

# 日本語 OpenVMS

---

## 日本語ライブラリ 利用者の手引き

AA-PU8NF-TE.2

2009年11月

本書は、日本語データを取り扱うための基本的な機能を提供する日本語ライブラリについての説明書です。

改訂 / 更新情報:	日本語 OpenVMS V7.3 『日本語ライブラリ利用者の手引き』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 V7.3-2 日本語 OpenVMS VAX V7.3

日本ヒューレット・パッカート株式会社

---

© 2004, 2009 Hewlett-Packard Development Company, L.P.

本書の著作権は日本ヒューレット・パッカート株式会社が保有しており、本書中の解説および図、表は日本ヒューレット・パッカート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、日本ヒューレット・パッカート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。

日本ヒューレット・パッカートは、日本ヒューレット・パッカートまたは日本ヒューレット・パッカートの指定する会社から納入された機器以外の機器で対象ソフトウェアを使用した場合、その性能あるいは信頼性について一切責任を負いかねます。

以下は、他社の商標です。

Adobe, Adobe Illustrator, POSTSCRIPT は米国 Adobe Systems 社の商標です。

BITSTREAM は米国 Bitstream 社の商標です。

Microsoft, MS および MS-DOS は米国 Microsoft 社の商標です。

Motif, OSF, OSF/1, OSF/Motif および Open Software Foundation は米国 Open Software Foundation 社の商標です。

その他のすべての商標および登録商標は、それぞれの所有者が保有しています。

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	ix
1 概要	
1.1 日本語ライブラリの構成	1-1
1.2 日本語ライブラリのリンク方法	1-1
1.2.1 漢字コード変換ライブラリ以外の日本語ライブラリをリンクする方 法	1-2
1.2.2 漢字コード変換ライブラリを含むすべての日本語ライブラリをリンクす る方法	1-2
1.2.3 漢字コード変換ライブラリのみをリンクする方法	1-2
2 日本語ライブラリの使用例	
2.1 FORTRAN における日本語の使用例	2-1
2.2 C における日本語の使用例	2-2
2.3 MACRO における日本語の使用例	2-2
3 汎用ライブラリ	
3.1 汎用ライブラリ・ルーチン一覧	3-1
3.2 日本語文字列操作ルーチン	3-3
JLB\$NOF_BYTE	3-4
JLB\$NOF_CHAR	3-6
JLB\$POS_CURR	3-7
JLB\$POS_NEXT	3-8
JLB\$POS_PREV	3-10
JLB\$POSITION	3-11
JLB\$STR_EQUAL	3-12
JLB\$STR_START	3-14
JLB\$STR_SEARCH	3-16
JLB\$TRIM	3-18
JLB\$TRUNC	3-20
JLB\$LOCC	3-22
JLB\$SKPC	3-24
JLB\$RCHAR	3-25
JLB\$RNEXT	3-26
JLB\$WCHAR	3-27
JLB\$WNEXT	3-28

3.3	文字列変換ルーチン	3-29
	JLB\$TRA_ROM_HALF	3-30
	JLB\$TRA_ROM_FULL	3-31
	JLB\$TRA_ROM_SIZE	3-32
	JLB\$TRA_ROM_LOWER	3-33
	JLB\$TRA_ROM_UPPER	3-34
	JLB\$TRA_ROM_CASE	3-35
	JLB\$TRA_ROM_KANA	3-36
	JLB\$TRA_KANA_HIRA	3-39
	JLB\$TRA_KANA_KATA	3-40
	JLB\$TRA_KANA_KANA	3-41
	JLB\$TRA_KANA_DAKU	3-42
	JLB\$TRA_KANA_HALF	3-43
	JLB\$TRA_KANA_FULL	3-45
	JLB\$TRA_SYMBOL	3-47
3.4	漢字変換入力ルーチン	3-49
	JLB\$GET_COMMAND	3-50
	JLB\$GET_INPUT	3-52
3.5	日本語ファイル名ルーチン	3-54
	JLB\$RMS_USER_VTF7	3-55
	JLB\$RMS_VTF7_USER	3-57
	JSY\$RMS_GET_ENCODING	3-59
	JSY\$RMS_LIST_ENCODING	3-60
	JSY\$RMS_SET_ENCODING	3-62
	JSY\$RMS_USER_VTF7	3-64
	JSY\$RMS_VTF7_USER	3-66
3.6	その他のルーチン	3-68
	JLB\$DATE_TIME	3-69
	JLB\$DEV_KANJI	3-72
	JLB\$DEV_KANJI_IN	3-73
4	基本ライブラリ	
4.1	基本ライブラリ・ルーチン一覧	4-1
4.2	日本語文字列操作ルーチン	4-3
	JSY\$CH_NEXT	4-4
	JSY\$CH_PREV	4-6
	JSY\$CH_CURR	4-7
	JSY\$POS_NEXT	4-8
	JSY\$POS_PREV	4-9
	JSY\$POS_CURR	4-11
	JSY\$CH_RCHAR	4-12
	JSY\$CH_RNEXT	4-13
	JSY\$CH_WCHAR	4-14
	JSY\$CH_WNEXT	4-15
	JSY\$CH_GCHAR	4-16

	JSY\$CH_GNEXT .....	4-17
	JSY\$CH_PCHAR .....	4-18
	JSY\$CH_PNEXT .....	4-19
	JSY\$CH_SIZE .....	4-21
	JSY\$CH_NBYTE .....	4-22
	JSY\$CH_NCHAR .....	4-23
	JSY\$POSITION .....	4-24
	JSY\$STR_EQUAL .....	4-25
	JSY\$STR_START .....	4-27
	JSY\$STR_SEARCH .....	4-29
	JSY\$TRIM .....	4-31
	JSY\$TRUNC .....	4-32
	JSY\$LOCC .....	4-33
	JSY\$SKPC .....	4-34
4.3	文字変換ルーチン .....	4-35
	JSY\$CHG_ROM_HALF .....	4-36
	JSY\$CHG_ROM_FULL .....	4-37
	JSY\$CHG_ROM_SIZE .....	4-38
	JSY\$CHG_ROM_LOWER .....	4-39
	JSY\$CHG_ROM_UPPER .....	4-40
	JSY\$CHG_ROM_CASE .....	4-41
	JSY\$CHG_KANA_HIRA .....	4-42
	JSY\$CHG_KANA_KATA .....	4-43
	JSY\$CHG_KANA_KANA .....	4-44
	JSY\$CHG_KANA_DAKU .....	4-45
	JSY\$CHG_KANA_HALF .....	4-46
	JSY\$CHG_KANA_FULL .....	4-47
	JSY\$CHG_GENERAL .....	4-48
	JSY\$CHG_JIS_KUTEN .....	4-49
	JSY\$CHG_KEISEN .....	4-50
4.4	文字列変換ルーチン .....	4-51
	JSY\$TRA_ROM_HALF .....	4-52
	JSY\$TRA_ROM_FULL .....	4-53
	JSY\$TRA_ROM_SIZE .....	4-54
	JSY\$TRA_ROM_LOWER .....	4-55
	JSY\$TRA_ROM_UPPER .....	4-56
	JSY\$TRA_ROM_CASE .....	4-57
	JSY\$TRA_ROM_KANA .....	4-58
	JSY\$TRA_KANA_HIRA .....	4-60
	JSY\$TRA_KANA_KATA .....	4-61
	JSY\$TRA_KANA_KANA .....	4-62
	JSY\$TRA_KANA_DAKU .....	4-63
	JSY\$TRA_KANA_HALF .....	4-64
	JSY\$TRA_KANA_FULL .....	4-65
	JSY\$TRA_SYMBOL .....	4-67

## 5 かな漢字変換ライブラリ

5.1	かな漢字変換ライブラリ・ルーチン一覧	5-1
5.2	複文節変換ルーチン	5-3
	JLB\$CNV_OPEN_DICTIONARY , JSY\$CNV_OPEN_DICTIONARY	5-4
	JLB\$CNV_CLOSE_DICTIONARY , JSY\$CNV_CLOSE_DICTIONARY	5-6
	JLB\$CNV_CONVERT , JSY\$CNV_CONVERT	5-7
	JLB\$CNV_ROM_CONVERT	5-9
	JLB\$CNV_GET_KANJI , JSY\$CNV_GET_KANJI	5-11
	JLB\$CNV_NEXT_WORD , JSY\$CNV_NEXT_WORD	5-13
	JLB\$CNV_PREV_WORD , JSY\$CNV_PREV_WORD	5-15
	JLB\$CNV_GET_CLAUSE , JSY\$CNV_GET_CLAUSE	5-17
	JSY\$CNV_GET_PHONETIC_CLAUSE	5-19
	JLB\$CNV_SHORTEN_CLAUSE , JSY\$CNV_SHORTEN_CLAUSE	5-21
	JLB\$CNV_EXTEND_CLAUSE , JSY\$CNV_EXTEND_CLAUSE	5-23
	JLB\$CNV_CLAUSE_HIRAGANA , JSY\$CNV_CLAUSE_HIRAGANA	5-25
	JLB\$CNV_CLAUSE_KATAKANA , JSY\$CNV_CLAUSE_KATAKANA	5-27
	JLB\$CNV_CLAUSE_FULL , JSY\$CNV_CLAUSE_FULL	5-29
	JLB\$CNV_CLAUSE_HALF , JSY\$CNV_CLAUSE_HALF	5-31
	JLB\$CNV_CLAUSE_SYMBOL , JSY\$CNV_CLAUSE_SYMBOL	5-33
	JLB\$CNV_CLAUSE_NOCONVERT , JSY\$CNV_CLAUSE_NOCONVERT	5-35
	JLB\$CNV_CLAUSE_DELETE , JSY\$CNV_CLAUSE_DELETE	5-37
	JLB\$CNV_LEARN , JSY\$CNV_LEARN	5-39
	JLB\$CNV_REGISTER_WORD , JSY\$CNV_REGISTER_WORD	5-41
	JLB\$CNV_DELETE_WORD , JSY\$CNV_DELETE_WORD	5-43
	JLB\$CNV_IO_ERROR , JSY\$CNV_IO_ERROR	5-45
5.3	単語単位変換ルーチン	5-48
	JLB\$TRA_DICINI , JSY\$TRA_DICINI	5-49
	JLB\$TRA_DICCLS , JSY\$TRA_DICCLS	5-50
	JLB\$ENT_TANGO , JSY\$ENT_TANGO	5-51
	JLB\$DEL_TANGO , JSY\$DEL_TANGO	5-53
	JLB\$TRA_ROM_TANGO	5-54
	JLB\$TRA_KANA_TANGO , JSY\$TRA_KANA_TANGO	5-56
	JLB\$TRA_TANGO_NEXT , JSY\$TRA_TANGO_NEXT	5-58
	JLB\$TRA_TANGO_PREV , JSY\$TRA_TANGO_PREV	5-60
	JLB\$TRA_TANGO_DONE , JSY\$TRA_TANGO_DONE	5-62
5.4	かな漢字変換辞書	5-63
5.4.1	システム辞書	5-63
5.4.2	個人辞書と文節学習辞書	5-63
5.4.3	個人辞書の使用モードの選択	5-64
5.4.4	個人辞書の学習モードの選択	5-65
5.4.5	個人辞書の単語数	5-66

5.5	注意事項および制限事項	5-67
5.5.1	変換の開始と終了	5-67
5.5.2	複文節変換と単語単位変換の混用	5-67
5.5.3	個人辞書を共用した場合の変換結果	5-68
5.5.4	辞書アクセス時のエラー	5-68
<b>6</b>	<b>漢字コード変換ライブラリ</b>	
6.1	漢字コード変換ライブラリ・ルーチン一覧	6-2
6.2	漢字コード変換ルーチン	6-3
	JLB\$TRA_KANJI_DEC_JIS	6-4
	JLB\$TRA_KANJI_JIS_DEC	6-6
	JLB\$TRA_KANJI_DEC_HITACHI	6-8
	JLB\$TRA_KANJI_HITACHI_DEC	6-10
	JLB\$TRA_KANJI_DEC_IBM	6-12
	JLB\$TRA_KANJI_IBM_DEC	6-14
	JLB\$TRA_KANJI_DEC_JEF	6-16
	JLB\$TRA_KANJI_JEF_DEC	6-18
	JLB\$TRA_KANJI_DEC_NEC	6-20
	JLB\$TRA_KANJI_NEC_DEC	6-22
	JLB\$TRA_KANJI_DEC_MSDOS	6-24
	JLB\$TRA_KANJI_MSDOS_DEC	6-26
	JLB\$TRA_KANJI_DEC_CPM	6-28
	JLB\$TRA_KANJI_CPM_DEC	6-30
6.3	1バイト・コード変換テーブル	6-32
	JSY\$GTBL_TO_ASCII	6-33
	JSY\$GTBL_TO_ASCII_K	6-34
	JSY\$GTBL_TO_NEC	6-35
	JSY\$GTBL_TO_NEC_K	6-37
	JSY\$GTBL_TO_MSDOS	6-39
	JSY\$GTBL_TO_MSDOS_K	6-41
	JSY\$GTBL_TO_EBCDIK	6-43
	JSY\$GTBL_EBCDIK_TO_ASCII	6-45
6.4	DEC 漢字コード変換ライブラリ・ルーチン	6-46
	JLB\$KCV_BEGIN_CONV	6-47
	JLB\$KCV_END_CONV	6-49
	JLB\$KCV_FIND	6-50
	JLB\$KCV_CONVERT	6-52

## A 変換キー配列

## B 変換対応表

B.1	ローマ字/かな変換対応表	B-1
B.2	ローマ字半角/全角コード変換対応表	B-6
B.3	記号変換対応表	B-8
B.3.1	1文字変換	B-8
B.3.2	2文字変換	B-8
B.3.3	3文字変換	B-11
B.3.4	区点コード変換	B-11
B.3.5	16進コード変換	B-12

## 索引

### 図

6-1	JSY\$GTBL_TO_ASCII テーブル	6-33
6-2	JSY\$GTBL_TO_ASCII_K テーブル	6-34
6-3	JSY\$GTBL_TO_NEC テーブル	6-36
6-4	JSY\$GTBL_TO_NEC_K テーブル	6-38
6-5	JSY\$GTBL_TO_MSDOS テーブル	6-40
6-6	JSY\$GTBL_TO_MSDOS_K テーブル	6-42
6-7	JSY\$GTBL_TO_EBCDIK テーブル	6-44
6-8	JSY\$GTBL_EBCDIK_TO_ASCII テーブル	6-45
A-1	LK401-JJ, LK401BJ の変換キーの機能一覧	A-2
A-2	数字キーパッドを使用した変換キー配置	A-3
B-1	ローマ字/かな変換対応表	B-2
B-2	ローマ字/かな変換対応表(つづき)	B-3
B-3	特殊文字変換対応表	B-5
B-4	1文字変換	B-8
B-5	2文字変換	B-9
B-6	2文字変換(つづき)	B-10
B-7	3文字変換	B-11

### 表

A-1	コントロール・キーを使用した変換キー一覧	A-1
B-1	ローマ字半角/全角コード変換対応表	B-7
B-2	区点コード変換	B-11
B-3	16進コード変換	B-12



## 本書の目的

本書は、日本語 OpenVMS でサポートされるプログラミング言語を使用して日本語アプリケーションを作成するための、ライブラリ・ルーチンについて解説します。

## 対象読者

本書は、少なくとも1つのプログラミング言語を理解し、OpenVMS オペレーティング・システムとランタイム・ライブラリの概念を理解しているシステム・プログラマおよびアプリケーションのプログラマを対象としています。

## 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	日本語ライブラリ・ルーチンの概要について説明します。
第2章	日本語ライブラリ・ルーチンの簡単な使用例について説明します。
第3章	汎用ライブラリのルーチンについて説明します。
第4章	基本ライブラリのルーチンについて説明します。
第5章	かな漢字変換ライブラリのルーチンについて説明します。
第6章	漢字コード変換ライブラリのルーチンについて説明します。
付録 A	かな漢字変換入力ルーチン、および、それを利用している日本語ユーティリティにおける変換キーの配列について説明します。
付録 B	日本語ライブラリ、および、それを利用する日本語ユーティリティでの各種変換対応表を示します。

## 関連資料

- 『OpenVMS RTL Library (LIB\$) Manual』
- 『OpenVMS RTL General Purpose (OTS\$) Manual』
- 『OpenVMS RTL Parallel Processing (PPL\$) Manual』
- 『OpenVMS RTL Screen Management (SMG\$) Manual』
- 『OpenVMS RTL String Manipulation (STR\$) Manual』

- 『VMS RTL Mathematics (MTH\$) Manual』
- 『VMS RTL DECtalk (DTK\$) Manual』

## 表記法

本書では、次の表記法を使用します。

表記法	意味
Ctrl/x	Ctrl/xという表記は、Ctrl キーを押しながら別のキーまたはポインティング・デバイス・ボタンを押すことを示します。
PF1 x	PF1 xという表記は、PF1 に定義されたキーを押してから、別のキーまたはポインティング・デバイス・ボタンを押すことを示します。
<span style="border: 1px solid black; padding: 2px;">Return</span>	例の中で、キー名が四角で囲まれている場合には、キーボード上でそのキーを押すことを示します。テキストの中では、キー名は四角で囲まれていません。 HTML 形式のドキュメントでは、キー名は四角ではなく、括弧で囲まれています。
...	例の中の水平方向の反復記号は、次のいずれかを示します。 <ul style="list-style-type: none"> <li>• 文中のオプションの引数が省略されている。</li> <li>• 前出の 1 つまたは複数の項目を繰り返すことができる。</li> <li>• パラメータや値などの情報をさらに入力できる。</li> </ul>
.	垂直方向の反復記号は、コードの例やコマンド形式の中の項目が省略されていることを示します。このように項目が省略されるのは、その項目が説明している内容にとって重要ではないからです。
( )	コマンドの形式の説明において、括弧は、複数のオプションを選択した場合に、選択したオプションを括弧で囲まなければならないことを示しています。
[ ]	コマンドの形式の説明において、大括弧で囲まれた要素は任意のオプションです。オプションをすべて選択しても、いずれか 1 つを選択しても、あるいは 1 つも選択しなくても構いません。ただし、OpenVMS ファイル指定のディレクトリ名の構文や、割り当て文の部分文字列指定の構文の中では、大括弧に囲まれた要素は省略できません。
[   ]	コマンド形式の説明では、括弧内の要素を分けている垂直棒線はオプションを 1 つまたは複数選択するか、または何も選択しないことを意味します。
{ }	コマンドの形式の説明において、中括弧で囲まれた要素は必須オプションです。いずれか 1 のオプションを指定しなければなりません。
太字	太字のテキストは、新しい用語、引数、属性、条件を示しています。
<i>italic text</i>	イタリック体のテキストは、重要な情報を示します。また、システム・メッセージ (たとえば内部エラー <i>number</i> )、コマンド・ライン (たとえば <i>PRODUCER=name</i> )、コマンド・パラメータ (たとえば <i>device-name</i> ) などの変数を示す場合にも使用されます。
UPPERCASE TEXT	英大文字のテキストは、コマンド、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。

表記法	意味
Monospace type	<p>モノスペース・タイプの文字は、コード例および会話型の画面表示を示します。</p> <p>C プログラミング言語では、テキスト中のモノスペース・タイプの文字は、キーワード、別々にコンパイルされた外部関数およびファイルの名前、構文の要約、または例に示される変数または識別子への参照などを示します。</p>
–	<p>コマンド形式の記述の最後、コマンド・ライン、コード・ラインにおいて、ハイフンは、要求に対する引数とその後の行に続くことを示します。</p>
数字	<p>特に明記しない限り、本文中の数字はすべて 10 進数です。10 進数以外 (2 進数, 8 進数, 16 進数) は、その旨を明記してあります。</p>

日本語ライブラリは、日本語データを取り扱うための基本的な機能を提供するライブラリです。基本的な日本語文字列操作、文字列変換、ローマ字/かな漢字変換、漢字コード変換などを含み、OpenVMS オペレーティング・システムでサポートされるすべてのプログラミング言語から呼び出すことができます。ユーザのプログラム内から呼び出すことにより、日本語アプリケーションを簡単に作成することができます。

## 1.1 日本語ライブラリの構成

OpenVMS の日本語ライブラリは、以下の 4 種類のライブラリで構成されています。

ライブラリの種類	内容	本書での説明箇所
汎用ライブラリ	プログラミング言語から標準的なインターフェイスで利用できる各種日本語処理ルーチン群	第 3 章
基本ライブラリ	1 文字単位の変換など、より細かな処理を行うためのルーチン群	第 4 章
かな漢字変換ライブラリ	かな漢字変換を行うためのルーチン群	第 5 章
漢字コード変換ライブラリ	DEC 漢字コードを他社の漢字コードへ、他社の漢字コードを DEC 漢字コードへ変換する漢字コード変換ルーチン群	第 6 章

これらのライブラリのうち、汎用/基本ライブラリ、かな漢字変換ライブラリは、次の共有イメージ・ファイルに含まれています。

`SYS$SHARE:JSYSHR.EXE`

また、漢字コード変換ライブラリは、次のオブジェクト・ライブラリ・ファイルに含まれています。

`JSY$LIBRARY:JSYLIB.OLB`

## 1.2 日本語ライブラリのリンク方法

ここでは、日本語ライブラリをプログラムとリンクする方法について説明します。

## 概要

### 1.2 日本語ライブラリのリンク方法

#### 1.2.1 漢字コード変換ライブラリ以外の日本語ライブラ리를リンクする方法

漢字コード変換ライブラリを除いた日本語ライブラ리를, ユーザのプログラムとリンクするには, 以下の例のように共有イメージ SYS\$SHARE:JSYSHR.EXE とリンクしてください。

```
$ LINK PROG, SYS$INPUT/OPTION  
SYS$SHARE:JSYSHR/SHARE  
$
```

または

```
$ LINK PROG, JSY$LIBRARY:JSYSHR/OPTION
```

#### 1.2.2 漢字コード変換ライブラリを含むすべての日本語ライブラ리를リンクする方法

漢字コード変換ライブラリを含んだ日本語ライブラ리를, ユーザのプログラムとリンクするには, 以下のようにオブジェクト・ライブラリ JSY\$LIBRARY:JSYLIB.OLB および共有イメージ SYS\$SHARE:JSYSHR.EXE とリンクしてください。

```
$ LINK PROG, JSY$LIBRARY:JSYLIB/LIBRARY, SYS$INPUT/OPTION  
SYS$SHARE:JSYSHR/SHARE
```

または

```
$ LINK PROG, JSY$LIBRARY:JSYLIB/LIBRARY, JSY$LIBRARY:JSYSHR/OPTION
```

#### 1.2.3 漢字コード変換ライブラリのみをリンクする方法

漢字コード変換ライブラリのための日本語ライブラ리를, ユーザのプログラムとリンクするには, 以下のようにオブジェクト・ライブラリ JSY\$LIBRARY:JSYLIB.OLB とリンクしてください。

```
$ LINK PROG, JSY$LIBRARY:JSYLIB/LIBRARY
```

---

## 日本語ライブラリの使用例

この章では、OpenVMS オペレーティング・システムでサポートされる各プログラミング言語からの日本語ライブラリの簡単な使用例を示します。標準の実行時ライブラリと同じように使用することができ、日本語エディタでプログラムを作成することにより、注釈や文字リテラルにも日本語を使用することができます。

---

### 2.1 FORTRAN における日本語の使用例

```
JSY$EXAMPLES:GETINPUT.FOR

$
$ type getinput.for
program      sample
character buff*40
integer stat

! 注釈に日本語が使えます。

type *, 'DEC FORTRAN では、文字リテラルに日本語が使えます。'

do while (jlb$get input(buff, 'データ入力 : ', leng))
    type *, 'データ出力 : ', buff
end do

end

$
$ fortran getinput
$ link  getinput, jsy$library:jsysshr/option
$ run  getinput
DEC FORTRAN では、文字リテラルに日本語が使えます。
データ入力 : 日本語データの
データ出力 : 日本語データの
データ入力 : 入出力が可能です。
データ出力 : 入出力が可能です。
データ入力 : *EXIT*
$
```

---

## 2.2 Cにおける日本語の使用例

```
JSY$EXAMPLES:GETINPUT.C
$
$ type getinput.c
#include <stdio.h>
#include <ssdef.h>
#include <descrip.h>
main () /* 注釈に日本語が使えます。*/
{
    static char buff[40];
    static char prom[] = "データ入力 : ";
    $DESCRIPTOR (buf,buff);
    $DESCRIPTOR (pro,prom);
    short len;

    printf("\nDEC C では、文字リテラルに日本語が使えます。 \n");
    while (jlb$get_input (&buf,&pro,&len) == SS$_NORMAL) {
        buff[len] = 0;
        printf("データ出力 : %s\n",buff);
    }
}
$
$ cc    getinput
$ link  getinput,sys$input:/option
sys$share:jsyshr.exe/share
*EXIT*
$ run  getinput
DEC C では、文字リテラルに日本語が使えます。
データ入力 : 日本語データの
データ出力 : 日本語データの
データ入力 : 入出力が可能です。
データ出力 : 入出力が可能です。
データ入力 : *EXIT*
$
```

---

## 2.3 MACROにおける日本語の使用例

```
JSY$EXAMPLES:GETINPUT.MAR
$
$ type getinput.mar
.title test

.psect data
.show
.macro string str
.ascid /str/
.endm string
```

```
data1: .ascid /MACRO では、文字リテラルに日本語が使えます。 /
data2: string <MACRO では、マクロに日本語が使えます。 >
inp: string <データ入力 : >
oup: .long b3-b1
     .address b1
buff: .long b3-b2
     .address b2
b1: .ascii /データ出力 : /
b2: .blkb 40
b3:
     .psect code
     .entry test,^m<> ; 注釈に日本語が使えます。
pushaq data1
calls #1,g^lib$put_output
pushaq data2
calls #1,g^lib$put_output
loop:
pushaq inp
pushaq buff
calls #2,g^jlb$get_input
blbc r0,eof
pushaq oup
calls #1,g^lib$put_output
brb loop
eof:
$exit_s
     .end test

$
$ macro getinput
$ link  getinput,jsy$library:jsyshr/option
$ run  getinput
MACRO では、文字リテラルに日本語が使えます。
MACRO では、マクロに日本語が使えます。
データ入力 : 日本語データの
データ出力 : 日本語データの
データ入力 : 入出力が可能です。
データ出力 : 入出力が可能です。
データ入力 : *EXIT*
$
```



---

## 汎用ライブラリ

汎用ライブラリとは、プログラミング言語から標準的なインタフェースで利用できる日本語処理ルーチン群です。

この章では、汎用ライブラリの次の項目について説明します。

- 汎用ライブラリ・ルーチン一覧 (第 3.1 節)
- 日本語文字列操作ルーチン (第 3.2 節)
- 文字列変換ルーチン (第 3.3 節)
- 漢字変換入力ルーチン (第 3.4 節)
- 日本語ファイル名ルーチン (第 3.5 節)
- その他のルーチン (第 3.6 節)

---

### 3.1 汎用ライブラリ・ルーチン一覧

#### 日本語文字列操作ルーチン

日本語データを扱うための基本的なルーチン群です。1 バイト/2 バイトの文字コードを正しく取り扱うために使用します。日本語文字列操作ルーチンの詳細は第 3.2 節を参照してください。

JLB\$NOF_BYTE	文字列が占めるバイト数の入手
JLB\$NOF_CHAR	文字列に含まれる文字数の入手
JLB\$POS_CURR	現在の文字の文字位置の入手
JLB\$POS_NEXT	次の文字の文字位置の入手
JLB\$POS_PREV	前の文字の文字位置の入手
JLB\$POSITION	文字列の検索
JLB\$STR_EQUAL	英大文字/小文字，全角/半角， ひらがな/カタカナ変換による文字列の比較
JLB\$STR_START	英大文字/小文字，全角/半角， ひらがな/カタカナ変換による文字列の比較
JLB\$STR_SEARCH	英大文字/小文字，全角/半角， ひらがな/カタカナ変換による文字列の検索
JLB\$TRIM	文字列の後の空白の切り捨て
JLB\$TRUNC	文字列の切り捨て
JLB\$LOCC	文字の検索
JLB\$SKPC	文字の飛越し
JLB\$RCHAR	文字列からの文字の取り出し

