

pxrubrica パッケージ

八登 崇之 (Takayuki YATO; aka “ZR”)

v0.1+ [2011/07/27]

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	4
2.6	拡張機能	5
2.7	拡張機能設定の命令	6
3	実装	6
3.1	前提パッケージ	6
3.2	エラーメッセージ	6
3.3	パラメタ	8
3.3.1	全般設定	8
3.3.2	ルビ呼出時の設定	10
3.4	補助手続	11
3.4.1	雑多な定義	11
3.4.2	数値計算	13
3.4.3	リスト分解	14
3.5	エンジン依存処理	18
3.6	パラメタ設定公開命令	21
3.7	ルビオプション解析	23
3.8	オプション整合性検査	29
3.9	フォントサイズ	31
3.10	ルビ用均等割り	32
3.11	小書き仮名の変換	35

3.12	ブロック毎の組版	37
3.13	命令の頑強化	41
3.14	致命的エラー対策	41
3.15	先読み処理	42
3.16	進入処理	44
3.16.1	前側進入処理	45
3.16.2	後側進入処理	46
3.17	メインです	47
3.17.1	エントリーポイント	47
3.17.2	入力検査	50
3.17.3	ルビ組版処理	51
3.17.4	前処理	55
3.17.5	後処理	57

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、和文ルビ (\ruby) では和文間空白、欧文ルビ (\rubby) の場合はゼロである。* 無指定の場合の行分割の可否 (ペナルティ) は p_TE_X 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。(以上の説明はゴースト処理無効の場合。)

〈モード〉は以下の値の何れか。

-	(無指定)	m	(< mono)	モノルビ	
h	(< head)	肩付き	g	(< group)	グループルビ
c	(< center)	中付き	j	(< jukugo)	熟語ルビ
P	(< primary)	上側配置			
S	(< secondary)	下側配置			

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。

〈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、和文ルビでは和文間空白、欧文ルビの場合はゼロである。
! 無指定の場合、段落末尾では進入が抑止される。* 無指定の場合の行分割の可否（ペナルティ）は pTeX 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）（以上の説明はゴースト処理無効の場合。）

〈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 ($\text{\begin{document}}$) 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。⟨オプション⟩ の指定方法は \ruby と同じだが、欧文ルビは常に (単純) グループルビとなるので、m、g、j の指定は無視される。
- \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[m]{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（詳しくは後述）。

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。）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ので、ルビ命令のオプションの：が不要になる。

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。

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。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」である。[既定 = |cjp|]
- `\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]

2.6 拡張機能

「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、 $\text{T}_{\text{E}}\text{X}$ の実現は非常に難しい。これを $\varepsilon\text{-pTeX}$ の拡張機能を用いて何とか実現したい。できたらいいな。

- 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。

2.7 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]

3 実装

3.1 前提パッケージ

`keyval` を使う予定 (まだ使っていない)。

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

3.2 エラーメッセージ

`\pxrr@error` エラー出力命令。

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

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

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

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

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

`\pxrr@eh@fatal` 致命的エラーのヘルプ。

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

`\pxrr@warn@must@group` 欧文ルビ、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     Movable 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@interror` 内部エラー。これが出てはいけない。:-)

```
67 \def\pxrr@interror#1{%
68   \pxrr@fatal@error{INTERNAL ERROR (#1)}%
69   \pxrr@eh@fatal
70 }
```

`\ifpxrrDebug` デバッグモード指定。

```
71 \newif\ifpxrrDebug
```

3.3 パラメタ

3.3.1 全般設定

`\pxrr@ruby@font` ルビ用フォント切替命令。

```
72 \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)。
73 `\def\pxrr@big@intr{1}`
74 `\def\pxrr@small@intr{0.5}`

`\pxrr@size@ratio` ルビ文字サイズ(`\rubysizeratio`)。実数値マクロ。
75 `\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` 76 `\def\pxrr@sprop@x{1}`
77 `\def\pxrr@sprop@y{2}`
`\pxrr@sprop@z` 78 `\def\pxrr@sprop@z{1}`

`\pxrr@sprop@hy` 伸縮配置比率(`\rubystretchprophead`)。実数値マクロ。
`\pxrr@sprop@hz` 79 `\def\pxrr@sprop@hy{1}`
80 `\def\pxrr@sprop@hz{1}`

`\pxrr@sprop@ex` 伸縮配置比率(`\rubystretchpropend`)。実数値マクロ。
`\pxrr@sprop@ey` 81 `\def\pxrr@sprop@ex{1}`
82 `\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。
83 `\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。
84 `\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。
85 `\def\pxrr@thtratio{0.5}`

`\pxrr@extra` 拡張機能実装方法(`\rubyuseextra`)。整数定数。
86 `\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。
87 `\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。
88 `\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。
89 `\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。
90 `\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。
91 `\newif\ifpxrr@break@jukugo \pxrr@edge@adjustfalse`

`\ifpxrr@d@bprotr` 突出を許すか否か。`\rubysetup` の〈前設定〉/〈後設定〉に由来する。スイッチ。

`\ifpxrr@d@aprotr` 92 `\newif\ifpxrr@d@bprotr \pxrr@d@bprotrtrue`
 93 `\newif\ifpxrr@d@aprotr \pxrr@d@aprotrtrue`

`\pxrr@d@bintr` 進入力。`\rubysetup` の〈前設定〉/〈後設定〉に由来する。`\pxrr@XXX@intr` または空（進
`\pxrr@d@aintr` 入無し）に展開されるマクロ。
 94 `\def\pxrr@d@bintr{}`
 95 `\def\pxrr@d@aintr{}`

`\ifpxrr@d@athead` 肩付き / 中付きの設定。`\rubysetup` の `c/h/H` の設定。整数定数：`0` = 中付き (`c`); `1` =
 肩付き (`h`); `2` = 拡張肩付き (`H`)
 96 `\chardef\pxrr@d@athead=0`

`\pxrr@d@mode` モノルビ (`m`)・グループルビ (`g`)・熟語ルビ (`j`) のいずれか。`\rubysetup` の設定値。オブ
 ション文字への暗黙の (`\let` された) 文字トークン。
 97 `\let\pxrr@d@mode=j`

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

`\pxrr@d@evensp` 親文字列が短い場合に均等割りを行うか。`0` = 行わない; `1` = 行う。`\rubysetup` の `e/E`
 の設定。整数定数。
 99 `\chardef\pxrr@d@evensp=1`

`\pxrr@d@fullsize` `\rubysetup` の `f/F` の設定。整数定数。
 100 `\chardef\pxrr@d@fullsize=0`

3.3.2 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` 101 `\newif\ifpxrr@bprotr \pxrr@bprotrfalse`
 102 `\newif\ifpxrr@aprotr \pxrr@aprotrfalse`

`\pxrr@bintr` 進入力。`\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` 103 `\def\pxrr@bintr{}`
 104 `\def\pxrr@aintr{}`

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

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

```
109 \newif\ifpxrr@bfintr \pxrr@bfintrfalse
110 \newif\ifpxrr@afintr \pxrr@afintrfalse
```

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

```
111 \chardef\pxrr@athead=0
```

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

```
112 \let\pxrr@mode=\@undefined
```

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

```
113 \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。

```
114 \chardef\pxrr@side=0
```

`\pxrr@evensp` 親文字列が短い場合に均等割りを行うか。0 = 行わない ; 1 = 行う。`\ruby` の `e / E` の設定。整数定数。

```
115 \chardef\pxrr@evensp=1
```

`\pxrr@fullsize` `\ruby` の `f / F` の設定。整数定数。

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

3.4 補助手続

3.4.1 雑多な定義

`\ifpxrr@ok` 汎用スイッチ。

```
117 \newif\ifpxrr@ok
```

`\pxrr@cmta` 汎用の整数レジスタ。

```
118 \newcount\pxrr@cmta
```

`\pxrr@cntr` 結果を格納する整数レジスタ。

```
119 \newcount\pxrr@cntr
```

`\pxrr@dima` 汎用の寸法レジスタ。

```
120 \newdimen\pxrr@dima
```

`\pxrr@boxa` 汎用のボックスレジスタ。

```
121 \newbox\pxrr@boxa
122 \newbox\pxrr@boxb
```

`\pxrr@boxr` 結果を格納するボックスレジスタ。

```
123 \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。

```
124 \chardef\pxrr@zero=0
```

`\pxrr@zeropt` 「0pt」という文字列。寸法値マクロへの代入に用いる。

```
125 \def\pxrr@zeropt{0pt}
```

`\pxrr@hfilx` `\pxrr@hfilx{〈実数〉}`：「〈実数〉fil」のグル を置く。

```
126 \def\pxrr@hfilx#1{%
127   \hskip\z@\@plus #1fil\relax
128 }
```

`\pxrr@res` 結果を格納するマクロ。

```
129 \let\pxrr@res\@empty
```

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

```
130 \def\pxrr@ifx#1{%
131   \ifx#1\expandafter\@firstoftwo
132   \else\expandafter\@secondoftwo
133   \fi
134 }
```

`\pxrr@ifnum` `\pxrr@ifnum{〈引数〉}{真}{偽}`：`\ifnum`〈引数〉を行うテスト。

```
135 \def\pxrr@ifnum#1{%
136   \ifnum#1\expandafter\@firstoftwo
137   \else\expandafter\@secondoftwo
138   \fi
139 }
```

`\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。

```
140 \def\pxrr@cslet#1{%
141   \expandafter\let\csname#1\endcsname
142 }
143 \def\pxrr@letcs#1#2{%
144   \expandafter\let\expandafter#1\csname#2\endcsname
145 }
146 \def\pxrr@csletcs#1#2{%
147   \expandafter\let\csname#1\endcsname
148   \csname#2\endcsname
149 }
```

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

```
150 \def\pxrr@setok#1{%
151   #1{\pxrr@oktrue}{\pxrr@okfalse}%
152 }
```

`\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。

```
153 \def\pxrr@appto#1#2{%
154   \expandafter\def\expandafter#1\expandafter{#1#2}%
155 }
```

`\pxrr@nil` ユニークトークン。

```
\pxrr@end 156 \def\pxrr@nil{\noexpand\pxrr@nil}
157 \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。

```
158 \def\pxrr@without@macro@trace#1{%
159   \chardef\pxrr@tracingmacros=\tracingmacros
160   \tracingmacros\z@
161   #1%
162   \tracingmacros\pxrr@tracingmacros
163 }
```

3.4.2 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。

```
164 \mathchardef\pxrr@invscale@ca=259
165 \def\pxrr@invscale#1#2{%
166   \begingroup
167     \@tempdima=#1\relax
168     \@tempdimb#2\p@\relax
169     \@tempcnta\@tempdima
170     \multiply\@tempcnta\@ccclvi
171     \divide\@tempcnta\@tempdimb
172     \multiply\@tempcnta\@ccclvi
173     \@tempcntb\p@
174     \divide\@tempcntb\@tempdimb
175     \advance\@tempcnta-\@tempcntb
176     \advance\@tempcnta-\tw@
177     \@tempdimb\@tempcnta\@ne
178     \advance\@tempcnta\@tempcntb
179     \advance\@tempcnta\@tempcntb
180     \advance\@tempcnta\pxrr@invscale@ca
181     \@tempdimc\@tempcnta\@ne
182     \@whiledim\@tempdimb<\@tempdimc\do{%
183       \@tempcntb\@tempdimb
184       \advance\@tempcntb\@tempdimc
185       \advance\@tempcntb\@ne
186       \divide\@tempcntb\@tw@
187       \ifdim #2\@tempcntb>\@tempdima
188         \advance\@tempcntb\@m@ne
189         \@tempdimc=\@tempcntb\@ne
```

```

190     \else
191     \@tempdimb=\@tempcntb\@ne
192     \fi}%
193     \xdef\pxrr@gtmpa{\the\@tempdimb}%
194     \endgroup
195     #1=\pxrr@gtmpa\relax
196 }

