

pxrubrica パッケージ

八登 崇之 (Takayuki YATO; aka “ZR”)

v1.1 [2017/04/10]

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	3
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	6
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	9
4.3.1	全般設定	9
4.3.2	ルビ呼出時の設定	11
4.4	補助手続	12
4.4.1	雑多な定義	12
4.4.2	数値計算	15
4.4.3	リスト分解	16
4.5	エンジン依存処理	20
4.6	パラメタ設定公開命令	26
4.7	ルビオプション解析	28
4.8	オプション整合性検査	34
4.9	フォントサイズ	36
4.10	ルビ用均等割り	38
4.11	小書き仮名の変換	41

4.12	ブロック毎の組版	42
4.13	命令の頑強化	47
4.14	致命的エラー対策	47
4.15	先読み処理	48
4.16	進入処理	50
4.16.1	前側進入処理	51
4.16.2	後側進入処理	52
4.17	メインです	54
4.17.1	エントリーポイント	54
4.17.2	入力検査	57
4.17.3	ルビ組版処理	58
4.17.4	前処理	62
4.17.5	後処理	63
4.18	デバッグ用出力	64

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。

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

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

– `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（あいうえおっや

ゆよわ、およびその片仮名)を対応の非小書き仮名に変換する。F指定はこの機能を無効にする。

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

(後進入設定)は以下の値。

|| 後突出禁止 > 後進入大
| 後進入無し) 後進入小

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

```
じゆく|ご
```

は 2 つのグループからなり、最初のもは 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_TE_X エンジンの仕様上の制約により、ルビ出力の進入と共存できない。（従って共存するような設定を試みるとエラーになる。）

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

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

2.5 パラメタ設定命令

基本的設定。

- `\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。
- `\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。

- `\rubysizeratio`{(実数)}
ルビサイズの親文字サイズに対する割合。[既定 = 0.5]
- `\rubystretchprop`{(X)}{(Y)}{(Z)}
ルビ用均等割りの比率の指定。[既定 = 1, 2, 1]
- `\rubystretchprophead`{(Y)}{(Z)}
前突出禁止時の均等割りの比率の指定。[既定 = 1, 1]
- `\rubystretchpropend`{(X)}{(Y)}
後突出禁止時の均等割りの比率の指定。[既定 = 1, 1]
- `\rubyyheightratio`{(実数)}
横組和文の高さの縦幅に対する割合。[既定 = 0.88]
- `\rubytheightratio`{(実数)}
縦組和文の「高さ」の「縦幅」に対する割合 (pTeX の縦組では「縦」と「横」が実際の逆になる)。[既定 = 0.5]

3 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、TeX での実現は非常に難しい。これを ε -pTeX 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: 拡張機能を無効にする。
 - 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{pxrurica}
3 \def\pxrr@error{%
4   \PackageError\pxrr@pkgname
5 }
6 \def\pxrr@warn{%
7   \PackageWarning\pxrr@pkgname
8 }
```

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

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

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

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

`\pxrr@eh@fatal` 致命的エラーのヘルプ。

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

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

`\pxrr@warn@must@group` 欧文ルビ、あるいは両側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@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)。

```

36 \def\pxrr@fatal@bad@intr{%
37   \pxrr@fatal@error{%
38     Intrusion disallowed when ghost is enabled%
39   }\pxrr@eh@fatal
40 }

```

`\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)。

```

41 \def\pxrr@fatal@bad@no@protr{%
42   \pxrr@fatal@error{%
43     Protrusion must be allowed for either end%
44   }\pxrr@eh@fatal
45 }

```

`\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。)

```

46 \def\pxrr@fatal@bad@length#1#2{%
47   \pxrr@fatal@error{%
48     Group count mismatch between the ruby and\MessageBreak

```

```

49   the body (#1 <> #2)%
50 } \pxrr@eh@fatal
51 }

```

`\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。

```

52 \def \pxrr@fatal@bad@mono{%
53   \pxrr@fatal@error{%
54     Mono-ruby must have a single group%
55   } \pxrr@eh@fatal
56 }

```

`\pxrr@fatal@bad@morable` 欧文ルビまたは両側ルビ（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。

```

57 \def \pxrr@fatal@bad@morable{%
58   \pxrr@fatal@error{%
59     Novable group ruby is not allowed here%
60   } \pxrr@eh@fatal
61 }

```

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

```

62 \def \pxrr@fatal@na@morable{%
63   \pxrr@fatal@error{%
64     Feature of movable group ruby is disabled%
65   } \pxrr@eh@fatal
66 }

```

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

```

67 \def \pxrr@warn@load@order#1{%
68   \pxrr@warn{%
69     This package should be loaded after '#1'%
70   }%
71 }

```

`\pxrr@interror` 内部エラー。これが出てはいけない。:-)

```

72 \def \pxrr@interror#1{%
73   \pxrr@fatal@error{INTERNAL ERROR (#1)}%
74   \pxrr@eh@fatal
75 }

```

`\ifpxrrDebug` デバッグモード指定。

```

76 \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。

```

77 \let\pxrr@ruby@font\@empty

\pxrr@big@intr 「大」と「小」の進入量 (\rubybigintrusion/\rubysmallintrusion)。実数値マクロ (数
\pxrr@small@intr 字列に展開される)。
78 \def\pxrr@big@intr{1}
79 \def\pxrr@small@intr{0.5}

\pxrr@size@ratio ルビ文字サイズ (\rubysizeratio)。実数値マクロ。
80 \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 81 \def\pxrr@sprop@x{1}
82 \def\pxrr@sprop@y{2}
\pxrr@sprop@z 83 \def\pxrr@sprop@z{1}

\pxrr@sprop@hy 伸縮配置比率 (\rubystretchprophead)。実数値マクロ。
\pxrr@sprop@hz 84 \def\pxrr@sprop@hy{1}
85 \def\pxrr@sprop@hz{1}

\pxrr@sprop@ex 伸縮配置比率 (\rubystretchpropend)。実数値マクロ。
\pxrr@sprop@ey 86 \def\pxrr@sprop@ex{1}
87 \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。
88 \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。
89 \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。
90 \def\pxrr@thtratio{0.5}

\pxrr@extra 拡張機能実装方法 (\rubyuseextra)。整数定数。
91 \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。
92 \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。
93 \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。
94 \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。
95 \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。
96 \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。
`97 \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` `98 \newif\ifpxrr@d@bprotr \pxrr@d@bprotrtrue`
`99 \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（進入無し）に展開されるマクロ。
`\pxrr@d@aintr` `100 \def\pxrr@d@bintr{}`
`101 \def\pxrr@d@aintr{}`

`\pxrr@d@athead` 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。
`102 \chardef\pxrr@d@athead=0`

`\pxrr@d@mode` モノルビ (`m`)・グループルビ (`g`)・熟語ルビ (`j`) のいずれか。`\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。
`103 \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。
`104 \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。
`105 \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。
`106 \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` `107 \newif\ifpxrr@bprotr \pxrr@bprotrfalse`
`108 \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` `109 \def\pxrr@bintr{}`
`110 \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。
`111 \let\pxrr@bscomp\relax`
`112 \let\pxrr@ascomp\relax`

`\ifpxrr@bnoobr` ルビ付文字の直前／直後で行分割を許すか。`\ruby` の `*` 指定に由来する。スイッチ。
`\ifpxrr@anoobr` ※ 既定値設定 (`\rubyssetup`) でこれに対応するものはない。
`113 \newif\ifpxrr@bnoobr \pxrr@bnoobrfalse`
`114 \newif\ifpxrr@anoobr \pxrr@anoobrfalse`

`\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。

```
115 \newif\ifpxrr@bfintr \pxrr@bfintrfalse
116 \newif\ifpxrr@afintr \pxrr@afintrfalse
```

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

```
117 \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。

```
118 \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。

```
119 \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。

```
120 \newif\ifpxrr@mode@given \pxrr@mode@givenfalse
121 \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。

```
122 \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。

```
123 \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。

```
124 \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。

```
125 \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。

```
126 \chardef\pxrr@fullsize=1
```

4.4 補助手続

4.4.1 雑多な定義

`\ifpxrr@ok` 汎用スイッチ。

```
127 \newif\ifpxrr@ok
```

`\pxrr@canta` 汎用の整数レジスタ。

```
128 \newcount\pxrr@canta
```

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

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

`\pxrr@boxa` 汎用のボックスレジスタ。
`\pxrr@boxb` 131 `\newbox\pxrr@boxa`
132 `\newbox\pxrr@boxb`

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

`\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。
134 `\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。
135 `\def\pxrr@zeropt{Opt}`

`\pxrr@hfilx` `\pxrr@hfilx{<実数>}` : 「<実数>fil」のグルーを置く。
136 `\def\pxrr@hfilx#1{%`
137 `\hskip\z@\@plus #1fil\relax`
138 `}`

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

`\pxrr@ifx` `\pxrr@ifx{<引数>}<真>}<偽>}` : `\ifx<引数>` を行うテスト。
140 `\def\pxrr@ifx#1{%`
141 `\ifx#1\expandafter\@firstoftwo`
142 `\else\expandafter\@secondoftwo`
143 `\fi`
144 `}`

`\pxrr@ifnum` `\pxrr@ifnum{<引数>}<真>}<偽>}` : `\ifnum<引数>` を行うテスト。
145 `\def\pxrr@ifnum#1{%`
146 `\ifnum#1\expandafter\@firstoftwo`
147 `\else\expandafter\@secondoftwo`
148 `\fi`
149 `}`

`\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。
150 `\def\pxrr@cslet#1{%`
151 `\expandafter\let\csname#1\endcsname`
152 `}`
153 `\def\pxrr@letcs#1#2{%`

```

154 \expandafter\let\expandafter#1\csname#2\endcsname
155 }
156 \def\pxrr@csletcs#1#2{%
157 \expandafter\let\csname#1\expandafter\endcsname
158 \csname#2\endcsname
159 }

```

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

```

160 \def\pxrr@setok#1{%
161 #1{\pxrr@oktrue}{\pxrr@okfalse}%
162 }

```

`\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。

```

163 \def\pxrr@appto#1#2{%
164 \expandafter\def\expandafter#1\expandafter{#1#2}%
165 }

```

`\pxrr@nil` ユニークトークン。

```

\pxrr@end 166 \def\pxrr@nil{\noexpand\pxrr@nil}
167 \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。

```

168 \def\pxrr@without@macro@trace#1{%
169 \chardef\pxrr@tracingmacros=\tracingmacros
170 \tracingmacros\z@
171 #1%
172 \tracingmacros\pxrr@tracingmacros
173 }

```

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

```

\pxrr@hbox@to 174 \def\pxrr@hbox#1{%
175 \hbox{%
176 \color@begingroup
177 #1%
178 \color@endgroup
179 }%
180 }
181 \def\pxrr@hbox@to#1#2{%
182 \pxrr@hbox@to@a{#1}%
183 }
184 \def\pxrr@hbox@to@a#1#2{%
185 \hbox to#1{%
186 \color@begingroup
187 #2%
188 \color@endgroup
189 }%
190 }

```

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。

```

191 \AtBeginDocument{%
192   \ifx\color@begingroup\relax
193     \ifx\color@endgroup\relax
194       \let\pxrr@hbox\hbox
195       \let\pxrr@hbox@to\hb@xt@
196       \let\pxrr@takeout@any@protr\pxrr@takeout@any@protr@nocolor
197     \fi
198   \fi
199 }
```

4.4.2 数値計算

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

```

200 \mathchardef\pxrr@invscale@ca=259
201 \def\pxrr@invscale#1#2{%
202   \begingroup
203     \@tempdima=#1\relax
204     \@tempdimb#2\p@\relax
205     \@tempcnta\@tempdima
206     \multiply\@tempcnta\@cclvi
207     \divide\@tempcnta\@tempdimb
208     \multiply\@tempcnta\@cclvi
209     \@tempcntb\p@
210     \divide\@tempcntb\@tempdimb
211     \advance\@tempcnta-\@tempcntb
212     \advance\@tempcnta-\tw@
213     \@tempdimb\@tempcnta\@ne
214     \advance\@tempcnta\@tempcntb
215     \advance\@tempcnta\@tempcntb
216     \advance\@tempcnta\pxrr@invscale@ca
217     \@tempdimc\@tempcnta\@ne
218     \@whiledim\@tempdimb<\@tempdimc\do{%
219       \@tempcntb\@tempdimb
220       \advance\@tempcntb\@tempdimc
221       \advance\@tempcntb\@ne
222       \divide\@tempcntb\tw@
223       \ifdim #2\@tempcntb>\@tempdima
224         \advance\@tempcntb\m@ne
225         \@tempdimc=\@tempcntb\@ne
226       \else
227         \@tempdimb=\@tempcntb\@ne
228       \fi}%
229     \xdef\pxrr@gtmpa{\the\@tempdimb}%
230   \endgroup
231   #1=\pxrr@gtmpa\relax
```

