

# pxrubrica パッケージ

八登 崇之 (Takayuki YATO; aka “ZR”)

v1.2 [2017/04/20]

## 目次

1	パッケージ読込	1
2	基本機能	1
2.1	用語集	1
2.2	ルビ用命令	1
2.3	入力文字列のグループの指定	4
2.4	ゴースト処理	4
2.5	パラメタ設定命令	5
3	将来の拡張機能(未実装)	6
3.1	拡張機能設定の命令	7
4	実装	7
4.1	前提パッケージ	7
4.2	エラーメッセージ	7
4.3	パラメタ	10
4.3.1	全般設定	10
4.3.2	ルビ呼出時の設定	12
4.4	変数	13
4.5	補助手続	14
4.5.1	雑多な定義	14
4.5.2	数値計算	16
4.5.3	リスト分解	18
4.6	エンジン依存処理	23
4.7	パラメタ設定公開命令	30
4.8	ルビオプション解析	32
4.9	オプション整合性検査	38
4.10	フォントサイズ	40
4.11	ルビ用均等割り	42

4.12	小書き仮名の変換 . . . . .	45
4.13	ブロック毎の組版 . . . . .	46
4.14	命令の頑強化 . . . . .	53
4.15	致命的エラー対策 . . . . .	53
4.16	先読み処理 . . . . .	54
4.17	進入処理 . . . . .	56
4.17.1	前側進入処理 . . . . .	57
4.17.2	後側進入処理 . . . . .	58
4.18	メインです . . . . .	59
4.18.1	エントリーポイント . . . . .	59
4.18.2	入力検査 . . . . .	64
4.18.3	ルビ組版処理 . . . . .	66
4.18.4	前処理 . . . . .	71
4.18.5	後処理 . . . . .	73
4.19	デバッグ用出力 . . . . .	73

## 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 基本機能

### 2.1 用語集

本パッケージで独自の意味をもつ単語を挙げる。

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

次の用語については、『日本語組版の要件』に従う。

ルビ、親文字、中付き、肩付き、モノルビ、グループルビ、熟語ルビ

## 2.2 ルビ用命令

- `\ruby[〈オプション〉]{〈親文字〉}{〈ルビ文字〉}`

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

ここで、〈オプション〉は以下の形式をもつ。

〈前進入設定〉〈前補助設定〉〈モード〉〈後補助設定〉〈後進入設定〉

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

〈前進入設定〉は以下の値の何れか。

|| 前突出禁止            < 前進入大  
| 前進入無し            ( 前進入小

〈前補助設定〉は以下の値の何れか。

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

– 空白挿入量の既定値は和文間空白である。

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

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

– ゴースト処理が有効の場合はこの設定は無視される。

〈モード〉は以下の値の何れか。

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

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

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

– e 指定時は、ルビが長い場合に親文字列をルビの長さに合わせて均等割りで配置する。E 指定時は、空きを入れずに中央揃えて配置する。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 `m` および `g`、複数ある場合は `g` と等価になる。

(後補助設定) は以下の値の何れか。

- : 和欧文間空白挿入      \* 行分割禁止
- . 空白挿入なし          ! 段落末で進入許可
- 空白挿入量の既定値は和文間空白である。
- \* 無指定の場合の行分割の可否は `pdfTeX` の標準の動作に従うのが原則だが、直後にあるものが文字でない場合、正しく動作しない (禁則が破れる) 可能性がある。従って、不適切な行分割が起こりうる場合は適宜 \* を指定する必要がある (なお、段落末尾で \* を指定してはならない)。
- ! 無指定の場合、段落末尾では進入が抑止される。
- ゴースト処理が有効の場合はこの設定は無視される。

(後進入設定) は以下の値。

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

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

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

## 2.3 入力文字列のグループの指定

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

例えば、ルビ文字列

```
じゆく|ご
```

は 2 つのグループからなり、最初のは 3 文字、後のものは 1 文字からなる。

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

```
ベクタ{\< (-) \>}
```

は 1 つのグループからなり、それは 4 つの《文字》からなる。

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

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

```
\ruby[m]{熟語}{じゆく|ご}
```

は、「熟 + じゆく」「語 + ご」の 2 つのブロックからなる。

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

```
\ruby[g]{五月雨}{さみだれ}
```

は、「五月雨 + さみだれ」の 1 つのブロックからなる。

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

## 2.4 ゴースト処理

「和文ゴースト処理」とは以下のようなものである：

和文ルビの親文字列出力の前後に全角空白文字を挿入する (ただしその空きを打ち消すように負の空きを同時に入れる) ことで、親文字列全体が、その外側から見たとき

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

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

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。

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

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

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

## 2.5 パラメタ設定命令

基本的設定。

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

`\rubyfontsetup{\mcfamily}`

- `\rubybigintrusion{⟨実数⟩}`  
「大」の進入量（ルビ全角単位）。[既定 = 1]
- `\rubysmallintrusion{⟨実数⟩}`  
「小」の進入量（ルビ全角単位）。[既定 = 0.5]
- `\rubymaxmargin{⟨実数⟩}`  
ルビ文字列の方が短い場合の、ルビ文字列の端の親文字列の端からの距離の上限値（親文字全角単位）。[既定 = 0.75]
- `\rubyintergap{⟨実数⟩}`  
ルビと親文字の間の空き（親文字全角単位）。[既定 = 0]
- `\rubyusejghost` / `\rubynousejghost`  
和文ゴースト処理を行う / 行わない。[既定 = 行わない]
- `\rubyuseaghost` / `\rubynouseaghost`  
欧文ゴースト処理を行う / 行わない。[既定 = 行わない]

詳細設定。通常はこれらの既定値を変える必要はないだろう。

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

---

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

### 3 将来の拡張機能(未実装)

(この節では、まだ実装されていないが、実現できればよいと考えている機能について述べる。)

「行分割の有無により親文字とルビ文字の相対位置が変化する」ような処理は、 $\text{T}_{\text{E}}\text{X}$  での実現は非常に難しい。これを  $\epsilon\text{-pT}_{\text{E}}\text{X}$  の拡張機能を用いて何とか実現したい。

- 可動グループルビ機能：例えば、  
`\ruby[g]{我思う|故に|我有り}{コギト・|エルゴ・|スム}`  
のようにグループルビで複数グループを指定すると、通常は「我思う故に我有り + コギト・エルゴ・スム」の 1 ブロックになるが、グループの区切りで行分割可能となり、例えば最初のグループの後で行分割された場合は、自動的に「我思う + コギト・」と「故に我有り + エルゴ・スム」の 2 ブロックでの組版に変化する。
- 行頭・行末での突出の自動補正：行頭（行末）に配置されたルビ付き文字列では、自動的に前（後）突出を禁止する。
- 熟語ルビの途中での行分割の許可：例えば、  
`\ruby[j]{熟語}{じゆく|ご}`  
の場合、結果はグループルビ処理の「熟語 + じゆくご」となるが、途中での行分割が可能で、その場合、「熟 + じゆく」「語 + ご」の 2 ブロックで出力される。

#### 3.1 拡張機能設定の命令

- `\rubyuseextra{<整数>}`  
拡張機能の実装方法。[既定 = 0]
  - 0：拡張機能は無効にする。
  - 1：まだよくわからないなにか（未実装）。
- `\rubyadjustatlineedge`/`\rubynoadjustatlineedge`  
行頭・行末での突出の自動補正を行う／行わない。[既定 = 行わない]
- `\rubybreakjukugo`/`\rubynobreakjukugo`  
モノルビ処理にならない熟語ルビで中間の行分割を許す／許さない。[既定 = 許さない]

## 4 実装

### 4.1 前提パッケージ

keyval を使う予定（まだ使っていない）。

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

## 4.2 エラーメッセージ

`\pxrr@error` エラー出力命令。

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

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

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

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

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

`\pxrr@eh@fatal` 致命的エラーのヘルプ。

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

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

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

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

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

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

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

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

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

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

`\pxrr@warn@load@order` Unicode TeX 用の日本語組版パッケージ（LuaTeX-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）。

```

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。 \rubyssetup 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。 \rubyssetup 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@thead` 肩付き／中付きの設定。 `\rubyssetup` 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@thead=0`

`\pxrr@d@mode` モノルビ (m) ・グループルビ (g) ・熟語ルビ (j) のいずれか。 `\rubyssetup` 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。 `\rubyssetup` 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。 `\rubyssetup` 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。 `\rubyssetup` の `f/F` の設定。 整数定数。  
115 `\chardef\pxrr@d@fullsize=0`

#### 4.3.2 ルビ呼出時の設定

`\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 (`\rubyssetup`) 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 (`\rubyssetup`) 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 (`\rubyssetup`) 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@thead` と同じ。 整数定数。  
126 `\chardef\pxrr@athead=0`

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

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

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

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

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

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

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

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

#### 4.4 変数

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

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

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

`\pxrr@ruby@count` `\pxrr@ruby@list` の長さ。整数値マクロ。  
139 `\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。  
140 `\let\pxrr@sruby@list\@undefined`

`\pxrr@sruby@count` `\pxrr@sruby@list` の長さ。整数値マクロ。  
141 `\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。  
142 `\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。  
143 `\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。  
144 `\let\pxrr@aspace\@undefined`

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

`\pxrr@all@input` 両側ルビの処理で使われる一時変数。  
146 `\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。  
147 `\newif\ifpxrr@ok`

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

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

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

`\pxrr@boxa` 汎用のボックスレジスタ。  
`\pxrr@boxb` 151 `\newbox\pxrr@boxa`  
152 `\newbox\pxrr@boxb`

`\pxrr@boxr` 結果を格納するボックスレジスタ。  
153 `\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。  
154 `\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。  
155 `\chardef\pxrr@zero=0`

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

```

\pxrr@hfilx \pxrr@hfilx{<実数>} : 「<実数>fil」のグルーを置く。
157 \def\pxrr@hfilx#1{%
158 \hskip\z@\@plus #1fil\relax
159 }

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

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

\pxrr@cond \pxrr@cond\ifXXX...\fi{<真>}<偽>} : 一般の TEX 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。
166 \@gobbletwo\if\if \def\pxrr@cond#1\fi{%
167 #1\expandafter\@firstoftwo
168 \else\expandafter\@secondoftwo
169 \fi
170 }

\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。
171 \def\pxrr@cslet#1{%
172 \expandafter\let\csname#1\endcsname
173 }
174 \def\pxrr@letcs#1#2{%
175 \expandafter\let\expandafter#1\csname#2\endcsname
176 }
177 \def\pxrr@csletcs#1#2{%
178 \expandafter\let\csname#1\expandafter\endcsname
179 \csname#2\endcsname
180 }

\pxrr@setok \pxrr@setok{<テスト>} : テストの結果を \ifpxrr@ok に返す。
181 \def\pxrr@setok#1{%
182 #1{\pxrr@oktrue}{\pxrr@okfalse}%
183 }

\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。
184 \def\pxrr@appto#1#2{%
185 \expandafter\def\expandafter#1\expandafter{#1#2}%
186 }

\pxrr@nil ユニークトークン。
\pxrr@end

```

```

187 \def\pxrr@nil{\noexpand\pxrr@nil}
188 \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。

```

189 \def\pxrr@without@macro@trace#1{%
190   \chardef\pxrr@tracingmacros@save=\tracingmacros
191   \tracingmacros\z@
192   #1%
193   \tracingmacros\pxrr@tracingmacros@save
194 }
195 \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 196 \def\pxrr@hbox#1{%
197   \hbox{%
198     \color@begingroup
199     #1%
200     \color@endgroup
201   }%
202 }
203 \def\pxrr@hbox@to#1#2{%
204   \pxrr@hbox@to@a{#1}%
205 }
206 \def\pxrr@hbox@to@a#1#2{%
207   \hbox to#1{%
208     \color@begingroup
209     #2%
210     \color@endgroup
211   }%
212 }

```

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。

```

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

```

#### 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>= $\frac{\text{実数}}{\text{寸法レジスタ}}$  の逆の演算を行う。

```

222 \mathchardef\pxrr@invscale@ca=259
223 \def\pxrr@invscale#1#2{%
224   \begingroup
225     \@tempdima=#1\relax
226     \@tempdimb#2\p@\relax
227     \@tempcnta\@tempdima
228     \multiply\@tempcnta\@ccclvi
229     \divide\@tempcnta\@tempdimb
230     \multiply\@tempcnta\@ccclvi
231     \@tempcntb\p@
232     \divide\@tempcntb\@tempdimb
233     \advance\@tempcnta-\@tempcntb
234     \advance\@tempcnta-\tw@
235     \@tempdimb\@tempcnta\@ne
236     \advance\@tempcnta\@tempcntb
237     \advance\@tempcnta\@tempcntb
238     \advance\@tempcnta\pxrr@invscale@ca
239     \@tempdimc\@tempcnta\@ne
240     \@whiledim\@tempdimb<\@tempdimc\do{%
241       \@tempcntb\@tempdimb
242       \advance\@tempcntb\@tempdimc
243       \advance\@tempcntb\@ne
244       \divide\@tempcntb\tw@
245       \ifdim #2\@tempcntb>\@tempdima
246         \advance\@tempcntb\m@ne
247         \@tempdimc=\@tempcntb\@ne
248       \else
249         \@tempdimb=\@tempcntb\@ne
250       \fi}%
251     \xdef\pxrr@tempa{\the\@tempdimb}%
252   \endgroup
253   #1=\pxrr@tempa\relax
254 }

```

`\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  $\text{iu}$  は ⟨入力単位⟩、 $\text{ou}$  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。

```

255 \def\pxrr@interpolate#1#2#3#4#5{%
256   \edef\pxrr@tempa{#1}%
257   \edef\pxrr@tempb{#2}%
258   \def\pxrr@tempd{#3}%
259   \setlength{\@tempdima}{#4}%

```

```

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

```

#### 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)。

```

287 \def\pxrr@decompose#1{%
288 \let\pxrr@res\@empty
289 \pxrr@cntr=\z@
290 \pxrr@decompose@loopa#1\pxrr@end
291 }
292 \def\pxrr@decompose@loopa{%
293 \futurelet\pxrr@token\pxrr@decompose@loopb
294 }
295 \def\pxrr@decompose@loopb{%
296 \pxrr@ifx{\pxrr@token\pxrr@end}{%