## 汎用ライブラリ

### 3.1 汎用ライブラリ・ルーチン一覧

JLB\$RNEXT	文字列からの文字の取り出し， およびインデックスの更新
JLB\$WCHAR	文字列への文字の書込み
JLB\$WNEXT	文字列への文字の書込み， およびインデックスの更新

#### 文字列変換ルーチン

文字列の各種変換を行うルーチン群です。文字列変換ルーチンの詳細は第 3.3 節を参照してください。

JLB\$TRA_ROM_HALF	ローマ文字全角から半角への変換
JLB\$TRA_ROM_FULL	ローマ文字半角から全角への変換
JLB\$TRA_ROM_SIZE	ローマ文字全角/半角の相互変換
JLB\$TRA_ROM_LOWER	英文字半角/全角の小文字への変換
JLB\$TRA_ROM_UPPER	英文字半角/全角の大文字への変換
JLB\$TRA_ROM_CASE	英文字全角/半角の大文字/小文字の相互変換
JLB\$TRA_ROM_KANA	ローマ字から全角ひらがな/カタカナへの変換
JLB\$TRA_KANA_HIRA	全角カタカナから全角ひらがなへの変換
JLB\$TRA_KANA_KATA	全角ひらがなから全角カタカナへの変換
JLB\$TRA_KANA_KANA	全角ひらがな/全角カタカナの相互変換
JLB\$TRA_KANA_DAKU	全角ひらがな/カタカナの濁点/半濁点処理
JLB\$TRA_KANA_HALF	全角ひらがな/カタカナから半角カタカナへの変換
JLB\$TRA_KANA_FULL	半角カタカナから全角ひらがな/カタカナへの変換
JLB\$TRA_SYMBOL	記号変換

#### 漢字変換入力ルーチン

ローマ字・かな漢字変換入力を簡単に行うためのルーチン群です。漢字変換入力ルーチンの詳細は第 3.4 節を参照してください。

JLB\$GET_COMMAND	LIB\$GET_COMMAND のローマ字・かな漢字変換入力版
JLB\$GET_INPUT	LIB\$GET_INPUT のローマ字・かな漢字変換入力版

#### 日本語ファイル名ルーチン

日本語ファイル名を使用するためのルーチンです。日本語ファイル名ルーチンの詳細は第 3.5 節を参照してください。

JLB\$RMS_USER_VTF7	指定されたコードセットから VTF-7 コードセットへ日本語ファイル名の文字コード変換を行う。
JLB\$RMS_VTF7_USER	VTF-7 コードセットから指定されたコードセットへ日本語ファイル名の文字コード変換を行う。
JSY\$RMS_GET_ENCODING	現在のファイル名コンバータの名前を取得する
JSY\$RMS_LIST_ENCODING	システムにインストールされているファイル名コンバータの名前を取得する

JSY\$RMS_SET_ENCODING	RMS ファイル名コンバータを有効または無効にする
JSY\$RMS_USER_VTF7	指定されたコードセットから VTF-7 コードセットへ日本語ファイル名の文字コード変換を行う。
JSY\$RMS_VTF7_USER	VTF-7 コードセットから指定されたコードセットへ日本語ファイル名の文字コード変換を行う。

#### その他のルーチン

JLB\$DATE_TIME	日付けおよび時間の入手
JLB\$DEV_KANJI	漢字端末かどうかのチェック
JLB\$DEV_KANJI_IN	漢字入力端末かどうかのチェック

これらのルーチンの詳細は第 3.6 節を参照してください。

---

## 3.2 日本語文字列操作ルーチン

日本語文字列操作ルーチンは、日本語データを扱うための基本的なルーチン群です。1 バイト/2 バイトの文字コードを正しく取り扱うために使用します。

---

## JLB\$NOF\_BYTE

文字列が占めるバイト数の入手

文字列内の指定された文字数によって占められるバイト数を返します。

---

### フォーマット

nof-byte = JLB\$NOF\_BYTE ( *src-str*, *nof-char* )

---

### 引数

src-str		
	JLB usage	入力文字列
	access	入力のみ
	mechanism	Descriptor 渡し
nof_char		
	JLB usage	文字数
	type	Longword
	access	入力のみ
	mechanism	Reference 渡し

---

### 戻り値

nof_byte	バイト数
----------	------

---

### 例

```
#include <stdio.h>
#include <descrip.h>
#include <string.h>
```

```
main()
{
    struct dsc$descriptor_s    src_str;
    int                        nof_byte;
    extern int  jlb$nof_byte();
    char        src_str_body[] = "日本語RTL";
    int        nof_char = 4;

    src_str.dsc$b_class = DSC$K_CLASS_S;
    src_str.dsc$b_dtype = DSC$K_DTYPE_T;
    src_str.dsc$w_length = strlen( src_str_body );
    src_str.dsc$a_pointer = src_str_body;

    nof_byte = jlb$nof_byte( &src_str, &nof_char );

    printf( "文字列 : \"%s\"    文字数 :%d\n 結果 :%d\n",
           src_str_body, nof_char, nof_byte );
}
```

---

## JLB\$NOF\_CHAR

文字列が含まれる文字数の入手

文字列全体に含まれる文字数を返します。

---

### フォーマット

nof-char = JLB\$NOF\_CHAR ( *src-str* )

---

### 引数

src-str		
JLB usage		入力文字列
access		入力のみ
mechanism		Descriptor 渡し

---

### 戻り値

nof-char	文字数
----------	-----

---

## JLB\$POS\_CURR

現在の文字の文字位置の入手

指定された文字位置が、文字の1バイト目、2バイト目のどちらを指しているかにかかわらず、常に現在指している文字の1バイト目の文字位置を返します。

---

### フォーマット

```
index = JLB$POS_CURR ( src-str, curr-index )
```

---

### 引数

src-str	
JLB usage	入力文字列
access	入力のみ
mechanism	Descriptor 渡し
curr-index	
JLB usage	現在の文字位置
type	Longword
access	入力のみ
mechanism	Reference 渡し

---

### 戻り値

index	文字位置
-------	------

---

## JLB\$POS\_NEXT

次の文字の文字位置の入手  
指定された文字位置が、文字の1バイト目、2バイト目のどちらを指しているかにかかわらず、常に現在指している文字の次の文字の、1バイト目の文字位置を返します。

---

### フォーマット

`index = JLB$POS_NEXT ( src-str, curr-index )`

---

### 引数

src-str	
JLB usage	入力文字列
access	入力のみ
mechanism	Descriptor 渡し
curr-index	
JLB usage	現在の文字位置
type	Longword
access	入力のみ
mechanism	Reference 渡し

---

### 戻り値

index	文字位置
	curr-index が src-str の長さより大きい場合は 0
	curr-index がマイナスの場合は 0



---

例

```
#include <stdio.h>
#include <descrip.h>
#include <string.h>

main()
{
    struct dsc$descriptor_s    src_str;
    int                        index;
    extern int  jlb$pos_next();
    char        src_str_body[] = "日本語RTL";
    int        curr_index = 1;

    src_str.dsc$b_class = DSC$K_CLASS_S;
    src_str.dsc$b_dtype = DSC$K_DTYPE_T;
    src_str.dsc$w_length = strlen( src_str_body );
    src_str.dsc$a_pointer = src_str_body;

    index = jlb$pos_next( &src_str, &curr_index );

    printf( "文字列 : \"%s\"      文字位置 : %d\n 結果 : %d\n",
           src_str_body, curr_index, index );
}
```

---

## JLB\$POS\_PREV

前の文字の文字位置の入手

指定された文字位置が、文字の1バイト目または2バイト目のどちらを指しているかにかかわらず、常に現在指している文字の前の文字の1バイト目の文字位置を返します。

---

### フォーマット

`index = JLB$POS_PREV ( src-str, curr-index )`

---

### 引数

src-str		
JLB usage		入力文字列
access		入力のみ
mechanism		Descriptor 渡し
curr-index		
JLB usage		現在の文字位置
type		Longword
access		入力のみ
mechanism		Reference 渡し

---

### 戻り値

index	文字位置
	curr-index が src-str の長さより大きい場合は 0
	curr-index がゼロ以下の場合には 0

---

## JLB\$POSITION

入力文字列中に含まれる検索文字列の先頭の文字位置を返します。

検索文字列が含まれていない場合は 0 を返します。

---

### フォーマット

`index = JLB$POSITION ( src-str, sub-str [, start-pos] )`

---

### 引数

src-str		
	JLB usage	入力文字列
	access	入力のみ
	mechanism	Descriptor 渡し
sub-str		
	JLB usage	検索文字列
	access	入力のみ
	mechanism	Descriptor 渡し
start-pos		
	JLB usage	検索開始文字位置 (省略時は 1)
	type	Longword
	access	入力のみ
	mechanism	Reference 渡し

---

### 戻り値

index	検索文字列が含まれている場合はその開始位置 検索文字列が含まれていなければ 0 検索文字列の長さがゼロの場合は 1
-------	---

---

## JLB\$STR\_EQUAL

英大文字/小文字，全角/半角，ひらがな/カタカナ変換による文字列の比較

---

### フォーマット

```
match = JLB$STR_EQUAL ( str1, str2 [, flg] )
```

---

### 引数

str1	JLB usage	比較対象文字列 1	
	access	入力のみ	
	mechanism	Descriptor 渡し	
str2	JLB usage	比較対象文字列 2	
	access	入力のみ	
	mechanism	Descriptor 渡し	
flg	JLB usage	変換フラグ (省略時は 0)	
	type	Byte	
	access	入力のみ	
	mechanism	Reference 渡し	
			比較の前に行う変換を指定する。
	bit 0	0 : 英大文字/小文字変換を行う 1 : 英大文字/小文字変換を行わない	
	bit 1	0 : 全角/半角変換を行う 1 : 全角/半角変換を行わない	
bit 2	0 : ひらがな/カタカナ変換を行う 1 : ひらがな/カタカナ変換を行わない		

---

### 戻り値

match	1: 文字列 1 と文字列 2 が等しい
-------	----------------------

0: 文字列 1 と文字列 2 が等しくない

---

## JLB\$STR\_START

英大文字/小文字，全角/半角，ひらがな/カタカナ変換による文字列の比較  
文字列 1 が文字列 2 で始まっているかどうかを調べます。

---

### フォーマット

```
match = JLB$STR_START ( str1, str2 [, flg] )
```

---

### 引数

str1	JLB usage	比較対象文字列 1	
	access	入力のみ	
	mechanism	Descriptor 渡し	
str2	JLB usage	比較対象文字列 2	
	access	入力のみ	
	mechanism	Descriptor 渡し	
flg	JLB usage	変換フラグ	
	type	Byte	
	access	入力のみ	
	type	Reference 渡し	
		比較の前に行う変換を指定する。	
	bit 0	0 : 英大文字/小文字変換を行う 1 : 英大文字/小文字変換を行わない	
	bit 1	0 : 全角/半角変換を行う 1 : 全角/半角変換を行わない	
bit 2	0 : ひらがな/カタカナ変換を行う 1 : ひらがな/カタカナ変換を行わない		

---

## 戻り値

match

1 : 文字列 1 が文字列 2 で始まっている

0 : 文字列 1 が文字列 2 で始まっていない

---

## JLB\$STR\_SEARCH

英大文字/小文字，全角/半角，ひらがな/カタカナ変換による文字列の検索

検索文字列が含まれていない場合は0を返します。

---

### フォーマット

index = JLB\$STR\_SEARCH ( *src-str*, *sub-str*[[, *start-pos*][, *flg*]] )

---

### 引数

src-str	JLB usage	入力文字列
	access	入力のみ
	mechanism	Descriptor 渡し
sub-str	JLB usage	検索文字列
	access	入力のみ
	mechanism	Descriptor 渡し
start-pos	JLB usage	検索開始文字位置 (省略時は 1)
	type	Longword
	access	入力のみ
	mechanism	Reference 渡し
flg	JLB usage	変換フラグ (省略時は 0)
	type	Byte
	access	入力のみ
	mechanism	Reference 渡し
		検索の前に行う変換を指定する。
bit 0	0	: 英大文字/小文字変換を行う
	1	: 英大文字/小文字変換を行わない
bit 1	0	: 全角/半角変換を行う
	1	: 全角/半角変換を行わない
bit 2	0	: ひらがな/カタカナ変換を行う



## 1 : ひらがな/カタカナ変換を行わない

## 戻り値

index	検索文字列が含まれている場合はその開始位置 検索文字列が含まれていなければ 0 検索文字列の長さがゼロの場合は 1
-------	---

## 例

```

#include <stdio.h>
#include <descrip.h>
#include <string.h>

main()
{
    struct dsc$dSCRIPTOR_s    src_str;
    struct dsc$dSCRIPTOR_s    sub_str;
    char                      flg;
    int                        index;
    extern int  jlb$str_search();
    char                      src_str_body[] = "日本語ライブラリ",
                              sub_str_body[] = "らいぶらり";
    int                        start_pos = 1;

    src_str.dsc$b_class = DSC$K_CLASS_S;
    src_str.dsc$b_dtype = DSC$K_DTYPE_T;
    src_str.dsc$w_length = strlen( src_str_body );
    src_str.dsc$a_pointer = src_str_body;

    sub_str.dsc$b_class = DSC$K_CLASS_S;
    sub_str.dsc$b_dtype = DSC$K_DTYPE_T;
    sub_str.dsc$w_length = strlen( sub_str_body );
    sub_str.dsc$a_pointer = sub_str_body;

    printf( "文字列 1:\%s\ "   文字列 2:\%s\ \n", src_str_body, sub_str_body);
    for ( flg = 0; flg < 8; ++flg ) {
        index = jlb$str_search( &src_str, &sub_str, &start_pos, &flg );
        printf( "変換フラグ :%1d   結果 :%d\n", flg, index );
    }
}

```

---

## JLB\$TRIM

文字列の後の空白の切り捨て

文字列の後に含まれる空白 (半角および全角) ならびにタブを切り捨てます。

---

### フォーマット

```
ret-status = JLB$TRIM ( dst-str, src-str [, out-len] )
```

---

### 引数

dst-str		
	JLB usage	出力文字列
	access	出力のみ
	mechanism	Descriptor 渡し
src-str		
	JLB usage	入力文字列
	access	入力のみ
	mechanism	Descriptor 渡し
out-len		
	JLB usage	出力文字列のバイト長 (省略時は長さが返りません。)
	type	Word
	access	出力のみ
	mechanism	Reference 渡し

---

### 戻り値

ret-status	SS\$_NORMAL LIB\$_STRTRU
------------	-----------------------------

## 例

```

#include <stdio.h>
#include <descrip.h>
#include <ssdef.h>
#include <libdef.h>
#include <string.h>

#include <ssdef.h>
#include <stsdef.h>
#include <varargs.h>

main()
{
    struct dsc$b_descriptor_s    dst_str;
    struct dsc$b_descriptor_s    src_str;
    int        ret_status;
    extern int  jlb$trim();
    char        src_str_body[] = "日本語    ";
    char        dst_str_body[] = "外国人    ";

    dst_str.dsc$b_class = src_str.dsc$b_class = DSC$b_CLASS_S;
    dst_str.dsc$b_dtype = src_str.dsc$b_dtype = DSC$b_DTYPE_T;
    src_str.dsc$b_length = strlen (src_str_body);
    dst_str.dsc$b_length = strlen (dst_str_body);
    src_str.dsc$a_pointer = src_str_body;
    dst_str.dsc$a_pointer = dst_str_body;

    printf( "入力文字列 : \"%s\"\n", src_str_body );

    ret_status = jlb$trim( &dst_str, &src_str, &dst_str.dsc$b_length );

    dst_str_body[ dst_str.dsc$b_length ] = '\0';
    printf( "出力文字列 : \"%s\"    文字列長 : %d    ステータス :",
           dst_str.dsc$a_pointer, dst_str.dsc$b_length );

    switch ( ret_status ) {
    case SS$NORMAL:
        printf( "SS$NORMAL" );
        break;
    case LIB$STRTRU:
        printf( "LIB$STRTRU" );
        break;
    }
    putchar( '\n' );
}

```

---

# JLB\$TRUNC

文字列の切り捨て

文字列を指定された長さで切り捨てます。

---

## フォーマット

```
ret-status = JLB$TRUNC ( dst-str, src-str, size [, out-len] )
```

---

## 引数

dst-str	JLB usage	出力文字列
	access	出力のみ
	mechanism	Descriptor 渡し
src-str	JLB usage	入力文字列
	access	入力のみ
	mechanism	Descriptor 渡し
size	JLB usage	切り捨てを行う長さ
	type	Word
	access	入力のみ
	mechanism	Reference 渡し
out-len	JLB usage	出力文字列のバイト長
	type	Word
	access	出力のみ
	mechanism	Reference 渡し

---

戻り値

ret-status

SS\$\_NORMAL

LIB\$\_STRTRU

---

## JLB\$LOCC

### 文字の検索

入力文字列中で最初に現れた検索文字の文字位置を返します。検索文字が含まれていない場合は 0 を返します。

---

### フォーマット

```
index = JLB$LOCC ( char-str, src-str )
```

---

### 引数

<i>char-str</i>		
JLB usage		検索文字
access		入力のみ
mechanism		Descriptor 渡し
<i>src-str</i>		
JLB usage		入力文字列
access		入力のみ
mechanism		Descriptor 渡し

---

### 戻り値

index	文字位置
-------	------

---

### 例

```
#include <stdio.h>
#include <descrip.h>
#include <string.h>
```

```
main()
{
    struct dsc$descriptor_s    char_str;
    struct dsc$descriptor_s    src_str;
    int                        index;
    extern int  jlb$locc();
    char        char_str_body[] = "本",
               src_str_body[]  = "日本語";

    char_str.dsc$b_class = src_str.dsc$b_class = DSC$K_CLASS_S;
    char_str.dsc$b_dtype = src_str.dsc$b_dtype = DSC$K_DTYPE_T;
    char_str.dsc$w_length = strlen( char_str_body );
    char_str.dsc$a_pointer = char_str_body;
    src_str.dsc$w_length = strlen( src_str_body );
    src_str.dsc$a_pointer = src_str_body;

    index = jlb$locc( &char_str, &src_str );

    printf( "検索文字 : '%s\'   入力文字列 : \"%s\"\n 結果 : %d\n",
           char_str_body, src_str_body, index );
}
```

---

## JLB\$SKPC

文字の飛び越し

入力文字列中で最初に現れた検索文字以外の文字位置を返します。検索文字以外の文字がない場合は 0 を返します。

---

### フォーマット

index = JLB\$SKPC ( *char-str*, *src-str* )

---

### 引数

char-str		
JLB usage	検索文字	
access	入力のみ	
mechanism	Descriptor 渡し	
src-str		
JLB usage	入力文字列	
access	入力のみ	
mechanism	Descriptor 渡し	

---

### 戻り値

index	文字位置
	char-str の長さがゼロの場合は 1



---

# JLB\$RCHAR

文字列からの文字の取り出し

---

## フォーマット

`char-code = JLB$RCHAR ( src-str [, pos] )`

---

## 引数

src-str	JLB usage	入力文字列
	access	入力のみ
	mechanism	Descriptor 渡し
pos	JLB usage	文字位置
	type	Longword
	access	入力のみ
	mechanism	Reference 渡し

---

## 戻り値

char-code                      文字コード

---

# JLB\$RNEXT

文字列からの文字の取り出しおよびインデックスの更新

---

## フォーマット

```
char-code = JLB$RNEXT ( src-str [, pos] )
```

---

## 引数

src-str		
	JLB usage	入力文字列
	access	入力のみ
	mechanism	Descriptor 渡し
pos		
	JLB usage	文字位置
	type	Longword
	access	更新
	mechanism	Reference 渡し

---

## 戻り値

char-code	文字コード
-----------	-------

---

# JLB\$WCHAR

文字列への文字の書き込み

---

## フォーマット

ret-status = JLB\$WCHAR ( *char-code*, *dst-str* [, *pos*] )

---

## 引数

char-code		
JLB usage		文字コード
type		Longword
access		入力のみ
mechanism		Reference 渡し
dst-str		
JLB usage		出力文字列
access		出力のみ
mechanism		Descriptor 渡し
pos		
JLB usage		文字位置
type		Longword
access		入力のみ
mechanism		Reference 渡し

---

## 戻り値

ret-status	1 : 正常終了
	0 : 出力文字列の長さが短い

---

# JLB\$WNEXT

文字列への文字の書き込みおよびインデックスの更新

---

## フォーマット

ret-status = JLB\$WNEXT ( *char-code*, *dst-str* [, *pos*] )

---

## 引数

char-code	
JLB usage :	文字コード
type	Longword
access	入力のみ
mechanism	Reference 渡し
dst-str	
JLB usage	出力文字列
access	出力のみ
mechanism	Descriptor 渡し
pos	
JLB usage	文字位置
type	Longword
access	更新
mechanism	Reference 渡し

---

## 戻り値

ret-status	1 : 正常終了
	0 : 出力文字列の長さが短い

### 3.3 文字列変換ルーチン

文字列変換ルーチンは、文字列の各種変換を行うルーチン群です。

---

## JLB\$TRA\_ROM\_HALF

ローマ文字全角から半角への変換

---

### フォーマット

```
ret-status = JLB$TRA_ROM_HALF ( dst-str, src-str [, out-len] )
```

---

### 引数

dst-str		
	JLB usage	変換後文字列
	access	出力のみ
	mechanism	Descriptor 渡し
src-str		
	JLB usage	変換前文字列
	access	入力のみ
	mechanism	Descriptor 渡し
out-len		
	JLB usage	変換結果のバイト長
	type	Word
	access	出力のみ
	mechanism	Reference 渡し

---

### 戻り値

ret-status	SS\$_NORMAL LIB\$_STRTRU
------------	-----------------------------

---

### 例

入力文字列	" a b c d E F G H i j k l M N O P "
出力文字列	"abcdEFGHijklMNOP"

---

# JLB\$TRA\_ROM\_FULL

ローマ文字半角から全角への変換

---

## フォーマット

```
ret-status = JLB$TRA_ROM_FULL ( dst-str, src-str [, out-len] )
```

---

## 引数

dst-str	JLB usage	変換後文字列
	access	出力のみ
	mechanism	Descriptor 渡し
src-str	JLB usage	変換前文字列
	access	入力のみ
	mechanism	Descriptor 渡し
out-len	JLB usage	変換結果のバイト長
	type	Word
	access	出力のみ
	mechanism	Reference 渡し

---

## 戻り値

ret-status	SS\$_NORMAL LIB\$_STRTRU
------------	-----------------------------

---

## 例

入力文字列	"abcdEFGHijklMNOP"
出力文字列	" a b c d E F G H i j k l M N O P "

---

## JLB\$TRA\_ROM\_SIZE

ローマ文字全角/半角の相互変換

---

### フォーマット

ret-status = JLB\$TRA\_ROM\_SIZE ( *dst-str*, *src-str* [, *out-len*] )

---

### 引数

dst-str	JLB usage	変換後文字列
	access	出力のみ
	mechanism	Descriptor 渡し
src-str	JLB usage	変換前文字列
	access	入力のみ
	mechanism	Descriptor 渡し
out-len	JLB usage	変換結果のバイト長
	type	Word
	access	出力のみ
	mechanism	Reference 渡し

---

### 戻り値

ret-status                               SS\$\_NORMAL  
LIB\$\_STRTRU

---

### 例

入力文字列	" a b c d E F G H i j k l M N O P "
出力文字列	"abcd E F G H i j k l M N O P "



---

## JLB\$TRA\_ROM\_LOWER

英文字半角/全角の小文字への変換

---

### フォーマット

ret-status = JLB\$TRA\_ROM\_LOWER ( *dst-str*, *src-str* [, *out-len*] )

---

### 引数

dst-str	JLB usage	変換後文字列
	access	出力のみ
	mechanism	Descriptor 渡し
src-str	JLB usage	変換前文字列
	access	入力のみ
	mechanism	Descriptor 渡し
out-len	JLB usage	変換結果のバイト長
	type	Word
	access	出力のみ
	mechanism	Reference 渡し

---

### 戻り値

ret-status	SS\$_NORMAL LIB\$_STRTRU
------------	-----------------------------

---

### 例

入力文字列	"abcdEFGH i j k l MNOP"
出力文字列	"abcdefghijklmnop"

---

## JLB\$TRA\_ROM\_UPPER

英文字半角/全角の大文字への変換

---

### フォーマット

```
ret-status = JLB$TRA_ROM_UPPER ( dst-str, src-str [, out-len] )
```

---

### 引数

dst-str		
JLB usage		変換後文字列
access		出力のみ
mechanism		Descriptor 渡し
src-str		
JLB usage		変換前文字列
access		入力のみ
mechanism		Descriptor 渡し
out-len		
JLB usage		変換結果のバイト長
type		Word
access		出力のみ
mechanism		Reference 渡し

---

### 戻り値

ret-status	SS\$_NORMAL LIB\$_STRTRU
------------	-----------------------------

---

### 例

入力文字列	"abcdEFGH i j k l MNOP"
出力文字列	"ABCDEFHG I J K L MNOP"

---

# JLB\$TRA\_ROM\_CASE

英文字全角/半角の大文字/小文字の相互変換

---

## フォーマット

```
ret-status = JLB$TRA_ROM_CASE ( dst-str, src-str [, out-len] )
```

---

## 引数

dst-str	JLB usage	変換後文字列
	access	出力のみ
	mechanism	Descriptor 渡し
src-str	JLB usage	変換前文字列
	access	入力のみ
	mechanism	Descriptor 渡し
out-len	JLB usage	変換結果のバイト長
	type	Word
	access	出力のみ
	mechanism	Reference 渡し

---

## 戻り値

ret-status	SS\$_NORMAL LIB\$_STRTRU
------------	-----------------------------

---

## 例

入力文字列	"abcdEFGH i j k l MNOP "
出力文字列	"ABCDefgh I J K L m n o p "

---

## JLB\$TRA\_ROM\_KANA

ローマ字から全角ひらがな/カタカナへの変換

---

### フォーマット

`ret-status = JLB$TRA_ROM_KANA ( dst-str, src-str [, flg [, out-len]] )`

---

### 引数

<code>dst-str</code>		
<code>JLB usage</code>		変換後文字列
<code>access</code>		出力のみ
<code>mechanism</code>		Descriptor 渡し
<code>src-str</code>		
<code>JLB usage</code>		変換前文字列
<code>access</code>		入力のみ
<code>mechanism</code>		Descriptor 渡し
<code>flg</code>		
<code>JLB usage</code>		変換指示フラグ (省略時は 0)
<code>type</code>		Byte
<code>access</code>		入力のみ
<code>mechanism</code>		Reference 渡し
	<code>bit 0</code>	0 : 全角ひらがなへ変換 1 : 全角カタカナへ変換
<code>out-len</code>		
<code>JLB usage</code>		変換結果のバイト長
<code>type</code>		Word
<code>access</code>		出力のみ
<code>mechanism</code>		Reference 渡し

---

### 戻り値

`ret-status`                      `SS$_NORMAL`

## LIB\$\_STRTRU

## 例

入力文字列と出力文字列の例:

入力文字列	"ro-maji"	
出力文字列	" ろーまじ "	ひらがな指定
出力文字列	" ローマジ "	カタカナ指定

サンプル・コード:

```
#include <stdio.h>
#include <descrip.h>
#include <string.h>
#include <ssdef.h>
#include <libdef.h>

main()
{
    struct dsc$descriptor_s    dst_str;
    struct dsc$descriptor_s    src_str;
    char                        flg;
    int                         ret_status;
    extern int jlb$tra_rom_kana();
    char                        dst_str_body[] = "                ";
    char                        src_str_body[] = "nihongo";

    dst_str.dsc$b_class = DSC$K_CLASS_S;
    dst_str.dsc$b_dtype = DSC$K_DTYPE_T;
    dst_str.dsc$w_length = strlen( dst_str_body );
    dst_str.dsc$a_pointer = dst_str_body;

    src_str.dsc$b_class = DSC$K_CLASS_S;
    src_str.dsc$b_dtype = DSC$K_DTYPE_T;
    src_str.dsc$w_length = strlen( src_str_body );
    src_str.dsc$a_pointer = src_str_body;

    for ( flg = 0 ; flg < 2 ; ++flg ) {
        printf( "変換前文字列 : \"%s\"      変換フラグ :%1d\n", src_str_body, flg );

        ret_status = jlb$tra_rom_kana( &dst_str, &src_str,
                                       &flg, &dst_str.dsc$w_length );
        dst_str_body[ dst_str.dsc$w_length ] = '\0';
        printf( "変換後文字列 : \"%s\"      文字列長 :%d      ステータス :",
               dst_str.dsc$a_pointer, dst_str.dsc$w_length );
    }
}
```

```
switch ( ret_status ) {  
case SS$ _NORMAL:  
    printf( "SS$ _NORMAL" );  
    break;  
case LIB$ _STRTRU:  
    printf( "LIB$ _STRTRU" );  
    break;  
}  
printf( "\n\n" );  
}  
}
```

