

日本語 OpenVMS

IMLIB/OpenVMS ライブラリ・ リファレンス・マニュアル

AA-PU8TE-TE

1999 年 4 月

本書は、ユーザが漢字変換のキー定義をできるようにするライブラリである IMLIB についての解説書です。

改訂/更新情報:

日本語 OpenVMS V6.2 『IMLIB
/OpenVMS ライブラリ・リファレン
ス・マニュアル』の改訂版です。

ソフトウェア・バージョン:

日本語 OpenVMS Alpha V7.2

日本語 OpenVMS VAX V7.2

コンパックコンピュータ株式会社

1999 年 4 月

本書の著作権はコンパックコンピュータ株式会社が保有しており、本書中の解説および図、表はコンパックの文書による許可なしに、その全体または一部を、いかなる場合にも再版あるいは複製することを禁じます。

また、本書に記載されている事項は、予告なく変更されることがありますので、あらかじめご承知おきください。万一、本書の記述に誤りがあった場合でも、コンパックは一切その責任を負いかねます。

本書で解説するソフトウェア (対象ソフトウェア) は、所定のライセンス契約が締結された場合に限り、その使用あるいは複製が許可されます。

© Compaq Computer Corporation 1999.

All Rights Reserved.

Printed in Singapore.

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

目次

まえがき	vii
1 IMLIB の機能	
1.1 ユーザによるかな漢字変換のキー定義	1-1
1.2 ユーザによるかな漢字変換の付加情報の指定	1-2
2 アプリケーションの作成	
2.1 IMLIB の初期化	2-1
2.2 IMLIB の終了	2-2
2.3 キー入力処理	2-2
2.3.1 アプリケーションとの関係	2-3
2.3.2 かな漢字変換入力インターフェイスの範囲	2-4
2.3.3 アプリケーションが用意するもの	2-5
2.3.4 処理の流れ	2-5
2.3.5 キー入力モジュール	2-7
2.3.5.1 OpenVMS	2-7
2.3.5.2 DECwindows	2-7
2.3.6 ローマ字かな変換	2-7
2.3.7 文字列バッファ	2-8
2.3.8 入力文字列編集	2-8
2.3.8.1 入力文字列編集の条件	2-9
2.3.8.2 入力文字列編集の実現	2-9
2.3.8.3 例外処理	2-9
2.3.8.4 オーバーストライク・モード	2-10
2.3.9 自動ローマ字かな変換における入力文字列編集	2-12

3	ACTION の実行	
3.1	ACTION の定義	3-1
4	アプリケーションの作成方法	
4.1	アプリケーションが用意する資源	4-1
4.2	ACTION による状態の遷移	4-1
4.3	ACTION の動作	4-4
5	IMLIB ライブラリ・ルーチン	
5.1	IMLIB ルーチンとは?	5-1
5.1.1	プログラミング言語	5-1
5.2	PROFILE に関連するルーチン	5-1
5.2.1	PROFILE のオープン	5-1
5.2.2	PROFILE のクローズ	5-2
5.2.3	PROFILE データの検索	5-2
5.2.4	PROFILE データの変更	5-2
5.2.5	PROFILE データの書き込み	5-2
5.3	KEYBIND に関連するルーチン	5-3
5.3.1	KEYBIND 情報の設定	5-3
5.3.2	ACTION を得るルーチン (C バインディング)	5-3
5.3.3	ACTION を得るルーチン (VMS バインディング)	5-3
5.3.4	STATE を初期化するルーチン	5-3
5.3.5	直前の STATE に戻すルーチン	5-4
5.4	KEYCODE に関連するルーチン	5-4
5.4.1	キー名文字列から KEYCODE への変換	5-4
5.4.2	KEYCODE に対応する文字を返すルーチン	5-4
5.4.3	半角カナの KEYCODE に対応する文字を返すルーチン	5-4
5.4.4	キーが発生するコードから KEYCODE への変換	5-5
5.4.5	KEYSYM から KEYCODE への変換	5-5
5.4.6	半角カナから KEYCODE への変換	5-5
5.5	IMLIB ルーチン	5-5
	OPEN PROFILE	5-6
	WRITE PROFILE	5-9
	CLOSE PROFILE	5-12
	SET PROFILE DATA	5-14
	GET PROFILE DATA	5-17

SET KEYBIND	5-20
GET KEY ACTION	5-24
SET KEY	5-26
GET ACTION	5-29
RECOVER KEY STATE	5-31
GET KEYCODE	5-33
GET CHARACTER	5-36
MC GET CHARACTER	5-39
INIT KEY STATE	5-43
ENCODE KEY	5-45
MC ENCODE KEY	5-48
KEYSYM TO KEYCODE	5-51
 6 プログラムの開発	
6.1 ヘッダ・ファイル	6-1
6.2 コンパイル・リンクの方法	6-2
6.2.1 C で書かれたアプリケーション・プログラム	6-2
6.2.2 FORTRAN で書かれたアプリケーション・プログラム	6-3
 A IMLIB がサポートする KEYSYM	
 B 定義済のシンボル	
B.1 エラー・シンボル	B-1
B.2 ACTION を示すシンボル	B-2
B.3 KEY を示すシンボル	B-4
B.4 Key Modifier を示すシンボル	B-9

索引



2-1	IMLIB とアプリケーションの関係	2-3
2-2	処理の流れ	2-6
2-3	オーバーストライク編集の例その 1: 固定法	2-11
2-4	オーバーストライク編集の例その 2: 浮動法	2-11
4-1	かな漢字変換の内部状態と状態間の遷移	4-2

表

4-1	状態間の遷移	4-13
A-1	サポートされる KEYSYM	A-1
B-1	エラー・シンボル	B-1
B-2	ACTION を示すシンボル	B-3
B-3	KEY を示すシンボル	B-4
B-4	Key Modifier を示すシンボル	B-9

まえがき

本書の目的

本書は、ユーザ・キー定義ライブラリ、IMLIB/KEYBIND を使用したアプリケーションの開発者のためのマニュアルです。IMLIB/KEYBIND は、ユーザが漢字変換のキー定義をできるようにするライブラリです。IMLIB を使用してアプリケーションを開発すると、ユーザによるキー定義で各種のプログラムを動作することができます。ユーザによるキー定義の方法については、『ユーザ・キー定義 利用者の手引き』を参照してください。

本書の構成

本書は 6 つの章と 2 つの付録から構成されています。

- | | |
|-------|---|
| 第 1 章 | IMLIB の機能について説明します。 |
| 第 2 章 | アプリケーションを作成するための具体的な方法について説明します。 |
| 第 3 章 | ACTION を実行するモジュールの作成方法について説明します。 |
| 第 4 章 | アプリケーションの作成について説明します。 |
| 第 5 章 | IMLIB ルーチンの一覧です。 |
| 第 6 章 | IMLIB を使ったプログラムの開発について説明します。 |
| 付録 A | KEYSYM TO KEYCODE がサポートする KEYSYM の一覧です。 |
| 付録 B | IMLIB で定義済みのシンボル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ています。

表記法	意味
<code>Return</code>	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、 <code>Return</code> は Return キーを押すことを示します。
<code>Ctrl/x</code>	<code>Ctrl/x</code> 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、 <code>Ctrl/c</code> は Ctrl キーと c 文字キーを同時に押します。
[]	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。

IMLIB の機能

この章は、ユーザ・キー定義ライブラリ IMLIB を使うことで実現できる機能について書かれています。

1.1 ユーザによるかな漢字変換のキー定義

IMLIB は、かな漢字変換のキーをユーザが定義することができる機能を提供します。IMLIB が定める規則に従ってアプリケーションを作成すると、ユーザは KEYBIND ファイルを書き換えることで、アプリケーションの中のかな漢字変換入力を自分の好みのキー定義で行うことができるようになります。KEYBIND ファイルについての詳細は、『ユーザ・キー定義 利用者の手引き』を参照してください。

IMLIB を使ったユーザ・キー定義をサポートすることには、次のような利点があります。

- ユーザが慣れたキー定義で、かな漢字変換を行うことができる
- IMLIB をサポートするすべてのアプリケーションで、同じキー定義を使うことができる
- IMLIB が定める規則に従えば、個々のキーの細かい動作を意識せずに、アプリケーションを作ることができる

アプリケーションを作るときに必要な規則は、本書に書かれています。ユーザによるかな漢字変換キー定義の方法については、『ユーザ・キー定義 利用者の手引き』を参照してください。

1.2 ユーザによるかな漢字変換の付加情報の指定

IMLIB は、KEYBIND ファイルによるかな漢字変換キー定義に加えて、かな漢字変換に関連する環境の指定を行う機能を、提供しています。このような付加情報の指定は、PROFILE ファイルで行います。PROFILE ファイルについての詳細は、『ユーザ・キー定義 利用者の手引き』を参照してください。

ユーザの指示どおりにアプリケーションが動作するためには、アプリケーションは PROFILE ファイルに書かれている情報を検索し、その指示に従った動作をするように作成されなければなりません。

PROFILE に書かれた情報を獲得するには GET PROFILE DATA を使います。

PROFILE には、複数のアプリケーションで共通に使われる情報と、個々のアプリケーションで使われる情報の両方を書くことができます。共通に使われる情報の INDEX は、『ユーザ・キー定義 利用者の手引き』を参照してください。アプリケーションの動作環境によっては、共通の INDEX が意味をなさない場合もありますので、アプリケーションには共通の INDEX をすべてサポートする義務はありません。しかし、できるだけ多くの共通の INDEX をサポートすることが望まれます。

アプリケーションごとの INDEX はアプリケーションが独自に決めることができます。INDEX は次のような形式に従ってください。

```
facility-name.any-string
```

たとえば、XYZ というプロダクトが、エラーのときにベルを鳴らすかどうかを PROFILE の中で指定するようなときには、INDEX は

```
XYZ.ERROR.bell
```

のようになります。その VALUE には、文字列 "enable" または "disable" をとるということをアプリケーションが独自に決めることができます。この場合は、PROFILE の中に

```
XYZ.ERROR.bell : enable
```

と書けば、そのアプリケーションでエラーが発生したときにはベルが鳴ることになります。

PROFILE の INDEX は大文字と小文字を区別しません。したがって XYZ.ERROR.bell と xyz.error.bell は同じ INDEX とみなされます。これに対して GET PROFILE DATA が返す VALUE は、PROFILE に書かれたとおりですので、大文字と小文字を区別して扱います。

アプリケーションの作成

この章には、アプリケーションを作成するための具体的な方法が書かれています。

IMLIB の機能は、アプリケーションの中で使われる状況によって、次のように分類できます。

- IMLIB の初期化
- IMLIB の終了
- キー入力処理

以下の節では、分類されたそれぞれの機能について説明します。

2.1 IMLIB の初期化

IMLIB の初期化は以下の手順に従って行われます。

1. PROFILE のオープン

OPEN PROFILE によって PROFILE をオープンします。OPEN PROFILE が返す UNIT ID は、これ以降 IMLIB のライブラリを呼び出すときに使われます。複数のかな漢字変換コンテキストを使うアプリケーションは、必要なコンテキストの数だけ OPEN PROFILE を呼びます。

2. PROFILE 情報の読み込み

GET PROFILE DATA によって、PROFILE に書かれた情報を読み込みます。PROFILE の情報に、キー定義以外のかな漢字変換の環境を定義するものが含まれています。アプリケーションは、できる限り PROFILE に書かれた情報に従って動作することが望まれます。

3. KEYBIND の設定

SET KEYBIND によって、バイナリ形式の KEYBIND ファイルを読み込みます。

以上で IMLIB の初期設定は完了します。次はキー入力処理に移ります。

2.2 IMLIB の終了

IMLIB の使用を終了するときに使われる機能について、以下に示します。

- PROFILE のクローズ

CLOSE PROFILE は、PROFILE ファイルをクローズして IMLIB の使用を終了します。CLOSE PROFILE は、IMLIB を使った機能をすべて終了した後に呼ばれます。

- PROFILE の変更

SET PROFILE DATA によって、PROFILE のデータを変更することができます。SET PROFILE DATA は、メモリ上のデータを変更します。変更された内容をファイルに書き込むには WRITE PROFILE を使います。

- PROFILE の書き込み

WRITE PROFILE は、SET PROFILE DATA で変更された PROFILE の内容をファイルに書き込むときに使われます。

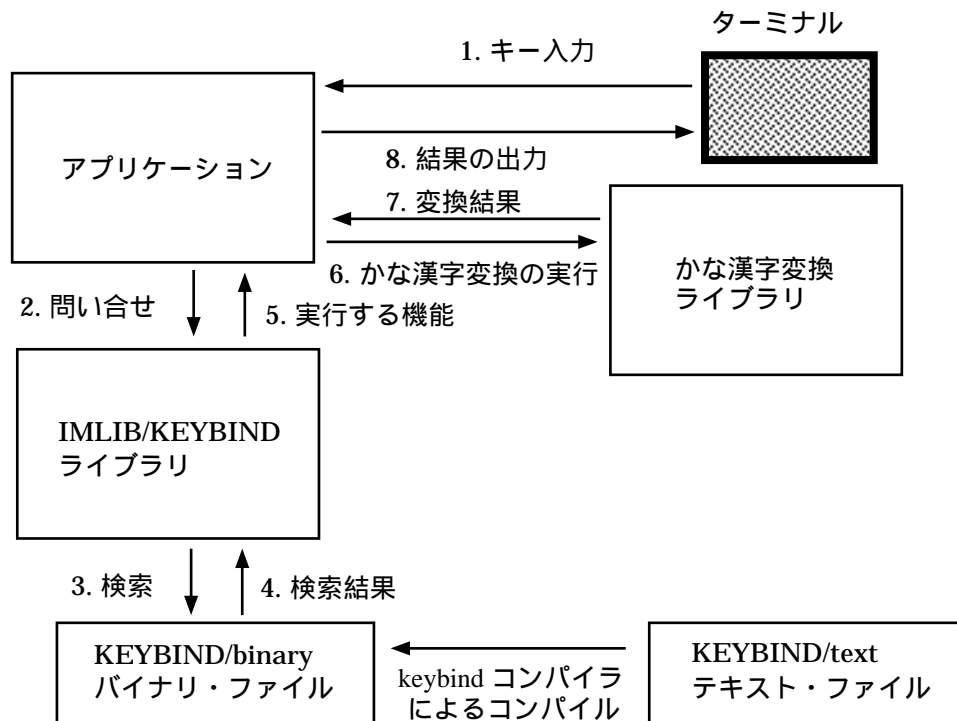
2.3 キー入力処理

キー入力処理は IMLIB を使う上での核となる部分です。この節は、キー入力処理を行うアプリケーションの作成方法について書かれています。

2.3.1 アプリケーションとの関係

アプリケーションが、IMLIB を使ってキー入力を処理する概念を示す図を、
図 2-1 に示します。

図 2-1 IMLIB とアプリケーションの関係



1. ターミナルからキーが入力される。
2. アプリケーションは、入力されたキーに何が定義されているのかを IMLIB に問い合わせる。

3. IMLIB は、問い合わせのあったキーを KEYBIND ファイルの情報に従って検索する (実際には KEYBIND ファイルの内容は、最初にメモリ上に読み込まれているのでメモリ中を検索する)。
4. 検索によって実行する機能 (ACTION) が得られる。
5. KEYBIND ファイルから得た情報を、アプリケーションに返す。
6. アプリケーションは、IMLIB から返された情報がかな漢字変換に関するものであった場合、かな漢字変換ライブラリを呼び出してかな漢字変換を実行する。
7. かな漢字変換の結果がアプリケーションに戻る。
8. 実行結果をターミナルに出力する。

