

pxrubrica パッケージ

八登 崇之 (Takayuki YATO; aka “ZR”)

v0.1 [2011/07/23]

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。
ここで、〈オプション〉は以下の形式をもつ。
`〈前進入設定〉〈前空き設定〉〈モード〉〈前空き設定〉〈後設定〉`

〈モード〉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 `\rubysetup` 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

〈前空き設定〉は以下の値の何れか。

`:` 和欧文間空白

〈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。

〈後空き設定〉は以下の値の何れか。

`:` 和欧文間空白

〈後進入設定〉は以下の値。

`||` 後突出禁止 `)` 後進入大
`|` 後進入無し `>` 後進入小

- `\jruby` [〈オプション〉] {〈親文字〉} {〈ルビ文字〉}

`\ruby` 命令の別名。 `\ruby` という命令名は他のパッケージとの衝突の可能性が高いので、`LATEX` 文書の本文開始時 (`\begin{document}`) に未定義である場合にのみ定義される。これに対して `\jruby` は常に定義される。なお、`\ruby` 以外の命令 (`\jruby` を含む) が定義済であった (命令名の衝突) 場合にはエラーとなる。

- `\aruby` [〈オプション〉] {〈親文字〉} {〈ルビ文字〉}

欧文ルビの命令。すなわち、欧文文字列の上側 (横組) / 右側 (縦組) にルビを付す。〈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。

- $\text{pT}_{\text{E}}\text{X}$ 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 = `|cjPl|`]
- `\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值 (親文字全角単位) [既定 = 0.75]
- `\rubyintergap{<実数>}`
ルビと親文字の間の空き (親文字全角単位) [既定 = 0]
- `\rubyusejghost` / `\rubynousejghost`
和文ゴースト処理を行う / 行わない。[既定 = 行わない]

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 ($\text{pT}_{\text{E}}\text{X}$ 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、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ での実現は非常に難しい。これを $\varepsilon\text{-pT}_{\mathrm{E}}\mathrm{X}$ 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{pxrubrica}
3 \def\pxrr@error{%
4   \PackageError\pxrr@pkgname
5 }
6 \def\pxrr@warn{%
7   \PackageWarning\pxrr@pkgname
8 }
```

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

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

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

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

`\pxrr@eh@fatal` 致命的エラーのヘルプ。

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

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

```
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@bad@athead` 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@movable` 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@movable{%
58   \pxrr@fatal@error{%
59     Movable group ruby is not allowed here%
60   }\pxrr@eh@fatal
61 }

```

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

```

62 \def\pxrr@fatal@na@movable{%
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@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 )
72 \def\pxrr@big@intr{1}
73 \def\pxrr@small@intr{0.5}

\pxrr@size@ratio ルビ文字サイズ( \rubysizeratio )、実数値マクロ。
74 \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 75 \def\pxrr@sprop@x{1}
\pxrr@sprop@z 76 \def\pxrr@sprop@y{2}
77 \def\pxrr@sprop@z{1}

\pxrr@sprop@hy 伸縮配置比率( \rubystretchprophead )、実数値マクロ。
\pxrr@sprop@hz 78 \def\pxrr@sprop@hy{1}
79 \def\pxrr@sprop@hz{1}

\pxrr@sprop@ex 伸縮配置比率( \rubystretchpropend )、実数値マクロ。
\pxrr@sprop@ey 80 \def\pxrr@sprop@ex{1}
81 \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。
82 \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。
83 \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。
84 \def\pxrr@thtratio{0.5}

\pxrr@extra Extra 実現方法( \rubyuseextra )、整数定数。
85 \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。
86 `\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。
87 `\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。
88 `\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 (`\[no]rubyadjustatlineedge`) スイッチ。
89 `\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 (`\[no]rubyadjustatlineedge`) スイッチ。
90 `\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` 91 `\newif\ifpxrr@d@bprotr \pxrr@d@bprotrtrue`
92 `\newif\ifpxrr@d@aprotr \pxrr@d@aprotrtrue`

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

`\ifpxrr@d@athead` 肩付きルビであるか否か (偽なら中付きルビ)。`\rubysetup` の `h/c` の設定。スイッチ。
95 `\newif\ifpxrr@d@athead`

`\pxrr@d@mode` モノルビ (`m`)・グルーブルビ (`g`)・熟語ルビ (`j`) のいずれか。`\rubysetup` の設定値。オプション文字への暗黙の (`\let` された) 文字トークン。
96 `\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。
97 `\chardef\pxrr@d@side=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` 98 `\newif\ifpxrr@bprotr \pxrr@bprotrfalse`
99 `\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` 100 `\def\pxrr@bintr{}`
101 `\def\pxrr@aintr{}`

`\pxrr@bscomp` 空き補正量。`\ruby` の : 指定に由来する。暗黙の文字トークン (未定義値は `\relax`)。
`\pxrr@ascomp` 既定値設定 (`\rubysetup`) でこれに対応するものはない。
102 `\let\pxrr@bscomp\relax`
103 `\let\pxrr@ascomp\relax`

`\ifpxrr@athead` 肩付きルビであるか否か (偽なら中付きルビ)。 `\ruby` の `h/c` の設定。スイッチ。

```
104 \newif\ifpxrr@athead
```

`\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。

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

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

```
106 \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。

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

3.4 補助手続

3.4.1 雑多な定義

`\ifpxrr@ok` 汎用スイッチ。

```
108 \newif\ifpxrr@ok
```

`\pxrr@canta` 汎用の整数レジスタ。

```
109 \newcount\pxrr@canta
```

`\pxrr@cntr` 結果を格納する整数レジスタ。

```
110 \newcount\pxrr@cntr
```

`\pxrr@dima` 汎用の寸法レジスタ。

```
111 \newdimen\pxrr@dima
```

`\pxrr@boxa` 汎用のボックスレジスタ。

```
112 \newbox\pxrr@boxa
```

```
113 \newbox\pxrr@boxb
```

`\pxrr@boxr` 結果を格納するボックスレジスタ。

```
114 \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。

```
115 \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。

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

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

```
117 \def\pxrr@hfilx#1{%
118   \hskip\z@\@plus #1fil\relax
119 }
```

```

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

\pxrr@ifx  \pxrr@ifx{<引数>}{<真>}{<偽>}： \ifx<引数> を行うテスト。
121 \def\pxrr@ifx#1{%
122   \ifx#1\expandafter\@firstoftwo
123   \else\expandafter\@secondoftwo
124   \fi
125 }

\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。
126 \def\pxrr@cslet#1{%
127   \expandafter\let\csname#1\endcsname
128 }
129 \def\pxrr@letcs#1#2{%
130   \expandafter\let\expandafter#1\csname#2\endcsname
131 }
132 \def\pxrr@csletcs#1#2{%
133   \expandafter\let\csname#1\expandafter\endcsname
134   \csname#2\endcsname
135 }

\pxrr@setok  \pxrr@setok{<テスト>}： テストの結果を \ifpxrr@ok に返す。
136 \def\pxrr@setok#1{%
137   #1{\pxrr@oktrue}{\pxrr@okfalse}%
138 }

\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。
139 \def\pxrr@appto#1#2{%
140   \expandafter\def\expandafter#1\expandafter{#1#2}%
141 }

\pxrr@nil   ユニークトークン。
\pxrr@end 142 \def\pxrr@nil{\noexpand\pxrr@nil}
143 \def\pxrr@end{\noexpand\pxrr@end}

\pxrr@use@fontsize  \pxrr@use@fontsize{<サイズ>}： フォントサイズを変更する。この命令の中ではマクロ
展開のトレースを無効にする。
144 \def\pxrr@use@fontsize#1{%
145   \chardef\pxrr@tracingmacros=\tracingmacros
146   \tracingmacros\z@
147   \fontsize{#1}{\z@}\selectfont
148   \tracingmacros\pxrr@tracingmacros
149 }

```

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、 $\langle \text{寸法レジスタ} \rangle = \langle \text{実数} \rangle \langle \text{寸法レジスタ} \rangle$ の逆の演算を行う。