```

```

297 \pxrr@appto\pxrr@res{\pxrr@post}%
298 }{%
299 \pxrr@setok{\pxrr@ifx{\pxrr@token\bgroup}}%
300 \pxrr@decompose@loopc
301 }%
302 }
303 \def\pxrr@decompose@loopc#1{%
304 \ifx\pxrr@res\@empty
305 \def\pxrr@res{\pxrr@pre}%
306 \else
307 \pxrr@appto\pxrr@res{\pxrr@inter}%
308 \fi
309 \ifpxrr@ok
310 \pxrr@appto\pxrr@res{{#1}}%
311 \else
312 \pxrr@appto\pxrr@res{{#1}}%
313 \fi
314 \advance\pxrr@cntr\@ne
315 \pxrr@decompose@loopa
316 }

```

`\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。

```

317 \def\pxrr@decompbar#1{%
318 \let\pxrr@res\@empty
319 \pxrr@cntr=\z@
320 \pxrr@decompbar@loopa\pxrr@nil#1\pxrr@end!%
321 }
322 \def\pxrr@decompbar@loopa#1|{%
323 \expandafter\pxrr@decompbar@loopb\expandafter{\@gobble#1}%
324 }
325 \def\pxrr@decompbar@loopb#1{%
326 \pxrr@decompbar@loopc#1\relax\pxrr@nil{#1}%
327 }
328 \def\pxrr@decompbar@loopc#1#2\pxrr@nil#3{%
329 \pxrr@ifx{#1\pxrr@end}{%
330 \pxrr@appto\pxrr@res{\pxrr@post}%
331 }{%
332 \ifx\pxrr@res\@empty
333 \def\pxrr@res{\pxrr@pre}%
334 \else
335 \pxrr@appto\pxrr@res{\pxrr@inter}%
336 \fi
337 \pxrr@appto\pxrr@res{{#3}}%
338 \advance\pxrr@cntr\@ne
339 \pxrr@decompbar@loopa\pxrr@nil
340 }%

```

341 }

`\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 :

```
\CSa = \pxrr@pre{<X1>}\pxrr@inter{<X2>}... \pxrr@inter{<Xn>}\pxrr@post
\CSb = \pxrr@pre{<Y1>}\pxrr@inter{<Y2>}... \pxrr@inter{<Yn>}\pxrr@post
```

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>}...
\pxrr@inter{<Xn>}{<Yn>}\pxrr@post
```

```
342 \def\pxrr@zip@list#1#2{%
343   \let\pxrr@res\@empty
344   \let\pxrr@post\relax
345   \let\pxrr@tempa#1\pxrr@appto\pxrr@tempa{}}%
346   \let\pxrr@tempb#2\pxrr@appto\pxrr@tempb{}}%
347   \pxrr@zip@list@loopa
348 }
349 \def\pxrr@zip@list@loopa{%
350   \expandafter\pxrr@zip@list@loopb\pxrr@tempa\pxrr@end
351 }
352 \def\pxrr@zip@list@loopb#1#2#3\pxrr@end{%
353   \pxrr@ifx{#1\relax}{%
354     \pxrr@zip@list@exit
355   }{%
356     \pxrr@appto\pxrr@res{#1{#2}}%
357     \def\pxrr@tempa{#3}%
358     \expandafter\pxrr@zip@list@loopc\pxrr@tempb\pxrr@end
359   }%
360 }
361 \def\pxrr@zip@list@loopc#1#2#3\pxrr@end{%
362   \pxrr@ifx{#1\relax}{%
363     \pxrr@interror{zip}%
364     \pxrr@appto\pxrr@res{}}%
365     \pxrr@zip@list@exit
366   }{%
367     \pxrr@appto\pxrr@res{#2}}%
368     \def\pxrr@tempb{#3}%
369     \pxrr@zip@list@loopa
370   }%
371 }
372 \def\pxrr@zip@list@exit{%
373   \pxrr@appto\pxrr@res{\pxrr@post}%
374 }
```

`\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>}... \pxrr@inter{<Xn>}\pxrr@post
\CSb = \pxrr@pre{<Y1>}\pxrr@inter{<Y2>}... \pxrr@inter{<Yn>}\pxrr@post
```

$\backslash\text{Csc} = \backslash\text{pxrr@pre}\{Z1\}\backslash\text{pxrr@inter}\{Z2\}\cdots\backslash\text{pxrr@inter}\{Zn\}\backslash\text{pxrr@post}$

この命令は  $\backslash\text{pxrr@res}$  を以下の内容に定義する。

$\backslash\text{pxrr@pre}\{X1\}\{Y1\}\{Z1\}\backslash\text{pxrr@inter}\{X2\}\{Y2\}\{Z2\}\cdots$   
 $\backslash\text{pxrr@inter}\{Xn\}\{Yn\}\{Zn\}\backslash\text{pxrr@post}$

```

375 \def\pxrr@tzip@list#1#2#3{%
376   \let\pxrr@res\@empty
377   \let\pxrr@post\relax
378   \let\pxrr@tempa#1\pxrr@appto\pxrr@tempa{}}%
379   \let\pxrr@tempb#2\pxrr@appto\pxrr@tempb{}}%
380   \let\pxrr@tempc#3\pxrr@appto\pxrr@tempc{}}%
381   \pxrr@tzip@list@loopa
382 }
383 \def\pxrr@tzip@list@loopa{%
384   \expandafter\pxrr@tzip@list@loopb\pxrr@tempa\pxrr@end
385 }
386 \def\pxrr@tzip@list@loopb#1#2#3\pxrr@end{%
387   \pxrr@ifx{#1\relax}{%
388     \pxrr@tzip@list@exit
389   }{%
390     \pxrr@appto\pxrr@res{#1{#2}}%
391     \def\pxrr@tempa{#3}%
392     \expandafter\pxrr@tzip@list@loopc\pxrr@tempb\pxrr@end
393   }%
394 }
395 \def\pxrr@tzip@list@loopc#1#2#3\pxrr@end{%
396   \pxrr@ifx{#1\relax}{%
397     \pxrr@interror{tzip}%
398     \pxrr@appto\pxrr@res{}}%
399     \pxrr@tzip@list@exit
400   }{%
401     \pxrr@appto\pxrr@res{#2}}%
402     \def\pxrr@tempb{#3}%
403     \expandafter\pxrr@tzip@list@loopd\pxrr@tempc\pxrr@end
404   }%
405 }
406 \def\pxrr@tzip@list@loopd#1#2#3\pxrr@end{%
407   \pxrr@ifx{#1\relax}{%
408     \pxrr@interror{tzip}%
409     \pxrr@appto\pxrr@res{}}%
410     \pxrr@tzip@list@exit
411   }{%
412     \pxrr@appto\pxrr@res{#2}}%
413     \def\pxrr@tempc{#3}%
414     \pxrr@tzip@list@loopa
415   }%
416 }

```

```

417 \def\pxrr@tzip@list@exit{%
418   \pxrr@appto\pxrr@res{\pxrr@post}%
419 }

```

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

$$\backslash\text{CSa} = \backslash\text{pxrr@pre}\{X_1\}\backslash\text{pxrr@inter}\{X_2\}\cdots\backslash\text{pxrr@inter}\{X_n\}\backslash\text{pxrr@post}$$

の時に、`\pxrr@res` を以下の内容に定義する。

$$\langle X_1 \rangle \langle X_2 \rangle \cdots \langle X_n \rangle$$

```

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

```

`\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

$$\backslash\text{CS} = \backslash\text{pxrr@pre}\{X_1\}\backslash\text{pxrr@inter}\{X_2\}\cdots\backslash\text{pxrr@inter}\{X_n\}\backslash\text{pxrr@post}$$

の時に、`\CS` を以下の内容で置き換える。

$$\backslash\text{pxrr@pre}\{X_1\}\langle X_2 \rangle \cdots \langle X_n \rangle \backslash\text{pxrr@post}$$

```

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

```

`\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。

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

```

436 \def\pxrr@zip@single#1#2{%
437   \expandafter\pxrr@zip@single@a\expandafter#1#2\pxrr@end
438 }
439 \def\pxrr@zip@single@a#1{%
440   \expandafter\pxrr@zip@single@b#1\pxrr@end
441 }

```

```

442 \def\pxrr@zip@single@b#1\pxrr@end#2\pxrr@end{%
443   \def\pxrr@res{\pxrr@pre{#1}{#2}\pxrr@post}%
444 }

```

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

$\langle X \rangle$ ;  $\langle Y \rangle$ ;  $\langle Z \rangle$

の時に、`\pxrr@res` を以下の内容に定義する。

`\pxrr@pre{<X>}{<Y>}{<Z>}\pxrr@post`

```

445 \def\pxrr@tzip@single#1#2#3{%
446   \expandafter\pxrr@tzip@single@a\expandafter#1\expandafter#2#3\pxrr@end
447 }
448 \def\pxrr@tzip@single@a#1#2{%
449   \expandafter\pxrr@tzip@single@b\expandafter#1#2\pxrr@end
450 }
451 \def\pxrr@tzip@single@b#1{%
452   \expandafter\pxrr@tzip@single@c#1\pxrr@end
453 }
454 \def\pxrr@tzip@single@c#1\pxrr@end#2\pxrr@end#3\pxrr@end{%
455   \def\pxrr@res{\pxrr@pre{#1}{#2}{#3}\pxrr@post}%
456 }

```

## 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。

```

457 \let\pxrr@x@tempa\@empty
458 \let\pxrr@x@tempb\@empty
459 \let\pxrr@x@tempa\@empty
460 \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。

```

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

```

`\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。

```

468 \pxrr@ifprimitive\kansuji{%
469   \pxrr@csletcs{ifpxrr@in@ptex}{iftrue}%
470 }{%
471   \pxrr@csletcs{ifpxrr@in@ptex}{iffalse}%

```

472 }

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

```
473 \pxrr@ifprimitive\enablecjktoken{%
474   \pxrr@csletcs{ifpxrr@in@uptex}{iftrue}%
475 }{%
476   \pxrr@csletcs{ifpxrr@in@uptex}{iffalse}%
477 }
```

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

```
478 \pxrr@ifprimitive\XeTeXrevision{%
479   \pxrr@csletcs{ifpxrr@in@xetex}{iftrue}%
480 }{%
481   \pxrr@csletcs{ifpxrr@in@xetex}{iffalse}%
482 }
```

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

```
483 \@ifpackageloaded{xeCJK}{%
484   \pxrr@csletcs{ifpxrr@in@xecjk}{iftrue}%
485 }{%
486   \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。

```
487   \AtBeginDocument{%
488     \@ifpackageloaded{xeCJK}{%
489       \pxrr@warn@load@order{xeCJK}%
490     }{}%
491   }%
492 }
```

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

```
493 \pxrr@ifprimitive\luatexrevision{%
494   \pxrr@csletcs{ifpxrr@in@luatex}{iftrue}%
495 }{%
496   \pxrr@csletcs{ifpxrr@in@luatex}{iffalse}%
497 }
```

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

```
498 \@ifpackageloaded{luatexja-core}{%
499   \pxrr@csletcs{ifpxrr@in@luatexja}{iftrue}%
500 }{%
501   \pxrr@csletcs{ifpxrr@in@luatexja}{iffalse}%
502   \AtBeginDocument{%
503     \@ifpackageloaded{luatexja-core}{%
504       \pxrr@warn@load@order{LuaTeX-ja}%
505     }{}%
506   }%
507 }
```

508 `\ifpxrr@in@xetex`

```

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

```

`\ifpxrr@in@unicode` 「和文」内部コードが<sup>3</sup>Unicode であるか。

```

522 \ifpxrr@in@xetex
523   \pxrr@csletcs{ifpxrr@in@unicode}{iftrue}%
524 \else\ifpxrr@in@luatex
525   \pxrr@csletcs{ifpxrr@in@unicode}{iftrue}%
526 \else\ifpxrr@in@uptex
527   \pxrr@csletcs{ifpxrr@in@unicode}{iftrue}%
528 \else
529   \pxrr@csletcs{ifpxrr@in@unicode}{iffalse}%
530 \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。

```

531 \def\pxrr@jc#1{%
532   \pxrr@jc@a#1\pxrr@nil
533 }
534 \ifpxrr@in@unicode
535   \def\pxrr@jc@a#1:#2\pxrr@nil{%
536     "#2\space
537   }
538 \else\ifpxrr@in@ptex
539   \def\pxrr@jc@a#1:#2\pxrr@nil{%
540     \jis"#1\space\space
541   }
542 \else
543   \def\pxrr@jc@a#1:#2\pxrr@nil{%
544     '?\space
545   }
546 \fi\fi

```

`\pxrr@jchardef` 和文用の `\chardef`。

```

547 \ifpxrr@in@uptex
548   \let\pxrr@jchardef\kchardef
549 \else
550   \let\pxrr@jchardef\chardef

```

551 \fi

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

552 \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。

553 \begingroup \catcode'\|=0

554 \gdef\pxrr@if@in@tate{%

555 \pxrr@cond|iftdir|fi

556 }

557 \endgroup

558 \else\ifpxrr@in@luatexja

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

559 \def\pxrr@if@in@tate{%

560 \pxrr@cond\ifnum\ltjgetparameter{direction}=\thr@@\fi

561 }

562 \else

それ以外は常に横組と見なす。

563 \let\pxrr@if@in@tate\@secondoftwo

564 \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。

pTeX 系の場合、`\kansuji` トリックを利用する。

565 \ifpxrr@in@ptex

566 \def\pxrr@get@jchar@token#1#2{%

567 \begingroup

568 \kansujichar\@ne=#2\relax

569 \xdef\pxrr@x@gtempa{\kansuji\@ne}%

570 \endgroup

571 \let#1\pxrr@x@gtempa

572 }

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

573 \else\ifpxrr@in@unicode

574 \def\pxrr@get@jchar@token#1#2{%

575 \begingroup

576 \lccode'\?=#2\relax

577 \lowercase{\xdef\pxrr@x@gtempa{?}}%

578 \endgroup

579 \let#1\pxrr@x@gtempa

580 }

それ以外ではダミー定義。

581 \else

582 \def\pxrr@get@jchar@token#1#2{%

583 \def#1{?}%

```
584 }
585 \fi\fi
```

`\pxrr@x@K` 適当な漢字（実際は <一>）のトークン。

```
586 \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。

```
587 \ifpxrr@in@ptex
588 \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。