2.3.2 かな漢字変換入力インターフェイスの範囲

かな漢字変換入力を行うには、いくつかのレベルがあります。IMLIB は、パーソナル・コンピュータ用に市販されているワードプロセッサ・プログラムの 1 つである「一太郎」¹と同程度のかな漢字変換入力インターフェイスの実現を前提に、設計されています。

市販されているパーソナル・コンピュータ上のワードプロセッサ・プログラムは一般に、日本語 OpenVMS 独自のかな漢字変換インターフェイス (JEDI, EVEJ などのインターフェイス) よりも、複雑なインターフェイスを持っています。最も典型的な例はスペース・キーの機能です。スペース・キーは、キーが押されたときの状態によって通常のスペース・キーとして動作したり、かな漢字変換キーとして動作したりします。また、自動ローマ字かな変換の機能なども、ワードプロセッサのインターフェイスの実現を複雑にしているものの 1 つです。

¹ 一太郎は株式会社ジャスト・システムの登録商標です。

2.3.3 アプリケーションが用意するもの

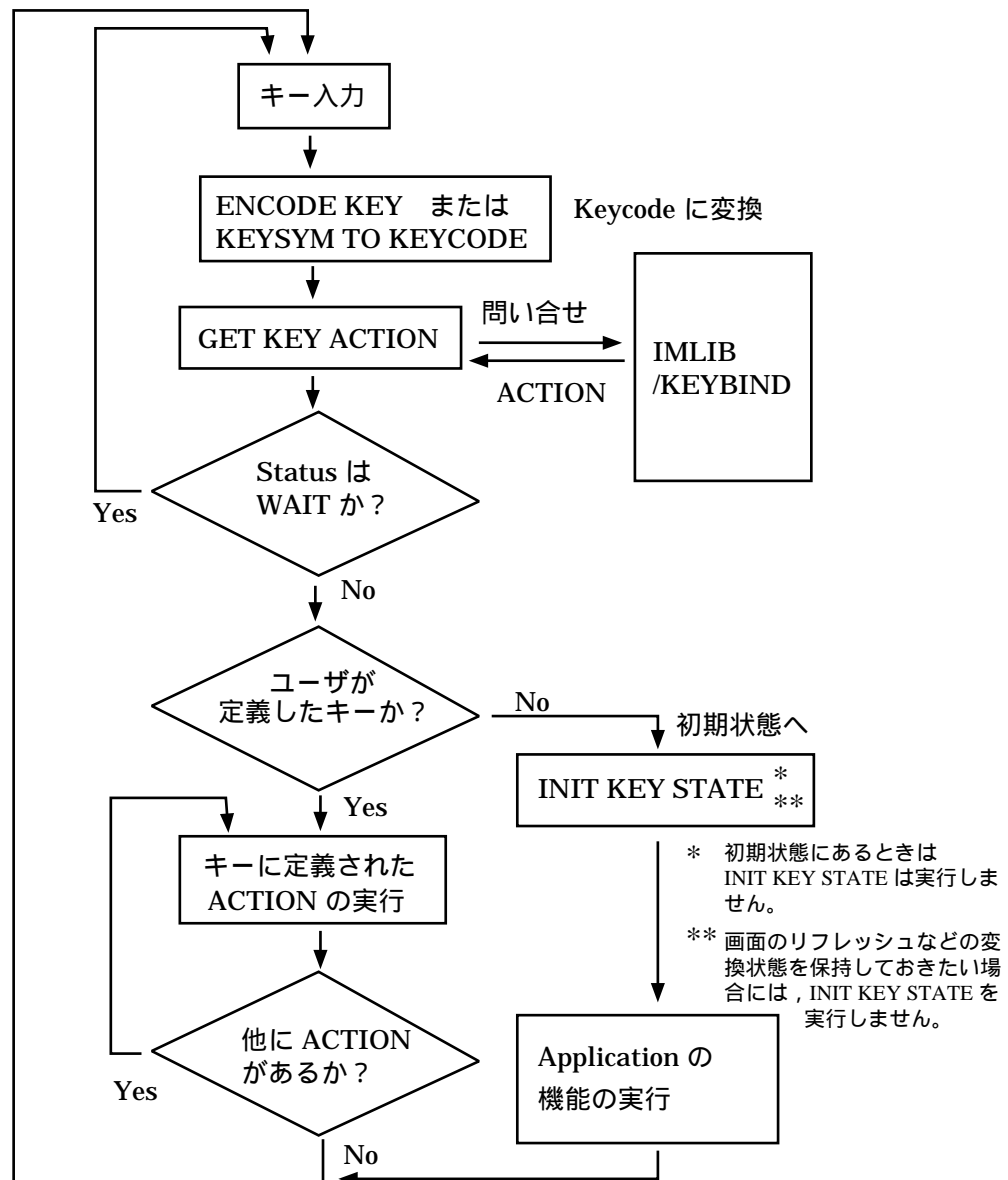
アプリケーションは、KEYBIND によって提供されるキー定義の情報や PROFILE によって指定される付加情報以外のものを、すべて用意しなければなりません。たとえば、以下に示すものはアプリケーションが用意するものです。

- ユーザのキー入力を受け取るルーチン
- ローマ字かな変換を行うルーチン
- 変換中の文字列を処理するための文字列バッファ
- IMLIB から返される ACTION を実行するルーチン
- かな漢字変換入力に関係しない、通常の編集キーを処理するルーチン
- かな漢字変換ルーチンとのインターフェイス
- 画面制御を行うルーチン

2.3.4 処理の流れ

IMLIB を使ったプログラムの中で実行される処理の流れを、図 2-2 に示します。これは IMLIB の C バインディング・インターフェイスで記述した例です。VMS バインディングの場合は、GET KEY ACTION が 2 つの呼び出し (SET KEY と GET ACTION) に分れます。これによって処理の流れも変わります。

図 2-2 処理の流れ



2.3.5 キー入力モジュール

キー入力モジュールは、ユーザのキー入力を受け取ります。IMLIB を使うためには、入力は 1 つのキーごとに行わなければなりません。キー入力のメカニズムは、オペレーティング・システムや使用しているユーザ環境によって異なります。IMLIB は、OpenVMS オペレーティング・システム、ULTRIX オペレーティング・システム、Digital UNIX オペレーティング・システム、およびそれらの上に構築されている DECwindows のインターフェイスを考慮して作られています。

2.3.5.1 OpenVMS

OpenVMS でのキー入力には、QIO が使われます。QIO は入力のターミネータとしてエスケープ・シーケンスを指定することができますので、キーボード上のファンクション・キー・数字キーパッドや編集キーを、1 回の QIO で読み込むことができます。読み込まれたキーは、エスケープ・シーケンス、文字キーともに ENCODE KEY を呼び出すことで IMLIB の KEYCODE に変換することができます。

2.3.5.2 DECwindows

DECwindows では、XLookupString によってキー入力を読み込みます。XLookupString は、X Window System で定義されている KEYSYM の値を返しますので、アプリケーションは、IMLIB が提供するルーチン KEYSYM TO KEYCODE を使って、KEYCODE の値を得ます。

2.3.6 ローマ字かな変換

ワードプロセッサなどでよく見られる、キー入力時の自動ローマ字かな変換のサポートは、アプリケーションが行う必要があります。自動ローマ字かな変換をサポートするかどうかは、アプリケーション開発者の任意です。

2.3.7 文字列バッファ

ワードプロセッサで見られるのと同様なユーザ・インターフェイスを提供するには、文字列バッファが3個必要です。それぞれのバッファの役割を以下に示します。

1. キー入力バッファ

ユーザが入力した文字キーを、そのまま保持しておくバッファです。このバッファの内容は、変換対象や入力文字列編集の対象となる文字列に対応します。変換が確定すると、このバッファの内容は消去されます。

2. キー・エコー・バッファ

自動ローマ字かな変換を行った結果を保持するバッファです。自動ローマ字かな変換を行わない場合や、かなキー入力を行った場合には、キー入力バッファの内容と同じになります。このバッファの内容は、変換対象や入力文字列編集の対象となります。変換が確定するとこのバッファの内容は消去されます。

3. 表示バッファ

スクリーン上と同じイメージを保持します。かな漢字変換の結果はこのバッファに反映されます。

2.3.8 入力文字列編集

ユーザにとって使いやすい入力インターフェイスを提供するためには、入力文字列を編集できる機能を提供することは不可欠です。入力文字列編集とは、たとえばユーザが"日本語"と入力しようとしたのに、間違えて"nijongo"というようなローマ字を入力してしまったときに、カーソルを"j"まで移動して、"h"と置き換えるような作業のことを言います。正しい文字と置きかわった時点で「変換キー」を押すと、かな漢字変換を行うことができるということになります。

入力文字列編集において問題となる点のうち、自動ローマ字かな変換を行っているときに限定される点については、第2.3.9項で説明します。この節では、入力文字列編集について一般的な説明を行います。

2.3.8.1 入力文字列編集の条件

入力文字列編集を行うのは、内部状態が「入力状態」にあるときです。次のような条件の場合に内部状態が「入力状態」になります。

- 「初期状態」において ACTION START が指示されたとき
- 「かな変換状態」または「漢字変換状態」において、ACTION RESTORE_STRING が指示されたとき

2.3.8.2 入力文字列編集の実現

入力文字列編集に関連する ACTION を以下に示します。

- ECHO
- MOVE_LEFT
- MOVE_RIGHT
- HEAD
- TAIL
- DELETE

入力文字列編集時には、上記の ACTION によって変換領域は解除しません。バッファの内容は ECHO と DELETE を除いて変化しません。ECHO のときは、カーソル位置に入力されたキーに対応する文字を挿入し、DELETE のときは、カーソルの左側の文字を削除します。文字の挿入と削除は、3 つのバッファすべてに対して適用されます。

2.3.8.3 例外処理

入力文字列編集における例外条件を、以下に示します。

- 編集領域の左端の文字にカーソルがあるときに MOVE_LEFT が指定された
- 編集領域の右端の文字にカーソルがあるときに MOVE_RIGHT が指定された
- 編集領域の左端の文字にカーソルがあるときに DELETE が指定された

上記のような場合にはアプリケーションは、PROFILE の INDEX "DEC-JAPANESE.OUTRANGE.cursorPosition" の値によって動作を決定します。この値が "none" のときは、何も実行しません。カーソルはその位置に留まります。この値が "done" のときは、現在の変換を終了して、アプリケーションがそのキーに対して定義した動作を実行します。

2.3.8.4 オーバーストライク・モード

文字セル端末用のアプリケーションの多くは、オーバーストライク・モード (上書き) をサポートしています。ここには、オーバーストライク・モードの扱いについて IMLIB が推奨する方法について記述します。

IMLIB は、通常の編集状態がオーバーストライク・モードであっても入力文字列編集時は、インサート・モード (挿入) の動きをする方法をお勧めします。かな漢字変換が終了して確定した文字列は、アプリケーションのオーバーストライク/インサート・モードの設定に従って、アプリケーションのバッファ中に上書き、または挿入されます。次にオーバーストライク・モードにおける入力文字列編集について、具体的な例をあげて説明します。IMLIB は 2 つの実現方法を併記します。どちらの方法を選択するかはアプリケーションの責任です。カーソル位置は "_" で示します。

図 2-3 オーバーストライク編集の例その 1: 固定法

入力前	日本語学習の現状
"kyouoku" と入力	日本語kyouoku_状
カーソル左移動2回	日本語kyou <u>o</u> ku 状
削除キー	日本語kyou <u>o</u> ku現状
"i" と入力	日本語kyou <u>i</u> ku 状
変換キー	日本語教育 <u>o</u> の現状
確定	日本語教育の <u>o</u> 現状

図 2-4 オーバーストライク編集の例その 2: 浮動法

入力前	日本語学習の現状
"kyouoku" と入力	日本語kyouoku学習の現状
カーソル左移動2回	日本語kyouoku学習の現状
削除キー	日本語kyou <u>o</u> ku学習の現状
"i" と入力	日本語kyou <u>i</u> ku学習の現状
変換キー	日本語教育 <u>o</u> 学習の現状
確定	日本語教育の <u>o</u> 現状

「固定法」は、スクリーン入力型のアプリケーション、たとえばエディタなどに適した実現方法です。すでにある文字列の位置が移動しないという利点があります。しかし、入力文字列が長すぎると元の文字列が見えなくなります。

これに対して「浮動法」は、行入力型のアプリケーションに適した実現方法といえます。元の文字列は編集中でも見ることができます。しかし、最終的には元の位置に戻る文字列も、編集では移動してしまうことになります。また、確定したときに上

書きされるので、上書きされた文字列は確定したときに突然消えることとなります。

2.3.9 自動ローマ字かな変換における入力文字列編集

ここでは、自動ローマ字かな変換を行う際に問題となることについて記述します。

自動ローマ字かな変換状態での入力文字列編集において、最も複雑な点は、「キー入力バッファ」の内容と「キー・エコー・バッファ」の内容が異なることです。ユーザから見えるのは「キー・エコー・バッファ」だけであるにもかかわらず、編集は両方のバッファに対して行わなければなりません。