232 }

`\pxrr@interpolate` `\pxrr@interpolate{⟨入力単位⟩}{⟨出力単位⟩}{⟨寸法レジスタ⟩}{(X1,Y1)(X2,Y2)⋯(Xn,Yn)}`: 線形補間を行う。すなわち、明示値

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

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

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

```
233 \def\pxrr@interpolate#1#2#3#4#5{%
234   \edef\pxrr@tempa{#1}%
235   \edef\pxrr@tempb{#2}%
236   \def\pxrr@tempd{#3}%
237   \setlength{\@tempdima}{#4}%
238   \edef\pxrr@tempc{(0,0)#5(*,*)}%
239   \expandafter\pxrr@interpolate@a\pxrr@tempc\@nil
240 }
241 \def\pxrr@interpolate@a(#1,#2)(#3,#4)(#5,#6){%
242   \if*#5%
243     \def\pxrr@tempc{\pxrr@interpolate@b{#1}{#2}{#3}{#4}}%
244   \else\ifdim\@tempdima<#3\pxrr@tempa
245     \def\pxrr@tempc{\pxrr@interpolate@b{#1}{#2}{#3}{#4}}%
246   \else
247     \def\pxrr@tempc{\pxrr@interpolate@a(#3,#4)(#5,#6)}%
248   \fi\fi
249   \pxrr@tempc
250 }
251 \def\pxrr@interpolate@b#1#2#3#4#5\@nil{%
252   \@tempdimb=-#1\pxrr@tempa
253   \advance\@tempdima\@tempdimb
254   \advance\@tempdimb#3\pxrr@tempa
255   \edef\pxrr@tempc{\strip@pt\@tempdimb}%
256   \pxrr@invscale\@tempdima\pxrr@tempc
257   \edef\pxrr@tempc{\strip@pt\@tempdima}%
258   \@tempdima=#4\pxrr@tempb
259   \@tempdimb=#2\pxrr@tempb
260   \advance\@tempdima-\@tempdimb
261   \@tempdima=\pxrr@tempc\@tempdima
262   \advance\@tempdima\@tempdimb
263   \pxrr@tempd=\@tempdima
264 }
```

4.4.3 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 ($\{\dots\}$ 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）。

```

265 \def\pxrr@decompose#1{%
266   \let\pxrr@res\@empty
267   \pxrr@cntr=\z@
268   \pxrr@decompose@loopa#1\pxrr@end
269 }
270 \def\pxrr@decompose@loopa{%
271   \futurelet\pxrr@tempa\pxrr@decompose@loopb
272 }
273 \def\pxrr@decompose@loopb{%
274   \pxrr@ifx{\pxrr@tempa\pxrr@end}{%
275     \pxrr@appto\pxrr@res{\pxrr@post}%
276   }{%
277     \pxrr@setok{\pxrr@ifx{\pxrr@tempa\bgroup}}%
278     \pxrr@decompose@loopc
279   }%
280 }
281 \def\pxrr@decompose@loopc#1{%
282   \ifx\pxrr@res\@empty
283     \def\pxrr@res{\pxrr@pre}%
284   \else
285     \pxrr@appto\pxrr@res{\pxrr@inter}%
286   \fi
287   \ifpxrr@ok
288     \pxrr@appto\pxrr@res{{#1}}%
289   \else
290     \pxrr@appto\pxrr@res{#1}%
291   \fi
292   \advance\pxrr@cntr\@ne
293   \pxrr@decompose@loopa
294 }

```

`\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。

```

295 \def\pxrr@decompbar#1{%
296   \let\pxrr@res\@empty
297   \pxrr@cntr=\z@
298   \pxrr@decompbar@loopa\pxrr@nil#1|\pxrr@end|%
299 }
300 \def\pxrr@decompbar@loopa#1{%
301   \expandafter\pxrr@decompbar@loopb\expandafter{\@gobble#1}%
302 }
303 \def\pxrr@decompbar@loopb#1{%

```

```

304 \pxrr@decompbar@loopc#1\relax\pxrr@nil{#1}%
305 }
306 \def\pxrr@decompbar@loopc#1#2\pxrr@nil#3{%
307 \pxrr@ifx{#1\pxrr@end}{%
308 \pxrr@appto\pxrr@res{\pxrr@post}%
309 }{%
310 \ifx\pxrr@res\@empty
311 \def\pxrr@res{\pxrr@pre}%
312 \else
313 \pxrr@appto\pxrr@res{\pxrr@inter}%
314 \fi
315 \pxrr@appto\pxrr@res{#{3}}%
316 \advance\pxrr@cntr\@ne
317 \pxrr@decompbar@loopa\pxrr@nil
318 }%
319 }

```

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

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

この命令は \pxrr@res を以下の内容に定義する。

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

```

320 \def\pxrr@zip@list#1#2{%
321 \let\pxrr@res\@empty
322 \let\pxrr@post\relax
323 \let\pxrr@tempa#1\pxrr@appto\pxrr@tempa{}}%
324 \let\pxrr@tempb#2\pxrr@appto\pxrr@tempb{}}%
325 \pxrr@zip@list@loopa
326 }
327 \def\pxrr@zip@list@loopa{%
328 \expandafter\pxrr@zip@list@loopb\pxrr@tempa\pxrr@end
329 }
330 \def\pxrr@zip@list@loopb#1#2#3\pxrr@end{%
331 \pxrr@ifx{#1\relax}{%
332 \pxrr@zip@list@exit
333 }{%
334 \pxrr@appto\pxrr@res{#1{#2}}%
335 \def\pxrr@tempa{#3}%
336 \expandafter\pxrr@zip@list@loopc\pxrr@tempb\pxrr@end
337 }%
338 }
339 \def\pxrr@zip@list@loopc#1#2#3\pxrr@end{%
340 \pxrr@ifx{#1\relax}{%
341 \pxrr@interror{zip}%
342 \pxrr@appto\pxrr@res{}}%

```

```

343 \pxrr@zip@list@exit
344 }{%
345 \pxrr@appto\pxrr@res{#{#2}}}%
346 \def\pxrr@tempb{#3}%
347 \pxrr@zip@list@loopa
348 }%
349 }
350 \def\pxrr@zip@list@exit{%
351 \pxrr@appto\pxrr@res{\pxrr@post}}%
352 }

```

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

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

の時に、`\pxrr@res` を以下の内容に定義する。

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

```

353 \def\pxrr@concat@list#1{%
354 \let\pxrr@res\@empty
355 \def\pxrr@pre##1{%
356 \pxrr@appto\pxrr@res{##1}}%
357 }%
358 \let\pxrr@inter\pxrr@pre
359 \let\pxrr@post\relax
360 #1%
361 }

```

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

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

の時に、`\pxrr@res` を以下の内容に定義する。

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

```

362 \def\pxrr@zip@single#1#2{%
363 \expandafter\pxrr@zip@single@a\expandafter#1#2\pxrr@end
364 }
365 \def\pxrr@zip@single@a#1{%
366 \expandafter\pxrr@zip@single@b#1\pxrr@end
367 }
368 \def\pxrr@zip@single@b#1\pxrr@end#2\pxrr@end{%
369 \def\pxrr@res{\pxrr@pre{#1}{#2}\pxrr@post}}%
370 }

```

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

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

の時に、`\pxrr@res` を以下の内容に定義する。

```

\pxrr@pre{<X>}{<Y>}{<Z>}\pxrr@post
371 \def\pxrr@tzip@single#1#2#3{%
372   \expandafter\pxrr@tzip@single@a\expandafter#1\expandafter#2#3\pxrr@end
373 }
374 \def\pxrr@tzip@single@a#1#2{%
375   \expandafter\pxrr@tzip@single@b\expandafter#1#2\pxrr@end
376 }
377 \def\pxrr@tzip@single@b#1{%
378   \expandafter\pxrr@tzip@single@c#1\pxrr@end
379 }
380 \def\pxrr@tzip@single@c#1\pxrr@end#2\pxrr@end#3\pxrr@end{%
381   \def\pxrr@res{\pxrr@pre{#1}{#2}{#3}\pxrr@post}%
382 }

```

4.5 エンジン依存処理

この小節のマクロ内で使われる変数。

```

383 \let\pxrr@x@tempa\@empty
384 \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。

```

385 \def\pxrr@ifprimitive#1{%
386   \edef\pxrr@x@tempa{\string#1}%
387   \edef\pxrr@x@tempb{\meaning#1}%
388   \ifx\pxrr@x@tempa\pxrr@x@tempb \expandafter\@firstoftwo
389   \else \expandafter\@secondoftwo
390   \fi
391 }

```

`\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。

```

392 \pxrr@ifprimitive\kansuji{%
393   \pxrr@csletcs{ifpxrr@in@ptex}{iftrue}%
394 }{%
395   \pxrr@csletcs{ifpxrr@in@ptex}{iffalse}%
396 }

```

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

```

397 \pxrr@ifprimitive\enablecjktoken{%
398   \pxrr@csletcs{ifpxrr@in@uptex}{iftrue}%
399 }{%
400   \pxrr@csletcs{ifpxrr@in@uptex}{iffalse}%
401 }

```

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

```

402 \pxrr@ifprimitive\XeTeXrevision{%

```

```

403 \pxrr@csletcs{ifpxrr@in@xetex}{iftrue}%
404 }{%
405 \pxrr@csletcs{ifpxrr@in@xetex}{iffalse}%
406 }

```

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

```

407 \@ifpackageloaded{xeCJK}{%
408 \pxrr@csletcs{ifpxrr@in@xecjk}{iftrue}%
409 }{%
410 \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。

```

411 \AtBeginDocument{%
412 \ifpackageloaded{xeCJK}{%
413 \pxrr@warn@load@order{xeCJK}%
414 }{%
415 }%
416 }

```

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

```

417 \pxrr@ifprimitive\luatexrevision{%
418 \pxrr@csletcs{ifpxrr@in@luatex}{iftrue}%
419 }{%
420 \pxrr@csletcs{ifpxrr@in@luatex}{iffalse}%
421 }

```

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

```

422 \@ifpackageloaded{luatexja-core}{%
423 \pxrr@csletcs{ifpxrr@in@luatexja}{iftrue}%
424 }{%
425 \pxrr@csletcs{ifpxrr@in@luatexja}{iffalse}%
426 \AtBeginDocument{%
427 \ifpackageloaded{luatexja-core}{%
428 \pxrr@warn@load@order{LuaTeX-ja}%
429 }{%
430 }%
431 }

```

```

432 \ifpxrr@in@xetex
433 \else\ifpxrr@in@luatex
434 \else\ifpxrr@in@ptex
435 \else
436 \pxrr@ifprimitive\pdftexrevision{%
437 \pxrr@warn{%
438 The engine in use seems to be pdfTeX,\MessageBreak
439 so safe mode is turned on%
440 }%
441 \AtEndOfPackage{%
442 \rubysafemode
443 }%

```

```
444 }
445 \fi\fi\fi
```

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

```
446 \ifpxrr@in@xetex
447 \pxrr@csletcs{ifpxrr@in@unicode}{iftrue}%
448 \else\ifpxrr@in@luatex
449 \pxrr@csletcs{ifpxrr@in@unicode}{iftrue}%
450 \else\ifpxrr@in@uptex
451 \pxrr@csletcs{ifpxrr@in@unicode}{iftrue}%
452 \else
453 \pxrr@csletcs{ifpxrr@in@unicode}{iffalse}%
454 \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。

```
455 \def\pxrr@jc#1{%
456 \pxrr@jc@a#1\pxrr@nil
457 }
458 \ifpxrr@in@unicode
459 \def\pxrr@jc@a#1:#2\pxrr@nil{%
460 "#2\space
461 }
462 \else\ifpxrr@in@ptex
463 \def\pxrr@jc@a#1:#2\pxrr@nil{%
464 \jis"#1\space\space
465 }
466 \else
467 \def\pxrr@jc@a#1:#2\pxrr@nil{%
468 ‘?\space
469 }
470 \fi\fi
```

`\pxrr@jchardef` 和文用の `\chardef`。

```
471 \ifpxrr@in@uptex
472 \let\pxrr@jchardef\kchardef
473 \else
474 \let\pxrr@jchardef\chardef
475 \fi
```

`\ifpxrr@in@tate` 縦組であるか。

※ p_TE_X 以外での縦組をサポートする予定はない。

