@fullsize>\z@\fi{%
1694     \pxrr@compose@block@pre@a
1695     }{%
1696     \pxrr@compose@block@pre@d
1697     }%
1698 }
1699 % {パターン}{親文字}{ルビ 1}{ルビ 2}
1700 \def\pxrr@compose@block@pre@a#1#2#3#4{%
1701     \def\pxrr@compose@block@tempa{#4}%
1702     \pxrr@transform@kana\pxrr@compose@block@tempa
1703     \expandafter\pxrr@compose@block@pre@b
1704     \expandafter{\pxrr@compose@block@tempa}{#1}{#2}{#3}%
1705 }
1706 % {ルビ 2}{パターン}{親文字}{ルビ 1}
1707 \def\pxrr@compose@block@pre@b#1#2#3#4{%
1708     \def\pxrr@compose@block@tempa{#4}%
1709     \pxrr@transform@kana\pxrr@compose@block@tempa
1710     \expandafter\pxrr@compose@block@pre@c
1711     \expandafter{\pxrr@compose@block@tempa}{#1}{#2}{#3}%
1712 }
1713 % {ルビ 1}{ルビ 2}{パターン}{親文字}
1714 \def\pxrr@compose@block@pre@c#1#2#3#4{%
1715     \pxrr@compose@block@pre@d{#3}{#4}{#1}{#2}%
1716 }
1717 \def\pxrr@compose@block@pre@d{%
1718     \pxrr@cond\ifnum\pxrr@evensp=\z@\fi{%

```

```

1719 \pxrr@compose@block@pre@e
1720 }{%
1721 \pxrr@compose@block@pre@f
1722 }%
1723 }
1724 % {パターン}{親文字}
1725 \def\pxrr@compose@block@pre@e#1#2{%
1726 \pxrr@compose@block@pre@f{#1}{#2}}%
1727 }
1728 \def\pxrr@compose@block@pre@f{%
1729 \pxrr@cond\ifnum\pxrr@revensp=\z@\fi{%
1730 \pxrr@compose@block@pre@g
1731 }{%
1732 \pxrr@compose@block@do
1733 }%
1734 }
1735 % {パターン}{親文字}{ルビ 1}{ルビ 2}
1736 \def\pxrr@compose@block@pre@g#1#2#3#4{%
1737 \pxrr@compose@block@do{#1}{#2}{#3}{#4}}%
1738 }
1739 \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。

```

1740 \def\pxrr@add@protect#1{%
1741 \expandafter\pxrr@add@protect@a
1742 \csname\expandafter@gobble\string#1\space\endcsname#1%
1743 }
1744 \def\pxrr@add@protect@a#1#2{%
1745 \let#1=#2%
1746 \def#2{\pxrr@check@protect\protect#1}%
1747 }
1748 \def\pxrr@check@protect{%
1749 \ifx\protect\@typeset@protect
1750 \expandafter@gobble
1751 \fi
1752 }

```

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。

```
1753 \let\pxrr@body@input\@empty
```

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

```
1754 \def\pxrr@prepare@fallback#1{%
1755   \pxrr@fatal@errorfalse
1756   \def\pxrr@body@input{#1}%
1757 }
```

`\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。

```
1758 \def\pxrr@fallback{%
1759   \pxrr@body@input
1760 }
```

`\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。

```
1761 \def\pxrr@if@alive{%
1762   \ifpxrr@fatal@error \expandafter\@gobble
1763   \else \expandafter\@firstofone
1764   \fi
1765 }
```

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。

```
1766 \def\pxrr@end@kinsoku{0}
```

`\pxrr@ruby@scan` 片側ルビ用の先読み処理。

```
1767 \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。

```
1768   \def\pxrr@tempc{%
1769     \edef\pxrr@end@kinsoku{\the\pxrr@cntr}%
1770     \pxrr@do@proc{#1}{#2}%
1771   }%
1772   \pxrr@check@kinsoku\pxrr@tempc
1773 }
```


`\pxrr@truby@scan` 両側ルビ用の先読み処理。

```
1774 \def\pxrr@truby@scan#1#2#3{%
1775   \def\pxrr@tempc{%
1776     \edef\pxrr@end@kinsoku{\the\pxrr@cntr}%
1777     \pxrr@do@proc{#1}{#2}{#3}%
1778   }%
1779   \pxrr@check@kinsoku\pxrr@tempc
1780 }
```

`\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。

```
1781 \def\pxrr@check@kinsoku#1{%
1782   \let\pxrr@tempb#1%
1783   \futurelet\pxrr@token\pxrr@check@kinsoku@a
1784 }
1785 \def\pxrr@check@kinsoku@a{%
1786   \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。

```
1787   \ifpxrr@abody\else
1788     \ifnum\pxrr@cntr=\@ne
1789       \pxrr@cntr\tw@
1790     \fi
1791   \fi
1792   \ifcase\pxrr@cntr
1793     \pxrr@cntr\z@
1794     \expandafter\pxrr@tempb
1795   \or
1796     \pxrr@cntr\@MM
1797     \expandafter\pxrr@tempb
1798   \else
1799     \expandafter\pxrr@check@kinsoku@b
1800   \fi
1801 }
```

`\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。

```
1802 \def\pxrr@check@kinsoku@b#1{%
1803   \pxrr@check@kinsoku@c#1#1%
1804 }
1805 \def\pxrr@check@kinsoku@c#1{%
1806   \pxrr@get@prebreakpenalty\pxrr@cntr{‘#1}%
1807   \pxrr@tempb
1808 }
```

`\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。

```
1809 \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。)

```
1810 \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。

```
1811 \ifcat\noexpand##1\relax
1812 \pxrr@cntr\z@
1813 \else\ifcat\noexpand##1\noexpand~%
1814 \pxrr@cntr\z@
1815 \else\ifcat\noexpand##1A%
1816 \pxrr@cntr@ne
1817 \else\ifcat\noexpand##10%
1818 \pxrr@cntr@ne
1819 \else
```

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

```
1820 \pxrr@cntr\z@
1821 \expandafter\pxrr@check@char@a\meaning##1#2\pxrr@nil
1822 \fi\fi\fi\fi
1823 }%
1824 \def\pxrr@check@char@a##1#2##2\pxrr@nil{%
1825 \ifcat @##10%
1826 \pxrr@cntr\tw@
1827 \fi
1828 }%
1829 }
```

規定の引数を用意して「定義マクロ」を呼ぶ。

```
1830 \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`。)

```
1831 \let\pxrr@auto@penalty\z@
```

`\pxrr@auto@icspace` 文字間の空き。寸法値マクロ。

```
1832 \let\pxrr@auto@icspace\pxrr@zeropt
```

`\pxrr@intr@amount` 進入の幅。寸法値マクロ。

```
1833 \let\pxrr@intr@amount\pxrr@zeropt
```

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

```
1834 \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。
1835 \ifpxrr@bnobr
1836 \let\pxrr@auto@penalty\@MM
1837 \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。
1838 \else
1839 \let\pxrr@auto@penalty\z@
1840 \if :\pxrr@bscomp
1841 \let\pxrr@auto@icspace\pxrr@iaiskip
1842 \else\if .\pxrr@bscomp
1843 \let\pxrr@auto@icspace\pxrr@zeropt
1844 \else
1845 \let\pxrr@auto@icspace\pxrr@iiskip
1846 \fi\fi
1847 \fi
1848 }
```

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

```
1849 \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。
1850 \if :\pxrr@bscomp\else
1851 \pxrr@bnobrtrue
1852 \fi
1853 \ifpxrr@bnobr
1854 \let\pxrr@auto@penalty\@MM
1855 \let\pxrr@auto@icspace\pxrr@zeropt
1856 \else
    この分岐は和欧文間空白挿入指定 (:) に限る。
1857 \let\pxrr@auto@penalty\z@
1858 \let\pxrr@auto@icspace\pxrr@iaiskip
1859 \fi
1860 }
```

4.17.1 前側進入処理

`\pxrr@intrude@head` 前側の進入処理。

```
1861 \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。)
1862 \ifpxrr@ghost\else
    実効の進入幅は \pxrr@bintr と \pxrr@bspace の小さい方。
1863 \let\pxrr@intr@amount\pxrr@bspace
```

```

1864 \ifdim\pxrr@bintr<\pxrr@intr@amount\relax
1865 \let\pxrr@intr@amount\pxrr@bintr
1866 \fi

```

\pxrr@auto@* の設定法は和文ルビと欧文ルビで処理が異なる。

```

1867 \ifpxrr@abody
1868 \pxrr@intrude@setauto@a
1869 \else
1870 \pxrr@intrude@setauto@j
1871 \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)。

```

1872 \ifpxrr@par@head
1873 \ifpxrr@bfintr
1874 \hskip-\pxrr@intr@amount\relax
1875 \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。

```

1876 \else
1877 % \penalty\pxrr@auto@penalty\relax
1878 \hskip-\pxrr@intr@amount\relax
1879 \hskip\pxrr@auto@icspace\relax
1880 \fi
1881 \fi
1882 }

```

\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。

```

1883 \def\pxrr@put@head@penalty{%
1884 \ifpxrr@ghost\else \ifpxrr@par@head\else
1885 \ifpxrr@abody
1886 \pxrr@intrude@setauto@a
1887 \else
1888 \pxrr@intrude@setauto@j
1889 \fi
1890 \ifnum\pxrr@auto@penalty=\z@\else
1891 \pxrr@canta\lastpenalty \unpenalty
1892 \advance\pxrr@canta\pxrr@auto@penalty\relax
1893 \penalty\pxrr@canta
1894 \fi
1895 \fi\fi
1896 }

```

4.17.2 後側進入処理

\pxrr@intrude@end 末尾での進入処理。

```
1897 \def\pxrr@intrude@end{%
1898   \ifpxrr@ghost\else
```

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

```
1899   \let\pxrr@intr@amount\pxrr@aspace
1900   \ifdim\pxrr@aintr<\pxrr@intr@amount\relax
1901     \let\pxrr@intr@amount\pxrr@aintr
1902   \fi
```

\pxrr@auto@* の設定法は和文ルビと欧文ルビで処理が異なる。

