日本語 OpenVMS
ユーザ・キー定義 利用者の手引き

AA-PU8UE-TE.2

2009 年 11 月

本書は，日本語 OpenVMS でかな漢字変換のキー定義を変更する方法についての解説書です。

改訂/更新情報:	日本語 OpenVMS V6.2 『ユーザ・キー定義 利用者の手引き』の改訂版です。
ソフトウェア・バージョン:	日本語 OpenVMS Integrity 日本語 OpenVMS Alpha V7.2 日本語 OpenVMS VAX V7.2

日本ヒューレット・パッカード株式会社

本書は，日本語 VAX DOCUMENT V 2.1を用いて作成しています。

目次

例

図

表

まえがき

本書の目的

本書は，かな漢字変換のキー定義を変更する方法について説明しています。かな漢字変換のキー定義は，ユーザ・キー定義ライブラリ (IMLIB) を使って作成されたアプリケーションを使用している場合にのみ，本書に書かれた方法によって変更することができます。 IMLIB についての詳細は，『IMLIB/OpenVMS ライブラリ・リファレンス・マニュアル』をご覧ください。

本書の構成

本書は 7 つの章と 3 つの付録から構成されています。

第 1 章　IMLIB の概要について説明します。
第 2 章　標準のキー定義とその使い方について説明します。
第 3 章　個々のキーの定義を変更する方法について説明します。
第 4 章　PROFILE について説明します。
第 5 章　細かいキー定義の変更 (入力シーケンスのカスタマイズ) を行う方法について説明します。
第 6 章　KEYBIND コンパイラの使い方について説明します。
第 7 章　KEYBIND ファイルを作成する時の注意点について説明します。
付録 A　KEYBIND ファイルで使うことのできるキーについて説明します。
付録 B　KEYBIND ファイルのシンタックスについて説明します。
付録 C　KEYBIND コンパイラのエラーについて説明します。

表記法

本書では，以下の表記法を使用します。

表記法	意味
Return	四角形で囲まれたこの記号は，キーボードのキーを押すことを示します。たとえば，Return は Return キーを押すことを示します。
Ctrl/x	Ctrl/x の記号は， Ctrl キーを押しながら，同時にあるキーを押すことを示します。たとえば，Ctrl/c は Ctrl キーと c 文字キーを同時に押します。

表記法	意味
[ ]	大括弧は，項目が省略可能であることを示します。
英大文字	英大文字は，コマンド，修飾子，パラメータ，ルーチン名，ファイル名，ファイル保護コード名，システム特権の短縮形を示します。

1

ユーザ・キー定義ライブラリ (IMLIB) の概要

1.1 IMLIB とは

IMLIB は，かな漢字変換のキー定義を，ユーザが自分の好みにあわせて変更するためのライブラリです。 IMLIB を使って作成されたアプリケーションでは，文字入力におけるかな漢字変換のキー定義を，ユーザが好みにあわせて変更することができます。また，複数のアプリケーションで共通のキー定義を使用することができます。

ユーザ・キー定義は， IMLIB を使ったアプリケーションにのみ有効です。使用しているアプリケーションにおいてユーザ・キー定義が使えるかどうかは，個々のアプリケーションのマニュアルを参照してください。

1.2 IMLIB の基本的な要素

IMLIB によるユーザ・キー定義を使用するにあたってユーザが知っておかなければならないものは以下のとおりです。

1. PROFILE

   かな漢字変換を行うときに必要となる情報のうち，キー定義以外のものがこのファイルに書かれます。 PROFILE ファイルは標準のものが提供されますが，ユーザが自分の好みにあわせて変更することもできます。

   たとえば，変換中のビデオ属性は， PROFILE ファイルを使ってユーザが指定することができます。これによって現在の文節を反転表示にしたり，下線付き表示にしたり，好みにあわせて変更できます。次に説明する KEYBIND ファイルの指定も PROFILE ファイルの中で行います。

2. KEYBIND ファイル

   かな漢字変換のキー定義を行うファイルです。 KEYBIND ファイルには「テキスト形式」と「バイナリ形式」があり，ユーザはテキスト形式で作成した後，KEYBIND コンパイラでバイナリ形式に変換しなければなりません。

   KEYBIND ファイルは，標準として JVMS, EVEJ, LEIA, TARO の 4 種類が提供されます。ユーザはこれらの定義の中で好みのものを選ぶことも，新しく作成することもできます。

3. KEYBIND コンパイラ

   テキスト形式の KEYBIND ファイルをバイナリ形式の KEYBIND ファイルに変換するときに使います。

2

標準のキー定義と使い方

2.1 提供されるキー定義とその特徴

IMLIB は，あらかじめ作成されたキー定義を 4 種類提供します。ユーザは，簡単な操作で，これらのキー定義のうちの 1 つを選択して使うことができます。ユーザが何の変更もしなかったときには， JVMS キーが使われます。

1. JVMS キー

   日本語 OpenVMS 標準のキー定義です。 JVMS キーを使うと，かな漢字変換キーと VMS の行編集キーとが重ならないという特徴があります。詳しくは，『日本語ライブラリ 利用者の手引き』を参照してください。

2. EVEJ キー

   EVEJ エディタのキー定義です。日本語 EVE の EVEJ キーパッド・モードで使われています。詳しくは『日本語 EVE ユーザーズ・ガイド』を参照してください。

3. LEIA キー

   LEIA エディタのキー定義です。数字キーパッドを使ってかな漢字変換を行います。詳しくは，『日本語ライブラリ 利用者の手引き』を参照してください。

4. TARO キー

   ワープロ・ソフトの一太郎[1]のキー定義に似たキー定義です。

IMLIB でのキー定義では，かな漢字変換およびかな漢字変換中の編集に関するもののみが定義できます。カーソル移動や単語消去のような通常の編集機能は，各アプリケーションが定義するキーで行われます。

2.2 EVEJ キー，LEIA キーまたは TARO キーを選択する方法

ユーザが何も選択しないときには，キー定義は JVMS キーが使われます。 EVEJ キー， LEIA キーまたは TARO キーを使うときには次のようにします。

- 論理名 IM$PROFILE を次のように定義する

```
$ DEFINE IM$PROFILE IM$PROFILE_EVEJ
                    (EVEJ キーを使う場合)
```

[1] 一太郎は株式会社ジャストシステムの登録商標です。

または

```
$ DEFINE IM$PROFILE IM$PROFILE_LEIA
                (LEIA キーを使う場合)
```

または

```
$ DEFINE IM$PROFILE IM$PROFILE_TARO
                (TARO キーを使う場合)
```

この定義をユーザの LOGIN.COM に入れておくと，以後のセッションで選択したキー定義を使うことができます。

新しいキー定義を使うには，アプリケーションを再起動してください。

3

個々のキー定義の変更

この章では，標準で提供されるキー定義ファイルを使って，個々のキーの定義を変更する方法について説明します。

3.1 標準のキー定義ファイル

標準で提供されるキー定義ファイルは，テキストとバイナリの両方の形式で提供されます。以下に提供されるファイルの説明をします。

バイナリ形式のキー定義ファイルは，テキスト形式のキー定義ファイルをコンパイルしたものです。アプリケーションが実行されるときには，バイナリ形式のキー定義ファイルが使われます。

提供されるバイナリ形式のキー定義ファイルを以下に示します。

キー定義ファイル	説明
SYS$LIBRARY:IM$KEY_JVMS.IM$DAT	JVMS キー
SYS$LIBRARY:IM$KEY_EVEJ.IM$DAT	EVEJ キー
SYS$LIBRARY:IM$KEY_LEIA.IM$DAT	LEIA キー
SYS$LIBRARY:IM$KEY_TARO.IM$DAT	TARO キー
SYS$LIBRARY:IM$KEY_JVMS_LEVEL2.IM$DAT	JVMS キー
SYS$LIBRARY:IM$KEY_EVEJ_LEVEL2.IM$DAT	EVEJ キー
SYS$LIBRARY:IM$KEY_LEIA_LEVEL2.IM$DAT	LEIA キー
SYS$LIBRARY:IM$KEY_TARO_LEVEL2.IM$DAT	TARO キー