ローマ字とひらがなは1対1に対応しないため、ひらがなが表示されている状態で、ローマ字の1文字だけを操作するということとはできません。編集時にカーソルはひらがなの上を1文字ずつ移動するので、「キー入力バッファ」には、どの位置にひらがなの境界があるのかを示す情報が必要になります。

たとえば、「キー入力バッファ」に"nihongo"という文字列が入っているとすると、「キー・エコー・バッファ」は"にほんご"となっています。このとき「キー入力バッファ」の中の"ni", "ho", "n", "go"がそれぞれ、ひらがなに対応しているという情報を持っておかなければなりません。「キー・エコー・バッファ」中で"ほ"が消去されると、「キー入力バッファ」においては"ho"が消去されることになります。これは、ひらがな文字1文字に対して、ローマ字が複数文字対応している例です。このとき"ho"は1編集単位です。

ローマ字複数文字に、ひらがな複数文字を対応させなければならない場合もあります。たとえば、ローマ字"ryoukin"に対する、ひらがな"りょうきん"がこれにあたります。この場合ひらがな"り"に対応するローマ字は存在しません。そのかわり、ひらがな"りょ"にローマ字"ryo"が対応します。従って、ローマ字文字列の中の"ryo", ひらがな文字列の中の"りょ"が編集単位となります。カーソルは編集単位でしか移動できませんので、「りょ」の間にカーソルが置かれることはありません。

上記 2 つの例からわかるように，編集単位を示す情報は「キー入力バッファ」と「キー・エコー・バッファ」の両方に必要です。この情報は，入力が行われるのと同時に付けられます。両方のバッファ内には，常に同じ数の編集単位が存在します。

ACTION の実行

アプリケーションは、GET KEY ACTION (VMS バインディングでは GET ACTION) によって返された ACTION を順番に実行します。この章は、ACTION を実行するモジュールの作成方法について書かれています。

3.1 ACTION の定義

- START

新しい入力を開始します。それまで確定していなかった文字列は確定し、変換領域はリセットされます。「漢字変換状態」にあるときには、かな漢字変換の学習も行わなければなりません。START の後、状態は「入力状態」に移ります。

- START_SELECTED

アプリケーションが提供する文字列を、かな漢字変換前の入力文字列とします。たとえば選択した領域をかな漢字変換の対象にするとときに、この機能が使われます。それまで確定していなかった文字列は確定し、変換領域はリセットされます。「漢字変換状態」にあるときには、かな漢字変換の学習も行わなければなりません。その後アプリケーションは文字列バッファに文字列を書き込みます。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場合には、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

ACTION の実行

3.1 ACTION の定義

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、ひらがなおよび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 (全角および半角) の小文字を、大文字に変換します。

UPPER は表示されている文字列に対して実行されます。

- LOWER

変換領域の中のアルファベット (全角および半角) の大文字を、小文字に変換します。

LOWER は表示されている文字列に対して実行されます。

- SYMBOL

変換領域の中のシンボル変換を実行します。詳しくは『日本語ユーティリティ利用者の手引き』を参照してください。変換できなかった場合は入力文字列に戻ります。

- 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。

ACTION の実行

3.1 ACTION の定義

- 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」に移ります。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

このとき押されたキーは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、そのキーに対して他の ACTION を同時に指定することはできません。NONE と他の ACTION を同時に指定したときその結果は保証されません。

1 度の GET KEY ACTION の呼び出しによって複数の ACTION が返されることもあります。このときにアプリケーションは、1 つ 1 つの ACTION を順番に実行します。VMS バインディングでは、SET KEY, GET ACTION の 2 つのルーチンによって GET KEY ACTION と同じ機能が実行されます。1 度の GET ACTION では、ACTION が 1 つだけ返されますので、IM_LASTACTION というステータスが返されるまで、繰り返し GET ACTION を call することになります。KEYBIND ファイルに定義されていないキーが押されたときには GET KEY ACTION から ACTION が返されません。これはそのキーはかな漢字変換キーではないという意味ですので、アプリケーションはかな漢字変換を終了して「初期状態」に戻し (INIT KEY STATE)、アプリケーションがそのキーに対応する機能を実行します。このとき、かな漢字変換を実行中で「漢字変換状態」にあると、変換結果の学習も行わなければなりません。

アプリケーションの作成方法

この章ではアプリケーションの作成方法について説明します。

4.1 アプリケーションが用意する資源

IMLIB を使うアプリケーションは、以下に示す資源を用意しなければなりません。

1. キー入力バッファ

キー入力に対応する文字列をそのままの形で保存するバッファ。

2. キー・エコー・バッファ

自動ローマ字かな変換をサポートする場合には、ローマ字変換後の文字列を保持するためにこのバッファが必要です。

3. 表示バッファ

画面表示と同じ文字列を保持するバッファ。

4.2 ACTION による状態の遷移

IMLIB を使うアプリケーションは、かな漢字変換における内部状態を知っておかなければなりません。状態は、起動時には「初期状態」にあり、その状態は実行される ACTION によって変化します。状態とその状態間の遷移を図 4-1 に示します。

図 4-1 かな漢字変換の内部状態と状態間の遷移

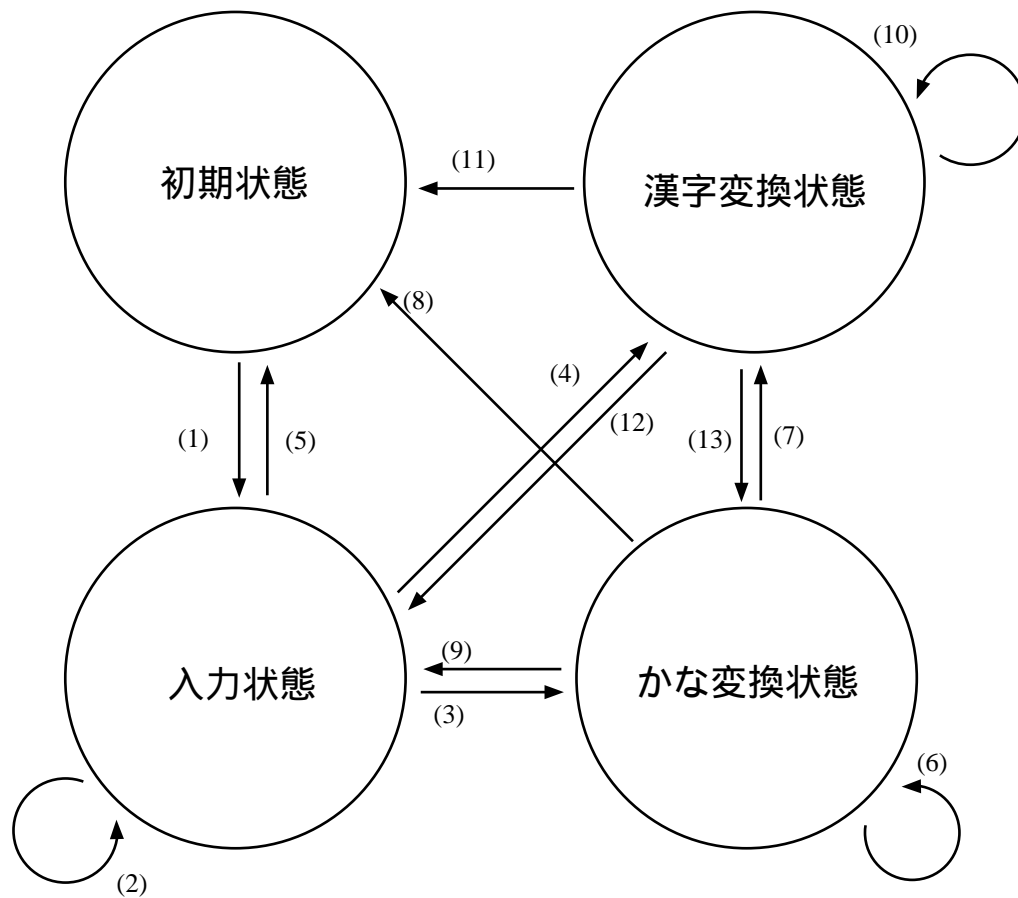


図 4-1 に示される 4 つの丸は内部状態を表し、それらをつなぐ矢印は状態間の遷移を示します。内部状態には、「初期状態」、「入力状態」、「かな変換状態」、「漢字変換状態」があります。それぞれの状態の意味を以下に示します。

- 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に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すので、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にこの状態です。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

入力した文字列に対してひらがな変換、カタカナ変換、半角カナ変換、全角変換、半角変換、シンボル変換などを実行した状態を総称して「かな変換状態」と呼びます。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。

状態の遷移は、GET KEY ACTION または SET KEY の呼び出しの際に得られる ACTION によって引き起こされます。状態遷移に付けられた番号に対応する ACTION を、以下に示します。ある状態にあるときに指定された ACTION が、この図の中になくときは、その状態においてその ACTION はエラーです。各 ACTION のエラー処理については、第 4.3 節を参照してください。

- (1) START, START_SELECTED
- (2) ECHO, MOVE_LEFT, MOVE_RIGHT, HEAD, TAIL, DELETE
- (3) HIRAGANA, KATAKANA, HANKAKU_KANA, HANKAKU, ZENKAKU, SYMBOL, UPPER, LOWER, RESTORE_ECHO

- (4) CONVERT
- (5) DONE
- (6) HIRAGANA, KATAKANA, HANKAKU_KANA , HANKAKU, ZENKAKU, SYMBOL, UPPER, LOWER
- (7) CONVERT
- (8) DONE
- (9) RESTORE_STRING
- (10) CLA_HIRAGANA, CLA_KATAKANA, CLA_HANKAKU_KANA , CLA_ZENKAKU, CLA_HANKAKU, NEXT_CLAUSE, PREV_CLAUSE, SHORTEN_CLAUSE, EXTEND_CLAUSE, NEXT_CANDIDATE, PREV_CANDIDATE
- (11) DONE
- (12) RESTORE_STRING
- (13) HIRAGANA, KATAKANA, HANKAKU_KANA, HANKAKU, ZENKAKU, SYMBOL, UPPER, LOWER, RESTORE_ECHO

4.3 ACTION の動作

以下に ACTION ごとに実行する機能を示します。同じ ACTION でも、そのときの状態によって実行する機能が異なります。

- START

- a. 初期状態のとき

バッファの初期設定を行い、入力できる状態にします。START を実行した後、ECHO によってバッファに文字が入っていきます。START によって入力状態に移ります。この遷移は図 4-1 の (1) で示されます。

- b. 入力状態、かな変換状態、漢字変換状態のとき

エラーです。エラー処理の方法はアプリケーションごとに定義されます。変換途中の文字列を確定するときには、INIT KEY STATE を呼んでください。また内部状態を「初期状態」に設定してください。

- **START_SELECTED**

- a. 初期状態のとき

バッファの初期設定を行った後、アプリケーションは選択領域の文字列をバッファに書き込みます。START_SELECTED によって入力状態に移ります。この遷移は図 4-1 の (1) で示されます。

- b. 入力状態、かな変換状態、漢字変換状態のとき

エラーです。エラー処理の方法はアプリケーションごとに定義されます。変換途中の文字列を確定するときには、INIT KEY STATE を呼んでください。また内部状態を「初期状態」に設定してください。

- **ECHO**

- a. 入力状態のとき

押されたキーに対応する文字を、入力バッファの現在の位置に挿入します。自動ローマ字かな変換モードのときには、キー入力バッファをローマ字かな変換したものをキー・エコー・バッファに挿入します。かなキーが押されたときに、アプリケーションが半角カナをキー入力バッファに入れるのか、全角かなを入れるのかは、PROFILE ファイルに書かれている DEC-JAPANESE.ECHO.kana という INDEX によって決めます。アプリケーションが半角カナをサポートしていないときには、半角カナを全角ひらがな、または全角カタカナに変換してキー入力バッファに入れてください。この遷移は図 4-1 の (2) で示されます。

- b. 初期状態、かな変換状態、漢字変換状態のとき

エラーです。エラー処理の方法はアプリケーションごとに定義されます。内部状態が「初期状態」でないときに、変換途中の文字列を確定する場合には INIT KEY STATE を呼んでください。また内部設定を「初期状態」に設定してください。

- **HIRAGANA, KATAKANA, HANKAKU_KANA , ZENKAKU, HANKAKU, UPPER, LOWER, SYMBOL**

- a. 入力状態のとき

キー入力バッファの内容に対して、ACTION に従った変換を行い、その結果を表示バッファに入れます。状態はかな変換状態に移ります。この遷移は図 4-1 の (3) で示されます。

b. かな変換状態のとき

キー入力バッファの内容に対して、ACTION に従った変換を行い、その結果を表示バッファに入れます。この遷移は図 4-1 の (6) で示されます。

c. 漢字変換状態のとき

キー入力バッファの内容に対して ACTION に従った変換を行い、その結果を表示バッファに入れます。内部状態はかな変換状態に移ります。この遷移は図 4-1 の (13) で示されます。

d. 初期状態のとき

エラーです。エラー処理の方法はアプリケーションごとに定義されます。

• CONVERT

a. 入力状態のとき

キー・エコー・バッファの内容に対してかな漢字変換を実行します。結果は表示バッファに入れます。状態は漢字変換状態に移ります。この遷移は図 4-1 の (4) で示されます。

b. かな変換状態のとき

キー・エコー・バッファの内容に対してかな漢字変換を実行します。結果は表示バッファに入れます。状態は漢字変換状態に移ります。この遷移は図 4-1 の (7) で示されます。

c. 初期状態のとき

エラーです。エラー処理の方法はアプリケーションごとに定義されます。

d. 漢字変換状態のとき

何も実行しません。

- CLA_HIRAGANA, CLA_KATAKANA, CLA_HANKAKU_KANA , CLA_ZENKAKU, CLA_HANKAKU, NEXT_CLAUSE, PREV_CLAUSE, SHORTEN_CLAUSE, EXTEND_CLAUSE, NEXT_CANDIDATE, PREV_CANDIDATE

a. 漢字変換状態のとき

現在の文節に対して ACTION に従った動作を行い、その結果を表示バッファに入れます。現在 DEC が提供するかな漢字変換ルーチンを使う場合には、CLA_ZENKAKU, CLA_HANKAKU は容易に実現することはできませんので、CLA_ZENKAKU, CLA_HANKAKU のサポートはアプリケーションの任意です。CLA_ZENKAKU, CLA_HANKAKU をサポートしないアプリケーションに、これらの ACTION が指定されたときには、アプリケーションは何も実行しません。この遷移は図 4-1 の (10) で示されます。

