本書に関する注意

本書は姫踊子草版 1.3 号系（繭姫）の開発に伴い、配列定義ファイルについて仕様を再記述したものです。

現時点でまだ開発段階にあるため、本書の内容が変更される可能性があります。また、本書の記述通りに動作しない場合、瑕疵である場合と単に未実装である場合とがあります。

姫踊子草・繭姫に関しては次の URL が示す情報も参考にしてください。

> キー入力入れ替えソフト姫踊子草情報頁
> http://www.vector.co.jp/authors/VA011751/software/himeodorikosou/
>
> (新)姫踊子草の楽屋裏
> http://blog6.fc2.com/suzumizaki/

本書は内容を改編せず、また実費以上の金銭を回収しないという条件の下で自由に再配布可能です。

なお、本書自体がまだ未完成（キー表現と実際のキーとの対応表などがない）です。ご了承ください。

鈴見咲君高

姫踊子草用配列定義ファイル仕様書・解説書

姫踊子草における配列定義ファイルには次の四種があります。

- かな配列定義ファイル
- 英字配列定義ファイル
- 機能キー配列定義ファイル
- Numlock 時配列定義ファイル

これらは姫踊子草の設定窓口内にある配列頁で指定できる四種と対応しています。原則的にこの四種は中身の記述方法が共通しており、その適用条件のみが異なっています。

文字符号化方法

配列定義ファイルは、姫踊子草で使う他のテキスト同様に Shift_JIS と UTF-8,UTF-16 で記述することができます。

旧版との互換を保つため、かな配列に限り「暗号化された配列定義ファイル」が用いられることがありますが、現在配布される姫踊子草にはそのようなファイルは添付されておりません。

UTF が利用可能となったのは版 1.3005 号構築#153 以降です。

元々 Shift_JIS(Codepage932)のみの対応だった関係から、UTF を使う場合は必ず BOM (Byte Order Mark=0xfeff)を符号化した文字がファイル先頭に付されていなくてはなりません。

従って UTF-8N と呼ばれる方法で保存されたファイルは姫踊子草では読み出すことができません。

読み込みについては、各々の文字符号が 0x10ffff を超えない場合に限り UTF-32 にも対応しています。姫踊子草から書き出すときは UTF-32 は用いられません。

UTF-8 の場合でも 0x10ffff を超える符号を持つ文字には対応しておりません。

また、符号が 0xffff を超える文字は単に 16 ビット文字×2 として扱われています。このため、漢直用には使用できる可能性がありますが、代行ファイルへ渡す文字としては使えません。

配列定義ファイルの構造

配列定義ファイルは次の各行から成っています。

配列定義ファイルの解釈はまず行単位で以下のいずれかとして判断されます。それぞれを次節以降で個別に解説していきます。

- 注釈行
- 可視指定(かな配列定義ファイルのみ)
- 制作中宣言
- 一括入力量情報
- 漢直字明示指定
- 特殊キー指定
- 略記手法指定
- 略記宣言記述
- 個別対応記述

注釈行

行頭が半角単引用符(終了側、0x27)になっている行はその行末まで注釈として扱われ、定義としての意味を持ちません。

```
'
'注釈なので意味を持ちません。
'
```

可視指定(かな配列定義ファイルのみ)

かな配列定義ファイルに限り、かつ、同ファイルにおいて必ず、その先頭行は次の文字列から開始する必要があります。

```
visible=1
```

これは、かつての姫踊子草が特許関係の憂慮から暗号化かな配列定義ファイルを提供していたことによるものです。従ってこの内容から始まっていない場合、かな配列定義ファイルは暗号化されているものとみなして処理されます。