提供されるテキスト形式のキー定義ファイルを以下に示します。

キー定義ファイル	説明
IM$EXAMPLES:IM$KEY_JVMS.DAT	JVMS キー用マクロ
IM$EXAMPLES:IM$KEY_EVEJ.DAT	EVEJ キー用マクロ
IM$EXAMPLES:IM$KEY_LEIA.DAT	LEIA キー用マクロ
IM$EXAMPLES:IM$KEY_COMMON_BODY.DAT	JVMS, EVEJ, LEIA キー用本体
IM$EXAMPLES:IM$KEY_TARO.DAT	TARO キー用マクロ
IM$EXAMPLES:IM$KEY_TARO_BODY.DAT	TARO キー用本体

キー定義ファイル	説明
IM$EXAMPLES:IM$KEY_JVMS_LEVEL2.DAT	JVMS キー用マクロ
IM$EXAMPLES:IM$KEY_EVEJ_LEVEL2.DAT	EVEJ キー用マクロ
IM$EXAMPLES:IM$KEY_LEIA_LEVEL2.DAT	LEIA キー用マクロ
IM$EXAMPLES:IM$KEY_COMMON_BODY_LEVEL2.DAT	JVMS, EVEJ, LEIA キー用本体
IM$EXAMPLES:IM$KEY_TARO_LEVEL2.DAT	TARO キー用マクロ
IM$EXAMPLES:IM$KEY_TARO_BODY_LEVEL2.DAT	TARO キー用本体

テキスト形式のキー定義ファイルは「マクロ」と「本体」に分かれています。「マクロ」で個々のキーの機能を定義し，「本体」でキー定義の状態の変化を記述しています。

ファイル名に「LEVEL2」を含むファイルは，半角カナ変換用のキー定義を追加したものです。これらを使用するには，アプリケーションが半角カナをサポートしていることが必要です。

3.2 キー定義を変更する方法

個々のキー定義を変更するには，マクロが書かれたファイルを変更してコンパイラでバイナリ形式に変換します。次にキー定義を変更する例を示します。キーボード上のキーの名称については付録 A を参照してください。

JVMS キーでは，ひらがな変換は Ctrl/L キーに定義されています。そこで，ひらがな変換キーを Ctrl/H キーに変更することを例に説明します。

1. JVMS キー用のマクロ・ファイルを自分のログイン・ディレクトリにコピーします。

```
$ SET DEFAULT SYS$LOGIN:
$ COPY IM$EXAMPLES:IM$KEY_JVMS.DAT MY_JVMS.DAT
```

2. エディタを使って MY_JVMS.DAT の中の 1 行を次のように変更します。

```
(変更前) hiragana_henkan = CTRL_L;
(変更後) hiragana_henkan = CTRL_H;
```

3. 変更されたファイルをコンパイルします。

```
$ KEYBIND MY_JVMS.DAT
```

コンパイルによって MY_JVMS.IM$DAT が作られます。

4. PROFILE を自分のログイン・ディレクトリにコピーします。

```
$ COPY IM$DEFAULTS:IM$PROFILE.DAT SYS$LOGIN:
```

5. エディタを使って，ログイン・ディレクトリにある IM$PROFILE.DAT の中の 1 行を次のように変更します。

```
(変更前) DEC-JAPANESE.KEY.keybind : IM$KEY_JVMS
(変更後) DEC-JAPANESE.KEY.keybind : SYS$LOGIN:MY_JVMS
```

以上で，キー定義は変更されます。新しいキー定義を使うには，アプリケーションを再起動してください。

4

PROFILE

この章では， PROFILE について詳しく説明します。

4.1 PROFILE とは？

PROFILE は，かな漢字変換におけるキー定義以外の付加情報が書かれたファイルです。ユーザは，システムが提供する PROFILE，ユーザ独自の PROFILE のどちらでも使うことができます。

PROFILE は通常のテキスト・ファイルで， INDEX とそれに対応する VALUE の 1 対 1 対応の形で，情報が書き込まれています。 IMLIB を使うアプリケーションは，PROFILE に書かれた情報を使って，かな漢字変換の動作を決定します。

4.2 標準の PROFILE

通常，標準の PROFILE として， IM$DEFAULTS:IM$PROFILE.DAT が使われます。論理名 IM$DEFAULTS は，

```
IM$DEFAULTS=SYS$LOGIN:,SYS$LIBRARY:
```

と定義されていますので，まず SYS$LOGIN:IM$PROFILE.DAT が検索され，それが存在しないときに SYS$LIBRARY:IM$PROFILE.DAT が使われます。したがって，ユーザ独自の PROFILE を使うときには，ディレクトリに SYS$LOGIN:IM$PROFILE.DAT という PROFILE を置いてください。また，PROFILE ファイルは，論理名 IM$PROFILE を使って直接指定することもできます。

4.3 PROFILE の構文規則

PROFILE の個々の INDEX と VALUE の定義はテキストファイルの 1 行の中で行われます。 INDEX と VALUE はコロン (:) をはさんだ形で以下のように書かれます。

```
INDEX-STRING : VALUE-STRING
```

INDEX-STRING と VALUE-STRING で使うことのできる文字は，以下の通りです。

- アルファベット文字と数字 (0..9, A..Z, a..z)

- ドル記号 ($), マイナス記号 (-), ピリオド (.), 下線 (_)

スペース文字とタブ文字はセパレータとして使われます。 INDEX-STRING 中にはセパレータを含むことはできませんが， VALUE-STRING 中にはセパレータを含むことができます。ただし，セパレータは， VALUE-STRING の最初の文字と最後の文字に使うことはできません。

1 行中で，感嘆符 (!) またはシャープ記号 (#) 以降の文字は，注釈として扱われます。

PROFILE の中で INDEX に指定される文字列は，大文字と小文字が区別されませんので，大文字と小文字のどちらを使っても，同じものとして扱われます。これに対して VALUE 文字列は，大文字と小文字が別の文字として扱われますので，指定するときには大文字と小文字を確実に区別しなければなりません。

INDEX にワイルドカードを使うことはできません。

4.4 共通に使用される INDEX

複数のアプリケーションで共通に使用するための INDEX とそれに対応する VALUE が定められています。ここではそれらの説明をします。アプリケーションは，できる限り共通の INDEX で指定した動作をするように作られていますので，ここで示される INDEX に対応する VALUE を変えることによって， IMLIB をサポートするすべてのアプリケーションの動作が変わります。

共通に使用される INDEX を以下に示します。かな漢字変換に関する共通の INDEX は，すべて "DEC-JAPANESE."で始まります。

共通に使用される INDEX

DEC-JAPANESE.CHAR.Codeset

VALUE: deckanji, sdeckanji, eucjp, sjis
説明: アプリケーションが使用する漢字コードセットを指定します。 deckanji は DEC 漢字， sdeckanji は SuperDEC 漢字， eucjp は日本語 EUC， sjis はシフト JIS です。省略時設定は DEC 漢字です。

DEC-JAPANESE.CHAR.jisKana

VALUE: on | off (小文字)
説明: コードセットとして DEC 漢字を使用しているときに半角カナを使用するかどうかを指定します。

DEC-JAPANESE.DISP.currentClauseRendition

VALUE: none | bold | blink | reverse | underline (小文字)

説明: 漢字変換時に，現在の文節に付けられるビデオ属性を指定します。

DEC-JAPANESE.DISP.inputRendition

VALUE: none | bold | blink | reverse | underline (小文字)
説明: 文字入力時に，変換範囲にある文字に付けられるビデオ属性を指定します。

DEC-JAPANESE.DISP.kanaRendition

VALUE: none | bold | blink | reverse | underline (小文字)
説明: ひらがな，カタカナ，全角，半角変換時に，変換範囲に付けられるビデオ属性を指定します。

DEC-JAPANESE.DISP.leadingClauseRendition

VALUE: none | bold | blink | reverse | underline (小文字)
説明: 漢字変換時に，現在の文節より前の文節に付けられるビデオ属性を指定します。