```
1903   \pxrr@csletcs{ifpxrr@bnobr}{ifpxrr@anobr}%
1904   \let\pxrr@bscomp\pxrr@ascomp
1905   \ifpxrr@abody
1906     \pxrr@intrude@setauto@a
1907   \else
1908     \pxrr@intrude@setauto@j
1909   \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。

```
1910   \ifnum\pxrr@auto@penalty=\z@
1911     \let\pxrr@auto@penalty\pxrr@end@kinsoku
1912   \fi
1913   \ifpxrr@afintr
```

段落末尾での進入を許す場合。

```
1914     \ifnum\pxrr@auto@penalty=\z@\else
1915       \penalty\pxrr@auto@penalty\relax
1916     \fi
1917     \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。

```
1918     \hskip\pxrr@auto@icspace\relax
1919     \ifnum\pxrr@auto@penalty=\z@\else
1920       \penalty\@MM
1921     \fi
1922   \else
```

段落末尾での進入を許さない場合。

```
1923     \@tempkipa-\pxrr@intr@amount\relax
1924     \advance\@tempkipa\pxrr@auto@icspace\relax
1925     \ifnum\pxrr@auto@penalty=\z@\else
1926       \penalty\pxrr@auto@penalty\relax
1927     \fi
1928     \hskip\@tempkipa
1929     \ifnum\pxrr@auto@penalty=\z@\else
1930       \penalty\@MM
1931     \fi
```

```

1932   \fi
1933   \fi
1934 }

```

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。

```

1935 \AtBeginDocument{%
1936   \providecommand*\ruby{\jruby}%
1937 }
1938 \newcommand*\jruby{%
1939   \pxrr@jprologue
1940   \pxrr@trubyfalse
1941   \pxrr@ruby
1942 }

```

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

```

1943 \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。

```

1944 \newcommand*\aruby{%
1945   \pxrr@aprologue
1946   \pxrr@trubyfalse
1947   \pxrr@ruby
1948 }
1949 \pxrr@add@protect\aruby

```

`\truby` 和文両側ルビの公開命令。

```

1950 \newcommand*\truby{%
1951   \pxrr@jprologue
1952   \pxrr@trubytrue
1953   \pxrr@ruby
1954 }
1955 \pxrr@add@protect\truby

```

`\atruby` 欧文両側ルビの公開命令。

```

1956 \newcommand*\atruby{%
1957   \pxrr@aprologue
1958   \pxrr@trubytrue
1959   \pxrr@ruby
1960 }
1961 \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。

```

1962 \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 1963 \let\pxrr@option\@empty
                1964 \let\pxrr@exoption\@empty
```

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

```
\pxrr@do@scan 1965 \let\pxrr@do@proc\@empty
                1966 \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。

```
1967 \def\pxrr@ruby{%
1968   \@testopt\pxrr@ruby@a{}%
1969 }
1970 \def\pxrr@ruby@a[#1]{%
1971   \def\pxrr@option{#1}%
1972   \@testopt\pxrr@ruby@b{}%
1973 }
1974 \def\pxrr@ruby@b[#1]{%
1975   \def\pxrr@exoption{#1}%
1976   \ifpxrr@truby
1977     \let\pxrr@do@proc\pxrr@truby@proc
1978     \let\pxrr@do@scan\pxrr@truby@scan
1979   \else
1980     \let\pxrr@do@proc\pxrr@ruby@proc
1981     \let\pxrr@do@scan\pxrr@ruby@scan
1982   \fi
1983   \pxrr@ruby@c
1984 }
1985 \def\pxrr@ruby@c{%
1986   \ifpxrr@ghost
1987     \expandafter\pxrr@do@proc
1988   \else
1989     \expandafter\pxrr@do@scan
1990   \fi
1991 }
```

`\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。

```
1992 \def\pxrr@mode@is@switching{%
1993   \if M\pxrr@mode T%
1994   \else\if J\pxrr@mode T%
1995   \else F%
1996   \fi\fi T%
1997 }
```

`\pxrr@bind@param` “呼出時変数” へのコピーを行う。

```
1998 \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。

```
1999 \ifpxrr@apply@combo
2000   \pxrr@apply@combofalse
2001   \pxrr@combotrue
2002   \pxrr@ck@bind@param
2003 \else
2004   \pxrr@combofalse
2005 \fi
2006 \let\pxrr@c@ruby@font\pxrr@ruby@font
2007 \let\pxrr@c@size@ratio\pxrr@size@ratio
2008 \let\pxrr@c@inter@gap\pxrr@inter@gap
2009 }
```

`\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。

```
2010 \def\pxrr@ruby@proc#1#2{%
2011   \pxrr@prepare@fallback{#1}%
      フォントサイズの変数を設定して、
2012   \pxrr@bind@param
2013   \pxrr@assign@fsize
      オプションを解析する。
2014   \pxrr@parse@option\pxrr@option
      ルビ文字入力をグループ列に分解する。
2015   \pxrr@decompbar{#2}%
2016   \let\pxrr@ruby@list\pxrr@res
2017   \edef\pxrr@ruby@count{\the\pxrr@cntr}%
2018   \let\pxrr@sruby@list\relax
      親文字入力をグループ列に分解する。
2019   \pxrr@decompbar{#1}%
2020   \let\pxrr@body@list\pxrr@res
2021   \edef\pxrr@body@count{\the\pxrr@cntr}%
      安全モードに関する処理を行う。
2022   \ifpxrr@safe@mode
2023     \pxrr@setup@safe@mode
2024   \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。
2025   \if\pxrr@mode@is@switching
2026     \pxrr@resolve@mode
2027   \fi
2028 \ifpxrr@Debug
2029   \pxrr@debug@show@input
2030 \fi
      入力検査を行い、パスした場合は組版処理に進む。
2031   \pxrr@if@alive{%
```



```

2032 \if g\pxrr@mode
2033 \pxrr@ruby@check@g
2034 \pxrr@if@alive{%
2035 \ifnum\pxrr@body@count>\@ne
2036 \pxrr@ruby@main@mg
2037 \else
2038 \pxrr@ruby@main@g
2039 \fi
2040 }%
2041 \else
2042 \pxrr@ruby@check@m
2043 \pxrr@if@alive{\pxrr@ruby@main@m}%
2044 \fi
2045 }%

後処理を行う。

2046 \pxrr@ruby@exit
2047 }

```

`\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。

```

2048 \def\pxrr@truby@proc#1#2#3{%
2049 \pxrr@prepare@fallback{#1}%

  フォントサイズの変数を設定して、
2050 \pxrr@bind@param
2051 \pxrr@assign@fsize

  オプションを解析する。
2052 \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。
2053 \def\pxrr@all@input{#{1}{#2}{#3}}%

  入力文字列のグループ分解を行う。
2054 \pxrr@decompbar{#3}%
2055 \let\pxrr@sruby@list\pxrr@res
2056 \edef\pxrr@sruby@count{\the\pxrr@cntr}%
2057 \pxrr@decompbar{#2}%
2058 \let\pxrr@ruby@list\pxrr@res
2059 \edef\pxrr@ruby@count{\the\pxrr@cntr}%
2060 \pxrr@decompbar{#1}%
2061 \let\pxrr@body@list\pxrr@res
2062 \edef\pxrr@body@count{\the\pxrr@cntr}%

  安全モードに関する処理を行う。
2063 \ifpxrr@safe@mode
2064 \pxrr@setup@safe@mode
2065 \fi
2066 \if\pxrr@mode@is@switching
2067 \pxrr@resolve@mode

```

```

2068 \fi
2069 \ifpxrrDebug
2070 \pxrr@debug@show@input
2071 \fi

```

入力検査を行い、パスした場合は組版処理に進む。

```

2072 \pxrr@if@alive{%
2073   \if g\pxrr@mode
2074     \pxrr@ruby@check@tg
2075     \pxrr@if@alive{\pxrr@ruby@main@tg}%
2076   \else
2077     \pxrr@ruby@check@tm
2078     \pxrr@if@alive{\pxrr@ruby@main@tm}%
2079   \fi
2080 }%

```

後処理を行う。

```

2081 \pxrr@ruby@exit
2082 }

```

`\pxrr@setup@safe@mode` 安全モード用の設定。

```

2083 \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。

```

2084 \let\pxrr@mode=g\relax
2085 \pxrr@unite@group\pxrr@body@list
2086 \def\pxrr@body@count{1}%
2087 \pxrr@unite@group\pxrr@ruby@list
2088 \def\pxrr@ruby@count{1}%
2089 \ifx\pxrr@sruby@list\relax\else
2090   \pxrr@unite@group\pxrr@sruby@list
2091   \def\pxrr@sruby@count{1}%
2092 \fi

```

“文字単位のスキャン”が必要な機能を無効にする。

```

2093 \chardef\pxrr@evensp\z@
2094 \chardef\pxrr@revensp\z@
2095 \chardef\pxrr@fullsize\z@
2096 }

```

`\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。

```

2097 \def\pxrr@resolve@mode{%
2098   \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、

```

2099   \ifnum\pxrr@ruby@count=\@ne
2100     \let\pxrr@pre\pxrr@decompose
2101     \let\pxrr@post\relax

```

```

2102     \pxrr@body@list
2103     \ifnum\pxrr@cntr=\@ne\else
2104         \let\pxrr@mode=g%
2105     \fi
2106 \fi

```

それ以外はモノルビ・熟語ルビを選択する。

```

2107     \if M\pxrr@mode \let\pxrr@mode=m\fi
2108     \if J\pxrr@mode \let\pxrr@mode=j\fi
2109 \ifpxrrDebug
2110     \pxrr@debug@show@resolve@mode
2111 \fi

```

\pxrr@check@option で行っている調整をやり直す。

```

2112     \if g\pxrr@mode
2113         \chardef\pxrr@athead\z@
2114     \fi
2115     \if g\pxrr@mode\else
2116         \chardef\pxrr@evensp\@ne
2117     \fi
2118 \else
2119     \pxrr@fatal@bad@switching
2120 \fi
2121 }

```

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。