現在の文節が、最後の文節であるときに NEXT_CLAUSE が指定された場合、または現在の文節が最初の文節であるときに、PREV_CLAUSE が指定された場合には指定された動作を実行することができません。この場合には、PROFILE の INDEX "DEC-JAPANESE.OUTRANGE.clauseNumber" の値によって動作を決定します。

- none

何も実行しません。

- rotate

現在の文節が最後の文節のときに NEXT_CLAUSE が指定された場合には最初の文節に、最初の文節のときに PREV_CLAUSE が指定された場合には最後の文節に移ります。

- done

現在の変換を終了してアプリケーションがそのキーに対して定義した動作を実行します。

現在の文節長が最小であるときに SHORTEN_CLAUSE が指定されたとき、または現在の文節長が最大であるときに EXTEND_CLAUSE が指定された場合には、指定された動作を実行することができません。この場合に

は、PROFILE の INDEX "DEC-JAPANESE.OUTRANGE.clauseSize" の値によって動作を決定します。

- none

何も実行しません。

- rotate

現在の文節長が最小であるときに SHORTEN_CLAUSE が指定された場合には、文節長を最大長に、現在の文節長が最大であるときに EXTEND_CLAUSE が指定された場合には、文節長を最小長にします。

- done

現在の変換を終了して、アプリケーションがそのキーに対して定義した動作を実行します。

- b. 初期状態、入力状態、かな変換状態のとき

エラーです。エラー処理の方法はアプリケーションごとに定義されます。内部状態が「初期状態」でないときに、変換途中の文字列を確定する場合には、INIT KEY STATE を呼んでください。また内部状態を「初期状態」に設定してください。

- HEAD, TAIL

- a. 入力状態のとき

変換領域の先頭あるいは最後にカーソルを移動します。この遷移は図 4-1 の (2) で示されます。

- b. 初期状態、かな変換状態、漢字変換状態のとき

エラーです。エラー処理の方法はアプリケーションごとに定義されます。内部状態が「初期状態」でないときに、変換途中の文字列を確定する場合には、INIT KEY STATE を呼んでください。また内部状態を「初期状態」に設定してください。

- MOVE_LEFT, MOVE_RIGHT

- a. 入力状態のとき

変換領域内でカーソルを左右に移動します。カーソルが変換領域を外れるような操作をしたときは、すべてのバッファをクリアして初期状態に戻す方法と、カーソル移動を行わない方法の2つがあります。この遷移は図 4-1 の (2) で示されます。

現在のカーソル位置が、変換領域の最初の文字にあるときに MOVE_LEFT が指定されたとき、または現在のカーソル位置が、変換領域の最後の文字にあるときに MOVE_RIGHT が指定されたときには、指定された動作を実行することができません。この場合には、PROFILE の INDEX"DEC-JAPANESE.OUTRANGE.cursorPosition"の値によって動作を決定します。

- none

何も実行しません。

- done

現在の変換を終了してアプリケーションがそのキーに対して定義した動作を実行します。

b. 初期状態、かな変換状態、漢字変換状態のとき

エラーです。エラー処理の方法はアプリケーションごとに定義されます。内部状態が「初期状態」でないときに、変換途中の文字列を確定する場合には、INIT KEY STATE を呼んでください。また内部状態を「初期状態」に設定してください。

- DELETE

a. 入力状態のとき

カーソルの左側の1文字を消去します。この遷移は図 4-1 の (2) で示されます。

b. 初期状態，かな変換状態，漢字変換状態のとき

エラーです。エラー処理の方法はアプリケーションごとに定義されます。内部状態が「初期状態」でないときに，変換途中の文字列を確定する場合には，INIT KEY STATE を呼んでください。また内部状態を「初期状態」に設定してください。

• DONE

a. 入力状態のとき

表示バッファの内容を入力文字列として格納し，すべてのバッファ内の文字をすべて消去します。スクリーンにビデオ属性が表示されていたら，それをリセットします。この遷移は図 4-1 の (5) で示されます。

b. かな変換状態のとき

表示バッファの内容を入力文字列として格納し，すべてのバッファ内の文字をすべて消去します。スクリーンにビデオ属性が表示されていたら，それをリセットします。この遷移は図 4-1 の (8) で示されます。

c. 漢字変換状態のとき

表示バッファの内容を入力文字列として格納し，すべてのバッファ内の文字をすべて消去します。スクリーンにビデオ属性が表示されていたら，それをリセットします。また現在の変換結果の学習を行います。この遷移は図 4-1 の (11) で示されます。

d. 初期状態のとき

何も実行しません。

• RESTORE_STRING

a. かな変換状態のとき

キー・エコー・バッファ内の文字列を，表示バッファにコピーします。状態は入力状態に移ります。この遷移は図 4-1 の (9) で示されます。

b. 漢字変換状態のとき

キー・エコー・バッファ内の文字列を表示バッファにコピーします。状態は入力状態に移ります。変換の学習は行いません。この遷移は図 4-1 の (12) で示されます。

c. 初期状態のとき

エラーです。エラー処理の方法はアプリケーションごとに定義されます。内部状態が「初期状態」でないときに、変換途中の文字列を確定する場合には、INIT KEY STATE を呼んでください。また内部状態を「初期状態」に設定してください。

d. 入力状態のとき

何も実行しません。

• RESTORE_ECHO

a. かな変換状態のとき

キー・エコー・バッファ内の文字列を表示バッファにコピーします。状態はかな変換状態に移ります。この遷移は図 4-1 の (6) で示されます。

b. 漢字変換状態のとき

キー・エコー・バッファ内の文字列を表示バッファにコピーします。状態はかな変換状態に移ります。この遷移は図 4-1 の (13) で示されます。

c. 初期状態のとき

エラーです。エラー処理の方法はアプリケーションごとに定義されます。内部状態が「初期状態」でないときに、変換途中の文字列を確定する場合には、INIT KEY STATE を呼んでください。また内部状態を「初期状態」に設定してください。

d. 入力状態のとき

キー・エコー・バッファ内の文字列を表示バッファにコピーします。状態は、かな変換状態に移ります。この遷移は図 4-1 の (3) で示されます。

- NONE

キーが定義されていないことを明示的に指定する ACTION です。この場合には、アプリケーションは GET KEY ACTION または SET KEY から、IM__KEYNOTDEFINED という条件値が返されたときと同じ動作をします。

- a. 初期状態のとき

すべてのバッファ内の文字をすべて消去します。スクリーンにビデオ属性が表示されていたら、それをリセットします。内部状態が「初期状態」でないときには、INIT KEY STATE を呼んでください。

- b. 入力状態、かな変換状態のとき

すべてのバッファ内の文字をすべて消去します。スクリーンにビデオ属性が表示されていたら、それをリセットします。また、INIT KEY STATE を呼んでください。

- c. 漢字変換状態のとき

現在の変換結果を学習させてから、上記の動作を実行します。

表 4-1 に、現在の状態と ACTION による状態遷移をまとめています。表の中の数字は図 4-1 の状態遷移を示す矢印に付けられた数字と一致しています。表の中で使われている表記は以下のとおりです。

- 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 4-1 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)

IMLIB ライブラリ・ルーチン

この章では、IMLIB ルーチンについて説明します。

5.1 IMLIB ルーチンとは？

IMLIB ルーチンとは、ユーザが、かな漢字変換のキー定義をできるようなアプリケーションを作成するためのルーチンです。

5.1.1 プログラミング言語

IMLIB ルーチンは、C 言語用のインターフェイス (C バインディング) と、C 以外の言語のためのインターフェイス (VMS バインディング) の 2 つのインターフェイスを提供します。

5.2 PROFILE に関連するルーチン

5.2.1 PROFILE のオープン

IMLIB を使うアプリケーションは、まず最初に PROFILE をオープンします (OPEN PROFILE)。OPEN PROFILE をしなければ、IMLIB の機能を使うことはできません。通常はアプリケーションの起動時に OPEN PROFILE が実行されます。OPEN PROFILE によって PROFILE の情報は、すべてメモリ中に読み込まれます。OPEN PROFILE が返す整数値 UNIT ID は、IMLIB の他のルーチンを呼ぶときに使われます。UNIT ID によって複数のコンテキストをコントロールします。

5.2.2 PROFILE のクローズ

アプリケーションは、IMLIB を使い終った後で PROFILE をクローズします (CLOSE PROFILE)。CLOSE PROFILE を行った後は、IMLIB の機能を使うことはできません。通常はアプリケーションの終了時に CLOSE PROFILE を実行します。

5.2.3 PROFILE データの検索

PROFILE に書かれたデータを検索するには、GET PROFILE DATA を使います。GET PROFILE DATA は、OPEN PROFILE によってメモリ中に読み込まれたデータを検索して、値を返します。

5.2.4 PROFILE データの変更

アプリケーションが PROFILE に書かれたデータを書き換えるには、SET PROFILE DATA を使います。SET PROFILE DATA は、メモリ中に読み込まれた PROFILE のデータを変更しますので、その後の GET PROFILE DATA は変更の影響を受けます。WRITE PROFILE によって変更されたメモリ中のデータが、ファイルに書き込まれます。

5.2.5 PROFILE データの書き込み

SET PROFILE DATA によって変更されたメモリ中の内容をファイルに反映させるには、WRITE PROFILE を使います。WRITE PROFILE を行わなければ、メモリ中の変更は CLOSE PROFILE によって失われます。

5.3 KEYBIND に関連するルーチン

5.3.1 KEYBIND 情報の設定

アプリケーションは、SET KEYBIND によってバイナリ形式の KEYBIND ファイルの内容をメモリ中に読み込みます。キーに対応する ACTION を得るためには、SET KEYBIND を実行しておかなければなりません。KEYBIND 情報は 1 つの UNIT ID につき 1 つだけ設定できます。すでに、指定した UNIT ID によって KEYBIND 情報が読み込まれていた場合には、古い情報は消えて新しいものと置き換わります。

5.3.2 ACTION を得るルーチン (C バインディング)

C バインディングでは、GET KEY ACTION によってキーに定義された ACTION の情報を得ます。GET KEY ACTION は 1 度に複数の ACTION を返しますので、アプリケーションは得られた ACTION を 1 つずつ実行します。

5.3.3 ACTION を得るルーチン (VMS バインディング)

VMS バインディングでは、ACTION を得るために 2 つのルーチンを使います。まず、SET KEY によってキーを通知して、GET ACTION によって ACTION を得ます。GET ACTION は 1 度に 1 つの ACTION だけしか返さないで、キーに複数の ACTION が定義されているときには、GET ACTION を繰り返し呼んで、得られた ACTION を 1 つずつ実行します。

5.3.4 STATE を初期化するルーチン

KEYBIND の内部状態を初期化するには、INIT KEY STATE を使います。INIT KEY STATE によって、STATE は強制的に "initial" に移ります。GET KEY ACTION (C バインディング) または GET ACTION (VMS バインディング) によって、KEYBIND 情報の中にそのキーが定義されていないというステータスが返されると、アプリケーションが「初期状態」にない場合には、INIT KEY STATE を呼んで KEYBIND の内部状態を初期化します。

5.3.5 直前の STATE に戻すルーチン

最近のキーまたは最近の複数キーによる ACTION が不正な場合、またはその ACTION の結果が不正の場合に、RECOVER KEY STATE によって、そのキーまたは複数キーが押される直前の STATE に戻すことができます。アプリケーションがその時点の状態を保存していれば、その状態から実行を再開できます。

5.4 KEYCODE に関連するルーチン

5.4.1 キー名文字列から KEYCODE への変換

GET KEYCODE は、キーの名前を示す文字列を、IMLIB が定義する KEYCODE に変換します。個々のキーのキー名については、『ユーザ・キー定義 利用者の手引き』を参照してください。

5.4.2 KEYCODE に対応する文字を返すルーチン

キーの中には、画面に表示される文字に対応するキーがあります。GET CHARACTER は、KEYCODE に対応する文字を得るのに使われます。指定された KEYCODE が、表示される文字に対応しない場合には、GET CHARACTER はそのことを示すステータスを返します。

5.4.3 半角カナの KEYCODE に対応する文字を返すルーチン

半角カナは文字コードセットによって若干コードが変わりますので、GET CHARACTER にコードセット指定のパラメータを加えた MC GET CHARACTER を使って文字を得ます。MC GET CHARACTER は半角カナの KEY CODE 以外の入力に対しては GET CHARACTER と同じ動作を行います。

5.4.4 キーが発生するコードから KEYCODE への変換

キーが発生するコードは、ENCODE KEY によってそのキーに対応する KEYCODE に変換されます。ENCODE KEY には数字キーパッド、編集キーパッド、ファンクション・キーが発生するエスケープ・シーケンスも正しく処理します。

5.4.5 KEYSYM から KEYCODE への変換

DECwindows のキー入力ルーチン XLookupString (3X11) が返す KEYSYM の値を KEYSYM TO KEYCODE に渡すと、KEYSYM は IMLIB の KEYCODE に変換されます。

5.4.6 半角カナから KEYCODE への変換

半角カナは文字コードセットによって若干コードが変わりますので、ENCODE KEY にコードセット指定のためのパラメータを加えた MC ENCODE KEY を使って KEYCODE に変換します。MC ENCODE KEY は半角カナ以外の入力に対しては ENCODE KEY と同じ動作を行います。

5.5 IMLIB ルーチン

IMLIB ルーチンのインターフェイスと機能についてに個々に示します。

OPEN PROFILE

PROFILE を読み込み，その context を示す unit ID を返します。

VMS バインディング

```
status=IM$OPEN_PROFILE(file, unit_id)
```

戻り値

OpenVMS 用法	cond_value
データ型	longword (unsigned)
アクセス	write_only
受け渡し方	by value

引数

file

OpenVMS 用法	char string
データ型	char string
アクセス	read only
受け渡し方	by descriptor

PROFILE のファイル指定です。このパラメータに長さ 0 の文字列が渡されると，論理名 IM\$PROFILE で指定されるファイルが，省略時の PROFILE として使われます。

unit_id

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	write only
受け渡し方	by reference

Unit ID と呼ばれる正の整数値が返されます。Unit ID は以後の IMLIB ルーチンの呼び出しで使われます。

C バインディング

```
status=lmOpenProfile(file, unit_id)
```

戻り値

OPEN PROFILE は条件値を返します。

引数

char *file (Input)

PROFILE のファイル指定です。このパラメータにヌル・ポインタ (NULL) が渡されると、論理名 IMSPROFILE で指定されるファイルが省略時の PROFILE として使われます。

int *unit_id (Output)

Unit ID と呼ばれる正の整数値が返されます。Unit ID は以後の IMLIB ルーチンの呼び出しで使われます。

説明

OPEN PROFILE は PROFILE ファイルを読み込み、メモリ上に PROFILE のデータ構造を作ります。OPEN PROFILE は、このデータ構造を示すための Unit ID を返します。Unit ID は、以後の IMLIB ルーチンの呼び出しにおいてここで作られたデータ構造を指定するために使われます。

OPEN PROFILE

省略時の PROFILE ファイルの default file specification は , "IMSDEFAULTS:.DAT" です。したがって , 論理名 IMSDEFAULTS で指定されるディレクトリ以外のディレクトリに PROFILE ファイルがある場合は , 論理名 IM\$PROFILE をディレクトリ名から指定しなければなりません。

論理名 IM\$DEFAULTS は , 通常 SYS\$LOGIN: と SYS\$LIBRARY: を指定しています。

戻される条件値

シンボル	重大度	メッセージ
IM_ _SUCCESS	Success	Normal successful completion.
IM_ _FILNOTFND	Error	File not found.
IM_ _CANNOTOPN	Error	Cannot open.
IM_ _INSVIRMEM	Error	Insufficient virtual memory.
IM_ _SYNTAXERR	Error	Syntax error.
IM_ _READERR	Error	File read error.
IM_ _INVSTRDES	Error	Invalid string descriptor.