DEC-JAPANESE.DISP.preEditColumn

VALUE: current | 整数 (小文字)
説明: かな漢字変換を行うスクリーン上のカラムを示します。
current は現在カーソルがあるカラムを示します。整数はスクリーン上のカラム位置を数字で指定するときに使います。

DEC-JAPANESE.DISP.preEditRow

VALUE: current | top | bottom | 整数 (小文字)
説明: かな漢字変換を行うスクリーン上の行を示します。
top は 1 行目， bottom は最下行， current は現在カーソルがある行を示します。整数はスクリーン上の行番号を数字で指定するときに使います。

DEC-JAPANESE.DISP.trailingClauseRendition

VALUE: none | bold | blink | reverse | underline (小文字)
説明: 漢字変換時に，現在の文節より後の文節に付けられるビデオ属性を指定します。

DEC-JAPANESE.ECHO.ascii

VALUE: hankaku | zenkaku (小文字)
説明: アルファベット文字を入力したときに半角で表示されるか全角で表示されるかを指定します。

DEC-JAPANESE.ECHO.autoRomanKana

VALUE: on | off (小文字)
説明: 入力時にローマ字から指定された字種への自動変換を行うかどうかを指定します。自動ローマ字かな変換した時のエコー字種は， DEC-JAPANESE.ECHO.kana で指定されたものが使われます。

DEC-JAPANESE.ECHO.kana

VALUE: jiskana | hiragana | katakana (小文字)
説明: カナが入力されたときの表示方法を指定します。
jiskana は半角のカナを， hiragana と katakana は全角を意味します。

DEC-JAPANESE.ECHO.romkanaDB

VALUE: ローマ字かな変換テーブルファイルを示す文字列
説明: ローマ字かな変換時に使用するローマ字・かなの対応を記述したファイル名を指定します。
無指定時はライブラリ内部の対応表を使用します。
ローマ字かな変換テーブルファイルのサンプルとして
IM$EXAMPLES:IM$ROMKANA_DB.TEMPLATE
というファイルが用意されています。

DEC-JAPANESE.KEY.keybind

VALUE: KEYBIND ファイルを示す文字列
説明: KEYBIND ファイルを示す文字列には，必要なら拡張子, ".im$dat"が付加されて使用されます。ただし，拡張子が明示的に指定されたときは指定されたものが使用されます。 IMLIB の SET KEYBIND が指定された keybind ファイルのロードに失敗した場合は, DEC_JAPANESE.key.keybind_1 のように "_数字 "のついた INDEX を 1 から 9 まで順に探します。

DEC-JAPANESE.OUTRANGE.clauseNumber

VALUE: none | rotate | done (小文字)
説明: 現在の文節が最初の文節にあるときに「前文節」キーが押された場合，または現在の文節が最後の文節にあるときに「次文節」キーが押された場合の動作を指定します。
none のときは何も実行しません。 rotate が指定されると，現在の文節が最初の文節にあるときには最後の文節に，現在の文節が最後にあるときには最初の文節に移動します。
done のときは，現在の変換を終了してアプリケーション指定の動作を実行します。

DEC-JAPANESE.OUTRANGE.clauseSize

VALUE: none | rotate | done (小文字)
説明: 文節の大きさが最小になっているときに「文節縮小」キーが押された場合，または文節の大きさが最大になっているときに「文節拡大」キーが押された場合の動作を指定します。
none のときは何も実行しません。 rotate が指定されると，文節の大きさが最小のときには最大に，文節の大きさが最大のときは最小になります。
done のときは，現在の変換を終了してアプリケーション指定の動作を実行します。

DEC-JAPANESE.OUTRANGE.cursorPosition

VALUE: none | done (小文字)
説明: カーソルが最初の文字にあるときに「左移動」キーが押された場合，またはカーソルが最後の文字にあるときに「右移動」キーが押された場合の動作を指定します。
none のときは何も実行しません。 done のときは，現在の変換を終了してアプリケーション指定の動作を実行します。

PROFILE の例を例 4–1 に示します。

例 4–1 JVMS **キー用の** PROFILE

```
DEC-JAPANESE.CHAR.jisKana : off
DEC-JAPANESE.DISP.currentClauseRendition : reverse
DEC-JAPANESE.DISP.inputRendition : bold
DEC-JAPANESE.DISP.kanaRenditon : bold
DEC-JAPANESE.DISP.leadingClauseRendition : none
DEC-JAPANESE.DISP.preEditColumn : current
DEC-JAPANESE.DISP.preEditRow : current
DEC-JAPANESE.DISP.trailingClauseRendition : none
DEC-JAPANESE.ECHO.ascii : hankaku
DEC-JAPANESE.ECHO.autoRomanKana : off
DEC-JAPANESE.ECHO.kana : hiragana
DEC-JAPANESE.KEY.keybind : IM$KEY_JVMS_LEVEL2
DEC-JAPANESE.KEY.keybind_1 : IM$KEY_JVMS
DEC-JAPANESE.OUTRANGE.clauseNumber : rotate
DEC-JAPANESE.OUTRANGE.clauseSize : none
DEC-JAPANESE.OUTRANGE.cursorPosition : done
```

5
入力シーケンスのカスタマイズ

前章までは，かな漢字変換のキーの割り当ての変更および PROFILE の変更という比較的簡単にキー定義を変更する方法について説明してきました。この章以降の章では，より細かいキー定義の変更 (入力シーケンスのカスタマイズ) を行う方法が説明されています。

入力シーケンスのカスタマイズは，かな漢字変換入力に関する基本的な知識が必要です。少しの変更でも，かな漢字変換の動作全体に影響を与えることがありますので，注意してください。

キー定義のシンタックスは付録 B を参照してください。

5.1 KEY と ACTION

キー定義はキーボード上のキーの名称と，実行される機能 (ACTION) の関係を記述した KEYBIND ファイルを作ることで行います。 1 つの KEY に対して複数の ACTION を定義することも，複数のキーに対して 1 つの ACTION を定義することもできます。

例 5–1 に KEYBIND ファイルの中で， KEY と ACTION の関係を記述した部分を示します。この例では Ctrl/K キーを押すとカタカナ変換，Ctrl/L キーを押すとひらがな変換をするように記述されています。また例の 3 行目に示されるように，コンマ (,) で区切ることで複数の ACTION を指定することもできます。

例 5–1 KEY と ACTION の記述

```
CTRL_K : KATAKANA;
CTRL_L : HIRAGANA;
TYPING_KEYS : START, ECHO;
```

5.2 定義できるキーの範囲

IMLIB のかな漢字変換キー定義が対象とするキーは， VT280 シリーズ端末および VT382 端末でサポートされるすべてのキーおよびいくつかの特殊キーです。 IMLIB がサポートするキー名は，付録 A を参照してください。 IMLIB がサポートするキー名のうち注意が必要なキーについて以下に説明します。

- CTRL_C

  Ctrl/Cキーは，OpenVMS では "Cancel"に使われますので，通常の端末設定では定義した動作を実行させることはできません。

- CTRL_O

  Ctrl/Oキーは，OpenVMS では画面出力の ON/OFF を切り替える目的で使われますので，通常の端末設定では定義した動作を実行させることはできません。

- CTRL_Q

  Ctrl/Qキーは，OpenVMS ではフロー・コントロールの "XON"に使われますので，通常の端末設定では定義した動作を実行させることはできません。

- CTRL_S

  Ctrl/Sキーは，OpenVMS ではフロー・コントロールの "XOFF"に使われますので，通常の端末設定では定義した動作を実行させることはできません。

- CTRL_Y

  Ctrl/Yキーは，OpenVMS では "Interrupt"に使われますので，通常の端末設定では定義した動作を実行させることはできません。

- OTHERS

  すべてのキーにマッチします。どのキーを押しても同じ動作をさせたいときに使用します。

- TYPING_KEYS

  すべての画面に表示されるキー (文字キー，数字キー) にマッチします。

- VOID

  どのキーともマッチしません。マクロ定義において，ある機能にどのキーも定義したくないときに使用します。

5.3 キーの組合せの表記法

KEYBIND ファイルには，複数のキーの組合せで 1 つの機能を実行させたり，2 つのキーに同じ機能を持たせるような記述をすることができます。例 5–2 に組み合わせたキーの記述の例を示します。