```

2122 \def\pxrr@ruby@check@g{%
2123     \ifnum\pxrr@body@count=\pxrr@ruby@count\relax
2124     \ifnum\pxrr@body@count=\@ne\else
2125         \ifpxrr@abody
2126             \pxrr@fatal@bad@movable
2127         \else\ifnum\pxrr@extra=\z@
2128             \pxrr@fatal@na@movable
2129         \fi\fi
2130     \fi
2131 \else
2132     \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
2133 \fi
2134 }

```

`\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。

```

2135 \def\pxrr@ruby@check@m{%

```

```

2136 \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。
2137 \let\pxrr@pre\pxrr@decompose
2138 \let\pxrr@post\relax
2139 \pxrr@body@list
2140 \let\pxrr@body@list\pxrr@res
2141 \edef\pxrr@body@count{\the\pxrr@cntr}%
2142 \ifnum\pxrr@body@count=\pxrr@ruby@count\relax\else
2143 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
2144 \fi
2145 \else
2146 \pxrr@fatal@bad@mono
2147 \fi
2148 }

```

\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。

```

2149 \def\pxrr@ruby@check@tg{%
2150 \ifnum\pxrr@body@count=\pxrr@ruby@count\relax\else
2151 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
2152 \fi
2153 \ifnum\pxrr@body@count=\pxrr@sruby@count\relax\else
2154 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@sruby@count
2155 \fi
2156 \pxrr@if@alive{%
2157 \ifnum\pxrr@body@count=\@ne\else
2158 \ifpxrr@abody
2159 \pxrr@fatal@bad@movable
2160 \else\ifnum\pxrr@extra=\z@
2161 \pxrr@fatal@na@movable
2162 \fi\fi
2163 \fi
2164 }%
2165 }

```

\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。

```

2166 \def\pxrr@ruby@check@tm{%
2167 \ifnum\pxrr@body@count=\@ne
2168 \let\pxrr@pre\pxrr@decompose
2169 \let\pxrr@post\relax
2170 \pxrr@body@list
2171 \let\pxrr@body@list\pxrr@res
2172 \edef\pxrr@body@count{\the\pxrr@cntr}%
2173 \ifnum\pxrr@body@count=\pxrr@ruby@count\relax\else
2174 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
2175 \fi
2176 \ifnum\pxrr@body@count=\pxrr@sruby@count\relax\else
2177 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@sruby@count
2178 \fi

```

```

2179 \else
2180   \pxrr@fatal@bad@mono
2181 \fi
2182 }

```

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。

```

2183 \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。

```

2184 \def\pxrr@check@par@head{%
2185   \ifvmode
2186     \pxrr@par@headtrue
2187   \else
2188     \pxrr@par@headfalse
2189   \fi
2190 }

```

`\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。

```

2191 \def\pxrr@if@last#1#2#3{%
2192   \ifx#3\pxrr@post #1%
2193   \else #2%
2194   \fi
2195   #3%
2196 }

```

`\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。

```

2197 \def\pxrr@inter@mono{%
2198   \hskip\pxrr@iiskip\relax
2199 }

```

`\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。

```

2200 \def\pxrr@takeout@any@protr@nocolor{%
2201   \ifpxrr@any@protr
2202     \aftergroup\pxrr@any@protrtrue
2203   \fi
2204 }

```

`color` 使用時の定義。

```

2205 \def\pxrr@takeout@any@protr{%
2206   \ifpxrr@any@protr
2207     \aftergroup\pxrr@takeout@any@protr@a
2208   \fi
2209 }
2210 \def\pxrr@takeout@any@protr@a{%
2211   \aftergroup\pxrr@any@protrtrue
2212 }

```

\pxrr@ruby@main@m モノルビ。

```

2213 \def\pxrr@ruby@main@m{%
2214   \pxrr@zip@list\pxrr@body@list\pxrr@ruby@list
2215   \let\pxrr@whole@list\pxrr@res
2216   \pxrr@check@par@head
2217   \pxrr@put@head@penalty
2218   \pxrr@any@protrfalse
2219 \ifpxrr@Debug
2220 \pxrr@debug@show@recomp
2221 \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。

```

2222 \let\pxrr@locate@head@\pxrr@locate@inner
2223 \let\pxrr@locate@end@\pxrr@locate@inner
2224 \let\pxrr@locate@sing@\pxrr@locate@inner
2225 \ifpxrr@aprotr\else
2226   \let\pxrr@locate@end@\pxrr@locate@end
2227   \let\pxrr@locate@sing@\pxrr@locate@end
2228 \fi
2229 \ifpxrr@bprotr\else
2230   \let\pxrr@locate@head@\pxrr@locate@head
2231   \let\pxrr@locate@sing@\pxrr@locate@head
2232 \fi
2233 \def\pxrr@pre##1##2{%
2234   \pxrr@if@last{%

```

単独ブロックの場合。

```

2235   \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
2236   \pxrr@intrude@head
2237   \unhbox\pxrr@boxr
2238   \pxrr@intrude@end
2239   \pxrr@takeout@any@protr
2240   }{%

```

先頭ブロックの場合。

```

2241   \pxrr@compose@block\pxrr@locate@head@{##1}{##2}%
2242   \pxrr@intrude@head
2243   \unhbox\pxrr@boxr
2244   }%
2245   }%

```

```

2246 \def\pxrr@inter##1##2{%
2247   \pxrr@if@last{%

```

末尾ブロックの場合。

```

2248     \pxrr@compose@block\pxrr@locate@end@{##1}{##2}%
2249     \pxrr@inter@mono
2250     \unhbox\pxrr@boxr
2251     \pxrr@intrude@end
2252     \pxrr@takeout@any@protr
2253   }{%

```

中間ブロックの場合。

```

2254     \pxrr@compose@block\pxrr@locate@inner{##1}{##2}%
2255     \pxrr@inter@mono
2256     \unhbox\pxrr@boxr
2257   }%
2258 }%
2259 \let\pxrr@post\@empty
2260 \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。

```

2261 \if j\pxrr@mode
2262   \ifpxrr@any@protr
2263     \pxrr@ruby@redo@j
2264   \fi
2265 \fi
2266 \unhbox\pxrr@boxr
2267 }

```

\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。

```

2268 \def\pxrr@ruby@redo@j{%
2269   \pxrr@concat@list\pxrr@body@list
2270   \let\pxrr@body@list\pxrr@res
2271   \pxrr@concat@list\pxrr@ruby@list
2272   \let\pxrr@ruby@list\pxrr@res
2273   \pxrr@zip@single\pxrr@body@list\pxrr@ruby@list
2274   \let\pxrr@whole@list\pxrr@res
2275 \ifpxrr@Debug
2276 \pxrr@debug@show@concat
2277 \fi
2278 \let\pxrr@locate@sing@\pxrr@locate@inner
2279 \ifpxrr@aprotr\else
2280   \let\pxrr@locate@sing@\pxrr@locate@end
2281 \fi
2282 \ifpxrr@bprotr\else
2283   \let\pxrr@locate@sing@\pxrr@locate@head
2284 \fi
2285 \def\pxrr@pre##1##2{%
2286   \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%

```

```

2287 \pxrr@intrude@head
2288 \unhbox\pxrr@boxr
2289 \pxrr@intrude@end
2290 }%
2291 \let\pxrr@inter\@undefined
2292 \let\pxrr@post\@empty
2293 \setbox\pxrr@boxr\pxrr@hbox{\pxrr@whole@list}%
2294 }

```

\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。

```

2295 \def\pxrr@ruby@main@g{%
2296 \pxrr@zip@list\pxrr@body@list\pxrr@ruby@list
2297 \let\pxrr@whole@list\pxrr@res
2298 \pxrr@check@par@head
2299 \pxrr@put@head@penalty
2300 \ifpxrr@Debug
2301 \pxrr@debug@show@recomp
2302 \fi
2303 \let\pxrr@locate@sing@\pxrr@locate@inner
2304 \ifpxrr@aprotr\else
2305 \let\pxrr@locate@sing@\pxrr@locate@end
2306 \fi
2307 \ifpxrr@bprotr\else
2308 \let\pxrr@locate@sing@\pxrr@locate@head
2309 \fi
2310 \def\pxrr@pre##1##2{%
2311 \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
2312 \pxrr@intrude@head
2313 \unhbox\pxrr@boxr
2314 \pxrr@intrude@end
2315 }%
2316 \let\pxrr@inter\@undefined
2317 \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。

```

2318 \pxrr@whole@list
2319 }

```

\pxrr@ruby@main@tm 両側のモノルビの場合。

```

2320 \def\pxrr@ruby@main@tm{%
2321 \pxrr@tzip@list\pxrr@body@list\pxrr@ruby@list\pxrr@sruby@list
2322 \let\pxrr@whole@list\pxrr@res
2323 \pxrr@check@par@head
2324 \pxrr@any@protrfalse
2325 \ifpxrr@Debug
2326 \pxrr@debug@show@recomp
2327 \fi

```