```
476 \ifpxrr@in@ptex
477 \pxrr@csletcs{ifpxrr@in@tate}{iftdir}
478 \else
479 \pxrr@csletcs{ifpxrr@in@tate}{iffalse}
480 \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。

```
481 \ifpxrr@in@ptex
482   \def\pxrr@get@jchar@token#1#2{%
483     \begingroup
484       \kansujichar\@ne=#2\relax
485       \xdef\pxrr@x@gtempa{\kansuji\@ne}%
486     \endgroup
487     \let#1\pxrr@x@gtempa
488   }
```

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

```
489 \else\ifpxrr@in@unicode
490   \def\pxrr@get@jchar@token#1#2{%
491     \begingroup
492       \lccode'\?=#2\relax
493       \lowercase{\xdef\pxrr@x@gtempa{?}}%
494     \endgroup
495     \let#1\pxrr@x@gtempa
496   }
```

それ以外ではダミー定義。

```
497 \else
498   \def\pxrr@get@jchar@token#1#2{%
499     \def#1{?}%
500   }
501 \fi\fi
```

\pxrr@x@K 適当な漢字（実際は <一>）のトークン。

```
502 \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。

```
503 \ifpxrr@in@ptex
504   \def\pxrr@get@iiskip#1{%
505     \pxrr@x@swafalse
506     \begingroup
507       \inhibitxspcode\pxrr@x@K\thr@@
508       \kanjiskip\p@
509       \setbox\z@\hbox{\noautospacing\pxrr@x@K\pxrr@x@K}%
510       \setbox\tw@\hbox{\pxrr@x@K\pxrr@x@K}%
511       \ifdim\wd\tw@>\wd\z@
512         \aftergroup\pxrr@x@swatruue
513       \fi
514     \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。

```

515 \edef#1{%
516 \ifpxrr@x@swa \the\kanjiskip
517 \else \pxrr@zeropt
518 \fi
519 }%
520 }

```

LuaTeX-ja 使用の場合。

```

521 \else\ifpxrr@in@luatexja
522 \def\pxrr@get@iiskip#1{%
523 \edef#1{%
524 \ifnum\ltjgetparameter{autospacing}=\@ne
525 \ltjgetparameter{kanjiskip}%
526 \else \pxrr@zeropt
527 \fi
528 }%
529 }

```

それ以外の場合はゼロとする。

```

530 \else
531 \def\pxrr@get@iiskip#1{%
532 \let#1\pxrr@zeropt
533 }
534 \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。

```

535 \ifpxrr@in@ptex
536 \def\pxrr@get@iaiskip#1{%
537 \pxrr@x@swafalse
538 \begingroup
539 \inhibitxspcode\pxrr@x@K\thr@@ \xspcode'X=\thr@@
540 \xkanjiskip\p@
541 \setbox\z@\hbox{\noautoxspacing\pxrr@x@K X}%
542 \setbox\tw@\hbox{\pxrr@x@K X}%
543 \ifdim\wd\tw@>\wd\z@
544 \aftergroup\pxrr@x@swatruue
545 \fi
546 \endgroup
547 \edef#1{%
548 \ifpxrr@x@swa \the\xkanjiskip
549 \else \pxrr@zeropt
550 \fi
551 }%
552 }

```

LuaTeX-ja 使用の場合。

```

553 \else\ifpxrr@in@luatexja
554 \def\pxrr@get@iaiskip#1{%
555 \edef#1{%

```

```

556     \ifnum\ltjgetparameter{autoxspacing}=\@ne
557     \ltjgetparameter{xkanjiskip}%
558     \else \pxrr@zeropt
559     \fi
560   }%
561 }

```

それ以外の場合は実際の組版結果から判断する。

```

562 \else
563   \def\pxrr@get@iaiskip#1{%
564     \begingroup
565     \setbox\z@\hbox{M\pxrr@x@K}%
566     \setbox\tw@\hbox{M\vrule\@width\z@\relax\pxrr@x@K}%
567     \@tempdima\wd\z@ \advance\@tempdima-\wd\tw@
568     \@tempdimb\@tempdima \divide\@tempdimb\thr@@
569     \xdef\pxrr@x@tempa{\the\@tempdima\space minus \the\@tempdimb}%
570   \endgroup
571   \let#1=\pxrr@x@tempa
572 }%
573 \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。

```

574 \ifpxrr@in@ptex
575   \def\pxrr@get@zwidth#1{%
576     \@tempdima=1zw\relax
577     \edef#1{\the\@tempdima}%
578   }

```

`\zw` が定義されている場合は `1\zw` とする。

```

579 \else\if\ifx\zw\@undefined T\else F\fi F% if defined
580   \def\pxrr@get@zwidth#1{%
581     \@tempdima=1\zw\relax
582     \edef#1{\the\@tempdima}%
583   }

```

`\jsZw` が定義されている場合は `1\jsZw` とする。

```

584 \else\if\ifx\jsZw\@undefined T\else F\fi F% if defined
585   \def\pxrr@get@zwidth#1{%
586     \@tempdima=1\jsZw\relax
587     \edef#1{\the\@tempdima}%
588   }

```

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。

```

589 \else\ifnum\pxrr@x@K>\@cclv
590   \def\pxrr@get@zwidth#1{%
591     \setbox\tw@\hbox{\pxrr@x@K}%
592     \@tempdima\wd\tw@
593     \ifdim\@tempdima>\z@\else \@tempdima\fontsize\p@ \fi
594     \edef#1{\the\@tempdima}%

```

```

595 }
    それ以外の場合は要求サイズと等しいとする。
596 \else
597   \def\pxrr@get@zwidth#1{%
598     \@tempdima\f@size\p@\relax
599     \edef#1{\the\@tempdima}%
600   }
601 \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。

```

602 \ifpxrr@in@ptex
603   \def\pxrr@get@prebreakpenalty#1#2{%
604     #1=\prebreakpenalty'#2\relax
605   }

```

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

```

606 \else\ifpxrr@in@luatexja
607   \def\pxrr@get@prebreakpenalty#1#2{%
608     #1=\ltjgetparameter{prebreakpenalty}{'#2}\relax
609   }

```

それ以外の場合はゼロとして扱う。

```

610 \else
611   \def\pxrr@get@prebreakpenalty#1#2{%
612     #1=\z@
613   }
614 \fi\fi

```

4.6 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。

```

615 \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。

```

616 \newcommand*\rubyssetup[1]{%
617   \pxrr@in@setuptrue
618   \pxrr@fatal@errorfalse
619   \pxrr@parse@option{#1}%
620   \ifpxrr@fatal@error\else
621     \pxrr@csletcs{ifpxrr@d@bprotr}{ifpxrr@bprotr}%
622     \pxrr@csletcs{ifpxrr@d@aprotr}{ifpxrr@aprotr}%
623     \let\pxrr@d@bintr\pxrr@bintr@
624     \let\pxrr@d@aintr\pxrr@aintr@
625     \let\pxrr@d@ahead\pxrr@ahead
626     \let\pxrr@d@mode\pxrr@mode

```

```

627 \let\pxrr@d@side\pxrr@side
628 \let\pxrr@d@evensp\pxrr@evensp
629 \let\pxrr@d@fullsize\pxrr@fullsize
630 \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。

```

631 \pxrr@in@setupfalse
632 }

```

\rubyfontsetup 対応するパラメタを設定する。

```

633 \newcommand*\rubyfontsetup{}
634 \def\rubyfontsetup#%
635 \def\pxrr@ruby@font
636 }

```

\rubybigintrusion 対応するパラメタを設定する。

```

\rubysmallintrusion 637 \newcommand*\rubybigintrusion[1]{%
\rubymaxmargin 638 \edef\pxrr@big@intr{#1}%
639 }
\rubyintergap 640 \newcommand*\rubysmallintrusion[1]{%
\rubysizeratio 641 \edef\pxrr@small@intr{#1}%
642 }
643 \newcommand*\rubymaxmargin[1]{%
644 \edef\pxrr@maxmargin{#1}%
645 }
646 \newcommand*\rubyintergap[1]{%
647 \edef\pxrr@inter@gap{#1}%
648 }
649 \newcommand*\rubysizeratio[1]{%
650 \edef\pxrr@size@ratio{#1}%
651 }

```

\rubyusejghost 対応するスイッチを設定する。

```

\rubynousejghost 652 \newcommand*\rubyusejghost{%
653 \pxrr@jghosttrue
654 }
655 \newcommand*\rubynousejghost{%
656 \pxrr@jghostfalse
657 }

```

\rubyuseaghost 対応するスイッチを設定する。

```

\rubynouseaghost 658 \newcommand*\rubyuseaghost{%
659 \pxrr@aghosttrue
660 \pxrr@setup@aghost
661 }
662 \newcommand*\rubynouseaghost{%
663 \pxrr@aghostfalse
664 }

```

```

\rubyadjustatlineedge 対応するスイッチを設定する。
\rubynoadjustatlineedge 665 \newcommand*\rubyadjustatlineedge{%
666   \pxrr@edge@adjusttrue
667 }
668 \newcommand*\rubynoadjustatlineedge{%
669   \pxrr@edge@adjustfalse
670 }

\rubybreakjukugo 対応するスイッチを設定する。
\rubynobreakjukugo 671 \newcommand*\rubybreakjukugo{%
672   \pxrr@break@jukugotrue
673 }
674 \newcommand*\rubynobreakjukugo{%
675   \pxrr@break@jukugofalse
676 }

\rubysafemode 対応するスイッチを設定する。
\rubynosafemode 677 \newcommand*\rubysafemode{%
678   \pxrr@safe@modetrue
679 }
680 \newcommand*\rubynosafemode{%
681   \pxrr@safe@modefalse
682 }

\rubystretchprop 対応するパラメタを設定する。
\rubystretchprophead 683 \newcommand*\rubystretchprop[3]{%
684   \edef\pxrr@sprop@x{#1}%
\rubystretchpropend 685   \edef\pxrr@sprop@y{#2}%
686   \edef\pxrr@sprop@z{#3}%
687 }
688 \newcommand*\rubystretchprophead[2]{%
689   \edef\pxrr@sprop@hy{#1}%
690   \edef\pxrr@sprop@hz{#2}%
691 }
692 \newcommand*\rubystretchpropend[2]{%
693   \edef\pxrr@sprop@ex{#1}%
694   \edef\pxrr@sprop@ey{#2}%
695 }

\rubyuseextra 残念ながら今のところは使用不可。
696 \newcommand*\rubyuseextra[1]{%
697   \pxrr@cmta=#1\relax
698   \ifnum\pxrr@cmta=\z@
699     \chardef\pxrr@extra\pxrr@cmta
700   \else
701     \pxrr@err@inv@value{\the\pxrr@cmta}%
702   \fi
703 }

```

4.7 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。)

```
704 \let\pxrr@bintr@\@empty
```

```
705 \let\pxrr@aintr@\@empty
```

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

```
706 \def\pxrr@doublebar{||}
```

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

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

入力が「||」の場合は、「|-|」に置き換える。

```
708 \edef\pxrr@tempa{#1}%
```

```
709 \ifx\pxrr@tempa\pxrr@doublebar
```

```
710 \def\pxrr@tempa{||}%
```

```
711 \fi
```

各パラメタの値を全般設定のもので初期化する。

```
712 \pxrr@csletcs{ifpxrr@bprotr}{ifpxrr@d@bprotr}%
```

```
713 \pxrr@csletcs{ifpxrr@aprotr}{ifpxrr@d@aprotr}%
```

```
714 \let\pxrr@bintr@\pxrr@d@bintr
```

```
715 \let\pxrr@aintr@\pxrr@d@aintr
```

```
716 \let\pxrr@athead@\pxrr@d@athead
```

```
717 \let\pxrr@mode\pxrr@d@mode
```

```
718 \let\pxrr@side\pxrr@d@side
```

```
719 \let\pxrr@evensp\pxrr@d@evensp
```

```
720 \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。

```
721 \let\pxrr@bscomp\relax
```

```
722 \let\pxrr@ascomp\relax
```

```
723 \pxrr@bnobrfalse
```

```
724 \pxrr@anobrfalse
```

```
725 \pxrr@bfintrfalse
```

```
726 \pxrr@afintrfalse
```

明示フラグを偽にする。

```
727 \pxrr@mode@givenfalse
```

```
728 \pxrr@athead@givenfalse
```