複数のキーの組合せで 1 つの機能を実行させるにはキー名をプラス記号 (+) でつなぎます。例の 1 行目は，KEYBIND ファイルの中の 1 行を抜き出したもので，PF1キーを押した後KP7キーを押すことで，カタカナ変換を実行することを指示しています。

2 つのキーのどちらを押しても同じ機能を実行させるにはキー名をコンマ (,) でつなぎます。例の 2 行目は，KEYBIND ファイルの中の 1 行を抜き出したもので，Ctrl/Lキーを押しても，Ctrl/Hキーを押してもひらがな変換を実行することを指示しています。

また例の 3 行目のようにかっこを使ってより複雑なキーの記述をすることもできます。この例は，PF1キーを押した後Ctrl/Fキーを押しても，あるいはCtrl/Gキーを押した後Ctrl/Fキーを押しても，半角変換を実行することを指示しています。

例 5–2 組み合わせたキーの記述

```
PF1 + KP7 : KATAKANA;
CTRL_L, CTRL_H : HIRAGANA;
(PF1, CTRL_G) + CTRL_F : HANKAKU;
```

5.4 ACTION とその動作

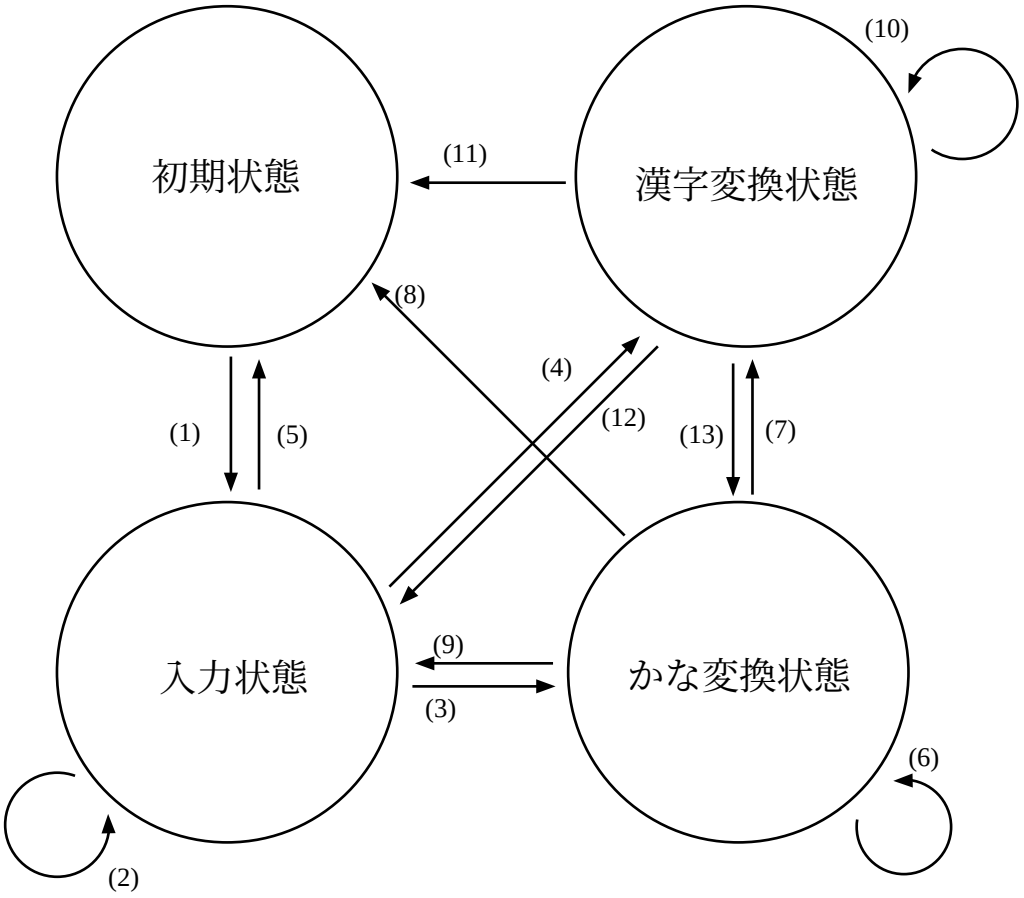
かな漢字変換を行うための個々の機能のことを ACTION と呼びます。かな漢字変換のキー定義を行う時，キーに対応する機能を ACTION の組合せで記述します。

ACTION によって実行される機能は，細かく規定されています。ユーザは ACTION によって，かな漢字変換におけるアプリケーションの動作をコントロールできます。

5.5 かな漢字変換の内部状態

かな漢字変換を実行しているときには，内部的にいくつかの状態が存在します。ACTION の実行は内部状態によって変わりますので， KEYBIND ファイルをカスタマイズするユーザはこれらの内部状態を意識しておかなければなりません。図 5–1 は，内部状態と状態間の遷移を説明しています。

図 5–1　かな漢字変換の内部状態と状態間の遷移



- 初期状態

  かな漢字変換を行っていない状態です。この状態にあるときアプリケーションは，かな漢字変換に関係しないアプリケーションの機能を実行します。たとえば，ユーザがかな漢字変換に関係しないところでカーソルを移動しているときが，この状態にあたります。カーソル移動はアプリケーションの機能であり，IMLIB は関係しません。

- 入力状態

  ユーザがかな漢字変換をするために，「ひらがな」や「ローマ字」を入力しているときがこの状態です。アプリケーションによっては，かな漢字変換の対象となる部分を高輝度やボールド表示にすることがあります。入力途中で間違いに気づいたときの編集は，入力状態で行われます。漢字変換をした後に入力の間違いに気づき，入力文字列に戻したときもこの状態になります。

- かな変換状態

入力した文字列を「カタカナ」，「全角」などに変換した状態です。
さらに， RESTORE_ECHO アクションによって一時的に変換が解除されたときもこの状態になります。

- 漢字変換状態

  かな漢字変換を行った後の状態です。文節縮小，文節移動，文節カタカナ変換などを行っているときはこの状態にあたります。

5.6 状態間の移動

図 5–1 に示される状態の間の遷移は， ACTION やその他の条件によって起こります。ここでは，状態間の遷移を引き起こす条件について説明します。

表 5–1 は ACTION による状態間の遷移を示す表です。この表は， ACTION を実行する前の状態と実行後の状態をまとめたものです。表の中で使われている表記は以下の通りです。

- カーソル

  カーソル移動に関連した ACTION です。 MOVE_LEFT, MOVE_RIGHT, HEAD, TAIL, DELETE がこれにあたります。

- かな変換

  単純な変換を行う ACTION です。 HIRAGANA, KATAKANA, HANKAKU_KANA, HANKAKU, ZENKAKU, SYMBOL, UPPER, LOWER がこれにあたります。

- 文節操作

  漢字変換中の文節に関連した ACTION です。 CLA_HIRAGANA, CLA_KATAKANA, CLA_HANKAKU_KANA, CLA_ZENKAKU, CLA_HANKAKU, NEXT_CLAUSE, PREV_CLAUSE, SHORTEN_CLAUSE, EXTEND_CLAUSE, NEXT_CANDIDATE, PREV_CANDIDATE がこれにあたります。

- error

  今の状態でこのような ACTION は許されないことを示します。
  KEYBIND ファイルに，このような ACTION が記述されていると， KEYBIND コンパイラは警告を出します。

- no action

  この ACTION が来たときにはアプリケーションは何も実行しないことを示します。

表 5–1 状態間の遷移

ACTION	初期状態	入力状態	かな変換状態	漢字変換状態
START	入力状態 (1)	error	error	error
START_SELECTED	入力状態 (1)	error	error	error
ECHO	error	入力状態 (2)	error	error
カーソル	error	入力状態 (2)	error	error
かな変換	error	かな変換状態 (3)	かな変換状態 (6)	かな変換状態 (13)
CONVERT	error	漢字変換状態 (4)	漢字変換状態 (7)	no action
DONE	no action	初期状態 (5)	初期状態 (8)	初期状態 (11)
文節操作	error	error	error	漢字変換状態 (10)
RESTORE_STRING	error	no action	入力状態 (9)	入力状態 (12)
RESTORE_ECHO	error	かな変換状態 (3)	かな変換状態 (6)	かな変換状態 (13)