```

2328 \let\pxrr@locate@head@\pxrr@locate@inner
2329 \let\pxrr@locate@end@\pxrr@locate@inner
2330 \let\pxrr@locate@sing@\pxrr@locate@inner
2331 \ifpxrr@aprotr\else
2332   \let\pxrr@locate@end@\pxrr@locate@end
2333   \let\pxrr@locate@sing@\pxrr@locate@end
2334 \fi
2335 \ifpxrr@bprotr\else
2336   \let\pxrr@locate@head@\pxrr@locate@head
2337   \let\pxrr@locate@sing@\pxrr@locate@head
2338 \fi
2339 \def\pxrr@pre##1##2##3{%
2340   \pxrr@if@last{%
2341     \pxrr@compose@twoside@block\pxrr@locate@sing@
2342     {##1}{##2}{##3}%
2343     \pxrr@intrude@head
2344     \unhbox\pxrr@boxr
2345     \pxrr@intrude@end
2346     \pxrr@takeout@any@protr
2347   }{%
2348     \pxrr@compose@twoside@block\pxrr@locate@head@
2349     {##1}{##2}{##3}%
2350     \pxrr@intrude@head
2351     \unhbox\pxrr@boxr
2352   }%
2353 }%
2354 \def\pxrr@inter##1##2##3{%
2355   \pxrr@if@last{%
2356     \pxrr@compose@twoside@block\pxrr@locate@end@
2357     {##1}{##2}{##3}%
2358     \pxrr@inter@mono
2359     \unhbox\pxrr@boxr
2360     \pxrr@intrude@end
2361     \pxrr@takeout@any@protr
2362   }{%
2363     \pxrr@compose@twoside@block\pxrr@locate@inner
2364     {##1}{##2}{##3}%
2365     \pxrr@inter@mono
2366     \unhbox\pxrr@boxr
2367   }%
2368 }%
2369 \let\pxrr@post\@empty
2370 \setbox\pxrr@boxr\pxrr@hbox{\pxrr@whole@list}%
2371 \unhbox\pxrr@boxr
2372 }

```

\pxrr@ruby@main@tg 両側の単純グループビの場合。

```

2373 \def\pxrr@ruby@main@tg{%
2374   \pxrr@check@par@head

```

```

2375 \pxrr@put@head@penalty
2376 \let\pxrr@locate@sing@\pxrr@locate@inner
2377 \ifpxrr@aprotr\else
2378   \let\pxrr@locate@sing@\pxrr@locate@end
2379 \fi
2380 \ifpxrr@bprotr\else
2381   \let\pxrr@locate@sing@\pxrr@locate@head
2382 \fi
2383 \expandafter\pxrr@compose@twoside@block\expandafter\pxrr@locate@sing@
2384 \pxrr@all@input
2385 \pxrr@intrude@head
2386 \unhbox\pxrr@boxr
2387 \pxrr@intrude@end
2388 }

```

`\pxrr@ruby@main@mg` 未実装 (呼出もない)。

```
2389 \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。

```
2390 \newif\ifpxrr@ghost
```

`\pxrr@jprologue` 和文ルビ用の開始処理。

```
2391 \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。

```

2392 \ifpxrr@jghost
2393   \pxrr@jghost@char
2394   \pxrr@inhibitglue
2395 \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。

```

2396 \begingroup
2397   \pxrr@abodyfalse
2398   \pxrr@csletcs{\ifpxrr@ghost}{\ifpxrr@jghost}%

```

出力した全角空白の幅だけ戻しておく。

```

2399 \ifpxrr@jghost
2400   \setbox\pxrr@boxa\hbox{\pxrr@jghost@char}%
2401   \kern-\wd\pxrr@boxa
2402 \fi
2403 }

```

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

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

```

2404 \chardef\pxrr@aghostchar=23 % compwordmark
2405 \let\pxrr@aghost\relax
2406 \let\pxrr@aghostfont\relax
2407 \def\pxrr@setup@aghost{%
2408   \global\let\pxrr@setup@aghost\relax
2409   \IfFileExists{t1lmr.fd}{%
2410     \begingroup
2411       \fontsize{2.5}{0}\usefont{T1}{lmr}{m}{n}%
2412     \endgroup
2413     \global\pxrr@letcs\pxrr@aghostfont{T1/lmr/m/n/2.5}%
2414     \gdef\pxrr@aghost{\pxrr@aghostfont\pxrr@aghostchar}%
2415     \pxrr@force@nonpunct@achar{\pxrr@aghostchar}%
2416   }{%else
2417     \pxrr@warn{Ghost embedding for \string\aruby\space
2418       is disabled,\MessageBreak
2419       since package lmodern is missing}%
2420     \global\pxrr@aghostfalse
2421     \global\let\pxrr@aghosttrue\relax
2422   }%
2423 }
```

`\pxrr@aprologue` 欧文ルビ用の開始処理。

```

2424 \def\pxrr@aprologue{%
2425   \ifpxrr@aghost
2426     \pxrr@aghost
2427     \fi
2428   \begingroup
2429     \pxrr@abodytrue
2430     \pxrr@csletcs{ifpxrr@ghost}{ifpxrr@aghost}%
2431 }
```

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。

```

2432 \def\pxrr@ruby@exit{%
2433   \ifpxrr@fatal@error
2434     \pxrr@fallback
2435     \fi
2436   \ifpxrr@abody
2437     \expandafter\pxrr@aepilogue
2438   \else
```

```

2439 \expandafter\pxrr@jepilogue
2440 \fi
2441 }

```

`\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。

```

2442 \def\pxrr@jepilogue{%
2443 \ifpxrr@jghost
2444 \setbox\pxrr@boxa\hbox{\pxrr@jghost@char}%
2445 \kern-\wd\pxrr@boxa
2446 \fi

```

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

```

2447 \endgroup
2448 \ifpxrr@jghost
2449 \pxrr@inhibitglue
2450 \pxrr@jghost@char
2451 \fi
2452 }

```

`\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。

```

2453 \def\pxrr@aepilogue{%
2454 \endgroup
2455 \ifpxrr@aghost
2456 \pxrr@aghost
2457 \fi
2458 }

```

4.19 デバッグ用出力

```

2459 \def\pxrr@debug@show@input{%
2460 \typeout{---\pxrr@pkgname\space input:^^J%
2461 \ifpxrr@abody = \meaning\ifpxrr@abody^^J%
2462 \ifpxrr@truby = \meaning\ifpxrr@truby^^J%
2463 pxrr@ruby@fsize = \pxrr@ruby@fsize^^J%
2464 pxrr@body@zw = \pxrr@body@zw^^J%
2465 pxrr@ruby@zw = \pxrr@ruby@zw^^J%
2466 pxrr@iiskip = \pxrr@iiskip^^J%
2467 pxrr@iaiskip = \pxrr@iaiskip^^J%
2468 pxrr@htratio = \pxrr@htratio^^J%
2469 pxrr@ruby@raise = \pxrr@ruby@raise^^J%
2470 pxrr@ruby@lower = \pxrr@ruby@lower^^J%
2471 \ifpxrr@bprotr = \meaning\ifpxrr@bprotr^^J%
2472 \ifpxrr@aprotr = \meaning\ifpxrr@aprotr^^J%
2473 pxrr@side = \the\pxrr@side^^J%
2474 pxrr@evensp = \the\pxrr@evensp^^J%
2475 pxrr@fullsize = \the\pxrr@fullsize^^J%
2476 pxrr@bscomp = \meaning\pxrr@bscomp^^J%
2477 pxrr@ascomp = \meaning\pxrr@ascomp^^J%
2478 \ifpxrr@bnoobr = \meaning\ifpxrr@bnoobr^^J%

```

```

2479 ifpxrr@anobr = \meaning\ifpxrr@anobr^^J%
2480 ifpxrr@bfintr = \meaning\ifpxrr@bfintr^^J%
2481 ifpxrr@afintr = \meaning\ifpxrr@afintr^^J%
2482 pxrr@bintr = \pxrr@bintr^^J%
2483 pxrr@aintr = \pxrr@aintr^^J%
2484 pxrr@ahead = \the\pxrr@ahead^^J%
2485 pxrr@mode = \meaning\pxrr@mode^^J%
2486 ifpxrr@ahead@given = \meaning\ifpxrr@ahead@given^^J%
2487 ifpxrr@mode@given = \meaning\ifpxrr@mode@given^^J%
2488 pxrr@body@list = \meaning\pxrr@body@list^^J%
2489 pxrr@body@count = \@nameuse{pxrr@body@count}^^J%
2490 pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
2491 pxrr@ruby@count = \@nameuse{pxrr@ruby@count}^^J%
2492 pxrr@end@kinsoku = \pxrr@end@kinsoku^^J%
2493 ----
2494 }%
2495 }
2496 \def\pxrr@debug@show@recomp{%
2497 \typeout{----\pxrr@pkgname\space recomp:^^J%
2498 pxrr@body@list = \meaning\pxrr@body@list^^J%
2499 pxrr@body@count = \pxrr@body@count^^J%
2500 pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
2501 pxrr@ruby@count = \pxrr@ruby@count^^J%
2502 pxrr@res = \meaning\pxrr@res^^J%
2503 ----
2504 }%
2505 }
2506 \def\pxrr@debug@show@concat{%
2507 \typeout{----\pxrr@pkgname\space concat:^^J%
2508 pxrr@body@list = \meaning\pxrr@body@list^^J%
2509 pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
2510 pxrr@whole@list = \meaning\pxrr@whole@list^^J%
2511 ----
2512 }%
2513 }
2514 \def\pxrr@debug@show@resolve@mode{%
2515 \typeout{----\pxrr@pkgname\space resolve-mode:
2516 \meaning\pxrr@mode}%
2517 }

```

5 実装 (圏点関連)

5.1 エラーメッセージ

指定の名前の圏点文字が未登録の場合。

```

2518 \def\pxrr@warn@na@kmark#1{%
2519 \pxrr@warn{Unavailable kenten mark '#1'}%
2520 }

```

パラメタ設定命令で無効な値が指定された場合。

```
2521 \def\pxrr@err@invalid@value#1{%
2522   \pxrr@error{Invalid value '#1'}%
2523   {\@eha}%
2524 }
```

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。

```
2525 \let\pxrr@k@ymark\@undefined
```

`\pxrr@k@ysmark` 横組の副の圏点マークのコード。

```
2526 \let\pxrr@k@ysmark\@undefined
```

`\pxrr@k@tmark` 縦組の主の圏点マークのコード。

```
2527 \let\pxrr@k@tmark\@undefined
```

`\pxrr@k@tsmark` 縦組の副の圏点マークのコード。

```
2528 \let\pxrr@k@tsmark\@undefined
```

圏点マークの初期値の設定。

```
2529 \AtEndOfPackage{%
2530   \pxrr@k@get@mark\pxrr@k@ymark{bullet*}%
2531   \pxrr@k@get@mark\pxrr@k@ysmark{sesame*}%
2532   \pxrr@k@get@mark\pxrr@k@tmark{sesame*}%
2533   \pxrr@k@get@mark\pxrr@k@tsmark{bullet*}%
2534 }
```