```
589 \pxrr@x@swafalse
590 \begingroup
591 \inhibitxspcode\pxrr@x@K\thr@@
592 \kanjiskip\p@
593 \setbox\z@\hbox{\noautospacing\pxrr@x@K\pxrr@x@K}%
594 \setbox\tw@\hbox{\pxrr@x@K\pxrr@x@K}%
595 \ifdim\wd\tw@>\wd\z@
596 \aftergroup\pxrr@x@swatruue
597 \fi
598 \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。

```
599 \edef#1{%
600 \ifpxrr@x@swa \the\kanjiskip
601 \else \pxrr@zeropt
602 \fi
603 }%
604 }
```

LuaTeX-ja 使用の場合。

```
605 \else\ifpxrr@in@luatexja
606 \def\pxrr@get@iiskip#1{%
607 \edef#1{%
608 \ifnum\ltjgetparameter{autospacing}=\@ne
609 \ltjgetparameter{kanjiskip}%
610 \else \pxrr@zeropt
611 \fi
612 }%
613 }
```

それ以外の場合はゼロとする。

```
614 \else
615 \def\pxrr@get@iiskip#1{%
616 \let#1\pxrr@zeropt
617 }
618 \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。

```
619 \ifpxrr@in@ptex
620   \def\pxrr@get@iaiskip#1{%
621     \pxrr@x@swafalse
622     \begingroup
623       \inhibitxspcode\pxrr@x@K\thr@@ \xspcode'X=\thr@@
624       \xkanjiskip\p@
625       \setbox\z@\hbox{\noautoxspacing\pxrr@x@K X}%
626       \setbox\tw@\hbox{\pxrr@x@K X}%
627       \ifdim\wd\tw@>\wd\z@
628         \aftergroup\pxrr@x@swatrue
629       \fi
630     \endgroup
631     \edef#1{%
632       \ifpxrr@x@swa \the\xkanjiskip
633       \else \pxrr@zeropt
634     \fi
635   }%
636 }
```

LuaTeX-ja 使用の場合。

```
637 \else\ifpxrr@in@luatexja
638   \def\pxrr@get@iaiskip#1{%
639     \edef#1{%
640       \ifnum\ltjgetparameter{autoxspacing}=\@ne
641         \ltjgetparameter{xkanjiskip}%
642       \else \pxrr@zeropt
643     \fi
644   }%
645 }
```

それ以外の場合は実際の組版結果から判断する。

```
646 \else
647   \def\pxrr@get@iaiskip#1{%
648     \begingroup
649       \setbox\z@\hbox{M\pxrr@x@K}%
650       \setbox\tw@\hbox{M\vrule\@width\z@\relax\pxrr@x@K}%
651       \@tempdima\wd\z@ \advance\@tempdima-\wd\tw@
652       \@tempdimb\@tempdima \divide\@tempdimb\thr@@
653       \xdef\pxrr@x@gtempa{\the\@tempdima\space minus \the\@tempdimb}%
654     \endgroup
655     \let#1=\pxrr@x@gtempa
656   }%
657 \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。

```
658 \ifpxrr@in@ptex
659   \def\pxrr@get@zwidth#1{%
```

```

660 \@tempdima=1zw\relax
661 \edef#1{\the\@tempdima}%
662 }

```

\zw が定義されている場合は 1\zw とする。

```

663 \else\if\ifx\zw\@undefined T\else F\fi F% if defined
664 \def\pxrr@get@zwidth#1{%
665 \@tempdima=1zw\relax
666 \edef#1{\the\@tempdima}%
667 }

```

\jsZw が定義されている場合は 1\jsZw とする。

```

668 \else\if\ifx\jsZw\@undefined T\else F\fi F% if defined
669 \def\pxrr@get@zwidth#1{%
670 \@tempdima=1\jsZw\relax
671 \edef#1{\the\@tempdima}%
672 }

```

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。

```

673 \else\ifnum\pxrr@x@K>\@cclv
674 \def\pxrr@get@zwidth#1{%
675 \setbox\tw\hbox{\pxrr@x@K}%
676 \@tempdima\wd\tw@
677 \ifdim\@tempdima>z@\else \@tempdima\f@size\p@ \fi
678 \edef#1{\the\@tempdima}%
679 }

```

それ以外の場合は要求サイズと等しいとする。

```

680 \else
681 \def\pxrr@get@zwidth#1{%
682 \@tempdima\f@size\p@\relax
683 \edef#1{\the\@tempdima}%
684 }
685 \fi\fi\fi\fi

```

\pxrr@get@prebreakpenalty \pxrr@get@prebreakpenalty\CS{<文字>} : 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、\prebreakpenalty を使う。

```

686 \ifpxrr@in@ptex
687 \def\pxrr@get@prebreakpenalty#1#2{%
688 #1=\prebreakpenalty'#2\relax
689 }

```

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

```

690 \else\ifpxrr@in@luatexja
691 \def\pxrr@get@prebreakpenalty#1#2{%
692 #1=\ltjgetparameter{prebreakpenalty}'#2\relax
693 }

```

それ以外の場合はゼロとして扱う。

```

694 \else
695   \def\pxrr@get@prebreakpenalty#1#2{%
696     #1=\z@
697   }
698 \fi\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。

```
699 \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。

```

700 \newcommand*\rubyssetup[1]{%
701   \pxrr@in@setuptrue
702   \pxrr@fatal@errorfalse
703   \pxrr@parse@option{#1}%
704   \ifpxrr@fatal@error\else
705     \pxrr@csletcs{ifpxrr@d@bprotr}{ifpxrr@bprotr}%
706     \pxrr@csletcs{ifpxrr@d@aprotr}{ifpxrr@aprotr}%
707     \let\pxrr@d@bintr\pxrr@bintr@
708     \let\pxrr@d@aintr\pxrr@aintr@
709     \let\pxrr@d@ahead\pxrr@ahead
710     \let\pxrr@d@mode\pxrr@mode
711     \let\pxrr@d@side\pxrr@side
712     \let\pxrr@d@evensp\pxrr@evensp
713     \let\pxrr@d@fullsize\pxrr@fullsize
714   \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。

```

715   \pxrr@in@setupfalse
716 }

```

`\rubyfontsetup` 対応するパラメタを設定する。

```

717 \newcommand*\rubyfontsetup{}
718 \def\rubyfontsetup#1{%
719   \def\pxrr@ruby@font
720 }

```

`\rubybigintrusion` 対応するパラメタを設定する。

```

\rubysmallintrusion 721 \newcommand*\rubybigintrusion[1]{%
\rubymaxmargin      722   \edef\pxrr@big@intr{#1}%
723 }
\rubyintergap       724 \newcommand*\rubysmallintrusion[1]{%
\rubysizeratio      725   \edef\pxrr@small@intr{#1}%
726 }
727 \newcommand*\rubymaxmargin[1]{%
728   \edef\pxrr@maxmargin{#1}%

```

```

729 }
730 \newcommand*\rubyintergap[1]{%
731   \edef\pxrr@inter@gap{#1}%
732 }
733 \newcommand*\rubysizeratio[1]{%
734   \edef\pxrr@size@ratio{#1}%
735 }

```

`\rubyusejghost` 対応するスイッチを設定する。

```

\rubynousejghost 736 \newcommand*\rubyusejghost{%
737   \pxrr@jghosttrue
738 }
739 \newcommand*\rubynousejghost{%
740   \pxrr@jghostfalse
741 }

```

`\rubyuseaghost` 対応するスイッチを設定する。

```

\rubynouseaghost 742 \newcommand*\rubyuseaghost{%
743   \pxrr@aghosttrue
744   \pxrr@setup@aghost
745 }
746 \newcommand*\rubynouseaghost{%
747   \pxrr@aghostfalse
748 }

```

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

```

\rubynoadjustatlineedge 749 \newcommand*\rubyadjustatlineedge{%
750   \pxrr@edge@adjusttrue
751 }
752 \newcommand*\rubynoadjustatlineedge{%
753   \pxrr@edge@adjustfalse
754 }

```

`\rubybreakjukugo` 対応するスイッチを設定する。

```

\rubynobreakjukugo 755 \newcommand*\rubybreakjukugo{%
756   \pxrr@break@jukugotrue
757 }
758 \newcommand*\rubynobreakjukugo{%
759   \pxrr@break@jukugofalse
760 }

```

`\rubysafemode` 対応するスイッチを設定する。

```

\rubynosafemode 761 \newcommand*\rubysafemode{%
762   \pxrr@safe@modetrue
763 }
764 \newcommand*\rubynosafemode{%
765   \pxrr@safe@modefalse
766 }

```

`\rubystretchprop` 対応するパラメタを設定する。

```

\rubystretchprophead 767 \newcommand*\rubystretchprop[3]{%
\rubystretchpropend 768   \edef\pxrr@sprop@x{#1}%
769   \edef\pxrr@sprop@y{#2}%
770   \edef\pxrr@sprop@z{#3}%
771 }
772 \newcommand*\rubystretchprophead[2]{%
773   \edef\pxrr@sprop@hy{#1}%
774   \edef\pxrr@sprop@hz{#2}%
775 }
776 \newcommand*\rubystretchpropend[2]{%
777   \edef\pxrr@sprop@ex{#1}%
778   \edef\pxrr@sprop@ey{#2}%
779 }

```

`\rubyuseextra` 残念ながら今のところは使用不可。

```

780 \newcommand*\rubyuseextra[1]{%
781   \pxrr@cmta=#1\relax
782   \ifnum\pxrr@cmta=\z@
783     \chardef\pxrr@extra\pxrr@cmta
784   \else
785     \pxrr@err@inv@value{\the\pxrr@cmta}%
786   \fi
787 }

```

## 4.8 ルビオプション解析

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

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

```

788 \let\pxrr@bintr@\@empty
789 \let\pxrr@aintr@\@empty

```

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

```

790 \def\pxrr@doublebar{||}

```

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

```

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

```

入力が「||」の場合は、「|-|」に置き換える。

```

792   \edef\pxrr@tempa{#1}%
793   \ifx\pxrr@tempa\pxrr@doublebar
794     \def\pxrr@tempa{|-|}%
795   \fi

```

各パラメタの値を全般設定のもので初期化する。

```

796   \pxrr@csletcs{ifpxrr@bprotr}{ifpxrr@d@bprotr}%
797   \pxrr@csletcs{ifpxrr@aprotr}{ifpxrr@d@aprotr}%

```

```

798 \let\pxrr@bintr@\pxrr@d@bintr
799 \let\pxrr@aintr@\pxrr@d@aintr
800 \let\pxrr@athead\pxrr@d@athead
801 \let\pxrr@mode\pxrr@d@mode
802 \let\pxrr@side\pxrr@d@side
803 \let\pxrr@evensp\pxrr@d@evensp
804 \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。

```

805 \let\pxrr@bscomp\relax
806 \let\pxrr@ascomp\relax
807 \pxrr@bnobrfalse
808 \pxrr@anobrfalse
809 \pxrr@bfintrfalse
810 \pxrr@afintrfalse

```

明示フラグを偽にする。

```

811 \pxrr@mode@givenfalse
812 \pxrr@athead@givenfalse

```

両側ルビの場合、基本モード既定値が M に固定される。

```

813 \ifpxrr@truby
814   \let\pxrr@mode=M%
815 \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。

```

816 \def\pxrr@po@FS{bi}%
817 \expandafter\pxrr@parse@option@loop\pxrr@tempa @\pxrr@end
818 }

```

有限状態機械のループ。

```

819 \def\pxrr@parse@option@loop#1{%
820 \ifpxrrDebug
821 \typeout{\pxrr@po@FS/#1[\@nameuse{pxrr@po@C@#1}]}%
822 \fi
823 \csname pxrr@po@PR@#1\endcsname
824 \expandafter\ifx\csname pxrr@po@C@#1\endcsname\relax
825   \let\pxrr@po@FS\relax
826 \else
827   \pxrr@letcs\pxrr@po@FS
828   {\pxrr@po@TR@\pxrr@po@FS @\@nameuse{pxrr@po@C@#1}]}%
829 \fi
830 \ifpxrrDebug
831 \typeout{->\pxrr@po@FS}%
832 \fi
833 \pxrr@ifx{\pxrr@po@FS\relax}{%
834   \pxrr@fatal@unx@letter{#1}%
835   \pxrr@parse@option@exit
836 }{%

```

```

837   \pxrr@parse@option@loop
838   }%
839 }

後処理。

840 \def\pxrr@parse@option@exit#1\pxrr@end{%
既定値設定 (\rubyssetup) である場合何もしない。
841   \ifpxrr@in@setup\else
両側ルビ命令の場合は、\pxrr@side の値を変更する。
842   \ifpxrr@truby
843     \chardef\pxrr@side\tw@
844   \fi

整合性検査を行う。

845   \pxrr@check@option
\pxrr@?@intr の値を設定する。
846   \@tempdima=\pxrr@ruby@zw\relax
847   \@tempdimb=\pxrr@or@zero\pxrr@bintr@\@tempdima
848   \edef\pxrr@bintr{\the\@tempdimb}%
849   \@tempdimb=\pxrr@or@zero\pxrr@aintr@\@tempdima
850   \edef\pxrr@aintr{\the\@tempdimb}%
851   \fi
852 }

```

\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。

```

853 \def\pxrr@or@zero#1{%
854   \ifx#1\@empty \pxrr@zero
855   \else #1%
856   \fi
857 }

```

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。

```

858 \def\pxrr@po@C@{F}
859 \@namedef{pxrr@po@C@|}{V}
860 \@namedef{pxrr@po@C@:}{S}
861 \@namedef{pxrr@po@C@.}{S}
862 \@namedef{pxrr@po@C@*}{S}
863 \@namedef{pxrr@po@C@!}{S}
864 \@namedef{pxrr@po@C@<}{B}
865 \@namedef{pxrr@po@C@(){B}
866 \@namedef{pxrr@po@C@>}{A}
867 \@namedef{pxrr@po@C@)}{A}
868 \@namedef{pxrr@po@C@-}{M}
869 \def\pxrr@po@C@c{M}
870 \def\pxrr@po@C@h{M}
871 \def\pxrr@po@C@H{M}

```

```

872 \def\pxrr@po@C@m{M}
873 \def\pxrr@po@C@g{M}
874 \def\pxrr@po@C@j{M}
875 \def\pxrr@po@C@M{M}
876 \def\pxrr@po@C@J{M}
877 \def\pxrr@po@C@P{M}
878 \def\pxrr@po@C@S{M}
879 \def\pxrr@po@C@e{M}
880 \def\pxrr@po@C@E{M}
881 \def\pxrr@po@C@f{M}
882 \def\pxrr@po@C@F{M}

```

機能プロセス。