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。

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

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

```
731 }
```

有限状態機械のループ。

```
732 \def\pxrr@parse@option@loop#1{%
733 \ifpxrrDebug
734 \typeout{\pxrr@po@FS/#1[\@nameuse{pxrr@po@C@#1}]}%
735 \fi
736 \csname pxrr@po@PR@#1\endcsname
737 \expandafter\ifx\csname pxrr@po@C@#1\endcsname\relax
738 \let\pxrr@po@FS\relax
739 \else
740 \pxrr@letcs\pxrr@po@FS
741 {\pxrr@po@TR@\pxrr@po@FS \@nameuse{pxrr@po@C@#1}]}%
742 \fi
743 \ifpxrrDebug
744 \typeout{->\pxrr@po@FS}%
745 \fi
746 \pxrr@ifx{\pxrr@po@FS\relax}{%
747 \pxrr@fatal@unx@letter{#1}%
748 \pxrr@parse@option@exit
749 }{%
750 \pxrr@parse@option@loop
751 }%
752 }
```

後処理。

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

既定値設定 (\rubyssetup) である場合もしない。

```
754 \ifpxrr@in@setup\else
```

両側ルビ命令の場合は、\pxrr@side の値を変更する。

```
755 \ifpxrr@truby
756 \chardef\pxrr@side\tw@
757 \fi
```

整合性検査を行う。

```
758 \pxrr@check@option
```

\pxrr@?intr の値を設定する。

```
759 \@tempdima=\pxrr@ruby@zw\relax
760 \@tempdimb=\pxrr@or@zero\pxrr@bintr@\@tempdima
761 \edef\pxrr@bintr{\the\@tempdimb}%
762 \@tempdimb=\pxrr@or@zero\pxrr@aintr@\@tempdima
763 \edef\pxrr@aintr{\the\@tempdimb}%
764 \fi
765 }
```

\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。

```
766 \def\pxrr@or@zero#1{%
767 \ifx#1\@empty \pxrr@zero
768 \else #1%
```

```

769 \fi
770 }

```

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。

```

771 \def\pxrr@po@C@{F}
772 \@namedef{pxrr@po@C@|}{V}
773 \@namedef{pxrr@po@C@:}{S}
774 \@namedef{pxrr@po@C@.}{S}
775 \@namedef{pxrr@po@C@*}{S}
776 \@namedef{pxrr@po@C@!}{S}
777 \@namedef{pxrr@po@C@<}{B}
778 \@namedef{pxrr@po@C@()}{B}
779 \@namedef{pxrr@po@C@>}{A}
780 \@namedef{pxrr@po@C@)}{A}
781 \@namedef{pxrr@po@C@-}{M}
782 \def\pxrr@po@C@c{M}
783 \def\pxrr@po@C@h{M}
784 \def\pxrr@po@C@H{M}
785 \def\pxrr@po@C@m{M}
786 \def\pxrr@po@C@g{M}
787 \def\pxrr@po@C@j{M}
788 \def\pxrr@po@C@P{M}
789 \def\pxrr@po@C@S{M}
790 \def\pxrr@po@C@e{M}
791 \def\pxrr@po@C@E{M}
792 \def\pxrr@po@C@f{M}
793 \def\pxrr@po@C@F{M}

```

機能プロセス。

```

794 \def\pxrr@po@PR@{%
795 \pxrr@parse@option@exit
796 }
797 \@namedef{pxrr@po@PR@|}{%
798 \csname pxrr@po@PRbar@\pxrr@po@FS\endcsname
799 }
800 \def\pxrr@po@PRbar@bi{%
801 \def\pxrr@bintr@{\pxrr@bprotrtrue
802 }
803 \def\pxrr@po@PRbar@bb{%
804 \pxrr@bprotrfalse
805 }
806 \def\pxrr@po@PRbar@bs{%
807 \def\pxrr@aintr@{\pxrr@aprotrtrue
808 }
809 \let\pxrr@po@PRbar@mi\pxrr@po@PRbar@bs
810 \let\pxrr@po@PRbar@as\pxrr@po@PRbar@bs
811 \let\pxrr@po@PRbar@ai\pxrr@po@PRbar@bs
812 \def\pxrr@po@PRbar@ab{%

```

```

813 \pxrr@aprotrfalse
814 }
815 \@namedef{pxrr@po@PR@:}{%
816 \csname pxrr@po@PRcolon@\pxrr@po@FS\endcsname
817 }
818 \def\pxrr@po@PRcolon@bi{%
819 \let\pxrr@bscomp=\relax
820 }
821 \let\pxrr@po@PRcolon@bb\pxrr@po@PRcolon@bi
822 \let\pxrr@po@PRcolon@bs\pxrr@po@PRcolon@bi
823 \def\pxrr@po@PRcolon@mi{%
824 \let\pxrr@ascomp=\relax
825 }
826 \let\pxrr@po@PRcolon@as\pxrr@po@PRcolon@mi
827 \@namedef{pxrr@po@PR@.}{%
828 \csname pxrr@po@PRdot@\pxrr@po@FS\endcsname
829 }
830 \def\pxrr@po@PRdot@bi{%
831 \let\pxrr@bscomp=\relax
832 }
833 \let\pxrr@po@PRdot@bb\pxrr@po@PRdot@bi
834 \let\pxrr@po@PRdot@bs\pxrr@po@PRdot@bi
835 \def\pxrr@po@PRdot@mi{%
836 \let\pxrr@ascomp=\relax
837 }
838 \let\pxrr@po@PRdot@as\pxrr@po@PRdot@mi
839 \@namedef{pxrr@po@PR@*}{%
840 \csname pxrr@po@PRstar@\pxrr@po@FS\endcsname
841 }
842 \def\pxrr@po@PRstar@bi{%
843 \pxrr@bnobrtrue
844 }
845 \let\pxrr@po@PRstar@bb\pxrr@po@PRstar@bi
846 \let\pxrr@po@PRstar@bs\pxrr@po@PRstar@bi
847 \def\pxrr@po@PRstar@mi{%
848 \pxrr@anobrtrue
849 }
850 \let\pxrr@po@PRstar@as\pxrr@po@PRstar@mi
851 \@namedef{pxrr@po@PR@!}{%
852 \csname pxrr@po@PRbang@\pxrr@po@FS\endcsname
853 }
854 \def\pxrr@po@PRbang@bi{%
855 \pxrr@bfintrtrue
856 }
857 \let\pxrr@po@PRbang@bb\pxrr@po@PRbang@bi
858 \let\pxrr@po@PRbang@bs\pxrr@po@PRbang@bi
859 \def\pxrr@po@PRbang@mi{%
860 \pxrr@afintrtrue
861 }

```

```

862 \let\pxrr@po@PRbang@as\pxrr@po@PRbang@mi
863 \@namedef{pxrr@po@PR<}{%
864   \def\pxrr@bintr@{\pxrr@big@intr}\pxrr@bprottrue
865 }
866 \@namedef{pxrr@po@PR@()}{%
867   \def\pxrr@bintr@{\pxrr@small@intr}\pxrr@bprottrue
868 }
869 \@namedef{pxrr@po@PR@>}{%
870   \def\pxrr@aintr@{\pxrr@big@intr}\pxrr@aprottrue
871 }
872 \@namedef{pxrr@po@PR@)}{%
873   \def\pxrr@aintr@{\pxrr@small@intr}\pxrr@aprottrue
874 }
875 \def\pxrr@po@PR@c{%
876   \chardef\pxrr@athead\z@
877   \pxrr@athead@giventru
878 }
879 \def\pxrr@po@PR@h{%
880   \chardef\pxrr@athead\@ne
881   \pxrr@athead@giventru
882 }
883 \def\pxrr@po@PR@H{%
884   \chardef\pxrr@athead\tw@
885   \pxrr@athead@giventru
886 }
887 \def\pxrr@po@PR@m{%
888   \let\pxrr@mode=m%
889   \pxrr@mode@giventru
890 }
891 \def\pxrr@po@PR@g{%
892   \let\pxrr@mode=g%
893   \pxrr@mode@giventru
894 }
895 \def\pxrr@po@PR@j{%
896   \let\pxrr@mode=j%
897   \pxrr@mode@giventru
898 }
899 \def\pxrr@po@PR@P{%
900   \chardef\pxrr@side\z@
901 }
902 \def\pxrr@po@PR@S{%
903   \chardef\pxrr@side\@ne
904 }
905 \def\pxrr@po@PR@E{%
906   \chardef\pxrr@evensp\z@
907 }
908 \def\pxrr@po@PR@e{%
909   \chardef\pxrr@evensp\@ne
910 }

```

```

911 \def\pxrr@po@PR@F{%
912   \chardef\pxrr@fullsize\z@
913 }
914 \def\pxrr@po@PR@f{%
915   \chardef\pxrr@fullsize\@ne
916 }

```

遷移表。

```

917 \def\pxrr@po@TR@bi@F{fi}
918 \def\pxrr@po@TR@bb@F{fi}
919 \def\pxrr@po@TR@bs@F{fi}
920 \def\pxrr@po@TR@mi@F{fi}
921 \def\pxrr@po@TR@as@F{fi}
922 \def\pxrr@po@TR@ai@F{fi}
923 \def\pxrr@po@TR@ab@F{fi}
924 \def\pxrr@po@TR@fi@F{fi}
925 \def\pxrr@po@TR@bi@V{bb}
926 \def\pxrr@po@TR@bb@V{bs}
927 \def\pxrr@po@TR@bs@V{ab}
928 \def\pxrr@po@TR@mi@V{ab}
929 \def\pxrr@po@TR@as@V{ab}
930 \def\pxrr@po@TR@ai@V{ab}
931 \def\pxrr@po@TR@ab@V{fi}
932 \def\pxrr@po@TR@bi@S{bs}
933 \def\pxrr@po@TR@bb@S{bs}
934 \def\pxrr@po@TR@bs@S{bs}
935 \def\pxrr@po@TR@mi@S{as}
936 \def\pxrr@po@TR@as@S{as}
937 \def\pxrr@po@TR@bi@B{bs}
938 \def\pxrr@po@TR@bi@M{mi}
939 \def\pxrr@po@TR@bb@M{mi}
940 \def\pxrr@po@TR@bs@M{mi}
941 \def\pxrr@po@TR@mi@M{mi}
942 \def\pxrr@po@TR@bi@A{fi}
943 \def\pxrr@po@TR@bb@A{fi}
944 \def\pxrr@po@TR@bs@A{fi}
945 \def\pxrr@po@TR@mi@A{fi}
946 \def\pxrr@po@TR@as@A{fi}
947 \def\pxrr@po@TR@ai@A{fi}

```

4.8 オプション整合性検査

`\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。

```

948 \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。

```

949   \ifpxrr@bprotr\else

```

```

950 \ifpxrr@aprotr\else
951 \pxrr@fatal@bad@no@protr
952 \fi
953 \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。

```

954 \pxrr@oktrue
955 \ifx\pxrr@bintr@\@empty\else
956 \pxrr@okfalse
957 \fi
958 \ifx\pxrr@aintr@\@empty\else
959 \pxrr@okfalse
960 \fi
961 \ifpxrr@ghost\else
962 \pxrr@oktrue
963 \fi
964 \ifpxrr@ok\else
965 \pxrr@fatal@bad@intr
966 \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。

```

967 \if g\pxrr@mode\else
968 \ifpxrr@abody
969 \let\pxrr@mode=g\relax
970 \ifpxrr@mode@given
971 \pxrr@warn@must@group
972 \fi
973 \fi
974 \fi

```

両側ルビではモノルビ (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。

```

975 \if g\pxrr@mode\else
976 \ifnum\pxrr@side=\tw@
977 \let\pxrr@mode=g\relax
978 \ifpxrr@mode@given
979 \pxrr@warn@must@group
980 \fi
981 \fi
982 \fi

```

肩付き指定 (h) に関する検査。

```

983 \ifnum\pxrr@ahead>\z@

```

横組みでは不可なので中付きに変更する。

```

984 \ifpxrr@in@tate\else
985 \chardef\pxrr@ahead\z@
986 \fi

```

グループルビでは不可なので中付きに変更する。

```

987 \if g\pxrr@mode
988 \chardef\pxrr@athead\z@
989 \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。

```

990 \ifnum\pxrr@athead=\z@
991 \ifpxrr@athead@given
992 \pxrr@warn@bad@athead
993 \fi
994 \fi
995 \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。

```

996 \ifpxrr@abody
997 \chardef\pxrr@evensp\z@
998 \fi

```

グルーブルビ以外では、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。)

```

999 \if g\pxrr@mode\else
1000 \chardef\pxrr@evensp@one
1001 \fi
1002 }

```

4.9 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。

```
1003 \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。

```
1004 \let\pxrr@body@zw\pxrr@zeropt
```

```
1005 \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。

```
1006 \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)。

```
1007 \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。

```
1008 \def\pxrr@htratio{0}
```

`\pxrr@iiskip` 和文間空白および和欧文間空白の量。

```
\pxrr@iaiskip 1009 \let\pxrr@iiskip\pxrr@zeropt
1010 \let\pxrr@iaiskip\pxrr@zeropt
```

`\pxrr@assign@fsize` 上記の変数（マクロ）を設定する。