`\pxrr@k@ruby@font` 圏点用フォント切替命令。

```
2535 \let\pxrr@k@ruby@font\@empty
```

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

```
2536 \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。

```
2537 \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。

```
2538 \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。

```
2539 \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。

```
2540 \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。

```
2541 \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。

```
2542 \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。

```
2543 \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。

```
2544 \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。

```
2545 \chardef\pxrr@k@full=0
```

`\pxrr@k@the@mark` 適用される圏点マークの命令。

```
2546 \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。

```
2547 \newif\ifpxrr@avail@UTF
```

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

```
2548 \def\pxrr@decide@avail@UTF{%
```

```
2549   \global\let\pxrr@decide@avail@UTF\relax
```

```
2550   \ifx\UTF\@undefined \global\pxrr@avail@UTFfalse
```

```
2551   \else \global\pxrr@avail@UTFtrue
```

```
2552   \fi
```

```
2553 }
```

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
```

```
2554 \def\pxrr@k@decompose#1{%
```

```
2555   \let\pxrr@res\@empty
```

```

2556 \pxrr@cntr=\z@
2557 \pxrr@k@decompose@loopa#1\pxrr@end
2558 }
2559 \def\pxrr@k@decompose@loopa{%
2560 \futurelet\pxrr@token\pxrr@k@decompose@loopb
2561 }
2562 \def\pxrr@k@decompose@loopb{%
2563 \pxrr@cond\ifx\pxrr@token\pxrr@end\fi{%
2564 \pxrr@appto\pxrr@res{\pxrr@post}%
2565 }{\pxrr@if@kspan@cmd\pxrr@token{%
2566 \pxrr@k@decompose@special\pxrr@k@decompose@kspan
2567 }{\pxrr@if@ruby@cmd\pxrr@token{%
2568 \pxrr@k@decompose@special\pxrr@k@decompose@ruby
2569 }{\pxrr@if@truby@cmd\pxrr@token{%
2570 \pxrr@k@decompose@special\pxrr@k@decompose@truby
2571 }{\pxrr@if@kenten@cmd\pxrr@token{%
2572 \pxrr@k@decompose@special\pxrr@k@decompose@kenten
2573 }{\pxrr@cond\ifx\pxrr@token\@sptoken\fi{%
2574 \pxrr@k@decompose@loope
2575 }{%
2576 \pxrr@setok{\pxrr@ifx{\pxrr@token\bgroup}}}%
2577 \pxrr@k@decompose@loopc
2578 }}}}]}%
2579 }
2580 \def\pxrr@k@decompose@loopc#1{%
2581 \pxrr@appto\pxrr@res{\pxrr@entry}%
2582 \ifpxrr@ok
2583 \pxrr@appto\pxrr@res{{{#1}}}%
2584 \else
2585 \pxrr@appto\pxrr@res{#{#1}}%
2586 \fi
2587 \pxrr@k@decompose@loopd
2588 }
2589 \def\pxrr@k@decompose@loopd{%
2590 \advance\pxrr@cntr\@ne
2591 \pxrr@k@decompose@loopa
2592 }
2593 \expandafter\def\expandafter\pxrr@k@decompose@loope\space{%
2594 \pxrr@okfalse
2595 \pxrr@k@decompose@loopc{ }%
2596 }
2597 \def\pxrr@k@decompose@special#1#2#3{%
2598 #1{#2}%
2599 }
2600 \def\pxrr@k@decompose@kspan#1#2{%
2601 \pxrr@appto\pxrr@res{\pxrr@entry@kspan{#1{#2}}}%
2602 \pxrr@k@decompose@loopd
2603 }
2604 \def\pxrr@k@decompose@ruby#1#2#3{%

```



```

2605 \pxrr@appto\pxrr@res{\pxrr@entry@ruby{#1{#2}{#3}}}%
2606 \pxrr@k@decompose@loopd
2607 }
2608 \def\pxrr@k@decompose@truby#1#2#3#4{%
2609 \pxrr@appto\pxrr@res{\pxrr@entry@ruby{#1{#2}{#3}{#4}}}%
2610 \pxrr@k@decompose@loopd
2611 }
2612 \def\pxrr@k@decompose@kenten#1#2{%
2613 \pxrr@appto\pxrr@res{\pxrr@entry@kenten{#1{#2}}}%
2614 \pxrr@k@decompose@loopd
2615 }
2616 \def\pxrr@cmd@ruby{\jruby}
2617 \def\pxrr@cmd@kenten{jkenten}
2618 \def\pxrr@if@ruby@cmd#1{%
2619 \if \ifcat\noexpand#1\relax
2620 \ifx#1\pxrr@cmd@ruby T%
2621 \else\ifx#1\jruby T%
2622 \else\ifx#1\aruby T%
2623 \else F%
2624 \fi\fi\fi
2625 \else F%
2626 \fi T\expandafter\@firstoftwo
2627 \else \expandafter\@secondoftwo
2628 \fi
2629 }
2630 \def\pxrr@if@truby@cmd#1{%
2631 \if \ifcat\noexpand#1\relax
2632 \ifx#1\truby T%
2633 \else\ifx#1\atruby T%
2634 \else F%
2635 \fi\fi
2636 \else F%
2637 \fi T\expandafter\@firstoftwo
2638 \else \expandafter\@secondoftwo
2639 \fi
2640 }
2641 \def\pxrr@if@kspan@cmd#1{%
2642 \pxrr@cond\ifx#1\kspan\fi
2643 }
2644 \def\pxrr@if@kenten@cmd#1{%
2645 \if \ifcat\noexpand#1\relax
2646 \ifx#1\pxrr@cmd@kenten T%
2647 \else\ifx#1\jkenten T%
2648 \else F%
2649 \fi\fi
2650 \else F%
2651 \fi T\expandafter\@firstoftwo
2652 \else \expandafter\@secondoftwo
2653 \fi

```

2654 }

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。

```
2655 \newcommand*\kentensetup[1]{%
2656   \pxrr@in@setuptrue
2657   \pxrr@fatal@errorfalse
2658   \pxrr@k@parse@option{#1}%
2659   \ifpxrr@fatal@error\else
2660     \let\pxrr@k@d@side\pxrr@side
2661     \let\pxrr@k@d@mark\pxrr@k@mark
2662     \let\pxrr@k@d@full\pxrr@k@full
2663   \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。

```
2664   \pxrr@in@setupfalse
2665 }
```

`\kentenfontsetup` 対応するパラメタを設定する。

```
2666 \newcommand*\kentenfontsetup{}
2667 \def\kentenfontsetup#{%
2668   \def\pxrr@k@ruby@font
2669 }
```

`\kentensizeratio` 対応するパラメタを設定する。

```
2670 \newcommand*\kentensizeratio[1]{%
2671   \edef\pxrr@k@size@ratio{#1}%
2672 }
```

`\kentenintergap` 対応するパラメタを設定する。

```
2673 \newcommand*\kentenintergap[1]{%
2674   \edef\pxrr@k@inter@gap{#1}%
2675 }
```

`\kentenrubyintergap` 対応するパラメタを設定する。

```
2676 \newcommand*\kentenrubyintergap[1]{%
2677   \edef\pxrr@k@ruby@inter@gap{#1}%
2678 }
```

`\kentenmarkinyoko` 対応するパラメタを設定する。

```
\kentenmarkinyoko 2679 \newcommand*\kentenmarkinyoko[1]{%
\kentenmarkintate 2680   \pxrr@k@get@mark\pxrr@k@ymark{#1}%
2681 }
\kentenmarkintate 2682 \newcommand*\kentenmarkintate[1]{%
2683   \pxrr@k@get@mark\pxrr@k@ysmark{#1}%
2684 }
```

```

2685 \newcommand*\kentenmarkintate[1]{%
2686   \pxrr@k@get@mark\pxrr@k@tmark{#1}%
2687 }
2688 \newcommand*\kentensubmarkintate[1]{%
2689   \pxrr@k@get@mark\pxrr@k@tsmark{#1}%
2690 }

```

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

```

2691 \chardef\pxrr@k@ruby@combo@ruby=0
2692 \chardef\pxrr@k@ruby@combo@both=1
2693 \newcommand*\kentenrubycombination[1]{%
2694   \pxrr@letcs\pxrr@tempa{\pxrr@k@ruby@combo@#1}%
2695   \ifx\pxrr@tempa\relax
2696     \pxrr@err@invalid@value{#1}%
2697   \else
2698     \let\pxrr@k@ruby@combo\pxrr@tempa
2699   \fi
2700 }

```

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。

```

2701 \def\pxrr@k@declare@mark#1{%
2702   \global\@namedef{\pxrr@k@mark@#1}%
2703 }

```

`\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。

```

2704 \def\pxrr@k@let@mark#1{%
2705   \global\pxrr@cslet{\pxrr@k@mark@#1}%
2706 }

```

`\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。

```

2707 \def\pxrr@k@get@mark#1#2{%
2708   \futurelet\pxrr@token\pxrr@k@get@mark@a#2\pxrr@nil#1%
2709 }
2710 \def\pxrr@k@get@mark@a{%
2711   \pxrr@cond\ifcat A\noexpand\pxrr@token\fi{%
2712     \pxrr@k@get@mark@c
2713   }{%else
2714     \pxrr@k@get@mark@b
2715   }%
2716 }
2717 \def\pxrr@k@get@mark@b#1\pxrr@nil#2{%
2718   \def#2{#1}%
2719 }

```

```

2720 \def\pxrr@k@get@mark@c#1#2\pxrr@nil#3{%
2721   \ifnum'#1<128
2722     \pxrr@letcs\pxrr@tempa{\pxrr@k@mark@c#1#2}%
2723     \ifx\pxrr@tempa\relax
2724       \pxrr@warn@na@kmark{#1#2}%
2725     \else
2726       \let#3\pxrr@tempa
2727     \fi
2728 \else
2729   \pxrr@k@get@mark@b#1#2\pxrr@nil#3%
2730 \fi
2731 }