文字符号化方法が UTF であるファイルの場合、この行は本質的に必要のないものですが、UTF であるかどうかを問わずこの文字列で開始することを規則とします。

かな配列以外の三種においてはこの行が存在してはいけません。現時点では特に警告も出しませんが、存在しないことを規則とします。

制作中宣言

定義ファイル内に次のような行がある場合、そのファイルの中身は制作途上のものであるとみなされます。

```
DraftData=1
```

文字入力操作には影響を与えませんが、設定窓口の配列頁においてこの行の有無が編集ボタンと複製ボタンの切り替え根拠となります。

制作中宣言がない定義ファイルは、複製ボタンで複製される際に自動的に同宣言が追加されます。そして、複製の結果新しく作られたファイルを選択すると、そのファイルを編集目的で姫踊子草から呼び出すことが可能となります。

このような形でデータ提供者の元ファイルを誤って変更しないように、また、姫踊子草の更新でせっかく変更したファイルが失われたりしないようにしています。

一つの配列定義ファイル内で DraftData 行は零回もしくは一回だけ使うようにしてください。

一括入力量情報(かな配列定義ファイルのみ)

定義ファイル内の各々の定義が濁音・半濁音・拗音のどこまでを一度に入力することになっているかを指示します。かな配列定義ファイルでのみ意味を持つ情報で、この値はタイピング練習ソフト対応機能で用いられます。書式は次の通りです。

```
TypeModeDefault=8
```

原則的には自動判定可能な情報なのですが、利用する練習ソフトや外来語系の表記に対してど

のように対応すべきか、という点で揺れが生じます。そのため、配列データ提供者が基本値として指定することにしています。

指定する値は次の表の和となります。後続の実例値も参考にしてください。一つの配列定義ファイル内で TypeModeDefault 行は一度だけ使うようにしてください。

値	意味
2	濁音の濁点を後で入力する
4	半濁音の半濁点を後で入力する
8	(拗音などの)後続する小書き文字を後で入力する
(32)	ローマ字入力練習ソフトについて一打で入力した拗音を一般的なローマ字で入力する ※版 1.3000 号以降では 32 を加えなくても必要に応じて常にこの動作です。

指定値	配列例
8	NICOLA/飛鳥
12	TRON
14	五十音系/旧 JIS/新 JIS
32(0)	複層系(ローマ字入力系)/姫踊子草(かな)配列

漢直字明示指定

配列定義ファイルの内容によって変換された結果が直接的なキー表現ではなかった場合、通常は代行定義ファイルでさらに変換を試みます。しかし、漢直(漢字直接入力)を行う場合は漢字を変換されては困ることになります。具体的には Backspace に割り当てられた「退」や Enter に割り当てられた「帰」などです。

そこで次のような行を記述しておくと、このような「退」「帰」を直接出力するものとして扱うことができます。

```
WantThemselves=1
```

この指定は途中で元に戻すことができます。上記の 1 を 0 に変えた次の形を使います。

```
WantThemselves=0
```

なお、1 および 0 の指定は省略形です。0 の場合は一文字も指定しなかった場合と等価とみなされ、1 の場合は次の指定と等価とみなされます。

```
WantThemselves=読句退削入帰避間上下左右前次家終挿変無大半仮制多暗
空数却未
```

省略形ではない一般的な書法は、上記のように単に漢直対象となる文字を改行符号までにならべたものとなります（上例で二行にまたがっているのは単に本書内で一行に書くことができないからです）。

代行定義ファイルにおいて独自の漢字割り当てを行っている場合は、上例に加えてその独自割り当て分の文字を追加記述する必要があります。なお、代行定義ファイルに新たな定義を加えること自体を推奨しません。

実質的に WantThemselves=1 はすべての個別対応記述の前に置いてあればよいのですが、「未」「却」を機能的な意味で使っている場合などは適宜指定値を切り替える必要があります。

特殊キー指定

特段の指定がない場合、Shift などをのぞいてほぼすべてのキーが単独打鍵もしくは同時打鍵の対象として処理されます。

しかしながら主に親指が担当するキーについては別の入力判定を使いたい場合があります。これに対応するのがと SemiShift= と MultiDownHold= の書式です。

```
SemiShift=L
```