WRITE PROFILE

PROFILE のデータ構造を指定したファイルに書き込みます。

VMS バインディング

status=IM\$WRITE_PROFILE(*unit_id*, *file*)

戻り値

OpenVMS 用法	cond_value
データ型	longword (unsigned)
アクセス	write_only
受け渡し方	by value

引数

unit_id

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	read only
受け渡し方	by reference

OPEN PROFILE によって返された整数値。

file

OpenVMS 用法	char string
データ型	char string
アクセス	read only
受け渡し方	by descriptor

WRITE PROFILE

書き込む PROFILE ファイル名の指定。文字列の長さが 0 のときは OPEN PROFILE で使われたファイルと同じファイル名が使われます。

C バインディング

```
status=lmWriteProfile(unit_id, file)
```

戻り値

WRITE PROFILE は条件値を返します。

引数

int unit_id (Input)

OPEN PROFILE によって返された整数値。

char *file (Input)

書き込む PROFILE ファイル名の指定。ヌル・ポインタが (NULL) 指定されたときは OPEN PROFILE で使われたファイルと同じファイル名が使われます。

説明

WRITE PROFILE は PROFILE のデータをファイルに書き込みます。

戻される条件値

シンボル	重大度	メッセージ
IM_ SUCCESS	Success	Normal successful completion.
IM_ CANNOTOPN	Error	Cannot open.
IM_ INSVIRMEM	Error	Insufficient virtual memory.
IM_ ILLUNIT	Error	Illegal unit ID.
IM_ WRITEERR	Error	File write error.
IM_ INVSTRDES	Error	Invalid string descriptor.
IM_ INSPRV	Error	Insufficient privilege or file protection violation

CLOSE PROFILE

CLOSE PROFILE は OPEN PROFILE と SET KEYBIND によってメモリ中に読み込まれた内容をすべて廃棄します。

VMS バインディング

status = IM\$CLOSE_PROFILE(*unit_id*)

戻り値

OpenVMS 用法	cond_value
データ型	longword (unsigned)
アクセス	write_only
受け渡し方	by value

引数

unit_id	
OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	read only
受け渡し方	by reference

OPEN PROFILE によって返された整数値。

C バインディング

status = ImCloseProfile(*unit_id*)

戻り値

CLOSE PROFILE は条件値を返します。

引数

int unit_id (Input)

OPEN PROFILE によって返された整数値。

説明

CLOSE PROFILE は PROFILE と KEYBIND の構造をすべて廃棄します。このルーチンは OPEN PROFILE と SET KEYBIND によって割り当てたメモリをすべて解放します。

戻される条件値

シンボル	重大度	メッセージ
IM_SUCCESS	Success	Normal successful completion.
IM_ILLUNIT	Error	Illegal unit ID.

SET PROFILE DATA

PROFILE のデータを変更または追加します。

VMS バインディング

status=IM\$SET_PROFILE_DATA(unit_id, index-string, value-string)

戻り値

OpenVMS 用法	cond_value
データ型	longword (unsigned)
アクセス	write_only
受け渡し方	by value

引数

unit_id

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	read only
受け渡し方	by reference

OPEN PROFILE によって返された整数値。

index-string

OpenVMS 用法	char string
データ型	char string
アクセス	read only
受け渡し方	by descriptor

INDEX 文字列。

value-string

OpenVMS 用法	char string
データ型	char string
アクセス	read only
受け渡し方	by descriptor

VALUE 文字列。

C バインディング

status=ImSetProfileData(unit_id, index-string, value-string)

戻り値

SET PROFILE DATA は条件値を返します。

引数

int unit_id (Input)

OPEN PROFILE によって返された整数値。

char *index-string (Input)

INDEX 文字列。

char *value-string (Input)

VALUE 文字列。

SET PROFILE DATA

説明

SET PROFILE DATA は PROFILE のデータを変更したり、新しいデータを追加したりするために使われます。指定した INDEX 文字列がすでに PROFILE に存在すると、その VALUE が指定したもので置きかわります。指定した INDEX 文字列が存在しない場合には新しい INDEX が作られます。

SET PROFILE DATA は、PROFILE のメモリ上のデータを変更します。変更されたデータをファイルに書き込むには WRITE PROFILE を使ってください。なお、INDEX 文字列は大文字・小文字を区別しません。

戻される条件値

シンボル	重大度	メッセージ
IM_ _SUCCESS	Success	Normal successful completion.
IM_ _INSVIRMEM	Error	Insufficient virtual memory.
IM_ _INVSTRDES	Error	Invalid string descriptor.
IM_ _ILLUNIT	Error	Illegal unit ID.

GET PROFILE DATA

PROFILE に書かれたデータを検索します。

VMS バインディング

status = IM\$GET_PROFILE_DATA(*unit_id*, *index-string*, *value-string*, *len*)

戻り値

OpenVMS 用法	cond_value
データ型	longword (unsigned)
アクセス	write_only
受け渡し方	by value

引数

unit_id

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	read only
受け渡し方	by reference

OPEN PROFILE によって返された整数値。

index-string

OpenVMS 用法	char string
データ型	char string
アクセス	read only
受け渡し方	by descriptor

INDEX 文字列。

GET PROFILE DATA

value-string

OpenVMS 用法	char string
データ型	char string
アクセス	write only
受け渡し方	by descriptor

INDEX 文字列に対応する VALUE 文字列。

value-length

OpenVMS 用法	word_unsigned
データ型	word (unsigned)
アクセス	write only
受け渡し方	by reference

GET PROFILE DATA によって得られた、VALUE 文字列のバイト長。これには、固定長文字列に対するパディング文字は含まれていません。value-length は、value-string の長さを格納する 1 ワードのアドレスです。value-string が value-string デスクリプタで指定された長さに切り取られた場合は、value-length にはその長さが入ります。

C バインディング

```
status=ImGetProfileData(unit_id, index-string, value-string, buflen)
```

戻り値

GET PROFILE DATA は条件値を返します。

引数

int unit_id (Input)

OPEN PROFILE によって返された整数値。

char *index-string (Input)

INDEX 文字列。

char *value-string (Output)

VALUE 文字列が返されるバッファのアドレス。

int buflen (Input)

VALUE 文字列が返されるバッファの長さ。

説明

GET PROFILE DATA は、PROFILE で指定されたデータを検索するために使われます。INDEX 文字列で指定された PROFILE データが、VALUE 文字列の受け取りバッファに入り切らない場合は、IM_ TRUNCATED が返され、途中までの値がバッファに書き込まれます。この状態になったら、アプリケーションはより大きいバッファを作って再度 GET PROFILE DATA を呼ぶ必要があります。なお、INDEX 文字列の大文字・小文字は区別されません。

戻される条件値

シンボル	重大度	メッセージ
IM_ SUCCESS	Success	Normal successful completion.
IM_ INSVIRTUALMEM	Error	Insufficient virtual memory.
IM_ ILLUNIT	Error	Illegal unit ID.
IM_ NOINDEX	Error	No index.
IM_ TRUNCATED	Warning	String truncated.
IM_ INVSTRDES	Error	Invalid string descriptor.
IM_ FATERRLIB	Fatal	Fatal library internal error.

SET KEYBIND

バイナリ形式の KEYBIND ファイルを読み込み、キー定義のデータ構造を作ります。

VMS バインディング

status = IM\$SET_KEYBIND(*unit_id*, *file*, *level*)

戻り値

OpenVMS 用法	cond_value
データ型	longword (unsigned)
アクセス	read_only
受け渡し方	by value

引数

unit_id

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	write only
受け渡し方	by reference

OPEN PROFILE によって返された条件値。

file

OpenVMS 用法	char string
データ型	char string
アクセス	read only
受け渡し方	by descriptor

KEYBIND ファイルのファイル指定をする文字列。このパラメータが長さ 0 の文字列のときは、PROFILE の DEC-JAPANESE.KEY.keybind という INDEX で指定されるファイルが、省略時の KEYBIND ファイルとして使われます。

level

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	read only
受け渡し方	by reference

アプリケーションがサポートする ACTION のレベルを指定します。このマニュアルに記述されている ACTION のレベルは "2" です。このパラメータには整数値 2 を指定してください。サポートレベルとして 1 を指定した場合、サポートレベルが 2 以上の KEYBIND ファイルを読み込むことはできません。

C バインディング

status=lmSetKeybind(unit_id, file, level)

戻り値

SET KEYBIND は条件値を返します。

引数

int unit_id (Input)

OPEN PROFILE によって返された整数値。

char *file (Input)

KEYBIND ファイルのファイル指定をする文字列。このパラメータにヌル・ポインタ (NULL) が指定されたときには、PROFILE の DEC-JAPANESE.KEY.keybind

SET KEYBIND

という INDEX で指定されるファイルが、省略時の KEYBIND ファイルとして使われます。

int level (Input)

アプリケーションがサポートする ACTION のレベルを指定します。このマニュアルに記述されている ACTION のレベルは "2" です。このパラメータには整数値 2 を指定してください。サポートレベルとして 1 を指定した場合、サポートレベルが 2 以上の KEYBIND ファイルを読み込むことはできません。

説明

SET KEYBIND は KEYBIND のデータ構造を作成し、そのデータ構造を PROFILE に結び付けます。指定した PROFILE に既に KEYBIND が存在するときには、古い KEYBIND が取り除かれ、新しく指定したものに置きかわります。

省略時の KEYBIND ファイルの default file specification は、"IMS\$DEFAULTS:..IMS\$DAT" です。したがって、論理名 IMS\$DEFAULTS で指定されるディレクトリ以外のディレクトリに KEYBIND ファイルがある場合は、PROFILE ファイル中の VALUE をディレクトリ名から指定しなければなりません。

論理名 IMS\$DEFAULTS は、通常 SYS\$LOGIN: と SYS\$LIBRARY: を指定しています。

戻される条件値

シンボル	重大度	メッセージ
IM_ _SUCCESS	Success	Normal successful completion.
IM_ _FILNOTFND	Error	File not found.
IM_ _CANNOTOPN	Error	Cannot open.
IM_ _INSVIRMEM	Error	Insufficient virtual memory.
IM_ _ILLUNIT	Error	Illegal unit ID.
IM_ _ILLFORMAT	Error	Illegal data format.
IM_ _ILLLEVEL	Error	Illegal level number or file level mismatch.
IM_ _NOINDEX	Error	No index.
IM_ _READERR	Error	File read error.
IM_ _INVSTRDES	Error	Invalid string descriptor.

GET KEY ACTION

キーに定義された ACTION を調べます。

VMS バインディング

VMS バインディングはありません。SET KEY および GET ACTION を参照してください。

C バインディング

```
status=lmGetKeyAction(unit_id, key_code, action_list)
```

戻り値

GET KEY ACTION は条件値を返します。

引数

int unit_id (Input)

OPEN PROFILE によって返された整数値。

int key_code (Input)

キーの整数表現 (KEYCODE)。

KEYCODE の定数表記は付録 B を参照してください。

int *action_list [] (Output)

ACTION が返される配列。配列の最後のエレメントは値 0 で示されます。

ACTION の定数表記は付録 B を参照してください。

説明

GET KEY ACTION は、キーに定義された ACTION を得るために使われます。ACTION は整数の配列として返されます。同じキーであっても、そのときの状態によって ACTION が変わることがありますので、キーが入力されるたびに GET KEY ACTION によって実行する ACTION を得てください。

"Key not defined"という条件値は、そのキーがその状態で、ユーザによって定義されていないことを示します。この場合は、アプリケーションが定義する機能を実行します。

"Wait for next key"という条件値は、GOLDキーのようなそのキーだけでは完結しないキーが入力されたことを示します。この場合、アプリケーションは GET KEY ACTION をもう一度呼んで、複数キーに定義された ACTION を得なければなりません。アプリケーションは、GET KEY ACTION を "Wait for next key"が出なくなるまで続けることになります。複数キーは最大 16 個まで許されています。

戻される条件値

シンボル	重大度	メッセージ
IM_ _SUCCESS	Success	Normal successful completion.
IM_ _ILLUNIT	Error	Illegal unit ID.
IM_ _INSVIRMEM	Error	Insufficient virtual memory.
IM_ _NOKEYBIND	Error	Keybind data is not set yet. Execute SET KEYBIND first.
IM_ _KEYNOTDEFINED	Information	The specified key is not defined. Execute application defined action.
IM_ _PARTIALKEY	Information	The key is middle of a multiple key definition. Wait for the next key to complete the key definition.
IM_ _NOACTION	Information	The key has no action.