```

883 \def\pxrr@po@PR@{%
884   \pxrr@parse@option@exit
885 }
886 \@namedef{pxrr@po@PR@|}{%
887   \csname pxrr@po@PRbar@\pxrr@po@FS\endcsname
888 }
889 \def\pxrr@po@PRbar@bi{%
890   \def\pxrr@bintr@{\pxrr@bprottrue
891 }
892 \def\pxrr@po@PRbar@bb{%
893   \pxrr@bprotrfalse
894 }
895 \def\pxrr@po@PRbar@bs{%
896   \def\pxrr@aintr@{\pxrr@aprotrtrue
897 }
898 \let\pxrr@po@PRbar@mi\pxrr@po@PRbar@bs
899 \let\pxrr@po@PRbar@as\pxrr@po@PRbar@bs
900 \let\pxrr@po@PRbar@ai\pxrr@po@PRbar@bs
901 \def\pxrr@po@PRbar@ab{%
902   \pxrr@aprotrfalse
903 }
904 \@namedef{pxrr@po@PR@:}{%
905   \csname pxrr@po@PRcolon@\pxrr@po@FS\endcsname
906 }
907 \def\pxrr@po@PRcolon@bi{%
908   \let\pxrr@bscomp:=\relax
909 }
910 \let\pxrr@po@PRcolon@bb\pxrr@po@PRcolon@bi
911 \let\pxrr@po@PRcolon@bs\pxrr@po@PRcolon@bi
912 \def\pxrr@po@PRcolon@mi{%
913   \let\pxrr@ascomp:=\relax
914 }
915 \let\pxrr@po@PRcolon@as\pxrr@po@PRcolon@mi
916 \@namedef{pxrr@po@PR@.}{%
917   \csname pxrr@po@PRdot@\pxrr@po@FS\endcsname
918 }

```

```

919 \def\pxrr@po@PRdot@bi{%
920   \let\pxrr@bscomp=. \relax
921 }
922 \let\pxrr@po@PRdot@bb\pxrr@po@PRdot@bi
923 \let\pxrr@po@PRdot@bs\pxrr@po@PRdot@bi
924 \def\pxrr@po@PRdot@mi{%
925   \let\pxrr@ascomp=. \relax
926 }
927 \let\pxrr@po@PRdot@as\pxrr@po@PRdot@mi
928 \@namedef{pxrr@po@PR@*}{%
929   \csname pxrr@po@PRstar@\pxrr@po@FS\endcsname
930 }
931 \def\pxrr@po@PRstar@bi{%
932   \pxrr@bnobrtrue
933 }
934 \let\pxrr@po@PRstar@bb\pxrr@po@PRstar@bi
935 \let\pxrr@po@PRstar@bs\pxrr@po@PRstar@bi
936 \def\pxrr@po@PRstar@mi{%
937   \pxrr@anobrtrue
938 }
939 \let\pxrr@po@PRstar@as\pxrr@po@PRstar@mi
940 \@namedef{pxrr@po@PR@!}{%
941   \csname pxrr@po@PRbang@\pxrr@po@FS\endcsname
942 }
943 \def\pxrr@po@PRbang@bi{%
944   \pxrr@bfintrtrue
945 }
946 \let\pxrr@po@PRbang@bb\pxrr@po@PRbang@bi
947 \let\pxrr@po@PRbang@bs\pxrr@po@PRbang@bi
948 \def\pxrr@po@PRbang@mi{%
949   \pxrr@afintrtrue
950 }
951 \let\pxrr@po@PRbang@as\pxrr@po@PRbang@mi
952 \@namedef{pxrr@po@PR@<}{%
953   \def\pxrr@bintr@{\pxrr@big@intr}\pxrr@bprottrue
954 }
955 \@namedef{pxrr@po@PR@}{%
956   \def\pxrr@bintr@{\pxrr@small@intr}\pxrr@bprottrue
957 }
958 \@namedef{pxrr@po@PR@>}{%
959   \def\pxrr@aintr@{\pxrr@big@intr}\pxrr@aprottrue
960 }
961 \@namedef{pxrr@po@PR@}{%
962   \def\pxrr@aintr@{\pxrr@small@intr}\pxrr@aprottrue
963 }
964 \def\pxrr@po@PR@c{%
965   \chardef\pxrr@athead\z@
966   \pxrr@athead@giventtrue
967 }

```

```

968 \def\pxrr@po@PR@h{%
969   \chardef\pxrr@thead\@ne
970   \pxrr@thead@giventru
971 }
972 \def\pxrr@po@PR@H{%
973   \chardef\pxrr@thead\tw@
974   \pxrr@thead@giventru
975 }
976 \def\pxrr@po@PR@m{%
977   \let\pxrr@mode=m%
978   \pxrr@mode@giventru
979 }
980 \def\pxrr@po@PR@g{%
981   \let\pxrr@mode=g%
982   \pxrr@mode@giventru
983 }
984 \def\pxrr@po@PR@j{%
985   \let\pxrr@mode=j%
986   \pxrr@mode@giventru
987 }
988 \def\pxrr@po@PR@M{%
989   \let\pxrr@mode=M%
990   \pxrr@mode@giventru
991 }
992 \def\pxrr@po@PR@J{%
993   \let\pxrr@mode=J%
994   \pxrr@mode@giventru
995 }
996 \def\pxrr@po@PR@P{%
997   \chardef\pxrr@side\z@
998 }
999 \def\pxrr@po@PR@S{%
1000   \chardef\pxrr@side\@ne
1001 }
1002 \def\pxrr@po@PR@E{%
1003   \chardef\pxrr@evensp\z@
1004 }
1005 \def\pxrr@po@PR@e{%
1006   \chardef\pxrr@evensp\@ne
1007 }
1008 \def\pxrr@po@PR@F{%
1009   \chardef\pxrr@fullsize\z@
1010 }
1011 \def\pxrr@po@PR@f{%
1012   \chardef\pxrr@fullsize\@ne
1013 }

```

遷移表。

```

1014 \def\pxrr@po@TR@bi@F{fi}

```

```

1015 \def\pxrr@po@TR@bb@F{fi}
1016 \def\pxrr@po@TR@bs@F{fi}
1017 \def\pxrr@po@TR@mi@F{fi}
1018 \def\pxrr@po@TR@as@F{fi}
1019 \def\pxrr@po@TR@ai@F{fi}
1020 \def\pxrr@po@TR@ab@F{fi}
1021 \def\pxrr@po@TR@fi@F{fi}
1022 \def\pxrr@po@TR@bi@V{bb}
1023 \def\pxrr@po@TR@bb@V{bs}
1024 \def\pxrr@po@TR@bs@V{ab}
1025 \def\pxrr@po@TR@mi@V{ab}
1026 \def\pxrr@po@TR@as@V{ab}
1027 \def\pxrr@po@TR@ai@V{ab}
1028 \def\pxrr@po@TR@ab@V{fi}
1029 \def\pxrr@po@TR@bi@S{bs}
1030 \def\pxrr@po@TR@bb@S{bs}
1031 \def\pxrr@po@TR@bs@S{bs}
1032 \def\pxrr@po@TR@mi@S{as}
1033 \def\pxrr@po@TR@as@S{as}
1034 \def\pxrr@po@TR@bi@B{bs}
1035 \def\pxrr@po@TR@bi@M{mi}
1036 \def\pxrr@po@TR@bb@M{mi}
1037 \def\pxrr@po@TR@bs@M{mi}
1038 \def\pxrr@po@TR@mi@M{mi}
1039 \def\pxrr@po@TR@bi@A{fi}
1040 \def\pxrr@po@TR@bb@A{fi}
1041 \def\pxrr@po@TR@bs@A{fi}
1042 \def\pxrr@po@TR@mi@A{fi}
1043 \def\pxrr@po@TR@as@A{fi}
1044 \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。

```

1045 \def\pxrr@mode@grand{%
1046   \if      m\pxrr@mode m%
1047   \else\if M\pxrr@mode m%
1048   \else\if j\pxrr@mode j%
1049   \else\if J\pxrr@mode j%
1050   \else\if g\pxrr@mode g%
1051   \else ?%
1052   \fi\fi\fi\fi\fi
1053 }

```

`\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。

```
1054 \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。

```
1055 \ifpxrr@bprotr\else
1056 \ifpxrr@aprotr\else
1057 \pxrr@fatal@bad@no@protr
1058 \fi
1059 \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。

```
1060 \pxrr@oktrue
1061 \ifx\pxrr@bintr@\@empty\else
1062 \pxrr@okfalse
1063 \fi
1064 \ifx\pxrr@aintr@\@empty\else
1065 \pxrr@okfalse
1066 \fi
1067 \ifpxrr@ghost\else
1068 \pxrr@oktrue
1069 \fi
1070 \ifpxrr@ok\else
1071 \pxrr@fatal@bad@intr
1072 \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。

```
1073 \if g\pxrr@mode\else
1074 \ifpxrr@abody
1075 \let\pxrr@mode=g\relax
1076 \ifpxrr@mode@given
1077 \pxrr@warn@must@group
1078 \fi
1079 \fi
1080 \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。

```
1081 \if \pxrr@mode@grand j%
1082 \ifnum\pxrr@side=\tw@
1083 \let\pxrr@mode=g\relax
1084 \ifpxrr@mode@given
1085 \pxrr@warn@bad@jukugo
1086 \fi
1087 \fi
1088 \fi
```

肩付き指定 (h) に関する検査。

```
1089 \ifnum\pxrr@athead>\z@
```

横組みでは不可なので中付きに変更する。

```
1090 \pxrr@if@in@tate{ }{%else
1091 \chardef\pxrr@athead\z@
1092 }%
```

グループルビでは不可なので中付きに変更する。

```
1093 \if g\pxrr@mode
1094 \chardef\pxrr@athead\z@
1095 \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。

```
1096 \ifnum\pxrr@athead=\z@
1097 \ifpxrr@athead@given
1098 \pxrr@warn@bad@athead
1099 \fi
1100 \fi
1101 \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。

```
1102 \ifpxrr@abody
1103 \chardef\pxrr@evensp\z@
1104 \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。)

```
1105 \if g\pxrr@mode\else
1106 \chardef\pxrr@evensp\@ne
1107 \fi
1108 }
```

## 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。

```
1109 \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。

```
1110 \let\pxrr@body@zw\pxrr@zeropt
1111 \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。

```
1112 \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)。

```

1113 \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。
1114 \def\pxrr@htratio{0}

\pxrr@iiskip 和文間空白および和欧文間空白の量。
\pxrr@iaiskip 1115 \let\pxrr@iiskip\pxrr@zeropt
1116 \let\pxrr@iaiskip\pxrr@zeropt

\pxrr@assign@fsize 上記の変数（マクロ）を設定する。
1117 \def\pxrr@assign@fsize{%
1118   \@tempdima=f@size\p@
1119   \@tempdima\pxrr@size@ratio\@tempdima
1120   \edef\pxrr@ruby@fsize{\the\@tempdima}%
1121   \pxrr@get@zwidth\pxrr@body@zw
1122   \begingroup
1123     \pxrr@use@ruby@font
1124     \pxrr@get@zwidth\pxrr@ruby@zw
1125     \global\let\pxrr@gtempa\pxrr@ruby@zw
1126   \endgroup
1127   \let\pxrr@ruby@zw\pxrr@gtempa
1128   \pxrr@get@iiskip\pxrr@iiskip
1129   \pxrr@get@iaiskip\pxrr@iaiskip

   \pxrr@htratio の値を設定する。
1130   \pxrr@if@in@tate{%
1131     \let\pxrr@htratio\pxrr@thtratio
1132   }{%
1133     \let\pxrr@htratio\pxrr@yhtratio
1134   }%

   \pxrr@ruby@raise の値を計算する。
1135   \@tempdima\pxrr@body@zw\relax
1136   \@tempdima\pxrr@htratio\@tempdima
1137   \@tempdimb\pxrr@ruby@zw\relax
1138   \advance\@tempdimb-\pxrr@htratio\@tempdimb
1139   \advance\@tempdima\@tempdimb
1140   \@tempdimb\pxrr@body@zw\relax
1141   \advance\@tempdima\pxrr@inter@gap\@tempdimb
1142   \edef\pxrr@ruby@raise{\the\@tempdima}%

   \pxrr@ruby@lower の値を計算する。
1143   \@tempdima\pxrr@body@zw\relax
1144   \advance\@tempdima-\pxrr@htratio\@tempdima
1145   \@tempdimb\pxrr@ruby@zw\relax
1146   \@tempdimb\pxrr@htratio\@tempdimb
1147   \advance\@tempdima\@tempdimb
1148   \@tempdimb\pxrr@body@zw\relax
1149   \advance\@tempdima\pxrr@inter@gap\@tempdimb
1150   \edef\pxrr@ruby@lower{\the\@tempdima}%

```

1151 }

`\pxrr@use@ruby@font` ルビ用のフォントに切り替える。

```
1152 \def\pxrr@use@ruby@font{%
1153   \pxrr@without@macro@trace{%
1154     \let\rubyfontsize\pxrr@ruby@fsize
1155     \fontsize{\pxrr@ruby@fsize}{\z@}\selectfont
1156     \pxrr@ruby@font
1157   }%
1158 }
```

#### 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 1159 \chardef\pxrr@locate@inner=1
\pxrr@locate@end 1160 \chardef\pxrr@locate@head=0
1161 \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。

```
1162 \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。

```
1163 \setbox#2\pxrr@hbox{#5}\@tempdima\wd#2%
```

```
1164 \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。

```
1165 \pxrr@decompose{#5}%
```

```
1166 \pxrr@evenspace@int{#1}{#2}{#3}{#4}%
```

```
1167 }
```

ここから実行を開始することもある。

```
1168 \def\pxrr@evenspace@int#1#2#3#4{%
```

比率パラメタの設定。

```
1169 \pxrr@save@listproc
```

```
1170 \ifcase#1%
```

```

1171 \pxrr@evenspace@param\pxrr@zero\pxrr@sprop@hy\pxrr@sprop@hz
1172 \or
1173 \pxrr@evenspace@param\pxrr@sprop@x\pxrr@sprop@y\pxrr@sprop@z
1174 \or
1175 \pxrr@evenspace@param\pxrr@sprop@ex\pxrr@sprop@ey\pxrr@zero
1176 \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。