---

# JLB\$TRA\_KANA\_HIRA

全角カタカナから全角ひらがなへの変換

---

## フォーマット

```
ret-status = JLB$TRA_KANA_HIRA ( dst-str, src-str [, out-len] )
```

---

## 引数

dst-str	JLB usage	変換後文字列
	access	出力のみ
	mechanism	Descriptor 渡し
src-str	JLB usage	変換前文字列
	access	入力のみ
	mechanism	Descriptor 渡し
out-len	JLB usage	変換結果のバイト長
	type	Word
	access	出力のみ
	mechanism	Reference 渡し

---

## 戻り値

ret-status	SS\$_NORMAL LIB\$_STRTRU
------------	-----------------------------

---

## 例

入力文字列	" あいうえおカキクケコ "
出力文字列	" あいうえおかきくけこ "

---

# JLB\$TRA\_KANA\_KATA

全角ひらがなから全角カタカナへの変換

---

## フォーマット

```
ret-status = JLB$TRA_KANA_KATA ( dst-str, src-str [, out-len] )
```

---

## 引数

dst-str	JLB usage	変換後文字列
	access	出力のみ
	mechanism	Descriptor 渡し
src-str	JLB usage	変換前文字列
	access	入力のみ
	mechanism	Descriptor 渡し
out-len	JLB usage	変換結果のバイト長
	type	Word
	access	出力のみ
	mechanism	Reference 渡し

---

## 戻り値

ret-status	SS\$_NORMAL LIB\$_STRTRU
------------	-----------------------------

---

## 例

入力文字列	" あいうえおかきくけこ "
出力文字列	" アイウエオカキクケコ "



---

# JLB\$TRA\_KANA\_KANA

全角ひらがな/全角カタカナの相互変換

---

## フォーマット

```
ret-status = JLB$TRA_KANA_KANA ( dst-str, src-str [, out-len] )
```

---

## 引数

dst-str	JLB usage	変換後文字列
	access	出力のみ
	mechanism	Descriptor 渡し
src-str	JLB usage	変換前文字列
	access	入力のみ
	mechanism	Descriptor 渡し
out-len	JLB usage	変換結果のバイト長
	type	Word
	access	出力のみ
	mechanism	Reference 渡し

---

## 戻り値

ret-status	SS\$_NORMAL LIB\$_STRTRU
------------	-----------------------------

---

## 例

入力文字列	" あいうえおかきくけこ "
出力文字列	" アイウエオかきくけこ "

---

# JLB\$TRA\_KANA\_DAKU

全角ひらがな/カタカナの濁点/半濁点処理

---

## フォーマット

```
ret-status = JLB$TRA_KANA_DAKU ( dst-str, src-str [, out-len] )
```

---

## 引数

dst-str		
JLB usage		変換後文字列
access		出力のみ
mechanism		Descriptor 渡し
src-str		
JLB usage		変換前文字列
access		入力のみ
mechanism		Descriptor 渡し
out-len		
JLB usage		変換結果のバイト長
type		Word
access		出力のみ
mechanism		Reference 渡し

---

## 戻り値

ret-status	SS\$_NORMAL LIB\$_STRTRU
------------	-----------------------------

---

## 例

入力文字列	" か き く け こ は ひ ぶ へ ほ "
出力文字列	" がぎぐげごばびぶべほ "

---

# JLB\$TRA\_KANA\_HALF

全角ひらがな/カタカナから半角カタカナへの変換

---

## フォーマット

ret-status = JLB\$TRA\_KANA\_HALF ( *dst-str*, *src-str* [, *out-len*] )

---

## 引数

dst-str		
JLB usage		変換後文字列
access		出力のみ
mechanism		Descriptor 渡し
src-str		
JLB usage		変換前文字列
access		入力のみ
mechanism		Descriptor 渡し
out-len		
JLB usage		変換結果のバイト長
type		Word
access		出力のみ
mechanism		Reference 渡し

---

## 戻り値

ret-status	SS\$_NORMAL LIB\$_STRTRU
------------	-----------------------------

---

## 例

入力文字列	" あいうえおカキクケコ "
出力文字列	" アイエオカキクケコ "

---

# JLB\$TRA\_KANA\_FULL

半角カタカナから全角ひらがな/カタカナへの変換

---

## フォーマット

ret-status = JLB\$TRA\_KANA\_FULL ( *dst-str*, *src-str* [, *flg* [, *out-len*]] )

---

## 引数

dst-str	JLB usage	変換後文字列
	access	出力のみ
	mechanism	Descriptor 渡し
src-str	JLB usage	変換前文字列
	access	入力のみ
	mechanism	Descriptor 渡し
flg	JLB usage	変換指示フラグ
	type	Byte
	access	入力のみ
	mechanism	Reference 渡し
	bit 0	0 : 全角ひらがなへ変換 1 : 全角カタカナへ変換
	bit 1	0 : 濁点/半濁点処理を行う 1 : 濁点/半濁点処理を行わない
out-len	JLB usage	変換結果のバイト長
	type	Word
	access	出力のみ
	mechanism	Reference 渡し

---

戻り値

ret-status

SS\$\_NORMAL  
LIB\$\_STRTRU

---

例

入力文字列	" アイエオカキケコ "	
出力文字列	" あいうえおかきくけこ "	ひらがな指定
出力文字列	" アイウエオカキクケコ "	カタカナ指定

---

## JLB\$TRA\_SYMBOL

記号変換

変換規則は第 B.3 節を参照

---

### フォーマット

```
ret-status = JLB$TRA_SYMBOL ( dst-str, src-str [, out-len] )
```

---

### 引数

<i>dst-str</i>	JLB usage	変換後文字列
	access	出力のみ
	mechanism	Descriptor 渡し
<i>src-str</i>	JLB usage	変換前文字列
	access	入力のみ
	mechanism	Descriptor 渡し
<i>out-len</i>	JLB usage	変換結果のバイト長
	type	Word
	access	出力のみ
	mechanism	Reference 渡し

---

### 戻り値

<i>ret-status</i>	SS\$_NORMAL
	LIB\$_STRTRU

---

例

入力文字列	"[ ( ) ] <> k<> 123"
出力文字列	"【 】" "



## 3.4 漢字変換入力ルーチン

漢字変換入力ルーチンは、ローマ字・かな漢字変換入力を簡単に行うためのルーチン群です。

---

# JLB\$GET\_COMMAND

LIB\$GET\_COMMAND のローマ字・かな漢字変換入力版

---

## フォーマット

```
ret-status = JLB$GET_COMMAND ( get-str [,prompt-str] [,out-len] [,src-str] )
```

---

## 引数

get-str		
JLB usage		入力文字列
access		出力のみ
mechanism		Descriptor 渡し
prompt-str		
JLB usage		プロンプト文字列
access		入力のみ
mechanism		Descriptor 渡し
out-len		
JLB usage		入力文字列のバイト長
type		Word
access		出力のみ
mechanism		Reference 渡し
src-str		
JLB usage		変換対象文字列
access		入力のみ
mechanism		Descriptor 渡し

---

## 説明

標準ランタイム・ライブラリの LIB\$GET\_COMMAND と同様に使用でき、ローマ字・かな漢字変換入力が可能です。変換対象文字列は入力を行う前に表示され、変換を行うことができます。

---

戻り値

ret-status

SS\$\_NORMAL

RMS\$\_EOF

---

# JLB\$GET\_INPUT

LIB\$GET\_INPUT のローマ字・かな漢字変換入力版

---

## フォーマット

```
ret-status = JLB$GET_INPUT ( get-str [,prompt-str] [,out-len] [,src-str] )
```

---

## 引数

get-str		
JLB usage		入力文字列
access		出力のみ
mechanism		Descriptor 渡し
prompt-str		
JLB usage		プロンプト文字列
access		入力のみ
mechanism		Descriptor 渡し
out-len		
JLB usage		入力文字列のバイト長
type		Word
access		出力のみ
mechanism		Reference 渡し
src-str		
JLB usage		変換対象文字列
access		入力のみ
mechanism		Descriptor 渡し

---

## 説明

標準ランタイム・ライブラリの LIB\$GET\_INPUT と同様に使用でき、ローマ字・かな漢字変換入力が可能です。変換対象文字列は入力を行う前に表示され、変換を行うことができます。

---

戻り値

ret-status

SS\$\_NORMAL

RMS\$\_EOF

## 3.5 日本語ファイル名ルーチン

日本語ファイル名ルーチンは、日本語ファイル名を使用するためのルーチンです。

---

**注意**

---

日本語ファイル名ルーチンは OpenVMS VAX ではサポートされていません。

---

---

## JLB\$RMS\_USER\_VTF7

指定されたコードセットから VTF-7 コードセットへ、日本語ファイル名の文字コード変換を行います。

---

### フォーマット

```
ret-status = JLB$RMS_USER_VTF7 ( [dst-str], src-str, out-len, [codeset] )
```

---

### 引数

dst-str		
JLB usage		変換結果出力領域
type		Byte string
access		出力のみ
mechanism		Descriptor 渡し
src-str		
JLB usage		変換対象文字列
type		Byte string
access		入力のみ
mechanism		Descriptor 渡し
out-len		
JLB usage		変換結果のバイト長
type		Word
access		出力のみ
mechanism		Reference 渡し
codeset		
JLB usage		コードセット名
type		Byte string
access		入力のみ
mechanism		Descriptor 渡し

---

## 説明

このルーチンは RMS を使用せずにファイルにアクセスするアプリケーションなどが、日本語ファイル名をサポートするために利用することができます。

日本語ファイル名は Windows NT のサポートするファイル名に準拠しているため、一部の文字の変換規則は iconv と異なります。

変換結果出力領域は Fixed-Length String Descriptor または Dynamic String Descriptor である必要があります。変換結果出力領域を省略した場合は (引数 `dst-str` がゼロ)、変換結果のバイト数のみを `out-len` に出力します。

コードセット名を省略した場合は (引数 `codeset` がゼロ) 現在のプロセスが使用しているコードセットを自動的に選択します。

---

## 戻り値

<code>ret-status</code>	<code>SS\$_NORMAL</code> <code>JSY\$_STRTRU</code>
-------------------------	---



---

## JLB\$RMS\_VTF7\_USER

VTF-7 コードセットから指定されたコードセットへ、日本語ファイル名の文字コード変換を行います。

---

### フォーマット

```
ret-status = JLB$RMS_VTF7_USER ( [dst-str], src-str, out-len, [codeset])
```

---

### 引数

dst-str		
JLB usage		変換結果出力領域
type		Byte string
access		出力のみ
mechanism		Descriptor 渡し
src-str		
JLB usage		変換対象文字列
type		Byte string
access		入力のみ
mechanism		Descriptor 渡し
out-len		
JLB usage		変換結果のバイト長
type		Word
access		出力のみ
mechanism		Reference 渡し
codeset		
JLB usage		コードセット名
type		Byte string
access		入力のみ
mechanism		Descriptor 渡し

---

## 説明

このルーチンは RMS を使用せずにファイルにアクセスするアプリケーションなどが、日本語ファイル名をサポートするために利用することができます。

日本語ファイル名は Windows NT のサポートするファイル名に準拠しているため、一部の文字の変換規則は iconv と異なります。

変換結果出力領域は Fixed-Length String Descriptor または Dynamic String Descriptor である必要があります。変換結果出力領域を省略した場合は (引数 `dst-str` がゼロ)、変換結果のバイト数のみを `out-len` に出力します。

コードセット名を省略した場合は (引数 `codeset` がゼロ)、現在のプロセスが使用しているコードセットを自動的に選択します。

---

## 戻り値

<code>ret-status</code>	<code>SS\$_NORMAL</code>
	<code>JSY\$_STRTRU</code>

---

## JSY\$RMS\_GET\_ENCODING

現在の RMS ファイル名コンバータの状態を取得します。

---

### フォーマット

```
ret-status = JSY$RMS_GET_ENCODING ( dst-str, dst-len )
```

---

### 引数

dst-str		
JLB usage		コンバータ名の出力領域の先頭を指すポインタ
type		文字列
access		出力のみ
mechanism		Value 渡し
dst-len		
JLB usage		コンバータ名のバイト長
type		Longword
access		出力のみ
mechanism		Reference 渡し

---

### 説明

コンバータが有効の場合は "sdeckanji" が返されます。コンバータが無効の場合は "DISABLE" が返されます。

---

### 戻り値

ret-status	SS\$_NORMAL
	SS\$_INSFARG
	SS\$_TOO_MANY_ARGS

---

## JSY\$RMS\_LIST\_ENCODING

システムにインストールされているファイル名コンバータの名前を取得します。

---

### フォーマット

```
ret-status = JSY$RMS_LIST_ENCODING ( context, dst-str, dst-len )
```

---

### 引数

context		
JLB usage		コンバータ管理用インデックス
type		Longword
access		更新
mechanism		Reference 渡し
dst-str		
JLB usage		コンバータ名の出力領域の先頭を指すポインタ
type		文字列
access		出力のみ
mechanism		Value 渡し
dst-len		
JLB usage		コンバータ名のバイト長
type		Longword
access		出力のみ
mechanism		Reference 渡し

---

### 説明

最初に JSY\$RMS\_LIST\_ENCODING を呼び出す時に、あらかじめ引数 context をゼロに初期化しておきます。そして JSY\$RMS\_LIST\_ENCODING を呼び出すごとに、システムにインストールされているファイル名コンバータを1つずつ返します。

全てのコンバータの名前を取得し終わると、戻り値として SS\$\_NOMOREITEMS が返されます。この場合、コンバータ名は返されず、引数 context も更新されません。

---

戻り値

ret-status	SS\$_NORMAL
	RMS\$_EXTNOTFOU
	SS\$_BADPARAM
	SS\$_INSFARG
	SS\$_NOMOREITEMS
	SS\$_TOO_MANY_ARGS

---

## JSY\$RMS\_SET\_ENCODING

RMS ファイル名コンバータを有効または無効に設定します。

---

### フォーマット

```
ret-status = JSY$RMS_SET_ENCODING ( src-str, src-len )
```

---

### 引数

src-str		
JLB usage		コンバータ名の先頭を指すポインタ
type		文字列
access		入力のみ
mechanism		Value 渡し
src-len		
JLB usage		コンバータ名のバイト長
type		Longword
access		入力のみ
mechanism		Value 渡し

---

### 説明

コンバータを有効にする場合はコンバータ名として "sdeckanji" を指定します。コンバータを無効にする場合はコンバータ名として "DEFAULT" を指定します。なお、大文字/小文字の違いは無視されます。

---

### 戻り値

ret-status	SS\$_NORMAL
	RMS\$_EXTNOTFOU
	SS\$_ACCVIO
	SS\$_BADPARAM

JSY\$RMS\_SET\_ENCODING

SS\$\_INSFARG

SS\$\_TOO\_MANY\_ARGS

---

## JSY\$RMS\_USER\_VTF7

指定されたコードセットから VTF-7 コードセットへ、日本語ファイル名の文字コード変換を行います。

---

### フォーマット

```
ret-status = JSY$RMS_USER_VTF7 ( src-str, src-len, [dst-str], [dst-len], out-len, [codeset],
                                  [codeset-len])
```

---

### 引数

src-str		
	JSY usage	変換対象文字列
	type	Byte string
	access	入力のみ
	mechanism	Reference 渡し
src-len		
	JSY usage	変換対象文字列のバイト長
	type	Longword
	access	入力のみ
	mechanism	Value 渡し
dst-str		
	JSY usage	変換結果出力領域
	type	Byte string
	access	出力のみ
	mechanism	Reference 渡し
dst-len		
	JSY usage	変換結果出力領域のバイト長
	type	Longword
	access	入力のみ
	mechanism	Value 渡し
out-len		
	JSY usage	変換結果のバイト長
	type	Longword
	access	出力のみ



	mechanism	Reference 渡し
codeset		
	JSY usage	コードセット名
	type	Byte string
	access	入力のみ
	mechanism	Reference 渡し
codeset-len		
	JSY usage	コードセット名のバイト長
	type	Longword
	access	入力のみ
	mechanism	Value 渡し

---

## 説明

このルーチンは RMS を使用せずにファイルにアクセスするアプリケーションなどが、日本語ファイル名をサポートするために利用することができます。

日本語ファイル名は Windows NT のサポートするファイル名に準拠しているため、一部の文字の変換規則は iconv と異なります。

変換結果出力領域を省略した場合は (引数 dst-str および dst-len がゼロ)、変換結果のバイト数のみを out-len に出力します。

変換対象文字列と変換結果出力領域が全く同じアドレスだった場合、JSY\$RMS\_USER\_VTF7 は自動的に内部バッファを確保し、正常に変換を行います。しかし変換対象文字列と変換結果出力領域の一部だけが重複していた場合は、結果は保証されません。

コードセット名を省略した場合は (引数 codeset および codeset-len がゼロ)、現在のプロセスが使用しているコードセットを自動的に選択します。

---

## 戻り値

ret-status	SS\$_NORMAL
	JSY\$_STRTRU

---

## JSY\$RMS\_VTF7\_USER

VTF-7 コードセットから指定されたコードセットへ、日本語ファイル名の文字コード変換を行います。

---

### フォーマット

```
ret-status = JSY$RMS_VTF7_USER ( src-str, src-len, [dst-str], [dst-len], out-len, [codeset],
                                [codeset-len])
```

---

### 引数

src-str		
	JSY usage	変換対象文字列
	type	Byte string
	access	入力のみ
	mechanism	Reference 渡し
src-len		
	JSY usage	変換対象文字列のバイト長
	type	Longword
	access	入力のみ
	mechanism	Value 渡し
dst-str		
	JSY usage	変換結果出力領域
	type	Byte string
	access	出力のみ
	mechanism	Reference 渡し
dst-len		
	JSY usage	変換結果出力領域のバイト長
	type	Longword
	access	入力のみ
	mechanism	Value 渡し
out-len		
	JSY usage	変換結果のバイト長
	type	Longword
	access	出力のみ

	mechanism	Reference 渡し
codeset		
	JSY usage	コードセット名
	type	Byte string
	access	入力のみ
	mechanism	Reference 渡し
codeset-len		
	JSY usage	コードセット名のバイト長
	type	Longword
	access	入力のみ
	mechanism	Value 渡し

---

## 説明

このルーチンは RMS を使用せずにファイルにアクセスするアプリケーションなどが、日本語ファイル名をサポートするために利用することができます。

日本語ファイル名は Windows NT のサポートするファイル名に準拠しているため、一部の文字の変換規則は iconv と異なります。

変換結果出力領域を省略した場合は (引数 dst-str および dst-len がゼロ)、変換結果のバイト数のみを out-len に出力します。

変換対象文字列と変換結果出力領域が全く同じアドレスだった場合、JSY\$RMS\_VTF7\_USER は自動的に内部バッファを確保し、正常に変換を行います。しかし変換対象文字列と変換結果出力領域の一部だけが重複していた場合は、結果は保証されません。

コードセット名を省略した場合は (引数 codeset および codeset-len がゼロ)、現在のプロセスが使用しているコードセットを自動的に選択します。

---

## 戻り値

ret-status	SS\$_NORMAL
	JSY\$_STRTRU

## 3.6 その他のルーチン

---

## JLB\$DATE\_TIME

現在の日付けおよび時間の入手

---

### フォーマット

```
ret-status = JLB$DATE_TIME ( dst-str [, flg[, out-len]] )
```

---

### 引数

dst-str	JLB usage	出力文字列
	access	出力のみ
	mechanism	Descriptor 渡し
flg	JLB usage	時間制の指定
	type	Byte
	access	入力のみ
	mechanism	Reference 渡し
		0 : 12 時間制 (省略時の値) 1 : 24 時間制
out-len	JLB usage	出力文字列のバイト長
	type	Word
	access	出力のみ
	mechanism	Reference 渡し

---

### 戻り値

ret-status	SS\$_NORMAL LIB\$_STRTRU
------------	-----------------------------

---

 説明

JLB\$DATE\_TIME で使用する元号の名称および日付は、論理名で定義されます。この論理名は、システムのスタートアップ時に、SYS\$STARTUP:JSY\$DT\_STARTUP.COM を実行することによって定義されます (SYS\$STARTUP:JSY\$STARTUP.COM の中から実行されます)。省略時には次のように定義します。

```
$      define/system/exec jsy$era_name -
          "平成", -
          "昭和", -
          "大正", -
          "明治"
$      define/system/exec jsy$era_begin_date -
          "19890108", -
          "19261425", -
          "19140730", -
          "18680908"
$      define/system/exec jsy$era_display_form 1
```

3 番目の論理名、JSY\$ERA\_DISPLAY\_FORMAT は、その元号の初年度を「1 年 (0)」で表示するか、または「元年 (1)」で表示するかを選択します。省略時の設定では「元年」です。

12 時間制と 24 時間制の表記の違いは、以下の例のとおりです。

```
12時間制の場合 平成元年 2月14日 (火) 午後 5時21分 3秒
24時間制の場合 平成元年 2月14日 (火) 17時21分 3秒
```

---

 例

```
#include <stdio.h>
#include <descrip.h>
#include <ssdef.h>
#include <libdef.h>

#define BUFLLEN 100

main()
{
    struct dsc$descriptor_s    dst_str;
    char        flg;
    short int    out_len;
    int        ret_status;
    char        buffer[ BUFLLEN ];
    extern int  jlb$date_time();
```

```
dst_str.dsc$b_class = DSC$K_CLASS_S;
dst_str.dsc$b_dtype = DSC$K_DTYPE_T;
dst_str.dsc$w_length = sizeof( buffer );
dst_str.dsc$a_pointer = buffer;

for ( flg = 0 ; flg < 2 ; ++flg ) {
    ret_status = jlb$date_time( &dst_str, &flg, &out_len );

    buffer[ out_len ] = '\0';
    printf( "フラグ :%1d\n 時間 :\"%s\"   ステータス :",
           flg, dst_str.dsc$a_pointer );

    switch ( ret_status ) {
    case SS$ _NORMAL:
        printf( "SS$ _NORMAL" );
        break;
    case LIB$ _STRTRU:
        printf( "LIB$ _STRTRU" );
        break;
    }
    putchar( '\n' );
}
}
```

---

## JLB\$DEV\_KANJI

漢字端末かどうかのチェック

---

### フォーマット

ret-status = JLB\$DEV\_KANJI ( *device-name* )

---

### 引数

device-name		
JLB usage		デバイス名
type		文字列
access		入力のみ
mechanism		Descriptor 渡し

---

### 説明

端末が VT80 に設定されているかどうか、または KANJIGEN の SET /OUTPUT=KANJI により出力が漢字に設定されているかどうかをチェックします。

---

### 戻り値

ret-status	1 : 漢字端末
	0 : 漢字端末ではない



---

## JLB\$DEV\_KANJI\_IN

漢字入力端末かどうかのチェック

---

### フォーマット

```
ret-status = JLB$DEV_KANJI_IN ( device-name )
```

---

### 引数

device-name		
JLB usage	デバイス名	
type	文字列	
access	入力のみ	
mechanism	Descriptor 渡し	

---

### 説明

端末が KANJIGEN の SET /INPUT=KANJI により、漢字入力に設定されているかどうかをチェックします。

---

### 戻り値

ret-status	1 : 漢字入力端末
	0 : 漢字入力端末ではない

---

## 基本ライブラリ

基本ライブラリとは、1文字単位の変換など、より細かな処理を行うためのルーチン群です。

この章では、基本ライブラリの次の項目について説明します。

- 基本ライブラリ・ルーチン一覧 (第 4.1 節)
- 日本語文字列操作ルーチン (第 4.2 節)
- 文字変換ルーチン (第 4.3 節)
- 文字列変換ルーチン (第 4.4 節)

---

### 4.1 基本ライブラリ・ルーチン一覧

#### 日本語文字列操作ルーチン

日本語データを扱うための基本的なルーチン群です。1バイト/2バイトの文字コードを正しく取り扱うために使用します。各ルーチンの詳細については第 4.2 節を参照してください。

JSY\$CH_NEXT	次の文字の文字位置の入手
JSY\$CH_PREV	前の文字の文字位置の入手
JSY\$CH_CURR	現在の文字の文字位置の入手
JSY\$POS_NEXT	次の文字の文字位置の入手、 および文字列の終わりのチェック
JSY\$POS_PREV	前の文字の文字位置の入手、 および文字列の終わりのチェック
JSY\$POS_CURR	現在の文字の文字位置の入手、 および文字列の終わりのチェック
JSY\$CH_RCHAR	文字列からの文字の取り出し
JSY\$CH_RNEXT	文字列からの文字の取り出し
JSY\$CH_WCHAR	文字列への文字の書き込み
JSY\$CH_WNEXT	文字列への文字の書き込み
JSY\$CH_GCHAR	文字列からの文字の取り出し、 および文字列の終わりのチェック
JSY\$CH_GNEXT	文字列からの文字の取り出し、 および文字列の終わりのチェック
JSY\$CH_PCHAR	文字列への文字の書き込み、 および文字列の終わりのチェック

## 基本ライブラリ

### 4.1 基本ライブラリ・ルーチン一覧

JSY\$CH_PNEXT	文字列への文字の書き込み， および文字列の終わりのチェック
JSY\$CH_SIZE	文字が占めるバイト数の入手
JSY\$CH_NBYTE	文字列が占めるバイト数の入手
JSY\$CH_NCHAR	文字列に含まれる文字数の入手
JSY\$POSITION	文字列の検索
JSY\$STR_EQUAL	英大文字/小文字，全角/半角， ひらがな/カタカナ変換による文字列の比較
JSY\$STR_START	英大文字/小文字，全角/半角， ひらがな/カタカナ変換による文字列の比較
JSY\$STR_SEARCH	英大文字/小文字，全角/半角， ひらがな/カタカナ変換による文字列の検索
JSY\$TRIM	文字列の後の空白の切り捨て
JSY\$TRUNC	文字列の切り捨て
JSY\$LOCC	文字の検索
JSY\$SKPC	文字の飛び越し

#### 文字変換ルーチン

文字単位の各種変換を行うルーチン群です。各ルーチンの詳細については第 4.3 節を参照してください。

JSY\$CHG_ROM_HALF	ローマ文字全角から半角への変換
JSY\$CHG_ROM_FULL	ローマ文字半角から全角への変換
JSY\$CHG_ROM_SIZE	ローマ文字全角/半角の相互変換
JSY\$CHG_ROM_LOWER	英文字半角/全角の小文字への変換
JSY\$CHG_ROM_UPPER	英文字半角/全角の大文字への変換
JSY\$CHG_ROM_CASE	英文字全角/半角の大文字/小文字の相互変換
JSY\$CHG_KANA_HIRA	全角カタカナから全角ひらがなへの変換
JSY\$CHG_KANA_KATA	全角ひらがなから全角カタカナへの変換
JSY\$CHG_KANA_KANA	全角ひらがな/全角カタカナの相互変換
JSY\$CHG_KANA_DAKU	全角ひらがな/カタカナの濁点/半濁点処理
JSY\$CHG_KANA_HALF	全角ひらがな/カタカナから半角カタカナへの変換
JSY\$CHG_KANA_FULL	半角カタカナから全角ひらがな/カタカナへの変換
JSY\$CHG_GENERAL	パラメータ指定による，英大文字/小文字，全角/半角， ひらがな/カタカナ
JSY\$CHG_JIS_KUTEN	JIS 区点番号から DEC 漢字コードへの変換
JSY\$CHG_KEISEN	数字から罫線文字への変換