```
1011 \def\pxrr@assign@fsize{%
1012   \@tempdima=\f@size\p@
1013   \@tempdima\pxrr@size@ratio\@tempdima
1014   \edef\pxrr@ruby@fsize{\the\@tempdima}%
1015   \pxrr@get@zwidth\pxrr@body@zw
1016   \begingroup
1017     \pxrr@use@ruby@font
1018     \pxrr@get@zwidth\pxrr@gtempa
1019     \global\let\pxrr@gtempa\pxrr@gtempa
1020   \endgroup
1021   \let\pxrr@ruby@zw\pxrr@gtempa
1022   \pxrr@get@iiskip\pxrr@iiskip
1023   \pxrr@get@iaiskip\pxrr@iaiskip
```

`\pxrr@htratio` の値を設定する。

```
1024 \ifpxrr@in@tate
1025   \let\pxrr@htratio\pxrr@thtratio
1026 \else
1027   \let\pxrr@htratio\pxrr@yhtratio
1028 \fi
```

`\pxrr@ruby@raise` の値を計算する。

```
1029 \@tempdima\pxrr@body@zw\relax
1030 \@tempdima\pxrr@htratio\@tempdima
1031 \@tempdimb\pxrr@ruby@zw\relax
1032 \advance\@tempdimb-\pxrr@htratio\@tempdimb
1033 \advance\@tempdima\@tempdimb
1034 \@tempdimb\pxrr@body@zw\relax
1035 \advance\@tempdima\pxrr@inter@gap\@tempdimb
1036 \edef\pxrr@ruby@raise{\the\@tempdima}%
```

`\pxrr@ruby@lower` の値を計算する。

```
1037 \@tempdima\pxrr@body@zw\relax
1038 \advance\@tempdima-\pxrr@htratio\@tempdima
1039 \@tempdimb\pxrr@ruby@zw\relax
1040 \@tempdimb\pxrr@htratio\@tempdimb
1041 \advance\@tempdima\@tempdimb
1042 \@tempdimb\pxrr@body@zw\relax
1043 \advance\@tempdima\pxrr@inter@gap\@tempdimb
1044 \edef\pxrr@ruby@lower{\the\@tempdima}%
1045 }
```

`\pxrr@use@ruby@font` ルビ用のフォントに切り替える。

```
1046 \def\pxrr@use@ruby@font{%
1047   \pxrr@without@macro@trace{%
```

```

1048 \let\rubyfontsize\pxrr@ruby@fsize
1049 \fontsize{\pxrr@ruby@fsize}{\z@}\selectfont
1050 \pxrr@ruby@font
1051 }%
1052 }

```

4.10 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 1053 \chardef\pxrr@locate@inner=1
\pxrr@locate@end 1054 \chardef\pxrr@locate@head=0
1055 \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。

```
1056 \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。

```
1057 \setbox#2\pxrr@hbox{#5}\@tempdima\wd#2%
1058 \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。

```
1059 \pxrr@decompose{#5}%
1060 \pxrr@evenspace@int{#1}{#2}{#3}{#4}%
1061 }

```

ここから実行を開始することもある。

```
1062 \def\pxrr@evenspace@int#1#2#3#4{%
```

比率パラメタの設定。

```

1063 \pxrr@save@listproc
1064 \ifcase#1%
1065 \pxrr@evenspace@param\pxrr@zero\pxrr@sprop@hy\pxrr@sprop@hz
1066 \or
1067 \pxrr@evenspace@param\pxrr@sprop@x\pxrr@sprop@y\pxrr@sprop@z
1068 \or
1069 \pxrr@evenspace@param\pxrr@sprop@ex\pxrr@sprop@ey\pxrr@zero

```

```

1070 \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。

1071 \pxrr@dima=\pxrr@cntr\p@
1072 \advance\pxrr@dima-\p@
1073 \pxrr@dima=\pxrr@sprop@y@\pxrr@dima
1074 \advance\pxrr@dima\pxrr@sprop@x@\p@
1075 \advance\pxrr@dima\pxrr@sprop@z@\p@
1076 \ifdim\pxrr@dima>\z@\else
1077 \ifnum#1>\z@
1078 \let\pxrr@sprop@x@\@ne
1079 \advance\pxrr@dima\p@
1080 \fi
1081 \ifnum#1<\tw@
1082 \let\pxrr@sprop@z@\@ne
1083 \advance\pxrr@dima\p@
1084 \fi
1085 \fi
1086 \edef\pxrr@tempa{\strip@pt\pxrr@dima}%
1087 \ifpxrr@Debug
1088 \typeout{\number\pxrr@sprop@x@:\number\pxrr@sprop@z@:\pxrr@tempa}%
1089 \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。

1090 \def\pxrr@pre##1{\pxrr@hfilx\pxrr@sprop@x@ ##1}%
1091 \def\pxrr@inter##1{\pxrr@hfilx\pxrr@sprop@y@ ##1}%
1092 \def\pxrr@post{\pxrr@hfilx\pxrr@sprop@z@}%
1093 \def\pxrr@makebox@res{%
1094 \setbox#2=\pxrr@hbox@to#4{#3\pxrr@res}%
1095 }%
1096 \pxrr@makebox@res

前後の空白の量を求める。

1097 \pxrr@dima\wd#2%
1098 \advance\pxrr@dima-\pxrr@natwd\relax
1099 \pxrr@invscale\pxrr@dima\pxrr@tempa
1100 \@tempdima\pxrr@sprop@x@\pxrr@dima
1101 \edef\pxrr@bspace{\the\@tempdima}%
1102 \@tempdima\pxrr@sprop@z@\pxrr@dima
1103 \edef\pxrr@aspace{\the\@tempdima}%
1104 \pxrr@restore@listproc
1105 \ifpxrr@Debug
1106 \typeout{\pxrr@bspace:\pxrr@aspace}%
1107 \fi
1108 }
1109 \def\pxrr@evenspace@param#1#2#3{%

```

```

1110 \let\pxrr@sprop@x@#1%
1111 \let\pxrr@sprop@y@#2%
1112 \let\pxrr@sprop@z@#3%
1113 }

```

`\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。

```

1114 \def\pxrr@adjust@margin{%
1115 \pxrr@save@listproc
1116 \@tempdima\pxrr@body@zw\relax
1117 \@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。

```

1118 \@tempswafalse
1119 \def\pxrr@pre##1{\pxrr@hfilx\pxrr@sprop@x@ ##1}%
1120 \def\pxrr@inter##1{\pxrr@hfilx\pxrr@sprop@y@ ##1}%
1121 \def\pxrr@post{\pxrr@hfilx\pxrr@sprop@z@}%
1122 \ifnum\pxrr@ctr>\@ne
1123 \ifdim\pxrr@bspace>\@tempdima
1124 \edef\pxrr@bspace{\the\@tempdima}%
1125 \def\pxrr@pre##1{\hskip\pxrr@bspace\relax ##1}%
1126 \@tempswatrue
1127 \fi
1128 \ifdim\pxrr@aspace>\@tempdima
1129 \edef\pxrr@aspace{\the\@tempdima}%
1130 \def\pxrr@post{\hskip\pxrr@aspace\relax}%
1131 \@tempswatrue
1132 \fi
1133 \fi

```

必要に応じて再調整を行う。

```

1134 \if@tempswa
1135 \pxrr@makebox@res
1136 \fi
1137 \pxrr@restore@listproc
1138 \ifpxrr@Debug
1139 \typeout{\pxrr@bspace:\pxrr@aspace}%
1140 \fi
1141 }

```

`\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。

```

1142 \def\pxrr@save@listproc{%
1143 \let\pxrr@pre@save\pxrr@pre
1144 \let\pxrr@inter@save\pxrr@inter
1145 \let\pxrr@post@save\pxrr@post
1146 }

```

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

```
1147 \def\pxrr@restore@listproc{%
1148   \let\pxrr@pre\pxrr@pre@save
1149   \let\pxrr@inter\pxrr@inter@save
1150   \let\pxrr@post\pxrr@post@save
1151 }
```

4.11 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。

```
1152 \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。

```
1153 \def\pxrr@transform@kana#1{%
1154   \let\pxrr@trans@res\@empty
1155   \def\pxrr@transform@kana@end\pxrr@end{%
1156     \let#1\pxrr@trans@res
1157   }%
1158   \expandafter\pxrr@transform@kana@loop@a#1\pxrr@end
1159 }
1160 \def\pxrr@transform@kana@loop@a{%
1161   \futurelet\pxrr@tempa\pxrr@transform@kana@loop@b
1162 }
1163 \def\pxrr@transform@kana@loop@b{%
1164   \ifx\pxrr@tempa\pxrr@end
1165     \let\pxrr@tempb\pxrr@transform@kana@end
1166   \else\ifx\pxrr@tempa\bgroup
1167     \let\pxrr@tempb\pxrr@transform@kana@loop@c
1168   \else\ifx\pxrr@tempa\@sptoken
1169     \let\pxrr@tempb\pxrr@transform@kana@loop@d
1170   \else
1171     \let\pxrr@tempb\pxrr@transform@kana@loop@e
1172   \fi\fi\fi
1173   \pxrr@tempb
1174 }
1175 \def\pxrr@transform@kana@loop@c#1{%
1176   \pxrr@appto\pxrr@trans@res{{#1}}%
1177   \pxrr@transform@kana@loop@a
1178 }
1179 \expandafter\def\expandafter\pxrr@transform@kana@loop@d\space{%
1180   \pxrr@appto\pxrr@trans@res{ }%
1181   \pxrr@transform@kana@loop@a
1182 }
1183 \def\pxrr@transform@kana@loop@e#1{%
1184   \expandafter\pxrr@transform@kana@loop@f\string#1\pxrr@nil#1%
1185 }
```

```

1186 \def\pxrr@transform@kana@loop@f#1#2\pxrr@nil#3{%
1187   \@tempswafalse
1188   \ifnum'#1>\@cclv
1189     \begingroup\expandafter\expandafter\expandafter\endgroup
1190     \expandafter\ifx\csname pxrr@nonsmall/#3\endcsname\relax\else
1191       \@tempswatruue
1192       \fi
1193       \fi
1194       \if@tempswa
1195         \edef\pxrr@tempa{%
1196           \noexpand\pxrr@appto\noexpand\pxrr@trans@res
1197             {\csname pxrr@nonsmall/#3\endcsname}%
1198           }%
1199         \pxrr@tempa
1200       \else
1201         \pxrr@appto\pxrr@trans@res{#3}%
1202       \fi
1203       \pxrr@transform@kana@loop@a
1204 }
1205 \def\pxrr@assign@nonsmall#1/#2\pxrr@nil{%
1206   \pxrr@get@jchar@token\pxrr@tempa{\pxrr@jc{#1}}%
1207   \pxrr@get@jchar@token\pxrr@tempb{\pxrr@jc{#2}}%
1208   \expandafter\edef\csname pxrr@nonsmall/\pxrr@tempa\endcsname
1209     {\pxrr@tempb}%
1210 }
1211 \@tfor\pxrr@tempc:=%
1212   {2421:3041/2422:3042}{2423:3043/2424:3044}%
1213   {2425:3045/2426:3046}{2427:3047/2428:3048}%
1214   {2429:3049/242A:304A}{2443:3063/2444:3064}%
1215   {2463:3083/2464:3084}{2465:3085/2466:3086}%
1216   {2467:3087/2468:3088}{246E:308E/246F:308F}%
1217   {2521:30A1/2522:30A2}{2523:30A3/2524:30A4}%
1218   {2525:30A5/2526:30A6}{2527:30A7/2528:30A8}%
1219   {2529:30A9/252A:30AA}{2543:30C3/2544:30C4}%
1220   {2563:30E3/2564:30E4}{2565:30E5/2566:30E6}%
1221   {2567:30E7/2568:30E8}{256E:30EE/256F:30EF}%
1222   \do{%
1223     \expandafter\pxrr@assign@nonsmall\pxrr@tempc\pxrr@nil
1224 }

```

4.12 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。

```
1225 \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。

```
1226 \newif\ifpxrr@any@protr
```

`\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）。

```
1227 \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。

```
1228 \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。

```
1229 \let\pxrr@compose@block@do\pxrr@compose@oneside@block@do
```

```
1230 \pxrr@compose@block@pre{#1}{#2}{#3}{}%
```

```
1231 }
```

こちらが本体。

```
1232 % #4 は空
```

```
1233 \def\pxrr@compose@oneside@block@do#1#2#3#4{%
```

```
1234 \setbox\pxrr@boxa\pxrr@hbox{#2}%
```

```
1235 \setbox\pxrr@boxr\pxrr@hbox{%
```

```
1236 \pxrr@use@ruby@font
```

```
1237 #3%
```

```
1238 }%
```

```
1239 \@tempdima\wd\pxrr@boxr
```

```
1240 \advance\@tempdima-\wd\pxrr@boxa
```

```
1241 \ifdim\pxrr@epsilon<\@tempdima
```