```

1177 \pxrr@dima=\pxrr@cntr\p@
1178 \advance\pxrr@dima-\p@
1179 \pxrr@dima=\pxrr@sprop@y@\pxrr@dima
1180 \advance\pxrr@dima\pxrr@sprop@x@\p@
1181 \advance\pxrr@dima\pxrr@sprop@z@\p@
1182 \ifdim\pxrr@dima>\z@\else
1183 \ifnum#1>\z@
1184 \let\pxrr@sprop@x@\@ne
1185 \advance\pxrr@dima\p@
1186 \fi
1187 \ifnum#1<\tw@
1188 \let\pxrr@sprop@z@\@ne
1189 \advance\pxrr@dima\p@
1190 \fi
1191 \fi
1192 \edef\pxrr@tempa{\strip@pt\pxrr@dima}%
1193 \ifpxrr@Debug
1194 \typeout{\number\pxrr@sprop@x@:\number\pxrr@sprop@z@:\pxrr@tempa}%
1195 \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。

```

1196 \def\pxrr@pre##1{\pxrr@hfilx\pxrr@sprop@x@ ##1}%
1197 \def\pxrr@inter##1{\pxrr@hfilx\pxrr@sprop@y@ ##1}%
1198 \def\pxrr@post{\pxrr@hfilx\pxrr@sprop@z@}%
1199 \def\pxrr@makebox@res{%
1200 \setbox#2=\pxrr@hbox@to#4{#3\pxrr@res}%
1201 }%
1202 \pxrr@makebox@res

```

前後の空白の量を求める。

```

1203 \pxrr@dima\wd#2%
1204 \advance\pxrr@dima-\pxrr@natwd\relax
1205 \pxrr@invscale\pxrr@dima\pxrr@tempa
1206 \@tempdima\pxrr@sprop@x@\pxrr@dima
1207 \edef\pxrr@bspace{\the\@tempdima}%
1208 \@tempdima\pxrr@sprop@z@\pxrr@dima
1209 \edef\pxrr@aspace{\the\@tempdima}%
1210 \pxrr@restore@listproc

```

```

1211 \ifpxrrDebug
1212 \typeout{\pxrr@bspace:\pxrr@aspace}%
1213 \fi
1214 }
1215 \def\pxrr@evenspace@param#1#2#3{%
1216   \let\pxrr@sprop@x@#1%
1217   \let\pxrr@sprop@y@#2%
1218   \let\pxrr@sprop@z@#3%
1219 }
1220 \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。

```

1221 \def\pxrr@adjust@margin{%
1222   \pxrr@save@listproc
1223   \@tempdima\pxrr@body@zw\relax
1224   \@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。

```

1225   \@tempswafalse
1226   \def\pxrr@pre##1{\pxrr@hfilx\pxrr@sprop@x@ ##1}%
1227   \def\pxrr@inter##1{\pxrr@hfilx\pxrr@sprop@y@ ##1}%
1228   \def\pxrr@post{\pxrr@hfilx\pxrr@sprop@z@}%
1229   \ifnum\pxrr@cntr>\@ne
1230     \ifdim\pxrr@bspace>\@tempdima
1231       \edef\pxrr@bspace{\the\@tempdima}%
1232       \def\pxrr@pre##1{\hskip\pxrr@bspace\relax ##1}%
1233       \@tempswatrue
1234     \fi
1235     \ifdim\pxrr@aspace>\@tempdima
1236       \edef\pxrr@aspace{\the\@tempdima}%
1237       \def\pxrr@post{\hskip\pxrr@aspace\relax}%
1238       \@tempswatrue
1239     \fi
1240   \fi

```

必要に応じて再調整を行う。

```

1241   \if@tempswa
1242     \pxrr@makebox@res
1243   \fi
1244   \pxrr@restore@listproc
1245 \ifpxrrDebug
1246 \typeout{\pxrr@bspace:\pxrr@aspace}%
1247 \fi
1248 }

```

`\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。

```
1249 \def\pxrr@save@listproc{%
1250   \let\pxrr@pre@save\pxrr@pre
1251   \let\pxrr@inter@save\pxrr@inter
1252   \let\pxrr@post@save\pxrr@post
1253 }
1254 \let\pxrr@pre@save\@undefined
1255 \let\pxrr@inter@save\@undefined
1256 \let\pxrr@post@save\@undefined
```

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

```
1257 \def\pxrr@restore@listproc{%
1258   \let\pxrr@pre\pxrr@pre@save
1259   \let\pxrr@inter\pxrr@inter@save
1260   \let\pxrr@post\pxrr@post@save
1261 }
```

## 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。

```
1262 \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。

```
1263 \def\pxrr@transform@kana#1{%
1264   \let\pxrr@trans@res\@empty
1265   \def\pxrr@transform@kana@end\pxrr@end{%
1266     \let#1\pxrr@trans@res
1267   }%
1268   \expandafter\pxrr@transform@kana@loop@a#1\pxrr@end
1269 }
1270 \def\pxrr@transform@kana@loop@a{%
1271   \futurelet\pxrr@token\pxrr@transform@kana@loop@b
1272 }
1273 \def\pxrr@transform@kana@loop@b{%
1274   \ifx\pxrr@token\pxrr@end
1275     \let\pxrr@tempb\pxrr@transform@kana@end
1276   \else\ifx\pxrr@token\bgroup
1277     \let\pxrr@tempb\pxrr@transform@kana@loop@c
1278   \else\ifx\pxrr@token\@sptoken
1279     \let\pxrr@tempb\pxrr@transform@kana@loop@d
1280   \else
1281     \let\pxrr@tempb\pxrr@transform@kana@loop@e
1282   \fi\fi\fi
1283   \pxrr@tempb
1284 }
1285 \def\pxrr@transform@kana@loop@c#1{%
```

```

1286 \pxrr@appto\pxrr@trans@res{#1}}%
1287 \pxrr@transform@kana@loop@a
1288 }
1289 \expandafter\def\expandafter\pxrr@transform@kana@loop@d\space{%
1290 \pxrr@appto\pxrr@trans@res{ }%
1291 \pxrr@transform@kana@loop@a
1292 }
1293 \def\pxrr@transform@kana@loop@e#1{%
1294 \expandafter\pxrr@transform@kana@loop@f\string#1\pxrr@nil#1%
1295 }
1296 \def\pxrr@transform@kana@loop@f#1#2\pxrr@nil#3{%
1297 \@tempwafalse
1298 \ifnum'#1>\@cclv
1299 \begingroup\expandafter\expandafter\expandafter\endgroup
1300 \expandafter\ifx\csname pxrr@nonsmall/#3\endcsname\relax\else
1301 \@tempwatrue
1302 \fi
1303 \fi
1304 \if@tempswa
1305 \edef\pxrr@tempa{%
1306 \noexpand\pxrr@appto\noexpand\pxrr@trans@res
1307 {\csname pxrr@nonsmall/#3\endcsname}%
1308 }%
1309 \pxrr@tempa
1310 \else
1311 \pxrr@appto\pxrr@trans@res{#3}%
1312 \fi
1313 \pxrr@transform@kana@loop@a
1314 }
1315 \def\pxrr@assign@nonsmall#1/#2\pxrr@nil{%
1316 \pxrr@get@jchar@token\pxrr@tempa{\pxrr@jc{#1}}%
1317 \pxrr@get@jchar@token\pxrr@tempb{\pxrr@jc{#2}}%
1318 \expandafter\edef\csname pxrr@nonsmall/\pxrr@tempa\endcsname
1319 {\pxrr@tempb}%
1320 }
1321 \@tfor\pxrr@tempc:=%
1322 {2421:3041/2422:3042}{2423:3043/2424:3044}%
1323 {2425:3045/2426:3046}{2427:3047/2428:3048}%
1324 {2429:3049/242A:304A}{2443:3063/2444:3064}%
1325 {2463:3083/2464:3084}{2465:3085/2466:3086}%
1326 {2467:3087/2468:3088}{246E:308E/246F:308F}%
1327 {2521:30A1/2522:30A2}{2523:30A3/2524:30A4}%
1328 {2525:30A5/2526:30A6}{2527:30A7/2528:30A8}%
1329 {2529:30A9/252A:30AA}{2543:30C3/2544:30C4}%
1330 {2563:30E3/2564:30E4}{2565:30E5/2566:30E6}%
1331 {2567:30E7/2568:30E8}{256E:30EE/256F:30EF}%
1332 \do{%
1333 \expandafter\pxrr@assign@nonsmall\pxrr@tempc\pxrr@nil
1334 }

```

#### 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。

```
1335 \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。

```
1336 \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。

```
1337 \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）。

```
1338 \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@ospace`に返す。

```
1339 \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。

```
1340 \let\pxrr@compose@block@do\pxrr@compose@oneside@block@do
```

```
1341 \pxrr@compose@block@pre{#1}{#2}{#3}{}%
```

```
1342 }
```

こちらが本体。

```
1343 % #4 は空
```

```
1344 \def\pxrr@compose@oneside@block@do#1#2#3#4{%
```

```
1345 \setbox\pxrr@boxa\pxrr@hbox{#2}%
```

```
1346 \setbox\pxrr@boxr\pxrr@hbox{%
```

```
1347 \pxrr@use@ruby@font
```

```
1348 #3%
```

```
1349 }%
```

```
1350 \@tempdima\wd\pxrr@boxr
```

```
1351 \advance\@tempdima-\wd\pxrr@boxa
```

```
1352 \ifdim\pxrr@epsilon<\@tempdima
```

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

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

```
1353 \pxrr@protrtrue
```

```
1354 \let\pxrr@locate@temp#1%
```

```
1355 \ifnum\pxrr@athead>\@ne
```

```
1356 \ifnum\pxrr@locate@temp=\pxrr@locate@inner
```

```

1357     \let\pxrr@locate@temp\pxrr@locate@head
1358     \fi
1359     \fi
1360     \pxrr@decompose{#2}%
1361     \edef\pxrr@natwd{\the\wd\pxrr@boxa}%
1362     \pxrr@evenspace@int\pxrr@locate@temp\pxrr@boxa\relax
1363     {\wd\pxrr@boxr}%
1364     \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。

```

1365     \pxrr@protrfalse
1366     \ifnum\pxrr@athead=\z@
1367     \pxrr@decompose{#3}%
1368     \edef\pxrr@natwd{\the\wd\pxrr@boxr}%
1369     \pxrr@evenspace@int{#1}\pxrr@boxr
1370     \pxrr@use@ruby@font{\wd\pxrr@boxa}%
1371     \pxrr@adjust@margin
1372     \fi
1373     \let\pxrr@bspace\pxrr@zeropt
1374     \let\pxrr@aspace\pxrr@zeropt
1375     \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）。

```

1376     \pxrr@protrfalse
1377     \let\pxrr@bspace\pxrr@zeropt
1378     \let\pxrr@aspace\pxrr@zeropt
1379     \fi\fi

```

実際に組版を行う。

```

1380     \setbox\z@\hbox{%
1381     \ifnum\pxrr@side=\z@
1382     \raise\pxrr@ruby@raise\box\pxrr@boxr
1383     \else
1384     \lower\pxrr@ruby@lower\box\pxrr@boxr
1385     \fi
1386     }%
1387     \ht\z@\z@ \dp\z@\z@
1388     \@tempdima\wd\z@
1389     \setbox\pxrr@boxr\hbox{%
1390     \box\z@
1391     \kern-\@tempdima
1392     \box\pxrr@boxa
1393     }%

```

`\ifpxrr@any@protr` を設定する。

```

1394     \ifpxrr@protr
1395     \pxrr@any@protrtrue

```

```
1396 \fi
1397 }
```

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

```
1398 \def\pxrr@compose@twoside@block{%
1399 \let\pxrr@compose@block@do\pxrr@compose@twoside@block@do
1400 \pxrr@compose@block@pre
1401 }
1402 \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。

```
1403 \setbox\pxrr@boxa\pxrr@hbox{#2}%
1404 \setbox\pxrr@boxr\pxrr@hbox{%
1405 \pxrr@use@ruby@font
1406 #3%
1407 }%
1408 \setbox\pxrr@boxb\pxrr@hbox{%
1409 \pxrr@use@ruby@font
1410 #4%
1411 }%
```

「何れかのルビが親文字列より長いか」を検査する。

```
1412 \@tempwafalse
1413 \@tempdima\wd\pxrr@boxr
1414 \advance\@tempdima-\wd\pxrr@boxa
1415 \ifdim\pxrr@epsilon<\@tempdima \@tempwatrue \fi
1416 \@tempdima\wd\pxrr@boxb
1417 \advance\@tempdima-\wd\pxrr@boxa
1418 \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。)

```
1419 \if@tempwa
1420 \pxrr@protrtrue
```

「拡張肩付き」指定の場合、前側の突出を抑止する。

```
1421 \let\pxrr@locate@temp#1%
1422 \ifnum\pxrr@athead>\@ne
1423 \ifnum\pxrr@locate@temp=\pxrr@locate@inner
1424 \let\pxrr@locate@temp\pxrr@locate@head
1425 \fi
1426 \fi
```

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。

```
1427 \ifdim\wd\pxrr@boxr<\wd\pxrr@boxb
1428 \pxrr@compose@twoside@block@sub{#2}{#3}%
1429 \pxrr@boxr\pxrr@boxb
```

```

1430 \else
1431 \pxrr@compose@twoside@block@sub{#2}{#4}%
1432 \pxrr@boxb\pxrr@boxr
1433 \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。

```

1434 \else
1435 \pxrr@protrfalse

```

肩付きルビの場合は組み直しを行わない。

```

1436 \ifnum\pxrr@athead=\z@
1437 \@tempdima\wd\pxrr@boxa
1438 \advance\@tempdima-\wd\pxrr@boxr
1439 \ifdim\pxrr@epsilon<\@tempdima
1440 \pxrr@decompose{#3}%
1441 \edef\pxrr@natwd{\the\wd\pxrr@boxr}%
1442 \pxrr@evenspace@int{#1}\pxrr@boxr
1443 \pxrr@use@ruby@font{\wd\pxrr@boxa}%
1444 \pxrr@adjust@margin
1445 \fi
1446 \@tempdima\wd\pxrr@boxa
1447 \advance\@tempdima-\wd\pxrr@boxb
1448 \ifdim\pxrr@epsilon<\@tempdima
1449 \pxrr@decompose{#4}%
1450 \edef\pxrr@natwd{\the\wd\pxrr@boxb}%
1451 \pxrr@evenspace@int{#1}\pxrr@boxb
1452 \pxrr@use@ruby@font{\wd\pxrr@boxa}%
1453 \pxrr@adjust@margin
1454 \fi
1455 \fi

```

\pxrr@?space はゼロに設定する。