#### 文字列変換ルーチン

文字列の各種変換を行うルーチン群です。各ルーチンの詳細については第 4.4 節を参照してください。

JSY\$TRA_ROM_HALF	ローマ文字全角から半角への変換
JSY\$TRA_ROM_FULL	ローマ文字半角から全角への変換

JSY\$TRA_ROM_SIZE	ローマ文字全角/半角の相互変換
JSY\$TRA_ROM_LOWER	英文字半角/全角の小文字への変換
JSY\$TRA_ROM_UPPER	英文字半角/全角の大文字への変換
JSY\$TRA_ROM_CASE	英文字全角/半角の大文字/小文字の相互変換
JSY\$TRA_ROM_KANA	ローマ字から全角ひらがな/カタカナへの変換
JSY\$TRA_KANA_HIRA	全角カタカナから全角ひらがなへの変換
JSY\$TRA_KANA_KATA	全角ひらがなから全角カタカナへの変換
JSY\$TRA_KANA_KANA	全角ひらがな/全角カタカナの相互変換
JSY\$TRA_KANA_DAKU	全角ひらがな/カタカナの濁点/半濁点処理
JSY\$TRA_KANA_HALF	全角ひらがな/カタカナから半角カタカナへの変換
JSY\$TRA_KANA_FULL	半角カタカナから全角ひらがな/カタカナへの変換
JSY\$TRA_SYMBOL	記号変換

---

## 4.2 日本語文字列操作ルーチン

日本語文字列操作ルーチンは、日本語データを扱うための基本的なルーチン群です。  
1バイト/2バイトの文字コードを正しく取り扱うために使用します。

---

## JSY\$CH\_NEXT

次の文字の文字位置の入手

---

### フォーマット

```
next_ptr = JSY$CH_NEXT ( char_ptr )
```

---

### 引数

char_ptr		
	JSY usage	文字の 1 バイト目を指すポインタ
	mechanism	Value 渡し

---

### 戻り値

next_ptr	1 文字次の文字の 1 バイト目を指すポインタ
----------	-------------------------

---

### 例

```
#include <stdio.h>
#include <string.h>

static void create_display_string();

main()
{
    char *char_ptr,
        *next_ptr;
    char string[] = "日本語RTL";
    char disp_str[32];
    int byte_offset = 4;
    extern char *jsy$ch_next();

    char_ptr = string + byte_offset;
    printf( "文字列 : \"%s\"\n", string );
    create_display_string( char_ptr, disp_str );
    printf( "実行前の位置の文字 : %s\n", disp_str );
}
```

```
    next_ptr = jsy$ch_next( char_ptr );
    create_display_string( next_ptr, disp_str );
    printf( "実行後の位置の文字 :%s\n", disp_str );
}

static void    create_display_string(src, disp)
char    *src;
char    *disp;
{
    int nof_byte_char = 1;
    unsigned char    check;

    check = (unsigned char)(*src);
    if ( ( 0x0A1 <= check ) && ( check <= 0x0FE ) )    /* 範囲内なら2バイト文字 */
        nof_byte_char = 2;

    strncpy( disp, src, nof_byte_char );
    disp[ nof_byte_char ] = '\0';
}
```

---

## JSY\$CH\_PREV

前の文字の文字位置の入手

---

### フォーマット

prev-ptr = JSY\$CH\_PREV ( *top-ptr*, *char-ptr* )

---

### 引数

top-ptr		
	JSY usage	文字列の先頭を指すポインタ
	mechanism	Value 渡し
char-ptr		
	JSY usage	文字の 1 バイト目を指すポインタ
	mechanism	Value 渡し

---

### 戻り値

prev-ptr	1 文字前の文字の 1 バイト目を指すポインタ char-ptr top-ptr の場合は 0
----------	--

---

## JSY\$CH\_CURR

現在の文字の文字位置の入手

---

### フォーマット

curr-ptr = JSY\$CH\_CURR ( *top-ptr*, *char-ptr* )

---

### 引数

top-ptr		
	JSY usage	文字列の先頭を指すポインタ
	mechanism	Value 渡し
char-ptr		
	JSY usage	文字の 1 バイト目または 2 バイト目を指すポインタ
	mechanism	Value 渡し

---

### 戻り値

curr-ptr	文字の 1 バイト目を指すポインタ char-ptr < top-ptr の場合は 0
----------	--



---

## JSY\$POS\_NEXT

次の文字の文字位置の入手，および文字列の終わりのチェック

---

### フォーマット

next-ptr = JSY\$POS\_NEXT ( *char-ptr*, *end-ptr* )

---

### 引数

char-ptr		
	JSY usage	文字の 1 バイト目を指すポインタ
	mechanism	Value 渡し
end-ptr		
	JSY usage	文字列の終わりを示すポインタ
	mechanism	Value 渡し

---

### 戻り値

next-ptr	1 文字次の文字の 1 バイト目を指すポインタ char-ptr end-ptr の場合は 0
----------	--

---

## JSY\$POS\_PREV

前の文字の文字位置の入手，および文字列の終わりのチェック

---

### フォーマット

```
prev-ptr = JSY$POS_PREV ( top-ptr, char-ptr, end-ptr )
```

---

### 引数

top-ptr		
	JSY usage	文字列の先頭を指すポインタ
	mechanism	Value 渡し
char-ptr		
	JSY usage	文字の 1 バイト目を指すポインタ
	mechanism	Value 渡し
end-ptr		
	JSY usage	文字列の終わりを示すポインタ
	mechanism	Value 渡し

---

### 戻り値

prev-ptr	1 文字前の文字の 1 バイト目を指すポインタ
char-ptr	top-ptr または char-ptr > end-ptr の場合は 0

---

### 例

```
#include <stdio.h>
#include <string.h>
#define MSB    0x80
static void    create_display_string();
```

```

main()
{
    char        *top_ptr,
               *char_ptr,
               *end_ptr,
               *prev_ptr;
    char        string[] = "日本語RTL";
    char        disp_str[32];
    int         byte_offset = 6;
    extern char *jsy$pos_prev();

    top_ptr = string;
    end_ptr = string + strlen( string );
    char_ptr = string + byte_offset;

    create_display_string( char_ptr, disp_str );
    printf( "文字列 : \"%s\"\n", string );
    printf( "実行前の位置の文字 : %s\n", disp_str );

    prev_ptr = jsy$pos_prev( top_ptr, char_ptr, end_ptr );

    create_display_string( prev_ptr, disp_str );
    printf( "実行後の位置の文字 : %s\n", disp_str );
}

static void    create_display_string(src, disp)
char    *src;
char    *disp;
{
    int nof_byte_char = 1;
    unsigned char    check;

    check = (unsigned char)(*src);
    if ( ( 0x0A1 <= check ) && ( check <= 0xFE ) )    /* 範囲内なら2バイト文字 */
        nof_byte_char = 2;

    strncpy( disp, src, nof_byte_char );
    disp[ nof_byte_char ] = '\0';
}

```

---

## JSY\$POS\_CURR

現在の文字の文字位置の入手，および文字列の終わりのチェック

---

### フォーマット

`curr-ptr = JSY$POS_CURR ( top-ptr, char-ptr, end-ptr )`

---

### 引数

<code>top-ptr</code>	JSY usage	文字列の先頭を指すポインタ
	mechanism	Value 渡し
<code>char-ptr</code>	JSY usage	文字の 1 バイト目または 2 バイト目を指すポインタ
	mechanism	Value 渡し
<code>end-ptr</code>	JSY usage	文字列の終わりを示すポインタ
	mechanism	Value 渡し

---

### 戻り値

`curr-ptr` 文字の 1 バイト目を指すポインタ  
`char-ptr < top-ptr` または `char-pt` `end-ptr` の場合は 0

---

## JSY\$CH\_RCHAR

文字列からの文字の取り出し

---

### フォーマット

char-code = JSY\$CH\_RCHAR ( *char-ptr* )

---

### 引数

char-ptr		
	JSY usage	文字の 1 バイト目を指すポインタ
	mechanism	Value 渡し

---

### 戻り値

char-code	ポインタが指している文字の文字コード
-----------	--------------------

---

## JSY\$CH\_RNEXT

文字列からの文字の取り出し，およびポインタの更新

---

### フォーマット

```
char-code = JSY$CH_RNEXT ( char-ptr )
```

---

### 引数

<i>char-ptr</i>	
JSY usage	文字の 1 バイト目を指すポインタ
mechanism	Reference 渡し

---

### 戻り値

<i>char-code</i>	ポインタが指している文字の文字コード
------------------	--------------------

---

## JSY\$CH\_WCHAR

文字列への文字の書き込み

---

### フォーマット

JSY\$CH\_WCHAR ( *char-code*, *char-ptr* )

---

### 引数

char-code

JSY usage	文字コード
mechanism	Value 渡し

char-ptr

JSY usage	文字の 1 バイト目を指すポインタ
mechanism	Value 渡し

---

## JSY\$CH\_WNEXT

文字列への文字の書き込みおよびポインタの更新

---

### フォーマット

JSY\$CH\_WNEXT ( *char-code*, *char-ptr* )

---

### 引数

*char-code*

JSY usage 文字コード

mechanism Value 渡し

*char-ptr*

JSY usage 文字の1バイト目を指すポインタ

mechanism Reference 渡し



---

## JSY\$CH\_GCHAR

文字列からの文字の取り出し，および文字列の終わりのチェック

---

### フォーマット

char-code = JSY\$CH\_GCHAR ( *char-ptr*, *end-ptr* )

---

### 引数

char-ptr		
	JSY usage	文字の 1 バイト目を指すポインタ
	mechanism	Value 渡し
end-ptr		
	JSY usage	文字列の終わりを示すポインタ
	mechanism	Value 渡し

---

### 戻り値

char-code	ポインタが指している文字の文字コード 文字列の終わりを越えていた場合は 'FFFF'(16 進)
-----------	---

---

## JSY\$CH\_GNEXT

文字列からの文字の取り出し，ポインタの更新，および文字列の終わりのチェック

---

### フォーマット

```
char-code = JSY$CH_GNEXT ( char-ptr, end-ptr )
```

---

### 引数

char-ptr		
	JSY usage	文字の 1 バイト目を指すポインタ
	mechanism	Reference 渡し
end-ptr		
	JSY usage	文字列の終わりを示すポインタ
	mechanism	Value 渡し

---

### 戻り値

char-code	ポインタが指している文字の文字コード 文字列の終わりを越えていた場合は 'FFFF'(16 進)
-----------	---

---

## JSY\$CH\_PCHAR

文字列への文字の書き込み，および文字列の終わりのチェック

---

### フォーマット

```
status = JSY$CH_PCHAR ( char-code, char-ptr, end-ptr )
```

---

### 引数

char-code		
JSY usage	文字コード	
mechanism	Value 渡し	
char-ptr		
JSY usage	文字の 1 バイト目を指すポインタ	
mechanism	Value 渡し	
end-ptr		
JSY usage	文字列の終わりを示すポインタ	
mechanism	Value 渡し	

---

### 戻り値

status	1 : 文字の書き込みができた場合 0 : 文字の書き込みができなかった場合 (end-ptr を越えた)
--------	---

---

## JSY\$CH\_PNEXT

文字列への文字の書き込み，ポインタの更新，および文字列の終わりのチェック

---

### フォーマット

```
status = JSY$CH_PNEXT ( char-code, char-ptr, end-ptr )
```

---

### 引数

char-code		
	JSY usage	文字コード
	mechanism	Value 渡し
char-ptr		
	JSY usage	文字の 1 バイト目を指すポインタ
	mechanism	Reference 渡し
end-ptr		
	JSY usage	文字列の終わりを示すポインタ
	mechanism	Value 渡し

---

### 戻り値

status	1 : 文字の書き込みができた場合 0 : 文字の書き込みができなかった場合 (end-ptr を越えた)
--------	---

---

### 例

```
#include <stdio.h>
#include <string.h>
#define ENCODE( c )\
( ( ( ( c ) & 0xff ) << 8 ) | ( ( ( c ) & 0xff00 ) >> 8 ) )
static void    create_display_string();
```

```

main()
{
    unsigned short int  char_code;
    char               *char_ptr,
                       *end_ptr;
    int                status;
    int                byte_offset = 4;
    char               string[] = "実行前の文字列";
    char               disp_str[32];
    extern int         jsy$ch_pnext();

    char_code = ENCODE( '後' );
    char_ptr = string + byte_offset;
    end_ptr = string + strlen( string );

    printf( "書き込む文字 :%c%c\n",
            ( char_code & 0xff00 ) >> 8, char_code & 0xff );
    create_display_string( char_ptr, disp_str );
    printf( "実行前の文字列 :\"%s\"   実行前の位置の文字 :%s\n",
            string, disp_str );

    status = jsy$ch_pnext( char_code, &char_ptr, end_ptr );

    create_display_string( char_ptr, disp_str );
    printf( "実行後の文字列 :\"%s\"   実行後の位置の文字 :%s   ステータス :%d\n",
            string, disp_str, status );
}

static void         create_display_string(src, disp)
char               *src;
char               *disp;
{
    int nof_byte_char = 1;
    unsigned char    check;

    check = (unsigned char)(*src);
    if ( ( 0x0A1 <= check ) && ( check <= 0x0FE ) )    /* 範囲内なら2バイト文字 */
        nof_byte_char = 2;

    strncpy( disp, src, nof_byte_char );
    disp[ nof_byte_char ] = '\0';
}

```

---

## JSY\$CH\_SIZE

文字が占めるバイト数の入手

---

### フォーマット

nof-byte = JSY\$CH\_SIZE ( *char-code* )

---

### 引数

char-code

JSY usage	文字コード
mechanism	Value 渡し

---

### 戻り値

nof-byte

文字コードが 255 以下の場合は 1 , 256 以上の場合は 2

---

## JSY\$CH\_NBYTE

文字列が占めるバイト数の入手

---

### フォーマット

nof-byte = JSY\$CH\_NBYTE ( *top-ptr*,#*nof-char* )

---

### 引数

top-ptr		
	JSY usage	文字列の先頭を指すポインタ
	mechanism	Value 渡し
nof-char		
	JSY usage	文字数
	mechanism	Value 渡し

---

### 戻り値

nof-byte                      top-ptr から nof-char 文字が占めるバイト数

---

## JSY\$CH\_NCHAR

文字列に含まれる文字数の入手

---

### フォーマット

nof-char = JSY\$CH\_NCHAR ( *top-ptr*,#*nof-byte* )

---

### 引数

top-ptr	JSY usage	文字列の先頭を指すポインタ
	mechanism	Value 渡し
nof-byte	JSY usage	バイト数
	mechanism	Value 渡し

---

### 戻り値

nof-char	top-ptr から nof-byte バイト内にある文字数
----------	--------------------------------



---

# JSY\$POSITION

文字列の検索

---

## フォーマット

address = JSY\$POSITION ( *src-str*, *src-len*, *sub-str*, *sub-len* )

---

## 引数

src-str	JSY usage	検索される文字列の先頭を指すポインタ
	mechanism	Value 渡し
src-len	JSY usage	検索される文字列のバイト長
	mechanism	Value 渡し
sub-str	JSY usage	検索する文字列の先頭を指すポインタ
	mechanism	Value 渡し
sub-len	JSY usage	検索する文字列のバイト長
	mechanism	Value 渡し

---

## 戻り値

address	検索する文字列が含まれていた場合は文字列の先頭アドレス 文字列が含まれていなければ 0 sub-len がゼロの場合は src-str
---------	---

---

## JSY\$STR\_EQUAL

英大文字/小文字，全角/半角，ひらがな/カタカナ変換による文字列の比較

---

### フォーマット

```
status = JSY$STR_EQUAL ( str1, len1, str2, len2, flg )
```

---

### 引数

str1	JSY usage mechanism	比較対象文字列 1 の先頭を指すポインタ Value 渡し
len1	JSY usage mechanism	比較対象文字列 1 のバイト長 Value 渡し
str2	JSY usage mechanism	比較対象文字列 2 の先頭を指すポインタ Value 渡し
len2	JSY usage mechanism	比較対象文字列 2 のバイト長 Value 渡し
flg	JSY usage mechanism	変換フラグ 比較の前に行う変換を指定する。 bit 0     0 : 英大文字/小文字変換を行う 1 : 英大文字/小文字変換を行わない bit 1     0 : 全角/半角変換を行う 1 : 全角/半角変換を行わない bit 2     0 : ひらがな/カタカナ変換を行う 1 : ひらがな/カタカナ変換を行わない Value 渡し

---

戻り値

ret-code

1 : 文字列 1 と文字列 2 が等しい

0 : 文字列 1 と文字列 2 が等しくない

---

## JSY\$STR\_START

英大文字/小文字，全角/半角，ひらがな/カタカナ変換による文字列の比較

文字列 1 が文字列 2 で始まっているかどうかを調べます。

---

### フォーマット

```
status = JSY$STR_START ( str1, len1, str2, len2, flg )
```

---

### 引数

str1	JSY usage mechanism	比較対象文字列 1 の先頭を指すポインタ Value 渡し
len1	JSY usage mechanism	比較対象文字列 1 のバイト長 Value 渡し
str2	JSY usage mechanism	比較対象文字列 2 の先頭を指すポインタ Value 渡し
len2	JSY usage mechanism	比較対象文字列 2 のバイト長 Value 渡し
flg	JSY usage mechanism	変換フラグ 比較の前に行う変換を指定する。 bit 0     0 : 英大文字/小文字変換を行う 1 : 英大文字/小文字変換を行わない bit 1     0 : 全角/半角変換を行う 1 : 全角/半角変換を行わない bit 2     0 : ひらがな/カタカナ変換を行う 1 : ひらがな/カタカナ変換を行わない Value 渡し

---

戻り値

ret-code

- 1 : 文字列 1 が文字列 2 で始まっている
- 0 : 文字列 1 が文字列 2 で始まっていない

---

# JSY\$STR\_SEARCH

英大文字/小文字，全角/半角，ひらがな/カタカナ変換による文字列の検索

---

## フォーマット

address = JSY\$STR\_SEARCH ( *src-str*, *src-len*, *sub-str*, *sub-len*, *flg* )

---

## 引数

src-str	JSY usage	検索される文字列の先頭を指すポインタ
	mechanism	Value 渡し
src-len	JSY usage	検索される文字列のバイト長
	mechanism	Value 渡し
sub-str	JSY usage	検索する文字列の先頭を指すポインタ
	mechanism	Value 渡し
sub-len	JSY usage	検索する文字列のバイト長
	mechanism	Value 渡し
flg	JSY usage	変換フラグ 検索の前に行う変換を指定する。
		bit 0     0 : 英大文字/小文字変換を行う 1 : 英大文字/小文字変換を行わない bit 1     0 : 全角/半角変換を行う 1 : 全角/半角変換を行わない bit 2     0 : ひらがな/カタカナ変換を行う 1 : ひらがな/カタカナ変換を行わない
	mechanism	Value 渡し

---

戻り値

address

検索する文字列が含まれていた場合は文字列の先頭のアドレス, そうでなければ 0

sub-len がゼロの場合は src-str

---

## JSY\$TRIM

文字列の後の空白の切り捨て

---

### フォーマット

```
res-len = JSY$TRIM ( str, len )
```

---

### 引数

str	JSY usage	文字列の先頭を指すポインタ
	mechanism	Value 渡し
len	JSY usage	文字列のバイト長
	mechanism	Value 渡し

---

### 戻り値

res-len	空白切り捨て後の文字列の長さ
---------	----------------



---

# JSY\$TRUNC

指定された長さでの文字列の切り捨て

---

## フォーマット

```
res-len = JSY$TRUNC ( str, len, trunc-len )
```

---

## 引数

str	JSY usage	文字列の先頭を指すポインタ
	mechanism	Value 渡し
len	JSY usage	文字列のバイト長
	mechanism	Value 渡し
trunc-len	JSY usage	切り捨てを行うバイト長
	mechanism	Value 渡し

---

## 戻り値

res-len	切り捨て後の文字列の長さ
---------	--------------

---

## JSY\$LOCC

文字の検索

---

### フォーマット

```
address = JSY$LOCC ( char-code, str, len )
```

---

### 引数

char-code		
	JSY usage	検索する文字コード
	mechanism	Value 渡し
str		
	JSY usage	文字列の先頭を指すポインタ
	mechanism	Value 渡し
len		
	JSY usage	文字列のバイト長
	mechanism	Value 渡し

---

### 戻り値

address	検索する文字が含まれていた場合はそのアドレス文字が含まれていなければ 0
---------	--------------------------------------

---

## JSY\$SKPC

文字の飛び越し

---

### フォーマット

```
address = JSY$SKPC ( char-code, str, len )
```

---

### 引数

char-code		
	JSY usage	飛び越しする文字コード
	mechanism	Value 渡し
str		
	JSY usage	文字列の先頭を指すポインタ
	mechanism	Value 渡し
len		
	JSY usage	文字列のバイト長
	mechanism	Value 渡し

---

### 戻り値

address	飛び越しする文字以外の文字がある場合はそのアドレスすべて飛び越す文字と同じであれば 0
---------	---

## 4.3 文字変換ルーチン

文字変換ルーチンは、文字単位の各種変換を行うルーチン群です。

---

## JSY\$CHG\_ROM\_HALF

ローマ文字全角から半角への変換

---

### フォーマット

res-code = JSY\$CHG\_ROM\_HALF ( *char-code* )

---

### 引数

char-code		
	JSY usage	文字コード
	mechanism	Value 渡し

---

### 戻り値

res-code	変換後の文字コード
----------	-----------

---

## JSY\$CHG\_ROM\_FULL

ローマ文字半角から全角への変換

---

### フォーマット

res-code = JSY\$CHG\_ROM\_FULL ( *char-code* )

---

### 引数

char-code

JSY usage	文字コード
mechanism	Value 渡し

---

### 戻り値

res-code

変換後の文字コード

---

## JSY\$CHG\_ROM\_SIZE

ローマ文字全角/半角の相互変換

---

### フォーマット

res-code = JSY\$CHG\_ROM\_SIZE ( *char-code* )

---

### 引数

char-code

JSY usage	文字コード
mechanism	Value 渡し

---

### 戻り値

res-code

変換後の文字コード

---

## JSY\$CHG\_ROM\_LOWER

英文字半角/全角の小文字への変換

---

### フォーマット

res-code = JSY\$CHG\_ROM\_LOWER ( *char-code* )

---

### 引数

char-code

JSY usage	文字コード
mechanism	Value 渡し

---

### 戻り値

res-code

変換後の文字コード



---

## JSY\$CHG\_ROM\_UPPER

英文字半角/全角の大文字への変換

---

### フォーマット

res-code = JSY\$CHG\_ROM\_UPPER ( *char-code* )

---

### 引数

char-code		
JSY usage		文字コード
mechanism		Value 渡し

---

### 戻り値

res-code	変換後の文字コード
----------	-----------

---

## JSY\$CHG\_ROM\_CASE

英文字全角/半角の大文字/小文字の相互変換

---

### フォーマット

res-code = JSY\$CHG\_ROM\_CASE ( *char-code* )

---

### 引数

char-code		
JSY usage		文字コード
mechanism		Value 渡し

---

### 戻り値

res-code	変換後の文字コード
----------	-----------

---

## JSY\$CHG\_KANA\_HIRA

全角カタカナから全角ひらがなへの変換

---

### フォーマット

res-code = JSY\$CHG\_KANA\_HIRA ( *char-code* )

---

### 引数

char-code		
JSY usage		文字コード
mechanism		Value 渡し

---

### 戻り値

res-code	変換後の文字コード
----------	-----------

---

# JSY\$CHG\_KANA\_KATA

全角ひらがなから全角カタカナへの変換

---

## フォーマット

res-code = JSY\$CHG\_KANA\_KATA ( *char-code* )

---

## 引数

char-code		
JSY usage		文字コード
mechanism		Value 渡し

---

## 戻り値

res-code	変換後の文字コード
----------	-----------

---

## JSY\$CHG\_KANA\_KANA

全角ひらがな/全角カタカナの相互変換

---

### フォーマット

res-code = JSY\$CHG\_KANA\_KANA ( *char-code* )

---

### 引数

char-code

JSY usage	文字コード
mechanism	Value 渡し

---

### 戻り値

res-code

変換後の文字コード

---

# JSY\$CHG\_KANA\_DAKU

全角ひらがな/カタカナの濁点/半濁点処理

---

## フォーマット

status = JSY\$CHG\_KANA\_DAKU ( *char-code*, *ten-code*, *res-code* )

---

## 引数

char-code	
JSY usage	かな文字コード
mechanism	Value 渡し
ten-code	
JSY usage	濁点/半濁点文字コード
mechanism	Value 渡し
res-code	
JSY usage	変換後の文字コード 変換が行われた場合は変換後の文字コード 変換が行われなかった場合は ten-code の値
type	Longword
mechanism	Reference 渡し

---

## 戻り値

status	1 : 変換が行われた 0 : 変換が行われなかった
--------	-------------------------------

---

## JSY\$CHG\_KANA\_HALF

全角ひらがな/カタカナから半角カタカナへの変換

---

### フォーマット

status = JSY\$CHG\_KANA\_HALF ( *char-code*, *dst-str*, *res-len* )

---

### 引数

char-code		
	JSY usage	かな文字コード
	mechanism	Value 渡し
dst-str		
	JSY usage	変換結果出力領域の先頭を指すポインタ
	mechanism	Value 渡し
res-len		
	JSY usage	変換結果のバイト長
	type	Longword
	mechanism	Reference 渡し

---

### 戻り値

status	1 : 変換が行われた
	0 : 変換が行われなかった

---

# JSY\$CHG\_KANA\_FULL

半角カタカナから全角ひらがな/カタカナへの変換

---

## フォーマット

res-code = JSY\$CHG\_KANA\_FULL ( *char-code*, *flg* )

---

## 引数

char-code		
	JSY usage	文字コード
	mechanism	Value 渡し
flg		
	JSY usage	変換指示フラグ
		0 : 全角ひらがなへの変換
		1 : 全角カタカナへの変換
	mechanism	Value 渡し

---

## 戻り値

res-code	変換後の文字コード
----------	-----------



---

## JSY\$CHG\_GENERAL

パラメータ指定による、英大文字/小文字、全角/半角、ひらがな/カタカナへの変換

---

### フォーマット

res-code = JSY\$CHG\_GENERAL ( *char-code*, *flg* )

---

### 引数

char-code	
JSY usage	文字コード
mechanism	Value 渡し
flg	
JSY usage	変換フラグ 検索の前に行う変換を指定する
bit 0	0 : 英大文字/小文字変換を行う 1 : 英大文字/小文字変換を行わない
bit 1	0 : 全角/半角変換を行う 1 : 全角/半角変換を行わない
bit 2	0 : ひらがな/カタカナ変換を行う 1 : ひらがな/カタカナ変換を行わない
mechanism	Value 渡し

---

### 戻り値

res-code	変換後の文字コード
----------	-----------

---

## JSY\$CHG\_JIS\_KUTEN

JIS 区点番号から DEC 漢字コードへの変換

区点番号	DEC 漢字コード
101 ~ 9494	DEC 漢字セット
10101 ~ 19494	拡張領域

---

### フォーマット

res-code = JSY\$CHG\_JIS\_KUTEN ( *kuten-code* )

---

### 引数

kuten-code		
	JSY usage	区点番号
	mechanism	Value 渡し

---

### 戻り値

res-code	変換後の文字コード
----------	-----------

---

## JSY\$CHG\_KEISEN

数字から罫線文字への変換

```
'7' ' ' '8' ' ' '9' ' '
'4' ' ' '5' ' ' '6' ' '
'1' ' ' '2' ' ' '3' ' '
'0' ' ' '-' ' ' ' '
```

---

### フォーマット

```
res-code = JSY$CHG_KEISEN ( char-code )
```

---

### 引数

char-code		
	JSY usage	文字コード
	mechanism	Value 渡し

---

### 戻り値

res-code	変換後の文字コード
----------	-----------

## 4.4 文字列変換ルーチン

文字列変換ルーチンは、文字列の各種変換を行うルーチン群です。

---

# JSY\$TRA\_ROM\_HALF

ローマ文字全角から半角への変換

---

## フォーマット

```
status = JSY$TRA_ROM_HALF ( src-str, src-len, dst-str, dst-len, out-len )
```

---

## 引数

src-str		
	JSY usage	変換対象文字列の先頭を指すポインタ
	mechanism	Value 渡し
src-len		
	JSY usage	変換対象文字列のバイト長
	mechanism	Value 渡し
dst-str		
	JSY usage	変換結果出力領域の先頭を指すポインタ
	mechanism	Value 渡し
dst-len		
	JSY usage	変換結果出力領域のバイト長
	mechanism	Value 渡し
out-len		
	JSY usage	変換結果のバイト長
	type	Longword
	mechanism	Reference 渡し