SET KEY

キーに定義された ACTION を得るために、キーを指定します。

VMS バインディング

status=IM\$SET_KEY(*unit_id*, *key_code*)

戻り値

OpenVMS 用法	cond_value
データ型	longword (unsigned)
アクセス	write_only
受け渡し方	by value

引数

unit_id

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	read only
受け渡し方	by reference

OPEN PROFILE によって返された条件値。

key_code

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	read only
受け渡し方	by reference

キーの整数表現 (KEYCODE)。

KEYCODE の定数表記は付録 B を参照してください。

C バインディング

C バインディングはありません。GET KEY ACTION を参照してください。

説明

SET KEY は GET ACTION と組み合わせられて、キーに定義された ACTION を得るために使われます。SET KEY でキーを指定し、GET ACTION を繰り返し呼ぶことで複数の ACTION を得ます。同じキーであってもそのときの状態によって ACTION が変わることがありますので、キーが入力されるたびに ACTION を得てください。

"Key not defined"という条件値は、そのキーがその状態で、ユーザによって定義されていないことを示します。この場合は、アプリケーションが定義する機能を実行します。

"Wait for next key"という条件値は、GOLDキーのようなそのキーだけでは完結しないキーが入力されたことを示します。この場合、アプリケーションは SET KEY をもう一度呼ばなければなりません。複数キーは最大 16 個まで許されています。アプリケーションは SET KEY を "Wait for next key"が出なくなるまで続けることになります。

SET KEY

戻される条件値

シンボル	重大度	メッセージ
IM_ SUCCESS	Success	Normal successful completion.
IM_ ILLUNIT	Error	Illegal unit ID.
IM_ INSVIRMEM	Error	Insufficient virtual memory.
IM_ NOKEYBIND	Error	Keybind data is not set yet. Execute SET KEYBIND first.
IM_ KEYNOTDEFINED	Information	The specified key is not defined. Execute application defined action.
IM_ PARTIALKEY	Information	The key is middle of a multiple key definition. Wait for the next key to complete the key definition.
IM_ NOACTION	Information	The key has no action.

GET ACTION

SET KEY によって指定されたキーに定義された ACTION を得ます。

VMS バインディング

status=IM\$GET_ACTION(*unit_id*, *action*)

戻り値

OpenVMS 用法	cond_value
データ型	longword (unsigned)
アクセス	write_only
受け渡し方	by value

引数

unit_id

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	read only
受け渡し方	by reference

OPEN PROFILE によって返された整数値。

action

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	write only
受け渡し方	by reference

次に実行される ACTION。

GET ACTION

ACTION の定数表記は付録 B を参照してください。

C バインディング

C バインディングはありません。GET KEY ACTION を参照してください。

説明

SET KEY は GET ACTION と組み合わせられて、キーに定義された ACTION を得るために使われます。SET KEY でキーを指定し、GET ACTION を繰り返し呼ぶことで、複数の ACTION を得ます。同じキーであってもそのときの状態によって ACTION が変わることがありますので、キーが入力されるたびに ACTION を得てください。最後のアクションを渡す際に、GET ACTION は action にアクションコードを入れて IM_LASTACTION を返します。IM_LASTACTION をエラーコードと間違えて処理しないようにしてください。

戻される条件値

シンボル	重大度	メッセージ
IM_SUCCESS	Success	Normal successful completion.
IM_LASTACTION	Success	This is the last action.
IM_ILLUNIT	Error	Illegal unit ID.
IM_KEYNOTSET	Error	Key is not specified yet. Call SET KEY before calling GET ACTION.

RECOVER KEY STATE

最後に GET KEY ACTION または SET KEY が実行された前の STATE に戻します。

VMS バインディング

```
status=IM$RECOVER_KEY_STATE(unit_id)
```

引数

unit_id

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	read only
受け渡し方	by reference

OPEN PROFILE によって返された条件値。

C バインディング

```
status=ImRecoverKeyState(unit_id)
```

戻り値

RECOVER KEY STATE は条件値を返します。

RECOVER KEY STATE

引数

int unit_id (Input)

OPEN PROFILE によって返された整数値。

説明

IMLIB に与えられた最近のキー，または最近の複数キーによる ACTION が不正の場合，あるいはその ACTION の実行結果が不正の場合に，変換中の文字列を確定することを避けるため，そのキーまたは複数キーが押される直前の STATE に回復します。

注意: 直前の STATE に戻すのは，アプリケーションの責任です。最近の ACTION に DONE が入っていた場合，DONE の前の状態に戻すことはできません。

戻される条件値

シンボル	重大度	メッセージ
IM_ _SUCCESS	Success	Normal successful completion.
IM_ _NORECOVER	Warning	No previous state to recover.
IM_ _ILLUNIT	Error	Illegal unit ID.
IM_ _NOKEYBIND	Error	Keybind data is not set up yet.

GET KEYCODE

キー名を示す文字列から、そのキーに対応する KEYCODE を得ます。

VMS バインディング

```
status=IM$GET_KEYCODE(key_name, keycode)
```

戻り値

OpenVMS 用法	cond_value
データ型	longword (unsigned)
アクセス	write_only
受け渡し方	by value

引数

key_name

OpenVMS 用法	char string
データ型	char string
アクセス	read only
受け渡し方	by descriptor

KEYCODE を知りたいキーのキー名を示す文字列。個々のキーのキー名については、『ユーザ・キー定義 利用者の手引き』を参照してください。

keycode

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	write only
受け渡し方	by reference

GET KEYCODE

キー名に対応する KEYCODE。

C バインディング

keycode=ImGetKeycode(key_name)

戻り値

GET KEY CODE はパラメータに渡されたキー名に対応する KEYCODE を返します。渡された文字列がキー名でない場合は GET KEYCODE は 0 を返します。

引数

char *key_name (Input)

KEYCODE を知りたいキーのキー名を示す文字列。

説明

GET KEYCODE は、キー名を示す文字列を整数値 KEYCODE に変換します。大文字・小文字は区別されません。

戻される条件値

シンボル	重大度	メッセージ
IM_ SUCCESS	Success	Normal successful completion.
IM_ NOTKEYNAME	Error	The string is not a key name.
IM_ INVSTRDES	Error	Invalid string descriptor.

GET CHARACTER

KEYCODE の表示文字を調べます。

VMS バインディング

status=IM\$GET_CHARACTER(*keycode*, *ch-string*, *ch-len*)

戻り値

OpenVMS 用法	cond_value
データ型	longword (unsigned)
アクセス	write_only
受け渡し方	by value

引数

keycode

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	read only
受け渡し方	by reference

整数値 KEYCODE。

ch-string

OpenVMS 用法	char string
データ型	char string
アクセス	write only
受け渡し方	by descriptor

KEYCODE によって指定されたキーの表示文字。指定された KEYCODE がカナキーの KEYCODE のときには、半角カタカナ・コードが返されます。

ch-len

OpenVMS 用法	word_unsigned
データ型	word (unsigned)
アクセス	write only
受け渡し方	by reference

"ch-string"に書き込まれた文字のバイト数。固定長文字列の場合のパディング文字は含みません。"ch-length"はこのバイト数を示す 1 ワードのアドレスです。文字列が ch-string デスクリプタによって指定された長さに切られたときには、"ch-length"はこの長さになります。

C バインディング

```
status=lmGetCharacter(keycode, ch)
```

戻り値

GET CHARACTER は条件値を返します。

引数

unsigned int keycode (Input)

整数値 KEYCODE。

unsigned char *ch (Output)

KEYCODE に対応する表示文字の文字列。この文字列はヌル文字で終了します。指定された KEYCODE がカナキーの KEYCODE のときには、半角カタカナ・コードが返されます。

GET CHARACTER

説明

GET CHARACTER は、KEYCODE に対応する表示文字を得るのに使われます。もし、'g' キーに対応する KEYCODE を GET CHARACTER に渡すと、文字列 "g" を得ることができます。

表示文字は最大 1 バイトなので、アプリケーションは "ch" のために、終端のヌル文字を含めて最低 2 バイト用意しなければなりません。

戻される条件値

シンボル	重大度	メッセージ
IM_ _SUCCESS	Success	Normal successful completion.
IM_ _NOPRINTABLE	Information	The keycode is not for a key of printable character.
IM_ _INVKEYCODE	Error	Invalid keycode
IM_ _INVSTRDES	Error	Invalid string descriptor.
IM_ _FATERRLIB	Fatal	Fatal library internal error.

MC GET CHARACTER

KEYCODE の表示文字を調べます。

VMS バインディング

status=IM\$MC_GET_CHARACTER(keycode, ch-string, ch-len, codeset)

戻り値

OpenVMS 用法	cond_value
データ型	longword (unsigned)
アクセス	write_only
受け渡し方	by value

引数

keycode

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	read only
受け渡し方	by reference

整数値 KEYCODE。

ch-string

OpenVMS 用法	char string
データ型	char string
アクセス	write only
受け渡し方	by descriptor

MC GET CHARACTER

KEYCODE によって指定されたキーの表示文字。指定された KEYCODE がカナキーの KEYCODE のときには、半角カタカナ・コードが返されます。

ch-len

OpenVMS 用法	word_unsigned
データ型	word (unsigned)
アクセス	write only
受け渡し方	by reference

"ch-string"に書き込まれた文字のバイト数。固定長文字列の場合のパディング文字は含みません。"ch-length"はこのバイト数を示す 1 ワードのアドレスです。文字列が ch-string デスクリプタによって指定された長さに切られた場合には、"ch-length"はこの長さになります。

codeset

OpenVMS 用法	longword_unsigned
データ型	longword_integer(unsigned)
アクセス	read only
受け渡し方	by reference

アプリケーションで使用しているコードセット。

C バインディング

```
status=lmMCGetCharacter(keycode, ch, codeset)
```

戻り値

GET CHARACTER は条件値を返します。

引数

unsigned int keycode (Input)

整数値 KEYCODE。

unsigned char *ch (Output)

KEYCODE に対応する表示文字の文字列。この文字列はヌル文字で終了します。
指定された KEYCODE がカナキーの KEYCODE のときには、半角カタカナ・コードが返されます。

int codeset (Input)

アプリケーションで使用しているコードセット。

説明

MC GET CHARACTER は、KEYCODE に対応する表示文字を得るのに使われます。もし、'g' キーに対応する KEYCODE を MC GET CHARACTER に渡すと、文字列 "g" を得ることができます。

MC GET CHARACTER は、コードセットにより異なる半角カナのコードに対応しています。コードセットして指定できるシンボルは次のとおりです。

シンボル	コードセット	半角カナコード
IM\$C_DECKANJI	DEC 漢字	0xA1 ~ 0xDF
IM\$C_SDECKANJI	SuperDEC 漢字	SS2(0x8E) + 0xA1 ~ 0xDF
IM\$C_EUCJP	日本語 EUC	SS2(0x8E) + 0xA1 ~ 0xDF
IM\$C_SJIS	シフト JIS	0xA1 ~ 0xDF

表示文字は最大 2 バイトなので、アプリケーションは "ch" のために、終端のヌル文字を含めて最低 3 バイト用意しなければなりません。

MC GET CHARACTER

戻される条件値

シンボル	重大度	メッセージ
IM_ _SUCCESS	Success	Normal successful completion.
IM_ _NOPRINTABLE	Information	The keycode is not for a key of printable character.
IM_ _INVKEYCODE	Error	Invalid keycode
IM_ _INVSTRDES	Error	Invalid string descriptor.
IM_ _FATERRLIB	Fatal	Fatal library internal error.
IM_ _INVCODESET	Error	Invalid Codeset Specified.

INIT KEY STATE

キーの STATE を初期化します。STATE は "initial"に移ります。

VMS バインディング

```
status=IM$INIT_KEY_STATE(unit_id)
```

引数

unit_id

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	read only
受け渡し方	by reference

OPEN PROFILE によって返された整数値。

C バインディング

```
status=ImInitKeyState(unit_id)
```

戻り値

INIT KEY STATE は条件値を返します。

INIT KEY STATE

引数

int unit_id (Input)

OPEN PROFILE によって返された整数値。

説明

INIT KEY STATE は、キーの STATE を初期化します。状態はシステム定義の STATE である "initial" に移ります。"initial" は SET KEYBIND が行われたときの最初の STATE です。

戻される条件値

シンボル	重大度	メッセージ
IM_ _SUCCESS	Success	Normal successful completion.
IM_ _ILLUNIT	Error	Illegal unit ID.
IM_ _NOKEYBIND	Error	Keybind data is not set yet. Execute SET KEYBIND first.

ENCODE KEY

通常の文字，数字キーに対応するコード，またはファンクション・キーやキーパッド・キーが発生するエスケープ・シーケンスから KEYCODE を作ります。

VMS バインディング

status = IM\$ENCODE_KEY(*string*, *keycode*)

戻り値

OpenVMS 用法	cond_value
データ型	longword (unsigned)
アクセス	write_only
受け渡し方	by value

引数

string

OpenVMS 用法	char string
データ型	char string
アクセス	read only
受け渡し方	by descriptor

文字列。単純な表示文字に対応するコードまたはエスケープ・シーケンスが指定できます。

keycode

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	write only
受け渡し方	by reference

ENCODE KEY

表示文字またはエスケープ・シーケンスに対応する KEYCODE。

C バインディング

```
keycode=ImEncodeKey(string, length)
```

戻り値

整数の KEYCODE の値を返します。エラーの場合は ENCODE KEY は 0 を返します。

引数

char *string (Input)

文字列。単純な表示文字に対応するコードまたはエスケープ・シーケンスが指定できます。

int length (Input)

第 1 パラメータの文字列の長さ。

説明

ENCODE KEY は、表示文字のコードまたはキーボードのキーが発生するエスケープ・シーケンスを、KEYCODE に変換します。エスケープ・シーケンスが正しくなかったり、サポートされていないときにはエラーとなります。

キーは 1 つずつ渡さなければなりません。2 つ以上のキーを同時に渡すとエラーとなります。サポートされている表示文字コードは、ASCII コードと半角カナコードです。ファンクション・キーなどのようにエスケープ・シーケンスが発生するキ

ーは、エスケープ・シーケンス全体を一度に ENCODE KEY に渡す必要があります。

戻される条件値

シンボル	重大度	メッセージ
IM__SUCCESS	Success	Normal successful completion.
IM__ILLESCAPE	Error	Illegal escape sequence.
IM__INVSTRDES	Error	Invalid string descriptor.