```

`\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 pt, X_n iu] の範囲外では両端の 2 点による外挿を行う。

```

197 \def\pxrr@interpolate#1#2#3#4#5{%
198   \edef\pxrr@tempa{#1}%
199   \edef\pxrr@tempb{#2}%
200   \def\pxrr@tempd{#3}%
201   \setlength{\@tempdima}{#4}%
202   \edef\pxrr@tempc{(0,0)#5(*,*)}%
203   \expandafter\pxrr@interpolate@a\pxrr@tempc\@nil
204 }
205 \def\pxrr@interpolate@a(#1,#2)(#3,#4)(#5,#6){%
206   \if#5%
207     \def\pxrr@tempc{\pxrr@interpolate@b{#1}{#2}{#3}{#4}}%
208   \else\ifdim\@tempdima<#3\pxrr@tempa
209     \def\pxrr@tempc{\pxrr@interpolate@b{#1}{#2}{#3}{#4}}%
210   \else
211     \def\pxrr@tempc{\pxrr@interpolate@a(#3,#4)(#5,#6)}%
212   \fi\fi
213   \pxrr@tempc
214 }
215 \def\pxrr@interpolate@b#1#2#3#4#5\@nil{%
216   \@tempdimb=-#1\pxrr@tempa
217   \advance\@tempdima\@tempdimb
218   \advance\@tempdimb#3\pxrr@tempa
219   \edef\pxrr@tempc{\strip@pt\@tempdimb}%
220   \pxrr@invscale\@tempdima\pxrr@tempc
221   \edef\pxrr@tempc{\strip@pt\@tempdima}%
222   \@tempdima=#4\pxrr@tempb
223   \@tempdimb=#2\pxrr@tempb
224   \advance\@tempdima-\@tempdimb
225   \@tempdima=\pxrr@tempc\@tempdima
226   \advance\@tempdima\@tempdimb
227   \pxrr@tempd=\@tempdima
228 }

```

3.4.3 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 (`{...}` 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)。

```
229 \def\pxrr@decompose#1{%
230   \let\pxrr@res\@empty
231   \pxrr@cntr=\z@
232   \pxrr@decompose@loopa#1\pxrr@end
233 }
234 \def\pxrr@decompose@loopa{%
235   \futurelet\pxrr@tempa\pxrr@decompose@loopb
236 }
237 \def\pxrr@decompose@loopb{%
238   \pxrr@ifx{\pxrr@tempa\pxrr@end}{%
239     \pxrr@appto\pxrr@res{\pxrr@post}%
240   }{%
241     \pxrr@setok{\pxrr@ifx{\pxrr@tempa\bgroup}}%
242     \pxrr@decompose@loopc
243   }%
244 }
245 \def\pxrr@decompose@loopc#1{%
246   \ifx\pxrr@res\@empty
247     \def\pxrr@res{\pxrr@pre}%
248   \else
249     \pxrr@appto\pxrr@res{\pxrr@inter}%
250   \fi
251   \ifpxrr@ok
252     \pxrr@appto\pxrr@res{{\#1}}%
253   \else
254     \pxrr@appto\pxrr@res{{\#1}}%
255   \fi
256   \advance\pxrr@cntr\@ne
257   \pxrr@decompose@loopa
258 }
```

`\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。

```
259 \def\pxrr@decompbar#1{%
260   \let\pxrr@res\@empty
261   \pxrr@cntr=\z@
```



```

262 \pxrr@decompbar@loopa\pxrr@nil#1|\pxrr@end|{%
263 }
264 \def\pxrr@decompbar@loopa#1|{%
265 \expandafter\pxrr@decompbar@loopb\expandafter{\@gobble#1}%
266 }
267 \def\pxrr@decompbar@loopb#1{%
268 \pxrr@decompbar@loopc#1\relax\pxrr@nil{#1}%
269 }
270 \def\pxrr@decompbar@loopc#1#2\pxrr@nil#3{%
271 \pxrr@ifx{#1\pxrr@end}{%
272 \pxrr@appto\pxrr@res{\pxrr@post}%
273 }{%
274 \ifx\pxrr@res\@empty
275 \def\pxrr@res{\pxrr@pre}%
276 \else
277 \pxrr@appto\pxrr@res{\pxrr@inter}%
278 \fi
279 \pxrr@appto\pxrr@res{#3}%
280 \advance\pxrr@cntr\@ne
281 \pxrr@decompbar@loopa\pxrr@nil
282 }%
283 }

```

\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

```

```

284 \def\pxrr@zip@list#1#2{%
285 \let\pxrr@res\@empty
286 \let\pxrr@post\relax
287 \let\pxrr@tempa#1\pxrr@appto\pxrr@tempa{}}%
288 \let\pxrr@tempb#2\pxrr@appto\pxrr@tempb{}}%
289 \pxrr@zip@list@loopa
290 }
291 \def\pxrr@zip@list@loopa{%
292 \expandafter\pxrr@zip@list@loopb\pxrr@tempa\pxrr@end
293 }
294 \def\pxrr@zip@list@loopb#1#2#3\pxrr@end{%
295 \pxrr@ifx{#1\relax}{%
296 \pxrr@zip@list@exit
297 }{%
298 \pxrr@appto\pxrr@res{#1{#2}}%
299 \def\pxrr@tempa{#3}%
300 \expandafter\pxrr@zip@list@loopc\pxrr@tempb\pxrr@end

```

```

301 }%
302 }
303 \def\pxrr@zip@list@loopc#1#2#3\pxrr@end{%
304   \pxrr@ifx{#1\relax}{%
305     \pxrr@interror{zip}%
306     \pxrr@appto\pxrr@res{}}%
307   \pxrr@zip@list@exit
308 }{%
309   \pxrr@appto\pxrr@res{#2}}%
310   \def\pxrr@tempb{#3}%
311   \pxrr@zip@list@loopa
312 }%
313 }
314 \def\pxrr@zip@list@exit{%
315   \pxrr@appto\pxrr@res{\pxrr@post}%
316 }

```

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

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

の時に、`\pxrr@res` を以下の内容に定義する。

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

```

317 \def\pxrr@concat@list#1{%
318   \let\pxrr@res\@empty
319   \def\pxrr@pre##1{%
320     \pxrr@appto\pxrr@res{##1}%
321   }%
322   \let\pxrr@inter\pxrr@pre
323   \let\pxrr@post\relax
324   #1%
325 }

```

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

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

の時に、`\pxrr@res` を以下の内容に定義する。

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

```

326 \def\pxrr@zip@single#1#2{%
327   \expandafter\pxrr@zip@single@a\expandafter#1#2\pxrr@end
328 }
329 \def\pxrr@zip@single@a#1{%
330   \expandafter\pxrr@zip@single@b#1\pxrr@end
331 }
332 \def\pxrr@zip@single@b#1\pxrr@end#2\pxrr@end{%
333   \def\pxrr@res{\pxrr@pre{#1}\pxrr@post}%
334 }

```

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

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

の時に、`\pxrr@res` を以下の内容に定義する。

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

```

335 \def\pxrr@tzip@single#1#2#3{%
336   \expandafter\pxrr@tzip@single@a\expandafter#1\expandafter#2#3\pxrr@end
337 }
338 \def\pxrr@tzip@single@a#1#2{%
339   \expandafter\pxrr@tzip@single@b\expandafter#1#2\pxrr@end
340 }
341 \def\pxrr@tzip@single@b#1{%
342   \expandafter\pxrr@tzip@single@c#1\pxrr@end
343 }
344 \def\pxrr@tzip@single@c#1\pxrr@end#2\pxrr@end#3\pxrr@end{%
345   \def\pxrr@res{\pxrr@pre{#1}{#2}{#3}\pxrr@post}%
346 }

```

3.5 エンジン依存処理

この小節のマクロ内で使われる変数。

```

347 \let\pxrr@x@tempa\@empty
348 \newif\ifpxrr@x@swa

```

`\pxrr@ifprimitive \pxrr@ifprimitive\CS{ \langle 真 \rangle }{ \langle 偽 \rangle }: \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。`

```

349 \def\pxrr@ifprimitive#1{%
350   \edef\pxrr@x@tempa{\string#1}%
351   \edef\pxrr@x@tempb{\meaning#1}%
352   \ifx\pxrr@x@tempa\pxrr@x@tempb \expandafter\@firstoftwo
353   \else \expandafter\@secondoftwo
354   \fi
355 }

```

`\ifpxrr@in@ptex` エンジンが $\mathrm{pT_E X}$ 系 ($\mathrm{upT_E X}$ 系を含む) であるか。 `\kansuji` のプリミティブテストで判定する。

```

356 \pxrr@ifprimitive\kansuji{%
357   \pxrr@csletcs{ifpxrr@in@ptex}{iftrue}%
358 }{%
359   \pxrr@csletcs{ifpxrr@in@ptex}{iffalse}%
360 }

```

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

```

361 \pxrr@ifprimitive\enablecjktoken{%
362   \pxrr@csletcs{ifpxrr@in@uptex}{iftrue}%
363 }{%

```

```

364 \pxrr@csletcs{ifpxrr@in@uptex}{iffalse}%
365 }

\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。
366 \pxrr@ifprimitive\XeTeXrevision{%
367 \pxrr@csletcs{ifpxrr@in@xetex}{iftrue}%
368 }{%
369 \pxrr@csletcs{ifpxrr@in@xetex}{iffalse}%
370 }

\ifpxrr@in@unicode 「和文」内部コードが Unicode であるか。
371 \ifpxrr@in@xetex
372 \pxrr@csletcs{ifpxrr@in@unicode}{iftrue}%
373 \else\ifpxrr@in@uptex
374 \pxrr@csletcs{ifpxrr@in@unicode}{iftrue}%
375 \else
376 \pxrr@csletcs{ifpxrr@in@unicode}{iffalse}%
377 \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〉:〈Unicode16 進 4 桁〉」の形式。
378 \def\pxrr@jc#1{%
379 \pxrr@jc@a#1\pxrr@nil
380 }
381 \ifpxrr@in@unicode
382 \def\pxrr@jc@a#1:#2\pxrr@nil{%
383 "#2\space
384 }
385 \else\ifpxrr@in@ptex
386 \def\pxrr@jc@a#1:#2\pxrr@nil{%
387 \jis"#1\space\space
388 }
389 \else
390 \def\pxrr@jc@a#1:#2\pxrr@nil{%
391 '?\space
392 }
393 \fi\fi

\pxrr@jchardef 和文用の \chardef。
394 \ifpxrr@in@uptex
395 \let\pxrr@jchardef\kchardef
396 \else
397 \let\pxrr@jchardef\chardef
398 \fi