---

## 戻り値

status	1 : 正常終了
	0 : 出力結果の切り捨てが行われた

---

# JSY\$TRA\_ROM\_FULL

ローマ文字半角から全角への変換

---

## フォーマット

status = JSY\$TRA\_ROM\_FULL ( *src-str*, *src-len*, *dst-str*, *dst-len*, *out-len* )

---

## 引数

src-str	JSY usage	変換対象文字列の先頭を指すポインタ
	mechanism	Value 渡し
src-len	JSY usage	変換対象文字列のバイト長
	mechanism	Value 渡し
dst-str	JSY usage	変換結果出力領域の先頭を指すポインタ
	mechanism	Value 渡し
dst-len	JSY usage	変換結果出力領域のバイト長
	mechanism	Value 渡し
out-len	JSY usage	変換結果のバイト長
	type	Longword
	mechanism	Reference 渡し

---

## 戻り値

status	1 : 正常終了
	0 : 出力結果の切り捨てが行われた

---

# JSY\$TRA\_ROM\_SIZE

ローマ文字全角/半角の相互変換

---

## フォーマット

status = JSY\$TRA\_ROM\_SIZE ( *src-str*, *src-len*, *dst-str*, *dst-len*, *out-len* )

---

## 引数

src-str	JSY usage	変換対象文字列の先頭を指すポインタ
	mechanism	Value 渡し
src-len	JSY usage	変換対象文字列のバイト長
	mechanism	Value 渡し
dst-str	JSY usage	変換結果出力領域の先頭を指すポインタ
	mechanism	Value 渡し
dst-len	JSY usage	変換結果出力領域のバイト長
	mechanism	Value 渡し
out-len	JSY usage	変換結果のバイト長
	type	Longword
	mechanism	Reference 渡し

---

## 戻り値

status	1	正常終了
	0	出力結果の切り捨てが行われた

---

# JSY\$TRA\_ROM\_LOWER

英文字半角/全角の小文字への変換

---

## フォーマット

status = JSY\$TRA\_ROM\_LOWER ( *src-str, src-len, dst-str, dst-len, out-len* )

---

## 引数

src-str	JSY usage	変換対象文字列の先頭を指すポインタ
	mechanism	Value 渡し
src-len	JSY usage	変換対象文字列のバイト長
	mechanism	Value 渡し
dst-str	JSY usage	変換結果出力領域の先頭を指すポインタ
	mechanism	Value 渡し
dst-len	JSY usage	変換結果出力領域のバイト長
	mechanism	Value 渡し
out-len	JSY usage	変換結果のバイト長
	type	Longword
	mechanism	Reference 渡し

---

## 戻り値

status	1 : 正常終了
	0 : 出力結果の切り捨てが行われた



---

# JSY\$TRA\_ROM\_UPPER

英文字半角/全角の大文字への変換

---

## フォーマット

```
status = JSY$TRA_ROM_UPPER ( src-str, src-len, dst-str, dst-len, out-len )
```

---

## 引数

src-str		
	JSY usage	変換対象文字列の先頭を指すポインタ
	mechanism	Value 渡し
src-len		
	JSY usage	変換対象文字列のバイト長
	mechanism	Value 渡し
dst-str		
	JSY usage	変換結果出力領域の先頭を指すポインタ
	mechanism	Value 渡し
dst-len		
	JSY usage	変換結果出力領域のバイト長
	mechanism	Value 渡し
out-len		
	JSY usage	変換結果のバイト長
	type	Longword
	mechanism	Reference 渡し

---

## 戻り値

status	1 : 正常終了
	0 : 出力結果の切り捨てが行われた

---

# JSY\$TRA\_ROM\_CASE

英文字全角/半角の大文字/小文字の相互変換

---

## フォーマット

```
status = JSY$TRA_ROM_CASE ( src-str, src-len, dst-str, dst-len, out-len )
```

---

## 引数

src-str	JSY usage	変換対象文字列の先頭を指すポインタ
	mechanism	Value 渡し
src-len	JSY usage	変換対象文字列のバイト長
	mechanism	Value 渡し
dst-str	JSY usage	変換結果出力領域の先頭を指すポインタ
	mechanism	Value 渡し
dst-len	JSY usage	変換結果出力領域のバイト長
	mechanism	Value 渡し
out-len	JSY usage	変換結果のバイト長
	type	Longword
	mechanism	Reference 渡し

---

## 戻り値

status	1 : 正常終了
	0 : 出力結果の切り捨てが行われた

---

# JSY\$TRA\_ROM\_KANA

ローマ字から全角ひらがな/カタカナへの変換

---

## フォーマット

status = JSY\$TRA\_ROM\_KANA ( src-str, src-len, #flg, dst-str, dst-len, out-len )

---

## 引数

src-str	JSY usage	変換対象文字列の先頭を指すポインタ
	mechanism	Value 渡し
src-len	JSY usage	変換対象文字列のバイト長
	mechanism	Value 渡し
flg	JSY usage	変換指示フラグ
	bit 0	0 : 全角ひらがなへ変換 1 : 全角カタカナへ変換
	mechanism	Value 渡し
dst-str	JSY usage	変換結果出力領域の先頭を指すポインタ
	mechanism	Value 渡し
dst-len	JSY usage	変換結果出力領域のバイト長
	mechanism	Value 渡し
out-len	JSY usage	変換結果のバイト長
	type	Longword
	mechanism	Reference 渡し

---

## 戻り値

status

1 : 正常終了

0 : 出力結果の切り捨てが行われた

---

# JSY\$TRA\_KANA\_HIRA

全角カタカナから全角ひらがなへの変換

---

## フォーマット

```
status = JSY$TRA_KANA_HIRA ( src-str, src-len, dst-str, dst-len, out-len )
```

---

## 引数

src-str	JSY usage	変換対象文字列の先頭を指すポインタ
	mechanism	Value 渡し
src-len	JSY usage	変換対象文字列のバイト長
	mechanism	Value 渡し
dst-str	JSY usage	変換結果出力領域の先頭を指すポインタ
	mechanism	Value 渡し
dst-len	JSY usage	変換結果出力領域のバイト長
	mechanism	Value 渡し
out-len	JSY usage	変換結果のバイト長
	type	Longword
	mechanism	Reference 渡し

---

## 戻り値

status	1 : 正常終了
	0 : 出力結果の切り捨てが行われた

---

# JSY\$TRA\_KANA\_KATA

全角ひらがなから全角カタカナへの変換

---

## フォーマット

status = JSY\$TRA\_KANA\_KATA ( *src-str*, *src-len*, *dst-str*, *dst-len*, *out-len* )

---

## 引数

src-str	JSY usage	変換対象文字列の先頭を指すポインタ
	mechanism	Value 渡し
src-len	JSY usage	変換対象文字列のバイト長
	mechanism	Value 渡し
dst-str	JSY usage	変換結果出力領域の先頭を指すポインタ
	mechanism	Value 渡し
dst-len	JSY usage	変換結果出力領域のバイト長
	mechanism	Value 渡し
out-len	JSY usage	変換結果のバイト長
	type	Longword
	mechanism	Reference 渡し

---

## 戻り値

status	1 : 正常終了
	0 : 出力結果の切り捨てが行われた

---

# JSY\$TRA\_KANA\_KANA

全角ひらがな/全角カタカナの相互変換

---

## フォーマット

```
status = JSY$TRA_KANA_KANA ( src-str, src-len, dst-str, dst-len, out-len )
```

---

## 引数

src-str		
	JSY usage	変換対象文字列の先頭を指すポインタ
	mechanism	Value 渡し
src-len		
	JSY usage	変換対象文字列のバイト長
	mechanism	Value 渡し
dst-str		
	JSY usage	変換結果出力領域の先頭を指すポインタ
	mechanism	Value 渡し
dst-len		
	JSY usage	変換結果出力領域のバイト長
	mechanism	Value 渡し
out-len		
	JSY usage	変換結果のバイト長
	type	Longword
	mechanism	Reference 渡し

---

## 戻り値

status	1 : 正常終了
	0 : 出力結果の切り捨てが行われた

---

# JSY\$TRA\_KANA\_DAKU

全角ひらがな/カタカナの濁点/半濁点処理

---

## フォーマット

status = JSY\$TRA\_KANA\_DAKU ( *src-str*, *src-len*, *dst-str*, *dst-len*, *out-len* )

---

## 引数

src-str	JSY usage	変換対象文字列の先頭を指すポインタ
	mechanism	Value 渡し
src-len	JSY usage	変換対象文字列のバイト長
	mechanism	Value 渡し
dst-str	JSY usage	変換結果出力領域の先頭を指すポインタ
	mechanism	Value 渡し
dst-len	JSY usage	変換結果出力領域のバイト長
	mechanism	Value 渡し
out-len	JSY usage	変換結果のバイト長
	type	Longword
	mechanism	Reference 渡し

---

## 戻り値

status	1 : 正常終了
	0 : 出力結果の切り捨てが行われた



---

# JSY\$TRA\_KANA\_HALF

全角ひらがな/カタカナから半角カタカナへの変換

---

## フォーマット

status = JSY\$TRA\_KANA\_HALF ( *src-str*, *src-len*, *dst-str*, *dst-len*, *out-len* )

---

## 引数

src-str		
	JSY usage	変換対象文字列の先頭を指すポインタ
	mechanism	Value 渡し
src-len		
	JSY usage	変換対象文字列のバイト長
	mechanism	Value 渡し
dst-str		
	JSY usage	変換結果出力領域の先頭を指すポインタ
	mechanism	Value 渡し
dst-len		
	JSY usage	変換結果出力領域のバイト長
	mechanism	Value 渡し
out-len		
	JSY usage	変換結果のバイト長
	type	Longword
	mechanism	Reference 渡し

---

## 戻り値

status	1 : 正常終了
	0 : 出力結果の切り捨てが行われた

---

# JSY\$TRA\_KANA\_FULL

半角カタカナから全角ひらがな/カタカナへの変換

---

## フォーマット

status = JSY\$TRA\_KANA\_FULL ( *src-str*, *src-len*, *#flg*, *dst-str*, *dst-len*, *out-len* )

---

## 引数

src-str	JSY usage	変換対象文字列の先頭を指すポインタ
	mechanism	Value 渡し
src-len	JSY usage	変換対象文字列のバイト長
	mechanism	Value 渡し
flg	JSY usage	変換指示フラグ
	bit 0	0 : 全角ひらがなへ変換 1 : 全角カタカナへ変換
	bit 1	0 : 濁点/半濁点処理を行う 1 : 濁点/半濁点処理を行わない
	mechanism	Value 渡し
dst-str	JSY usage	変換結果出力領域の先頭を指すポインタ
	mechanism	Value 渡し
dst-len	JSY usage	変換結果出力領域のバイト長
	mechanism	Value 渡し
out-len	JSY usage	変換結果のバイト長
	type	Longword
	mechanism	Reference 渡し

---

戻り値

status

1 : 正常終了

0 : 出力結果の切り捨てが行われた

---

## JSY\$TRA\_SYMBOL

記号変換  
変換規則は第 B.3 節を参照

---

### フォーマット

status = JSY\$TRA\_SYMBOL ( *src-str, src-len, dst-str, dst-len, out-len* )

---

### 引数

src-str	JSY usage	変換対象文字列の先頭を示すポインタ
	mechanism	Value 渡し
src-len	JSY usage	変換対象文字列のバイト長
	mechanism	Value 渡し
dst-str	JSY usage	変換結果出力領域の先頭を示すポインタ
	mechanism	Value 渡し
dst-len	JSY usagen	変換結果出力領域のバイト長
	mechanism	Value 渡し
out-len	JSY usagen	変換結果のバイト長
	type	Longword
	mechanism	Reference 渡し

---

### 戻り値

status	1	: 正常終了
	0	: 出力結果の切り捨てが行われた

## かな漢字変換ライブラリ

かな漢字変換ライブラリとは、かな漢字変換を行うライブラリ・ルーチン群です。

同一の機能をもったルーチンが JLB\$xxxxx と JSY\$xxxxx の 2 種類あり、JLB\$xxxxx はディスクリプタ形式で文字列の受け渡しを行うなど、VMS 標準のインターフェイスになっています。

この章では、かな漢字変換ライブラリの次の項目について説明します。

- かな漢字変換ライブラリ・ルーチン一覧 (第 5.1 節)
- 複文節変換ルーチン (第 5.2 節)
- 単語単位変換ルーチン (第 5.3 節)
- かな漢字変換辞書 (第 5.4 節)
- 注意事項および制限事項 (第 5.5 節)

### 5.1 かな漢字変換ライブラリ・ルーチン一覧

#### 複文節変換ルーチン

複文節かな漢字変換を行うルーチン群です。各ルーチンの詳細については第 5.2 節を参照してください。

JLB\$CNV_OPEN_DICTIONARY	辞書のオープン
JSY\$CNV_OPEN_DICTIONARY	辞書のオープン
JLB\$CNV_CLOSE_DICTIONARY	辞書のクローズ
JSY\$CNV_CLOSE_DICTIONARY	辞書のクローズ
JLB\$CNV_CONVERT	かな漢字変換
JSY\$CNV_CONVERT	かな漢字変換
JLB\$CNV_ROM_CONVERT	ローマ字漢字変換
JLB\$CNV_GET_KANJI	漢字文字列要求
JSY\$CNV_GET_KANJI	漢字文字列要求
JLB\$CNV_NEXT_WORD	自立語次候補
JSY\$CNV_NEXT_WORD	自立語次候補
JLB\$CNV_PREV_WORD	自立語前候補
JSY\$CNV_PREV_WORD	自立語前候補
JLB\$CNV_GET_CLAUSE	文節位置情報
JSY\$CNV_GET_CLAUSE	文節位置情報

## かな漢字変換ライブラリ

### 5.1 かな漢字変換ライブラリ・ルーチン一覧

JSY\$CNV_GET_PHONETIC_CLAUSE	読み文字列位置情報
JLB\$CNV_SHORTEN_CLAUSE	文節縮小
JSY\$CNV_SHORTEN_CLAUSE	文節縮小
JLB\$CNV_EXTEND_CLAUSE	文節伸張
JSY\$CNV_EXTEND_CLAUSE	文節伸張
JLB\$CNV_CLAUSE_HIRAGANA	文節ひらがな変換
JSY\$CNV_CLAUSE_HIRAGANA	文節ひらがな変換
JLB\$CNV_CLAUSE_KATAKANA	文節カタカナ変換
JSY\$CNV_CLAUSE_KATAKANA	文節カタカナ変換
JLB\$CNV_CLAUSE_FULL	文節全角変換
JSY\$CNV_CLAUSE_FULL	文節全角変換
JLB\$CNV_CLAUSE_HALF	文節半角変換
JSY\$CNV_CLAUSE_HALF	文節半角変換
JLB\$CNV_CLAUSE_SYMBOL	文節記号変換
JSY\$CNV_CLAUSE_SYMBOL	文節記号変換
JLB\$CNV_CLAUSE_NOCONVERT	文節無変換
JSY\$CNV_CLAUSE_NOCONVERT	文節無変換
JLB\$CNV_CLAUSE_DELETE	自立語削除
JSY\$CNV_CLAUSE_DELETE	自立語削除
JLB\$CNV_LEARN	変換確定と学習
JSY\$CNV_LEARN	変換確定と学習
JLB\$CNV_REGISTER_WORD	単語登録
JSY\$CNV_REGISTER_WORD	単語登録
JLB\$CNV_DELETE_WORD	単語削除
JSY\$CNV_DELETE_WORD	単語削除
JLB\$CNV_IO_ERROR	辞書 I/O エラー詳細情報
JSY\$CNV_IO_ERROR	辞書 I/O エラー詳細情報

#### 単語単位変換ルーチン

単語単位でかな漢字変換を行うルーチン群です。各ルーチンの詳細については第 5.3 節を参照してください。

JLB\$TRA_DICINI	辞書のオープン
JSY\$TRA_DICINI	辞書のオープン
JLB\$TRA_DICCLS	辞書のクローズ
JSY\$TRA_DICCLS	辞書のクローズ
JLB\$ENT_TANGO	単語登録
JSY\$ENT_TANGO	単語登録
JLB\$DEL_TANGO	単語削除
JSY\$DEL_TANGO	単語削除
JLB\$TRA_ROM_TANGO	ローマ字・かな単語変換
JLB\$TRA_KANA_TANGO	かな単語変換

JSY\$TRA_KANA_TANGO	かな単語変換
JLB\$TRA_TANGO_NEXT	単語次候補
JSY\$TRA_TANGO_NEXT	単語次候補
JLB\$TRA_TANGO_PREV	単語前候補
JSY\$TRA_TANGO_PREV	単語前候補
JLB\$TRA_TANGO_DONE	単語候補の決定
JSY\$TRA_TANGO_DONE	単語候補の決定

## 5.2 複文節変換ルーチン

複文節変換ルーチンは、複文節かな漢字変換を行うルーチン群です。

複文節かな漢字変換の各ルーチン (JLB\$CNV\_XXXXX および JSY\$CNV\_XXXXX) は、その戻り値として変換のステータスを返します。ディレクトリ JSY\$LIBRARY 内に、FORTRAN, PASCAL, PL/I, C, MACRO, BLISS の各言語用にステータスをシンボル定義したファイルがあります。ファイル名はそれぞれ以下のとおりです。

```
JSYDEF.FOR  FORTRAN
JSYDEF.H    C
JSYDEF.MAR  MACRO
JSYDEF.R32  BLISS(REQUIRE)
JSYDEF.L32  BLISS(LIBRARY)
```

複文節かな漢字変換ルーチンは次のような手順で呼び出すことができます。

```
JLB$CNV_OPEN_DICTIONARY( )           辞書のオープン
読み込みの入力

JLB$CNV_CONVERT( . . . )             かな漢字変換 (変換開始)

    JLB$CNV_GET_KANJI( . . . )       変換結果の要求 (変換処理)

    JLB$CNV_NEXT_WORD( . . . )       自立語次候補
または
    JLB$CNV_PREV_WORD( . . . )       自立語前候補
または
    JLB$CNV_SHORTEN_CLAUSE( . . . )  文節縮小
または
    JLB$CNV_EXTEND_CLAUSE( . . . )   文節拡大
    .
    .
    .
JLB$CNV_LEARN( )                     変換確定 (学習機能)
単語の出力
JLB$CNV_CLOSE_DICTIONARY( )         辞書のクローズ
```

---

## JLB\$CNV\_OPEN\_DICTIONARY , JSY\$CNV\_OPEN\_DICTIONARY

### 辞書のオープン

---

#### フォーマット

status = JLB\$CNV\_OPEN\_DICTIONARY

引数なし

---

#### フォーマット

status = JSY\$CNV\_OPEN\_DICTIONARY

引数なし

---

#### 説明

かな漢字変換用システム辞書および個人辞書をオープンします。文節学習辞書が存在するときは、その内容をメモリ中に読み込みます。ただし、論理名 JSY\$KOJIN\_MODE に "3" (個人辞書を使用しない) が定義されているときは、個人辞書および文節学習辞書をオープンしません。文節学習の詳細は第 5.4 節「かな漢字変換辞書」を参照してください。

---

#### 戻り値

status	
SS\$_NORMAL	正常終了
JSY\$_NOTDICFIL	システム辞書または個人辞書のフォーマットが誤っている。JLB\$CNV_IO_ERROR または JSY\$CNV_IO_ERROR ルーチン呼び出せば、システム辞書か個人辞書かの判定ができる。
JSY\$_RMSERR	辞書 I/O の最中に RMS のエラーが起きた。JLB\$CNV_IO_ERROR または JSY\$CNV_IO_ERROR ルーチン呼び出せば、エラーの詳細がわかる。



JSY\$\_CNVINTERR

変換ルーチンの内部エラー (ダイナミック・メモリ不足) が起きた。

---

## JLB\$CNV\_CLOSE\_DICTIONARY , JSY\$CNV\_CLOSE\_DICTIONARY

辞書のクローズ

---

### フォーマット

status = JLB\$CNV\_CLOSE\_DICTIONARY

引数なし

---

### フォーマット

status = JSY\$CNV\_CLOSE\_DICTIONARY

引数なし

---

### 説明

かな漢字変換用システム辞書および個人辞書をクローズし、文節学習データを文節学習辞書に書き込みます。文節学習の詳細は第 5.4 節「かな漢字変換辞書」を参照してください。

---

### 戻り値

status

SS\$\_NORMAL

正常終了

---

## JLB\$CNV\_CONVERT , JSY\$CNV\_CONVERT

かな漢字変換

---

### フォーマット

status = JLB\$CNV\_CONVERT ( *yomi-str* )

*yomi-str*

JLB usage	変換対象 (読み) 文字列
type	文字列データ
access	入力のみ
mechanism	Descriptor 渡し

---

### フォーマット

status = JSY\$CNV\_CONVERT ( *yomi-str*, *yomi-len* )

*yomi-str*

JSY usage	変換対象 (読み) 文字列
type	文字列データ
access	入力のみ
michanism	Reference 渡し

*yomi-len*

JSY usage	変換対象文字列の長さ (バイト長)
type	Longword
access	入力のみ
michanism	Value 渡し

---

### 説明

新しい読みで変換を開始します。読み文字列の内、全角ひらがな、半角・全角の数字部分を漢字文字列に変換し、変換ルーチンの内部バッファに格納します。変換した漢字文字列は JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンによりユーザに返します。入力文字列中の全角ひらがな、半角・全角の数字以外の文字は、そのまま漢字文字列に組み込まれます。

---

戻り値

status	
SS\$_NORMAL	正常終了
JSY\$_CANNOTCNV	変換できない。 読み文字列が正しくない。あるいは、語句が辞書に登録されていない。
JSY\$_CNVINTERR	変換ルーチンの内部エラーが起きた。
JSY\$_DICNOTOPN	辞書がオープンされていない。
JSY\$_PHONTOOLNG	読み文字列が長すぎる。 506 バイト (全角ひらがな 253 文字分) 以下を指定すること。
JSY\$_RMSERR	辞書 I/O の最中に RMS のエラーが起きた。 JLB\$CNV_IO_ERROR または JSY\$CNV_IO_ERROR ルーチン呼び出せば、エラーの詳細がわかる。

---

# JLB\$CNV\_ROM\_CONVERT

ローマ字漢字変換

---

## フォーマット

status = JLB\$CNV\_ROM\_CONVERT ( *yomi-str* )

---

## 引数

<i>yomi-str</i>		
JLB usage		変換対象 (読み) 文字列
type		文字列データ
access		入力のみ
michanism		Descriptor 渡し

---

## 説明

新しい読みで変換を開始します。ローマ字の読み文字列をひらがな変換 (JLB\$TRA\_ROM\_KANA) した後、漢字文字列に変換し、変換ルーチンの内部バッファに格納します。変換後の漢字文字列は JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンによりユーザに返します。入力文字列中にローマ字以外の文字がある場合、全角ひらがなと数字は漢字変換されますが、それ以外の文字はそのまま漢字文字列に組み込まれます。

---

## 戻り値

status		
SS\$_NORMAL		正常終了
JSY\$_CANNOTCNV		変換できない。
		読み文字列が正しくない。あるいは、語句が辞書に登録されていない。
JSY\$_CNVINTERR		変換ルーチンの内部エラーが起きた。

## JLB\$CNV\_ROM\_CONVERT

JSY\$\_DICNOTOPN

辞書がオープンされていない。

JSY\$\_PHONTOOLNG

読み文字列が長すぎる。

全角ひらがな換算で 253 文字分以下を指定すること。

JSY\$\_RMSERR

辞書 I/O の最中に RMS のエラーが起きた。

JLB\$CNV\_IO\_ERROR または JSY\$CNV\_IO\_ERROR  
ルーチンを呼び出せば、エラーの詳細がわかる。

---

## JLB\$CNV\_GET\_KANJI , JSY\$CNV\_GET\_KANJI

漢字文字列要求ルーチン

JLB\$CNV\_CONVERT , JLB\$CNV\_NEXT\_WORD , JLB\$CNV\_PREV\_WORD  
などが作った変換結果 , すなわち , 現在変換ルーチンの内部バッファに格納され  
ている漢字文字列 (変換後文字列) を返します。

---

### フォーマット

status = JLB\$CNV\_GET\_KANJI ( *kanji-str, clause-no* [, *out-len*] )

kanji-str

JLB usage	変換結果 (漢字文字列) 出力領域
type	文字列データ
access	出力のみ
michanism	Descriptor 渡し

clause-no

JLB usage	変換結果文字列を構成する文節の数
type	Longword
access	出力のみ
michanism	Reference 渡し

out-len

JLB usage	変換結果の長さ (バイト長)
type	Word (Unsigned)
access	出力のみ
michanism	Reference 渡し

---

### フォーマット

status = JSY\$CNV\_GET\_KANJI ( *kanji-str, kanji-len, out-len, clause-no* )

kanji-str

JSY usage	変換結果 (漢字文字列) 出力領域
type	文字列データ
access	出力のみ
michanism	Value 渡し

kanji-len

## JLB\$CNV\_GET\_KANJI , JSY\$CNV\_GET\_KANJI

	JSY usage	変換結果出力領域の長さ (バイト長)
	type	Longword
	access	入力のみ
	michanism	Value 渡し
out-len		
	JSY usage	変換結果の長さ (バイト長)
	type	Longword
	access	出力のみ
	michanism	Reference 渡し
clause-no		
	JSY usage	変換結果文字列を構成する文節の数
	type	Longword
	access	出力のみ
	michanism	Reference 渡し

---

## 戻り値

status	
SS\$_NORMAL	正常終了
JSY\$_INVCALL	ルーチンの呼び出し順序が正しくない。 変換開始ルーチンを呼び出した後に呼び出すこと。
JSY\$_STRTRU	出力文字列の切り捨てが行われた。



---

## JLB\$CNV\_NEXT\_WORD , JSY\$CNV\_NEXT\_WORD

### 自立語次候補ルーチン

---

#### フォーマット

status = JLB\$CNV\_NEXT\_WORD ( *clause-no* )

clause-no

JLB usage	次候補漢字列を要求する文節の番号
type	Longword
access	入力のみ
mechanism	Reference 渡し
	1 以上 JLB\$CNV_GET_KANJI または JSY\$CNV_GET_KANJI ルーチンで返った文節数以下を指定する。

---

#### フォーマット

status = JSY\$CNV\_NEXT\_WORD ( *clause-no* )

clause-no

JSY usage	次候補漢字列を要求する文節の番号
type	Longword
access	入力のみ
mechanism	Value 渡し
	1 以上 JLB\$CNV_GET_KANJI または JSY\$CNV_GET_KANJI ルーチンで返った文節数以下を指定する。

---

#### 説明

引数で指定された文節の自立語の次候補を採用して、新しい漢字文字列を作り、変換ルーチンの内部バッファに格納します。JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンにより新しい漢字文字列を返します。漢字文字列の文節の数と区切りは変化しません。また、最後の候補のときに次候補が要求されると、JSY\$\_ROUNDK のステータスで、先頭の候補を返します。

---

戻り値

status	
SS\$_NORMAL	正常終了
JSY\$_ROUNDK	先頭の候補に戻った。(正常終了)
JSY\$_CNVINTERR	変換ルーチンの内部エラーが起きた。
JSY\$_INVCALL	ルーチンの呼び出し順序が正しくない。 変換開始ルーチンを呼び出した後に呼び出すこと。
JSY\$_INVCLSNUM	文節番号の指定が正しくない。
JSY\$_RMSERR	辞書 I/O の最中に RMS のエラーが起きた。 JLB\$CNV_IO_ERROR または JSY\$CNV_IO_ERROR ルーチンを呼び出せば、エラーの詳細がわかる。