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

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

```
1242 \pxrr@protrtrue
```

```
1243 \let\pxrr@locate@temp#1%
```

```
1244 \ifnum\pxrr@athead>\@ne
```

```
1245 \ifnum\pxrr@locate@temp=\pxrr@locate@inner
```

```
1246 \let\pxrr@locate@temp\pxrr@locate@head
```

```
1247 \fi
```

```
1248 \fi
```

```
1249 \pxrr@decompose{#2}%
```

```
1250 \edef\pxrr@natwd{\the\wd\pxrr@boxa}%
```

```
1251 \pxrr@evenspace@int\pxrr@locate@temp\pxrr@boxa\relax
```

```
1252 {\wd\pxrr@boxr}%
```

```
1253 \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。

```

1254 \pxrr@protrfalse
1255 \ifnum\pxrr@athead=\z@
1256 \pxrr@decompose{#3}%
1257 \edef\pxrr@natwd{\the\wd\pxrr@boxr}%
1258 \pxrr@evenspace@int{#1}\pxrr@boxr
1259 \pxrr@use@ruby@font{\wd\pxrr@boxa}%
1260 \pxrr@adjust@margin
1261 \fi
1262 \let\pxrr@bspace\pxrr@zeropt
1263 \let\pxrr@aspace\pxrr@zeropt
1264 \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）。

```

1265 \pxrr@protrfalse
1266 \let\pxrr@bspace\pxrr@zeropt
1267 \let\pxrr@aspace\pxrr@zeropt
1268 \fi\fi

```

実際に組版を行う。

```

1269 \setbox\z@\hbox{%
1270 \ifnum\pxrr@side=\z@
1271 \raise\pxrr@ruby@raise\box\pxrr@boxr
1272 \else
1273 \lower\pxrr@ruby@lower\box\pxrr@boxr
1274 \fi
1275 }%
1276 \ht\z@\z@ \dp\z@\z@
1277 \@tempdima\wd\z@
1278 \setbox\pxrr@boxr\hbox{%
1279 \box\z@
1280 \kern-\@tempdima
1281 \box\pxrr@boxa
1282 }%

```

\ifpxrr@any@protr を設定する。

```

1283 \ifpxrr@protr
1284 \pxrr@any@protrtrue
1285 \fi
1286 }

```

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

```

1287 \def\pxrr@compose@twoside@block{%
1288 \let\pxrr@compose@block@do\pxrr@compose@twoside@block@do
1289 \pxrr@compose@block@pre
1290 }
1291 \def\pxrr@compose@twoside@block@do#1#2#3#4{%
1292 \setbox\pxrr@boxa\pxrr@hbox{#2}%
1293 \setbox\pxrr@boxr\pxrr@hbox{%
1294 \pxrr@use@ruby@font

```

```

1295     #3%
1296   }%
1297   \setbox\pxrr@boxb\pxrr@hbox{%
1298     \pxrr@use@ruby@font
1299     #4%
1300   }%

```

3つのボックスの最大の幅を求める。これが全体の幅となる。

```

1301   \@tempdima\wd\pxrr@boxa
1302   \ifdim\@tempdima<\wd\pxrr@boxr
1303     \@tempdima\wd\pxrr@boxr
1304   \fi
1305   \ifdim\@tempdima<\wd\pxrr@boxb
1306     \@tempdima\wd\pxrr@boxb
1307   \fi
1308   \edef\pxrr@maxwd{\the\@tempdima}%
1309   \advance\@tempdima-\pxrr@epsilon\relax
1310   \edef\pxrr@maxwdx{\the\@tempdima}%

```

全体の幅より短いボックスを均等割りで組み直す。

```

1311   \ifdim\pxrr@maxwdx>\wd\pxrr@boxr
1312     \pxrr@decompose{#3}%
1313     \edef\pxrr@natwd{\the\wd\pxrr@boxr}%
1314     \pxrr@evenspace@int{#1}\pxrr@boxr
1315     \pxrr@use@ruby@font{\pxrr@maxwd}%
1316     \pxrr@adjust@margin
1317   \fi
1318   \ifdim\pxrr@maxwdx>\wd\pxrr@boxb
1319     \pxrr@decompose{#4}%
1320     \edef\pxrr@natwd{\the\wd\pxrr@boxb}%
1321     \pxrr@evenspace@int{#1}\pxrr@boxb
1322     \pxrr@use@ruby@font{\pxrr@maxwd}%
1323     \pxrr@adjust@margin
1324   \fi

```

親文字列のボックスを最後に処理して、その `\pxrr@?space` の値を以降の処理で用いる。

(親文字列が短くない場合は `\pxrr@?space` はゼロ。)

```

1325   \ifdim\pxrr@maxwdx>\wd\pxrr@boxa
1326     \pxrr@decompose{#2}%
1327     \edef\pxrr@natwd{\the\wd\pxrr@boxa}%
1328     \pxrr@evenspace@int{#1}\pxrr@boxa\relax{\pxrr@maxwd}%
1329   \else
1330     \let\pxrr@bspace\pxrr@zeropt
1331     \let\pxrr@aspace\pxrr@zeropt
1332   \fi

```

実際に組版を行う。

```

1333   \setbox\z@\hbox{%
1334     \@tempdima\wd\pxrr@boxr
1335     \raise\pxrr@ruby@raise\box\pxrr@boxr

```

```

1336 \kern-\@tempdima
1337 \lower\pxrr@ruby@lower\box\pxrr@boxb
1338 }%
1339 \ht\z@\z@ \dp\z@\z@
1340 \@tempdima\wd\z@
1341 \setbox\pxrr@boxr\hbox{%
1342 \box\z@
1343 \kern-\@tempdima
1344 \box\pxrr@boxa
1345 }%
1346 }

```

`\pxrr@compose@block@pre` `\pxrr@compose@block@pre{<パターン>}{r <親文字>}{<ルビ 1>}{<ルビ 2>}`: 親文字列・ルビ文字列の加工を行う。

※ 両側ルビ対応のため、ルビ用引数が2つある。

```

1347 \def\pxrr@compose@block@pre{%
    f 指定時は小書き仮名の変換を施す。
1348 \pxrr@ifnum{\pxrr@fullsize>\z@}{%
1349 \pxrr@compose@block@pre@a
1350 }{%
1351 \pxrr@compose@block@pre@d
1352 }%
1353 }
1354 % {パターン}{親文字}{ルビ 1}{ルビ 2}
1355 \def\pxrr@compose@block@pre@a#1#2#3#4{%
1356 \def\pxrr@compose@block@tempa{#4}%
1357 \pxrr@transform@kana\pxrr@compose@block@tempa
1358 \expandafter\pxrr@compose@block@pre@b
1359 \expandafter{\pxrr@compose@block@tempa}{#1}{#2}{#3}%
1360 }
1361 % {ルビ 2}{パターン}{親文字}{ルビ 1}
1362 \def\pxrr@compose@block@pre@b#1#2#3#4{%
1363 \def\pxrr@compose@block@tempa{#4}%
1364 \pxrr@transform@kana\pxrr@compose@block@tempa
1365 \expandafter\pxrr@compose@block@pre@c
1366 \expandafter{\pxrr@compose@block@tempa}{#1}{#2}{#3}%
1367 }
1368 % {ルビ 1}{ルビ 2}{パターン}{親文字}
1369 \def\pxrr@compose@block@pre@c#1#2#3#4{%
1370 \pxrr@compose@block@pre@d{#3}{#4}{#1}{#2}%
1371 }
1372 \def\pxrr@compose@block@pre@d{%
1373 \pxrr@ifnum{\pxrr@evensp=\z@}{%
1374 \pxrr@compose@block@pre@e
1375 }{%
1376 \pxrr@compose@block@pre@f
1377 }%
1378 }

```

```

1379 % {パターン}{親文字}
1380 \def\pxrr@compose@block@pre@e#1#2{%
1381   \pxrr@compose@block@pre@f{#1}{#2}}%
1382 }
1383 \def\pxrr@compose@block@pre@f{%
1384   \pxrr@ifnum{\pxrr@reversp=\z@}{%
1385     \pxrr@compose@block@pre@g
1386   }{%
1387     \pxrr@compose@block@do
1388   }%
1389 }
1390 % {パターン}{親文字}{ルビ 1}{ルビ 2}
1391 \def\pxrr@compose@block@pre@g#1#2#3#4{%
1392   \pxrr@compose@block@do{#1}{#2}{#3}{#4}}%
1393 }

```

4.13 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。

```

1394 \def\pxrr@add@protect#1{%
1395   \expandafter\pxrr@add@protect@a
1396   \csname\expandafter\@gobble\string#1\space\endcsname#1%
1397 }
1398 \def\pxrr@add@protect@a#1#2{%
1399   \let#1=#2%
1400   \def#2{\pxrr@check@protect\protect#1}%
1401 }
1402 \def\pxrr@check@protect{%
1403   \ifx\protect\@typeset@protect
1404     \expandafter\@gobble
1405   \fi
1406 }

```

4.14 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。
1407 `\let\pxrr@body@input\@empty`

`\pxrr@prepare@fallback` `\pxrr@prepare@fallback{〈親文字列〉}` :
1408 `\def\pxrr@prepare@fallback#1{%`
1409 `\pxrr@fatal@errorfalse`
1410 `\def\pxrr@body@input{#1}%`
1411 `}`

`\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。
1412 `\def\pxrr@fallback{%`
1413 `\pxrr@body@input`
1414 `}`

`\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。
1415 `\def\pxrr@if@alive{%`
1416 `\ifpxrr@fatal@error \expandafter\@gobble`
1417 `\else \expandafter\@firstofone`
1418 `\fi`
1419 `}`

4.15 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。
1420 `\def\pxrr@end@kinsoku{0}`

`\pxrr@ruby@scan` 片側ルビ用の先読み処理。
1421 `\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。
1422 `\def\pxrr@tempc{%`
1423 `\edef\pxrr@end@kinsoku{\the\pxrr@cntr}%`
1424 `\pxrr@do@proc{#1}{#2}%`
1425 `}%`
1426 `\pxrr@check@kinsoku\pxrr@tempc`
1427 `}`

`\pxrr@truby@scan` 両側ルビ用の先読み処理。
1428 `\def\pxrr@truby@scan#1#2#3{%`
1429 `\def\pxrr@tempc{%`
1430 `\edef\pxrr@end@kinsoku{\the\pxrr@cntr}%`
1431 `\pxrr@do@proc{#1}{#2}{#3}%`

```

1432 }%
1433 \pxrr@check@kinsoku\pxrr@tempc
1434 }

```

`\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。

```

1435 \def\pxrr@check@kinsoku#1{%
1436   \let\pxrr@tempb#1%
1437   \futurelet\pxrr@tempa\pxrr@check@kinsoku@a
1438 }
1439 \def\pxrr@check@kinsoku@a{%
1440   \pxrr@check@char\pxrr@tempa

```

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。

```

1441   \ifpxrr@abody\else
1442     \ifnum\pxrr@cntr=\@ne
1443       \pxrr@cntr\tw@
1444     \fi
1445   \fi
1446   \ifcase\pxrr@cntr
1447     \pxrr@cntr\z@
1448     \expandafter\pxrr@tempb
1449   \or
1450     \pxrr@cntr\@MM
1451     \expandafter\pxrr@tempb
1452   \else
1453     \expandafter\pxrr@check@kinsoku@b
1454   \fi
1455 }

```

`\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。

```

1456 \def\pxrr@check@kinsoku@b#1{%
1457   \pxrr@check@kinsoku@c#1#1%
1458 }
1459 \def\pxrr@check@kinsoku@c#1{%
1460   \pxrr@get@prebreakpenalty\pxrr@cntr{#1}%
1461   \pxrr@tempb
1462 }

```

`\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。

```

1463 \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。)
1464 \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。
1465 \ifcat\noexpand##1\relax
1466 \pxrr@cntr\z@
1467 \else\ifcat\noexpand##1\noexpand~%
1468 \pxrr@cntr\z@
1469 \else\ifcat\noexpand##1A%
1470 \pxrr@cntr\@ne
1471 \else\ifcat\noexpand##10%
1472 \pxrr@cntr\@ne
1473 \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。)
1474 \pxrr@cntr\z@
1475 \expandafter\pxrr@check@char@a\meaning##1#2\pxrr@nil
1476 \fi\fi\fi\fi
1477 }%
1478 \def\pxrr@check@char@a##1#2##2\pxrr@nil{%
1479 \ifcat @##1@%
1480 \pxrr@cntr\tw@
1481 \fi
1482 }%
1483 }
    規定の引数を用意して「定義マクロ」を呼ぶ。
1484 \expandafter\pxrr@tempa\string\kanji\pxrr@nil