```

1456 \let\pxrr@bspace\pxrr@zeropt
1457 \let\pxrr@aspace\pxrr@zeropt
1458 \fi

```

実際に組版を行う。

```

1459 \setbox\z@\hbox{%
1460 \@tempdima\wd\pxrr@boxr
1461 \raise\pxrr@ruby@raise\box\pxrr@boxr
1462 \kern-\@tempdima
1463 \lower\pxrr@ruby@lower\box\pxrr@boxb
1464 }%
1465 \ht\z@\z@ \dp\z@\z@
1466 \@tempdima\wd\z@
1467 \setbox\pxrr@boxr\hbox{%
1468 \box\z@
1469 \kern-\@tempdima
1470 \box\pxrr@boxa

```

```
1471 }%
1472 }
```

`\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”（均等割りに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。

```
1473 \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。

```
1474 \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。

```
1475 \def\pxrr@compose@twoside@block@sub#1#2#3#4{%
1476   \pxrr@decompose{#1}%
1477   \edef\pxrr@natwd{\the\wd\pxrr@boxa}%
1478   \pxrr@evenspace@int\pxrr@locate@temp\pxrr@boxa\relax{\wd#4}%
1479   \@tempdima\wd#4%
1480   \advance\@tempdima-\pxrr@bspace\relax
1481   \advance\@tempdima-\pxrr@aspace\relax
1482   \edef\pxrr@body@wd{\the\@tempdima}%
1483   \advance\@tempdima-\wd#3%
1484   \ifdim\pxrr@epsilon<\@tempdima
1485     \edef\pxrr@restore@margin@values{%
1486       \edef\noexpand\pxrr@bspace{\pxrr@bspace}%
1487       \edef\noexpand\pxrr@aspace{\pxrr@aspace}%
1488     }%
1489     \pxrr@decompose{#2}%
1490     \edef\pxrr@natwd{\the\wd#3}%
1491     \pxrr@evenspace@int\pxrr@locate@temp#3%
1492     \pxrr@use@ruby@font{\pxrr@body@wd}%
1493     \pxrr@adjust@margin
1494     \pxrr@restore@margin@values
1495     \setbox#3\hbox{%
1496       \kern\pxrr@bspace\relax
1497       \box#3%
1498     }%
1499   \else
1500     \ifnum\pxrr@locate@temp=\pxrr@locate@head
1501       \@tempdima\z@
1502     \else\ifnum\pxrr@locate@temp=\pxrr@locate@inner
1503       \@tempdima.5\@tempdima
1504     \fi\fi
1505     \advance\@tempdima\pxrr@bspace\relax
1506     \setbox#3\hbox{%
```

```

1507     \kern\@tempdima
1508     \box#3%
1509   }%
1510   \fi
1511 }
1512 %     \end{macrocode}
1513 % \end{macro}
1514 %
1515 % \begin{macro}{\pxrr@compose@block@pre}
1516 % |\pxrr@compose@block@pre{|\jmeta{パターン}}|{|^A
1517 %r \jmeta{親文字}}{|\jmeta{ルビ 1}}{|\jmeta{ルビ 2}}|\Means
1518 % 親文字列・ルビ文字列の加工を行う。
1519 % \Note 両側ルビ対応のため、ルビ用引数が 2 つある。
1520 %     \begin{macrocode}
1521 \def\pxrr@compose@block@pre{%
    f 指定時は小書き仮名の変換を施す。
1522   \pxrr@cond\ifnum\pxrr@fullsize>\z@\fi{%
1523     \pxrr@compose@block@pre@a
1524   }{%
1525     \pxrr@compose@block@pre@d
1526   }%
1527 }
1528 % {パターン}{親文字}{ルビ 1}{ルビ 2}
1529 \def\pxrr@compose@block@pre@a#1#2#3#4{%
1530   \def\pxrr@compose@block@tempa{#4}%
1531   \pxrr@transform@kana\pxrr@compose@block@tempa
1532   \expandafter\pxrr@compose@block@pre@b
1533   \expandafter{\pxrr@compose@block@tempa}{#1}{#2}{#3}%
1534 }
1535 % {ルビ 2}{パターン}{親文字}{ルビ 1}
1536 \def\pxrr@compose@block@pre@b#1#2#3#4{%
1537   \def\pxrr@compose@block@tempa{#4}%
1538   \pxrr@transform@kana\pxrr@compose@block@tempa
1539   \expandafter\pxrr@compose@block@pre@c
1540   \expandafter{\pxrr@compose@block@tempa}{#1}{#2}{#3}%
1541 }
1542 % {ルビ 1}{ルビ 2}{パターン}{親文字}
1543 \def\pxrr@compose@block@pre@c#1#2#3#4{%
1544   \pxrr@compose@block@pre@d{#3}{#4}{#1}{#2}%
1545 }
1546 \def\pxrr@compose@block@pre@d{%
1547   \pxrr@cond\ifnum\pxrr@evensp=\z@\fi{%
1548     \pxrr@compose@block@pre@e
1549   }{%
1550     \pxrr@compose@block@pre@f
1551   }%
1552 }
1553 % {パターン}{親文字}

```

```

1554 \def\pxrr@compose@block@pre@e#1#2{%
1555   \pxrr@compose@block@pre@f{#1}{#2}}%
1556 }
1557 \def\pxrr@compose@block@pre@f{%
1558   \pxrr@cond\ifnum\pxrr@revensp=\z@\fi{%
1559     \pxrr@compose@block@pre@g
1560   }{%
1561     \pxrr@compose@block@do
1562   }%
1563 }
1564 % {パターン}{親文字}{ルビ 1}{ルビ 2}
1565 \def\pxrr@compose@block@pre@g#1#2#3#4{%
1566   \pxrr@compose@block@do{#1}{#2}{#3}{#4}}%
1567 }
1568 \let\pxrr@compose@block@tempa\@undefined

```

#### 4.14 命令の頑強化

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

※ `\CS` は「制御語」(制御記号でなく)である必要がある。

```

1569 \def\pxrr@add@protect#1{%
1570   \expandafter\pxrr@add@protect@a
1571   \csname\expandafter\@gobble\string#1\space\endcsname#1%
1572 }
1573 \def\pxrr@add@protect@a#1#2{%
1574   \let#1=#2%
1575   \def#2{\pxrr@check@protect\protect#1}%
1576 }
1577 \def\pxrr@check@protect{%
1578   \ifx\protect\@typeset@protect
1579     \expandafter\@gobble
1580   \fi
1581 }

```

#### 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。  
1582 `\let\pxrr@body@input\@empty`

`\pxrr@prepare@fallback` `\pxrr@prepare@fallback{〈親文字列〉}` :  
1583 `\def\pxrr@prepare@fallback#1{%`  
1584 `\pxrr@fatal@errorfalse`  
1585 `\def\pxrr@body@input{#1}%`  
1586 `}`

`\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。  
1587 `\def\pxrr@fallback{%`  
1588 `\pxrr@body@input`  
1589 `}`

`\pxrr@if@alive` `\pxrr@if@alive{〈コード〉}` : 致命的エラーが未発生の場合に限り、〈コード〉に展開する。  
1590 `\def\pxrr@if@alive{%`  
1591 `\ifpxrr@fatal@error \expandafter\@gobble`  
1592 `\else \expandafter\@firstofone`  
1593 `\fi`  
1594 `}`

#### 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。  
1595 `\def\pxrr@end@kinsoku{0}`

`\pxrr@ruby@scan` 片側ルビ用の先読み処理。  
1596 `\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。  
1597 `\def\pxrr@tempc{%`  
1598 `\edef\pxrr@end@kinsoku{\the\pxrr@cntr}%`  
1599 `\pxrr@do@proc{#1}{#2}%`  
1600 `}%`  
1601 `\pxrr@check@kinsoku\pxrr@tempc`  
1602 `}`

`\pxrr@truby@scan` 両側ルビ用の先読み処理。  
1603 `\def\pxrr@truby@scan#1#2#3{%`  
1604 `\def\pxrr@tempc{%`  
1605 `\edef\pxrr@end@kinsoku{\the\pxrr@cntr}%`  
1606 `\pxrr@do@proc{#1}{#2}{#3}%`

```

1607 }%
1608 \pxrr@check@kinsoku\pxrr@tempc
1609 }

```

`\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。

```

1610 \def\pxrr@check@kinsoku#1{%
1611   \let\pxrr@tempb#1%
1612   \futurelet\pxrr@token\pxrr@check@kinsoku@a
1613 }
1614 \def\pxrr@check@kinsoku@a{%
1615   \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。

```

1616   \ifpxrr@abody\else
1617     \ifnum\pxrr@cntr=\@ne
1618       \pxrr@cntr\tw@
1619     \fi
1620   \fi
1621   \ifcase\pxrr@cntr
1622     \pxrr@cntr\z@
1623     \expandafter\pxrr@tempb
1624   \or
1625     \pxrr@cntr\@MM
1626     \expandafter\pxrr@tempb
1627   \else
1628     \expandafter\pxrr@check@kinsoku@b
1629   \fi
1630 }

```

`\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。

```

1631 \def\pxrr@check@kinsoku@b#1{%
1632   \pxrr@check@kinsoku@c#1#1%
1633 }
1634 \def\pxrr@check@kinsoku@c#1{%
1635   \pxrr@get@prebreakpenalty\pxrr@cntr{#1}%
1636   \pxrr@tempb
1637 }

```

`\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。

```
1638 \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。)

```
1639 \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。

```
1640 \ifcat\noexpand##1\relax
1641 \pxrr@cntr\z@
1642 \else\ifcat\noexpand##1\noexpand~%
1643 \pxrr@cntr\z@
1644 \else\ifcat\noexpand##1A%
1645 \pxrr@cntr\@ne
1646 \else\ifcat\noexpand##10%
1647 \pxrr@cntr\@ne
1648 \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。)

```
1649 \pxrr@cntr\z@
1650 \expandafter\pxrr@check@char@a\meaning##1#2\pxrr@nil
1651 \fi\fi\fi\fi
1652 }%
1653 \def\pxrr@check@char@a##1#2##2\pxrr@nil{%
1654 \ifcat @##1@%
1655 \pxrr@cntr\tw@
1656 \fi
1657 }%
1658 }
```

規定の引数を用意して「定義マクロ」を呼ぶ。

```
1659 \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。)

```
1660 \let\pxrr@auto@penalty\z@
```

\pxrr@auto@icspace 文字間の空き。寸法値マクロ。

```
1661 \let\pxrr@auto@icspace\pxrr@zeropt
```

\pxrr@intr@amount 進入の幅。寸法値マクロ。

```
1662 \let\pxrr@intr@amount\pxrr@zeropt
```

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

```
1663 \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。

```
1664 \ifpxrr@bnoBr
```

```

1665 \let\pxrr@auto@penalty\@MM
1666 \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。

```

1667 \else
1668 \let\pxrr@auto@penalty\z@
1669 \if:\pxrr@bscomp
1670 \let\pxrr@auto@icspace\pxrr@iaiskip
1671 \else\if.\pxrr@bscomp
1672 \let\pxrr@auto@icspace\pxrr@zeropt
1673 \else
1674 \let\pxrr@auto@icspace\pxrr@iiskip
1675 \fi\fi
1676 \fi
1677 }

```

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

```

1678 \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。

```

1679 \if:\pxrr@bscomp\else
1680 \pxrr@bnobrtrue
1681 \fi
1682 \ifpxrr@bnobr
1683 \let\pxrr@auto@penalty\@MM
1684 \let\pxrr@auto@icspace\pxrr@zeropt
1685 \else

```

この分岐は和欧文間空白挿入指定 (:) に限る。

```

1686 \let\pxrr@auto@penalty\z@
1687 \let\pxrr@auto@icspace\pxrr@iaiskip
1688 \fi
1689 }

```

#### 4.17.1 前側進入処理

`\pxrr@intrude@head` 前側の進入処理。

```

1690 \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。)

```

1691 \ifpxrr@ghost\else

```

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

```

1692 \let\pxrr@intr@amount\pxrr@bspace
1693 \ifdim\pxrr@bintr<\pxrr@intr@amount\relax
1694 \let\pxrr@intr@amount\pxrr@bintr
1695 \fi

```

`\pxrr@auto@*` の設定法は和文ルビと欧文ルビで処理が異なる。

```

1696 \ifpxrr@abody
1697 \pxrr@intrude@setauto@a
1698 \else
1699 \pxrr@intrude@setauto@j
1700 \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)。

```

1701 \ifpxrr@par@head
1702 \ifpxrr@bfintr
1703 \hskip-\pxrr@intr@amount\relax
1704 \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。

```

1705 \else
1706 % \penalty\pxrr@auto@penalty\relax
1707 \hskip-\pxrr@intr@amount\relax
1708 \hskip\pxrr@auto@icspace\relax
1709 \fi
1710 \fi
1711 }

```

\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。

```

1712 \def\pxrr@put@head@penalty{%
1713 \ifpxrr@ghost\else \ifpxrr@par@head\else
1714 \ifpxrr@abody
1715 \pxrr@intrude@setauto@a
1716 \else
1717 \pxrr@intrude@setauto@j
1718 \fi
1719 \ifnum\pxrr@auto@penalty=\z@\else
1720 \pxrr@canta\lastpenalty \unpenalty
1721 \advance\pxrr@canta\pxrr@auto@penalty\relax
1722 \penalty\pxrr@canta
1723 \fi
1724 \fi\fi
1725 }

```

#### 4.17.2 後側進入処理

\pxrr@intrude@end 末尾での進入処理。

```

1726 \def\pxrr@intrude@end{%
1727 \ifpxrr@ghost\else

```

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

```

1728 \let\pxrr@intr@amount\pxrr@aspace
1729 \ifdim\pxrr@aintr<\pxrr@intr@amount\relax
1730 \let\pxrr@intr@amount\pxrr@aintr
1731 \fi

```

\pxrr@auto@\* の設定法は和文ルビと欧文ルビで処理が異なる。

```

1732 \pxrr@csletcs{ifpxrr@bnober}{ifpxrr@anober}%
1733 \let\pxrr@bscomp\pxrr@ascomp
1734 \ifpxrr@abody
1735 \pxrr@intrude@setauto@a
1736 \else
1737 \pxrr@intrude@setauto@j
1738 \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。