---

## JLB\$CNV\_PREV\_WORD , JSY\$CNV\_PREV\_WORD

### 自立語前候補ルーチン

---

#### フォーマット

status = JLB\$CNV\_PREV\_WORD ( *clause-no* )

clause-no

JLB usage	前候補漢字列を要求する文節の番号
type	Longword
access	入力のみ
mechanism	Reference 渡し
	1 以上 JLB\$CNV_GET_KANJI または JSY\$CNV_GET_KANJI ルーチンで返った文節数以下を指定する。

---

#### フォーマット

status = JSY\$CNV\_PREV\_WORD ( *clause-no* )

clause-no

JSY usage	前候補漢字列を要求する文節の番号
type	Longword
access	入力のみ
mechanism	Value 渡し
	1 以上 JLB\$CNV_GET_KANJI または JSY\$CNV_GET_KANJI ルーチンで返った文節数以下を指定する。

---

#### 説明

引数で指定された文節の自立語の前候補を採用して、新しい漢字文字列を作り、変換ルーチンの内部バッファに格納します。JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンにより新しい漢字文字列を返します。漢字文字列の文節の数と区切りは変化しません。

## 戻り値

status	
SS\$_NORMAL	正常終了
JSY\$_CNVINTERR	変換ルーチンの内部エラーが起きた。
JSY\$_INVCALL	ルーチンの呼び出し順序が正しくない。 変換開始ルーチンを呼び出した後に呼び出すこと。
JSY\$_INVCLSNUM	文節番号の指定が正しくない。
JSY\$_RMSERR	辞書 I/O の最中に RMS のエラーが起きた。 JLB\$CNV_IO_ERROR または JSY\$CNV_IO_ERROR ルーチンを呼び出せば、エラーの詳細がわかる。

---

## JLB\$CNV\_GET\_CLAUSE , JSY\$CNV\_GET\_CLAUSE

---

### 文節位置情報ルーチン

引数で指定した文節の各種情報を返します。

---

### フォーマット

status = JLB\$CNV\_GET\_CLAUSE ( *clause-no, offset, length, flag* )

#### clause-no

JLB usage	文節番号
type	Longword
access	入力のみ
mechanism	Reference 渡し
	1 以上 JLB\$CNV_GET_KANJI または JSY\$CNV_GET_KANJI ルーチンで返った文節数以下を指定する。

#### offset

JLB usage	指定した文節の位置
type	Longword
access	出力のみ
mechanism	Reference 渡し
	指定した文節の位置情報を JLB\$CNV_GET_KANJI または JSY\$CNV_GET_KANJI で返った漢字文字列の先頭からのオフセット (バイト単位) で返す。

#### length

JLB usage	文節の長さ (バイト長)
type	Word (Unsigned)
access	入力のみ
mechanism	Reference 渡し

#### flag

JLB usage	変換情報
type	Longword
access	出力のみ
mechanism	Reference 渡し
	値は常に 1 ( V4.5 より , この引数の内容は意味を持ちません)。

---

**フォーマット**

status = JSY\$CNV\_GET\_CLAUSE ( clause-no, offset, length, flag )

**clause-no**

JSY usage	文節番号
type	Longword
access	入力のみ
mechanism	Value 渡し

1 以上 JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンで返った文節数以下を指定する。

**offset**

JSY usage	指定した文節の位置
type	Longword
access	出力のみ
mechanism	Reference 渡し

指定した文節の位置情報を JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI で返った漢字文字列の先頭からのオフセット (バイト単位) で返す。

**length**

JSY usage	文節の長さ (バイト長)
type	Longword
access	出力のみ
mechanism	Reference 渡し

**flag**

JSY usage	変換情報
type	Longword
access	出力のみ
mechanism	Reference 渡し

値は常に 1 ( V4.5 より , この引数の内容は意味を持ちません)。

---

**戻り値**
**status**

SS\$_NORMAL	正常終了
JSY\$_INVCALL	ルーチンの呼び出し順序が正しくない。 変換開始ルーチンを呼び出した後に呼び出すこと。
JSY\$_INVCLSNM	文節番号の指定が正しくない。

---

## JSY\$CNV\_GET\_PHONETIC\_CLAUSE

文節に対応する入力文字列の位置情報ルーチン

引数で指定した文節に対応する入力文字列の位置情報を返します。指定した文節の読みが得られます。

---

### フォーマット

status = JSY\$CNV\_GET\_PHONETIC\_CLAUSE ( clause-no, offset, length )

---

### 引数

#### clause-no

JSY usage	文節番号
type	Longword
access	入力のみ
mechanism	Value 渡し
	1 以上 JLB\$CNV_GET_KANJI または JSY\$CNV_GET_KANJI ルーチンで返った文節数以下を指定する。

#### offset

JSY usage	指定した文節に対応する入力文字列の位置
type	Longword
access	出力のみ
mechanism	Reference 渡し
	JLB\$CNV_CONVERT または JSY\$CNV_CONVERT ルーチンに引き渡した読み文字列の先頭からのオフセット (バイト単位) で返す。
	JLB\$CNV_ROM_CONVERT または JSY\$CNV_ROM_CONVERT ルーチンを使用していた場合は、引き渡したローマ字列をかな変換した (JLB\$TRA_ROM_KANA) 結果を、入力文字列とみなし、このかな文字列の先頭からのオフセット (バイト単位) で返す。

#### length

JSY usage	読み文字列の長さ (バイト長)
type	Longword
access	出力のみ
mechanism	Reference 渡し

戻り値

status	
SS\$_NORMAL	正常終了
JSY\$_INVCALL	ルーチンの呼び出し順序が正しくない。 変換開始ルーチンを呼び出した後に呼び出すこと。
JSY\$_INVCLSNUM	文節番号の指定が正しくない。



---

## JLB\$CNV\_SHORTEN\_CLAUSE , JSY\$CNV\_SHORTEN\_CLAUSE

文節縮小ルーチン

---

### フォーマット

status = JLB\$CNV\_SHORTEN\_CLAUSE ( *clause-no* )

clause-no

JLB usage	文節番号
type	Longword
access	入力のみ
mechanism	Reference 渡し

1 以上 JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンで返った文節数以下を指定する。

---

### フォーマット

status = JSY\$CNV\_SHORTEN\_CLAUSE ( *clause-no* )

clause-no

JSY usage	文節番号
type	Longword
access	入力のみ
mechanism	Value 渡し

1 以上 JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンで返った文節数以下を指定する。

---

### 説明

引数で指定された文節の長さ縮小し、文法解析をやり直して新しい漢字文字列を作ります。指定された文節以降の文節は、長さや区切りが変わることもあります。JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンにより、新しい漢字文字列を返します。

---

**戻り値**

status	
SS\$_NORMAL	正常終了
JSY\$_CANNOTCNV	文節を縮小すると変換ができない。
JSY\$_CNVINTERR	変換ルーチンの内部エラーが起きた。
JSY\$_INVCALL	ルーチンの呼び出し順序が正しくない。 変換開始ルーチンを呼び出した後に呼び出すこと。
JSY\$_INVCLSNUM	文節番号の指定が正しくない。
JSY\$_RMSERR	辞書 I/O の最中に RMS のエラーが起きた。 JLB\$CNV_IO_ERROR または JSY\$CNV_IO_ERROR ルーチンを呼び出せば、エラーの詳細がわかる。

---

## JLB\$CNV\_EXTEND\_CLAUSE , JSY\$CNV\_EXTEND\_CLAUSE

文節長伸長ルーチン

---

### フォーマット

status = JLB\$CNV\_EXTEND\_CLAUSE ( *clause-no* )

clause-no

JLB usage	文節番号
type	Longword
access	入力のみ
mechanism	Reference 渡し

1 以上 JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンで返った文節数以下を指定する。

---

### フォーマット

status = JSY\$CNV\_EXTEND\_CLAUSE ( *clause-no* )

clause-no

JSY usage	文節番号
type	Longword
access	入力のみ
mechanism	Value 渡し

1 以上 JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンで返った文節数以下を指定する。

---

### 説明

引数で指定された文節の長さを長くします。すでに文節が縮小されていたときは、直前の文節の長さに戻します。指定された文節以降の文節は、長さや区切りが変わることもあります。JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンにより新しい漢字文字列を返します。

---

戻り値

status	
SS\$_NORMAL	正常終了
JSY\$_CANNOTCNV	文節を縮小すると変換ができない。
JSY\$_CNVINTERR	変換ルーチンの内部エラーが起きた。
JSY\$_INVCALL	ルーチンの呼び出し順序が正しくない。 変換開始ルーチンを呼び出した後に呼び出すこと。
JSY\$_INVCLSNUM	文節番号の指定が正しくない。
JSY\$_RMSERR	辞書 I/O の最中に RMS のエラーが起きた。 JLB\$CNV_IO_ERROR または JSY\$CNV_IO_ERROR ルーチンを呼び出せば、エラーの詳細がわかる。

---

## JLB\$CNV\_CLAUSE\_HIRAGANA , JSY\$CNV\_CLAUSE\_HIRAGANA

文節ひらがな変換ルーチン

---

### フォーマット

status = JLB\$CNV\_CLAUSE\_HIRAGANA ( *clause-no* )

clause-no

JLB usage	文節番号
type	Longword
access	入力のみ
mechanism	Reference 渡し

1 以上 JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンで返った文節数以下を指定する。

---

### フォーマット

status = JSY\$CNV\_CLAUSE\_HIRAGANA ( *clause-no* )

clause-no

JSY usage	文節番号
type	Longword
access	入力のみ
mechanism	Value 渡し

1 以上 JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンで返った文節数以下を指定する。

---

### 説明

指定された文節の自立語をひらがなに変換します。変換結果は変換ルーチンの内部バッファに格納し、JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンによりユーザに返します。

## 戻り値

status	
SS\$_NORMAL	正常終了
JSY\$_INVCALL	ルーチンの呼び出し順序が正しくない。 変換開始ルーチンを呼び出した後に呼び出すこと。
JSY\$_INVCLSNUM	文節番号の指定が正しくない。

---

## JLB\$CNV\_CLAUSE\_KATAKANA , JSY\$CNV\_CLAUSE\_KATAKANA

文節カタカナ変換ルーチン

---

### フォーマット

status = JLB\$CNV\_CLAUSE\_KATAKANA ( *clause-no* )

clause-no

JLB usage	文節番号
type	Longword
access	入力のみ
mechanism	Reference 渡し

1 以上 JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンで返った文節数以下を指定する。

---

### フォーマット

status = JSY\$CNV\_CLAUSE\_KATAKANA ( *clause-no* )

clause-no

JSY usage	文節番号
type	Longword
access	入力のみ
mechanism	Value 渡し

1 以上 JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンで返った文節数以下を指定する。

---

### 説明

指定された文節をカタカナに変換します。このルーチンが一回目に呼ばれたときは、文節の自立語のみをカタカナに変換し、2 回目に呼ばれたときは文節全体をカタカナ変換します。変換結果は変換ルーチンの内部バッファに格納し、JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンによりユーザに返します。

たとえば、"ねこにこばん"を漢字変換すると最初は"猫に/小判"(/は文節の区切り)となります。ここで第一文節を文節カタカナ変換すると"ネコに/小判"となり、再度第一文節を文節カタカナ変換すると"ネコニ/小判"となります。

## 戻り値

status	
SS\$_NORMAL	正常終了
JSY\$_INVCALL	ルーチンの呼び出しが正しくない。 変換開始ルーチンを呼び出した後に呼び出すこと。
JSY\$_INVCLSNUM	文節番号の指定順序が正しくない。
JSY\$_CNVINTERR	変換ルーチンの内部エラーが起きた。



---

## JLB\$CNV\_CLAUSE\_FULL , JSY\$CNV\_CLAUSE\_FULL

### 文節全角変換ルーチン

指定された文節を全角文字に変換します。変換結果は変換ルーチンの内部バッファに格納し、JLB\$CNV\_GET\_KANJIまたはJSY\$CNV\_GET\_KANJIルーチンによりユーザに返します。

---

### フォーマット

status = JLB\$CNV\_CLAUSE\_FULL ( *clause-no* )

clause-no

JLB usage	文節番号
type	Longword
access	入力のみ
mechanism	Reference 渡し

1 以上 JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンで返った文節数以下を指定する。

---

### フォーマット

status = JSY\$CNV\_CLAUSE\_FULL ( *clause-no* )

clause-no

JSY usage	文節番号
type	Longword
access	入力のみ
mechanism	Value 渡し

1 以上 JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンで返った文節数以下を指定する。

---

### 戻り値

status	
SS\$_NORMAL	正常終了
JSY\$_INVCALL	ルーチンの呼び出し順序が正しくない。

JLB\$CNV\_CLAUSE\_FULL , JSY\$CNV\_CLAUSE\_FULL

JSY\$\_INVCLSNUM

JSY\$\_CNVINTERR

変換開始ルーチン呼び出した後に呼び出すこと。

文節番号の指定が正しくない。

変換ルーチンの内部エラーが起きた。

---

## JLB\$CNV\_CLAUSE\_HALF , JSY\$CNV\_CLAUSE\_HALF

### 文節半角変換ルーチン

指定された文節を半角文字に変換します。変換結果は変換ルーチンの内部バッファに格納し、JLB\$CNV\_GET\_KANJIまたはJSY\$CNV\_GET\_KANJIルーチンによりユーザに返します。

---

### フォーマット

status = JLB\$CNV\_CLAUSE\_HALF ( *clause-no* )

clause-no

JLB usage	文節番号
type	Longword
access	入力のみ
mechanism	Reference 渡し
	1 以上 JLB\$CNV_GET_KANJI または JSY\$CNV_GET_KANJI ルーチンで返った文節数以下を指定する。

---

### フォーマット

status = JSY\$CNV\_CLAUSE\_HALF ( *clause-no* )

clause-no

JSY usage	文節番号
type	Longword
access	入力のみ
mechanism	Value 渡し
	1 以上 JLB\$CNV_GET_KANJI または JSY\$CNV_GET_KANJI ルーチンで返った文節数以下を指定する。

---

### 戻り値

status	
SS\$_NORMAL	正常終了

JLB\$CNV\_CLAUSE\_HALF , JSY\$CNV\_CLAUSE\_HALF

JSY\$\_INVCALL

ルーチンの呼び出し順序が正しくない。

変換開始ルーチンを呼び出した後に呼び出すこと。

JSY\$\_INVCLSNUM

文節番号の指定が正しくない。

JSY\$\_CNVINIERR

変換ルーチンの内部エラーが起きた。

---

## JLB\$CNV\_CLAUSE\_SYMBOL , JSY\$CNV\_CLAUSE\_SYMBOL

### 文節記号変換ルーチン

指定された文節を特殊記号に変換します。変換結果は変換ルーチンの内部バッファに格納し、JLB\$CNV\_GET\_KANJIまたはJSY\$CNV\_GET\_KANJIルーチンによりユーザに返します。

---

### フォーマット

status = JLB\$CNV\_CLAUSE\_SYMBOL ( *clause-no* )

clause-no

JLB usage	文節番号
type	Longword
access	入力のみ
mechanism	Reference 渡し
	1 以上 JLB\$CNV_GET_KANJI または JSY\$CNV_GET_KANJI ルーチンで返った文節数以下を指定する。

---

### フォーマット

status = JSY\$CNV\_CLAUSE\_SYMBOL ( *clause-no* )

clause-no

JSY usage	文節番号
type	Longword
access	入力のみ
mechanism	Value 渡し
	1 以上 JLB\$CNV_GET_KANJI または JSY\$CNV_GET_KANJI ルーチンで返った文節数以下を指定する。

---

### 戻り値

status	
SS\$_NORMAL	正常終了

JLB\$CNV\_CLAUSE\_SYMBOL , JSY\$CNV\_CLAUSE\_SYMBOL

JSY\$\_INVCALL

ルーチンの呼び出し順序が正しくない。

変換開始ルーチンを呼び出した後に呼び出すこと。

JSY\$\_INVCLSNUM

文節番号の指定が正しくない。

JSY\$\_CNVINIERR

変換ルーチンの内部エラーが起きた。

---

## JLB\$CNV\_CLAUSE\_NOCONVERT , JSY\$CNV\_CLAUSE\_NOCONVERT

文節無変換ルーチン

---

### フォーマット

status = JLB\$CNV\_CLAUSE\_NOCONVERT ( *clause-no* )

clause-no

JLB usage	文節番号
type	Longword
access	入力のみ
mechanism	Reference 渡し

1 以上 JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンで返った文節数以下を指定する。

---

### フォーマット

status = JSY\$CNV\_CLAUSE\_NOCONVERT ( *clause-no* )

clause-no

JSY usage	文節番号
type	Longword
access	入力のみ
mechanism	Value 渡し

1 以上 JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンで返った文節数以下を指定する。

---

### 説明

指定された文節の読み文字列をそのまま漢字文字列に組み込みます。ただし、最初に JLB\$TRA\_ROM\_CONVERT または JSY\$TRA\_ROM\_CONVERT ルーチンが使用されていた場合は、ローマ字をひらがな変換した結果を漢字文字列に組み込みます。結果は変換ルーチンの内部バッファに格納し、JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンによりユーザに返します。

## 戻り値

status	
SS\$_NORMAL	正常終了
JSY\$_INVCALL	ルーチンの呼び出し順序が正しくない。 変換開始ルーチンを呼び出した後に呼び出すこと。
JSY\$_INVCLSNUM	文節番号の指定が正しくない。
JSY\$_CNVINIERR	変換ルーチンの内部エラーが起きた。



---

## JLB\$CNV\_CLAUSE\_DELETE , JSY\$CNV\_CLAUSE\_DELETE

### 自立語削除ルーチン

---

#### フォーマット

status = JLB\$CNV\_CLAUSE\_DELETE ( *clause-no* )

clause-no

JLB usage	文節番号
type	Longword
access	入力のみ
mechanism	Reference 渡し

1 以上 JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンで返った文節数以下を指定する。

---

#### フォーマット

status = JSY\$CNV\_CLAUSE\_DELETE ( *clause-no* )

clause-no

JSY usage	文節番号
type	Longword
access	入力のみ
mechanism	Value 渡し

1 以上 JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI ルーチンで返った文節数以下を指定する。

---

#### 説明

指定された文節の自立語を個人辞書から削除します。この後、JLB\$CNV\_GET\_KANJI または JSY\$CNV\_GET\_KANJI で漢字文字列を得た場合、削除した自立語部分には読みの長さ分、半角の@ (単価記号) が入れられます。

---

戻り値

status	
SS\$_NORMAL	正常終了
JSY\$_INVCALL	ルーチンの呼び出し順序が正しくない。 変換開始ルーチンを呼び出した後に呼び出すこと。
JSY\$_INVCLSNUM	文節番号の指定が正しくない。
JSY\$_RMSERR	辞書 I/O の最中に RMS のエラーが起きた。 JLB\$CNV_IO_ERROR または JSY\$CNV_IO_ERROR ルーチンを呼び出せば、エラーの詳細がわかる。
JSY\$_WRDNOTFND	削除しようとした語句が個人辞書に存在しない。
JSY\$_CNVINIERR	変換ルーチンの内部エラーが起きた。

---

## JLB\$CNV\_LEARN , JSY\$CNV\_LEARN

変換確定ルーチン

---

### フォーマット

status = JLB\$CNV\_LEARN

引数なし

---

### フォーマット

status = JSY\$CNV\_LEARN

引数なし

---

### 説明

変換を終了し、変換漢字列内の単語を個人辞書に学習します。また、メモリ中の文節学習データの更新を行います。ただし、文節学習データの文節学習辞書への書き込みは行われません。個人辞書への学習および文節学習の詳細は第 5.4 節「かな漢字変換辞書」を参照してください。

---

### 戻り値

status

SS\$\_NORMAL

JSY\$\_INDEXFULL

JSY\$\_INVCALL

正常終了

個人辞書ファイルにこれ以上単語を登録できない。個人辞書編集ユーティリティで、辞書を再構成するするか、または論理名 JSY\$KOJIN に新しいファイルを割り当てること。

個人辞書に登録可能な単語数については、第 5.4 節「かな漢字変換辞書」を参照のこと。

ルーチンの呼び出し順序が正しくない。

変換開始ルーチンを呼び出した後に呼び出すこと。

JLB\$CNV\_LEARN , JSY\$CNV\_LEARN

JSY\$\_RMSERR

辞書 I/O の最中に RMS のエラーが起きた。JLB\$CNV\_  
IO\_ERROR または JSY\$CNV\_IO\_ERROR ルーチンと呼  
び出せば、エラーの詳細がわかる。

JSY\$\_CNVINIERR

変換ルーチンの内部エラーが起きた。

---

## JLB\$CNV\_REGISTER\_WORD , JSY\$CNV\_REGISTER\_WORD

### 単語登録ルーチン

個人辞書にユーザ定義の単語を登録します。

---

### フォーマット

status = JLB\$CNV\_REGISTER\_WORD ( *yomi-str*, *kanji-str* )

#### yomi-str

JLB usage	登録する単語の読み
type	文字列データ
access	入力のみ
mechanism	Descriptor 渡し
	全角ひらがなで 16 文字以内を指定する。ただし、濁点・半濁点も 1 文字として数える。

#### kanji-str

JLB usage	登録する単語の文字列
type	文字列データ
access	入力のみ
mechanism	Descriptor 渡し
	80 バイト (漢字 40 文字分) 以内を指定する。 文字列内には全角・半角の各種文字を含めることができる。

---

### フォーマット

status = JSY\$CNV\_REGISTER\_WORD ( *yomi-str*, *yomi-len*, *kanji-str*, *kanji-len* )

#### yomi-str

JSY usage	登録する単語の読みを格納している文字列
type	文字列データ
access	入力のみ
mechanism	Reference 渡し
	全角ひらがなで 16 文字以内を指定する。ただし、濁点・半濁点も 1 文字として数える。

#### yomi-len

	JSY usage	読み文字列の長さ (バイト長)
	type	Longword
	access	入力のみ
	mechanism	Value 渡し
		全角ひらがなで 16 文字以内を指定する。ただし、濁点・半濁点も 1 文字として数える。
kanji-str		
	JSY usage	登録する単語の文字列
	type	文字列データ
	access	入力のみ
	mechanism	Reference 渡し
		80 バイト (漢字 40 文字分) 以内を指定する。 文字列内には全角・半角の各種文字を含めることができる。
kanji-len		
	JSY usage	単語の文字列の長さ (バイト長)
	type	Longword
	access	入力のみ
	mechanism	Value 渡し

---

## 戻り値

status		
SS\$_NORMAL		正常終了
SS\$_DICNOTOPN		辞書がオープンされていない。
JSY\$_INDEXFULL		個人辞書ファイルにこれ以上単語を登録できない。個人辞書編集ユーティリティで、辞書を再構成するするか、または論理名 JSY\$KOJIN に新しいファイルを割り当てること。
		個人辞書に登録可能な単語数については、第 5.4 節「かな漢字変換辞書」を参照のこと。
JSY\$_INVPHONSTR		指定した読みが正しくない。
		読み文字列は全角ひらがなでなければならない。
JSY\$_PHONTOOLNG		指定した読み文字列が長すぎる。
		辞書に登録する 1 つの語句に対して全角ひらがなで 16 文字以下でなければならない。ただし、濁点・半濁点も 1 文字として数える。
JSY\$_RMSERR		辞書 I/O の最中に RMS のエラーが起きた。JLB\$CNV_IO_ERROR または JSY\$CNV_IO_ERROR ルーチンを呼び出せば、エラーの詳細がわかる。
JSY\$_WRDTOOLNG		登録しようとした単語の文字列が長すぎる。単語の文字列は 80 バイト (漢字 40 文字分) 以下でなければならない。
JSY\$_CNVINIERR		変換ルーチンの内部エラーが起きた。

---

## JLB\$CNV\_DELETE\_WORD , JSY\$CNV\_DELETE\_WORD

### 単語削除ルーチン

指定された単語を個人辞書から削除します。

---

### フォーマット

status = JLB\$CNV\_DELETE\_WORD ( *yomi-str, kanji-str* )

#### yomi-str

JLB usage	削除する単語の読み
type	文字列データ
access	入力のみ
mechanism	Descriptor 渡し
	全角ひらがなで 16 文字以内を指定する。ただし、濁点・半濁点も 1 文字として数える。

#### kanji-str

JLB usage	削除する単語の文字列
type	文字列データ
access	入力のみ
mechanism	Descriptor 渡し

---

### フォーマット

status = JSY\$CNV\_DELETE\_WORD ( *yomi-str, yomi-len, kanji-str, kanji-len* )

#### yomi-str

JSY usage	削除する単語の読みを格納している文字列
type	文字列データ
access	入力のみ
mechanism	Reference 渡し
	全角ひらがなで 16 文字以内を指定する。ただし、濁点・半濁点も 1 文字として数える。

#### yomi-len

JSY usage	読み文字列の長さ (バイト長)
type	Longword

	access	入力のみ
	mechanism	Value 渡し
kanji-str		
	JSY usage	削除する単語の文字列
	type	文字列データ
	access	入力のみ
	mechanism	Value 渡し
kanji-len		
	JSY usage	読み文字列の長さ (バイト長)
	type	Longword
	access	入力のみ
	mechanism	Value 渡し

---

## 戻り値

status	
SS\$_NORMAL	正常終了
SS\$_DICNOTOPN	辞書がオープンされていない。
JSY\$_INVPHONSTR	指定した読みが正しくない。 読み文字列は全角ひらがなでなければならない。
JSY\$_PHONTOOLNG	指定した読み文字列が長すぎる。全角ひらがなで 16 文字以下でなければならない。ただし、濁点・半濁点も 1 文字として数える。
JSY\$_RMSERR	辞書 I/O の最中に RMS のエラーが起きた。JLB\$CNV_IO_ERROR または JSY\$CNV_IO_ERROR ルーチン呼び出せば、エラーの詳細がわかる。
JSY\$_WRDNOTFND	削除しようとした語句が個人辞書に存在しない。
JSY\$_WRDTOOLNG	削除しようとした単語の文字列が長すぎる。単語の文字列は 80 バイト (漢字 40 文字分) 以下でなければならない。
JSY\$_CNVINIERR	変換ルーチンの内部エラーが起きた。



---

## JLB\$CNV\_IO\_ERROR , JSY\$CNV\_IO\_ERROR

辞書 I/O ステータス・ルーチン

---

### フォーマット

status = JLB\$CNV\_IO\_ERROR ( *file-name*, *dic-type*, *rms-sts*, *rms-stv*, *name-len* )

#### file-name

JLB usage	ファイル名の格納領域
type	文字列データ
access	出力のみ
mechanism	Descriptor 渡し エラーの発生したファイルの名前が返る。

#### dic-type

JLB usage	辞書の種類
type	Longword
access	出力のみ
mechanism	Reference 渡し 1 = システム辞書 2 = 個人辞書

#### rms-sts

JLB usage	RMS ステータス 1
type	Longword
access	出力のみ
mechanism	Reference 渡し

#### rms-stv

JLB usage	RMS ステータス 2
type	Longword
access	出力のみ
mechanism	Reference 渡し

#### name-len

JLB usage	ファイル名の長さ
type	Word (Unsigned)
access	出力のみ
mechanism	Reference 渡し

---

**フォーマット**

status = JSY\$CNV\_IO\_ERROR ( *file-name*, *file-len*, *name-len*, *dic-type*, *rms-sts* )

**file-name**

JSY usage	ファイル名の格納領域
type	文字列データ
access	出力のみ
mechanism	Reference 渡し

エラーの発生したファイルの名前が返る。

**file-len**

JSY usage	ファイル名格納領域のサイズ
type	Longword
access	入力のみ
mechanism	Value 渡し

**name-len**

JSY usage	ファイル名の長さ
type	Longword
access	出力のみ
mechanism	Reference 渡し

**dic-type**

JSY usage	辞書の種類
type	Longword
access	出力のみ
mechanism	Reference 渡し

1 = システム辞書  
2 = 個人辞書

**rms-sts**

JSY usage	RMS ステータス
type	2 Longword
access	出力のみ
mechanism	Reference 渡し

---

**説明**

直前の JLB\$CNV\_xxxxxx または JSY\$CNV\_xxxxxx 呼び出しにより発生した辞書 I/O エラーの詳細情報を返します。JLB\$CNV\_xxxxxx および JSY\$CNV\_xxxxxx ルーチンでは辞書 I/O エラーが発生した場合、JSY\$\_RMSERR のステータスを返します。その場合にこのルーチンを読み出すことにより、詳細なエラー情報を知ることができます。エラーが発生していないときは rms-sts に RMS\$\_NORMAL のステータスがセットされますが、辞書のファイル名などは不定です。