\ifpxrr@in@tate 縦組であるか。
pTEX 以外での縦組をサポートする予定はない。
399 \ifpxrr@in@ptex
400 \pxrr@csletcs{ifpxrr@in@tate}{iftdir}

```

```

401 \else
402   \pxrr@csletcs{ifpxrr@in@tate}{iffalse}
403 \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。

```

404 \def\pxrr@get@jchar@token#1#2{%
405   \begingroup
406     \kansujichar\@ne=#2\relax
407     \xdef\pxrr@x@gtempa{\kansuji\@ne}%
408   \endgroup
409   \let#1\pxrr@x@gtempa
410 }
411 \ifpxrr@in@unicode\else
412   \ifpxrr@in@ptex\else
413     \def\pxrr@get@jchar@token#1#2{%
414       \def#1{?}%
415     }
416   \fi
417 \fi

```

`\pxrr@x@K` 適当な漢字（実際は ー）のトークン。

```

418 \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。

```

419 \ifpxrr@in@ptex
420   \def\pxrr@get@iiskip#1{%
421     \pxrr@x@swafalse
422     \begingroup
423       \inhibitxspcode\pxrr@x@K\thr@@
424       \kanjiskip\p@
425       \setbox\z@\hbox{\noautospace\pxrr@x@K\pxrr@x@K}%
426       \setbox\tw@\hbox{\pxrr@x@K\pxrr@x@K}%
427       \ifdim\wd\tw@>\wd\z@
428         \aftergroup\pxrr@x@swatrue
429       \fi
430     \endgroup
431     \edef#1{%
432       \ifpxrr@x@swa \the\kanjiskip
433       \else \pxrr@zeropt
434       \fi
435     }%
436   }
437 \else
438   \def\pxrr@get@iiskip#1{%
439     \let#1\pxrr@zeropt
440   }
441 \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。

```
442 \ifpxrr@in@ptex
443   \def\pxrr@get@iaiskip#1{%
444     \pxrr@x@swafalse
445     \begingroup
446       \inhibitxspcode\pxrr@x@K\thr@@ \xspcode'X=\thr@@
447       \xkanjiskip\p@
448       \setbox\z@\hbox{\noautoxspacing\pxrr@x@K X}%
449       \setbox\tw@\hbox{\pxrr@x@K X}%
450       \ifdim\wd\tw@>\wd\z@
451         \aftergroup\pxrr@x@swatrue
452       \fi
453     \endgroup
454     \edef#1{%
455       \ifpxrr@x@swa \the\xkanjiskip
456       \else \pxrr@zeropt
457       \fi
458     }%
459   }
460 \else
461   \def\pxrr@get@iaiskip#1{%
462     \let#1\pxrr@zeropt
463   }
464 \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。

```
465 \ifpxrr@in@ptex
466   \def\pxrr@get@zwidth#1{%
467     \@tempdima=1zw\relax
468     \edef#1{\the\@tempdima}%
469   }
470 \else
471   \def\pxrr@get@zwidth#1{%
472     \@tempdima=1em\relax
473     \edef#1{\the\@tempdima}%
474   }
475 \fi
```

3.6 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。

```
476 \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。

```
477 \newcommand*\rubyssetup[1]{%
478   \pxrr@in@setuptrue
479   \pxrr@fatal@errorfalse
```

```

480 \pxrr@parse@option{#1}%
481 \ifpxrr@fatal@error\else
482 \pxrr@csletcs{ifpxrr@d@bprotr}{ifpxrr@bprotr}%
483 \pxrr@csletcs{ifpxrr@d@aprotr}{ifpxrr@aprotr}%
484 \let\pxrr@d@bintr\pxrr@bintr@
485 \let\pxrr@d@aintr\pxrr@aintr@
486 \let\pxrr@d@athead\pxrr@athead
487 \let\pxrr@d@mode\pxrr@mode
488 \let\pxrr@d@side\pxrr@side
489 \let\pxrr@d@evensp\pxrr@evensp
490 \let\pxrr@d@fullsize\pxrr@fullsize
491 \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。

```

492 \pxrr@in@setupfalse
493 }

```

\rubyfontsetup 対応するパラメタを設定する。

```

494 \newcommand*\rubyfontsetup{}
495 \def\rubyfontsetup{%
496 \def\pxrr@ruby@font
497 }

```

\rubybigintrusion 対応するパラメタを設定する。

```

\rubysmallintrusion 498 \newcommand*\rubybigintrusion[1]{%
\rubymaxmargin 499 \edef\pxrr@big@intr{#1}%
500 }
\rubyintergap 501 \newcommand*\rubysmallintrusion[1]{%
\rubysizeratio 502 \edef\pxrr@small@intr{#1}%
503 }
504 \newcommand*\rubymaxmargin[1]{%
505 \edef\pxrr@maxmargin{#1}%
506 }
507 \newcommand*\rubyintergap[1]{%
508 \edef\pxrr@inter@gap{#1}%
509 }
510 \newcommand*\rubysizeratio[1]{%
511 \edef\pxrr@size@ratio{#1}%
512 }

```

\rubyusejghost 対応するスイッチを設定する。

```

\rubynousejghost 513 \newcommand*\rubyusejghost{%
514 \pxrr@jghosttrue
515 }
516 \newcommand*\rubynousejghost{%
517 \pxrr@jghostfalse
518 }

```

```

\rubyuseaghost 対応するスイッチを設定する。
\rubynouseaghost 519 \newcommand*\rubyuseaghost{%
                    520   \pxrr@aghosttrue
                    521 }
                    522 \newcommand*\rubynouseaghost{%
                    523   \pxrr@aghostfalse
                    524 }

\rubyadjustatlineedge 対応するスイッチを設定する。
\rubynoadjustatlineedge 525 \newcommand*\rubyadjustatlineedge{%
                        526   \pxrr@edge@adjusttrue
                        527 }
                        528 \newcommand*\rubynoadjustatlineedge{%
                        529   \pxrr@edge@adjustfalse
                        530 }

\rubybreakjukugo 対応するスイッチを設定する。
\rubynobreakjukugo 531 \newcommand*\rubybreakjukugo{%
                    532   \pxrr@break@jukugotrue
                    533 }
                    534 \newcommand*\rubynobreakjukugo{%
                    535   \pxrr@break@jukugofalse
                    536 }

\rubystretchprop 対応するパラメタを設定する。
\rubystretchprophead 537 \newcommand*\rubystretchprop[3]{%
\rubystretchpropend 538   \edef\pxrr@sprop@x{#1}%
                    539   \edef\pxrr@sprop@y{#2}%
                    540   \edef\pxrr@sprop@z{#3}%
                    541 }
                    542 \newcommand*\rubystretchprophead[2]{%
                    543   \edef\pxrr@sprop@hy{#1}%
                    544   \edef\pxrr@sprop@hz{#2}%
                    545 }
                    546 \newcommand*\rubystretchpropend[2]{%
                    547   \edef\pxrr@sprop@ex{#1}%
                    548   \edef\pxrr@sprop@ey{#2}%
                    549 }

\rubyuseextra 残念ながら今のところは使用不可。
                    550 \newcommand*\rubyuseextra[1]{%
                    551   \pxrr@cmta=#1\relax
                    552   \ifnum\pxrr@cmta=\z@
                    553     \chardef\pxrr@extra\pxrr@cmta
                    554   \else
                    555     \pxrr@err@inv@value{\the\pxrr@cmta}%
                    556   \fi
                    557 }

```


3.7 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。)

```
558 \let\pxrr@bintr@\@empty
```

```
559 \let\pxrr@aintr@\@empty
```

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

```
560 \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。

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

入力が「||」の場合は、「|-|」に置き換える。

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

```
563 \ifx\pxrr@tempa\pxrr@doublebar
```

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

```
565 \fi
```

各パラメタの値を全般設定のもので初期化する。

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

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

```
568 \let\pxrr@bintr@\pxrr@d@bintr
```

```
569 \let\pxrr@aintr@\pxrr@d@aintr
```

```
570 \let\pxrr@athead@\pxrr@d@athead
```

```
571 \let\pxrr@mode\pxrr@d@mode
```

```
572 \let\pxrr@side\pxrr@d@side
```

```
573 \let\pxrr@evensp\pxrr@d@evensp
```

```
574 \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。

```
575 \let\pxrr@bscomp\relax
```

```
576 \let\pxrr@ascomp\relax
```

```
577 \pxrr@bnobrfalse
```

```
578 \pxrr@anobrfalse
```

```
579 \pxrr@bfintrfalse
```

```
580 \pxrr@afintrfalse
```

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。

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

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

```
583 }
```

有限状態機械のループ。

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

```
585 \ifpxrrDebug
```

```

586 \typeout{\pxrr@po@FS/#1[@@nameuse{\pxrr@po@C@#1}]}%
587 \fi
588 \csname pxrr@po@PR@#1\endcsname
589 \expandafter\ifx\csname pxrr@po@C@#1\endcsname\relax
590 \let\pxrr@po@FS\relax
591 \else
592 \pxrr@letcs\pxrr@po@FS
593 {\pxrr@po@TR@\pxrr@po@FS @@nameuse{\pxrr@po@C@#1}}%
594 \fi
595 \ifpxrrDebug
596 \typeout{->\pxrr@po@FS}%
597 \fi
598 \pxrr@ifx{\pxrr@po@FS\relax}{%
599 \pxrr@fatal@unx@letter{#1}%
600 \pxrr@parse@option@exit
601 }{%
602 \pxrr@parse@option@loop
603 }%
604 }

```

後処理。

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

既定値設定（\rubysetup）である場合何もしない。

```
606 \ifpxrr@in@setup\else
```

両側ルビ命令の場合は、\pxrr@side の値を変更する。

```
607 \ifpxrr@truby
608 \chardef\pxrr@side\tw@
609 \fi

```

整合性検査を行う。

```
610 \pxrr@check@option

\pxrr@?@intr の値を設定する。
611 \@tempdima=\pxrr@ruby@zw\relax
612 \@tempdimb=\pxrr@or@zero\pxrr@bintr@\@tempdima
613 \edef\pxrr@bintr{\the\@tempdimb}%
614 \@tempdimb=\pxrr@or@zero\pxrr@aintr@\@tempdima
615 \edef\pxrr@aintr{\the\@tempdimb}%
616 \fi
617 }

```

\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。

```

618 \def\pxrr@or@zero#1{%
619 \ifx#1@empty \pxrr@zero
620 \else #1%
621 \fi
622 }

```

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。

```
623 \def\pxrr@po@C@{F}
624 \@namedef{pxrr@po@C@|}{V}
625 \@namedef{pxrr@po@C@:}{S}
626 \@namedef{pxrr@po@C@.}{S}
627 \@namedef{pxrr@po@C@*}{S}
628 \@namedef{pxrr@po@C@!}{S}
629 \@namedef{pxrr@po@C@<}{B}
630 \@namedef{pxrr@po@C@()}{B}
631 \@namedef{pxrr@po@C@>}{A}
632 \@namedef{pxrr@po@C@)}{A}
633 \@namedef{pxrr@po@C@-}{M}
634 \def\pxrr@po@C@c{M}
635 \def\pxrr@po@C@h{M}
636 \def\pxrr@po@C@H{M}
637 \def\pxrr@po@C@m{M}
638 \def\pxrr@po@C@g{M}
639 \def\pxrr@po@C@j{M}
640 \def\pxrr@po@C@P{M}
641 \def\pxrr@po@C@S{M}
642 \def\pxrr@po@C@e{M}
643 \def\pxrr@po@C@E{M}
644 \def\pxrr@po@C@f{M}
645 \def\pxrr@po@C@F{M}
```

機能プロセス。