```

150 \mathchardef\pxrr@invscale@ca=259
151 \def\pxrr@invscale#1#2{%
152   \begingroup
153     \@tempdima=#1\relax
154     \@tempdimb#2\p@\relax
155     \@tempcnta\@tempdima
156     \multiply\@tempcnta\@ccclvi
157     \divide\@tempcnta\@tempdimb
158     \multiply\@tempcnta\@ccclvi
159     \@tempcntb\p@
160     \divide\@tempcntb\@tempdimb
161     \advance\@tempcnta-\@tempcntb
162     \advance\@tempcnta-\tw@
163     \@tempdimb\@tempcnta\@ne
164     \advance\@tempcnta\@tempcntb
165     \advance\@tempcnta\@tempcntb
166     \advance\@tempcnta\pxrr@invscale@ca
167     \@tempdimc\@tempcnta\@ne
168     \@whiledim\@tempdimb<\@tempdimc\do{%
169       \@tempcntb\@tempdimb
170       \advance\@tempcntb\@tempdimc
171       \advance\@tempcntb\@ne
172       \divide\@tempcntb\tw@
173       \ifdim #2\@tempcntb>\@tempdima
174         \advance\@tempcntb\m@ne
175         \@tempdimc=\@tempcntb\@ne
176       \else
177         \@tempdimb=\@tempcntb\@ne
178       \fi}%
179     \xdef\pxrr@gtmpa{\the\@tempdimb}%
180   \endgroup
181   #1=\pxrr@gtmpa\relax
182 }
```

`\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。

```

183 \def\pxrr@interpolate#1#2#3#4#5{%
184   \edef\pxrr@tempa{#1}%
185   \edef\pxrr@tempb{#2}%
186   \def\pxrr@tempd{#3}%
187   \setlength{\@tempdima}{#4}%
188   \edef\pxrr@tempc{(0,0)#5(*,*)}%
189   \expandafter\pxrr@interpolate@a\pxrr@tempc\@nil
190 }
191 \def\pxrr@interpolate@a(#1,#2)(#3,#4)(#5,#6){%
192   \if*#5%
193     \def\pxrr@tempc{\pxrr@interpolate@b{#1}{#2}{#3}{#4}}%
194     \else\ifdim\@tempdima<#3\pxrr@tempa
195       \def\pxrr@tempc{\pxrr@interpolate@b{#1}{#2}{#3}{#4}}%
196     \else
197       \def\pxrr@tempc{\pxrr@interpolate@a(#3,#4)(#5,#6)}%
198     \fi\fi
199     \pxrr@tempc
200 }
201 \def\pxrr@interpolate@b#1#2#3#4#5\@nil{%
202   \@tempdimb=-#1\pxrr@tempa
203   \advance\@tempdima\@tempdimb
204   \advance\@tempdimb#3\pxrr@tempa
205   \edef\pxrr@tempc{\strip@pt\@tempdimb}%
206   \pxrr@invscale\@tempdima\pxrr@tempc
207   \edef\pxrr@tempc{\strip@pt\@tempdima}%
208   \@tempdima=#4\pxrr@tempb
209   \@tempdimb=#2\pxrr@tempb
210   \advance\@tempdima-\@tempdimb
211   \@tempdima=\pxrr@tempc\@tempdima
212   \advance\@tempdima\@tempdimb
213   \pxrr@tempd=\@tempdima
214 }

```

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)。

```

215 \def\pxrr@decompose#1{%
216   \let\pxrr@res\@empty
217   \pxrr@cntr=\z@
218   \pxrr@decompose@loopa#1\pxrr@end
219 }

```

```

220 \def\pxrr@decompose@loopa{%
221   \futurelet\pxrr@tempa\pxrr@decompose@loopb
222 }
223 \def\pxrr@decompose@loopb{%
224   \pxrr@ifx{\pxrr@tempa\pxrr@end}{%
225     \pxrr@appto\pxrr@res{\pxrr@post}%
226   }{%
227     \pxrr@setok{\pxrr@ifx{\pxrr@tempa\bgroup}}%
228     \pxrr@decompose@loopc
229   }%
230 }
231 \def\pxrr@decompose@loopc#1{%
232   \ifx\pxrr@res\@empty
233     \def\pxrr@res{\pxrr@pre}%
234   \else
235     \pxrr@appto\pxrr@res{\pxrr@inter}%
236   \fi
237   \ifpxrr@ok
238     \pxrr@appto\pxrr@res{{{#1}}}%
239   \else
240     \pxrr@appto\pxrr@res{#{#1}}%
241   \fi
242   \advance\pxrr@cntr\@ne
243   \pxrr@decompose@loopa
244 }

```

\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。

```

245 \def\pxrr@decompbar#1{%
246   \let\pxrr@res\@empty
247   \pxrr@cntr=\z@
248   \pxrr@decompbar@loopa\pxrr@nil#1\pxrr@end|%
249 }
250 \def\pxrr@decompbar@loopa#1|{%
251   \expandafter\pxrr@decompbar@loopb\expandafter{\@gobble#1}%
252 }
253 \def\pxrr@decompbar@loopb#1{%
254   \pxrr@decompbar@loopc#1\relax\pxrr@nil{#1}%
255 }
256 \def\pxrr@decompbar@loopc#1#2\pxrr@nil#3{%
257   \pxrr@ifx{#1\pxrr@end}{%
258     \pxrr@appto\pxrr@res{\pxrr@post}%
259   }{%
260     \ifx\pxrr@res\@empty
261       \def\pxrr@res{\pxrr@pre}%
262     \else
263       \pxrr@appto\pxrr@res{\pxrr@inter}%

```

```

264     \fi
265     \pxrr@appto\pxrr@res{#{3}}%
266     \advance\pxrr@cntr\@ne
267     \pxrr@decompbar@loopa\pxrr@nil
268 }%
269 }

```

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

270 \def\pxrr@zip@list#1#2{%
271   \let\pxrr@res\@empty
272   \let\pxrr@post\relax
273   \let\pxrr@tempa#1\pxrr@appto\pxrr@tempa{}}%
274   \let\pxrr@tempb#2\pxrr@appto\pxrr@tempb{}}%
275   \pxrr@zip@list@loopa
276 }
277 \def\pxrr@zip@list@loopa{%
278   \expandafter\pxrr@zip@list@loopb\pxrr@tempa\pxrr@end
279 }
280 \def\pxrr@zip@list@loopb#1#2#3\pxrr@end{%
281   \pxrr@ifx{#1\relax}{%
282     \pxrr@zip@list@exit
283   }{%
284     \pxrr@appto\pxrr@res{#{#2}}%
285     \def\pxrr@tempa{#{3}}%
286     \expandafter\pxrr@zip@list@loopc\pxrr@tempb\pxrr@end
287   }%
288 }
289 \def\pxrr@zip@list@loopc#1#2#3\pxrr@end{%
290   \pxrr@ifx{#1\relax}{%
291     \pxrr@interror{zip}%
292     \pxrr@appto\pxrr@res{}}%
293     \pxrr@zip@list@exit
294   }{%
295     \pxrr@appto\pxrr@res{#{#2}}%
296     \def\pxrr@tempb{#{3}}%
297     \pxrr@zip@list@loopa
298   }%
299 }
300 \def\pxrr@zip@list@exit{%
301   \pxrr@appto\pxrr@res{\pxrr@post}%
302 }

```

`\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。

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

```

303 \def\pxrr@concat@list#1{%
304   \let\pxrr@res\@empty
305   \def\pxrr@pre##1{%
306     \pxrr@appto\pxrr@res{##1}%
307   }%
308   \let\pxrr@inter\pxrr@pre
309   \let\pxrr@post\relax
310   #1%
311 }
```

`\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。

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

```

312 \def\pxrr@zip@single#1#2{%
313   \expandafter\pxrr@zip@single@a\expandafter#1#2\pxrr@end
314 }
315 \def\pxrr@zip@single@a#1{%
316   \expandafter\pxrr@zip@single@b#1\pxrr@end
317 }
318 \def\pxrr@zip@single@b#1\pxrr@end#2\pxrr@end{%
319   \def\pxrr@res{\pxrr@pre{#1}\{#2\}\pxrr@post}%
320 }
```

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

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

の時に、`\pxrr@res` を以下の内容に定義する。

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

```

321 \def\pxrr@tzip@single#1#2#3{%
322   \expandafter\pxrr@tzip@single@a\expandafter#1\expandafter#2#3\pxrr@end
323 }
324 \def\pxrr@tzip@single@a#1#2{%
325   \expandafter\pxrr@tzip@single@b\expandafter#1#2\pxrr@end
326 }
327 \def\pxrr@tzip@single@b#1{%
328   \expandafter\pxrr@tzip@single@c#1\pxrr@end
329 }
```



```

330 \def\pxrr@tzip@single@c#1\pxrr@end#2\pxrr@end#3\pxrr@end{%
331   \def\pxrr@res{\pxrr@pre{#1}{#2}{#3}\pxrr@post}%
332 }