5.7 ACTION の意味

- START

  新しい入力を開始します。 START の後，状態は「入力状態」に移ります。

- START_SELECTED

  アプリケーションのセレクトの部分を変換対象にします。 START_SELECTED の後，状態は「入力状態」に移ります。

- ECHO

  入力されたキーに対応した文字を表示します。自動ローマ字かな変換が指定されているときには，ローマ字をかなに変換して表示します。入力されたキーがかなキーのときの表示方法 (半角カタカナ，全角ひらがななど) はアプリケーションの任意です。もしその文字が表示できる文字でないときには，何も実行しません。アプリケーションは関係するバッファにこの文字を格納します。この機能は「入力状態」にあるときのみ有効です。

- DELETE

  カーソルの直前の文字を削除します。この機能は「入力状態」にあって，カーソルが変換領域の中にあるときのみ有効です。

- MOVE_LEFT

  カーソルを左に移動します。この機能は「入力状態」にあって，カーソルが変換領域の中にあるときのみ有効です。

- MOVE_RIGHT

  カーソルを右に移動します。この機能は「入力状態」にあって，カーソルが変換領域の中にあるときのみ有効です。

- HEAD

  カーソルを変換領域の先頭に移動します。この機能は「入力状態」にあるときのみ有効です。

- TAIL

  カーソルを変換領域の末尾に移動します。この機能は「入力状態」にあるときのみ有効です。

- HIRAGANA

  変換領域の中のアルファベットとカタカナをひらがなに変換します。

- KATAKANA

  変換領域の中のアルファベットとひらがなをカタカナに変換します。

- HANKAKU_KANA

  変換領域の中のアルファベットとひらがな，カタカナを半角カナに変換します。

- ZENKAKU

  変換領域の中の半角文字を全角に変換します。

- HANKAKU

  変換領域の中の全角文字を半角に変換します。

- UPPER

  変換領域の中のアルファベットの小文字を大文字に変換します。

- LOWER

  変換領域の中のアルファベットの大文字を小文字に変換します。

- SYMBOL

  変換領域の中のシンボル変換を実行します。 JIS 区点入力， DEC 漢字コード入力ができます。

- CONVERT

  変換領域に対してかな漢字変換を実行します。この機能は「入力状態」または「かな変換状態」にあるときのみ有効です。

- NEXT_CANDIDATE

  現在の文節に対して次候補を得る操作を実行します。この機能は「漢字変換状態」にあるときのみ有効です。

- PREV_CANDIDATE

  現在の文節に対して前候補を得る操作を実行します。この機能は「漢字変換状態」にあるときのみ有効です。

- CLA_HIRAGANA

  現在の文節中のカタカナをひらがなに変換します。この機能は「漢字変換状態」にあるときのみ有効です。

- CLA_KATAKANA

  現在の文節中のひらがなをカタカナに変換します。この機能は「漢字変換状態」にあるときのみ有効です。

- CLA_HANKAKU_KANA

  現在の文節中のひらがな，カタカナを半角カナに変換します。この機能は「漢字変換状態」にあるときのみ有効です。

- CLA_ZENKAKU

  現在の文節中の半角のアルファベットを全角に変換します。この機能は「漢字変換状態」にあるときのみ有効です。現在のかな漢字変換の方法では，この機能はサポートできません。

- CLA_HANKAKU

  現在の文節中の全角のアルファベットを半角に変換します。この機能は「漢字変換状態」にあるときのみ有効です。現在のかな漢字変換の方法では，この機能はサポートできません。

- NEXT_CLAUSE

  現在の文節を文末方向に 1 つ移動します。この機能は「漢字変換状態」にあるときのみ有効です。

- PREV_CLAUSE

  現在の文節を文頭方向に 1 つ移動します。この機能は「漢字変換状態」にあるときのみ有効です。

- SHORTEN_CLAUSE

  現在の文節を縮小します。この機能は「漢字変換状態」にあるときのみ有効です。

- EXTEND_CLAUSE

  現在の文節を拡大します。この機能は「漢字変換状態」にあるときのみ有効です。

- DONE

  「初期状態」に戻ります。「漢字変換状態」にある場合は，変換の学習を行います。バッファの内容をすべてクリアし，画面上のビデオ属性を消します。

- RESTORE_STRING

  変換領域内を入力文字列に戻して，「入力状態」になります。これにより，文字列の編集ができるようになります。

- RESTORE_ECHO

  変換領域内を入力文字列に戻して，「かな変換状態」へ移ります。次の入力文字が変換操作以外の入力であれば，確定後，入力に応じた操作を行います。

- NONE

そのキーが定義されていないことを明示的に指定します。 NONE を指定したときには, NONE に対して他の ACTION を同時に指定することはできません。NONE で他の ACTION を同時に指定したときの結果は保証されません。

ACTION には上に示したものに加えて GOTO が存在します。 GOTO は特殊な ACTION で， KEYBIND ファイルの中の STATE を移動するために使われます。STATE については，第 5.8 節を参照してください。 GOTO は， 1 つのキーにつき最後の ACTION として 1 つだけ指定することができます。

5.8 STATE とは？

IMLIB では， 1 つのキーに，場面により異なる機能を定義することができるようになっています。これを可能にするのが STATE です。キー定義は，必ずどれかの STATE に属しますので， 1 つのキーには存在する STATE の数だけ違った機能を持たせることができます。

アプリケーションを起動したとき最初に属する STATE は "initial"です。 STATE の名称のうち "initial"だけはシステムで定義されていて， KEYBIND ファイルの中には必ず 1 つだけ STATE "initial"が存在します。

その他の STATE の名称はユーザが自由に決めることができます。ただし， STATE の名称と GOTO ACTION により指定される次の STATE は互いに対応がとれていなければなりません。

5.9 STATE の記述方法

例 5–3 に KEYBIND ファイルの中で STATE を記述する例を示します。 1 つの STATE はSTATE "string" =で始まり，END;で終了します。この間に書かれるキーと ACTION の関係は，その STATE 内でのみ有効です。

例 5–3 STATE の記述例

```
STATE "initial" =
    PF1 : START, GOTO "input_state";
END;

STATE "input_state" =
    PF1 : DONE, GOTO "initial";
    TYPING_KEYS : ECHO;
END;
```

この例を順に説明します。

1. 最初の STATE は "initial"です。このキー定義によると "initial"ではPF1キーだけが定義されています。
2. PF1キーを押すと， START が実行されて入力が始まる状態になります。また GOTO によって "input_state"に移ります。
3. "input_state"では TYPING_KEYS(表示されるキー) を押すと ECHO が実行されます。
4. 再びPF1キーを押すと DONE によって入力状態を終了して， GOTO "initial"によって "initial"に戻ります。

5.10 マクロの使い方

KEYBIND ファイルのキー名の記述には，マクロを使うことができます。マクロを使うことによってキーの割り当ての変更が容易になります。例 5–4 はマクロを使わない KEYBIND ファイルの記述例です。同じ記述をマクロを使って書くと例 5–5 のようになります。

マクロを使うと，あるキーに割り当てられていた機能を他のキーへ変更することが，マクロの変更だけでできるようになります。マクロ部分だけを別ファイルとすることもできますので，本体を記述したファイルを変更せずにキーの割り当てだけを変更できます。

また，マクロを使うと複数の STATE で定義されたキーの変更が容易になります。キー定義は STATE ごとに行わなければなりません。したがって，複数の STATE で同じ機能を実行しているようなときに，キー割り当てを変更しようとすると，マクロを使っていない場合は，各 STATE に書かれている該当するキーをすべて書き換えなければなりません。しかし，例 5–5 のようにマクロを使って記述すれば，マクロ部分をhiragana_henkan = CTRL_Lというように変更するだけで，すべての STATE におけるひらがな変換のキーはCtrl/Lに変えることができます。