---

戻り値

status

SS\$\_NORMAL

JSY\$\_STRTRU

正常終了

出力文字列の切り捨てが行われた。

---

## 5.3 単語単位変換ルーチン

単語単位変換ルーチンは、単語単位かな漢字変換を行うルーチン群です。

かな漢字変換ルーチンは次のような手順で呼び出すことができます。

JLB\$TRA_DICINI	辞書のオープン
読みの入力	
JLB\$TRA_ROM_KANA	ローマ字かな変換
JLB\$TRA_KANA_TANGO(...)	かな単語変換
または	
JLB\$TRA_ROM_TANGO(...)	ローマ字単語変換
JLB\$TRA_TANGO_NEXT(...)	単語次候補
または	
JLB\$TRA_TANGO_PREV(...)	単語前候補
または	
JLB\$DEL_TANGO	単語削除
JLB\$TRA_TANGO_DONE	単語の決定 (学習)
単語の出力	
JLB\$TRA_DICCLS	辞書のクローズ

---

## JLB\$TRA\_DICINI , JSY\$TRA\_DICINI

辞書のオープン

単語辞書および個人辞書をオープンします。

---

### フォーマット

status = JLB\$TRA\_DICINI

引数なし

戻り値

status	SS\$_NORMAL	正常終了
--------	-------------	------

---

### フォーマット

status = JSY\$TRA\_DICINI

引数なし

戻り値

status	SS\$_NORMAL	正常終了
	0	変換ルーチンの内部エラーが起きた。

---

#### 注意

RMS のエラーが発生した時は、その RMS ステータスを用い、ルーチン内部で LIB\$STOP を実行して終了します。

---

---

## JLB\$TRA\_DICCLS , JSY\$TRA\_DICCLS

辞書のクローズ

単語辞書および個人辞書をクローズします。

---

### フォーマット

status = JLB\$TRA\_DICCLS

引数なし

---

### フォーマット

status = JSY\$TRA\_DICCLS

引数なし

---

### 戻り値

status

SS\$\_NORMAL

正常終了

---

## JLB\$ENT\_TANGO , JSY\$ENT\_TANGO

### 単語登録

---

#### フォーマット

status = JLB\$ENT\_TANGO ( *kana-str, tango-str* )

##### kana-str

JLB usage	読み文字列 (全角ひらがな/全角カタカナ)
type	文字列データ
access	入力のみ
mechanism	Descriptor 渡し

##### tango-str

JLB usage	登録する単語文字列
type	文字列データ
access	入力のみ
mechanism	Descriptor 渡し

---

#### フォーマット

status = JSY\$ENT\_TANGO ( *kana-str, kana-len, tango-str, tango-len* )

##### kana-str

JSY usage	読み文字列 (全角ひらがな/全角カタカナ)
type	文字列データ
access	入力のみ
mechanism	Reference 渡し

##### kana-len

JSY usage	読み文字列のバイト長
type	Longword
access	入力のみ
mechanism	Value 渡し

##### tango-str

JSY usage	登録する単語文字列
type	文字列データ
access	入力のみ
mechanism	Reference 渡し

tango-len		
JSY usage		登録する単語文字列のバイト長
type		Longword
access		入力のみ
mechanism		Value 渡し

---

## 説明

新しい単語を個人辞書に登録します。読み文字列は全角ひらがなまたは全角カタカナで 16 文字以内でなければなりません。ただし、濁点/半濁点も 1 文字として数えます。単語文字列は最長 80 バイト (漢字 40 文字分) まで登録できます。

---

## 戻り値

status		
SS\$_NORMAL		正常終了
0		単語が登録されなかった



---

## JLB\$DEL\_TANGO , JSY\$DEL\_TANGO

### 単語削除

個人辞書から単語を削除します。このルーチンは「かな単語変換ルーチン」と共に使用します。削除しようとする単語に変換した時点で、このルーチン呼び出し、単語を削除します。

---

### フォーマット

status = JLB\$DEL\_TANGO

引数なし

戻り値

status	SS\$_NORMAL	正常終了
--------	-------------	------

---

### フォーマット

status = JSY\$DEL\_TANGO

引数なし

戻り値

status	SS\$_NORMAL	正常終了
	0	ユーザ登録の単語ではない

---

# JLB\$TRA\_ROM\_TANGO

ローマ字・かな単語変換

---

## フォーマット

status = JLB\$TRA\_ROM\_TANGO ( *tango-str*, *yomi-str* [, *tango-len*] )

---

## 引数

tango-str	
JLB usage	変換後文字列
type	文字列データ
access	出力のみ
mechanism	Descriptor 渡し
yomi-str	
JLB usage	変換前文字列
type	文字列データ
access	入力のみ
mechanism	Descriptor 渡し
tango-len	
JLB usage	変換結果のバイト長
type	Word (Unsigned)
access	出力のみ
mechanism	Reference 渡し

---

## 説明

単語の読みから漢字に変換します。変換前文字列に対しローマ字/かな変換を行い、最初の全角ひらがな/カタカナの部分のみを変換対象とし、それ以外の部分は、そのまま、変換した単語の前後に結合します。続けて JLB\$TRA\_TANGO\_NEXT または JLB\$TRA\_TANGO\_PREV を呼び出す場合は、変換前文字列として同じものを渡さなければなりません。

---

戻り値

status

SS\$\_NORMAL

0

正常終了

単語が登録されていない

---

## JLB\$TRA\_KANA\_TANGO , JSY\$TRA\_KANA\_TANGO

### かな単語変換

単語の読みから漢字に変換します。変換前文字列の最初の全角ひらがな/カタカナの部分のみを変換対象とし、それ以外の部分はそのまま、変換した単語の前後に結合します。続けて JLB\$TRA\_TANGO\_NEXT または JLB\$TRA\_TANGO\_PREV を呼び出す場合は、変換前文字列として同じものを渡さなければなりません。

---

### フォーマット

status = JLB\$TRA\_KANA\_TANGO ( *tango-str*, *kana-str* [, *tango-len*] )

#### tango-str

JLB usage	変換後文字列
type	文字列データ
access	出力のみ
mechanism	Descriptor 渡し

#### kana-str

JLB usage	変換前文字列 (読みの部分は全角ひらがな/全角カタカナ)
type	文字列データ
access	入力のみ
mechanism	Descriptor 渡し

#### tango-len

JLB usage	変換結果のバイト長
type	Word (Unsigned)
access	出力のみ
mechanism	Reference 渡し

### 戻り値

status	SS\$_NORMAL	正常終了
	0	単語が登録されていない

---

**フォーマット**

status = JSY\$TRA\_KANA\_TANGO ( *src-str, src-len, dst-str, dst-len, out-len* )

**src-str**

JSY usage 変換対象文字列  
 type 文字列データ  
 access 入力のみ  
 mechanism Reference 渡し

**src-len**

JSY usage 変換対象文字列のバイト長  
 type Longword  
 access 入力のみ  
 mechanism Value 渡し

**dst-str**

JSY usage 変換結果出力領域  
 type 文字列データ  
 access 出力のみ  
 mechanism Reference 渡し

**dst-len**

JSY usage 変換結果出力領域のバイト長  
 type Longword  
 access 入力のみ  
 mechanism Value 渡し

**out-len**

JSY usage 変換結果のバイト長  
 type Longword  
 access 出力のみ  
 mechanism Reference 渡し

**戻り値**

status	1	正常終了
	0	出力結果の切り捨てが行われた または、変換ルーチンの内部エラーが起きた

---

## JLB\$TRA\_TANGO\_NEXT , JSY\$TRA\_TANGO\_NEXT

### 単語次候補

単語の次候補を求めます。あらかじめ、JLB\$TRA\_KANA\_TANGO または JLB\$TRA\_ROM\_TANGO が呼び出されている必要があります、変換前文字列は同じものでなければなりません。すべての候補が一巡した後は、最初の候補に戻ります。

---

### フォーマット

status = JLB\$TRA\_TANGO\_NEXT ( *tango-str*, *yomi-str* [, *tango-len*] )

#### tango-str

JLB usage	変換後文字列
type	文字列データ
access	出力のみ
mechanism	Descriptor 渡し

#### yomi-str

JLB usage	変換前文字列
type	文字列データ
access	入力のみ
mechanism	Descriptor 渡し

#### tango-len

JLB usage	変換結果のバイト長
type	Word (Unsigned)
access	出力のみ
mechanism	Reference 渡し

### 戻り値

status	SS\$_NORMAL	正常終了
	0	変換ルーチンの内部エラーが起きた

---

### フォーマット

status = JSY\$TRA\_TANGO\_NEXT ( *src-str*, *src-len*, *dst-str*, *dst-len*, *out-len* )

#### src-str

	JSY usage		変換対象文字列
	type		文字列データ
	access		入力のみ
	mechanism		Reference 渡し
src-len			
	JSY usage		変換対象文字列のバイト長
	type		Longword
	access		入力のみ
	mechanism		Value 渡し
dst-str			
	JSY usage		変換結果出力領域
	type		文字列データ
	access		出力のみ
	mechanism		Reference 渡し
dst-len			
	JSY usage		変換結果出力領域のバイト長
	type		Longword
	access		入力のみ
	mechanism		Value 渡し
dst-str			
	JSY usage		変換結果出力領域
	type		Longword
	access		出力のみ
	mechanism		Reference 渡し
戻り値			
	status	1	正常終了
		0	出力結果の切り捨てが行われた または、変換ルーチンの内部エラーが起きた

---

## JLB\$TRA\_TANGO\_PREV , JSY\$TRA\_TANGO\_PREV

### 単語前候補

単語の前候補を求めます。あらかじめ、JLB\$TRA\_KANA\_TANGO または JLB\$TRA\_ROM\_TANGO が呼び出されている必要があります、変換前文字列は同じものでなければなりません。すべての候補が一巡した後は、最初の候補に戻ります。

---

### フォーマット

status = JLB\$TRA\_TANGO\_PREV ( tango-str, yomi-str [, tango-len] )

#### tango-str

JLB usage	変換後文字列
type	文字列データ
access	出力のみ
mechanism	Descriptor 渡し

#### yomi-str

JLB usage	変換前文字列
type	文字列データ
access	入力のみ
mechanism	Descriptor 渡し

#### tango-len

JLB usage	変換結果のバイト長
type	Word (Unsigned)
access	出力のみ
mechanism	Reference 渡し

### 戻り値

status	SS\$_NORMAL	正常終了
	0	変換ルーチンの内部エラーが起きた

---

### フォーマット

status = JSY\$TRA\_TANGO\_PREV ( src-str, src-len, dst-str, dst-len, out-len )

#### src-str



	JSY usage		変換対象文字列
	type		文字列データ
	access		入力のみ
	mechanism		Reference 渡し
src-len			
	JSY usage		変換前文字列
	type		Longword
	access		入力のみ
	mechanism		Value 渡し
dst-str			
	JSY usage		変換結果出力領域
	type		文字列データ
	access		出力のみ
	mechanism		Reference 渡し
dst-len			
	JSY usage		変換結果出力領域のバイト長
	type		Longword
	access		入力のみ
	mechanism		Value 渡し
out-len			
	JSY usage		変換結果のバイト長
	type		Longword
	access		出力のみ
	mechanism		Reference 渡し
戻り値			
status	1		正常終了
	0		出力結果の切り捨てが行われた または、変換ルーチンの内部エラーが起きた

---

## JLB\$TRA\_TANGO\_DONE , JSY\$TRA\_TANGO\_DONE

### 単語候補の決定

単語候補の決定を行い、学習機能により次回の変換時に最初に現れるようにします。これ呼び出さずに別の読みによる単語変換を行った場合は、学習機能は働きません。

---

### フォーマット

status = JLB\$TRA\_TANGO\_DONE

引数なし

---

### フォーマット

status = JSY\$TRA\_TANGO\_DONE

引数なし

---

### 戻り値

status

SS\$\_NORMAL

0

正常終了

変換ルーチンの内部エラーが起きた

## 5.4 かな漢字変換辞書

かな漢字変換は、システム辞書、個人辞書と文節学習辞書の3つを用いて行われます。この節では、システム辞書、個人辞書および文節学習辞書の使用方法などについて説明します。

### 5.4.1 システム辞書

システム辞書は通常システムに1つ存在し、すべてのプロセスが共有します。省略時設定のファイル名は、`JSY$DICTIONARY:JSYTANGO.JISHO`です。この辞書には約10万語の単語、地名、氏名などが登録されています。論理名`JSY$TANGO`が定義されている場合は、そのファイルをシステム辞書として使用します。システム辞書は参照のみ可能で、その内容を変更することはできません。

### 5.4.2 個人辞書と文節学習辞書

プロセス単位で使用する個人辞書には2つの種類があります。1つは、ユーザが使用した単語の使用順や単語登録の情報を保存するもの、もう1つは、かな漢字変換における文節の区切りや自立語、付属語の組み合わせなど、文節に関連する情報を保存するものです。

通常、前者を個人辞書、後者を文節学習辞書と呼びます。1つのプロセスは1つの個人辞書と1つの文節学習辞書を使用します。複数のプロセスが、同時に1つの個人辞書と1つの文節学習辞書を共有して使用することが可能です。

#### 1. 個人辞書

個人辞書の省略時設定のファイル名は`SYS$LOGIN:JSYKOJIN.JISHO`です。論理名`JSY$KOJIN`が定義されている場合は、そのファイルを個人辞書として扱います。ただし、ネットワーク上の別ノードにある個人辞書へのアクセスはできません。

個人辞書が存在しない場合は、辞書オープン・ルーチンが自動的に作成します。かな漢字変換ルーチンは個人辞書の単語をシステム辞書の単語に優先して使用します。

この辞書には、ユーザの使用した単語が使用順に登録されます。また、日本語エディタなどを使用中に単語登録を行った場合にも、この辞書に登録されます。個人辞書編集ユーティリティを用いて、個人辞書を参照・追加・変更することもできます。

個人辞書への単語の登録には次の方法があります。

- 学習ルーチンで登録する (`JLB$CNV_LEARN` など)
- 単語登録ルーチンで登録する (`JLB$CNV_REGISTER_WORD` など)
- 辞書編集ユーティリティ (`JDICEDIT`) で登録する

個人辞書から単語を削除するには次の方法があります。

- 単語削除ルーチンで削除する (JLB\$CNV\_DELETE\_WORD など)
- 辞書編集ユーティリティ (JDICEDIT) で削除する

## 2. 文節学習辞書

論理名 JSY\$LEARN が定義されていない場合は、SYS\$LOGIN:JSY\$LEARN.DAT が使用されます。ただし、ネットワーク上の別ノードにある文節学習辞書へのアクセスはできません。文節学習辞書が存在しない場合は、辞書オープン・ルーチンが自動的に作成します。

この辞書には、かな漢字変換における候補選択で最後に選ばれた単語が自立語と付属語とからなるとき、付属語も含む文節の情報を保存し、再使用するために用います。また、同じ読みに対して複数の文節区切りが可能な場合、文節の区切り方の情報もこの辞書に学習します。

例えば、「きしゃのきしゃがきしゃできしゃした」を、次候補・文節移動などを使って正しい文章に変換します。

貴社の/記者が/汽車で/帰社した/

次回からは、上記の文章が1回で正しく変換されます。

文節学習辞書に文節学習データが書き込まれるのは、JLB\$CNV\_CLOSE\_DICTIONARY または JSY\$CNV\_CLOSE\_DICTIONARY ルーチンが呼ばれたときです。JLB\$CNV\_LEARN または JSY\$CNV\_LEARN ルーチンが呼ばれた時点では、メモリ中のデータの更新が行われるだけで文節学習辞書への書き込みは行われません。

### 5.4.3 個人辞書の使用モードの選択

個人辞書の使用モードを、論理名 JSY\$KOJIN\_MODE の値で指定することができます。特に指定しない場合には0(共有モード)になります。

- 個人辞書・共有モード  
(\$ ASSIGN 0 JSY\$KOJIN\_MODE)

個人辞書を参照・更新します。JLB\$CNV\_LEARN または JSY\$CNV\_LEARN ルーチンを呼ぶと学習機能が働きます。論理名 JSY\$KOJIN\_MODE が定義されていないときは、このモードです。この場合、1つの個人辞書を複数のプロセスが共有できます。

- 個人辞書・学習モード  
(\$ ASSIGN 1 JSY\$KOJIN\_MODE)

個人辞書を参照・更新します。JLB\$CNV\_LEARN または JSY\$CNV\_LEARN ルーチンを呼ぶと学習機能が働きます。この場合、個人辞書は1つのプロセスが専有します。

- 個人辞書・参照モード  
(\$ ASSIGN 2 JSY\$KOJIN\_MODE)  
個人辞書を参照して変換を行います。個人辞書への単語登録・学習機能は働きません。それぞれのプロセスがこのモードのときは、1つの個人辞書を複数のプロセスが共有できます。
- 個人辞書・不使用モード  
(\$ ASSIGN 3 JSY\$KOJIN\_MODE)  
個人辞書をまったく使用せず、システム辞書のみで変換が行われます。

#### 5.4.4 個人辞書の学習モードの選択

個人辞書の使用モードが“共有モード”または“学習モード”の場合、JLB\$CNV\_LEARN および JSY\$CNV\_LEARN ルーチンは、漢字変換した単語と1度漢字変換した後に文節ひらがな・カタカナ変換した単語を個人辞書に学習します。また JLB\$CNV\_CLOSE\_DICTIONARY および JSY\$CNV\_CLOSE\_DICTIONARY ルーチンは、文節学習結果を文節学習辞書に保存します。論理名 JSY\$KOJIN\_LEARN は、このうち文節ひらがな・カタカナ変換した単語の学習モードと文節学習のモードを制御します。

以下のいずれのモードの場合も、漢字変換された単語は学習します。

- 漢字・ひらがな・カタカナおよび文節学習モード  
(\$ ASSIGN 0 JSY\$KOJIN\_LEARN)  
漢字変換された単語を学習します。  
文節ひらがな変換された単語を学習します。  
文節カタカナ変換された単語を学習します。  
文節学習結果を文節学習辞書に保存します。  
論理名 JSY\$KOJIN\_LEARN が定義されていない場合は、このモードです。  
文節学習を行わない場合には、JSY\$KOJIN\_LEARN に0以外の値を指定します。
- 漢字・ひらがな・カタカナ学習モード  
(\$ ASSIGN 1 JSY\$KOJIN\_LEARN)  
漢字変換された単語を学習します。  
文節ひらがな変換された単語を学習します。  
文節カタカナ変換された単語を学習します。  
文節学習結果を文節学習辞書に保存しません。
- 漢字・ひらがな学習モード  
(\$ ASSIGN 2 JSY\$KOJIN\_LEARN)  
漢字変換された単語を学習します。  
文節ひらがな変換された単語を学習します。  
文節カタカナ変換された単語は学習しません。

文節学習結果を文節学習辞書に保存しません。

- 漢字・カタカナ学習モード  
(\$ ASSIGN 3 JSY\$KOJIN\_LEARN)

漢字変換された単語を学習します。  
文節ひらがな変換された単語は学習しません。  
文節カタカナ変換された単語を学習します。  
文節学習結果を文節学習辞書に保存しません。

- 漢字学習モード  
(\$ ASSIGN 4 JSY\$KOJIN\_LEARN)

漢字変換された単語を学習します。  
文節ひらがな変換された単語は学習しません。  
文節カタカナ変換された単語は学習しません。  
文節学習結果を文節学習辞書に保存しません。

#### 5.4.5 個人辞書の単語数

個人辞書に登録できる単語数には一部制限があります。

- 個人辞書に登録できる単語数は、最大約 100,000 語です。ただし、登録単語数が多くなると、かな漢字変換実行時の単語の検索に時間がかかるようになります。実用上の限界は、30,000 語程度と考えてください。
- 同じ読みをもつ単語(同音異表記単語)の数には制限があり、この制限を越えて単語を登録しようとした場合は、ステータスが JSY\$\_INDEXFULL のエラーになり、登録できません。

1つの読みに対して複数の表記が個人辞書に存在するとき、この読みに対して割り当てられる個人辞書のデータ・ブロックの最大は4ブロック(2048バイト)です。したがって、同じ読みをもつ単語の登録数には限りがあります。しかし、読みの長さ、各々単語の表記の長さ、文法情報の数と組み合わせにより、登録可能な単語の数は変化します。以下におおよその目安を説明します。

例

読みの長さ :  $x$  文字 (例 : しゃいん 4文字)  
表記の平均の長さ :  $y$  文字 (例 : 佐々木一郎, 田中浩二郎... 平均5文字)  
各々の表記の文法情報の平均数 :  $z$  個 (例 : 人名 1個)

このとき、同一読みの登録可能な単語数の目安は、

$$2048 - x - 1$$

で表わされます。

$$2y + z + 3$$

例の場合は、 $(2048 - 4 - 1) \div (2 \times 5 + 1 + 3) = 145.9\dots$   
となり、約145語登録できるという目安が得られます。

---

注意

---

1つの表記が複数の文法情報を持つとき、その文法情報の組み合わせによっては、文法情報データが縮小されて登録されることもあります。

---

---

## 5.5 注意事項および制限事項

かな漢字変換ルーチンを使用するときの注意事項および制限事項について説明します。

### 5.5.1 変換の開始と終了

かな漢字変換ルーチン群は、同時に2つ以上の読みを対象として変換を進行することはできません。変換開始ルーチンが呼び出された時点から新しい読み文字列による変換を開始します。また、学習機能ルーチンや単語登録/削除ルーチンを呼び出した後、再びかな漢字変換を行う場合は、変換開始ルーチンを呼び出し、読みを指定する必要があります。

### 5.5.2 複文節変換と単語単位変換の混用

1つの実行イメージで複文節変換(JLB\$CNV\_XXXXXX および JSY\$CNV\_XXXXXX)と単語単位変換(JLB\$TRA\_XXXXXX および JSY\$TRA\_XXXXXX)の両方のルーチンを使用する場合は次のような注意が必要です。

ある読みによる、かな漢字変換が進行している間の変換処理(次候補要求など)および学習機能は、変換を開始したルーチンにより、複文節変換あるいは単語単位変換のどちらかのルーチンを統一して使用しなければなりません。

たとえば、複文節変換のJLB\$CNV\_CONVERTルーチンにより変換を開始した場合、次候補要求には必ずJLB\$CNV\_NEXT\_WORDルーチンを呼び出し、また、学習機能を働かせるためにはJLB\$CNV\_LEARNルーチンを呼び出さなければなりません。同様に、単語単位変換のJLB\$TRA\_KANA\_TANGOルーチンにより変換を開始した場合には、次候補要求ルーチンとしてJLB\$TRA\_TANGO\_NEXTルーチンを、学習機能を働かせるにはJLB\$TRA\_TANGO\_DONEルーチンを呼び出さなければなりません。

1つの読みによる変換の進行中に複文節かな漢字変換ルーチン群(JLB\$CNV\_XXXXXX または JSY\$CNV\_XXXXXX)と単語単位かな漢字変換ルーチン群(JLB\$TRA\_XXXXXX または JSY\$TRA\_XXXXXX)を混用した場合の結果については保証されません。

### 5.5.3 個人辞書を共用した場合の変換結果

日本語 OpenVMS Alpha では、複数のプロセスで個人辞書を共有することが可能になっていますが、この場合、学習結果は JSY\$CNV\_LEARN または JLB\$CNV\_LEARN が呼ばれた時点で個人辞書に反映されます。したがって、プロセス A で学習した単語と同じものをプロセス B がその直後に再学習した場合など、プロセス A から見た場合に直前の学習結果が反映されていないように見える場合があります。これは、日本語エディタなどのリカバリ機能を持つユーティリティに対して影響があります。

### 5.5.4 辞書アクセス時のエラー

複数のプロセスから同時に 1 つの個人辞書を共有している場合、個人辞書に対して書き込みが行われている間は他のプロセスからはアクセスできません。したがって、かな漢字変換ライブラリ内部では、辞書が解放されるのを最大 3 秒間だけ待ちます。その間に個人辞書が解放されれば通常どおり変換可能です。しかし時間内に辞書が解放されなかった場合、JSY\$\_RMSERR を返します。この場合、関数 jsy\$cnv\_io\_error または jlb\$cnv\_io\_error を呼び出せば RMS\$\_FLK が得られ、上記の事態が起きたことの確認ができます。

個人辞書がロックされるのは、次の 6 つの関数です。

jsy\$cnv_open_dictionary	jlb\$cnv_open_dictionary	辞書のオープン
jsy\$cnv_convert	jlb\$cnv_convert	かな漢字変換
jsy\$cnv_clause_delete	jlb\$cnv_clause_delete	自立語削除
jsy\$cnv_register_word	jlb\$cnv_register_word	単語登録
jsy\$cnv_delete_word	jlb\$cnv_delete_word	単語削除
jsy\$cnv_learn	jlb\$cnv_learn	変換確定と学習

個人辞書が存在している場合は、jsy\$cnv\_open\_dictionary と jsy\$cnv\_convert、jsy\$cnv\_open\_dictionary および jsy\$cnv\_convert、あるいは jsy\$cnv\_convert どうしは互いに、同時に辞書アクセスができます。



## 漢字コード変換ライブラリ

この章では、漢字コード変換ライブラリについて説明します。

漢字コード変換ライブラリ・ルーチン一覧(第 6.1 節)

漢字コード変換ルーチン(第 6.2 節)

1 バイト・コード変換テーブル(第 6.3 節)

DEC 漢字コード変換ライブラリ・ルーチン(第 6.4 節)

漢字コード変換ライブラリとは、漢字コードの変換を行うライブラリです。DEC 漢字コードを他社の漢字コードに、または他社漢字コードを DEC 漢字コードに変換することができます。

- 現在のバージョンでは以下のコード体系との変換が可能です。

JIS	HITACHI	IBM	JEF (富士通)
NEC	CP/M	MSDOS	

- 入力としての文字列は、漢字コードと制御コードからなることを原則とします。
- ただし、DEC コード 各社コード、NEC DEC、MSDOS DEC、CP/M DEC の場合には入力の文字列に 1 バイト・データ・コードが混在していても変換することができます。入力の文字列と出力の文字列のバイト数は変化しません。
- DEC JIS、DEC IBM、DEC JEF、DEC HITACHI の場合の入力の文字列に 1 バイト・データ・コードが混在していた場合、出力の文字列には 1 バイト・コードと漢字コードを区別するシフト・コードは付加しません。
- 各社の入力の文字列に 1 バイトのカタカナ・コードが混在していた場合、DEC の文字列にはカタカナ・コードと漢字コードを区別するシフト・コードは付加しません。出力された文字列を再び元の文字列に変換しようとした場合、次のような問題があります。
- JIS 第 1 ~ 94 区の漢字コードと制御コードだけからなる文字列ならば、元に戻ります。
- 外字などの漢字コードは' '(DEC: %X'A2A2')に変換しますので元には戻りません。
- DEC IBM、DEC JEF、DEC CP/M では DEC の全角のスペース・コードが半角のスペース・コード(制御コード) 2 バイトに変換しますので元には戻りません。

- DEC JIS, DEC IBM, DEC JEF, DEC HITACHI, DEC NEC( SE )の時に入力の文字列に1バイト・データ・コードが混在していた場合, 出力文字列にはシフト・コードを付加していないため1バイト・コードと漢字コードの区別がつかず, 元には戻りません。各社 DECの場合も入力にカタカナ・コードが混在しているときには, 元には戻りません。
- JLB\$TRA\_KANJI\_XXX\_XXX ルーチンでは, 出力文字列にダイナミック・デスクリプタを指定した場合, 出力文字列は512バイトに制限されます(出力文字列が512バイトを越える場合, 512バイトで切り捨てられ, 戻り値としてLIB\$\_STRTRUが返されます)。512バイト以上の入力文字列を変換する場合には, スタティック・デスクリプタを使用してください。