```

3.5 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。

```

333 \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。

```

334 \newcommand*\rubyssetup[1]{%
335   \pxrr@in@setuptrue
336   \pxrr@fatal@errorfalse
337   \pxrr@parse@option{#1}%
338   \ifpxrr@fatal@error\else
339     \pxrr@csletcs{ifpxrr@d@bprotr}{ifpxrr@bprotr}%
340     \pxrr@csletcs{ifpxrr@d@aprotr}{ifpxrr@aprotr}%
341     \let\pxrr@d@bintr\pxrr@bintr@
342     \let\pxrr@d@aintr\pxrr@aintr@
343     \pxrr@csletcs{ifpxrr@d@athead}{ifpxrr@athead}%
344     \let\pxrr@d@mode\pxrr@mode
345     \let\pxrr@d@side\pxrr@side
346   \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。

```

347   \pxrr@in@setupfalse
348 }

```

`\rubybigintrusion` 対応するパラメタを設定する。

```

\rubysmallintrusion 349 \newcommand*\rubybigintrusion[1]{%
  \rubymaxmargin 350   \edef\pxrr@big@intr{#1}%
  351 }
  \rubyintergap 352 \newcommand*\rubysmallintrusion[1]{%
\rubysizeratio 353   \edef\pxrr@small@intr{#1}%
  354 }
  355 \newcommand*\rubymaxmargin[1]{%
  356   \edef\pxrr@maxmargin{#1}%
  357 }
  358 \newcommand*\rubyintergap[1]{%
  359   \edef\pxrr@inter@gap{#1}%
  360 }
  361 \newcommand*\rubysizeratio[1]{%
  362   \edef\pxrr@size@ratio{#1}%
  363 }

```

`\rubyusejghost` 対応するスイッチを設定する。

```

\rubynousejghost 364 \newcommand*\rubyusejghost{%

```

```

365 \pxrr@jghosttrue
366 }
367 \newcommand*\rubynousejghost{%
368 \pxrr@jghostfalse
369 }

```

\rubyuseaghost 対応するスイッチを設定する。

```

\rubynouseaghost 370 \newcommand*\rubyuseaghost{%
371 \pxrr@aghosttrue
372 }
373 \newcommand*\rubynouseaghost{%
374 \pxrr@aghostfalse
375 }

```

\rubyadjustatlineedge 対応するスイッチを設定する。

```

\rubynoadjustatlineedge 376 \newcommand*\rubyadjustatlineedge{%
377 \pxrr@edge@adjusttrue
378 }
379 \newcommand*\rubynoadjustatlineedge{%
380 \pxrr@edge@adjustfalse
381 }

```

\rubybreakjukugo 対応するスイッチを設定する。

```

\rubynobreakjukugo 382 \newcommand*\rubybreakjukugo{%
383 \pxrr@break@jukugotrue
384 }
385 \newcommand*\rubynobreakjukugo{%
386 \pxrr@break@jukugofalse
387 }

```

\rubystretchprop 対応するパラメタを設定する。

```

\rubystretchprophead 388 \newcommand*\rubystretchprop[3]{%
\rubystretchpropend 389 \edef\pxrr@sprop@x{#1}%
390 \edef\pxrr@sprop@y{#2}%
391 \edef\pxrr@sprop@z{#3}%
392 }
393 \newcommand*\rubystretchprophead[2]{%
394 \edef\pxrr@sprop@hy{#1}%
395 \edef\pxrr@sprop@hz{#2}%
396 }
397 \newcommand*\rubystretchpropend[2]{%
398 \edef\pxrr@sprop@ex{#1}%
399 \edef\pxrr@sprop@ey{#2}%
400 }

```

\rubyuseextra 残念ながら今のところは使用不可。

```

401 \newcommand*\rubyuseextra[1]{%
402 \pxrr@cmta=#1\relax
403 \ifnum\pxrr@cmta=\z@

```

```

404 \chardef\pxrr@extra\pxrr@cmta
405 \else
406 \pxrr@err@inv@value{\the\pxrr@cmta}%
407 \fi
408 }

```

3.6 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。)

```

409 \let\pxrr@bintr@\empty
410 \let\pxrr@aintr@\empty

```

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

```

411 \def\pxrr@doublebar{||}

```

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

```

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

```

入力が「||」の場合は、「|-|」に置き換える。

```

413 \edef\pxrr@tempa{#1}%
414 \ifx\pxrr@tempa\pxrr@doublebar
415 \def\pxrr@tempa{||}%
416 \fi

```

各パラメタの値を全般設定のもので初期化する。

```

417 \pxrr@csletcs{ifpxrr@bprotr}{ifpxrr@d@bprotr}%
418 \pxrr@csletcs{ifpxrr@aprotr}{ifpxrr@d@aprotr}%
419 \let\pxrr@bintr@\pxrr@d@bintr
420 \let\pxrr@aintr@\pxrr@d@aintr
421 \pxrr@csletcs{ifpxrr@thead}{ifpxrr@d@thead}%
422 \let\pxrr@mode\pxrr@d@mode
423 \let\pxrr@side\pxrr@d@side

```

次の2つの既定値は常に `\relax` (無効) である。

```

424 \let\pxrr@bscomp\relax
425 \let\pxrr@ascomp\relax

```

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。

```

426 \def\pxrr@po@FS{bi}%
427 \expandafter\pxrr@parse@option@loop\pxrr@tempa @\pxrr@end
428 }

```

有限状態機械のループ。

```

429 \def\pxrr@parse@option@loop#1{%
430 \ifpxrr@Debug
431 \typeout{\pxrr@po@FS/#1[\@nameuse{pxrr@po@C@#1}]}%

```

```

432 \fi
433 \csname pxrr@po@PR@#1\endcsname
434 \pxrr@letcs\pxrr@po@FS
435 {pxrr@po@TR@\pxrr@po@FS @\@nameuse{pxrr@po@C@#1}}%
436 \ifpxrrDebug
437 \typeout{->\pxrr@po@FS}%
438 \fi
439 \pxrr@ifx{\pxrr@po@FS\relax}{%
440 \pxrr@fatal@unx@letter{#1}%
441 \pxrr@parse@option@exit
442 }{%
443 \pxrr@parse@option@loop
444 }%
445 }

```

後処理。

```

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

```

両側ルビ命令の場合は、\pxrr@side の値を変更する。

```

447 \ifpxrr@truby
448 \chardef\pxrr@side\tw@
449 \fi

```

既定値設定 (\rubyssetup) でない場合は整合性検査を行う。

```

450 \ifpxrr@in@setup\else
451 \pxrr@check@option
452 \fi

```

\pxrr@?intr の値を設定する。

```

453 \@tempdima=\pxrr@ruby@zw\relax
454 \@tempdimb=\pxrr@or@zero\pxrr@bintr@\@tempdima
455 \edef\pxrr@bintr{\the\@tempdimb}%
456 \@tempdimb=\pxrr@or@zero\pxrr@aintr@\@tempdima
457 \edef\pxrr@aintr{\the\@tempdimb}%
458 }

```

\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。

```

459 \def\pxrr@or@zero#1{%
460 \ifx#1\@empty \pxrr@zero
461 \else #1%
462 \fi
463 }

```

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。

```

464 \def\pxrr@po@C@#{F}
465 \@namedef{pxrr@po@C@|}{V}
466 \@namedef{pxrr@po@C@:}{S}
467 \@namedef{pxrr@po@C@*}{S}
468 \@namedef{pxrr@po@C@<}{B}

```

```

469 \@namedef{pxrr@po@C@({}{B}
470 \@namedef{pxrr@po@C@>}{A}
471 \@namedef{pxrr@po@C@)}{A}
472 \@namedef{pxrr@po@C@-}{M}
473 \def\pxrr@po@C@h{M}
474 \def\pxrr@po@C@c{M}
475 \def\pxrr@po@C@m{M}
476 \def\pxrr@po@C@g{M}
477 \def\pxrr@po@C@j{M}
478 \def\pxrr@po@C@P{M}
479 \def\pxrr@po@C@S{M}