SemiShift で指定されたキーは、単独で押下されている間は何も出力しません。その状態で他のキーが押されるか、単独でキーが離上された時点で入力文字が決まります。

木村氏の提唱した SandS をすべてのキーに拡張したものと考えることができます。

```
MultiDownHold=LR
```

MultiDownHold で指定されたキーが連続する同時打鍵に共通のものである場合、同時打鍵ではなく押しっぱなしにしておくことができます。

これは飛鳥配列のために実装されたものですが、親指キーに限らず任意のキーでこれを用いることができます。

SemiShift と MultiDownHold は共に複数のキーを指定できますが、両方に同一のキーを指定することは（論理的に）できません。相互に重複がなければ SemiShift と MultiDownHold を一つの配列定義ファイル内で使うことはできます。

SemiShift 行と MultiDownHold 行は各々零回もしくは一回だけ一つの定義ファイル内に指定するようにしてください。

残りの部分は実際の配列定義に使われることになります。

初期の姫踊子草は、一つまたは二つのキーを同時に押下する操作一回(同時打鍵含む)で入力文字が決まる配列のみに対応していました。これを単層系と呼び、単層系の配列は個別対応記述のみで定義することができます。

これに対し、ローマ字入力などのように同時押下されない複数のキーが決まった順序で押下されて初めて入力文字が決まる配列があります。こちらは(単打)複層系と呼び、複層系の配列は少なくとも略記手法指定がないと定義することができません。

略記宣言記述は、単層・複層を問わず制作された配列データが冗長になるのを避けるために作られた仕様です。しかしながら、単打ではない複層系、つまり複層系をなす各打鍵に同時押下・同時打鍵が含まれている場合は略記宣言記述も必須となります。

個別対応記述(単層系の場合)

一行の行頭から改行符号までを水平タブ符号(0x09)で分割し、ファイルの先頭側にキー操作、右側に変換結果を書くことで配列を定義していきます。

この分割のための水平タブ符号は一つ以上あればいくつ連続してもかまいませんが、その他の符号は半角空白なども含めて分割符号としては認められませんので注意してください。

以後の例では水平タブを[TAB]で表します。

```
q[TAB]5
```

上記の例は、Q キーが押されたときに 5 キーを押したことにするという指定です。

```
q[TAB]question
```

上記のように書き換えると Q キーを押しただけで QUESTION の8つのキーをこの順番で押したかのような動作を行います。

```
q[TAB]か
```

上記のように、キー表現ではなくひらがななどが指定されたときは別に指定している代行定義ファイルの内容と照合が行われ、結果として「か」が出力されるようになります。

```
q[TAB]かい
```

先ほどと同様に、キー一つの操作に対して複数の文字を指定するとその順序で一度に処理され、結果としてこの場合「かい」が出力されるようになります。

```
q[TAB]かわ
w[TAB]き
e[TAB]くま
r[TAB]けだま
t[TAB]こころ
```

このように、一行に一組ずつ指定する方法のほか、次のようにまとめて記述することも可能です。

```
qwert[TAB]かわ$き$くま$けだま$こころ
```

[TAB]の左側には区切りがないのに対し、右側は半角$記号で区切られている点に注意してください。

ところで姫踊子草では、多くのキーを同時打鍵に用いることができます。同時打鍵の指定には半角中括弧 { } を用いて次のように記述します。

```
{ef}[TAB]どうじ
```

上記の場合、e と f をほぼ同時に打鍵することで「どうじ」の文字が出力されることになります。この中括弧の中に記されたキーは順序が逆になっても意味は変わりません。従って{ef}と{fe}は同じ意味となり、もし両方があった場合は重複指定となり論理的に成立しません。