```
646 \def\pxrr@po@PR@{%
647   \pxrr@parse@option@exit
648 }
649 \@namedef{pxrr@po@PR@|}{%
650   \csname pxrr@po@PRbar@\pxrr@po@FS\endcsname
651 }
652 \def\pxrr@po@PRbar@bi{%
653   \def\pxrr@bintr@{}\pxrr@bprottrue
654 }
655 \def\pxrr@po@PRbar@bb{%
656   \pxrr@bprotrfalse
657 }
658 \def\pxrr@po@PRbar@bs{%
659   \def\pxrr@aintr@{}\pxrr@aprotrtrue
660 }
661 \let\pxrr@po@PRbar@mi\pxrr@po@PRbar@bs
662 \let\pxrr@po@PRbar@as\pxrr@po@PRbar@bs
663 \let\pxrr@po@PRbar@ai\pxrr@po@PRbar@bs
664 \def\pxrr@po@PRbar@ab{%
665   \pxrr@aprotrfalse
666 }
667 \@namedef{pxrr@po@PR@:}{%
668   \csname pxrr@po@PRcolon@\pxrr@po@FS\endcsname
```

```

669 }
670 \def\pxrr@po@PRcolon@bi{%
671   \let\pxrr@bscomp=: \relax
672 }
673 \let\pxrr@po@PRcolon@bb\pxrr@po@PRcolon@bi
674 \let\pxrr@po@PRcolon@bs\pxrr@po@PRcolon@bi
675 \def\pxrr@po@PRcolon@mi{%
676   \let\pxrr@ascomp=: \relax
677 }
678 \let\pxrr@po@PRcolon@as\pxrr@po@PRcolon@mi
679 \@namedef{pxrr@po@PR@.}{%
680   \csname pxrr@po@PRdot@\pxrr@po@FS\endcsname
681 }
682 \def\pxrr@po@PRdot@bi{%
683   \let\pxrr@bscomp=. \relax
684 }
685 \let\pxrr@po@PRdot@bb\pxrr@po@PRdot@bi
686 \let\pxrr@po@PRdot@bs\pxrr@po@PRdot@bi
687 \def\pxrr@po@PRdot@mi{%
688   \let\pxrr@ascomp=. \relax
689 }
690 \let\pxrr@po@PRdot@as\pxrr@po@PRdot@mi
691 \@namedef{pxrr@po@PR@*}{%
692   \csname pxrr@po@PRstar@\pxrr@po@FS\endcsname
693 }
694 \def\pxrr@po@PRstar@bi{%
695   \pxrr@bnobrtrue
696 }
697 \let\pxrr@po@PRstar@bb\pxrr@po@PRstar@bi
698 \let\pxrr@po@PRstar@bs\pxrr@po@PRstar@bi
699 \def\pxrr@po@PRstar@mi{%
700   \pxrr@anobrtrue
701 }
702 \let\pxrr@po@PRstar@as\pxrr@po@PRstar@mi
703 \@namedef{pxrr@po@PR@!}{%
704   \csname pxrr@po@PRbang@\pxrr@po@FS\endcsname
705 }
706 \def\pxrr@po@PRbang@bi{%
707   \pxrr@bfintrtrue
708 }
709 \let\pxrr@po@PRbang@bb\pxrr@po@PRbang@bi
710 \let\pxrr@po@PRbang@bs\pxrr@po@PRbang@bi
711 \def\pxrr@po@PRbang@mi{%
712   \pxrr@afintrtrue
713 }
714 \let\pxrr@po@PRbang@as\pxrr@po@PRbang@mi
715 \@namedef{pxrr@po@PR@<}{%
716   \def\pxrr@bintr@{\pxrr@big@intr}\pxrr@bprottrue
717 }

```

```

718 \@namedef{pxrr@po@PR@{}}{%
719   \def\pxrr@bintr@{\pxrr@small@intr}\pxrr@bprottrue
720 }
721 \@namedef{pxrr@po@PR@>}{%
722   \def\pxrr@aintr@{\pxrr@big@intr}\pxrr@aprottrue
723 }
724 \@namedef{pxrr@po@PR@}{%
725   \def\pxrr@aintr@{\pxrr@small@intr}\pxrr@aprottrue
726 }
727 \def\pxrr@po@PR@c{%
728   \chardef\pxrr@athead\z@
729 }
730 \def\pxrr@po@PR@h{%
731   \chardef\pxrr@athead\@ne
732 }
733 \def\pxrr@po@PR@H{%
734   \chardef\pxrr@athead\tw@
735 }
736 \def\pxrr@po@PR@m{%
737   \let\pxrr@mode=m%
738 }
739 \def\pxrr@po@PR@g{%
740   \let\pxrr@mode=g%
741 }
742 \def\pxrr@po@PR@j{%
743   \let\pxrr@mode=j%
744 }
745 \def\pxrr@po@PR@P{%
746   \chardef\pxrr@side\z@
747 }
748 \def\pxrr@po@PR@S{%
749   \chardef\pxrr@side\@ne
750 }
751 \def\pxrr@po@PR@E{%
752   \chardef\pxrr@evensp\z@
753 }
754 \def\pxrr@po@PR@e{%
755   \chardef\pxrr@evensp\@ne
756 }
757 \def\pxrr@po@PR@F{%
758   \chardef\pxrr@fullsize\z@
759 }
760 \def\pxrr@po@PR@f{%
761   \chardef\pxrr@fullsize\@ne
762 }

  遷移表。
763 \def\pxrr@po@TR@bi@F{fi}
764 \def\pxrr@po@TR@bb@F{fi}

```

```

765 \def\pxrr@po@TR@bs@F{fi}
766 \def\pxrr@po@TR@mi@F{fi}
767 \def\pxrr@po@TR@as@F{fi}
768 \def\pxrr@po@TR@ai@F{fi}
769 \def\pxrr@po@TR@ab@F{fi}
770 \def\pxrr@po@TR@fi@F{fi}
771 \def\pxrr@po@TR@bi@V{bb}
772 \def\pxrr@po@TR@bb@V{bs}
773 \def\pxrr@po@TR@bs@V{ab}
774 \def\pxrr@po@TR@mi@V{ab}
775 \def\pxrr@po@TR@as@V{ab}
776 \def\pxrr@po@TR@ai@V{ab}
777 \def\pxrr@po@TR@ab@V{fi}
778 \def\pxrr@po@TR@bi@S{bs}
779 \def\pxrr@po@TR@bb@S{bs}
780 \def\pxrr@po@TR@bs@S{bs}
781 \def\pxrr@po@TR@mi@S{as}
782 \def\pxrr@po@TR@as@S{as}
783 \def\pxrr@po@TR@bi@B{bs}
784 \def\pxrr@po@TR@bi@M{mi}
785 \def\pxrr@po@TR@bb@M{mi}
786 \def\pxrr@po@TR@bs@M{mi}
787 \def\pxrr@po@TR@mi@M{mi}
788 \def\pxrr@po@TR@bi@A{fi}
789 \def\pxrr@po@TR@bb@A{fi}
790 \def\pxrr@po@TR@bs@A{fi}
791 \def\pxrr@po@TR@mi@A{fi}
792 \def\pxrr@po@TR@as@A{fi}
793 \def\pxrr@po@TR@ai@A{fi}

```

3.8 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。

```

794 \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。

```

795 \ifpxrr@bprotr\else
796 \ifpxrr@aprotr\else
797 \pxrr@fatal@bad@no@protr
798 \fi
799 \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。

```

800 \pxrr@oktrue
801 \ifx\pxrr@bintr@\@empty\else
802 \pxrr@okfalse
803 \fi

```

```

804 \ifx\pxrr@aintr@\@empty\else
805   \pxrr@okfalse
806 \fi
807 \ifpxrr@ghost\else
808   \pxrr@oktrue
809 \fi
810 \ifpxrr@ok\else
811   \pxrr@fatal@bad@intr
812 \fi

```

モノルビ (m) ・ 熟語ルビ (j) に関する検査。

```

813 \if g\pxrr@mode\else

```

欧文ルビでは不可なのでグループルビに変更する。

```

814   \ifpxrr@abody
815     \let\pxrr@mode=g\relax
816 \fi

```

両側ルビでは不可なのでグループルビに変更する。

```

817   \ifnum\pxrr@side=\tw@
818     \let\pxrr@mode=g\relax
819 \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。

```

820 \if g\pxrr@mode
821   \if g\pxrr@d@mode
822     \pxrr@warn@must@group
823   \fi
824 \fi
825 \fi

```

肩付き指定 (h) に関する検査。

```

826 \ifnum\pxrr@athead>\z@

```

横組みでは不可なので中付きに変更する。

```

827   \ifpxrr@in@tate\else
828     \pxrr@athead\z@
829 \fi

```

グループルビでは不可なので中付きに変更する。

```

830 \if g\pxrr@mode
831   \pxrr@athead\z@
832 \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。

```

833 \ifnum\pxrr@athead=\z@
834   \ifnum\pxrr@d@athead>\z@
835     \pxrr@warn@bad@athead
836   \fi
837 \fi
838 \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。

```
839 \ifpxrr@abody
840   \chardef\pxrr@evensp\z@
841 \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。)

```
842 \if g\pxrr@mode\else
843   \chardef\pxrr@evensp\@ne
844 \fi
845 }
```

3.9 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。

```
846 \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。p_TE_X 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。

`\pxrr@ruby@zw`

```
847 \let\pxrr@body@zw\pxrr@zeropt
848 \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。

```
849 \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)。

```
850 \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。

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

`\pxrr@iiskip` 和文間空白および和欧文間空白の量。

`\pxrr@iaiskip` 852 `\let\pxrr@iiskip\pxrr@zeropt`

```
853 \let\pxrr@iaiskip\pxrr@zeropt
```

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

```
854 \def\pxrr@assign@fsize{%
855   \@tempdima=\f@size\p@
856   \@tempdima\pxrr@size@ratio\@tempdima
857   \edef\pxrr@ruby@fsize{\the\@tempdima}%
858   \pxrr@get@zwidth\pxrr@body@zw
859   \begingroup
860     \pxrr@use@ruby@font
```



```

861 \pxrr@get@zwidth\pxrr@gtempa
862 \global\let\pxrr@gtempa\pxrr@gtempa
863 \endgroup
864 \let\pxrr@ruby@zw\pxrr@gtempa
865 \pxrr@get@iiskip\pxrr@iiskip
866 \pxrr@get@iaiskip\pxrr@iaiskip

\pxrr@htratio の値を設定する。
867 \ifpxrr@in@tate
868 \let\pxrr@htratio\pxrr@thtratio
869 \else
870 \let\pxrr@htratio\pxrr@yhtratio
871 \fi