```

機能プロセス。

```

480 \def\pxrr@po@PR@{
481   \pxrr@parse@option@exit
482 }
483 \@namedef{pxrr@po@PR@|}{
484   \csname pxrr@po@PRbar@\pxrr@po@FS\endcsname
485 }
486 \def\pxrr@po@PRbar@bi{
487   \def\pxrr@bintr@{}\pxrr@bprottrue
488 }
489 \def\pxrr@po@PRbar@bb{
490   \pxrr@bprotrfalse
491 }
492 \def\pxrr@po@PRbar@mi{
493   \def\pxrr@aintr@{}\pxrr@aprotrtrue
494 }
495 \def\pxrr@po@PRbar@ab{
496   \pxrr@aprotrfalse
497 }
498 \@namedef{pxrr@po@PR@:}{
499   \csname pxrr@po@PRcolon@\pxrr@po@FS\endcsname
500 }
501 \def\pxrr@po@PRcolon@bi{
502   \let\pxrr@bscomp=: \relax
503 }
504 \let\pxrr@po@PRcolon@bb\pxrr@po@PRcolon@bi
505 \let\pxrr@po@PRcolon@bs\pxrr@po@PRcolon@bi
506 \def\pxrr@po@PRcolon@ai{
507   \let\pxrr@ascomp=: \relax
508 }
509 \@namedef{pxrr@po@PR@*}{
510   \csname pxrr@po@PRstar@\pxrr@po@FS\endcsname
511 }
512 \def\pxrr@po@PRstar@bi{
513   \let\pxrr@bscomp=* \relax
514 }
515 \let\pxrr@po@PRstar@bb\pxrr@po@PRstar@bi

```

```

516 \let\pxrr@po@PRstar@bs\pxrr@po@PRstar@bi
517 \def\pxrr@po@PRstar@ai{%
518   \let\pxrr@ascomp=*\relax
519 }
520 \@namedef{pxrr@po@PR@<}{%
521   \def\pxrr@bintr@{\pxrr@big@intr}\pxrr@bprottrue
522 }
523 \@namedef{pxrr@po@PR@<}{%
524   \def\pxrr@bintr@{\pxrr@small@intr}\pxrr@bprottrue
525 }
526 \@namedef{pxrr@po@PR@>}{%
527   \def\pxrr@aintr@{\pxrr@big@intr}\pxrr@aprottrue
528 }
529 \@namedef{pxrr@po@PR@}{%
530   \def\pxrr@aintr@{\pxrr@small@intr}\pxrr@aprottrue
531 }
532 \def\pxrr@po@PR@h{%
533   \pxrr@atheadtrue
534 }
535 \def\pxrr@po@PR@c{%
536   \pxrr@atheadfalse
537 }
538 \def\pxrr@po@PR@m{%
539   \let\pxrr@mode=m%
540 }
541 \def\pxrr@po@PR@g{%
542   \let\pxrr@mode=g%
543 }
544 \def\pxrr@po@PR@j{%
545   \let\pxrr@mode=j%
546 }
547 \def\pxrr@po@PR@P{%
548   \chardef\pxrr@side\z@
549 }
550 \def\pxrr@po@PR@S{%
551   \chardef\pxrr@side\@ne
552 }

```

遷移表。

```

553 \def\pxrr@po@TR@bi@F{fi}
554 \def\pxrr@po@TR@bb@F{fi}
555 \def\pxrr@po@TR@bs@F{fi}
556 \def\pxrr@po@TR@mi@F{fi}
557 \def\pxrr@po@TR@ai@F{fi}
558 \def\pxrr@po@TR@ab@F{fi}
559 \def\pxrr@po@TR@fi@F{fi}
560 \def\pxrr@po@TR@bi@V{bb}
561 \def\pxrr@po@TR@bb@V{bs}
562 \def\pxrr@po@TR@bs@V{ab}

```

```

563 \def\pxrr@po@TR@mi@V{ab}
564 \def\pxrr@po@TR@ai@V{ab}
565 \def\pxrr@po@TR@ab@V{fi}
566 \def\pxrr@po@TR@bi@S{mi}
567 \def\pxrr@po@TR@bb@S{mi}
568 \def\pxrr@po@TR@bs@S{mi}
569 \def\pxrr@po@TR@mi@S{ai}
570 \def\pxrr@po@TR@bi@B{bs}
571 \def\pxrr@po@TR@bi@M{mi}
572 \def\pxrr@po@TR@bb@M{mi}
573 \def\pxrr@po@TR@bs@M{mi}
574 \def\pxrr@po@TR@mi@M{mi}
575 \def\pxrr@po@TR@bi@A{fi}
576 \def\pxrr@po@TR@bb@A{fi}
577 \def\pxrr@po@TR@bs@A{fi}
578 \def\pxrr@po@TR@mi@A{fi}
579 \def\pxrr@po@TR@ai@A{fi}

```

3.7 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。

```
580 \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。

```

581 \ifpxrr@bprotr\else
582 \ifpxrr@aprotr\else
583 \pxrr@fatal@bad@no@protr
584 \fi
585 \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。

```

586 \pxrr@oktrue
587 \ifx\pxrr@bintr@\@empty\else
588 \pxrr@okfalse
589 \fi
590 \ifx\pxrr@aintr@\@empty\else
591 \pxrr@okfalse
592 \fi
593 \ifpxrr@ghost\else
594 \pxrr@oktrue
595 \fi
596 \ifpxrr@ok\else
597 \pxrr@fatal@bad@intr
598 \fi

```

モノルビ (m) ・ 熟語ルビ (j) に関する検査。

```
599 \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。

```
600 \ifpxrr@abody
601 \let\pxrr@mode=g\relax
602 \fi
```

両側ルビでは不可なのでグループルビに変更する。

```
603 \ifnum\pxrr@side=\tw@
604 \let\pxrr@mode=g\relax
605 \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。

```
606 \if g\pxrr@mode
607 \if g\pxrr@d@mode
608 \pxrr@warn@must@group
609 \fi
610 \fi
611 \fi
```

肩付き指定 (h) に関する検査。

```
612 \ifpxrr@athead
```

横組みでは不可なので中付きに変更する。

```
613 \ifydir
614 \pxrr@atheadfalse
615 \fi
```

グループルビでは不可なので中付きに変更する。

```
616 \if g\pxrr@mode
617 \pxrr@atheadfalse
618 \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。

```
619 \ifpxrr@athead\else
620 \ifpxrr@d@athead\else
621 \pxrr@warn@bad@athead
622 \fi
623 \fi
624 \fi
625 }
```

3.8 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。

```
626 \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。

```
627 \let\pxrr@body@zw\pxrr@zeropt
628 \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。

```
629 \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）。

```
630 \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。

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

`\pxrr@assign@fsize` 上記の変数（マクロ）を設定する。

```
632 \def\pxrr@assign@fsize{%
633   \@tempdima=\f@size\p@
634   \@tempdima\pxrr@size@ratio\@tempdima
635   \edef\pxrr@ruby@fsize{\the\@tempdima}%
636   \@tempdima=1zw\relax
637   \edef\pxrr@body@zw{\the\@tempdima}%
638   \begingroup
639     \pxrr@use@fontsize{\pxrr@ruby@fsize}%
640     \@tempdima=1zw\relax
641     \xdef\pxrr@gtempa{\the\@tempdima}%
642   \endgroup
643   \let\pxrr@ruby@zw\pxrr@gtempa
```

`\pxrr@htratio` の値を設定する。

```
644 \iftdir
645   \let\pxrr@htratio\pxrr@thtratio
646 \else
647   \let\pxrr@htratio\pxrr@yhtratio
648 \fi
```

`\pxrr@ruby@raise` の値を計算する。

```
649 \@tempdima\pxrr@body@zw\relax
650 \@tempdima\pxrr@htratio\@tempdima
651 \@tempdimb\pxrr@ruby@zw\relax
652 \advance\@tempdimb-\pxrr@htratio\@tempdimb
653 \advance\@tempdima\@tempdimb
654 \@tempdimb\pxrr@body@zw\relax
655 \advance\@tempdima\pxrr@inter@gap\@tempdimb
656 \edef\pxrr@ruby@raise{\the\@tempdima}%
```

`\pxrr@ruby@lower` の値を計算する。

```
657 \@tempdima\pxrr@body@zw\relax
658 \advance\@tempdima-\pxrr@htratio\@tempdima
659 \@tempdimb\pxrr@ruby@zw\relax
```

```

660 \@tempdimb\pxrr@htratio\@tempdimb
661 \advance\@tempdima\@tempdimb
662 \@tempdimb\pxrr@body@zw\relax
663 \advance\@tempdima\pxrr@inter@gap\@tempdimb
664 \edef\pxrr@ruby@lower{\the\@tempdima}%
665 }

```

3.9 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 666 \chardef\pxrr@locate@inner=1
\pxrr@locate@end 667 \chardef\pxrr@locate@head=0
668 \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。