```

1739 \ifnum\pxrr@auto@penalty=\z@
1740 \let\pxrr@auto@penalty\pxrr@end@kinsoku
1741 \fi
1742 \ifpxrr@afintr

```

段落末尾での進入を許す場合。

```

1743 \ifnum\pxrr@auto@penalty=\z@\else
1744 \penalty\pxrr@auto@penalty\relax
1745 \fi
1746 \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。

```

1747 \hskip\pxrr@auto@icspace\relax
1748 \ifnum\pxrr@auto@penalty=\z@\else
1749 \penalty\@MM
1750 \fi
1751 \else

```

段落末尾での進入を許さない場合。

```

1752 \@tempkipa-\pxrr@intr@amount\relax
1753 \advance\@tempkipa\pxrr@auto@icspace\relax
1754 \ifnum\pxrr@auto@penalty=\z@\else
1755 \penalty\pxrr@auto@penalty\relax
1756 \fi
1757 \hskip\@tempkipa
1758 \ifnum\pxrr@auto@penalty=\z@\else
1759 \penalty\@MM
1760 \fi
1761 \fi
1762 \fi
1763 }

```

## 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。

```
1764 \AtBeginDocument{%
1765   \providecommand*\ruby{\jruby}%
1766 }
1767 \newcommand*\jruby{%
1768   \pxrr@jprologue
1769   \pxrr@trubyfalse
1770   \pxrr@ruby
1771 }
```

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

```
1772 \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。

```
1773 \newcommand*\aruby{%
1774   \pxrr@aprologue
1775   \pxrr@trubyfalse
1776   \pxrr@ruby
1777 }
1778 \pxrr@add@protect\aruby
```

`\truby` 和文両側ルビの公開命令。

```
1779 \newcommand*\truby{%
1780   \pxrr@jprologue
1781   \pxrr@trubytrue
1782   \pxrr@ruby
1783 }
1784 \pxrr@add@protect\truby
```

`\atruby` 欧文両側ルビの公開命令。

```
1785 \newcommand*\atruby{%
1786   \pxrr@aprologue
1787   \pxrr@trubytrue
1788   \pxrr@ruby
1789 }
1790 \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。

```
1791 \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 1792 \let\pxrr@option\@empty
1793 \let\pxrr@exoption\@empty
```

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

```
\pxrr@do@scan 1794 \let\pxrr@do@proc\@empty
1795 \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。

```
1796 \def\pxrr@ruby{%
1797   \@testopt\pxrr@ruby@a}%
1798 }
1799 \def\pxrr@ruby@a[#1]{%
1800   \def\pxrr@option{#1}%
1801   \@testopt\pxrr@ruby@b}%
1802 }
1803 \def\pxrr@ruby@b[#1]{%
1804   \def\pxrr@exoption{#1}%
1805   \ifpxrr@truby
1806     \let\pxrr@do@proc\pxrr@truby@proc
1807     \let\pxrr@do@scan\pxrr@truby@scan
1808   \else
1809     \let\pxrr@do@proc\pxrr@ruby@proc
1810     \let\pxrr@do@scan\pxrr@ruby@scan
1811   \fi
1812   \pxrr@ruby@c
1813 }
1814 \def\pxrr@ruby@c{%
1815   \ifpxrr@ghost
1816     \expandafter\pxrr@do@proc
1817   \else
1818     \expandafter\pxrr@do@scan
1819   \fi
1820 }
```

`\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。

```
1821 \def\pxrr@mode@is@switching{%
1822   \if M\pxrr@mode T%
1823   \else\if J\pxrr@mode T%
1824   \else F%
1825   \fi\fi T%
1826 }
```

`\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。

```
1827 \def\pxrr@ruby@proc#1#2{%
1828   \pxrr@prepare@fallback{#1}%
1829   \pxrr@assign@fsize
```

オプションを解析する。

```
1830 \pxrr@parse@option\pxrr@option
```

ルビ文字入力をグループ列に分解する。

```
1831 \pxrr@decompbar{#2}%
```

```
1832 \let\pxrr@ruby@list\pxrr@res
```

```
1833 \edef\pxrr@ruby@count{\the\pxrr@cntr}%
```

```
1834 \let\pxrr@sruby@list\relax
```

親文字入力をグループ列に分解する。

```
1835 \pxrr@decompbar{#1}%
```

```
1836 \let\pxrr@body@list\pxrr@res
```

```
1837 \edef\pxrr@body@count{\the\pxrr@cntr}%
```

安全モードに関する処理を行う。

```
1838 \ifpxrr@safe@mode
```

```
1839 \pxrr@setup@safe@mode
```

```
1840 \fi
```

モードが“選択的”である場合、“普通の”モード ( $m \cdot j \cdot g$ ) に帰着させる。

```
1841 \if\pxrr@mode@is@switching
```

```
1842 \pxrr@resolve@mode
```

```
1843 \fi
```

```
1844 \ifpxrr@Debug
```

```
1845 \pxrr@debug@show@input
```

```
1846 \fi
```

入力検査を行い、パスした場合は組版処理に進む。

```
1847 \pxrr@if@alive{%
```

```
1848 \if g\pxrr@mode
```

```
1849 \pxrr@ruby@check@g
```

```
1850 \pxrr@if@alive{%
```

```
1851 \ifnum\pxrr@body@count>\@ne
```

```
1852 \pxrr@ruby@main@g
```

```
1853 \else
```

```
1854 \pxrr@ruby@main@g
```

```
1855 \fi
```

```
1856 }%
```

```
1857 \else
```

```
1858 \pxrr@ruby@check@m
```

```
1859 \pxrr@if@alive{\pxrr@ruby@main@m}%
```

```
1860 \fi
```

```
1861 }%
```

後処理を行う。

```
1862 \pxrr@ruby@exit
```

```
1863 }
```

`\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  
の手續の本体。

```
1864 \def\pxrr@truby@proc#1#2#3{%
```

```
1865 \pxrr@prepare@fallback{#1}%
```

フォントサイズの変数を設定して、

```
1866 \pxrr@assign@fsize
```

オプションを解析する。

```
1867 \pxrr@parse@option\pxrr@option
```

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

```
1868 \def\pxrr@all@input{#{1}-#{2}-#{3}}%
```

入力文字列のグループ分解を行う。

```
1869 \pxrr@decompbar{#3}%
```

```
1870 \let\pxrr@sruby@list\pxrr@res
```

```
1871 \edef\pxrr@sruby@count{\the\pxrr@cntr}%
```

```
1872 \pxrr@decompbar{#2}%
```

```
1873 \let\pxrr@ruby@list\pxrr@res
```

```
1874 \edef\pxrr@ruby@count{\the\pxrr@cntr}%
```

```
1875 \pxrr@decompbar{#1}%
```

```
1876 \let\pxrr@body@list\pxrr@res
```

```
1877 \edef\pxrr@body@count{\the\pxrr@cntr}%
```

安全モードに関する処理を行う。

```
1878 \ifpxrr@safe@mode
```

```
1879 \pxrr@setup@safe@mode
```

```
1880 \fi
```

```
1881 \if\pxrr@mode@is@switching
```

```
1882 \pxrr@resolve@mode
```

```
1883 \fi
```

```
1884 \ifpxrr@Debug
```

```
1885 \pxrr@debug@show@input
```

```
1886 \fi
```

入力検査を行い、パスした場合は組版処理に進む。

```
1887 \pxrr@if@alive{%
```

```
1888 \if g\pxrr@mode
```

```
1889 \pxrr@ruby@check@tg
```

```
1890 \pxrr@if@alive{\pxrr@ruby@main@tg}}%
```

```
1891 \else
```

```
1892 \pxrr@ruby@check@tm
```

```
1893 \pxrr@if@alive{\pxrr@ruby@main@tm}}%
```

```
1894 \fi
```

```
1895 }%
```

後処理を行う。

```
1896 \pxrr@ruby@exit
```

```
1897 }
```

`\pxrr@setup@safe@mode` 安全モード用の設定。

```
1898 \def\pxrr@setup@safe@mode{%
```

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

```

1899 \let\pxrr@mode=g\relax
1900 \pxrr@unite@group\pxrr@body@list
1901 \def\pxrr@body@count{1}%
1902 \pxrr@unite@group\pxrr@ruby@list
1903 \def\pxrr@ruby@count{1}%
1904 \ifx\pxrr@sruby@list\relax\else
1905   \pxrr@unite@group\pxrr@sruby@list
1906   \def\pxrr@sruby@count{1}%
1907 \fi

```

“文字単位のスキャン”が必要な機能を無効にする。

```

1908 \chardef\pxrr@evensp\z@
1909 \chardef\pxrr@revensp\z@
1910 \chardef\pxrr@fullsize\z@
1911 }

```

`\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。

```

1912 \def\pxrr@resolve@mode{%
1913   \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、

```

1914     \ifnum\pxrr@ruby@count=\@ne
1915       \let\pxrr@pre\pxrr@decompose
1916       \let\pxrr@post\relax
1917       \pxrr@body@list
1918     \ifnum\pxrr@cntr=\@ne\else
1919       \let\pxrr@mode=g%
1920     \fi
1921 \fi

```

それ以外はモノルビ・熟語ルビを選択する。

```

1922   \if M\pxrr@mode \let\pxrr@mode=m\fi
1923   \if J\pxrr@mode \let\pxrr@mode=j\fi
1924 \ifpxrr@Debug
1925   \pxrr@debug@show@resolve@mode
1926 \fi

```

`\pxrr@check@option`で行っている調整をやり直す。

```

1927   \if g\pxrr@mode
1928     \chardef\pxrr@athead\z@
1929   \fi
1930   \if g\pxrr@mode\else
1931     \chardef\pxrr@evensp\@ne
1932   \fi
1933 \else
1934   \pxrr@fatal@bad@switching
1935 \fi
1936 }

```

#### 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。

```
1937 \def\pxrr@ruby@check@g{%
1938   \ifnum\pxrr@body@count=\pxrr@ruby@count\relax
1939     \ifnum\pxrr@body@count=\@ne\else
1940       \ifpxrr@abody
1941         \pxrr@fatal@bad@movable
1942       \else\ifnum\pxrr@extra=\z@
1943         \pxrr@fatal@na@movable
1944       \fi\fi
1945     \fi
1946   \else
1947     \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
1948   \fi
1949 }
```

`\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。

```
1950 \def\pxrr@ruby@check@m{%
1951   \ifnum\pxrr@body@count=\@ne
1952     \let\pxrr@pre\pxrr@decompose
1953     \let\pxrr@post\relax
1954     \pxrr@body@list
1955     \let\pxrr@body@list\pxrr@res
1956     \edef\pxrr@body@count{\the\pxrr@cntr}%
1957     \ifnum\pxrr@body@count=\pxrr@ruby@count\relax\else
1958       \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
1959     \fi
1960   \else
1961     \pxrr@fatal@bad@mono
1962   \fi
1963 }
```

`\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。

```
1964 \def\pxrr@ruby@check@tg{%
1965   \ifnum\pxrr@body@count=\pxrr@ruby@count\relax\else
1966     \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
1967   \fi
1968   \ifnum\pxrr@body@count=\pxrr@sruby@count\relax\else
1969     \pxrr@fatal@bad@length\pxrr@body@count\pxrr@sruby@count
1970   \fi
```

```

1971 \pxrr@if@alive{%
1972   \ifnum\pxrr@body@count=\@ne\else
1973     \ifpxrr@abody
1974       \pxrr@fatal@bad@movable
1975     \else\ifnum\pxrr@extra=\z@
1976       \pxrr@fatal@na@movable
1977     \fi\fi
1978   \fi
1979 }%
1980 }

```

`\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。

```

1981 \def\pxrr@ruby@check@tm{%
1982   \ifnum\pxrr@body@count=\@ne
1983     \let\pxrr@pre\pxrr@decompose
1984     \let\pxrr@post\relax
1985     \pxrr@body@list
1986     \let\pxrr@body@list\pxrr@res
1987     \edef\pxrr@body@count{\the\pxrr@cntr}%
1988     \ifnum\pxrr@body@count=\pxrr@ruby@count\relax\else
1989       \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
1990     \fi
1991     \ifnum\pxrr@body@count=\pxrr@sruby@count\relax\else
1992       \pxrr@fatal@bad@length\pxrr@body@count\pxrr@sruby@count
1993     \fi
1994   \else
1995     \pxrr@fatal@bad@mono
1996   \fi
1997 }

```

#### 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。

```
1998 \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。

```

1999 \def\pxrr@check@par@head{%
2000   \ifvmode
2001     \pxrr@par@headtrue
2002   \else
2003     \pxrr@par@headfalse
2004   \fi
2005 }

```

`\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@preinterpret` の本体の末尾でなければならない。

```

2006 \def\pxrr@if@last#1#2#3{%
2007   \ifx#3\pxrr@post #1%
2008   \else #2%
2009   \fi
2010   #3%
2011 }

```

`\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。

```

2012 \def\pxrr@inter@mono{%
2013   \hskip\pxrr@iiskip\relax
2014 }

```

`\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。

```

2015 \def\pxrr@takeout@any@protr@nocolor{%
2016   \ifpxrr@any@protr
2017     \aftergroup\pxrr@any@protrtrue
2018   \fi
2019 }

```

color 使用時の定義。

```

2020 \def\pxrr@takeout@any@protr{%
2021   \ifpxrr@any@protr
2022     \aftergroup\pxrr@takeout@any@protr@a
2023   \fi
2024 }
2025 \def\pxrr@takeout@any@protr@a{%
2026   \aftergroup\pxrr@any@protrtrue
2027 }

```

`\pxrr@ruby@main@m` モノルビ。

```

2028 \def\pxrr@ruby@main@m{%
2029   \pxrr@zip@list\pxrr@body@list\pxrr@ruby@list
2030   \let\pxrr@whole@list\pxrr@res
2031   \pxrr@check@par@head
2032   \pxrr@put@head@penalty
2033   \pxrr@any@protrfalse
2034   \ifpxrr@Debug
2035   \pxrr@debug@show@recomp
2036   \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。

```

2037   \let\pxrr@locate@head@\pxrr@locate@inner
2038   \let\pxrr@locate@end@\pxrr@locate@inner