コンパイラは，マクロが展開できなくなるまで，あるいは展開された文字列がキー名に一致するまでマクロ展開を繰り返します。ただし，マクロ展開は最大 10 回までに制限されています。マクロ展開が 10 回を超えると，コンパイラはエラーを出力して終了します。

例 5–4 マクロを使わないキー定義

(次ページに続く)

例 5–4 (続き) マクロを使わないキー定義

```
! キー定義本体
STATE "x" =
   .
   .
   .
    CTRL_L : HIRAGANA;
   .
   .
   .
END;

STATE "y" =
   .
   .
   .
    CTRL_L : HIRAGANA;
   .
   .
   .
END;
```

例 5–5 マクロを使ったキー定義

```
! マクロ部分
hiragana_henkan = CTRL_L;

! キー定義本体
STATE "x" =
   .
   .
   .
    hiragana_henkan : HIRAGANA;
   .
   .
   .
END;

STATE "y" =
   .
   .
   .
    hiragana_henkan : HIRAGANA;
   .
   .
   .
END;
```

6

KEYBIND コンパイラ

この章では KEYBIND コンパイラの使い方について説明します。

コンパイラのエラー・メッセージについての詳細は，付録 C を参照してください。

6.1 KEYBIND コンパイラとは？

KEYBIND コンパイラは，テキスト形式の KEYBIND ファイルをバイナリ形式の KEYBIND ファイルに変換します。

アプリケーションは，バイナリ形式の KEYBIND ファイルからキー定義に関する情報を得ます。したがって，テキスト形式で KEYBIND ファイルを作った後は，必ず KEYBIND コンパイラによって，バイナリ形式に変換しておかなければなりません。

6.2 KEYBIND コンパイラのコマンドと修飾子

KEYBIND コンパイラは KEYBIND コマンドで起動されます。 KEYBIND コマンドは以下のようなフォーマットになっています。

```
$ KEYBIND [/修飾子] ファイル名
```

以下に KEYBIND コマンドの例を示します。

```
$ KEYBIND /LIST MY_KEYBIND
```

この例では， MY_KEYBIND.DAT というテキスト形式の KEYBIND ファイルを MY_KEYBIND.IM$DAT というバイナリ形式の KEYBIND ファイルに変換します。また MY_KEYBIND.LIS という名前のリスト・ファイルが作られます。

6.2.1 修飾子

以下に KEYBIND コマンドの修飾子の説明をします。

- /[NO]LIST[=ファイル名]

  コンパイル・リストの出力ファイル名を指定します。/NOLIST のときは，コンパイル・リストを出力しません。省略時の値は/NOLIST です。

  ファイル名を省略した時は，ソース・ファイルと同じファイル名に.LIS というファイル・タイプを付けたものをファイル名として使います。

- /[NO]BINARY[=ファイル名]

  コンパイルの結果として作られる，バイナリ形式のファイルのファイル名を指定します。/NOBINARY のときは，バイナリ形式のファイルを作りません。省略時の値は/BINARY です。

  ファイル名を省略した時は，ソース・ファイルと同じファイル名に.IM$DAT というファイル・タイプを付けたものをファイル名として使います。

- /[NO]CHECK

  キー定義ファイルの中で指定される ACTION の，組合せの正しさを検証するかどうかを指定します。省略時の値は/CHECK です。

6.2.2 ファイル名

テキスト形式の KEYBIND ファイルのファイル名を指定します。ファイル・タイプを省略したときは，.DAT がファイル・タイプとして使われます。

6.3 新しい ACTION を含んだファイル

IMLIB 1.1 で新しくサポートされた RESTORE_ECHO ACTION および半角カナ変換 ACTION (HANKAKU_KANA, CLA_HANKAKU_KANA) を使用したファイルをコンパイルすると，出力されるバイナリ形式の KEYBIND ファイルのサポートレベルが 2 になります。アプリケーションは KEYBIND ファイルを読み込むときに，自分がサポートしているアクションに従ったレベルを指定します。このとき読み込もうとした KEYBIND ファイルのサポートレベルが，アプリケーションの指定したサポートレベルより高い場合は， KEYBIND ファイルの読み込みは失敗します。

PROFILE の DEC-JAPANESE.KEY.keybind で指定された KEYBIND ファイルの読み込みに失敗した場合は，最大 9 回まで再試行が行われます。再試行に使われるファイル名は上記の INDEX に_1，_2，などを付けた INDEX で指定できます。

7

KEYBIND ファイル作成例

この章では JVMS キーの KEYBIND ファイルを例にして， KEYBIND ファイルを作るときの注意点について書かれています。

JVMS キーの KEYBIND ファイルを例 7–1 に示します。説明のため実際に提供されているものと違っている部分があります。

例 7–1　JVMS キーの KEYBIND ファイル

```
!
! JVMS変換キー定義ファイル（システムテンプレート）
!
gold            = CTRL_G;1
kakutei         = CTRL_N;
kanji_henkan    = NULL, gold + CTRL_K;2
hiragana_henkan = CTRL_L;
katakana_henkan = CTRL_K;
zenkaku_henkan  = CTRL_F;
hankaku_henkan  = gold + CTRL_F;
kigou_henkan    = GS;
oomoji          = VOID;3
komoji          = VOID;
ji_bunsetsu     = CTRL_P;
zen_bunsetsu    = gold + CTRL_P;
tansyuku        = US;
sintyou         = gold + US;
zen_kouho       = gold + (NULL, CTRL_L);4
kaijo           = CTRL_N;
sakujo          = DEL;
hidari          = LEFT;
migi            = RIGHT;

%INCLUDE (IM$KEY_COMMON_BODY.DAT)5
```

1　CTRL_G を gold というマクロ名に定義します。

2　NULL または gold + CTRL_K を kanji_henkan というマクロ名に定義します。

3　VOID はどのキーにもマッチしないキーのシンボルです。 JVMS キーには oomoji や komoji に対応するキーが定義されていないので， VOID を使います。 oomoji, komoji は本体で使われているので，ここでこの行を省略することはできません。

4　zen_kouho というマクロ名は gold + NULL または gold + CTRL_L を意味します。

(次ページに続く)

例 7–1 (続き) JVMS キーの KEYBIND ファイル

5 ディレクティブ%INCLUDE によってキー定義ファイルの本体 (IM$KEY_COMMON_BODY.DAT) を読み込みます。

IM$KEY_COMMON_BODY.DAT の内容を例 7–2 に示します。

例 7–2 IM$KEY_COMMON_BODY.DAT の内容

```
!
! JVMS/EVEJ/LEIA変換キー定義ファイル本体（システムテンプレート）
!
!
! 初期状態
!
STATE "initial" =
 TYPING_KEYS : START, ECHO, GOTO "inputting";1
END;
!
! 入力状態
!
STATE "inputting" =
 kanji_henkan : CONVERT, GOTO "kk_converting";2
 hiragana_henkan : HIRAGANA, GOTO "converting";3
 katakana_henkan : KATAKANA, GOTO "converting";
 zenkaku_henkan : ZENKAKU, GOTO "converting";
 hankaku_henkan : HANKAKU, GOTO "converting";
 kigou_henkan : SYMBOL, GOTO "converting";
 oomoji  : UPPER, GOTO "converting";
 komoji  : LOWER, GOTO "converting";
 kakutei  : DONE, GOTO "initial";
 sakujo  : DELETE;
 hidari  : MOVE_LEFT;4
 migi  : MOVE_RIGHT;
 TYPING_KEYS : ECHO;
END;
!
! かな変換状態
!
STATE "converting" =
 kanji_henkan : CONVERT, GOTO "kk_converting";
 hiragana_henkan : HIRAGANA;
 katakana_henkan : KATAKANA;
 zenkaku_henkan : ZENKAKU;
 hankaku_henkan : HANKAKU;
 kigou_henkan : SYMBOL;
 oomoji  : UPPER;
 komoji  : LOWER;
 kaijo  : RESTORE_STRING, GOTO "inputting";5
```