```
669 \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。

```

670 \setbox#2\hbox{#5}\@tempdima\wd#2%
671 \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。

```

672 \pxrr@decompose{#5}%
673 \pxrr@evenspace@int{#1}{#2}{#3}{#4}%
674 }

```

ここから実行を開始することもある。

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

比率パラメタの設定。

```

676 \pxrr@save@listproc
677 \ifcase#1%
678 \pxrr@evenspace@param\pxrr@zero\pxrr@sprop@hy\pxrr@sprop@hz
679 \or
680 \pxrr@evenspace@param\pxrr@sprop@x\pxrr@sprop@y\pxrr@sprop@z
681 \or

```

```

682 \pxrr@evenspace@param\pxrr@sprop@ex\pxrr@sprop@ey\pxrr@zero
683 \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。

```

684 \pxrr@dima=\pxrr@cntr\p@
685 \advance\pxrr@dima-\p@
686 \pxrr@dima=\pxrr@sprop@y@\pxrr@dima
687 \advance\pxrr@dima\pxrr@sprop@x@\p@
688 \advance\pxrr@dima\pxrr@sprop@z@\p@
689 \ifdim\pxrr@dima>z@\else
690 \ifnum#1>z@
691 \let\pxrr@sprop@x@\@ne
692 \advance\pxrr@dima\p@
693 \fi
694 \ifnum#1<\tw@
695 \let\pxrr@sprop@z@\@ne
696 \advance\pxrr@dima\p@
697 \fi
698 \fi
699 \edef\pxrr@tempa{\strip@pt\pxrr@dima}%
700 \ifpxrrDebug
701 \typeout{\number\pxrr@sprop@x@:\number\pxrr@sprop@z@:\pxrr@tempa}%
702 \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。

```

703 \def\pxrr@pre##1{\pxrr@hfilx\pxrr@sprop@x@ ##1}%
704 \def\pxrr@inter##1{\pxrr@hfilx\pxrr@sprop@y@ ##1}%
705 \def\pxrr@post{\pxrr@hfilx\pxrr@sprop@z@}%
706 \def\pxrr@makebox@res{%
707 \setbox#2=\hb@xt@#4{#3\pxrr@res}%
708 }%
709 \pxrr@makebox@res

```

前後の空白の量を求める。

```

710 \pxrr@dima\wd#2%
711 \advance\pxrr@dima-\pxrr@natwd\relax
712 \pxrr@invscale\pxrr@dima\pxrr@tempa
713 \@tempdima\pxrr@sprop@x@\pxrr@dima
714 \edef\pxrr@bspace{\the\@tempdima}%
715 \@tempdima\pxrr@sprop@z@\pxrr@dima
716 \edef\pxrr@aspace{\the\@tempdima}%
717 \pxrr@restore@listproc
718 \ifpxrrDebug
719 \typeout{\pxrr@bspace:\pxrr@aspace}%
720 \fi
721 }

```

```

722 \def\pxrr@evenspace@param#1#2#3{%
723   \let\pxrr@sprop@x@#1%
724   \let\pxrr@sprop@y@#2%
725   \let\pxrr@sprop@z@#3%
726 }

```

\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。

```

727 \def\pxrr@adjust@margin{%
728   \pxrr@save@listproc
729   \@tempdima\pxrr@body@zw\relax
730   \@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。

```

731   \@tempswafalse
732   \def\pxrr@pre##1{\pxrr@hfilx\pxrr@sprop@x@ ##1}%
733   \def\pxrr@inter##1{\pxrr@hfilx\pxrr@sprop@y@ ##1}%
734   \def\pxrr@post{\pxrr@hfilx\pxrr@sprop@z@}%
735   \ifnum\pxrr@cntr>\@ne
736     \ifdim\pxrr@bspace>\@tempdima
737       \edef\pxrr@bspace{\the\@tempdima}%
738       \def\pxrr@pre##1{\hskip\pxrr@bspace\relax ##1}%
739       \@tempswatrue
740     \fi
741     \ifdim\pxrr@aspace>\@tempdima
742       \edef\pxrr@aspace{\the\@tempdima}%
743       \def\pxrr@post{\hskip\pxrr@aspace\relax}%
744       \@tempswatrue
745     \fi
746   \fi

```

必要に応じて再調整を行う。

```

747   \if@tempswa
748     \pxrr@makebox@res
749   \fi
750   \pxrr@restore@listproc
751   \ifpxrr@Debug
752     \typeout{\pxrr@bspace:\pxrr@aspace}%
753   \fi
754 }

```

\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。

```

755 \def\pxrr@save@listproc{%
756   \let\pxrr@pre@save\pxrr@pre
757   \let\pxrr@inter@save\pxrr@inter
758   \let\pxrr@post@save\pxrr@post

```

```
759 }
```

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

```
760 \def\pxrr@restore@listproc{%
761   \let\pxrr@pre\pxrr@pre@save
762   \let\pxrr@inter\pxrr@inter@save
763   \let\pxrr@post\pxrr@post@save
764 }
```

3.10 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。

```
765 \def\pxrr@add@protect#1{%
766   \expandafter\pxrr@add@protect@a
767   \csname\expandafter\@gobble\string#1\space\endcsname#1%
768 }
769 \def\pxrr@add@protect@a#1#2{%
770   \let#1=#2%
771   \def#2{\pxrr@check@protect\protect#1}%
772 }
773 \def\pxrr@check@protect{%
774   \ifx\protect\@typeset@protect
775     \expandafter\@gobble
776   \fi
777 }
```

3.11 ブロック毎の処理

`\ifpxrr@protr` ルビ文字列の突出があるか。スイッチ。

```
778 \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。

```
779 \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)。

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

`\pxrr@compose@block` `\pxrr@compose@block{〈パターン〉}{r 〈親文字ブロック〉}{l 〈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。

```

781 \def\pxrr@compose@block#1#2#3{%
782   \setbox\pxrr@boxa\hbox{#2}%
783   \setbox\pxrr@boxr\hbox{%
784     \pxrr@use@fontsize{\pxrr@ruby@fsize}%
785     #3%
786   }%
787   \@tempdima\wd\pxrr@boxr
788   \advance\@tempdima-\wd\pxrr@boxa
789   \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。

```

790   \pxrr@protrtrue
791   \pxrr@decompose{#2}%
792   \edef\pxrr@natwd{\the\wd\pxrr@boxa}%
793   \pxrr@evenspace@int{#1}\pxrr@boxa\relax{\wd\pxrr@boxr}%
794   \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。

```

795   \pxrr@protrfalse
796   \ifpxrr@athead\else
797     \pxrr@decompose{#3}%
798     \edef\pxrr@natwd{\the\wd\pxrr@boxr}%
799     \pxrr@evenspace@int{#1}\pxrr@boxr
800     {\pxrr@use@fontsize{\pxrr@ruby@fsize}}{\wd\pxrr@boxa}%
801     \pxrr@adjust@margin
802   \fi
803   \let\pxrr@bspace\pxrr@zeropt
804   \let\pxrr@aspace\pxrr@zeropt
805   \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）。

```

806   \pxrr@protrfalse
807   \let\pxrr@bspace\pxrr@zeropt
808   \let\pxrr@aspace\pxrr@zeropt
809   \fi\fi

```

実際に組版を行う。

```

810   \setbox\z@\hbox{%
811     \ifnum\pxrr@side=\z@
812       \raise\pxrr@ruby@raise\box\pxrr@boxr

```

```

813     \else
814         \lower\pxrr@ruby@lower\box\pxrr@boxr
815     \fi
816 }%
817 \ht\z@\z@ \dp\z@\z@
818 \@tempdima\wd\z@
819 \setbox\pxrr@boxr\hbox{%
820     \box\z@
821     \kern-\@tempdima
822     \box\pxrr@boxa
823 }%

\ifpxrr@any@protr を設定する。
824 \ifpxrr@protr
825     \pxrr@any@protrtrue
826 \fi
827 }

```

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

```

828 \def\pxrr@compose@twoside@block#1#2#3#4{%
829     \setbox\pxrr@boxa\hbox{#2}%
830     \setbox\pxrr@boxr\hbox{%
831         \pxrr@use@fontsize{\pxrr@ruby@fsize}%
832         #3%
833     }%
834     \setbox\pxrr@boxb\hbox{%
835         \pxrr@use@fontsize{\pxrr@ruby@fsize}%
836         #4%
837     }%

```

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。