```

```

2039 \let\pxrr@locate@sing@\pxrr@locate@inner
2040 \ifpxrr@aprotr\else
2041   \let\pxrr@locate@end@\pxrr@locate@end
2042   \let\pxrr@locate@sing@\pxrr@locate@end
2043 \fi
2044 \ifpxrr@bprotr\else
2045   \let\pxrr@locate@head@\pxrr@locate@head
2046   \let\pxrr@locate@sing@\pxrr@locate@head
2047 \fi
2048 \def\pxrr@pre##1##2{%
2049   \pxrr@if@last{%

```

単独ブロックの場合。

```

2050   \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
2051   \pxrr@intrude@head
2052   \unhbox\pxrr@boxr
2053   \pxrr@intrude@end
2054   \pxrr@takeout@any@protr
2055   }{%

```

先頭ブロックの場合。

```

2056   \pxrr@compose@block\pxrr@locate@head@{##1}{##2}%
2057   \pxrr@intrude@head
2058   \unhbox\pxrr@boxr
2059   }%
2060 }%
2061 \def\pxrr@inter##1##2{%
2062   \pxrr@if@last{%

```

末尾ブロックの場合。

```

2063   \pxrr@compose@block\pxrr@locate@end@{##1}{##2}%
2064   \pxrr@inter@mono
2065   \unhbox\pxrr@boxr
2066   \pxrr@intrude@end
2067   \pxrr@takeout@any@protr
2068   }{%

```

中間ブロックの場合。

```

2069   \pxrr@compose@block\pxrr@locate@inner{##1}{##2}%
2070   \pxrr@inter@mono
2071   \unhbox\pxrr@boxr
2072   }%
2073 }%
2074 \let\pxrr@post\@empty
2075 \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。

```

2076 \if j\pxrr@mode
2077   \ifpxrr@any@protr
2078     \pxrr@ruby@redo@j
2079   \fi

```

```

2080 \fi
2081 \unhbox\pxrr@boxr
2082 }

```

\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。

```

2083 \def\pxrr@ruby@redo@j{%
2084 \pxrr@concat@list\pxrr@body@list
2085 \let\pxrr@body@list\pxrr@res
2086 \pxrr@concat@list\pxrr@ruby@list
2087 \let\pxrr@ruby@list\pxrr@res
2088 \pxrr@zip@single\pxrr@body@list\pxrr@ruby@list
2089 \let\pxrr@whole@list\pxrr@res
2090 \ifpxrrDebug
2091 \pxrr@debug@show@concat
2092 \fi
2093 \let\pxrr@locate@sing@\pxrr@locate@inner
2094 \ifpxrr@aprotr\else
2095 \let\pxrr@locate@sing@\pxrr@locate@end
2096 \fi
2097 \ifpxrr@bprotr\else
2098 \let\pxrr@locate@sing@\pxrr@locate@head
2099 \fi
2100 \def\pxrr@pre##1##2{%
2101 \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
2102 \pxrr@intrude@head
2103 \unhbox\pxrr@boxr
2104 \pxrr@intrude@end
2105 }%
2106 \let\pxrr@inter\@undefined
2107 \let\pxrr@post\@empty
2108 \setbox\pxrr@boxr\pxrr@hbox{\pxrr@whole@list}%
2109 }

```

\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。

```

2110 \def\pxrr@ruby@main@g{%
2111 \pxrr@zip@list\pxrr@body@list\pxrr@ruby@list
2112 \let\pxrr@whole@list\pxrr@res
2113 \pxrr@check@par@head
2114 \pxrr@put@head@penalty
2115 \ifpxrrDebug
2116 \pxrr@debug@show@recomp
2117 \fi
2118 \let\pxrr@locate@sing@\pxrr@locate@inner
2119 \ifpxrr@aprotr\else
2120 \let\pxrr@locate@sing@\pxrr@locate@end

```

```

2121 \fi
2122 \ifpxrr@bprotr\else
2123 \let\pxrr@locate@sing@\pxrr@locate@head
2124 \fi
2125 \def\pxrr@pre##1##2{%
2126 \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
2127 \pxrr@intrude@head
2128 \unhbox\pxrr@boxr
2129 \pxrr@intrude@end
2130 }%
2131 \let\pxrr@inter\@undefined
2132 \let\pxrr@post\@empty

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

2133 \pxrr@whole@list
2134 }

```

\pxrr@ruby@main@tm 両側のモノルビの場合。

```

2135 \def\pxrr@ruby@main@tm{%
2136 \pxrr@tzip@list\pxrr@body@list\pxrr@ruby@list\pxrr@sruby@list
2137 \let\pxrr@whole@list\pxrr@res
2138 \pxrr@check@par@head
2139 \pxrr@any@protrfalse
2140 \ifpxrrDebug
2141 \pxrr@debug@show@recomp
2142 \fi
2143 \let\pxrr@locate@head@\pxrr@locate@inner
2144 \let\pxrr@locate@end@\pxrr@locate@inner
2145 \let\pxrr@locate@sing@\pxrr@locate@inner
2146 \ifpxrr@aprotr\else
2147 \let\pxrr@locate@end@\pxrr@locate@end
2148 \let\pxrr@locate@sing@\pxrr@locate@end
2149 \fi
2150 \ifpxrr@bprotr\else
2151 \let\pxrr@locate@head@\pxrr@locate@head
2152 \let\pxrr@locate@sing@\pxrr@locate@head
2153 \fi
2154 \def\pxrr@pre##1##2##3{%
2155 \pxrr@if@last{%
2156 \pxrr@compose@twoside@block\pxrr@locate@sing@
2157 {##1}{##2}{##3}%
2158 \pxrr@intrude@head
2159 \unhbox\pxrr@boxr
2160 \pxrr@intrude@end
2161 \pxrr@takeout@any@protr
2162 }{%
2163 \pxrr@compose@twoside@block\pxrr@locate@head@
2164 {##1}{##2}{##3}%
2165 \pxrr@intrude@head

```

```

2166     \unhbox\pxrr@boxr
2167   }%
2168 }%
2169 \def\pxrr@inter##1##2##3{%
2170   \pxrr@if@last{%
2171     \pxrr@compose@twoside@block\pxrr@locate@end@
2172     {##1}{##2}{##3}%
2173   \pxrr@inter@mono
2174   \unhbox\pxrr@boxr
2175   \pxrr@intrude@end
2176   \pxrr@takeout@any@protr
2177   }{%
2178     \pxrr@compose@twoside@block\pxrr@locate@inner
2179     {##1}{##2}{##3}%
2180     \pxrr@inter@mono
2181     \unhbox\pxrr@boxr
2182   }%
2183 }%
2184 \let\pxrr@post\@empty
2185 \setbox\pxrr@boxr\pxrr@hbox{\pxrr@whole@list}%
2186 \unhbox\pxrr@boxr
2187 }

```

`\pxrr@ruby@main@tg` 両側の単純グループビの場合。

```

2188 \def\pxrr@ruby@main@tg{%
2189   \pxrr@check@par@head
2190   \pxrr@put@head@penalty
2191   \let\pxrr@locate@sing@\pxrr@locate@inner
2192   \ifpxrr@aprotr\else
2193     \let\pxrr@locate@sing@\pxrr@locate@end
2194   \fi
2195   \ifpxrr@bprotr\else
2196     \let\pxrr@locate@sing@\pxrr@locate@head
2197   \fi
2198   \expandafter\pxrr@compose@twoside@block\expandafter\pxrr@locate@sing@
2199   \pxrr@all@input
2200   \pxrr@intrude@head
2201   \unhbox\pxrr@boxr
2202   \pxrr@intrude@end
2203 }

```

`\pxrr@ruby@main@mg` 未実装（呼出もない）。

```

2204 \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。

```

2205 \newif\ifpxrr@ghost

\pxrr@zspace 全角空白文字。文字そのものをファイルに含ませたくないなので chardef にする。
2206 \pxrr@jchardef\pxrr@zspace=\pxrr@jc{2121:3000}

\pxrr@jprologue 和文ルビ用の開始処理。
2207 \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。
2208     \ifpxrr@jghost
2209     \pxrr@zspace
2210     \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。
2211     \begingroup
2212     \pxrr@abodyfalse
2213     \pxrr@csletcs{ifpxrr@ghost}{ifpxrr@jghost}%

    出力した全角空白の幅だけ戻しておく。
2214     \ifpxrr@jghost
2215     \setbox\pxrr@boxa\hbox{\pxrr@zspace}%
2216     \kern-\wd\pxrr@boxa
2217     \fi
2218 }

\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。）
2219 \chardef\pxrr@aghostchar=23 % compwordmark
2220 \let\pxrr@aghost\relax
2221 \let\pxrr@aghostfont\relax
2222 \def\pxrr@setup@aghost{%
2223     \global\let\pxrr@setup@aghost\relax
2224     \IfFileExists{t1lmr.fd}{%
2225         \begingroup
2226         \fontsize{2.5}{0}\usefont{T1}{lmr}{m}{n}%
2227         \endgroup
2228         \global\pxrr@letcs\pxrr@aghostfont{T1/lmr/m/n/2.5}%
2229         \gdef\pxrr@aghost{{\pxrr@aghostfont\pxrr@aghostchar}}%
2230         \global\xspacecode\pxrr@aghostchar=3 %
2231     }{%else
2232         \pxrr@warn{Ghost embedding for \string\aruby\space
2233             is disabled,\MessageBreak
2234             since package lmodern is missing}%
2235         \global\pxrr@aghostfalse

```

```

2236 \global\let\pxrr@aghosttrue\relax
2237 }%
2238 }

```

`\pxrr@aprologue` 欧文ルビ用の開始処理。

```

2239 \def\pxrr@aprologue{%
2240 \ifpxrr@aghost
2241 \pxrr@aghost
2242 \fi
2243 \begingroup
2244 \pxrr@abodytrue
2245 \pxrr@csletcs{ifpxrr@ghost}{ifpxrr@aghost}%
2246 }

```

#### 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。

```

2247 \def\pxrr@ruby@exit{%
2248 \ifpxrr@fatal@error
2249 \pxrr@fallback
2250 \fi
2251 \ifpxrr@abody
2252 \expandafter\pxrr@aepilogue
2253 \else
2254 \expandafter\pxrr@jepilogue
2255 \fi
2256 }

```

`\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。

```

2257 \def\pxrr@jepilogue{%
2258 \ifpxrr@jghost
2259 \setbox\pxrr@boxa\hbox{\pxrr@zspace}%
2260 \kern-\wd\pxrr@boxa
2261 \fi

```

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

```

2262 \endgroup
2263 \ifpxrr@jghost
2264 \pxrr@zspace
2265 \fi
2266 }

```

`\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。

```

2267 \def\pxrr@aepilogue{%
2268 \endgroup
2269 \ifpxrr@aghost

```

```

2270 \pxrr@aghost
2271 \fi
2272 }

```

#### 4.19 デバッグ用出力

```

2273 \def\pxrr@debug@show@input{%
2274 \typeout{----\pxrr@pkgname\space input:^^J%
2275 ifpxrr@abody = \meaning\ifpxrr@abody^^J%
2276 ifpxrr@truby = \meaning\ifpxrr@truby^^J%
2277 pxrr@ruby@fsize = \pxrr@ruby@fsize^^J%
2278 pxrr@body@zw = \pxrr@body@zw^^J%
2279 pxrr@ruby@zw = \pxrr@ruby@zw^^J%
2280 pxrr@iiskip = \pxrr@iiskip^^J%
2281 pxrr@iaiskip = \pxrr@iaiskip^^J%
2282 pxrr@htratio = \pxrr@htratio^^J%
2283 pxrr@ruby@raise = \pxrr@ruby@raise^^J%
2284 pxrr@ruby@lower = \pxrr@ruby@lower^^J%
2285 ifpxrr@bprotr = \meaning\ifpxrr@bprotr^^J%
2286 ifpxrr@aprotr = \meaning\ifpxrr@aprotr^^J%
2287 pxrr@side = \the\pxrr@side^^J%
2288 pxrr@evensp = \the\pxrr@evensp^^J%
2289 pxrr@fullsize = \the\pxrr@fullsize^^J%
2290 pxrr@bscomp = \meaning\pxrr@bscomp^^J%
2291 pxrr@ascomp = \meaning\pxrr@ascomp^^J%
2292 ifpxrr@bnoabr = \meaning\ifpxrr@bnoabr^^J%
2293 ifpxrr@anoabr = \meaning\ifpxrr@anoabr^^J%
2294 ifpxrr@bfintr = \meaning\ifpxrr@bfintr^^J%
2295 ifpxrr@afintr = \meaning\ifpxrr@afintr^^J%
2296 pxrr@bintr = \pxrr@bintr^^J%
2297 pxrr@aintr = \pxrr@aintr^^J%
2298 pxrr@athead = \the\pxrr@athead^^J%
2299 pxrr@mode = \meaning\pxrr@mode^^J%
2300 ifpxrr@athead@given = \meaning\ifpxrr@athead@given^^J%
2301 ifpxrr@mode@given = \meaning\ifpxrr@mode@given^^J%
2302 pxrr@body@list = \meaning\pxrr@body@list^^J%
2303 pxrr@body@count = \@nameuse{pxrr@body@count}^^J%
2304 pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
2305 pxrr@ruby@count = \@nameuse{pxrr@ruby@count}^^J%
2306 pxrr@end@kinsoku = \pxrr@end@kinsoku^^J%
2307 ----
2308 }%
2309 }
2310 \def\pxrr@debug@show@recomp{%
2311 \typeout{----\pxrr@pkgname\space recomp:^^J%
2312 pxrr@body@list = \meaning\pxrr@body@list^^J%
2313 pxrr@body@count = \pxrr@body@count^^J%
2314 pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%

```

```

2315     pxrr@ruby@count = \pxrr@ruby@count^^J%
2316     pxrr@res = \meaning\pxrr@res^^J%
2317     ----
2318   }%
2319 }
2320 \def\pxrr@debug@show@concat{%
2321   \typeout{----\pxrr@pkgname\space concat:^^J%
2322     pxrr@body@list = \meaning\pxrr@body@list^^J%
2323     pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
2324     pxrr@whole@list = \meaning\pxrr@whole@list^^J%
2325     ----
2326   }%
2327 }
2328 \def\pxrr@debug@show@resolve@mode{%
2329   \typeout{----\pxrr@pkgname\space resolve-mode:
2330     \meaning\pxrr@mode}%
2331 }

```