(次ページに続く)

例 7–2 (続き) IM$KEY_COMMON_BODY.DAT の内容

```
 TYPING_KEYS : DONE, START, ECHO, GOTO "inputting";6
END;
!
! 漢字変換状態
!
STATE "kk_converting" =
 kanji_henkan : NEXT_CANDIDATE;
 hiragana_henkan : CLA_HIRAGANA;
 katakana_henkan : CLA_KATAKANA;
 zenkaku_henkan : ZENKAKU;
 hankaku_henkan : HANKAKU;
 kigou_henkan : SYMBOL;
 oomoji  : UPPER;
 komoji  : LOWER;
 ji_bunsetsu : NEXT_CLAUSE;
 zen_bunsetsu : PREV_CLAUSE;
 tansyuku : SHORTEN_CLAUSE;
 sintyou  : EXTEND_CLAUSE;
 zen_kouho : PREV_CANDIDATE;
 kaijo  : RESTORE_STRING, GOTO "inputting";
 TYPING_KEYS : DONE, START, ECHO, GOTO "inputting";
END;
```

1 STATE"initial"には TYPING_KEYS だけしか定義されていません。画面に表示されるキー以外が入力されるとアプリケーションが動作を決めます。画面に表示されるキーが入力されると入力を START し，入力されたキーを ECHO して，STATE は "inputting"に移ります。

2 マクロ kanji_henkan に定義されたキーが入力されると CONVERT によってかな漢字変換を行ない， STATE は "kk_converting"に移ります。

3 マクロ hiragana_henkan, katakana_henkan などに定義されたキーが入力されるとそれぞれの動作を行ない， STATE は "converting"に移ります。

4 ここで，定義されている MOVE_LEFT, MOVE_RIGHT は入力文字列の編集時のキー定義です。通常の編集時のカーソルの移動はアプリケーションが定義するので，ユーザはキー定義を行ないません。

5 RESTORE_STRING は，一度変換を始めた文字列をもう一度入力文字列に戻します。 STATE も "inputting"に戻すことで，文字列の編集関係のキーが使えるようになります。

6 STATE が "converting"の時に TYPING_KEYS が入力されると現在の状態が確定して新規に文字列が入力されます。
DONE によって変換は確定し，初期状態に戻りますので， DONE を省略することはできません。 DONE を省略すると START がエラーになります。

A

キー名

KEYBIND ファイルで使うことのできるキーの名前を表 A–1 に示します。 IMLIB は以下のキーボードを対象にしています。

- LK201-AJ
- LK401-AJ
- LK421-AJ
- LK401-JJ
- LK401-BJ
- LK421-JJ

表 A–1 キー名リスト

キー名	対応するキー
"X"	文字キー "X" (大文字と小文字は区別されます)
"カ "	文字キー "カ " (カナ入力状態)
"\ """	ダブル・クォーテーションマーク・キー
"\ \ "	バック・スラッシュ・キー
KP0	数字キーパッドの 0 キー
KP1	数字キーパッドの 1 キー
KP2	数字キーパッドの 2 キー
KP3	数字キーパッドの 3 キー
KP4	数字キーパッドの 4 キー
KP5	数字キーパッドの 5 キー
KP6	数字キーパッドの 6 キー
KP7	数字キーパッドの 7 キー
KP8	数字キーパッドの 8 キー
KP9	数字キーパッドの 9 キー
COMMA	数字キーパッドの , キー
MINUS	数字キーパッドの - キー
PERIOD	数字キーパッドの . キー

(次ページに続く)

表 A–1 (続き) キー名リスト

キー名	対応するキー
ENTER	数字キーパッドのEnterキー
PF1	数字キーパッドのPF1キー
PF2	数字キーパッドのPF2キー
PF3	数字キーパッドのPF3キー
PF4	数字キーパッドのPF4キー
F6	ファンクションキーのF6キー
F7	ファンクションキーのF7キー
F8	ファンクションキーのF8キー
F9	ファンクションキーのF9キー
F10	ファンクションキーのF10キー
F11	ファンクションキーのF11キー
F12	ファンクションキーのF12キー
F13	ファンクションキーのF13キー
F14	ファンクションキーのF14キー
HELP	ファンクションキーのHelpキー
DO	ファンクションキーのDoキー
F17	ファンクションキーのF17キー
F18	ファンクションキーのF18キー
F19	ファンクションキーのF19キー
F20	ファンクションキーのF20キー
NULL	Ctrl/**スペース**
CTRL_A	Ctrl/A
CTRL_B	Ctrl/B
CTRL_C	Ctrl/C
CTRL_D	Ctrl/D
CTRL_E	Ctrl/E
CTRL_F	Ctrl/F
CTRL_G	Ctrl/G
CTRL_H	Ctrl/H
TAB	Tabキー
CTRL_J	Ctrl/J
CTRL_K	Ctrl/K
CTRL_L	Ctrl/L
RETURN	Returnキー
CTRL_N	Ctrl/N
CTRL_O	Ctrl/O
CTRL_P	Ctrl/P

(次ページに続く)

表 A–1 (続き) キー名リスト

キー名	対応するキー
CTRL_Q	Ctrl/Q
CTRL_R	Ctrl/R
CTRL_S	Ctrl/S
CTRL_T	Ctrl/T
CTRL_U	Ctrl/U
CTRL_V	Ctrl/V
CTRL_W	Ctrl/W
CTRL_X	Ctrl/X
CTRL_Y	Ctrl/Y
CTRL_Z	Ctrl/Z
ESC	Ctrl/[
FS	Ctrl/\
GS	Ctrl/]
RS	Ctrl/‘
US	Ctrl//
LEFT	編集キーパッドの←キー
RIGHT	編集キーパッドの→キー
UP	編集キーパッドの↑キー
DOWN	編集キーパッドの↓キー
FIND	編集キーパッドのFindキー
INSERT_HERE	編集キーパッドのInsert Hereキー
REMOVE	編集キーパッドのRemoveキー
SELECT	編集キーパッドのSelectキー
PREV_SCREEN	編集キーパッドのPrev Screenキー
NEXT_SCREEN	編集キーパッドのNext Screenキー
DEL	メイン・キーパッドの<Xキー
SHFT_UP	SHIFT + 編集キーパッドの↑キー (前文節移動)
SHFT_DOWN	SHIFT + 編集キーパッドの↓キー (後文節移動)
SHFT_LEFT	SHIFT + 編集キーパッドの←キー (文節縮小)
SHFT_RIGHT	SHIFT + 編集キーパッドの→キー (文節拡大)
CTRL_UP	CTRL + 編集キーパッドの↑キー
CTRL_DOWN	CTRL + 編集キーパッドの↓キー
CTRL_LEFT	CTRL + 編集キーパッドの←キー
CTRL_RIGHT	CTRL + 編集キーパッドの→キー
CTRL_SHFT_ALT_UP	CTRL + SHIFT + ALT + 編集キーパッドの↑キー
CTRL_SHFT_ALT_DOWN	CTRL + SHIFT + ALT + 編集キーパッドの↓キー
CTRL_SHFT_ALT_LEFT	CTRL + SHIFT + ALT + 編集キーパッドの←キー

(次ページに続く)