```

4.16 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。)
1485 \let\pxrr@auto@penalty\z@

\pxrr@auto@icspace 文字間の空き。寸法値マクロ。
1486 \let\pxrr@auto@icspace\pxrr@zeropt

\pxrr@intr@amount 進入の幅。寸法値マクロ。
1487 \let\pxrr@intr@amount\pxrr@zeropt

\pxrr@intrude@setauto@j 和文の場合の \pxrr@auto@* の設定。
1488 \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。
1489 \ifpxrr@bnoBr

```

```

1490 \let\pxrr@auto@penalty\@MM
1491 \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。

```

1492 \else
1493 \let\pxrr@auto@penalty\z@
1494 \if :\pxrr@bscomp
1495 \let\pxrr@auto@icspace\pxrr@iaiskip
1496 \else\if .\pxrr@bscomp
1497 \let\pxrr@auto@icspace\pxrr@zeropt
1498 \else
1499 \let\pxrr@auto@icspace\pxrr@iiskip
1500 \fi\fi
1501 \fi
1502 }

```

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

```

1503 \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。

```

1504 \if :\pxrr@bscomp\else
1505 \pxrr@bnobrtrue
1506 \fi
1507 \ifpxrr@bnobr
1508 \let\pxrr@auto@penalty\@MM
1509 \let\pxrr@auto@icspace\pxrr@zeropt
1510 \else

```

この分岐は和欧文間空白挿入指定 (:) に限る。

```

1511 \let\pxrr@auto@penalty\z@
1512 \let\pxrr@auto@icspace\pxrr@iaiskip
1513 \fi
1514 }

```

4.16.1 前側進入処理

`\pxrr@intrude@head` 前側の進入処理。

```

1515 \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。)

```

1516 \ifpxrr@ghost\else

```

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

```

1517 \let\pxrr@intr@amount\pxrr@bspace
1518 \ifdim\pxrr@bintr<\pxrr@intr@amount\relax
1519 \let\pxrr@intr@amount\pxrr@bintr
1520 \fi

```

`\pxrr@auto@*` の設定法は和文ルビと欧文ルビで処理が異なる。

```

1521 \ifpxrr@abody
1522 \pxrr@intrude@setauto@a
1523 \else
1524 \pxrr@intrude@setauto@j
1525 \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)。

```

1526 \ifpxrr@par@head
1527 \ifpxrr@bfintr
1528 \hskip-\pxrr@intr@amount\relax
1529 \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。

```

1530 \else
1531 % \penalty\pxrr@auto@penalty\relax
1532 \hskip-\pxrr@intr@amount\relax
1533 \hskip\pxrr@auto@icspace\relax
1534 \fi
1535 \fi
1536 }

```

\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。

```

1537 \def\pxrr@put@head@penalty{%
1538 \ifpxrr@ghost\else \ifpxrr@par@head\else
1539 \ifpxrr@abody
1540 \pxrr@intrude@setauto@a
1541 \else
1542 \pxrr@intrude@setauto@j
1543 \fi
1544 \ifnum\pxrr@auto@penalty=\z@\else
1545 \pxrr@canta\lastpenalty \unpenalty
1546 \advance\pxrr@canta\pxrr@auto@penalty\relax
1547 \penalty\pxrr@canta
1548 \fi
1549 \fi\fi
1550 }

```

4.16.2 後側進入処理

\pxrr@intrude@end 末尾での進入処理。

```

1551 \def\pxrr@intrude@end{%
1552 \ifpxrr@ghost\else

```

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

```

1553 \let\pxrr@intr@amount\pxrr@aspace
1554 \ifdim\pxrr@aintr<\pxrr@intr@amount\relax
1555 \let\pxrr@intr@amount\pxrr@aintr
1556 \fi

```

\pxrr@auto@* の設定法は和文ルビと欧文ルビで処理が異なる。

```

1557 \pxrr@csletcs{ifpxrr@bnober}{ifpxrr@anober}%
1558 \let\pxrr@bscomp\pxrr@ascomp
1559 \ifpxrr@abody
1560 \pxrr@intrude@setauto@a
1561 \else
1562 \pxrr@intrude@setauto@j
1563 \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。

```

1564 \ifnum\pxrr@auto@penalty=\z@
1565 \let\pxrr@auto@penalty\pxrr@end@kinsoku
1566 \fi
1567 \ifpxrr@afintr

```

段落末尾での進入を許す場合。

```

1568 \ifnum\pxrr@auto@penalty=\z@\else
1569 \penalty\pxrr@auto@penalty\relax
1570 \fi
1571 \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。

```

1572 \hskip\pxrr@auto@icspace\relax
1573 \ifnum\pxrr@auto@penalty=\z@\else
1574 \penalty\@MM
1575 \fi
1576 \else

```

段落末尾での進入を許さない場合。

```

1577 \@tempkipa-\pxrr@intr@amount\relax
1578 \advance\@tempkipa\pxrr@auto@icspace\relax
1579 \ifnum\pxrr@auto@penalty=\z@\else
1580 \penalty\pxrr@auto@penalty\relax
1581 \fi
1582 \hskip\@tempkipa
1583 \ifnum\pxrr@auto@penalty=\z@\else
1584 \penalty\@MM
1585 \fi
1586 \fi
1587 \fi
1588 }

```

4.17 メインです

4.17.1 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。

```
1589 \AtBeginDocument{%
1590   \providecommand*\ruby{\jruby}%
1591 }
1592 \newcommand*\jruby{%
1593   \pxrr@jprologue
1594   \pxrr@trubyfalse
1595   \pxrr@ruby
1596 }
```

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

```
1597 \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。

```
1598 \newcommand*\aruby{%
1599   \pxrr@aprologue
1600   \pxrr@trubyfalse
1601   \pxrr@ruby
1602 }
1603 \pxrr@add@protect\aruby
```

`\truby` 和文両側ルビの公開命令。

```
1604 \newcommand*\truby{%
1605   \pxrr@jprologue
1606   \pxrr@trubytrue
1607   \pxrr@ruby
1608 }
1609 \pxrr@add@protect\truby
```

`\atruby` 欧文両側ルビの公開命令。

```
1610 \newcommand*\atruby{%
1611   \pxrr@aprologue
1612   \pxrr@trubytrue
1613   \pxrr@ruby
1614 }
1615 \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。

```
1616 \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 1617 \let\pxrr@option\@empty
1618 \let\pxrr@exoption\@empty
```

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

```
\pxrr@do@scan 1619 \let\pxrr@do@proc\@empty
                1620 \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。

```
1621 \def\pxrr@ruby{%
1622   \@testopt\pxrr@ruby@a{}%
1623 }
1624 \def\pxrr@ruby@a[#1]{%
1625   \def\pxrr@option{#1}%
1626   \@testopt\pxrr@ruby@b{}%
1627 }
1628 \def\pxrr@ruby@b[#1]{%
1629   \def\pxrr@exoption{#1}%
1630   \ifpxrr@truby
1631     \let\pxrr@do@proc\pxrr@truby@proc
1632     \let\pxrr@do@scan\pxrr@truby@scan
1633   \else
1634     \let\pxrr@do@proc\pxrr@ruby@proc
1635     \let\pxrr@do@scan\pxrr@ruby@scan
1636   \fi
1637   \pxrr@ruby@c
1638 }
1639 \def\pxrr@ruby@c{%
1640   \ifpxrr@ghost
1641     \expandafter\pxrr@do@proc
1642   \else
1643     \expandafter\pxrr@do@scan
1644   \fi
1645 }
```

`\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。

```
1646 \def\pxrr@ruby@proc#1#2{%
1647   \pxrr@prepare@fallback{#1}%
    フォントサイズの変数を設定して、
1648   \pxrr@assign@fsize
    オプションを解析する。
1649   \pxrr@parse@option\pxrr@option
    ルビ文字入力をグループ列に分解する。
1650   \pxrr@decompbar{#2}%
1651   \let\pxrr@ruby@list\pxrr@res
1652   \edef\pxrr@ruby@count{\the\pxrr@cntr}%
    親文字入力をグループ列に分解する。
1653   \pxrr@decompbar{#1}%
1654   \let\pxrr@body@list\pxrr@res
```

```

1655 \edef\pxrr@body@count{\the\pxrr@cntr}%
1656 \ifpxrrDebug
1657 \pxrr@debug@show@input
1658 \fi
1659 \ifpxrr@safe@mode
1660 \pxrr@setup@safe@mode
1661 \fi

```

入力検査を行い、パスした場合は組版処理に進む。

```

1662 \pxrr@if@alive{%
1663 \if g\pxrr@mode
1664 \pxrr@ruby@check@g
1665 \pxrr@if@alive{%
1666 \ifnum\pxrr@body@count>\@ne
1667 \pxrr@ruby@main@g
1668 \else
1669 \pxrr@ruby@main@g
1670 \fi
1671 }%
1672 \else
1673 \pxrr@ruby@check@m
1674 \pxrr@if@alive{\pxrr@ruby@main@m}%
1675 \fi
1676 }%

```

後処理を行う。

```

1677 \pxrr@ruby@exit
1678 }

```

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

```

1679 \def\pxrr@truby@proc#1#2#3{%
1680 \pxrr@prepare@fallback{#1}%

```

フォントサイズの変数を設定して、

```
1681 \pxrr@assign@fsize
```

オプションを解析する。

```
1682 \pxrr@parse@option\pxrr@option
```

両側ルビの場合、入力文字列をグループ分解せずに、そのままの引数列の形でマクロに記憶する。

```

1683 \def\pxrr@all@input{#{1}-#{2}-#{3}}%
1684 \ifpxrrDebug
1685 \pxrr@debug@show@input
1686 \fi
1687 \ifpxrr@safe@mode
1688 \pxrr@setup@safe@mode
1689 \fi

```

入力検査を行い、パスした場合は組版処理に進む。

```

1690 \pxrr@if@alive{%
1691   \pxrr@ruby@check@tg
1692   \pxrr@if@alive{\pxrr@ruby@main@tg}%
1693 }%

```

後処理を行う。

```

1694 \pxrr@ruby@exit
1695 }

```

`\pxrr@setup@safe@mode` 安全モード用の設定。

```

1696 \def\pxrr@setup@safe@mode{%
1697   \let\pxrr@mode=g%
1698   \chardef\pxrr@evensp\z@
1699   \chardef\pxrr@revensp\z@
1700   \chardef\pxrr@fullsize\z@
1701 }

```

4.17.2 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。

```

1702 \def\pxrr@ruby@check@g{%
1703   \ifnum\pxrr@body@count=\pxrr@ruby@count\relax
1704   \ifnum\pxrr@body@count=\@ne\else
1705     \ifpxrr@abody
1706       \pxrr@fatal@bad@movable
1707     \else\ifnum\pxrr@extra=\z@
1708       \pxrr@fatal@na@movable
1709     \fi\fi
1710   \fi
1711 \else
1712   \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
1713 \fi
1714 }

```

`\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。

```

1715 \def\pxrr@ruby@check@m{%
1716   \ifnum\pxrr@body@count=\@ne

```

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

```

1717   \let\pxrr@pre\pxrr@decompose
1718   \let\pxrr@post\relax
1719   \pxrr@body@list
1720   \let\pxrr@body@list\pxrr@res
1721   \edef\pxrr@body@count{\the\pxrr@cntr}%

```

```

1722 \ifnum\pxrr@body@count=\pxrr@ruby@count\relax\else
1723 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
1724 \fi
1725 \else
1726 \pxrr@fatal@bad@mono
1727 \fi
1728 }

```

`\pxrr@ruby@check@tg` 両側ルビの場合、ここで検査する内容は無い。(両側ルビの入力文字列はグループ分割されず、常に単一グループとして扱われる。)

```

1729 \def\pxrr@ruby@check@tg{%
1730 }

```

4.17.3 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。

```

1731 \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。

```

1732 \def\pxrr@check@par@head{%
1733 \ifvmode
1734 \pxrr@par@headtrue
1735 \else
1736 \pxrr@par@headfalse
1737 \fi
1738 }

```

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

```

1739 \def\pxrr@if@last#1#2#3{%
1740 \ifx#3\pxrr@post #1%
1741 \else #2%
1742 \fi
1743 #3%
1744 }

```

`\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。

```

1745 \def\pxrr@inter@mono{%
1746 \hskip\pxrr@iiskip\relax
1747 }