MC ENCODE KEY

通常の文字，数字キーに対応するコード，またはファンクション・キーやキーボード・キーが発生するエスケープ・シーケンスから KEYCODE を作ります。

VMS バインディング

status = IM\$MC_ENCODE_KEY(*string*, *keycode*, *codeset*)

戻り値

OpenVMS 用法	cond_value
データ型	longword (unsigned)
アクセス	write_only
受け渡し方	by value

引数

string

OpenVMS 用法	char string
データ型	char string
アクセス	read only
受け渡し方	by descriptor

文字列。単純な表示文字に対応するコードまたはエスケープ・シーケンスが指定できます。

keycode

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	write only
受け渡し方	by reference

表示文字またはエスケープ・シーケンスに対応する KEYCODE。

codeset

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	read only
受け渡し方	by reference

アプリケーションが使用しているコードセット。

C バインディング

keycode=ImMCEncodeKey(string, length, codeset)

戻り値

整数の KEYCODE の値を返します。エラーの場合は ENCODE KEY は 0 を返します。

引数

char *string (Input)

文字列。単純な表示文字に対応するコードまたはエスケープ・シーケンスが指定できます。

int length (Input)

第 1 パラメータの文字列の長さ。

int codeset (Input)

アプリケーションが使用しているコードセット。

 説明

MC ENCODE KEY は、表示文字のコードまたはキーボードのキーが発生するエスケープ・シーケンスを、KEYCODE に変換します。エスケープ・シーケンスが正しくなかったり、サポートされていないときにはエラーとなります。

MC ENCODE KEY は、コードセットによって異なる半角カナのコードに対応しています。コードセットとして指定できるシンボルは次のとおりです。

シンボル	コードセット	半角カナコード
IM\$C_DECKANJI	DEC 漢字	0xA1 ~ 0xDF
IM\$C_SDECKANJI	SuperDEC 漢字	SS2(0x8E) + 0xA1 ~ 0xDF
IM\$C_EUCJP	日本語 EUC	SS2(0x8E) + 0xA1 ~ 0xDF
IM\$C_SJIS	シフト JIS	0xA1 ~ 0xDF

半角カナコード以外の入力に対する出力は ENCODE_KEY と同じです。

 戻される条件値

シンボル	重大度	メッセージ
IM_ _SUCCESS	Success	Normal successful completion.
IM_ _ILLESCAPE	Error	Illegal escape sequence.
IM_ _INVSTRDES	Error	Invalid string descriptor.
IM_ _INVCODESET	Error	Invalid string descriptor.

KEYSYM TO KEYCODE

DECwindows の KEYSYM の値を対応する KEYCODE に変換します。

VMS バインディング

status = IM\$KEYSYM_TO_KEYCODE(*keysym*, *modifier*, *keycode*)

戻り値

OpenVMS 用法	cond_value
データ型	longword (unsigned)
アクセス	write_only
受け渡し方	by value

引数

keysym

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	read only
受け渡し方	by reference

DECwindows の KEYSYM の値。

modifier

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	read only
受け渡し方	by reference

モディファイア・キーを示すビット列。

KEYSYM TO KEYCODE

keycode

OpenVMS 用法	longword_unsigned
データ型	longword_integer (unsigned)
アクセス	write only
受け渡し方	by reference

KEYCODE を示す整数値。

C バインディング

keycode=ImKeysymToKeycode(keysym, modifier)

戻り値

KEYSYM に対応する KEYCODE を返します。KEYSYM に対応する KEYCODE がないときは 0 を返します。

引数

unsigned int keysym (Input)

DECwindows の KEYSYM の値。

unsigned int modifier (Input)

モディファイア・キーを示すビット列。

説明

KEYSYM TO KEYCODE は、DECwindows の環境で使われる KEYSYM の値を、IMLIB の KEYCODE に変換します。コントロール・キーのようなモディファイア・キーは、パラメータで指定します。

すべてのモディファイアがサポートされているわけではありませんが、サポートされていないキーが押されたことを示すために、モディファイアは必ず指定してください。

モディファイアを指定するためには以下に示す定数値を使います。

- IM_M_MOD_SHIFT
- IM_M_MOD_LOCK
- IM_M_MOD_CTRL
- IM_M_MOD_MOD_1 (ALT)
- IM_M_MOD_MOD_2 (カナ LOCK)

モディファイアのビット列は、XLookupString によって返されるビット列と同じなので、DECwindows のアプリケーションは XLookupString が返す値をそのまま KEYSYM TO KEYCODE に渡すことができます。

KEYSYM TO KEYCODE がサポートする KEYSYM の値は、付録 A を参照してください。

戻される条件値

シンボル	重大度	メッセージ
IM_SUCCESS	Success	Normal successful completion.
IM_UNSUPKEYSYM	Error	Unsupported keysym.

この章は、IMLIB を使ったプログラムの開発について書かれています。IMLIB は、C および FORTRAN によるアプリケーションプログラムの開発をサポートしています。

6.1 ヘッダ・ファイル

IMLIB は、エラー・シンボル、KEY のシンボル、ACTION のシンボルを定義した以下のようなヘッダ・ファイルを提供しています。

SYSSLIBRARY:IM\$DEF.H (C のヘッダファイル)
SYSSLIBRARY:IM\$DEF.FOR (FORTRAN のヘッダファイル)

アプリケーションは、ソース・ファイルの中にヘッダ・ファイルを次のような形で取り込みます。

例：ヘッダ・ファイルの取り込み (C の場合)

```
.  
. .  
. .  
#include <IM$DEF.H>  
. .  
. .  
. .
```

例：ヘッダ・ファイルの取り込み (FORTRAN の場合)

```
.  
.   
.   
INCLUDE 'SYS$LIBRARY:IM$DEF.FOR'  
.   
.   
.
```

6.2 コンパイル・リンクの方法

ここでは、作成されたアプリケーション・プログラムの、コンパイルおよびリンクについて説明します。IMLIB は、シェアラブル・ライブラリの形式で次の場所に提供されています。

SYSS\$SHARE:IM\$SHR.EXE

6.2.1 C で書かれたアプリケーション・プログラム

C 言語で書かれたアプリケーション・プログラムが、現在のディレクトリに APPLICATION.C という名前で置かれている場合に、コンパイルおよびリンクは以下のように行われます。通常 APPLICATION.C は日本語ライブラリ (JSYSHR) を使っていますので、リンクのオプションには JSYSHR.EXE が含まれます。

なお、OpenVMS AXP オペレーティング・システムを利用している場合は、SYSS\$SHARE:VAXCRTL.EXE をリンクする必要はありません。

```
$ CC APPLICATION.C  
$ LINK APPLICATION,SY$INPUT/OPTION  
SY$SHARE:VAXCTRL/SHARE  
SY$SHARE:IM$SHR/SHARE  
SY$SHARE:JSYSHR/SHARE  
[CTRL/Z]  
$
```

この結果 APPLICATION.EXE という実行イメージが現在のディレクトリに作成されます。

6.2.2 FORTRAN で書かれたアプリケーション・プログラム

FORTRAN 言語で書かれたアプリケーション・プログラムが、現在のディレクトリに APPLICATION.FOR という名前で置かれているとするとコンパイルおよびリンクは以下のように行われます。通常 APPLICATION.FOR は、日本語ライブラリ (JSYSHR) を使っていますので、リンクのオプションには JSYSHR.EXE が含まれます。

```
$ FORTRAN APPLICATION.FOR  
$ LINK APPLICATION,SY$INPUT/OPTION  
SY$SHARE:IM$SHR/SHARE  
SY$SHARE:JSYSHR/SHARE  
[CTRL/Z]  
$
```

この結果 APPLICATION.EXE という実行イメージが現在のディレクトリに作成されます。

IMLIB がサポートする KEYSYM

KEYSYM TO KEYCODE がサポートする KEYSYM を表 A-1 に示します。

表 A-1 サポートされる KEYSYM

Byte3	Byte4 Range	説明	文字セット
000	032 — 126	ASCII の表示文字	Latin-1
004	161 — 223	すべての半角カナ文字	Kana
255	009	TAB	Keyboard
255	013	RETURN	Keyboard
255	081	LEFT	Keyboard
255	082	UP	Keyboard
255	083	RIGHT	Keyboard
255	084	DOWN	Keyboard
255	085	PREVIOUS	Keyboard
255	086	NEXT	Keyboard
255	096	SELECT	Keyboard
255	098	DO	Keyboard
255	099	INSERT HERE	Keyboard
255	104	FIND	Keyboard
255	106	HELP	Keyboard
255	141	KEYPAD ENTER	Keyboard
255	145	KEYPAD PF1	Keyboard
255	146	KEYPAD PF2	Keyboard
255	147	KEYPAD PF3	Keyboard

(次ページに続く)

表 A-1 (続き) サポートされる KEYSYM

Byte3	Byte4 Range	説明	文字セット
255	148	KEYPAD PF4	Keyboard
255	172	KEYPAD COMMA	Keyboard
255	173	KEYPAD MINUS SIGN	Keyboard
255	174	KEYPAD DECIMAL POINT	Keyboard
255	176	KEYPAD DIGIT ZERO	Keyboard
255	177	KEYPAD DIGIT ONE	Keyboard
255	178	KEYPAD DIGIT TWO	Keyboard
255	179	KEYPAD DIGIT THREE	Keyboard
255	180	KEYPAD DIGIT FOUR	Keyboard
255	181	KEYPAD DIGIT FIVE	Keyboard
255	182	KEYPAD DIGIT SIX	Keyboard
255	183	KEYPAD DIGIT SEVEN	Keyboard
255	184	KEYPAD DIGIT EIGHT	Keyboard
255	185	KEYPAD DIGIT NINE	Keyboard
255	190	F1	Keyboard
255	191	F2	Keyboard
255	192	F3	Keyboard
255	193	F4	Keyboard
255	194	F5	Keyboard
255	195	F6	Keyboard
255	196	F7	Keyboard
255	197	F8	Keyboard
255	198	F9	Keyboard
255	199	F10	Keyboard
255	200	F11	Keyboard
255	201	F12	Keyboard
255	202	F13	Keyboard
255	203	F14	Keyboard

(次ページに続く)

表 A-1 (続き) サポートされる KEYSYM

Byte3	Byte4 Range	説明	文字セット
255	206	F17	Keyboard
255	207	F18	Keyboard
255	208	F19	Keyboard
255	209	F20	Keyboard
255	255	DELETE	Keyboard
255	33	Kanji	Keyboard
255	34	Muhenkan	Keyboard
255	37	HiraganaHenkan	Keyboard

定義済のシンボル

IMLIB が定義するシンボルは、以下のヘッダ・ファイルとして提供されています。

- ・ SYSSLIBRARY:IM\$DEF.H (C 用のヘッダ・ファイル)
- ・ SYSSLIBRARY:IM\$DEF.FOR (FORTRAN 用のヘッダ・ファイル)

B.1 エラー・シンボル

ライブラリが返すエラーのシンボルを表 B-1 に示します。

表 B-1 エラー・シンボル

シンボル	意味
IM__SUCCESS	正常終了
IM__LASTACTION	複数 ACTION の最後の ACTION である
IM__NOPRINTABLE	表示される文字でない
IM__PARTIALKEY	複数キーの途中である
IM__KEYNOTDEFINED	キーが定義されていない
IM__NOACTION	指定されたキーに ACTION が存在しない
IM__TRUNCATED	文字列が切り捨てられた

(次ページに続く)

表 B-1 (続き) エラー・シンボル

シンボル	意味
IM__NORECOVER	リカバー STATE がない
IM__SYNTAXERR	構文エラー
IM__INSVIRMEM	仮想メモリが足りない
IM__CANNOTOPN	ファイルがオープンできない
IM__ILLUNIT	正しくない UNIT 番号が指定された
IM__NOINDEX	PROFILE ファイルに INDEX が存在しない
IM__ILLFORMAT	KEYBIND ファイルのフォーマットが間違っている
IM__FILNOTFND	ファイルが見つからない
IM__NOKEYBIND	KEYBIND のデータがセットされていない
IM__NOTKEYNAME	KEY の名前を示す文字列が間違っている
IM__INVKEYCODE	KEYCODE が正しくない
IM__ILLESCAPE	エスケープ・シーケンスが正しくない
IM__UNSUPKEYSYM	サポートされない KEYSYM が指定された
IM__READERR	ファイルを読み込めない
IM__WRITEERR	ファイルに書き込みができない
IM__KEYNOTSET	GET ACTION の前に SET KEY が呼ばれていない
IM__INVSTRDES	文字列デスク립タが正しくない
IM__ILLLEVEL	KEYBIND ファイルのレベルが合わない
IM__FATERRLIB	ライブラリの致命的なエラー
IM__INVCODESET	不正なコードセットが指定された

B.2 ACTION を示すシンボル

ライブラリがサポートする ACTION のシンボルを表 B-2 に示します。

表 B-2 ACTION を示すシンボル

ACTION	シンボル
CONVERT	IM_ACTION_CONVERT
HIRAGANA	IM_ACTION_HIRAGANA
KATAKANA	IM_ACTION_KATAKANA
HANKAKU_KANA	IM_ACTION_HANKAKU_KANA
ZENKAKU	IM_ACTION_ZENKAKU
HANKAKU	IM_ACTION_HANKAKU
UPPER	IM_ACTION_UPPER
LOWER	IM_ACTION_LOWER
CLA_HIRAGANA	IM_ACTION_CLA_HIRAGANA
CLA_KATAKANA	IM_ACTION_CLA_KATAKANA
CLA_HANKAKU_KANA	IM_ACTION_CLA_HANKAKU_KANA
CLA_ZENKAKU	IM_ACTION_CLA_ZENKAKU
CLA_HANKAKU	IM_ACTION_CLA_HANKAKU
NEXT_CLAUSE	IM_ACTION_NEXT_CLAUSE
PREV_CLAUSE	IM_ACTION_PREV_CLAUSE
SHORTEN_CLAUSE	IM_ACTION_SHORTEN_CLAUSE
EXTEND_CLAUSE	IM_ACTION_EXTEND_CLAUSE
ECHO	IM_ACTION_ECHO
MOVE_LEFT	IM_ACTION_MOVE_LEFT
MOVE_RIGHT	IM_ACTION_MOVE_RIGHT
DELETE	IM_ACTION_DELETE
RESTORE_STRING	IM_ACTION_RESTORE_STRING
RESTORE_ECHO	IM_ACTION_RESTORE_ECHO
NONE	IM_ACTION_NONE
START	IM_ACTION_START
DONE	IM_ACTION_DONE
NEXT_CANDIDATE	IM_ACTION_NEXT_CANDIDATE
PREV_CANDIDATE	IM_ACTION_PREV_CANDIDATE