表 A–1 (続き) キー名リスト

キー名	対応するキー
CTRL_SHFT_ALT_RIGHT	CTRL + SHIFT + ALT + 編集キーパッドの→キー
CTRL_SHFT_UP	CTRL + SHIFT + 編集キーパッドの↑キー
CTRL_SHFT_DOWN	CTRL + SHIFT + 編集キーパッドの↓キー
CTRL_SHFT_LEFT	CTRL + SHIFT + 編集キーパッドの←キー
CTRL_SHFT_RIGHT	CTRL + SHIFT + 編集キーパッドの→キー
CTRL_ALT_UP	CTRL + ALT キーパッドの↑キー
CTRL_ALT_DOWN	CTRL + ALT キーパッドの↓キー
CTRL_ALT_LEFT	CTRL + ALT + 編集キーパッドの←キー
CTRL_ALT_RIGHT	CTRL + ALT + 編集キーパッドの→キー
ALT_UP	ALT + 編集キーパッドの↑キー
ALT_DOWN	ALT + 編集キーパッドの↓キー
ALT_LEFT	ALT + 編集キーパッドの←キー
ALT_RIGHT	ALT + 編集キーパッドの→キー
ALT_SHFT_UP	SHIFT + ALT + 編集キーパッドの↑キー
ALT_SHFT_DOWN	SHIFT + ALT + 編集キーパッドの↓キー
ALT_SHFT_LEFT	SHIFT + ALT + 編集キーパッドの←キー
ALT_SHFT_RIGHT	SHIFT + ALT + 編集キーパッドの→キー
JFK_HENKAN	変換キー
JFK_CTRL_HENKAN	CTRL + 変換キー
JFK_ALT_HENKAN	ALT + 変換キー
JFK_SHFT_HENKAN	SHIFT + 変換キー (前候補)
JFK_CTRL_SHFT_HENKAN	CTRL + SHIFT + 変換キー
JFK_CTRL_ALT_HENKAN	CTRL + ALT + 変換キー
JFK_CTRL_SHFT_ALT_HENKAN	CTRL + SHIFT + ALT + 変換キー
JFK_MUHENKAN	無変換キー
JFK_SHFT_MUHENKAN	SHIFT + 無変換キー (入力状態へ戻る)
JFK_CTRL_MUHENKAN	CTRL + 無変換キー
JFK_ALT_MUHENKAN	ALT + 無変換キー (記号変換)
JFK_CTRL_SHFT_MUHENKAN	CTRL + SHIFT + 無変換キー
JFK_CTRL_ALT_MUHENKAN	CTRL + ALT + 無変換キー
JFK_CTRL_SHFT_ALT_MUHENKAN	CTRL + SHIFT + ALT + 無変換キー
JFK_HIRAGANA	ひらがなキー
JFK_SHFT_HIRAGANA	SHIFT + ひらがなキー (カタカナ変換)
JFK_CTRL_HIRAGANA	CTRL + ひらがな変換キー (半角カタカナ変換)

(次ページに続く)

表 A–1 (続き) キー名リスト

キー名	対応するキー
JFK_ALT_HIRAGANA	ALT + ひらがな変換キー (全角英数字変換)
JFK_CTRL_SHFT_ HIRAGANA	CTRL + SHIFT + ひらがな変換キー (半角カタカナ変換)
JFK_CTRL_ALT_ HIRAGANA	CTRL + ALT + ひらがな変換キー
JFK_SHFT_ALT_ HIRAGANA	SHIFT + ALT + ひらがな変換キー
JFK_CTRL_SHFT_ALT_ HIRAGANA	CTRL + SHIFT + ALT + ひらがな変換キー
OTHERS	すべてのキー
TYPING_KEYS	すべての文字キー
VOID	どのキーにもマッチしない

B

KEYBIND ファイルのシンタックス

図 B–1 に KEYBIND ファイルのシンタックスを示します。このシンタックスの記述規則は以下のとおりです。

- 大文字はキーワードを示します (終端記号)。
- アイデンティファイアは大カッコ[ ]を使って示します (終端記号)。アイデンティファイアには，キー名[keyname]， ACTION 名[action]，マクロ名[macro_identifier]， STATE 名[state_string]があります。
- 非終端記号は小文字で示します。

KEYBIND ファイルで使われるディレクティブには以下のものがあります。

- %include (ファイル名)
  指定したファイルを読み込む。

図 B–1 KEYBIND ファイルのシンタックス

```
## Syntax ##

    keybind ::= macro_def_list states | states

## Syntax of keydef_header ##

    macro_def_list ::= macro_def | macro_def macro_def_list
    macro_def      ::= macro_name = key_choice ;
    macro_name     ::= [macro_identifier]

## Syntax of keydef_body ##

    states         ::= a_state | a_state states
    a_state        ::= STATE state_name = key_def_list END ;
    state_name     ::= [state_string]
    key_def_list   ::= key_def | key_def key_def_list
    key_def        ::= key_choice : actions ;

    key_choice     ::= key_sequence | key_sequence , key_choice
    key_sequence   ::= keys | keys + key_sequence
    keys           ::= ( key_choice ) | a_key

    a_key          ::= key | macro_name
                           #(defined macro_name in keydef_header)
    key            ::= [keyname]

    actions        ::= an_action | an_action , actions | goto_action
    an_action      ::= [action]
    goto_action    ::= GOTO state_name
```

C

KEYBIND コンパイラのエラー

CANNOTOPN, Cannot open
重大度: Error
説明: ファイルを開くことができません。
ユーザの処置: ファイルのプロテクションを，ファイルを開くことができるように変更してください。

FILNOTFND, File not found
重大度: Error
説明: 指定された KEYBIND ファイルがありません。
ユーザの処置: KEYBIND ファイルを正しく指定してください。

ILLCHAR, Character *char* is illegal
重大度: Error
説明: 正しくない文字が含まれています。
ユーザの処置: 正しくない文字を取り除いてください。

ILLKEY, Keyname *string* is illegal
重大度: Error
説明: 正しくないキー名が指定されました。
ユーザの処置: 正しいキー名を使ってください。

ILLDIRECTIVE, *string* is not a directive
重大度: Error
説明: ディレクティブでない文字列がディレクティブとして指定されています。
ユーザの処置: 正しいディレクティブを指定してください。

ILLKEYWORD, *string* is not a keyword
重大度: Error
説明: 正しくないキーワードが指定されました。
ユーザの処置: キー名または ACTION 名が間違っていますので，正しく変更してください。

INSVIRMEM, Insufficient virtual memory
重大度: Error
説明: 仮想メモリが不足しています。
ユーザの処置: システム管理者に相談してください。

INVACTION, Action *string* is invalid in this state
重大度: Warning
説明: STATE に対応しない ACTION が書かれています。
ユーザの処置: STATE と ACTION の対応を正しくしてください。

MACROSYNTAX, Macro syntax error
重大度: Error
説明: マクロ構文が正しくありません。
ユーザの処置: マクロ構文を正しく書き直してください。

NEXTLEVEL, Exceed maximum nest level
重大度: Error
説明: INCLUDE のネスティングが制限値を超えました。
ユーザの処置: INCLUDE のネスティングを少なくするるように, KEYBIND ファイルを書き直してください。

NOINITIAL, No initial state
重大度: Error
説明: STATE"initial"がありません。
ユーザの処置: KEYBIND ファイルには必ず STATE"initial"が必要です。STATE"initial"を使って KEYBIND ファイルを書き直してください。

NOKEYDEF, State *string* has no key-action definition.
重大度: Error
説明: 中味のない STATE は許されていません。
ユーザの処置: STATE に 1 つ以上のキー定義文を書いてください。

NOPAREN, Include file is not in parentheses
重大度: Error
説明: ディレクティブ INCLUDE ファイル指定文字列にカッコがついていません。
ユーザの処置: INCLUDE のファイル指定にはカッコをつけてください。

NOSTATE, No state statement
重大度: Error
説明: STATE 文がありません。
ユーザの処置: STATE 文を使ってください。

READERR, File read error
重大度: Error
説明: ファイルを読むことができません。
ユーザの処置: 正しい KEYBIND ファイルかどうかを調べてください。

SYNTAXERR, Syntax error
重大度: Error
説明: 構文上の誤りがあります。
ユーザの処置: 構文上の誤りを正しく書き直してください。

WRITEERR, File write error

重大度: Error

説明: ファイル書き込み中にエラーが発生しました。

ユーザの処置: ユーザに割り当てられたディスク・クォータ，およびディスクのフリー・スペースを調べてください。

索引

S

T

U

V

Z

キ

ニ

ヒ

ホ

マ

日本語 OpenVMS
ユーザ・キー定義 利用者の手引き

1999 年 4 月　発行

日本ヒューレット・パッカード株式会社

〒 102-0076 東京都千代田区五番町 7 番地

電話　(03)3512-5700 (大代表)

AA-PU8UE-TE