\pxrr@ruby@raise の値を計算する。
872 \@tempdima\pxrr@body@zw\relax
873 \@tempdima\pxrr@htratio\@tempdima
874 \@tempdimb\pxrr@ruby@zw\relax
875 \advance\@tempdimb-\pxrr@htratio\@tempdimb
876 \advance\@tempdima\@tempdimb
877 \@tempdimb\pxrr@body@zw\relax
878 \advance\@tempdima\pxrr@inter@gap\@tempdimb
879 \edef\pxrr@ruby@raise{\the\@tempdima}%

\pxrr@ruby@lower の値を計算する。
880 \@tempdima\pxrr@body@zw\relax
881 \advance\@tempdima-\pxrr@htratio\@tempdima
882 \@tempdimb\pxrr@ruby@zw\relax
883 \@tempdimb\pxrr@htratio\@tempdimb
884 \advance\@tempdima\@tempdimb
885 \@tempdimb\pxrr@body@zw\relax
886 \advance\@tempdima\pxrr@inter@gap\@tempdimb
887 \edef\pxrr@ruby@lower{\the\@tempdima}%
888 }

\pxrr@use@ruby@font
889 \def\pxrr@use@ruby@font{%
890 \pxrr@without@macro@trace{%
891 \let\rubyfontsize\pxrr@ruby@fsize
892 \fontsize{\pxrr@ruby@fsize}{\z@}\selectfont
893 \pxrr@ruby@font
894 }%
895 }

```

3.10 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 896 \chardef\pxrr@locate@inner=1
\pxrr@locate@end 897 \chardef\pxrr@locate@head=0

```

898 \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。

899 \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。

900 \setbox#2\hbox{#5}\@tempdima\wd#2%

901 \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。

902 \pxrr@decompose{#5}%

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

904 }

ここから実行を開始することもある。

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

比率パラメタの設定。

906 \pxrr@save@listproc

907 \ifcase#1%

908 \pxrr@evenspace@param\pxrr@zero\pxrr@sprop@hy\pxrr@sprop@hz

909 \or

910 \pxrr@evenspace@param\pxrr@sprop@x\pxrr@sprop@y\pxrr@sprop@z

911 \or

912 \pxrr@evenspace@param\pxrr@sprop@ex\pxrr@sprop@ey\pxrr@zero

913 \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。

914 \pxrr@dima=\pxrr@cntr\p@

915 \advance\pxrr@dima-\p@

916 \pxrr@dima=\pxrr@sprop@y\pxrr@dima

917 \advance\pxrr@dima\pxrr@sprop@x\p@

918 \advance\pxrr@dima\pxrr@sprop@z\p@

919 \ifdim\pxrr@dima>\z@ \else

920 \ifnum#1>\z@

921 \let\pxrr@sprop@x@\@ne

```

922     \advance\pxrr@dima\p@
923     \fi
924     \ifnum#1<\tw@
925         \let\pxrr@sprop@z@\@ne
926         \advance\pxrr@dima\p@
927     \fi
928     \fi
929     \edef\pxrr@tempa{\strip@pt\pxrr@dima}%
930 \ifpxrrDebug
931 \typeout{\number\pxrr@sprop@x@:\number\pxrr@sprop@z@:\pxrr@tempa}%
932 \fi

```

\pxrr@pre/inter/post にグル 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。

```

933 \def\pxrr@pre##1{\pxrr@hfilx\pxrr@sprop@x@ ##1}%
934 \def\pxrr@inter##1{\pxrr@hfilx\pxrr@sprop@y@ ##1}%
935 \def\pxrr@post{\pxrr@hfilx\pxrr@sprop@z@}%
936 \def\pxrr@makebox@res{%
937     \setbox#2=\hb@xt@#4{#3\pxrr@res}%
938 }%
939 \pxrr@makebox@res

```

前後の空白の量を求める。

```

940 \pxrr@dima\wd#2%
941 \advance\pxrr@dima-\pxrr@natwd\relax
942 \pxrr@invscale\pxrr@dima\pxrr@tempa
943 \@tempdima\pxrr@sprop@x@\pxrr@dima
944 \edef\pxrr@bspace{\the\@tempdima}%
945 \@tempdima\pxrr@sprop@z@\pxrr@dima
946 \edef\pxrr@aspace{\the\@tempdima}%
947 \pxrr@restore@listproc
948 \ifpxrrDebug
949 \typeout{\pxrr@bspace:\pxrr@aspace}%
950 \fi
951 }
952 \def\pxrr@evenspace@param#1#2#3{%
953     \let\pxrr@sprop@x@#1%
954     \let\pxrr@sprop@y@#2%
955     \let\pxrr@sprop@z@#3%
956 }

```

\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。

```

957 \def\pxrr@adjust@margin{%
958     \pxrr@save@listproc
959     \@tempdima\pxrr@body@zw\relax
960     \@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。

```

961 \@tempswafalse
962 \def\pxrr@pre##1{\pxrr@hfilx\pxrr@sprop@x@ ##1}%
963 \def\pxrr@inter##1{\pxrr@hfilx\pxrr@sprop@y@ ##1}%
964 \def\pxrr@post{\pxrr@hfilx\pxrr@sprop@z@}%
965 \ifnum\pxrr@cntr>\@ne
966   \ifdim\pxrr@bspace>\@tempdima
967     \edef\pxrr@bspace{\the\@tempdima}%
968     \def\pxrr@pre##1{\hskip\pxrr@bspace\relax ##1}%
969     \@tempswatrue
970   \fi
971   \ifdim\pxrr@aspace>\@tempdima
972     \edef\pxrr@aspace{\the\@tempdima}%
973     \def\pxrr@post{\hskip\pxrr@aspace\relax}%
974     \@tempswatrue
975   \fi
976 \fi

```

必要に応じて再調整を行う。

```

977 \if@tempswa
978   \pxrr@makebox@res
979 \fi
980 \pxrr@restore@listproc
981 \ifpxrr@Debug
982 \typeout{\pxrr@bspace:\pxrr@aspace}%
983 \fi
984 }

```

\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。

```

985 \def\pxrr@save@listproc{%
986   \let\pxrr@pre@save\pxrr@pre
987   \let\pxrr@inter@save\pxrr@inter
988   \let\pxrr@post@save\pxrr@post
989 }

```

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

```

990 \def\pxrr@restore@listproc{%
991   \let\pxrr@pre\pxrr@pre@save
992   \let\pxrr@inter\pxrr@inter@save
993   \let\pxrr@post\pxrr@post@save
994 }

```

3.11 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。

```

995 \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。

```

996 \def\pxrr@transform@kana#1{%
997   \let\pxrr@trans@res\@empty
998   \def\pxrr@transform@kana@end\pxrr@end{%
999     \let#1\pxrr@trans@res
1000   }%
1001   \expandafter\pxrr@transform@kana@loop@a#1\pxrr@end
1002 }
1003 \def\pxrr@transform@kana@loop@a{%
1004   \futurelet\pxrr@tempa\pxrr@transform@kana@loop@b
1005 }
1006 \def\pxrr@transform@kana@loop@b{%
1007   \ifx\pxrr@tempa\pxrr@end
1008     \let\pxrr@tempb\pxrr@transform@kana@end
1009   \else\ifx\pxrr@tempa\bgroup
1010     \let\pxrr@tempb\pxrr@transform@kana@loop@c
1011   \else\ifx\pxrr@tempa\@sptoken
1012     \let\pxrr@tempb\pxrr@transform@kana@loop@d
1013   \else
1014     \let\pxrr@tempb\pxrr@transform@kana@loop@e
1015   \fi\fi\fi
1016   \pxrr@tempb
1017 }
1018 \def\pxrr@transform@kana@loop@c#1{%
1019   \pxrr@appto\pxrr@trans@res{{#1}}%
1020   \pxrr@transform@kana@loop@a
1021 }
1022 \expandafter\def\expandafter\pxrr@transform@kana@loop@d\space{%
1023   \pxrr@appto\pxrr@trans@res{ }%
1024   \pxrr@transform@kana@loop@a
1025 }
1026 \def\pxrr@transform@kana@loop@e#1{%
1027   \expandafter\pxrr@transform@kana@loop@f\string#1\pxrr@nil#1%
1028 }
1029 \def\pxrr@transform@kana@loop@f#1#2\pxrr@nil#3{%
1030   \@tempswafalse
1031   \ifnum'#1>\@cclv
1032     \begingroup\expandafter\expandafter\expandafter\endgroup
1033     \expandafter\ifx\csname pxrr@nonsmall/#3\endcsname\relax\else
1034       \@tempswatrue
1035     \fi
1036   \fi
1037   \if@tempswa
1038     \edef\pxrr@tempa{%
1039       \noexpand\pxrr@appto\noexpand\pxrr@trans@res
1040       {\csname pxrr@nonsmall/#3\endcsname}%
1041     }%

```

```

1042 \pxrr@tempa
1043 \else
1044 \pxrr@appto\pxrr@trans@res{#3}%
1045 \fi
1046 \pxrr@transform@kana@loop@a
1047 }
1048 \def\pxrr@assign@nonsmall#1/#2\pxrr@nil{%
1049 \pxrr@get@jchar@token\pxrr@tempa{\pxrr@jc{#1}}%
1050 \pxrr@get@jchar@token\pxrr@tempb{\pxrr@jc{#2}}%
1051 \expandafter\edef\csname pxrr@nonsmall/\pxrr@tempa\endcsname
1052 {\pxrr@tempb}%
1053 }
1054 \@tfor\pxrr@tempc:=%
1055 {2421:3041/2422:3042}-{2423:3043/2424:3044}%
1056 {2425:3045/2426:3046}-{2427:3047/2428:3048}%
1057 {2429:3049/242A:304A}-{2443:3063/2444:3064}%
1058 {2463:3083/2464:3084}-{2465:3085/2466:3086}%
1059 {2467:3087/2468:3088}-{246E:308E/246F:308F}%
1060 {2521:30A1/2522:30A2}-{2523:30A3/2524:30A4}%
1061 {2525:30A5/2526:30A6}-{2527:30A7/2528:30A8}%
1062 {2529:30A9/252A:30AA}-{2543:30C3/2544:30C4}%
1063 {2563:30E3/2564:30E4}-{2565:30E5/2566:30E6}%
1064 {2567:30E7/2568:30E8}-{256E:30EE/256F:30EF}%
1065 \do{%
1066 \expandafter\pxrr@assign@nonsmall\pxrr@tempc\pxrr@nil
1067 }

```

3.12 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。

```
1068 \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。

```
1069 \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は、差はないものとみなす（演算誤差対策）

```
1070 \def\pxrr@epsilon{0.01pt}
```

\pxrr@compose@block \pxrr@compose@block{〈パターン〉}{r 〈親文字ブロック〉}{〈ルビ文字ブロック〉}：1つのブロックの組版処理。〈パターン〉は \pxrr@evenspace と同じ意味。突出があるかを \ifpxrr@protr に返し、前と後の突出の量をそれぞれ \pxrr@bspace と \pxrr@aspace に返す。

```
1071 \def\pxrr@compose@block{%
```

本体の前に加工処理を介入させる。 \pxrr@compose@block@do に本体マクロを \let する。

```
1072 \let\pxrr@compose@block@do\pxrr@compose@oneside@block@do
```

```

1073 \pxrr@compose@block@pre
1074 }