```

`\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。

```

2732 \def\pxrr@k@declare@mark@char#1#2{%
2733   \pxrr@k@declare@mark@char@a{#1}#2\pxrr@end
2734 }
2735 \def\pxrr@k@declare@mark@char@a#1#2:#3\pxrr@end{%
2736   \pxrr@jchardef\pxrr@tempa\pxrr@jc{#2:#3}%
2737   \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。

```

2738     \pxrr@k@declare@mark{#1}{\pxrr@k@char@UTF{#1}{#3}}%
2739   \else
2740     \pxrr@k@let@mark{#1}\pxrr@tempa
2741   \fi
2742 }

```

`\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。

```

2743 \def\pxrr@k@char@UTF#1#2{%
2744   \pxrr@decide@avail@UTF
2745   \ifpxrr@avail@UTF
2746     \pxrr@k@declare@mark{#1}{\UTF{#2}}%
2747     \UTF{#2}%
2748   \else
2749     \pxrr@k@let@mark{#1}\@empty
2750     \pxrr@warn@na@kmark{#1}%
2751   \fi
2752 }

```

標準サポートの圏点マークの定義。

```

2753 \pxrr@k@declare@mark@char{bullet} {2121:2022}
2754 \pxrr@k@declare@mark@char{triangle}{2225:25B2}
2755 \pxrr@k@declare@mark@char{Triangle}{2224:25B3}
2756 \pxrr@k@declare@mark@char{fisheye} {2121:25C9}
2757 \pxrr@k@declare@mark@char{Circle} {217B:25CB}

```

```

2758 \pxrr@k@declare@mark@char{bullseye}{217D:25CE}
2759 \pxrr@k@declare@mark@char{circle} {217C:25CF}
2760 \pxrr@k@declare@mark@char{Bullet} {2121:25E6}
2761 \pxrr@k@declare@mark@char{sesame} {2121:FE45}
2762 \pxrr@k@declare@mark@char{Sesame} {2121:FE46}
2763 \pxrr@jchardef\pxrr@ja@dot=\pxrr@jc{2126:30FB}
2764 \pxrr@jchardef\pxrr@ja@comma=\pxrr@jc{2122:3001}
2765 \pxrr@k@declare@mark{bullet*}{%
2766   \pxrr@dima=\pxrr@ruby@zw\relax
2767   \hb@xt@\pxrr@dima{%
2768     \kern-.5\pxrr@dima
2769     \pxrr@if@in@tate{ }\lower.38\pxrr@dima}%
2770     \hb@xt@2\pxrr@dima{%
2771       \pxrr@dima=\f@size\p@
2772       \fontsize{2\pxrr@dima}{\z@}\selectfont
2773       \hss
2774       \pxrr@ja@dot
2775       \hss
2776     }%
2777     \hss
2778   }%
2779 }
2780 \pxrr@k@declare@mark{sesame*}{%
2781   \pxrr@dima=\pxrr@ruby@zw\relax
2782   \hb@xt@\pxrr@dima{%
2783     \pxrr@if@in@tate{\kern.1\pxrr@dima}{\kern.05\pxrr@dima}%
2784     \pxrr@if@in@tate{\lower.85\pxrr@dima}{\raise.3\pxrr@dima}%
2785     \hbox{%
2786       \pxrr@dima=\f@size\p@
2787       \fontsize{2.4\pxrr@dima}{\z@}\selectfont
2788       \pxrr@ja@comma
2789     }%
2790     \hss
2791   }%
2792 }

```

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。

```

2793 \def\pxrr@k@parse@option#1{%
2794   \edef\pxrr@tempa{#1}%
2795   \let\pxrr@side\pxrr@k@d@side
2796   \let\pxrr@k@mark\pxrr@k@d@mark
2797   \let\pxrr@k@full\pxrr@k@d@full
2798   \expandafter\pxrr@k@parse@option@loop\pxrr@tempa @\pxrr@end
2799 }
2800 \def\pxrr@k@parse@option@loop#1{%

```

圈点オプションの解析器は“有限状態”を持たないので非常に単純である。

```
2801 \pxrr@letcs\pxrr@tempa{pxrr@k@po@PR@#1}%
2802 \pxrr@cond\ifx\pxrr@tempa\relax\fi{%
2803   \pxrr@fatal@knx@letter{#1}%
2804   \pxrr@k@parse@option@exit
2805 }{%
2806   \pxrr@tempa
2807   \pxrr@k@parse@option@loop
2808 }%
2809 }
2810 \def\pxrr@k@parse@option@exit#1\pxrr@end{%
2811   \ifpxrr@in@setup\else
2812     \pxrr@k@check@option
```

ここで `\pxrr@k@the@mark` を適切に定義する。

```
2813   \pxrr@if@in@tate{%
2814     \ifcase\pxrr@k@mark \let\pxrr@k@the@mark\pxrr@k@tmark
2815     \or \let\pxrr@k@the@mark\pxrr@k@tsmark
2816     \fi
2817   }{%
2818     \ifcase\pxrr@k@mark \let\pxrr@k@the@mark\pxrr@k@ymark
2819     \or \let\pxrr@k@the@mark\pxrr@k@ysmark
2820     \fi
2821   }%
2822 \fi
2823 }
2824 \def\pxrr@k@po@PR@{%
2825   \pxrr@k@parse@option@exit
2826 }
2827 \def\pxrr@k@po@PR@P{%
2828   \chardef\pxrr@side\z@
2829 }
2830 \def\pxrr@k@po@PR@S{%
2831   \chardef\pxrr@side\@ne
2832 }
2833 \def\pxrr@k@po@PR@p{%
2834   \chardef\pxrr@k@mark\z@
2835 }
2836 \def\pxrr@k@po@PR@s{%
2837   \chardef\pxrr@k@mark\@ne
2838 }
2839 \def\pxrr@k@po@PR@F{%
2840   \chardef\pxrr@k@full\z@
2841 }
2842 \def\pxrr@k@po@PR@f{%
2843   \chardef\pxrr@k@full\@ne
2844 }
```

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。

```
2845 \def\pxrr@k@check@option{%
2846 }
```

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。

```
2847 \def\pxrr@k@compose@block#1#2{%
2848   \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。

```
2849   \pxrr@k@make@rep@list{\copy\pxrr@boxb}{#2}%
2850   \let\pxrr@natwd\pxrr@zeropt
2851   \pxrr@evenspace@int\pxrr@locate@inner\pxrr@boxr
2852     \relax{\wd\pxrr@boxa}%
2853   \setbox\z@\hbox{%
2854     \ifnum\pxrr@side=\z@
2855       \raise\pxrr@ruby@raise\box\pxrr@boxr
2856     \else
2857       \lower\pxrr@ruby@lower\box\pxrr@boxr
2858     \fi
2859   }%
2860   \ht\z@\z@ \dp\z@\z@
2861   \@tempdima\wd\z@
2862   \setbox\pxrr@boxr\hbox{%
2863     \box\z@
2864     \kern-\@tempdima
2865     \box\pxrr@boxa
2866   }%
2867 }
```

`\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。

```
2868 \def\pxrr@k@make@rep@list#1#2{%
2869   \def\pxrr@res{\pxrr@pre{#1}}%
2870   \pxrr@cntr=#2\relax
2871   \ifnum\pxrr@cntr>\@ne
```

```

2872 \@tempcnta\pxrr@cntr \advance\@tempcnta\m@ne
2873 \@whilenum{\@tempcnta>\z}\do{%
2874 \pxrr@appto\pxrr@res{\pxrr@inter{#1}}%
2875 \advance\@tempcnta\m@ne
2876 }%
2877 \fi
2878 \pxrr@appto\pxrr@res{\pxrr@post}%
2879 }

```

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@kenten@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。
2880 \newif\ifpxrr@k@first@entry

```

```

\ifpxrr@k@last@entry  末尾の項目であるか。
2881 \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。
2882 \newif\ifpxrr@k@prev@is@block

```

```

\pxrr@k@accum@res  累積の直接出力。
2883 \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。
2884 \mathchardef\pxrr@k@prebreakpenalty\z@

```

```

\pxrr@k@postbreakpenalty  圏点項目の後禁則ペナルティ。
2885 \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。
2886 \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。

```
2887 \def\pxrr@k@list@pre{%
2888   \pxrr@k@first@entrytrue
2889   \pxrr@k@last@entryfalse
2890   \pxrr@k@prev@is@blockfalse
2891   \let\pxrr@k@accum@res\@empty
2892   \chardef\pxrr@k@block@seq@state\z@
2893 }
```

`\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。

```
2894 \def\pxrr@k@entry@with#1#2{%
2895   \pxrr@if@last{%
2896     \pxrr@k@last@entrytrue
2897     \pxrr@k@entry@with@a#1{#2}%
2898   }{%
2899     \pxrr@k@entry@with@a#1{#2}%
2900   }%
2901 }
2902 \def\pxrr@k@entry@with@a#1#2{%
2903   \mathchardef\pxrr@k@prebreakpenalty\z@
2904   \mathchardef\pxrr@k@postbreakpenalty\z@
```

下請けマクロを実行して結果を得る。

```
2905   #1{#2}%
2906   %\typeout{%
2907   %first=\meaning\ifpxrr@k@first@entry^^J%
2908   %last=\meaning\ifpxrr@k@last@entry^^J%
2909   %prev=\meaning\ifpxrr@k@prev@is@block^^J%
2910   %res=\meaning\pxrr@res^^J%
2911   %type=\meaning\pxrr@k@entry@res@type^^J%
2912   %prepen=\the\pxrr@k@prebreakpenalty^^J%
2913   %postpen=\the\pxrr@k@postbreakpenalty}%
```