```

838 \@tempdima\wd\pxrr@boxa
839 \ifdim\@tempdima<\wd\pxrr@boxr
840     \@tempdima\wd\pxrr@boxr
841 \fi
842 \ifdim\@tempdima<\wd\pxrr@boxb
843     \@tempdima\wd\pxrr@boxb
844 \fi
845 \edef\pxrr@maxwd{\the\@tempdima}%
846 \advance\@tempdima-\pxrr@epsilon\relax
847 \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。

```

848 \ifdim\pxrr@maxwdx>\wd\pxrr@boxa
849     \pxrr@decompose{#2}%
850     \edef\pxrr@natwd{\the\wd\pxrr@boxa}%
851     \pxrr@evenspace@int{#1}\pxrr@boxa\relax{\pxrr@maxwd}%
852 \fi
853 \ifdim\pxrr@maxwdx>\wd\pxrr@boxr
854     \pxrr@decompose{#3}%

```

```

855 \edef\pxrr@natwd{\the\wd\pxrr@boxr}%
856 \pxrr@evenspace@int{#1}\pxrr@boxr
857 {\pxrr@use@fontsize{\pxrr@ruby@fsize}}{\pxrr@maxwd}%
858 \pxrr@adjust@margin
859 \fi
860 \ifdim\pxrr@maxwdx>\wd\pxrr@boxb
861 \pxrr@decompose{#4}%
862 \edef\pxrr@natwd{\the\wd\pxrr@boxb}%
863 \pxrr@evenspace@int{#1}\pxrr@boxb
864 {\pxrr@use@fontsize{\pxrr@ruby@fsize}}{\pxrr@maxwd}%
865 \pxrr@adjust@margin
866 \fi

```

実際に組版を行う。

```

867 \setbox\z@\hbox{%
868 \@tempdima\wd\pxrr@boxr
869 \raise\pxrr@ruby@raise\box\pxrr@boxr
870 \kern-\@tempdima
871 \lower\pxrr@ruby@lower\box\pxrr@boxb
872 }%
873 \ht\z@\z@ \dp\z@\z@
874 \@tempdima\wd\z@
875 \setbox\pxrr@boxr\hbox{%
876 \box\z@
877 \kern-\@tempdima
878 \box\pxrr@boxa
879 }%
880 }

```

3.12 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。

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

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

```

882 \def\pxrr@prepare@fallback#1{%
883 \pxrr@fatal@errorfalse
884 \def\pxrr@body@input{#1}%
885 }

```

`\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。

```

886 \def\pxrr@fallback{%
887 \pxrr@body@input
888 }

```

`\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。


```

889 \def\pxrr@if@alive{%
890   \ifpxrr@fatal@error \expandafter\@gobble
891   \else \expandafter\@firstofone
892   \fi
893 }

```

3.13 メインです

3.13.1 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。

```

894 \AtBeginDocument{%
895   \providecommand*\{ruby}\{jruby}%
896 }
897 \newcommand*\{jruby}{%
898   \pxrr@jprologue
899   \pxrr@trubyfalse
900   \pxrr@ruby
901 }

```

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

```

902 \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。

```

903 \newcommand*\{aruby}{%
904   \pxrr@aprologue
905   \pxrr@trubyfalse
906   \pxrr@ruby
907 }
908 \pxrr@add@protect\aruby

```

`\truby` 和文両側ルビの公開命令。

```

909 \newcommand*\{truby}{%
910   \pxrr@jprologue
911   \pxrr@trubytrue
912   \pxrr@ruby
913 }
914 \pxrr@add@protect\truby

```

`\atruby` 欧文両側ルビの公開命令。

```

915 \newcommand*\{atruby}{%
916   \pxrr@aprologue
917   \pxrr@trubytrue
918   \pxrr@ruby
919 }
920 \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。
921 \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 922 \let\pxrr@option\@empty
923 \let\pxrr@exoption\@empty

\pxrr@do@proc \pxrr@ruby の処理中に使われる。
\pxrr@do@scan 924 \let\pxrr@do@proc\@empty
925 \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。
926 \def\pxrr@ruby{%
927   \@testopt\pxrr@ruby@a{}%
928 }
929 \def\pxrr@ruby@a[#1]{%
930   \def\pxrr@option{#1}%
931   \@testopt\pxrr@ruby@b{}%
932 }
933 \def\pxrr@ruby@b[#1]{%
934   \def\pxrr@exoption{#1}%
935   \ifpxrr@truby
936     \let\pxrr@do@proc\pxrr@truby@proc
937     \let\pxrr@do@scan\pxrr@truby@scan
938   \else
939     \let\pxrr@do@proc\pxrr@ruby@proc
940     \let\pxrr@do@scan\pxrr@ruby@scan
941   \fi
942   \pxrr@ruby@c
943 }
944 \def\pxrr@ruby@c{%
945   \ifpxrr@ghost
946     \expandafter\pxrr@do@proc
947   \else
948     \expandafter\pxrr@do@scan
949   \fi
950 }

\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。
951 \def\pxrr@ruby@proc#1#2{%
952   \pxrr@prepare@fallback{#1}%

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

              オプションを解析する。
954   \pxrr@parse@option\pxrr@option

```

ルビ文字入力をグループ列に分解する。

```
955 \pxrr@decompbar{#2}%
956 \let\pxrr@ruby@list\pxrr@res
957 \edef\pxrr@ruby@count{\the\pxrr@cntr}%
```

親文字入力をグループ列に分解する。

```
958 \pxrr@decompbar{#1}%
959 \let\pxrr@body@list\pxrr@res
960 \edef\pxrr@body@count{\the\pxrr@cntr}%
961 \ifpxrr@Debug
962 \pxrr@debug@show@input
963 \fi
```

入力検査を行い、パスした場合は組版処理に進む。

```
964 \pxrr@if@alive{%
965   \if g\pxrr@mode
966     \pxrr@ruby@check@g
967     \pxrr@if@alive{%
968       \ifnum\pxrr@body@count>\@ne
969         \pxrr@ruby@main@m
970       \else
971         \pxrr@ruby@main@g
972       \fi
973     }%
974   \else
975     \pxrr@ruby@check@m
976     \pxrr@if@alive{\pxrr@ruby@main@m}%
977   \fi
978 }%
```

後処理を行う。

```
979 \pxrr@ruby@exit
980 }
```

`\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。

```
981 \def\pxrr@truby@proc#1#2#3{%
982 \pxrr@prepare@fallback{#1}%
```

フォントサイズの変数を設定して、

```
983 \pxrr@assign@fsize
```

オプションを解析する。

```
984 \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。

```
985 \def\pxrr@all@input{#{1}{#2}{#3}}%
986 \ifpxrr@Debug
987 \pxrr@debug@show@input
988 \fi
```

入力検査を行い、パスした場合は組版処理に進む。

```
989 \pxrr@if@alive{%
990   \pxrr@ruby@check@tg
991   \pxrr@if@alive{\pxrr@ruby@main@tg}%
992 }%
```

後処理を行う。

```
993 \pxrr@ruby@exit
994 }
```

3.13.2 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、`\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べきである。

`\pxrr@ruby@scan` 片側ルビ用の先読み処理。

```
995 \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。
996   \def\pxrr@tempc{%
997     \edef\pxrr@end@kinsoku{\the\pxrr@cntr}%
998     \pxrr@do@proc{#1}{#2}%
999   }%
1000   \pxrr@check@kinsoku\pxrr@tempc
1001 }
```

`\pxrr@truby@scan` 両側ルビ用の先読み処理。

```
1002 \def\pxrr@truby@scan#1#2#3{%
1003   \def\pxrr@tempc{%
1004     \edef\pxrr@end@kinsoku{\the\pxrr@cntr}%
1005     \pxrr@do@proc{#1}{#2}{#3}%
1006   }%
1007   \pxrr@check@kinsoku\pxrr@tempc
1008 }
```

`\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 `\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。

```
1009 \def\pxrr@check@kinsoku#1{%
1010   \let\pxrr@tempb#1%
1011   \futurelet\pxrr@tempa\pxrr@check@kinsoku@a
1012 }
1013 \def\pxrr@check@kinsoku@a{%
1014   \pxrr@check@char\pxrr@tempa
1015   \if@tempswa
```

```

1016 \expandafter\pxrr@check@kinsoku@b
1017 \else
1018 \pxrr@cntr\z@
1019 \expandafter\pxrr@tempb
1020 \fi
1021 }