```

こちらが本体。

```

1075 \def\pxrr@compose@oneside@block@do#1#2#3{%
1076 \setbox\pxrr@boxa\hbox{#2}%
1077 \setbox\pxrr@boxr\hbox{%
1078 \pxrr@use@ruby@font
1079 #3%
1080 }%
1081 \@tempdima\wd\pxrr@boxr
1082 \advance\@tempdima-\wd\pxrr@boxa
1083 \ifdim\pxrr@epsilon<\@tempdima

```

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

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

```

1084 \pxrr@protrtrue
1085 \let\pxrr@locate@temp#1%
1086 \ifnum\pxrr@athead>\@ne
1087 \ifnum\pxrr@locate@temp=\pxrr@locate@inner
1088 \let\pxrr@locate@temp\pxrr@locate@head
1089 \fi
1090 \fi
1091 \pxrr@decompose{#2}%
1092 \edef\pxrr@natwd{\the\wd\pxrr@boxa}%
1093 \pxrr@evenspace@int\pxrr@locate@temp\pxrr@boxa\relax
1094 {\wd\pxrr@boxr}%
1095 \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。

```

1096 \pxrr@protrfalse
1097 \ifnum\pxrr@athead=\z@
1098 \pxrr@decompose{#3}%
1099 \edef\pxrr@natwd{\the\wd\pxrr@boxr}%
1100 \pxrr@evenspace@int{#1}\pxrr@boxr
1101 \pxrr@use@ruby@font{\wd\pxrr@boxa}%
1102 \pxrr@adjust@margin
1103 \fi
1104 \let\pxrr@bspace\pxrr@zeropt
1105 \let\pxrr@aspace\pxrr@zeropt
1106 \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）。

```

1107 \pxrr@protrfalse
1108 \let\pxrr@bspace\pxrr@zeropt

```

```

1109 \let\pxrr@aspace\pxrr@zeropt
1110 \fi\fi

```

実際に組版を行う。

```

1111 \setbox\z@\hbox{%
1112   \ifnum\pxrr@side=\z@
1113     \raise\pxrr@ruby@raise\box\pxrr@boxr
1114   \else
1115     \lower\pxrr@ruby@lower\box\pxrr@boxr
1116   \fi
1117 }%
1118 \ht\z@\z@ \dp\z@\z@
1119 \@tempdima\wd\z@
1120 \setbox\pxrr@boxr\hbox{%
1121   \box\z@
1122   \kern-\@tempdima
1123   \box\pxrr@boxa
1124 }%

```

\ifpxrr@any@protr を設定する。

```

1125 \ifpxrr@protr
1126   \pxrr@any@protrtrue
1127 \fi
1128 }

```

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

```

1129 \def\pxrr@compose@twoside@block{%
1130   \let\pxrr@compose@block@do\pxrr@compose@twoside@block@do
1131   \pxrr@compose@block@pre
1132 }
1133 \def\pxrr@compose@twoside@block@do#1#2#3#4{%
1134   \setbox\pxrr@boxa\hbox{#2}%
1135   \setbox\pxrr@boxr\hbox{%
1136     \pxrr@use@ruby@font
1137     #3%
1138   }%
1139   \setbox\pxrr@boxb\hbox{%
1140     \pxrr@use@ruby@font
1141     #4%
1142   }%

```

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。

```

1143 \@tempdima\wd\pxrr@boxa
1144 \ifdim\@tempdima<\wd\pxrr@boxr
1145   \@tempdima\wd\pxrr@boxr
1146 \fi
1147 \ifdim\@tempdima<\wd\pxrr@boxb
1148   \@tempdima\wd\pxrr@boxb
1149 \fi
1150 \edef\pxrr@maxwd{\the\@tempdima}%

```



```

1151 \advance\@tempdima-\pxrr@epsilon\relax
1152 \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。

```

1153 \ifdim\pxrr@maxwdx>\wd\pxrr@boxr
1154 \pxrr@decompose{#3}%
1155 \edef\pxrr@natwd{\the\wd\pxrr@boxr}%
1156 \pxrr@evenspace@int{#1}\pxrr@boxr
1157 \pxrr@use@ruby@font{\pxrr@maxwd}%
1158 \pxrr@adjust@margin
1159 \fi
1160 \ifdim\pxrr@maxwdx>\wd\pxrr@boxb
1161 \pxrr@decompose{#4}%
1162 \edef\pxrr@natwd{\the\wd\pxrr@boxb}%
1163 \pxrr@evenspace@int{#1}\pxrr@boxb
1164 \pxrr@use@ruby@font{\pxrr@maxwd}%
1165 \pxrr@adjust@margin
1166 \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。)

```

1167 \ifdim\pxrr@maxwdx>\wd\pxrr@boxa
1168 \pxrr@decompose{#2}%
1169 \edef\pxrr@natwd{\the\wd\pxrr@boxa}%
1170 \pxrr@evenspace@int{#1}\pxrr@boxa\relax{\pxrr@maxwd}%
1171 \else
1172 \let\pxrr@bspace\pxrr@zeropt
1173 \let\pxrr@aspace\pxrr@zeropt
1174 \fi

```

実際に組版を行う。

```

1175 \setbox\z@\hbox{%
1176 \@tempdima\wd\pxrr@boxr
1177 \raise\pxrr@ruby@raise\box\pxrr@boxr
1178 \kern-\@tempdima
1179 \lower\pxrr@ruby@lower\box\pxrr@boxb
1180 }%
1181 \ht\z@\z@ \dp\z@\z@
1182 \@tempdima\wd\z@
1183 \setbox\pxrr@boxr\hbox{%
1184 \box\z@
1185 \kern-\@tempdima
1186 \box\pxrr@boxa
1187 }%
1188 }

```

\pxrr@compose@block@pre 親文字列の加工を行う。

```

1189 \def\pxrr@compose@block@pre{%
  f 指定時は小書き仮名の変換を施す。

```

```

1190 \pxrr@ifnum{\pxrr@fullsize>\z@}{%
1191   \pxrr@compose@block@pre@a
1192 }{%
1193   \pxrr@compose@block@pre@c
1194 }%
1195 }
1196 \def\pxrr@compose@block@pre@a#1#2#3{%
1197   \def\pxrr@compose@block@tempa{#3}%
1198   \pxrr@transform@kana\pxrr@compose@block@tempa
1199   \expandafter\pxrr@compose@block@pre@b
1200   \expandafter{\pxrr@compose@block@tempa}{#1}{#2}%
1201 }
1202 \def\pxrr@compose@block@pre@b#1#2#3{%
1203   \pxrr@compose@block@pre@c{#2}{#3}{#1}%
1204 }
1205 \def\pxrr@compose@block@pre@c{%
1206   \pxrr@ifnum{\pxrr@evensp=\z@}{%
1207     \pxrr@compose@block@pre@d
1208   }{%
1209     \pxrr@compose@block@do
1210   }%
1211 }
1212 \def\pxrr@compose@block@pre@d#1#2{%
1213   \pxrr@compose@block@do{#1}{#2}}%
1214 }

```

3.13 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 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。

```

1215 \def\pxrr@add@protect#1{%
1216   \expandafter\pxrr@add@protect@a
1217   \csname\expandafter\@gobble\string#1\space\endcsname#1%
1218 }
1219 \def\pxrr@add@protect@a#1#2{%
1220   \let#1=#2%
1221   \def#2{\pxrr@check@protect\protect#1}%
1222 }
1223 \def\pxrr@check@protect{%

```

```

1224 \ifx\protect\@typeset@protect
1225 \expandafter\@gobble
1226 \fi
1227 }

```

3.14 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。

```

1228 \let\pxrr@body@input\@empty

```

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

```

1229 \def\pxrr@prepare@fallback#1{%
1230 \pxrr@fatal@errorfalse
1231 \def\pxrr@body@input{#1}%
1232 }

```

`\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。

```

1233 \def\pxrr@fallback{%
1234 \pxrr@body@input
1235 }

```

`\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。

```

1236 \def\pxrr@if@alive{%
1237 \ifpxrr@fatal@error \expandafter\@gobble
1238 \else \expandafter\@firstofone
1239 \fi
1240 }

```

3.15 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。

```

1241 \def\pxrr@end@kinsoku{0}

```

`\pxrr@ruby@scan` 片側ルビ用の先読み処理。

```

1242 \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。

```

1243 \def\pxrr@tempc{%
1244 \edef\pxrr@end@kinsoku{\the\pxrr@cntr}%
1245 \pxrr@do@proc{#1}{#2}%

```

```

1246 }%
1247 \pxrr@check@kinsoku\pxrr@tempc
1248 }

```

\pxrr@truby@scan 両側ルビ用の先読み処理。

```

1249 \def\pxrr@truby@scan#1#2#3{%
1250   \def\pxrr@tempc{%
1251     \edef\pxrr@end@kinsoku{\the\pxrr@cntr}%
1252     \pxrr@do@proc{#1}{#2}{#3}%
1253   }%
1254   \pxrr@check@kinsoku\pxrr@tempc
1255 }

```

\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。

```

1256 \def\pxrr@check@kinsoku#1{%
1257   \let\pxrr@tempb#1%
1258   \futurelet\pxrr@tempa\pxrr@check@kinsoku@a
1259 }
1260 \def\pxrr@check@kinsoku@a{%
1261   \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。

```

1262 \ifpxrr@abody\else
1263   \ifnum\pxrr@cntr=\@ne
1264     \pxrr@cntr\tw@
1265   \fi
1266 \fi
1267 \ifcase\pxrr@cntr
1268   \pxrr@cntr\z@
1269   \expandafter\pxrr@tempb
1270 \or
1271   \pxrr@cntr\@MM
1272   \expandafter\pxrr@tempb
1273 \else
1274   \expandafter\pxrr@check@kinsoku@b
1275 \fi
1276 }

```

\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。

```

1277 \def\pxrr@check@kinsoku@b#1{%
1278   \pxrr@check@kinsoku@c#1#1%
1279 }

```

```

1280 \def\pxrr@check@kinsoku@c#1{%
1281   \pxrr@cntr\prebreakpenalty'#1\relax
1282   \pxrr@tempb
1283 }

```

\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。

```

1284 \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。)

```

1285   \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。

```

1286     \ifcat\noexpand##1\relax
1287       \pxrr@cntr\z@
1288     \else\ifcat\noexpand##1\noexpand~%
1289       \pxrr@cntr\z@
1290     \else\ifcat\noexpand##1A%
1291       \pxrr@cntr\@ne
1292     \else\ifcat\noexpand##10%
1293       \pxrr@cntr\@ne
1294     \else

```

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

```

1295       \pxrr@cntr\z@
1296       \expandafter\pxrr@check@char@a\meaning##1#2\pxrr@nil
1297       \fi\fi\fi\fi
1298     }%
1299   \def\pxrr@check@char@a##1#2##2\pxrr@nil{%
1300     \ifcat @##10%
1301       \pxrr@cntr\tw@
1302     \fi
1303   }%
1304 }

```

規定の引数を用意して「定義マクロ」を呼ぶ。

```

1305 \expandafter\pxrr@tempa\string\kanji\pxrr@nil

```

3.16 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。)

```

1306 \let\pxrr@auto@penalty\z@