累積直接出力の処理。

```
2914   \ifnum\pxrr@k@entry@res@type=\z@
2915     \expandafter\pxrr@appto\expandafter\pxrr@k@accum@res
2916     \expandafter{\pxrr@res}%
2917   \else
2918     \pxrr@k@accum@res
2919     \let\pxrr@k@accum@res\@empty
2920   \fi
```

前禁則ペナルティを入れる。

```
2921   \ifnum\pxrr@k@prebreakpenalty>\z@
2922     \@tempcntb\lastpenalty \unpenalty
2923     \advance\@tempcntb\pxrr@k@prebreakpenalty
2924     \penalty\@tempcntb
2925   \fi
```

圏点ブロックが連続する場合は和文間空白を入れる。

```
2926 \ifnum\pxrr@k@entry@res@type=\tw@
2927   \ifpxrr@k@prev@is@block
2928     \pxrr@inter@mono
2929   \fi
2930   \pxrr@k@prev@is@blocktrue
2931 \else
2932   \pxrr@k@prev@is@blockfalse
2933 \fi
```

ボックスの結果を実際に出力する。

```
2934 \ifnum\pxrr@k@entry@res@type>\z@
2935   \unhbox\pxrr@boxr
2936 \fi
```

後禁則ペナルティを入れる。

```
2937 \ifnum\pxrr@k@postbreakpenalty>\z@
2938   \penalty\pxrr@k@postbreakpenalty
2939 \fi
```

次の項目に進む。

```
2940 \pxrr@k@first@entryfalse
2941 }
```

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

```
2942 \def\pxrr@k@list@post{%
2943   \pxrr@k@accum@res
2944   \let\pxrr@k@accum@res\@empty
2945 }
```

`\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。

```
2946 \def\pxrr@k@kenten@entry{%
2947   \pxrr@k@entry@with\pxrr@k@kenten@entry@
2948 }
2949 \def\pxrr@k@kenten@entry@#1{%
2950   \pxrr@k@check@char{#1}%
2951   \ifpxrr@ok
2952     \pxrr@k@compose@block{#1}\@ne
2953     \chardef\pxrr@k@entry@res@type=\tw@
2954   \else
2955     \def\pxrr@res{#1}%
2956     \chardef\pxrr@k@entry@res@type=\z@
2957   \fi
2958 }
```

`\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。

```
2959 \def\pxrr@k@kenten@entry@kspan{%
2960   \pxrr@k@entry@with\pxrr@k@kenten@entry@kspan@
```

```

2961 }
2962 \def\pxrr@kenten@entry@kspan@#1{%
2963   \pxrr@kenten@entry@kspan@a#1%
2964 }
2965 \def\pxrr@kenten@entry@kspan@a#1{%
    \kspan (= #1) が * 付かを調べる。
2966   \@ifstar{%
2967     \@testopt\pxrr@kenten@entry@kspan@c{%}
2968   }{%
2969     \@testopt\pxrr@kenten@entry@kspan@b{%}
2970   }%
2971 }
2972 \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。
2973   \setbox\z@\pxrr@hbox{#2}%
2974   \@tempdima\pxrr@body@zw\relax
2975   \@tempdimb\wd\z@ \advance\@tempdimb.25\@tempdima
2976   \divide\@tempdimb\@tempdima
2977   \edef\pxrr@kenten@entry@tempa{\number\@tempdimb}%
2978   \pxrr@k@compose@block{#2}\pxrr@kenten@entry@tempa
2979   \chardef\pxrr@k@entry@res@type=\tw@
2980 }
2981 \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。
2982   \def\pxrr@res{#2}%
2983   \chardef\pxrr@k@entry@res@type=\z@
2984 }

```

`\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。

```

2985 \def\pxrr@kenten@entry@kenten{%
2986   \pxrr@k@entry@with\pxrr@kenten@entry@kenten@
2987 }
2988 \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。
2989   \setbox\pxrr@boxr\hbox{#1}%
2990   \chardef\pxrr@k@entry@res@type=\@ne
2991 }

```

`\pxrr@kenten@entry@ruby` ルビ命令の圈点項目。

```

2992 \def\pxrr@kenten@entry@ruby{%
2993   \pxrr@k@entry@with\pxrr@kenten@entry@ruby@
2994 }
2995 \def\pxrr@kenten@entry@ruby@#1{%
2996   \pxrr@apply@combotrue
2997   \setbox\pxrr@boxr\hbox{#1}%

```

```
2998 \chardef\pxrr@k@entry@res@type=\@ne
2999 }
```

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。

```
3000 \newcommand*\kspan{%
3001   \@ifstar{%
3002     \@testopt\pxrr@kspan@a{}%
3003   }{%
3004     \@testopt\pxrr@kspan@a{}%
3005   }%
3006 }
3007 \pxrr@add@protect\kspan
3008 \def\pxrr@kspan@a[#1]#2{%
3009   \begingroup
3010     #2%
3011   \endgroup
3012 }
```

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）。

```
3013 \def\pxrr@k@check@char#1{%
3014   \futurelet\pxrr@token\pxrr@k@check@char@a#1\pxrr@end
3015 }
3016 \def\pxrr@k@check@char@a#1\pxrr@end{%
3017   \pxrr@cond\ifx\pxrr@token\bgroup\fi%
```

グループには圏点を付ける。

```
3018   \pxrr@oktrue
3019 }{\pxrr@cond\ifx\pxrr@token\@sptoken\fi%
```

欧文空白には圏点を付けない。

```
3020   \pxrr@okfalse
3021 }{%
3022   \pxrr@check@char\pxrr@token
3023   \ifcase\pxrr@cntr
```

通常文字でないので圏点を付けない。

```
3024   \pxrr@okfalse
3025   \or
```

欧文の通常文字。圏点を付ける。

```
3026 \pxrr@oktrue
3027 \chardef\pxrr@check@char@temp\z@
3028 \or
```

和文の通常文字。圏点を付ける。

```
3029 \pxrr@oktrue
3030 \chardef\pxrr@check@char@temp\@ne
3031 \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。

```
3032 \ifnum\pxrr@k@full=\z@\ifpxrr@ok
3033 \pxrr@check@punct@char{'#1}\pxrr@check@char@temp
3034 \ifpxrr@ok \pxrr@okfalse
3035 \else \pxrr@oktrue
3036 \fi
3037 \fi\fi
3038 \ifpxrr@ok
3039 \pxrr@get@prebreakpenalty\@tempcnta{'#1}%
3040 \mathchardef\pxrr@k@prebreakpenalty\@tempcnta
3041 \pxrr@get@postbreakpenalty\@tempcnta{'#1}%
3042 \mathchardef\pxrr@k@postbreakpenalty\@tempcnta
3043 \fi
3044 }}%
3045 }
```

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。

```
3046 \AtBeginDocument{%
3047 \providecommand*\kenten{\jkenten}%
3048 }
3049 \newcommand*\jkenten{%
3050 \pxrr@k@prologue
3051 \pxrr@kenten
3052 }
3053 \pxrr@add@protect\jkenten
```

`\pxrr@kenten` オプションの処理を行う。

```
3054 \def\pxrr@kenten{%
3055 \@testopt\pxrr@kenten@a{%}
3056 }
3057 \def\pxrr@kenten@a[#1]{%
3058 \def\pxrr@option{#1}%
3059 \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。

```
3060 \expandafter\@firstofone
3061 \else
3062 \expandafter\pxrr@kenten@proc
3063 \fi
3064 }
```

\pxrr@k@bind@param “呼出時変数” へのコピーを行う。

```
3065 \def\pxrr@k@bind@param{%
3066 \let\pxrr@c@ruby@font\pxrr@k@ruby@font
3067 \let\pxrr@c@size@ratio\pxrr@k@size@ratio
3068 \let\pxrr@c@inter@gap\pxrr@k@inter@gap
3069 }
```

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

```
3070 \def\pxrr@kenten@proc#1{%
3071 \pxrr@prepare@fallback{#1}%
3072 \pxrr@k@bind@param
3073 \pxrr@assign@fsize
3074 \pxrr@k@parse@option\pxrr@option
3075 \pxrr@if@alive{%
3076 \pxrr@k@decompose{#1}%
3077 \let\pxrr@body@list\pxrr@res
3078 \pxrr@kenten@main
3079 }%
3080 \pxrr@kenten@exit
3081 }
```

5.11.2 組版処理

\pxrr@kenten@main 圏点の組版処理。

```
3082 \def\pxrr@kenten@main{%
3083 \setbox\pxrr@boxb\pxrr@hbox@to\z@f%
3084 \pxrr@use@ruby@font
3085 \hss\pxrr@k@the@mark\hss
3086 }%
3087 \let\pxrr@entry\pxrr@kenten@entry
3088 \let\pxrr@entry@kspan\pxrr@kenten@entry@kspan
3089 \let\pxrr@entry@ruby\pxrr@kenten@entry@ruby
3090 \let\pxrr@entry@kenten\pxrr@kenten@entry@kenten
3091 \let\pxrr@post\pxrr@k@list@post
3092 \pxrr@k@list@pre
3093 \pxrr@body@list
3094 }
```

5.11.3 前処理

`\pxrr@jprologue` 圏点用の開始処理。

```
3095 \def\pxrr@k@prologue{%
3096   \ifpxrr@k@ghost
3097     \pxrr@jghost@char
3098     \pxrr@inhibitglue
3099   \fi
3100   \begingroup
3101   \ifpxrr@k@ghost
3102     \setbox\pxrr@boxa\hbox{\pxrr@jghost@char}%
3103     \kern-\wd\pxrr@boxa
3104   \fi
3105 }
```

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。

```
3106 \def\pxrr@kenten@exit{%
3107   \ifpxrr@fatal@error
3108     \pxrr@fallback
3109   \fi
3110   \pxrr@k@epilogue
3111 }
```

`\pxrr@jepilogue` 終了処理。

```
3112 \def\pxrr@k@epilogue{%
3113   \ifpxrr@k@ghost
3114     \setbox\pxrr@boxa\hbox{\pxrr@jghost@char}%
3115     \kern-\wd\pxrr@boxa
3116   \fi
3117   \endgroup
3118   \ifpxrr@k@ghost
3119     \pxrr@inhibitglue
3120     \pxrr@jghost@char
3121   \fi
3122 }
```