```

`\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。

```
1748 \def\pxrr@takeout@any@protr@nocolor{%
1749   \ifpxrr@any@protr
1750     \aftergroup\pxrr@any@protrtrue
1751   \fi
1752 }
```

color 使用時の定義。

```
1753 \def\pxrr@takeout@any@protr{%
1754   \ifpxrr@any@protr
1755     \aftergroup\pxrr@takeout@any@protr@a
1756   \fi
1757 }
1758 \def\pxrr@takeout@any@protr@a{%
1759   \aftergroup\pxrr@any@protrtrue
1760 }
```

\pxrr@ruby@main@m モノルビ。

```
1761 \def\pxrr@ruby@main@m{%
1762   \pxrr@zip@list\pxrr@body@list\pxrr@ruby@list
1763   \let\pxrr@whole@list\pxrr@res
1764   \pxrr@check@par@head
1765   \pxrr@put@head@penalty
1766   \pxrr@any@protrfalse
1767   \ifpxrr@Debug
1768   \pxrr@debug@show@recomp
1769   \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。

```
1770   \let\pxrr@locate@head@\pxrr@locate@inner
1771   \let\pxrr@locate@end@\pxrr@locate@inner
1772   \let\pxrr@locate@sing@\pxrr@locate@inner
1773   \ifpxrr@aprotr\else
1774     \let\pxrr@locate@end@\pxrr@locate@end
1775     \let\pxrr@locate@sing@\pxrr@locate@end
1776   \fi
1777   \ifpxrr@bprotr\else
1778     \let\pxrr@locate@head@\pxrr@locate@head
1779     \let\pxrr@locate@sing@\pxrr@locate@head
1780   \fi
1781   \def\pxrr@pre##1##2{%
1782     \pxrr@if@last{%
```

単独ブロックの場合。

```
1783     \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
1784     \pxrr@intrude@head
1785     \unhbox\pxrr@boxr
1786     \pxrr@intrude@end
```

```

1787 \pxrr@takeout@any@protr
1788 }{%

```

先頭ブロックの場合。

```

1789 \pxrr@compose@block\pxrr@locate@head@{##1}{##2}%
1790 \pxrr@intrude@head
1791 \unhbox\pxrr@boxr
1792 }%
1793 }%
1794 \def\pxrr@inter##1##2{%
1795 \pxrr@if@last{%

```

末尾ブロックの場合。

```

1796 \pxrr@compose@block\pxrr@locate@end@{##1}{##2}%
1797 \pxrr@inter@mono
1798 \unhbox\pxrr@boxr
1799 \pxrr@intrude@end
1800 \pxrr@takeout@any@protr
1801 }{%

```

中間ブロックの場合。

```

1802 \pxrr@compose@block\pxrr@locate@inner@{##1}{##2}%
1803 \pxrr@inter@mono
1804 \unhbox\pxrr@boxr
1805 }%
1806 }%
1807 \let\pxrr@post\@empty
1808 \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。

```

1809 \if j\pxrr@mode
1810 \ifpxrr@any@protr
1811 \pxrr@ruby@redo@j
1812 \fi
1813 \fi
1814 \unhbox\pxrr@boxr
1815 }

```

`\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。

```

1816 \def\pxrr@ruby@redo@j{%
1817 \pxrr@concat@list\pxrr@body@list
1818 \let\pxrr@body@list\pxrr@res
1819 \pxrr@concat@list\pxrr@ruby@list
1820 \let\pxrr@ruby@list\pxrr@res
1821 \pxrr@zip@single\pxrr@body@list\pxrr@ruby@list
1822 \let\pxrr@whole@list\pxrr@res
1823 \ifpxrr@Debug
1824 \pxrr@debug@show@concat
1825 \fi

```

```

1826 \let\pxrr@locate@sing@\pxrr@locate@inner
1827 \ifpxrr@aprotr\else
1828   \let\pxrr@locate@sing@\pxrr@locate@end
1829 \fi
1830 \ifpxrr@bprotr\else
1831   \let\pxrr@locate@sing@\pxrr@locate@head
1832 \fi
1833 \def\pxrr@pre##1##2{%
1834   \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
1835   \pxrr@intrude@head
1836   \unhbox\pxrr@boxr
1837   \pxrr@intrude@end
1838 }%
1839 \let\pxrr@inter\@undefined
1840 \let\pxrr@post\@empty
1841 \setbox\pxrr@boxr\pxrr@hbox{\pxrr@whole@list}%
1842 }

```

\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。

```

1843 \def\pxrr@ruby@main@gf%
1844   \pxrr@zip@list\pxrr@body@list\pxrr@ruby@list
1845   \let\pxrr@whole@list\pxrr@res
1846   \pxrr@check@par@head
1847   \pxrr@put@head@penalty
1848 \ifpxrr@Debug
1849 \pxrr@debug@show@recomp
1850 \fi
1851 \let\pxrr@locate@sing@\pxrr@locate@inner
1852 \ifpxrr@aprotr\else
1853   \let\pxrr@locate@sing@\pxrr@locate@end
1854 \fi
1855 \ifpxrr@bprotr\else
1856   \let\pxrr@locate@sing@\pxrr@locate@head
1857 \fi
1858 \def\pxrr@pre##1##2{%
1859   \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
1860   \pxrr@intrude@head
1861   \unhbox\pxrr@boxr
1862   \pxrr@intrude@end
1863 }%
1864 \let\pxrr@inter\@undefined
1865 \let\pxrr@post\@empty

```

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

```

1866 \pxrr@whole@list
1867 }

```

`\pxrr@ruby@main@tg` 両側ルビ（必ず単純グループルビである）の場合。

```
1868 \def\pxrr@ruby@main@tg{%
1869   \pxrr@check@par@head
1870   \pxrr@put@head@penalty
1871   \let\pxrr@locate@sing@\pxrr@locate@inner
1872   \ifpxrr@aprotr\else
1873     \let\pxrr@locate@sing@\pxrr@locate@end
1874   \fi
1875   \ifpxrr@bprotr\else
1876     \let\pxrr@locate@sing@\pxrr@locate@head
1877   \fi
1878   \expandafter\pxrr@compose@twoside@block\expandafter\pxrr@locate@sing@
1879   \pxrr@all@input
1880   \pxrr@intrude@head
1881   \unhbox\pxrr@boxr
1882   \pxrr@intrude@end
1883 }
```

4.17.4 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。

```
1884 \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。

```
1885 \pxrr@jchardef\pxrr@zspace=\pxrr@jc{2121:3000}
```

`\pxrr@jprologue` 和文ルビ用の開始処理。

```
1886 \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。

```
1887   \ifpxrr@jghost
1888     \pxrr@zspace
1889   \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。

```
1890   \begingroup
1891     \pxrr@abodyfalse
1892     \pxrr@csletcs{ifpxrr@ghost}{ifpxrr@jghost}%
```

出力した全角空白の幅だけ戻しておく。

```
1893     \ifpxrr@jghost
1894       \setbox\pxrr@boxa\hbox{\pxrr@zspace}%
1895       \kern-\wd\pxrr@boxa
1896     \fi
1897 }
```

`\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 T_EX 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。)

```

1898 \def\pxrr@setup@aghost{%
1899   \global\let\pxrr@setup@aghost\relax
1900   \IfFileExists{t1lmr.fd}{%
1901     \begingroup
1902       \fontsize{2.5}{0}\usefont{T1}{lmr}{m}{n}%
1903     \endgroup
1904     \global\pxrr@letcs\pxrr@aghostfont{T1/lmr/m/n/2.5}%
1905     \global\chardef\pxrr@aghostchar=23 % compwordmark
1906     \gdef\pxrr@aghost{\pxrr@aghostfont\pxrr@aghostchar}%
1907     \global\xsocode\pxrr@aghostchar=3 %
1908   }{%else
1909     \pxrr@warn{Ghost embedding for \string\aruby\space
1910       is disabled,\MessageBreak
1911       since package lmodern is missing}%
1912     \global\pxrr@aghostfalse
1913     \global\let\pxrr@aghosttrue\relax
1914   }%
1915 }

```

`\pxrr@aprologue` 欧文ルビ用の開始処理。

```

1916 \def\pxrr@aprologue{%
1917   \ifpxrr@aghost
1918     \pxrr@aghost
1919     \fi
1920   \begingroup
1921     \pxrr@abodytrue
1922     \pxrr@csletcs{ifpxrr@ghost}{ifpxrr@aghost}%
1923 }

```

4.17.5 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。

```

1924 \def\pxrr@ruby@exit{%
1925   \ifpxrr@fatal@error
1926     \pxrr@fallback
1927     \fi
1928   \ifpxrr@abody
1929     \expandafter\pxrr@aepilogue

```

```

1930 \else
1931   \expandafter\pxrr@jepilogue
1932 \fi
1933 }

```

`\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。

```

1934 \def\pxrr@jepilogue{%
1935   \ifpxrr@jghost
1936     \setbox\pxrr@boxa\hbox{\pxrr@zspace}%
1937     \kern-\wd\pxrr@boxa
1938   \fi

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

1939 \endgroup
1940 \ifpxrr@jghost
1941   \pxrr@zspace
1942 \fi
1943 }

```

`\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。

```

1944 \def\pxrr@aepilogue{%
1945   \endgroup
1946   \ifpxrr@aghost
1947     \pxrr@aghost
1948   \fi
1949 }

```

4.18 デバッグ用出力

```

1950 \def\pxrr@debug@show@input{%
1951   \typeout{---\pxrr@pkgname\space input:^^J%
1952     ifpxrr@abody = \meaning\ifpxrr@abody^^J%
1953     ifpxrr@truby = \meaning\ifpxrr@truby^^J%
1954     pxrr@ruby@fsize = \pxrr@ruby@fsize^^J%
1955     pxrr@body@zw = \pxrr@body@zw^^J%
1956     pxrr@ruby@zw = \pxrr@ruby@zw^^J%
1957     pxrr@iiskip = \pxrr@iiskip^^J%
1958     pxrr@iaiskip = \pxrr@iaiskip^^J%
1959     pxrr@htratio = \pxrr@htratio^^J%
1960     pxrr@ruby@raise = \pxrr@ruby@raise^^J%
1961     pxrr@ruby@lower = \pxrr@ruby@lower^^J%
1962     ifpxrr@bprotr = \meaning\ifpxrr@bprotr^^J%
1963     ifpxrr@aprotr = \meaning\ifpxrr@aprotr^^J%
1964     pxrr@side = \the\pxrr@side^^J%
1965     pxrr@evensp = \the\pxrr@evensp^^J%
1966     pxrr@fullsize = \the\pxrr@fullsize^^J%
1967     pxrr@bscomp = \meaning\pxrr@bscomp^^J%
1968     pxrr@ascomp = \meaning\pxrr@ascomp^^J%
1969     ifpxrr@bnobr = \meaning\ifpxrr@bnobr^^J%

```

```

1970   ifpxrr@anobr = \meaning\ifpxrr@anobr^^J%
1971   ifpxrr@bfintr = \meaning\ifpxrr@bfintr^^J%
1972   ifpxrr@afintr = \meaning\ifpxrr@afintr^^J%
1973   pxrr@bintr = \pxrr@bintr^^J%
1974   pxrr@aintr = \pxrr@aintr^^J%
1975   pxrr@ahead = \the\pxrr@ahead^^J%
1976   pxrr@mode = \meaning\pxrr@mode^^J%
1977   ifpxrr@ahead@given = \meaning\ifpxrr@ahead@given^^J%
1978   ifpxrr@mode@given = \meaning\ifpxrr@mode@given^^J%
1979   pxrr@body@list = \meaning\pxrr@body@list^^J%
1980   pxrr@body@count = \@nameuse{pxrr@body@count}^^J%
1981   pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
1982   pxrr@ruby@count = \@nameuse{pxrr@ruby@count}^^J%
1983   pxrr@end@kinsoku = \pxrr@end@kinsoku^^J%
1984   ----
1985 }%
1986 }
1987 \def\pxrr@debug@show@recomp{%
1988   \typeout{----\pxrr@pkgname\space recomp:^^J%
1989     pxrr@body@list = \meaning\pxrr@body@list^^J%
1990     pxrr@body@count = \pxrr@body@count^^J%
1991     pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
1992     pxrr@ruby@count = \pxrr@ruby@count^^J%
1993     pxrr@res = \meaning\pxrr@res^^J%
1994     ----
1995 }%
1996 }
1997 \def\pxrr@debug@show@concat{%
1998   \typeout{----\pxrr@pkgname\space concat:^^J%
1999     pxrr@body@list = \meaning\pxrr@body@list^^J%
2000     pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
2001     pxrr@whole@list = \meaning\pxrr@whole@list^^J%
2002     ----
2003 }%
2004 }

```