```

\pxrr@auto@icspace 文字間の空き。寸法値マクロ。

```

1307 \let\pxrr@auto@icspace\pxrr@zeropt

```

\pxrr@intr@amount 進入の幅。寸法値マクロ。

```
1308 \let\pxrr@intr@amount\pxrr@zeropt
```

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

```
1309 \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。

```
1310 \ifpxrr@bnoobr
```

```
1311 \let\pxrr@auto@penalty\@MM
```

```
1312 \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。

```
1313 \else
```

```
1314 \let\pxrr@auto@penalty\z@
```

```
1315 \if : \pxrr@bscomp
```

```
1316 \let\pxrr@auto@icspace\pxrr@iaiskip
```

```
1317 \else\if . \pxrr@bscomp
```

```
1318 \let\pxrr@auto@icspace\pxrr@zeropt
```

```
1319 \else
```

```
1320 \let\pxrr@auto@icspace\pxrr@iiskip
```

```
1321 \fi\fi
```

```
1322 \fi
```

```
1323 }
```

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

```
1324 \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。

```
1325 \if : \pxrr@bscomp\else
```

```
1326 \pxrr@bnoobrtrue
```

```
1327 \fi
```

```
1328 \ifpxrr@bnoobr
```

```
1329 \let\pxrr@auto@penalty\@MM
```

```
1330 \let\pxrr@auto@icspace\pxrr@zeropt
```

```
1331 \else
```

この分岐は和欧文間空白挿入指定 (:) に限る。

```
1332 \let\pxrr@auto@penalty\z@
```

```
1333 \let\pxrr@auto@icspace\pxrr@iaiskip
```

```
1334 \fi
```

```
1335 }
```

3.16.1 前側進入処理

\pxrr@intrude@head 前側の進入処理。

```
1336 \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。)

```
1337 \ifpxrr@ghost\else
```

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

```
1338 \let\pxrr@intr@amount\pxrr@bpace
1339 \ifdim\pxrr@bintr<\pxrr@intr@amount\relax
1340 \let\pxrr@intr@amount\pxrr@bintr
1341 \fi
```

\pxrr@auto@* の設定法は和文ルビと欧文ルビで処理が異なる。

```
1342 \ifpxrr@abody
1343 \pxrr@intrude@setauto@a
1344 \else
1345 \pxrr@intrude@setauto@j
1346 \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)。

```
1347 \ifpxrr@par@head
1348 \ifpxrr@bfintr
1349 \hskip-\pxrr@intr@amount\relax
1350 \fi
```

段落冒頭でない場合、ペナルティ、字間空きのグルー、進入用のグルーを順番に入れる。

```
1351 \else
1352 \penalty\pxrr@auto@penalty\relax
1353 \hskip-\pxrr@intr@amount\relax
1354 \hskip\pxrr@auto@icspace\relax
1355 \fi
1356 \fi
1357 }
```

3.16.2 後側進入処理

\pxrr@intrude@end 末尾での進入処理。

```
1358 \def\pxrr@intrude@end{%
1359 \ifpxrr@ghost\else
```

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

```
1360 \let\pxrr@intr@amount\pxrr@aspace
1361 \ifdim\pxrr@aintr<\pxrr@intr@amount\relax
1362 \let\pxrr@intr@amount\pxrr@aintr
1363 \fi
```

\pxrr@auto@* の設定法は和文ルビと欧文ルビで処理が異なる。

```
1364 \ifpxrr@abody
1365 \pxrr@intrude@setauto@a
1366 \else
1367 \pxrr@intrude@setauto@j
1368 \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。

```

1369 \ifnum\pxrr@auto@penalty<\@MM
1370 \let\pxrr@auto@penalty\pxrr@end@kinsoku
1371 \fi
1372 \penalty\pxrr@auto@penalty\relax
1373 \hskip-\pxrr@intr@amount\relax
1374 \hskip\pxrr@auto@icspace\relax

```

本物の前禁則ペナルティ（負かも知れない）はここに加算される。ここで行分割してはいけないのでペナルティ値を 20000 にする。

```

1375 \penalty\@MM
1376 \fi
1377 }

```

3.17 メインです

3.17.1 エントリーポイント

`\ruby` と文ルビの公開命令。`\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。

```

1378 \AtBeginDocument{%
1379 \providecommand*\ruby{\jruby}%
1380 }
1381 \newcommand*\jruby{%
1382 \pxrr@jprologue
1383 \pxrr@trubyfalse
1384 \pxrr@ruby
1385 }

```

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

```

1386 \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。

```

1387 \newcommand*\aruby{%
1388 \pxrr@aprologue
1389 \pxrr@trubyfalse
1390 \pxrr@ruby
1391 }
1392 \pxrr@add@protect\aruby

```

`\truby` 和文両側ルビの公開命令。

```

1393 \newcommand*\truby{%
1394 \pxrr@jprologue
1395 \pxrr@trubytrue
1396 \pxrr@ruby
1397 }
1398 \pxrr@add@protect\truby

```

`\atruby` 欧文両側ルビの公開命令。

```

1399 \newcommand*\atruby{%

```



```

1400 \pxrr@aprologue
1401 \pxrr@trubytrue
1402 \pxrr@ruby
1403 }
1404 \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。
1405 \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 1406 \let\pxrr@option\@empty
1407 \let\pxrr@exoption\@empty

\pxrr@do@proc \pxrr@ruby の処理中に使われる。
\pxrr@do@scan 1408 \let\pxrr@do@proc\@empty
1409 \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。
1410 \def\pxrr@ruby{%
1411   \@testopt\pxrr@ruby@a{}%
1412 }
1413 \def\pxrr@ruby@a[#1]{%
1414   \def\pxrr@option{#1}%
1415   \@testopt\pxrr@ruby@b{}%
1416 }
1417 \def\pxrr@ruby@b[#1]{%
1418   \def\pxrr@exoption{#1}%
1419   \ifpxrr@truby
1420     \let\pxrr@do@proc\pxrr@truby@proc
1421     \let\pxrr@do@scan\pxrr@truby@scan
1422   \else
1423     \let\pxrr@do@proc\pxrr@ruby@proc
1424     \let\pxrr@do@scan\pxrr@ruby@scan
1425   \fi
1426   \pxrr@ruby@c
1427 }
1428 \def\pxrr@ruby@c{%
1429   \ifpxrr@ghost
1430     \expandafter\pxrr@do@proc
1431   \else
1432     \expandafter\pxrr@do@scan
1433   \fi
1434 }

\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。
1435 \def\pxrr@ruby@proc#1#2{%
1436   \pxrr@prepare@fallback{#1}%

```

フォントサイズの変数を設定して、

```
1437 \pxrr@assign@fsize
```

オプションを解析する。

```
1438 \pxrr@parse@option\pxrr@option
```

ルビ文字入力をグループ列に分解する。

```
1439 \pxrr@decompbar{#2}%
```

```
1440 \let\pxrr@ruby@list\pxrr@res
```

```
1441 \edef\pxrr@ruby@count{\the\pxrr@cntr}%
```

親文字入力をグループ列に分解する。

```
1442 \pxrr@decompbar{#1}%
```

```
1443 \let\pxrr@body@list\pxrr@res
```

```
1444 \edef\pxrr@body@count{\the\pxrr@cntr}%
```

```
1445 \ifpxrrDebug
```

```
1446 \pxrr@debug@show@input
```

```
1447 \fi
```

入力検査を行い、パスした場合は組版処理に進む。

```
1448 \pxrr@if@alive{%
```

```
1449 \if g\pxrr@mode
```

```
1450 \pxrr@ruby@check@g
```

```
1451 \pxrr@if@alive{%
```

```
1452 \ifnum\pxrr@body@count>\@ne
```

```
1453 \pxrr@ruby@main@mg
```

```
1454 \else
```

```
1455 \pxrr@ruby@main@g
```

```
1456 \fi
```

```
1457 }%
```

```
1458 \else
```

```
1459 \pxrr@ruby@check@m
```

```
1460 \pxrr@if@alive{\pxrr@ruby@main@m}%
```

```
1461 \fi
```

```
1462 }%
```

後処理を行う。

```
1463 \pxrr@ruby@exit
```

```
1464 }
```

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

```
1465 \def\pxrr@truby@proc#1#2#3{%
```

```
1466 \pxrr@prepare@fallback{#1}%
```

フォントサイズの変数を設定して、

```
1467 \pxrr@assign@fsize
```

オプションを解析する。

```
1468 \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。

```
1469 \def\pxrr@all@input{#{1}-{#2}-{#3}}%
1470 \ifpxrr@Debug
1471 \pxrr@debug@show@input
1472 \fi
```

入力検査を行い、パスした場合は組版処理に進む。

```
1473 \pxrr@if@alive{%
1474 \pxrr@ruby@check@tg
1475 \pxrr@if@alive{\pxrr@ruby@main@tg}%
1476 }%
```

後処理を行う。

```
1477 \pxrr@ruby@exit
1478 }
```

3.17.2 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。

```
1479 \def\pxrr@ruby@check@g{%
1480 \ifnum\pxrr@body@count=\pxrr@ruby@count\relax
1481 \ifnum\pxrr@body@count=\@ne\else
1482 \ifpxrr@abody
1483 \pxrr@fatal@bad@movable
1484 \else\ifnum\pxrr@extra=\z@
1485 \pxrr@fatal@na@movable
1486 \fi\fi
1487 \fi
1488 \else
1489 \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
1490 \fi
1491 }
```

`\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。

```
1492 \def\pxrr@ruby@check@m{%
1493 \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。

```
1494 \let\pxrr@pre\pxrr@decompose
1495 \let\pxrr@post\relax
1496 \pxrr@body@list
1497 \let\pxrr@body@list\pxrr@res
1498 \edef\pxrr@body@count{\the\pxrr@cntr}%
```

```

1499     \ifnum\pxrr@body@count=\pxrr@ruby@count\relax\else
1500         \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
1501     \fi
1502 \else
1503     \pxrr@fatal@bad@mono
1504 \fi
1505 }

```

`\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。)

```

1506 \def\pxrr@ruby@check@tg{%
1507 }

```

3.17.3 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。

```

1508 \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。

```

1509 \def\pxrr@check@par@head{%
1510     \ifvmode
1511         \pxrr@par@headtrue
1512     \else
1513         \pxrr@par@headfalse
1514     \fi
1515 }

```

`\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。

```

1516 \def\pxrr@if@last#1#2#3{%
1517     \ifx#3\pxrr@post #1%
1518     \else #2%
1519     \fi
1520     #3%
1521 }

```

`\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。

```

1522 \def\pxrr@inter@mono{%
1523     \hskip\pxrr@iiskip\relax
1524 }

```

`\pxrr@takeout@any@protr` `\ifpxrr@any@protr` の値をグループの外に出す。

```

1525 \def\pxrr@takeout@any@protr{%
1526     \ifpxrr@any@protr
1527         \aftergroup\pxrr@any@protrtrue
1528     \fi

```

1529 }

\pxrr@ruby@main@m モノルビ。

```
1530 \def\pxrr@ruby@main@m{%
1531   \pxrr@zip@list\pxrr@body@list\pxrr@ruby@list
1532   \let\pxrr@whole@list\pxrr@res
1533   \pxrr@check@par@head
1534   \pxrr@any@protrfalse
1535   \ifpxrr@Debug
1536   \pxrr@debug@show@recomp
1537   \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。