```

\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。

```

1022 \def\pxrr@check@kinsoku@b#1{%
1023 \pxrr@check@kinsoku@c#1#1%
1024 }
1025 \def\pxrr@check@kinsoku@c#1{%
1026 \pxrr@cntr\prebreakpenalty'#1\relax
1027 \pxrr@tempb
1028 }

```

\pxrr@check@char \pxrr@check@char\CS: トークン \CS が「通常文字」であるかを \if@tempswa に返す。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、少々特殊な処理をしている。まず \pxrr@check@char を定義するためのマクロを用意する。

```

1029 \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。）

```

1030 \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。

```

1031 \ifcat\noexpand##1\relax
1032 \@tempwafalse
1033 \else\ifcat\noexpand##1\noexpand~%
1034 \@tempwafalse
1035 \else\ifcat\noexpand##1A%
1036 \@tempwattrue
1037 \else\ifcat\noexpand##10%
1038 \@tempwattrue
1039 \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。）

```

1040 \@tempwafalse
1041 \expandafter\pxrr@check@char@a\meaning##1#2\pxrr@nil
1042 \fi\fi\fi\fi
1043 }%
1044 \def\pxrr@check@char@a##1#2##2\pxrr@nil{%
1045 \ifcat @##10%
1046 \@tempwattrue
1047 \fi
1048 }%

```

1049 }

規定の引数を用意して「定義マクロ」を呼ぶ。

1050 \expandafter\pxrr@tempa\string\kanji\pxrr@nil

3.13.3 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。

```
1051 \def\pxrr@ruby@check@g{%
1052   \ifnum\pxrr@body@count=\pxrr@ruby@count\relax
1053     \ifnum\pxrr@body@count=\@ne\else
1054       \ifpxrr@abody
1055         \pxrr@fatal@bad@movable
1056       \else\ifnum\pxrr@extra=\z@
1057         \pxrr@fatal@na@movable
1058       \fi\fi
1059     \fi
1060   \else
1061     \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
1062   \fi
1063 }
```

\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。

```
1064 \def\pxrr@ruby@check@m{%
1065   \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。

1066     \let\pxrr@pre\pxrr@decompose
1067     \let\pxrr@post\relax
1068     \pxrr@body@list
1069     \let\pxrr@body@list\pxrr@res
1070     \edef\pxrr@body@count{\the\pxrr@cuntr}%
1071     \ifnum\pxrr@body@count=\pxrr@ruby@count\relax\else
1072       \pxrr@fatal@bad@length\pxrr@body@count\pxrr@ruby@count
1073     \fi
1074   \else
1075     \pxrr@fatal@bad@mono
1076   \fi
1077 }
```

\pxrr@ruby@check@tg 両側ルビの場合、ここで検査する内容はない。（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。）

```
1078 \def\pxrr@ruby@check@tg{%
1079 }
```

3.13.4 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。

```
1080 \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。

```
1081 \def\pxrr@check@par@head{%
1082   \ifvmode
1083     \pxrr@par@headtrue
1084   \else
1085     \pxrr@par@headfalse
1086   \fi
1087 }
```

`\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。

```
1088 \def\pxrr@if@last#1#2#3{%
1089   \ifx#3\pxrr@post #1%
1090   \else #2%
1091   \fi
1092   #3%
1093 }
```

`\pxrr@inter@mono` モノルビのブロック間に挿入される空き。

```
1094 \def\pxrr@inter@mono{%
1095   \hskip\kanjiskip
1096 }
```

`\pxrr@intrude@head` 先頭での進入処理。

```
1097 \def\pxrr@intrude@head{%
```

ゴースト処理が行われている場合は、こちらの処理は行わない。(だから進入が扱えない。)

```
1098   \ifpxrr@ghost\else
```

段落冒頭では処理なし。

```
1099   \ifpxrr@par@head\else
```

前空き補正の処理。

```
1100     \if *\pxrr@bscomp
1101       \penalty\@M
1102     \else\if :\pxrr@bscomp
1103       \hskip\xkanjiskip
1104     \else
1105       \hskip\kanjiskip
1106     \fi\fi
```

実際の進入の量を求め、その量の負のグルを入れる。

```

1107      \pxrr@dima\pxrr@bspace\relax
1108      \ifdim\pxrr@bintr<\pxrr@dima
1109      \pxrr@dima\pxrr@bintr\relax
1110      \fi
1111      \hskip-\pxrr@dima
1112      \fi
1113      \fi
1114 }

```

\pxrr@intrude@end 末尾での進入処理。

```

1115 \def\pxrr@intrude@end{%
1116   \ifpxrr@ghost\else
1117     \pxrr@dima\pxrr@aspace\relax
1118     \ifdim\pxrr@aintr<\pxrr@dima
1119     \pxrr@dima\pxrr@aintr\relax
1120     \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。

```

1121   \if *\pxrr@bscomp
1122     \penalty\@M
1123     \hskip-\pxrr@dima
1124   \else\if :\pxrr@bscomp
1125     \penalty\pxrr@end@kinsoku
1126     \hskip-\pxrr@dima
1127     \hskip\xkanjiskip
1128   \else
1129     \penalty\pxrr@end@kinsoku
1130     \hskip-\pxrr@dima
1131     \hskip\xkanjiskip
1132   \fi\fi

```

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大きな値にする。

```

1133     \penalty\@MM
1134     \fi
1135 }

```

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

```

1136 \def\pxrr@takeout@any@protr{%
1137   \ifpxrr@any@protr
1138     \aftergroup\pxrr@any@protrtrue
1139   \fi
1140 }

```

\pxrr@ruby@main@m モノルビ。

```

1141 \def\pxrr@ruby@main@m{%
1142   \pxrr@zip@list\pxrr@body@list\pxrr@ruby@list
1143   \let\pxrr@whole@list\pxrr@res
1144   \pxrr@check@par@head

```



```

1145 \pxrr@any@protrfalse
1146 \ifpxrrDebug
1147 \pxrr@debug@show@recomp
1148 \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。

```

1149 \let\pxrr@locate@head@\pxrr@locate@inner
1150 \let\pxrr@locate@end@\pxrr@locate@inner
1151 \let\pxrr@locate@sing@\pxrr@locate@inner
1152 \ifpxrr@aprotr\else
1153 \let\pxrr@locate@end@\pxrr@locate@end
1154 \let\pxrr@locate@sing@\pxrr@locate@end
1155 \fi
1156 \ifpxrr@bprotr\else
1157 \let\pxrr@locate@head@\pxrr@locate@head
1158 \let\pxrr@locate@sing@\pxrr@locate@head
1159 \fi
1160 \def\pxrr@pre##1##2{%
1161 \pxrr@if@last{%

```

単独ブロックの場合。

```

1162 \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
1163 \pxrr@intrude@head
1164 \unhbox\pxrr@boxr
1165 \pxrr@intrude@end
1166 \pxrr@takeout@any@protr
1167 }{%

```

先頭ブロックの場合。

```

1168 \pxrr@compose@block\pxrr@locate@head@{##1}{##2}%
1169 \pxrr@intrude@head
1170 \unhbox\pxrr@boxr
1171 }%
1172 }%
1173 \def\pxrr@inter##1##2{%
1174 \pxrr@if@last{%

```

末尾ブロックの場合。

```

1175 \pxrr@compose@block\pxrr@locate@end@{##1}{##2}%
1176 \pxrr@inter@mono
1177 \unhbox\pxrr@boxr
1178 \pxrr@intrude@end
1179 \pxrr@takeout@any@protr
1180 }{%

```

中間ブロックの場合。

```

1181 \pxrr@compose@block\pxrr@locate@inner{##1}{##2}%
1182 \pxrr@inter@mono
1183 \unhbox\pxrr@boxr

```

```

1184 }%
1185 }%
1186 \let\pxrr@post\@empty
1187 \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。

1188 \if j\pxrr@mode
1189   \ifpxrr@any@protr
1190     \pxrr@ruby@redo@j
1191   \fi
1192 \fi
1193 \unhbox\pxrr@boxr
1194 }