(次ページに続く)

表 B-2 (続き) ACTION を示すシンボル

ACTION	シンボル
HEAD	IM_ACTION_HEAD
TAIL	IM_ACTION_TAIL
SYMBOL	IM_ACTION_SYMBOL
DEC_KANJI_CODE	IM_ACTION_DEC_KANJI_CODE
START_SELECTED	IM_ACTION_START_SELECTED

B.3 KEY を示すシンボル

ライブラリがサポートする KEY のシンボルを表 B-3 に示します。

表 B-3 KEY を示すシンボル

KEY	シンボル
NULL	IM_KEY_NULL
CTRL_A	IM_KEY_CTRL_A
CTRL_B	IM_KEY_CTRL_B
CTRL_C	IM_KEY_CTRL_C
CTRL_D	IM_KEY_CTRL_D
CTRL_E	IM_KEY_CTRL_E
CTRL_F	IM_KEY_CTRL_F
CTRL_G	IM_KEY_CTRL_G
CTRL_H	IM_KEY_CTRL_H
TAB	IM_KEY_CTRL_I

(次ページに続く)

表 B-3 (続き) KEY を示すシンボル

KEY	シンボル
CTRL_J	IM_KEY_CTRL_J
CTRL_K	IM_KEY_CTRL_K
CTRL_L	IM_KEY_CTRL_L
RETURN	IM_KEY_CTRL_M
CTRL_N	IM_KEY_CTRL_N
CTRL_O	IM_KEY_CTRL_O
CTRL_P	IM_KEY_CTRL_P
CTRL_Q	IM_KEY_CTRL_Q
CTRL_R	IM_KEY_CTRL_R
CTRL_S	IM_KEY_CTRL_S
CTRL_T	IM_KEY_CTRL_T
CTRL_U	IM_KEY_CTRL_U
CTRL_V	IM_KEY_CTRL_V
CTRL_W	IM_KEY_CTRL_W
CTRL_X	IM_KEY_CTRL_X
CTRL_Y	IM_KEY_CTRL_Y
CTRL_Z	IM_KEY_CTRL_Z
ESC	IM_KEY_ESC
FS	IM_KEY_FS
GS	IM_KEY_GS
RS	IM_KEY_RS
US	IM_KEY_US
DEL	IM_KEY_DEL
F6	IM_KEY_F6
F7	IM_KEY_F7
F8	IM_KEY_F8
F9	IM_KEY_F9
F10	IM_KEY_F10

(次ページに続く)

定義済のシンボル
B.3 KEY を示すシンボル

表 B-3 (続き) KEY を示すシンボル

KEY	シンボル
F11	IM_KEY_F11
F12	IM_KEY_F12
F13	IM_KEY_F13
F14	IM_KEY_F14
HELP	IM_KEY_HELP
DO	IM_KEY_DO
F17	IM_KEY_F17
F18	IM_KEY_F18
F19	IM_KEY_F19
F20	IM_KEY_F20
KP0	IM_KEY_KP0
KP1	IM_KEY_KP1
KP2	IM_KEY_KP2
KP3	IM_KEY_KP3
KP4	IM_KEY_KP4
KP5	IM_KEY_KP5
KP6	IM_KEY_KP6
KP7	IM_KEY_KP7
KP8	IM_KEY_KP8
KP9	IM_KEY_KP9
MINUS	IM_KEY_MINUS
COMMA	IM_KEY_COMMA
PERIOD	IM_KEY_PERIOD
ENTER	IM_KEY_ENTER
PF1	IM_KEY_PF1
PF2	IM_KEY_PF2
PF3	IM_KEY_PF3
PF4	IM_KEY_PF4

(次ページに続く)

表 B-3 (続き) KEY を示すシンボル

KEY	シンボル
LEFT	IM_KEY_LEFT
UP	IM_KEY_UP
RIGHT	IM_KEY_RIGHT
DOWN	IM_KEY_DOWN
PREV_SCREEN	IM_KEY_PREV_SCREEN
NEXT_SCREEN	IM_KEY_NEXT_SCREEN
SELECT	IM_KEY_SELECT
INSERT_HERE	IM_KEY_INSERT_HERE
FIND	IM_KEY_FIND
REMOVE	IM_KEY_REMOVE
SHIFT + <input type="checkbox"/> キー (前文節移動)	IM_KEY_SHFT_UP
SHIFT + <input type="checkbox"/> キー (次文節移動)	IM_KEY_SHFT_DOWN
SHIFT + <input type="checkbox"/> キー (文節縮小)	IM_KEY_SHFT_LEFT
SHIFT + <input type="checkbox"/> キー (文節拡大)	IM_KEY_SHFT_RIGHT
CTRL + <input type="checkbox"/> キー	IM_KEY_CTRL_UP
CTRL + <input type="checkbox"/> キー	IM_KEY_CTRL_DOWN
CTRL + <input type="checkbox"/> キー	IM_KEY_CTRL_LEFT
CTRL + <input type="checkbox"/> キー	IM_KEY_CTRL_RIGHT
ALT + <input type="checkbox"/> キー	IM_KEY_ALT_UP
ALT + <input type="checkbox"/> キー	IM_KEY_ALT_DOWN
ALT + <input type="checkbox"/> キー	IM_KEY_ALT_LEFT
ALT + <input type="checkbox"/> キー	IM_KEY_ALT_RIGHT
CTRL + ALT + <input type="checkbox"/> キー	IM_KEY_CTRL_ALT_UP
CTRL + ALT + <input type="checkbox"/> キー	IM_KEY_CTRL_ALT_DOWN
CTRL + ALT + <input type="checkbox"/> キー	IM_KEY_CTRL_ALT_LEFT
CTRL + ALT + <input type="checkbox"/> キー	IM_KEY_CTRL_ALT_RIGHT
CTRL + SHIFT + <input type="checkbox"/> キー	IM_KEY_CTRL_SHFT_UP
CTRL + SHIFT + <input type="checkbox"/> キー	IM_KEY_CTRL_SHFT_DOWN

(次ページに続く)

定義済のシンボル
B.3 KEY を示すシンボル

表 B-3 (続き) KEY を示すシンボル

KEY	シンボル
CTRL + SHIFT + □キー	IM_KEY_CTRL_SHFT_LEFT
CTRL + SHIFT + □キー	IM_KEY_CTRL_SHFT_RIGHT
SHIFT + ALT + □キー	IM_KEY_SHFT_ALT_UP
SHIFT + ALT + □キー	IM_KEY_SHFT_ALT_DOWN
SHIFT + ALT + □キー	IM_KEY_SHFT_ALT_LEFT
SHIFT + ALT + □キー	IM_KEY_SHFT_ALT_RIGHT
CTRL + SHIFT + ALT + □キー	IM_KEY_CTRL_SHFT_ALT_UP
CTRL + SHIFT + ALT + □キー	IM_KEY_CTRL_SHFT_ALT_DOWN
CTRL + SHIFT + ALT + □キー	IM_KEY_CTRL_SHFT_ALT_LEFT
CTRL + SHIFT + ALT + □キー	IM_KEY_CTRL_SHFT_ALT_RIGHT
変換	IM_JFK_HENKAN
SHIFT + 変換 (前候補)	IM_JFK_SHFT_MAEKOUHO
CTRL + 変換	IM_JFK_CTRL_HENKAN
ALT + 変換	IM_JFK_ALT_HENKAN
CTRL + SHIFT + 変換	IM_JFK_CTRL_SHFT_HENKAN
CTRL + ALT + 変換	IM_JFK_CTRL_ALT_HENKAN
CTRL + SHIFT + ALT + 変換	IM_JFK_CTRL_SHFT_ALT_HENKAN
無変換	IM_JFK_MUHENKAN
SHIFT + 無変換 (入力状態へ戻る)	IM_JFK_SHFT_MUHENKAN
CTRL + 無変換	IM_JFK_CTRL_MUHENKAN
ALT + 無変換 (記号変換)	IM_JFK_ALT_MUHENKAN
CTRL + SHIFT + 無変換	IM_JFK_CTRL_SHFT_MUHENKAN
CTRL + ALT + 無変換	IM_JFK_CTRL_ALT_MUHENKAN
CTRL + SHIFT + ALT + 無変換	IM_JFK_CTRL_SHFT_ALT_MUHENKAN
ひらがな変換	IM_JFK_HIRAGANA
SHIFT + ひらがな変換 (カタカナ変換)	IM_JFK_KATAKANA
CTRL + ひらがな変換 (半角カナ変換)	IM_JFK_CTRL_HIRAGANA
ALT + ひらがな変換	IM_JFK_ALT_HIRAGANA

(次ページに続く)

表 B-3 (続き) KEY を示すシンボル

KEY	シンボル
CTRL + SHIFT + ひらがな変換 (半角カナ変換)	IM_JFK_CTRL_SHFT_HIRAGANA
CTRL + ALT + ひらがな変換	IM_JFK_CTRL_ALT_HIRAGANA
CTRL + SHIFT + ALT + ひらがな変換	IM_JFK_CTRL_SHFT_ALT_HIRAGANA
SS3	IM_KEY_SS3
CSI	IM_KEY_CSI
VOID	IM_KEY_VOID
TYPING_KEYS	IM_KEY_TYPING_KEYS
OTHERS	IM_KEY_OTHERS

B.4 Key Modifier を示すシンボル

ライブラリがサポートする Key Modifier のシンボルを表 B-4 に示します。

表 B-4 Key Modifier を示すシンボル

Modifier	シンボル
SHIFT	IM_M_MOD_SHIFT
LOCK	IM_M_MOD_LOCK
CTRL	IM_M_MOD_CTRL
MOD_1 (ALT)	IM_M_MOD_1
MOD_2 (Kana)	IM_M_MOD_2
MOD_3	IM_M_MOD_3
MOD_4	IM_M_MOD_4

(次ページに続く)

定義済のシンボル
B.4 Key Modifier を示すシンボル

表 B-4 (続き) Key Modifier を示すシンボル

Modifier	シンボル
MOD_5	IM_M_MOD_5

A

ACTION	3-1
C バインディング	5-3
VMS バインディング	5-3
動作	4-4

C

CLA_HANKAKU	3-4, 4-7
CLA_HANKAKU_KANA	3-4
CLA_HIRAGANA	3-3, 4-7
CLA_KATAKANA	3-3, 4-7
CLA_ZENKAKU	3-4, 4-7
CLOSE PROFILE	5-11
CONVERT	3-3, 4-6

D

DECwindows	2-7
DELETE	3-2, 4-9
DONE	3-4, 4-10

E

ECHO	3-1, 4-5
ENCODE KEY	5-45, 5-48
EXTEND_CLAUSE	3-4, 4-7

G

GET ACTION	5-28
GET CHARACTER	5-35
GET KEY ACTION	3-1, 5-23
GET KEYCODE	5-32

GET PROFILE DATA	1-2, 5-16
------------------	-----------

H

HANKAKU	3-2, 4-5
HANKAKU_KANA	3-2
HEAD	3-2, 4-8
HIRAGANA	3-2, 4-5

I

IMLIB	
アプリケーションとの関係	2-3
機能	1-1
終了	2-2
初期化	2-1
IMLIB ライブラリ・ルーチン	5-1
IMLIB ルーチン	5-1, 5-5
INDEX	1-2
INIT KEY STATE	5-43

K

KATAKANA	3-2, 4-5
KEYBIND	
情報の設定	5-3
設定	2-2
KEYCODE	5-4
KEYSYM	5-5, A-1
KEYSYM TO KEYCODE	5-51

L

LOWER 3-3, 4-5

M

MC GET CHARACTER 5-38

MOVE_LEFT 3-2, 4-8

MOVE_RIGHT 3-2, 4-8

N

NEXT_CANDIDATE 3-3, 4-7

NEXT_CLAUSE 3-4, 4-7

NONE 3-5, 4-12

O

OPEN PROFILE 5-6

OpenVMS 2-7

P

PREV_CANDIDATE 4-7

PREV_CANDITATE 3-3

PREV_CLAUSE 3-4, 4-7

PROFILE 1-2, 5-1

 オープン 2-1, 5-1

 書き込み 2-2

 クローズ 2-2, 5-2

 情報の読み込み 2-1

 データの書き込み 5-2

 データの検索 5-2

 データの変更 5-2

 変更 2-2

Q

QIO 2-7

R

RECOVER KEY STATE 5-31

RESTORE_ECHO 3-5, 4-11

RESTORE_STRING 3-4, 4-10

S

SET KEY 5-25

SET KEYBIND 5-19

SET PROFILE DATA 5-13

SHORTEN_CLAUSE 3-4

START 3-1, 4-4

START_SELECTED 3-1, 4-5

SYMBOL 3-3, 4-5

T

TAIL 3-2, 4-8

U

UPPER 3-3, 4-5

W

WRITE PROFILE 5-8

Z

ZENKAKU 3-2, 4-5

ア

アプリケーション 2-1, 4-1

オ

オーバーストライク・モード 2-10

カ

カーソル	4-12
かな漢字変換	
キー定義	1-1
内部状態	4-1
入力インターフェイス	2-4
かな変換状態	4-3
漢字変換状態	4-3

キ

キー・エコー・バッファ	2-8, 2-12, 4-1
キー入力	
処理	2-2
バッファ	2-8, 2-12, 4-1
モジュール	2-7

シ

自動ローマ字かな変換	2-12
初期状態	4-3

ニ

入力状態	4-3
入力文字列編集	2-8

ヒ

表示バッファ	2-8, 4-1
--------	----------

フ

プログラミング言語	5-1
文節操作	4-12

モ

文字列バッファ	2-8
---------	-----

ユ

ユーザ・キー定義	1-1
----------	-----

ロ

ローマ字かな変換	2-7
----------	-----

日本語 OpenVMS
IMLIB/OpenVMS ライブラリ・リファレンス・マニュアル

1999 年 4 月 発行

コンパックコンピュータ株式会社

〒 140-8641 東京都品川区東品川 2-2-24 天王洲セントラルタワー

電話 (03)5463-6600 (大代表)

AA-PU8TE-TE