```
1538   \let\pxrr@locate@head@\pxrr@locate@inner
1539   \let\pxrr@locate@end@\pxrr@locate@inner
1540   \let\pxrr@locate@sing@\pxrr@locate@inner
1541   \ifpxrr@aprotr\else
1542     \let\pxrr@locate@end@\pxrr@locate@end
1543     \let\pxrr@locate@sing@\pxrr@locate@end
1544   \fi
1545   \ifpxrr@bprotr\else
1546     \let\pxrr@locate@head@\pxrr@locate@head
1547     \let\pxrr@locate@sing@\pxrr@locate@head
1548   \fi
1549   \def\pxrr@pre##1##2{%
1550     \pxrr@if@last{%
```

単独ブロックの場合。

```
1551     \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
1552     \pxrr@intrude@head
1553     \unhbox\pxrr@boxr
1554     \pxrr@intrude@end
1555     \pxrr@takeout@any@protr
1556   }{%
```

先頭ブロックの場合。

```
1557     \pxrr@compose@block\pxrr@locate@head@{##1}{##2}%
1558     \pxrr@intrude@head
1559     \unhbox\pxrr@boxr
1560   }%
1561 }%
1562 \def\pxrr@inter##1##2{%
1563   \pxrr@if@last{%
```

末尾ブロックの場合。

```
1564     \pxrr@compose@block\pxrr@locate@end@{##1}{##2}%
1565     \pxrr@inter@mono
1566     \unhbox\pxrr@boxr
1567     \pxrr@intrude@end
```

```

1568     \pxrr@takeout@any@protr
1569 }{%

```

中間ブロックの場合。

```

1570     \pxrr@compose@block\pxrr@locate@inner{##1}{##2}%
1571     \pxrr@inter@mono
1572     \unhbox\pxrr@boxr
1573 }%
1574 }%
1575 \let\pxrr@post\@empty
1576 \setbox\pxrr@boxr\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。

```

1577 \if j\pxrr@mode
1578     \ifpxrr@any@protr
1579         \pxrr@ruby@redo@j
1580     \fi
1581 \fi
1582 \unhbox\pxrr@boxr
1583 }

```

\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、単純にグルーブルビの組み方にする。

```

1584 \def\pxrr@ruby@redo@j{%
1585     \pxrr@concat@list\pxrr@body@list
1586     \let\pxrr@body@list\pxrr@res
1587     \pxrr@concat@list\pxrr@ruby@list
1588     \let\pxrr@ruby@list\pxrr@res
1589     \pxrr@zip@single\pxrr@body@list\pxrr@ruby@list
1590     \let\pxrr@whole@list\pxrr@res
1591 \ifpxrr@Debug
1592 \pxrr@debug@show@concat
1593 \fi
1594     \let\pxrr@locate@sing@\pxrr@locate@inner
1595     \ifpxrr@aprotr\else
1596         \let\pxrr@locate@sing@\pxrr@locate@end
1597     \fi
1598     \ifpxrr@bprotr\else
1599         \let\pxrr@locate@sing@\pxrr@locate@head
1600     \fi
1601     \def\pxrr@pre##1##2{%
1602         \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
1603         \pxrr@intrude@head
1604         \unhbox\pxrr@boxr
1605         \pxrr@intrude@end
1606     }%
1607     \let\pxrr@inter\@undefined
1608     \let\pxrr@post\@empty
1609     \setbox\pxrr@boxr\hbox{\pxrr@whole@list}%

```

1610 }

\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。

```
1611 \def\pxrr@ruby@main@g{%
1612   \pxrr@zip@list\pxrr@body@list\pxrr@ruby@list
1613   \let\pxrr@whole@list\pxrr@res
1614   \pxrr@check@par@head
1615   \ifpxrr@Debug
1616     \pxrr@debug@show@recomp
1617   \fi
1618   \let\pxrr@locate@sing@\pxrr@locate@inner
1619   \ifpxrr@aprotr\else
1620     \let\pxrr@locate@sing@\pxrr@locate@end
1621   \fi
1622   \ifpxrr@bprotr\else
1623     \let\pxrr@locate@sing@\pxrr@locate@head
1624   \fi
1625   \def\pxrr@pre##1##2{%
1626     \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
1627     \pxrr@intrude@head
1628     \unhbox\pxrr@boxr
1629     \pxrr@intrude@end
1630   }%
1631   \let\pxrr@inter\@undefined
1632   \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。

```
1633   \pxrr@whole@list
1634 }
```

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

```
1635 \def\pxrr@ruby@main@tg{%
1636   \pxrr@check@par@head
1637   \let\pxrr@locate@sing@\pxrr@locate@inner
1638   \ifpxrr@aprotr\else
1639     \let\pxrr@locate@sing@\pxrr@locate@end
1640   \fi
1641   \ifpxrr@bprotr\else
1642     \let\pxrr@locate@sing@\pxrr@locate@head
1643   \fi
1644   \expandafter\pxrr@compose@twoside@block\expandafter\pxrr@locate@sing@
1645   \pxrr@all@input
1646   \pxrr@intrude@head
1647   \unhbox\pxrr@boxr
1648   \pxrr@intrude@end
1649 }
```

```

1650 \def\pxrr@debug@show@input{%
1651   \typeout{----\pxrr@pkgname\space input:^^J%
1652     ifpxrr@abody = \meaning\ifpxrr@abody^^J%
1653     ifpxrr@truby = \meaning\ifpxrr@truby^^J%
1654     pxrr@ruby@fsize = \pxrr@ruby@fsize^^J%
1655     pxrr@body@zw = \pxrr@body@zw^^J%
1656     pxrr@ruby@zw = \pxrr@ruby@zw^^J%
1657     pxrr@iiskip = \pxrr@iiskip^^J%
1658     pxrr@iaiskip = \pxrr@iaiskip^^J%
1659     pxrr@htratio = \pxrr@htratio^^J%
1660     pxrr@ruby@raise = \pxrr@ruby@raise^^J%
1661     pxrr@ruby@lower = \pxrr@ruby@lower^^J%
1662     ifpxrr@bprotr = \meaning\ifpxrr@bprotr^^J%
1663     ifpxrr@aprotr = \meaning\ifpxrr@aprotr^^J%
1664     pxrr@side = \the\pxrr@side^^J%
1665     pxrr@evensp = \the\pxrr@evensp^^J%
1666     pxrr@fullsize = \the\pxrr@fullsize^^J%
1667     pxrr@bscomp = \meaning\pxrr@bscomp^^J%
1668     pxrr@ascomp = \meaning\pxrr@ascomp^^J%
1669     ifpxrr@bnobr = \meaning\ifpxrr@bnobr^^J%
1670     ifpxrr@anobr = \meaning\ifpxrr@anobr^^J%
1671     ifpxrr@bfintr = \meaning\ifpxrr@bfintr^^J%
1672     ifpxrr@afintr = \meaning\ifpxrr@afintr^^J%
1673     pxrr@bintr = \pxrr@bintr^^J%
1674     pxrr@aintr = \pxrr@aintr^^J%
1675     pxrr@athead = \the\pxrr@athead^^J%
1676     pxrr@mode = \meaning\pxrr@mode^^J%
1677     pxrr@body@list = \meaning\pxrr@body@list^^J%
1678     pxrr@body@count = \@nameuse{pxrr@body@count}^^J%
1679     pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
1680     pxrr@ruby@count = \@nameuse{pxrr@ruby@count}^^J%
1681     pxrr@end@kinsoku = \pxrr@end@kinsoku^^J%
1682     ----
1683   }%
1684 }
1685 \def\pxrr@debug@show@recomp{%
1686   \typeout{----\pxrr@pkgname\space recomp:^^J%
1687     pxrr@body@list = \meaning\pxrr@body@list^^J%
1688     pxrr@body@count = \pxrr@body@count^^J%
1689     pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
1690     pxrr@ruby@count = \pxrr@ruby@count^^J%
1691     pxrr@res = \meaning\pxrr@res^^J%
1692     ----
1693   }%
1694 }
1695 \def\pxrr@debug@show@concat{%
1696   \typeout{----\pxrr@pkgname\space concat:^^J%
1697     pxrr@body@list = \meaning\pxrr@body@list^^J%
1698     pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%

```



```

1699     pxrr@whole@list = \meaning\pxrr@whole@list^^J%
1700     ----
1701   }%
1702 }

```

3.17.4 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。

```
1703 \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のだから `chardef` にする。

```
1704 \pxrr@jchardef\pxrr@zspace=\pxrr@jc{2121:3000}
```

`\pxrr@jprologue` 和文ルビ用の開始処理。

```
1705 \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。

```

1706   \ifpxrr@jghost
1707     \pxrr@zspace
1708   \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。

```

1709   \begingroup
1710     \pxrr@abodyfalse
1711     \pxrr@csletcs\ifpxrr@ghost\ifpxrr@jghost}%

```

出力した全角空白の幅だけ戻しておく。

```

1712     \ifpxrr@jghost
1713       \setbox\pxrr@boxa\hbox{\pxrr@zspace}%
1714       \kern-\wd\pxrr@boxa
1715     \fi
1716 }

```

`\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。）

```

1717 \ifpxrr@aghost
1718   \IfFileExists{t1lmr.fd}{%
1719     \begingroup
1720       \fontsize{2.5}{0}\usefont{T1}{lmr}{m}{n}
1721     \endgroup
1722     \pxrr@letcs\pxrr@aghostfont{T1/lmr/m/n/2.5}%
1723     \chardef\pxrr@aghostchar=23 % compwordmark

```

```

1724 \def\pxrr@aghost{{\pxrr@aghostfont\pxrr@aghostchar}}%
1725 \xspcode\pxrr@aghostchar=3 %
1726 }{%else
1727 \oxrr@warn{Ghost embedding for \string\aruby\space
1728 is disabled,\MessageBreak
1729 since package lmodern is missing}%
1730 \pxrr@aghostfalse
1731 \let\pxrr@aghosttrue\relax
1732 }%
1733 \fi

```

\pxrr@aprologue 欧文ルビ用の開始処理。

```

1734 \def\pxrr@aprologue{%
1735 \ifpxrr@aghost
1736 \pxrr@aghost
1737 \fi
1738 \begingroup
1739 \pxrr@abodytrue
1740 \pxrr@csletcs{\ifpxrr@ghost}{\ifpxrr@aghost}%
1741 }

```

3.17.5 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。

```

1742 \def\pxrr@ruby@exit{%
1743 \ifpxrr@fatal@error
1744 \pxrr@fallback
1745 \fi
1746 \ifpxrr@abody
1747 \expandafter\pxrr@aepilogue
1748 \else
1749 \expandafter\pxrr@jepilogue
1750 \fi
1751 }

```

\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。

```

1752 \def\pxrr@jepilogue{%
1753 \ifpxrr@jghost
1754 \setbox\pxrr@boxa\hbox{\pxrr@zspace}%
1755 \kern-\wd\pxrr@boxa
1756 \fi

```

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

```

1757 \endgroup
1758 \ifpxrr@jghost
1759 \pxrr@zspace

```

```
1760 \fi
1761 }
```

\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。

```
1762 \def\pxrr@aepilogue{%
1763 \endgroup
1764 \ifpxrr@aghost
1765 \pxrr@aghost
1766 \fi
1767 }
```