```

\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。

```

1195 \def\pxrr@ruby@redo@j{%
1196   \pxrr@concat@list\pxrr@body@list
1197   \let\pxrr@body@list\pxrr@res
1198   \pxrr@concat@list\pxrr@ruby@list
1199   \let\pxrr@ruby@list\pxrr@res
1200   \pxrr@zip@single\pxrr@body@list\pxrr@ruby@list
1201   \let\pxrr@whole@list\pxrr@res
1202 \ifpxrr@Debug
1203 \pxrr@debug@show@concat
1204 \fi
1205 \let\pxrr@locate@sing@\pxrr@locate@inner
1206 \ifpxrr@aprotr\else
1207   \let\pxrr@locate@sing@\pxrr@locate@end
1208 \fi
1209 \ifpxrr@bprotr\else
1210   \let\pxrr@locate@sing@\pxrr@locate@head
1211 \fi
1212 \def\pxrr@pre##1##2{%
1213   \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
1214   \pxrr@intrude@head
1215   \unhbox\pxrr@boxr
1216   \pxrr@intrude@end
1217 }%
1218 \let\pxrr@inter\@undefined
1219 \let\pxrr@post\@empty
1220 \setbox\pxrr@boxr\hbox{\pxrr@whole@list}%
1221 }

```

\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。

```

1222 \def\pxrr@ruby@main@g{%
1223   \pxrr@zip@list\pxrr@body@list\pxrr@ruby@list

```

```

1224 \let\pxrr@whole@list\pxrr@res
1225 \pxrr@check@par@head
1226 \ifpxrr@Debug
1227 \pxrr@debug@show@recomp
1228 \fi
1229 \let\pxrr@locate@sing@\pxrr@locate@inner
1230 \ifpxrr@aprotr\else
1231 \let\pxrr@locate@sing@\pxrr@locate@end
1232 \fi
1233 \ifpxrr@bprotr\else
1234 \let\pxrr@locate@sing@\pxrr@locate@head
1235 \fi
1236 \def\pxrr@pre##1##2{%
1237 \pxrr@compose@block\pxrr@locate@sing@{##1}{##2}%
1238 \pxrr@intrude@head
1239 \unhbox\pxrr@boxr
1240 \pxrr@intrude@end
1241 }%
1242 \let\pxrr@inter\@undefined
1243 \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。

```

1244 \pxrr@whole@list
1245 }

```

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

```

1246 \def\pxrr@ruby@main@tg{%
1247 \pxrr@check@par@head
1248 \let\pxrr@locate@sing@\pxrr@locate@inner
1249 \ifpxrr@aprotr\else
1250 \let\pxrr@locate@sing@\pxrr@locate@end
1251 \fi
1252 \ifpxrr@bprotr\else
1253 \let\pxrr@locate@sing@\pxrr@locate@head
1254 \fi
1255 \expandafter\pxrr@compose@twoside@block\expandafter\pxrr@locate@sing@
1256 \pxrr@all@input
1257 \pxrr@intrude@head
1258 \unhbox\pxrr@boxr
1259 \pxrr@intrude@end
1260 }

1261 \def\pxrr@debug@show@input{%
1262 \typeout{----\pxrr@pkgname\space input:^^J%
1263 ifpxrr@abody = \meaning\ifpxrr@abody^^J%
1264 ifpxrr@truby = \meaning\ifpxrr@truby^^J%
1265 pxrr@ruby@fsize = \pxrr@ruby@fsize^^J%
1266 pxrr@body@zw = \pxrr@body@zw^^J%
1267 pxrr@ruby@zw = \pxrr@ruby@zw^^J%
1268 pxrr@htratio = \pxrr@htratio^^J%

```

```

1269 pxrr@ruby@raise = \pxrr@ruby@raise^^J%
1270 pxrr@ruby@lower = \pxrr@ruby@lower^^J%
1271 ifpxrr@bprotr = \meaning\ifpxrr@bprotr^^J%
1272 ifpxrr@aprotr = \meaning\ifpxrr@aprotr^^J%
1273 pxrr@side = \the\pxrr@side^^J%
1274 pxrr@bscomp = \meaning\pxrr@bscomp^^J%
1275 pxrr@ascomp = \meaning\pxrr@ascomp^^J%
1276 pxrr@bintr = \pxrr@bintr^^J%
1277 pxrr@aintr = \pxrr@aintr^^J%
1278 ifpxrr@athead = \meaning\ifpxrr@athead^^J%
1279 pxrr@mode = \meaning\pxrr@mode^^J%
1280 pxrr@body@list = \meaning\pxrr@body@list^^J%
1281 pxrr@body@count = \@nameuse{pxrr@body@count}^^J%
1282 pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
1283 pxrr@ruby@count = \@nameuse{pxrr@ruby@count}^^J%
1284 ----
1285 }%
1286 }
1287 \def\pxrr@debug@show@recomp{%
1288 \typeout{----\pxrr@pkgname\space recomp:^^J%
1289 pxrr@body@list = \meaning\pxrr@body@list^^J%
1290 pxrr@body@count = \pxrr@body@count^^J%
1291 pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
1292 pxrr@ruby@count = \pxrr@ruby@count^^J%
1293 pxrr@res = \meaning\pxrr@res^^J%
1294 ----
1295 }%
1296 }
1297 \def\pxrr@debug@show@concat{%
1298 \typeout{----\pxrr@pkgname\space concat:^^J%
1299 pxrr@body@list = \meaning\pxrr@body@list^^J%
1300 pxrr@ruby@list = \meaning\pxrr@ruby@list^^J%
1301 pxrr@whole@list = \meaning\pxrr@whole@list^^J%
1302 ----
1303 }%
1304 }

```

3.13.5 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。

```
1305 \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で `chardef` にする。

```
1306 \chardef\pxrr@zspace=\jis"2121\relax
```

`\pxrr@jprologue` 和文ルビ用の開始処理。

```
1307 \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。

```
1308 \ifpxrr@jghost
1309 \pxrr@zspace
1310 \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。

```
1311 \begingroup
1312 \pxrr@abodyfalse
1313 \pxrr@csletcs{ifpxrr@ghost}{ifpxrr@jghost}%
```

出力した全角空白の幅だけ戻しておく。

```
1314 \ifpxrr@jghost
1315 \setbox\pxrr@boxa\hbox{\pxrr@zspace}%
1316 \kern-\wd\pxrr@boxa
1317 \fi
1318 }
```

`\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 2.5pt 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。）

```
1319 \ifpxrr@aghost
1320 \IfFileExists{t1lmr.fd}{%
1321 \begingroup
1322 \fontsize{2.5}{0}\usefont{T1}{lmr}{m}{n}
1323 \endgroup
1324 \pxrr@letcs\pxrr@aghostfont{T1/lmr/m/n/2.5}%
1325 \chardef\pxrr@aghostchar=23 % compwordmark
1326 \def\pxrr@aghost{\pxrr@aghostfont\pxrr@aghostchar}%
1327 \xspcode\pxrr@aghostchar=3 %
1328 }{%else
1329 \oxrr@warn{Ghost embedding for \string\aruby\space
1330 is disabled,\MessageBreak
1331 since package lmodern is missing}%
1332 \pxrr@aghostfalse
1333 \let\pxrr@aghosttrue\relax
1334 }%
1335 \fi
```

`\pxrr@aprologue` 欧文ルビ用の開始処理。

```
1336 \def\pxrr@aprologue{%
1337 \ifpxrr@aghost
1338 \pxrr@aghost
1339 \fi
1340 \begingroup
```

```

1341 \pxrr@abodytrue
1342 \pxrr@csletcs{ifpxrr@ghost}{ifpxrr@aghost}%
1343 }

```

3.13.6 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。

```

1344 \def\pxrr@ruby@exit{%
1345 \ifpxrr@fatal@error
1346 \pxrr@fallback
1347 \fi
1348 \ifpxrr@abody
1349 \expandafter\pxrr@aepilogue
1350 \else
1351 \expandafter\pxrr@jepilogue
1352 \fi
1353 }

```

`\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。

```

1354 \def\pxrr@jepilogue{%
1355 \ifpxrr@jghost
1356 \setbox\pxrr@boxa\hbox{\pxrr@zspace}%
1357 \kern-\wd\pxrr@boxa
1358 \fi

\pxrr@?prologue の中の \begingroup で始まるグループを閉じる。
1359 \endgroup
1360 \ifpxrr@jghost
1361 \pxrr@zspace
1362 \fi
1363 }

```

`\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。

```

1364 \def\pxrr@aepilogue{%
1365 \endgroup
1366 \ifpxrr@aghost
1367 \pxrr@aghost
1368 \fi
1369 }

```