5.12 デバッグ用出力

```
3123 \def\pxrr@debug@show@kenten@input{%
3124   \typeout{%
3125     pxrr@k@the@mark=\meaning\pxrr@k@the@mark^^J%
3126     pxrr@side=\meaning\pxrr@side^^J%
3127     pxrr@body@list=\meaning\pxrr@body@list^^J%
3128   }%
3129 }
```

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。
3130 \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。
3131 \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 3132 \let\pxrr@ck@ruby@font\relax
\pxrr@ck@inter@gap 3133 \let\pxrr@ck@size@ratio\relax
3134 \let\pxrr@ck@inter@gap\relax
\pxrr@ck@ruby@inter@gap 3135 \let\pxrr@ck@ruby@inter@gap\relax
\pxrr@ck@side 3136 \let\pxrr@ck@side\relax
\pxrr@ck@the@mark 3137 \let\pxrr@ck@the@mark\relax
3138 \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。
3139 \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。
3140 \newif\ifpxrr@ck@kenten@end

\pxrr@ck@bind@param “呼出時変数” へのコピーを行う。
3141 \def\pxrr@ck@bind@param{%
3142 \let\pxrr@ck@ruby@font\pxrr@c@ruby@font
3143 \let\pxrr@ck@size@ratio\pxrr@c@size@ratio
3144 \let\pxrr@ck@inter@gap\pxrr@c@inter@gap
3145 \let\pxrr@ck@ruby@inter@gap\pxrr@k@ruby@inter@gap
3146 \let\pxrr@ck@side\pxrr@side
3147 \let\pxrr@ck@the@mark\pxrr@k@the@mark
3148 \let\pxrr@ck@ruby@combo\pxrr@k@ruby@combo
3149 \pxrr@csletcs{ifpxrr@ck@kenten@head}{ifpxrr@k@first@entry}%
3150 \pxrr@csletcs{ifpxrr@ck@kenten@end}{ifpxrr@k@last@entry}%
3151 }
```

6.2 その他の変数

```
\pxrr@ck@zw 圏点の全角幅。
3152 \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。
3153 \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。

```
3154 \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。

```
3155 \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。

```
3156 \def\pxrr@ck@check@option{%
3157   \ifpxrr@ck@kenten@head
3158     \let\pxrr@bintr@\@empty
3159     \let\pxrr@bscomp=. \relax
3160     \pxrr@bnobrtrue
3161   \fi
3162 \ifpxrr@ck@kenten@end
3163   \let\pxrr@aintr@\@empty
3164   \let\pxrr@ascomp=. \relax
3165   \pxrr@anobrtrue
3166 \fi
3167 }
```

6.4 フォントサイズ

`\pxrr@ck@assign@fsize` フォントに関連する設定。

```
3168 \def\pxrr@ck@assign@fsize{%
  \pxrr@ck@zw の値を求める。
3169   \begingroup
3170     \@tempdima=\f@size\p@
3171     \@tempdima\pxrr@ck@size@ratio\@tempdima
3172     \edef\pxrr@ruby@fsize{\the\@tempdima}%
3173     \let\pxrr@c@ruby@font\pxrr@ck@ruby@font
3174     \pxrr@use@ruby@font
3175     \pxrr@get@zwidth\pxrr@ck@zw
3176     \global\let\pxrr@gtempa\pxrr@ck@zw
3177   \endgroup
3178   \let\pxrr@ck@zw\pxrr@gtempa

  \pxrr@ck@raise@P、\pxrr@ck@raise@S の値を計算する。
3179   \ifcase\pxrr@ck@side
    圏点側が P の場合。
3180     \@tempdimc\pxrr@ck@zw
3181     \advance\@tempdimc-\pxrr@htratio\@tempdimc
3182     \@tempdima\pxrr@ruby@raise\relax
3183     \@tempdimb\pxrr@ruby@zw\relax
```

```

3184 \advance\@tempdima\pxrr@htratio\@tempdimb
3185 \@tempdimb\pxrr@body@zw\relax
3186 \advance\@tempdima\pxrr@ck@ruby@inter@gap\@tempdimb
3187 \advance\@tempdima\@tempdimc
3188 \edef\pxrr@ck@raise@P{\the\@tempdima}%
3189 \@tempdima\pxrr@body@zw\relax
3190 \@tempdima\pxrr@htratio\@tempdima
3191 \@tempdimb\pxrr@body@zw\relax
3192 \advance\@tempdima\pxrr@ck@inter@gap\@tempdimb
3193 \advance\@tempdima\@tempdimc
3194 \edef\pxrr@ck@raise@S{\the\@tempdima}%
3195 \let\pxrr@ck@raise@t\pxrr@ck@raise@P
3196 \or

```

圏点側が S の場合。

```

3197 \@tempdimc\pxrr@ck@zw
3198 \@tempdimc\pxrr@htratio\@tempdimc
3199 \@tempdima-\pxrr@ruby@lower\relax
3200 \@tempdimb\pxrr@ruby@zw\relax
3201 \advance\@tempdimb-\pxrr@htratio\@tempdimb
3202 \advance\@tempdima-\@tempdimb
3203 \@tempdimb\pxrr@body@zw\relax
3204 \advance\@tempdima-\pxrr@ck@ruby@inter@gap\@tempdimb
3205 \advance\@tempdima-\@tempdimc
3206 \edef\pxrr@ck@raise@S{\the\@tempdima}%
3207 \@tempdima-\pxrr@body@zw\relax
3208 \advance\@tempdima-\pxrr@htratio\@tempdima
3209 \@tempdimb\pxrr@body@zw\relax
3210 \advance\@tempdima-\pxrr@ck@inter@gap\@tempdimb
3211 \advance\@tempdima-\@tempdimc
3212 \edef\pxrr@ck@raise@P{\the\@tempdima}%
3213 \let\pxrr@ck@raise@t\pxrr@ck@raise@S
3214 \fi
3215 }

```

6.5 ブロック毎の組版

`\pxrr@ck@body@natwd` 親文字列の自然長。

```
3216 \let\pxrr@ck@body@natwd\relax
```

`\pxrr@ck@locate` 圏点列のパターン指定。

```
3217 \let\pxrr@ck@locate\relax
```

`\pxrr@ck@kenten@list` 圏点列のリスト。

```
3218 \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。

```
3219 \def\pxrr@ck@compose#1{%
```

圏点を組んだボックスを作る。

```
3220 \setbox\tw@\pxrr@hbox@to\z@{%
3221 \@tempdima=\f@size\p@
3222 \@tempdima\pxrr@ck@size@ratio\@tempdima
3223 \edef\pxrr@ruby@fsize{\the\@tempdima}%
3224 \let\pxrr@c@ruby@font\pxrr@ck@ruby@font
3225 \pxrr@use@ruby@font
3226 \hss\pxrr@ck@the@mark\hss
3227 }%
```

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。

```
3228 \pxrr@save@listproc
3229 \pxrr@decompose{#1}%
3230 \def\pxrr@pre{%
3231 \let\pxrr@res\@empty
3232 \pxrr@ck@compose@entry\pxrr@pre
3233 }%
3234 \def\pxrr@inter{%
3235 \pxrr@ck@compose@entry\pxrr@inter
3236 }%
3237 \def\pxrr@post{%
3238 \pxrr@appto\pxrr@res{\pxrr@post}%
3239 }%
3240 \pxrr@res
3241 \pxrr@restore@listproc
3242 \let\pxrr@natwd\pxrr@ck@body@natwd
```

圏点リストを均等配置する。

```
3243 \pxrr@evenspace@int\pxrr@ck@locate\pxrr@boxb\relax
3244 {\wd\pxrr@boxa}%
```

合成処理。

```
3245 \setbox\z@\hbox{%
3246 \unhcopy\z@
3247 \kern-\wd\z@
3248 \ifcase\pxrr@side
3249 \raise\pxrr@ck@raise@P
3250 \or
3251 \raise\pxrr@ck@raise@S
3252 \or
3253 \raise\pxrr@ck@raise@t
3254 \fi
3255 \hb@xt@\wd\pxrr@boxa{\hss\copy\pxrr@boxb\hss}%
3256 }%
3257 }
3258 \def\pxrr@ck@compose@entry#1#2{%
3259 \setbox\pxrr@boxb\pxrr@hbox{#2}%
```

```

3260 \edef\pxrr@tempa{%
3261     \noexpand\pxrr@appto\noexpand\pxrr@res{\noexpand#1{%
3262         \hb@xt@\the\wd\pxrr@boxb{\hss\copy\tw@\hss}}}%
3263     }\pxrr@tempa
3264 }

```

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。

```
3265 \def\pxrr@dumb@sub#1#2#3{#1}
```

`\pxrr@dumb@ruby` 無難なルビ命令。

```

3266 \def\pxrr@dumb@ruby{%
3267     \pxrr@dumb@sub\pxrr@dumb@ruby@
3268 }
3269 \def\pxrr@dumb@ruby@#1#2#3{#1}

```

`\pxrr@dumb@truby` 無難な両側ルビ命令。

```

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

```

3274 \def\pxrr@dumb@kenten{%
3275     \pxrr@dumb@sub\pxrr@dumb@kenten@
3276 }
3277 \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。

```

3278 \providecommand*\pdfstringdefPreHook{}
3279 \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。

```

3280 \ifx\pxrr@cmd@ruby\ruby
3281     \let\ruby\pxrr@dumb@ruby
3282 \fi
3283 \let\jruby\pxrr@dumb@ruby
3284 \let\aruby\pxrr@dumb@ruby
3285 \let\truby\pxrr@dumb@truby
3286 \let\atruby\pxrr@dumb@truby
3287 \ifx\pxrr@cmd@kenten\kenten
3288     \let\kenten\pxrr@dumb@kenten
3289 \fi

```

```
3290 \let\kspan\pxrr@dumb@kenten  
3291 }
```