中括弧の中で指定したキーが特殊キー指定（前述）に含まれていた場合、そのキーを先に押すことが前提となったり（SemiShift）、連続して使われる場合の処理が変化したり（MultiDownHold）します。

また、Shift キーを意味する H については特に指定がなくとも先に押下していることが前提となります。

いずれの場合でも同時に押下されたものを一打鍵として扱う点は同じです。[TAB]の左側と右側の対応付けにおいては中括弧付きの {ef} と中括弧なしの q は同じ一打鍵として扱われることになります。

```
{ef}qwert[TAB]どうじ$かわ$き$くま$けだま$こころ
```

略記宣言記述（単層系の場合）

同時打鍵に対する出力を定義していくと片方が同じキーの一群をまとめた方が整理しやすくなります。

```
{La}{Ls}{Ld}{Lf}{Lg}[TAB]ぱ$ば$げ$ど$ぞ
```

このような場合は先頭が半角等号になっている行を作り、そこで共通するキーを指定しておくと、次行以降では残りの方だけを指定すれば同時打鍵に対応する出力を定義できます。

```
=L
asdfg[TAB]ぱ$ば$げ$ど$ぞ
```

共通キーの指定は改めて他のキーを指定するか、指定を解除するまで有効です。指定の解除は等号だけの行を記述することで行います。

```
=
'
' 以後の指定は中括弧がない限り単独打鍵の定義となります。
'
```

略記手法指定（複層系の定義方法）

複層系の配列を定義する場合は、これまで説明した単層系の場合に加えて何らかの形で定義方法を拡張する必要があります。

方法の一つは略記宣言に、順番にも意味を持たせた複数の打鍵を並べて書くことで、個別対応記述の部分は最後の一打についてだけ[TAB]の左側に置くというものです。この方法を使うために、あらかじめ次のような行を記述しておきます。

```
StrokeMode=2
```

StrokeMode=2 を記述した次の行からは略記宣言に複数の打鍵表現を記すことができます。

```
=jp
agfds[TAB]ふぁ$ふぃ$ふ$ふぇ$ふぉ
{La}{Lg}{Lf}{Ld}{Ls}[TAB]ふぁー$ふぃー$ふー$ふぇー$ふぉー
```

上記の例の場合、jpa という順に入力すると「ふぁ」が出力されます。三打目が左親指との同時打鍵であれば「ふぁー」になります。

最後の一打鍵が同時打鍵であって、単層系の時のように略記を行いたい場合は中括弧で共通キーひとつを括ります。

```
=jp
agfds[TAB]ふぁ$ふぃ$ふ$ふぇ$ふぉ
=jp{L}
agfds[TAB]ふぁー$ふぃー$ふー$ふぇー$ふぉー
```

略記宣言の末尾以外では、同時押下・同時打鍵を示す中括弧の中に二つのキー表記が必要となります。

```
={Lj}p{L}
agfds[TAB]ひゃー$ひぃー$ふゅー$ひぇー$ひょー
```

複層系の定義にはもうひとつの方法があります。

```
StrokeMode=1
```

StrokeMode=1 の場合、個別対応記述における中括弧の意味が変化し、単独打鍵の連続操作を意味するようになります。同時押下・同時打鍵の指定はできません。同時押下の意味を失っていますので、三つ以上のキー表現を記すことはできます。

StrokeMode=1 の場合でも、略記宣言の部分は StorkeMode=2 の場合と同じ効果を持ちますから、重打複層系の配列を定義することも可能です。

ただし、略記宣言の末尾で同時押下・同時打鍵の一方を指定していて、個別対応記述側が中括弧でくくられている場合の動作は未定義です。現在は単にその部分だけ無効になりますが、将来的に変更する可能性がありますので避けてください。

最後に、略記手法指定 StrokeMode は必要に応じて同一定義ファイル内で切り替えて使用できます。単層系しか指定できない初期状態に戻すときは次の記述を行います。

```
StrokeMode=0
```