## 6.1 漢字コード変換ライブラリ・ルーチン一覧

### 漢字コード変換ルーチン

DEC 漢字コードと各社の漢字コードの変換を行うためのルーチン群です。各ルーチンの詳細は第6.2節を参照してください。

JLB\$TRA_KANJI_DEC_JIS( ... )	DEC コード	JIS コード
JLB\$TRA_KANJI_DEC_HITACHI( ... )	DEC コード	HITACHI コード
JLB\$TRA_KANJI_DEC_IBM( ... )	DEC コード	IBM コード
JLB\$TRA_KANJI_DEC_JEF( ... )	DEC コード	JEF コード
JLB\$TRA_KANJI_DEC_NEC( ... )	DEC コード	NEC コード
JLB\$TRA_KANJI_DEC_MSDOS( ... )	DEC コード	MSDOS コード
JLB\$TRA_KANJI_DEC_CPM( ... )	DEC コード	CP/M コード
JLB\$TRA_KANJI_JIS_DEC( ... )	JIS コード	DEC コード
JLB\$TRA_KANJI_HITACHI_DEC( ... )	HITACHI コード	DEC コード
JLB\$TRA_KANJI_IBM_DEC( ... )	IBM コード	DEC コード
JLB\$TRA_KANJI_JEF_DEC( ... )	JEF コード	DEC コード
JLB\$TRA_KANJI_NEC_DEC( ... )	NEC コード	DEC コード
JLB\$TRA_KANJI_MSDOS_DEC( ... )	MSDOS コード	DEC コード
JLB\$TRA_KANJI_CPM_DEC( ... )	CP/M コード	DEC コード

### 1バイト・コード変換テーブル

1バイト・コードを変換する時に使用する256バイトのコード変換テーブルです。各ルーチンの詳細は第6.3節を参照してください。

```
JSY$GTBL_TO_ASCII
JSY$GTBL_TO_ASCII_K
JSY$GTBL_TO_NEC
JSY$GTBL_TO_NEC_K
JSY$GTBL_TO_MSDOS
```

JSY\$GTBL\_TO\_MSDOS\_K  
JSY\$GTBL\_TO\_EBCDIK  
JSY\$GTBL\_EBCDIK\_TO\_ASCII

#### DEC 漢字コード変換ライブラリ・ルーチン一覧

DEC 漢字セット中の漢字コードを、同じDEC 漢字セット中の他の漢字コードに変換するためのルーチン群です。各ルーチンの詳細は第 6.4 節を参照してください。

JLB\$KCV\_BEGIN\_CONV  
JLB\$KCV\_END\_CONV  
JLB\$KCV\_FIND  
JLB\$KCV\_CONVERT

---

## 6.2 漢字コード変換ルーチン

漢字コード変換ルーチンは、DEC 漢字コードと各社の漢字コードの変換を行うためのルーチン群です。

---

## JLB\$TRA\_KANJI\_DEC\_JIS

DEC コード JIS コード

- DEC の漢字文字列を JIS の漢字文字列に変換します。
- DEC の JIS 第 1 ~ 94 区以外の漢字コードは%X'2222' ( ' ' ) に変換します。
- DEC 文字列に 1 バイト・コードが混在しているときには、省略時設定の 1 バイト変換テーブル JSY\$GTBL\_TO\_ASCII によって、変換して出力します。1 バイト変換テーブルが指定されているときには、そのテーブルによって変換して出力します (シフト・コードは付加しません)。

---

### フォーマット

status.wlc.v = JLB\$TRA\_KANJI\_DEC\_JIS ( *dst-str.wt.dx*, *src-str.rt.dx* [[*out-len.www.r*] [[*trans-tbl.rt.r*]] ] )

---

### 引数

dst-str

JLB usage	出力文字列
type	文字列データ
access	出力のみ
mechanism	Descriptor 渡し

src-str

JLB usage	入力文字列
type	文字列データ
access	入力のみ
mechanism	Descriptor 渡し

out-len

JLB usage	出力文字列のバイト長
type	Word(unsigned)
access	出力のみ
mechanism	Reference 渡し

trans-tbl

JLB usage	256 バイトの 1 バイト・コード変換テーブル
type	テーブル
access	入力のみ

mechanism      Reference 渡し

---

戻り値

status

SS\$\_NORMAL

LIB\$\_STRTRU

正常終了

出力結果の切り捨てが行われた

---

## JLB\$TRA\_KANJI\_JIS\_DEC

### JIS コード DEC コード

- JIS の漢字文字列を DEC の漢字文字列に変換します。  
JIS の JIS 第 1 ~ 94 区以外の漢字コードは%X'A2A2' ( ' ) に変換します。
- JIS の%X'00' ~ %X'20', %X'7F' ~ %X'FF' は、制御コードとして省略時設定の 1 バイト変換テーブル JSY\$GTBL\_TO\_ASCII によって変換して出力します。  
1 バイト変換テーブルが指定されているときには、そのテーブルによって変換して出力します。
- JIS 文字列に、制御コード以外の 1 バイト・コードやシフト・コードが混在しているときには、正しい結果は得られません。

---

### フォーマット

```
status.wlc.v = JLB$TRA_KANJI_JIS_DEC ( dst-str.wt.dx, src-str.rt.dx [[out-len.www.r] [[trans-tbl.rt.r]] ] )
```

---

### 引数

#### dst-str

JLB usage	出力文字列
type	文字列データ
access	出力のみ
mechanism	Descriptor 渡し

#### src-str

JLB usage	入力文字列
type	文字列データ
access	入力のみ
mechanism	Descriptor 渡し

#### out-len

JLB usage	出力文字列のバイト長
type	Word(unsigned)
access	出力のみ
mechanism	Reference 渡し

#### trans-tbl

JLB usage	256 バイトの 1 バイト・コード変換テーブル
-----------	--------------------------

type	テーブル
access	入力のみ
mechanism	Reference 渡し

---

## 戻り値

status	
SS\$_NORMAL	正常終了
LIB\$_STRTRU	出力結果の切り捨てが行われた

---

## JLB\$TRA\_KANJI\_DEC\_HITACHI

DEC コード HITACHI コード

- DEC の漢字文字列を HITACHI の漢字文字列に変換します。
- DEC の JIS 第 1 ~ 94 区以外の漢字コードは%X'A2A2' ( ' ' )に変換します。
- DEC 文字列に 1 バイト・コードが混在しているときには、省略時設定の 1 バイト変換テーブル JSY\$GTBL\_TO\_EBCDIK によって、変換して出力します。1 バイト変換テーブルが指定されているときには、そのテーブルによって変換して出力します (シフト・コードは付加しません)。

---

### フォーマット

```
status.wlc.v = JLB$TRA_KANJI_DEC_HITACHI ( dst-str.wt.dx, src-str.rt.dx [,out-len.www.r]
                                          [,trans-tbl.rt.r]] )
```

---

### 引数

dst-str		
JLB usage		出力文字列
type		文字列データ
access		出力のみ
mechanism		Descriptor 渡し
src-str		
JLB usage		入力文字列
type		文字列データ
access		入力のみ
mechanism		Descriptor 渡し
out-len		
JLB usage		出力文字列のバイト長
type		Word(unsigned)
access		出力のみ
mechanism		Reference 渡し
trans-tbl		
JLB usage		256 バイトの 1 バイト・コード変換テーブル
type		テーブル



access	入力のみ
mechanism	Reference 渡し

---

## 戻り値

status	
SS\$_NORMAL	正常終了
LIB\$_STRTRU	出力結果の切り捨てが行われた

---

## JLB\$TRA\_KANJI\_HITACHI\_DEC

### HITACHI コード DEC コード

- HITACHI の漢字文字列を DEC の漢字文字列に変換します。
- HITACHI の JIS 第 1 ~ 94 区以外の漢字コードは%X'A2A2' ( ' ' ) に変換します。
- HITACHI の%X'00' ~ %X'40', %X'FF' は、制御コードとして省略時設定の 1 バイト変換テーブル JSY\$GTBL\_EBCDIK\_TO\_ASCII によって変換して出力します。1 バイト変換テーブルが指定されているときには、そのテーブルによって変換して出力します。
- HITACHI 文字列に、制御コード以外の 1 バイト・コードやシフト・コードが混在しているときには、正しい結果は得られません。

---

### フォーマット

```
status.wlc.v = JLB$TRA_KANJI_HITACHI_DEC ( dst-str.wt.dx, src-str.rt.dx [, [out-len.www.r]
                                          [, [trans-tbl.rt.r]] ] )
```

---

### 引数

dst-str	JLB usage	出力文字列
	type	文字列データ
	access	出力のみ
	mechanism	Descriptor 渡し
src-str	JLB usage	入力文字列
	type	文字列データ
	access	入力のみ
	mechanism	Descriptor 渡し
out-len	JLB usage	出力文字列のバイト長
	type	Word(unsigned)
	access	出力のみ
	mechanism	Reference 渡し

## trans-tbl

JLB usage	256 バイトの 1 バイト・コード変換テーブル
type	テーブル
access	入力のみ
mechanism	Reference 渡し

---

戻り値

## status

SS\$\_NORMAL

正常終了

LIB\$\_STRTRU

出力結果の切り捨てが行われた

---

## JLB\$TRA\_KANJI\_DEC\_IBM

DEC コード IBM コード

- DEC の漢字文字列を IBM の漢字文字列に変換します。
- DEC の JIS 第 1 ~ 94 区以外の漢字コードは%X'44E9' ( ' ) に変換します。
- DEC の%X'A1A1'(space) は%X'4040' に変換します。
- DEC 文字列に 1 バイト・コードが混在しているときには、省略時設定の 1 バイト変換テーブル JSY\$GTBL\_TO\_EBCDIK によって、変換して出力します。1 バイト変換テーブルが指定されているときには、そのテーブルによって変換して出力します(シフト・コードは付加しません)。

---

### フォーマット

status.wlc.v = JLB\$TRA\_KANJI\_DEC\_IBM ( *dst-str.wt.dx*, *src-str.rt.dx* [,*[out-len.www.r]*] [,*[trans-tbl.rt.r]]*] )

---

### 引数

dst-str	JLB usage	出力文字列
	type	文字列データ
	access	出力のみ
	mechanism	Descriptor 渡し
src-str	JLB usage	入力文字列
	type	文字列データ
	access	入力のみ
	mechanism	Descriptor 渡し
out-len	JLB usage	出力文字列のバイト長
	type	Word(unsigned)
	access	出力のみ
	mechanism	Reference 渡し
trans-tbl	JLB usage	256 バイトの 1 バイト・コード変換テーブル
	type	テーブル

access	入力のみ
mechanism	Reference 渡し

---

## 戻り値

status	
SS\$_NORMAL	正常終了
LIB\$_STRTRU	出力結果の切り捨てが行われた

---

## JLB\$TRA\_KANJI\_IBM\_DEC

### IBM コード DEC コード

- IBM の漢字文字列を DEC の漢字文字列に変換します。
- IBM の JIS C6220 で定義されていない漢字コードは%X'A2A2' ( ' ') に変換します。
- IBM の%X'00' ~ %X'40', %X'FF' は、制御コードとして省略時設定の 1 バイト変換テーブル JSY\$GTBL\_EBCDIK\_TO\_ASCII によって変換して出力します。1 バイト変換テーブルが指定されているときには、そのテーブルによって変換して出力します。
- IBM 文字列に、制御コード以外の 1 バイト・コードやシフト・コードが混在しているときには、正しい結果は得られません。

---

### フォーマット

```
status.wlc.v = JLB$TRA_KANJI_IBM_DEC ( dst-str.wt.dx, src-str.rt.dx [, [out-len.www.r] [, [trans-tbl.rt.r]] ] )
```

---

### 引数

dst-str		
	JLB usage	出力文字列
	type	文字列データ
	access	出力のみ
	mechanism	Descriptor 渡し
src-str		
	JLB usage	入力文字列
	type	文字列データ
	access	入力のみ
	mechanism	Descriptor 渡し
out-len		
	JLB usage	出力文字列のバイト長
	type	Word(unsigned)
	access	出力のみ
	mechanism	Reference 渡し
trans-tbl		

JLB usage	256 バイトの 1 バイト・コード変換テーブル
type	テーブル
access	入力のみ
mechanism	Reference 渡し

---

## 戻り値

status	
SS\$_NORMAL	正常終了
LIB\$_STRTRU	出力結果の切り捨てが行われた

---

## JLB\$TRA\_KANJI\_DEC\_JEF

### DEC コード JEF コード

- DEC の漢字文字列を JEF (富士通) の漢字文字列に変換します。
- DEC の JIS 第 1 ~ 94 区以外の漢字コードは%X'A2A2' ( ' ') に変換します。
- DEC の%X'A1A1' (space) は%X'4040' に変換します。
- DEC 文字列に 1 バイト・コードが混在しているときには、省略時設定の 1 バイト変換テーブル JSY\$GTBL\_TO\_EBCDIK によって、変換して出力します。1 バイト変換テーブルが指定されているときには、そのテーブルによって変換して出力します (シフト・コードは付加しません)。

---

### フォーマット

```
status.wlc.v = JLB$TRA_KANJI_DEC_JEF ( dst-str.wt.dx, src-str.rt.dx [, [out-len.www.r] [, [trans-tbl.rt.r]] ] )
```

---

### 引数

dst-str		
JLB usage		出力文字列
type		文字列データ
access		出力のみ
mechanism		Descriptor 渡し
src-str		
JLB usage		入力文字列
type		文字列データ
access		入力のみ
mechanism		Descriptor 渡し
out-len		
JLB usage		出力文字列のバイト長
type		Word(unsigned)
access		入力のみ
mechanism		Reference 渡し
trans-tbl		
JLB usage		256 バイトの 1 バイト・コード変換テーブル
type		テーブル



access	入力のみ
mechanism	Reference 渡し

---

## 戻り値

status	
SS\$ _NORMAL	正常終了
LIB\$ _STRTRU	出力結果の切り捨てが行われた

---

## JLB\$TRA\_KANJI\_JEF\_DEC

JEF コード DEC コード

- JEF の漢字文字列を DEC の漢字文字列に変換します。
- JEF の JIS 第 1 ~ 94 区以外の漢字コードは%X'A2A2' ( ' ' )に変換します。
- JEF の%X'00' ~ %X'40', %X'FF' は、制御コードとして省略時設定の 1 バイト変換テーブル JSY\$GTBL\_EBCDIK\_TO\_ASCII によって変換して出力します。1 バイト変換テーブルが指定されているときには、そのテーブルによって変換して出力します。
- JEF 文字列に、制御コード以外の 1 バイト・コードやシフト・コードが混在しているときには、正しい結果は得られません。

---

### フォーマット

status.wc.v = JLB\$TRA\_KANJI\_JEF\_DEC ( *dst-str.wt.dx*, *src-str.rt.dx* [, *[out-len.www.r]* [, *[trans-tbl.rt.r]*] ] )

---

### 引数

dst-str	
JLB usage	出力文字列
type	文字列データ
access	出力のみ
mechanism	Descriptor 渡し
src-str	
JLB usage	入力文字列
type	文字列データ
access	入力のみ
mechanism	Descriptor 渡し
out-len	
JLB usage	出力文字列のバイト長
type	Word(unsigned)
access	入力のみ
mechanism	Reference 渡し
trans-tbl	
JLB usage	256 バイトの 1 バイト・コード変換テーブル

type	テーブル
access	入力のみ
mechanism	Reference 渡し

---

## 戻り値

status	
SS\$_NORMAL	正常終了
LIB\$_STRTRU	出力結果の切り捨てが行われた

---

## JLB\$TRA\_KANJI\_DEC\_NEC

DEC コード NEC コード

- DEC の漢字文字列を NEC (内部コード) の漢字文字列に変換します。
- DEC の JIS 第 1 ~ 94 区以外の漢字コードは%X'6122' ( ' ) に変換します。
- DEC 文字列に 1 バイト・コードが混在している場合には、省略時設定の 1 バイト変換テーブル JSY\$GTBL\_TO\_NEC によって、変換して出力します。1 バイト変換テーブルが指定されているときには、そのテーブルによって変換して出力します (シフト・コードは付加しません)。

---

### フォーマット

status.wlc.v = JLB\$TRA\_KANJI\_DEC\_NEC ( *dst-str.wt.dx*, *src-str.rt.dx* [, [*out-len.www.r*] [, [*trans-tbl.rt.r*]] )

---

### 引数

dst-str

JLB usage	出力文字列
type	文字列データ
access	出力のみ
mechanism	Descriptor 渡し

src-str

JLB usage	入力文字列
type	文字列データ
access	入力のみ
mechanism	Descriptor 渡し

out-len

JLB usage	出力文字列のバイト長
type	Word(unsigned)
access	出力のみ
mechanism	Reference 渡し

trans-tbl

JLB usage	256 バイトの 1 バイト・コード変換テーブル
type	テーブル
access	入力のみ

mechanism      Reference 渡し

---

戻り値

status

SS\$\_NORMAL

LIB\$\_STRTRU

正常終了

出力結果の切り捨てが行われた

---

## JLB\$TRA\_KANJI\_NEC\_DEC

NEC コード DEC コード

- NEC (内部コード) の漢字文字列を DEC の漢字文字列に変換します。
- NEC の JIS 第 1 ~ 94 区以外の漢字コードは%X'A2A2' ( ' ' ) に変換します。
- NEC の文字列に 1 バイト・コードが混在している場合には、省略時設定の 1 バイト変換テーブル JSY\$GTBL\_TO\_ASCII によって、変換して出力します。1 バイト変換テーブルが指定されているときには、そのテーブルによって変換して出力します。

---

### フォーマット

status.wlc.v = JLB\$TRA\_KANJI\_NEC\_DEC ( *dst-str.wt.dx*, *src-str.rt.dx* [,*out-len.www.r*] [,*trans-tbl.rt.r*] ] )

---

### 引数

dst-str

JLB usage	出力文字列
type	文字列データ
access	出力のみ
mechanism	Descriptor 渡し

src-str

JLB usage	入力文字列
type	文字列データ
access	入力のみ
mechanism	Descriptor 渡し

out-len

JLB usage	出力文字列のバイト長
type	Word(unsigned)
access	出力のみ
mechanism	Reference 渡し

trans-tbl

JLB usage	256 バイトの 1 バイト・コード変換テーブル
type	テーブル
access	入力のみ

mechanism      Reference 渡し

---

戻り値

status

SS\$\_NORMAL

LIB\$\_STRTRU

正常終了

出力結果の切り捨てが行われた

---

## JLB\$TRA\_KANJI\_DEC\_MSDOS

DEC コード MSDOS コード

- DEC の漢字文字列を MSDOS の漢字文字列に変換します。
- DEC の JIS 第 1 ~ 94 区以外の漢字コードは%X'81A0' ( ' ) に変換します。
- DEC 文字列に 1 バイト・コードが混在している場合には、省略時設定の 1 バイト変換テーブル JSY\$GTBL\_TO\_MSDOS によって、変換して出力します。1 バイト変換テーブルが指定されているときには、そのテーブルによって変換して出力します (シフト・コードは付加しません)。

---

### フォーマット

```
status.wlc.v = JLB$TRA_KANJI_DEC_MSDOS ( dst-str.wt.dx, src-str.rt.dx [, [out-len.www.r]
                                         [, [trans-tbl.rt.r]] ] )
```

---

### 引数

dst-str		
JLB usage		出力文字列
type		文字列データ
access		出力のみ
mechanism		Descriptor 渡し
src-str		
JLB usage		入力文字列
type		文字列データ
access		入力のみ
mechanism		Descriptor 渡し
out-len		
JLB usage		出力文字列のバイト長
type		Word(unsigned)
access		出力のみ
mechanism		Reference 渡し
trans-tbl		
JLB usage		256 バイトの 1 バイト・コード変換テーブル
type		テーブル



access	入力のみ
mechanism	Reference 渡し

---

## 戻り値

status	
SS\$_NORMAL	正常終了
LIB\$_STRTRU	出力結果の切り捨てが行われた

---

## JLB\$TRA\_KANJI\_MSDOS\_DEC

MSDOS コード DEC コード

- MSDOS の漢字文字列を DEC の漢字文字列に変換します。
- MSDOS の JIS 第 1 ~ 94 区以外の漢字コードは%X'A2A2' ( ' ) に変換します。
- MSDOS の文字列に 1 バイト・コードが混在している場合には、省略時設定の 1 バイト変換テーブル JSY\$GTBL\_TO\_MSDOS によって、変換して出力します。1 バイト変換テーブルが指定されているときには、そのテーブルによって変換して出力します。

---

### フォーマット

```
status.wlc.v = JLB$TRA_KANJI_MSDOS_DEC ( dst-str.wt.dx, src-str.rt.dx [, [out-len.www.r]
                                     [, [trans-tbl.rt.r]] ] )
```

---

### 引数

dst-str		
	JLB usage	出力文字
	type	文字列データ
	access	出力のみ
	mechanism	Descriptor 渡し
src-str		
	JLB usage	入力文字列
	type	文字列データ
	access	入力のみ
	mechanism	Descriptor 渡し
out-len		
	JLB usage	出力文字列のバイト長
	type	Word(unsigned)
	access	出力のみ
	mechanism	Reference 渡し
trans-tbl		
	JLB usage	256 バイトの 1 バイト・コード変換テーブル

type	テーブル
access	入力のみ
mechanism	Reference 渡し

---

## 戻り値

status	
SS\$_NORMAL	正常終了
LIB\$_STRTRU	出力結果の切り捨てが行われた

---

## JLB\$TRA\_KANJI\_DEC\_CPM

### DEC コード CP/M コード

- DEC の漢字文字列を CP/M の漢字文字列に変換します。
- DEC の JIS 第 1 ~ 94 区以外の漢字コードは%X'81A0' ( ' ) に変換します。
- DEC の%X'A1A1' (space) は%X'2020' に変換します。
- DEC 文字列に 1 バイト・コードが混在している場合には、省略時設定の 1 バイト変換テーブル JSY\$GTBL\_TO\_MSDOS によって、変換して出力します。1 バイト変換テーブルが指定されているときには、そのテーブルによって変換して出力します(シフト・コードは付加しません)。

---

### フォーマット

```
status.wlc.v = JLB$TRA_KANJI_DEC_CPM ( dst-str.wt.dx, src-str.rt.dx [,out-len.www.r] [,trans-tbl.rt.r]] )
```

---

### 引数

#### dst-str

JLB usage	出力文字列
type	文字列データ
access	出力のみ
mechanism	Descriptor 渡し

#### src-str

JLB usage	入力文字列
type	文字列データ
access	入力のみ
mechanism	Descriptor 渡し

#### out-len

JLB usage	出力文字列のバイト長
type	Word(unsigned)
access	出力のみ
mechanism	Reference 渡し

#### trans-tbl

JLB usage	256 バイトの 1 バイト・コード変換テーブル
type	テーブル

access	入力のみ
mechanism	Reference 渡し

---

## 戻り値

status	
SS\$_NORMAL	正常終了
LIB\$_STRTRU	出力結果の切り捨てが行われた

---

## JLB\$TRA\_KANJI\_CPM\_DEC

CP/M コード DEC コード

- CP/M の漢字文字列を DEC の漢字文字列に変換します。
- CP/M の JIS 第 1 ~ 94 区以外の漢字コードは%X'A2A2' ( ' ) に変換します。
- CP/M の文字列に 1 バイト・コードが混在している場合には、省略時設定の 1 バイト変換テーブル JSY\$GTBL\_TO\_MSDOS によって、変換して出力します。1 バイト変換テーブルが指定されているときには、そのテーブルによって変換して出力します。

---

### フォーマット

status.wlc.v = JLB\$TRA\_KANJI\_CPM\_DEC ( *dst-str.wt.dx*, *src-str.rt.dx* [[*out-len.www.r*] [[*trans-tbl.rt.r*]] )

---

### 引数

dst-str

JLB usage	出力文字列
type	文字列データ
access	出力のみ
mechanism	Descriptor 渡し

src-str

JLB usage	入力文字列
type	文字列データ
access	入力のみ
mechptor	Descriptor 渡し

out-len

JLB usage	出力文字列のバイト長
type	Word(unsigned)
access	出力のみ
mechanism	Reference 渡し

trans-tbl

JLB usage	256 バイトの 1 バイト・コード変換テーブル
type	テーブル
access	入力のみ

mechanism      Reference 渡し

---

戻り値

status	
SS\$_NORMAL	正常終了
LIB\$_STRTRU	出力結果の切り捨てが行われた

## 6.3 1バイト・コード変換テーブル

1バイト・コード変換テーブルは、1バイト・コードを変換する際に使用する256バイトのコード変換テーブルです。



---

## JSY\$GTBL\_TO\_ASCII

- ASCII コードに変換します。
- ただし、カタカナ・コードは'¥'(%X'5C')に置換します。

図 6-1 JSY\$GTBL\_TO\_ASCII テーブル

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	10	20	30	40	50	60	70	80	90	A0	*1	*1	*1	*1	*1
1	01	11	21	31	41	51	61	71	81	91	*1	*1	*1	*1	*1	*1
2	02	12	22	32	42	52	62	72	82	92	*1	*1	*1	*1	*1	*1
3	03	13	23	33	43	53	63	73	83	93	*1	*1	*1	*1	*1	*1
4	04	14	24	34	44	54	64	74	84	94	*1	*1	*1	*1	*1	*1
5	05	15	25	35	45	55	65	75	85	95	*1	*1	*1	*1	*1	*1
6	06	16	26	36	46	56	66	76	86	96	*1	*1	*1	*1	*1	*1
7	07	17	27	37	47	57	67	77	87	97	*1	*1	*1	*1	*1	*1
8	08	18	28	38	48	58	68	78	88	98	*1	*1	*1	*1	*1	*1
9	09	19	29	39	49	59	69	79	89	99	*1	*1	*1	*1	*1	*1
A	0A	1A	2A	3A	4A	5A	6A	7A	8S	9S	*1	*1	*1	*1	*1	*1
B	0B	1B	2B	3B	4B	5B	6B	7B	8B	9B	*1	*1	*1	*1	*1	*1
C	0C	1C	2C	3C	4C	5C	6C	7C	8C	9C	*1	*1	*1	*1	*1	*1
D	0D	1D	2D	3D	4D	5D	6D	7D	8D	9D	*1	*1	*1	*1	*1	*1
E	0E	1E	2E	3E	4E	5E	6E	7E	8E	9E	*1	*1	*1	*1	*1	*1
F	0F	1F	2F	3F	4F	5F	6F	7F	8F	9F	*1	*1	*1	*1	*1	FF

※\*1 は%X'5C'

---

## JSY\$GTBL\_TO\_ASCII\_K

- ASCII コードに変換します。

図 6-2 JSY\$GTBL\_TO\_ASCII\_K テーブル

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
1	01	11	21	31	41	51	61	71	81	91	A1	B1	C1	D1	E1	F1
2	02	12	22	32	42	52	62	72	82	92	A2	B2	C2	D2	E2	F2
3	03	13	23	33	43	53	63	73	83	93	A3	B3	C3	D3	E3	F3
4	04	14	24	34	44	54	64	74	84	94	A4	B4	C4	D4	E4	F4
5	05	15	25	35	45	55	65	75	85	95	A5	B5	C5	D5	E5	F5
6	06	16	26	36	46	56	66	76	86	96	A6	B6	C6	D6	E6	F6
7	07	17	27	37	47	57	67	77	87	97	A7	B7	C7	D7	E7	F7
8	08	18	28	38	48	58	68	78	88	98	A8	B8	C8	D8	E8	F8
9	09	19	29	39	49	59	69	79	89	99	A9	B9	C9	D9	E9	F9
A	0A	1A	2A	3A	4A	5A	6A	7A	8A	9A	AA	BA	CA	DA	EA	FA
B	0B	1B	2B	3B	4B	5B	6B	7B	8B	9B	AB	BB	CB	DB	EB	FB
C	0C	1C	2C	3C	4C	5C	6C	7C	8C	9C	AC	BC	CC	DC	EC	FC
D	0D	1D	2D	3D	4D	5D	6D	7D	8D	9D	AD	BD	CD	DD	ED	FD
E	0E	1E	2E	3E	4E	5E	6E	7E	8E	9E	AE	BE	CE	DE	EE	FE
F	0F	1F	2F	3F	4F	5F	6F	7F	8F	9F	AF	BF	CF	DF	EF	FF

※\*1 は%X'5C'

---

## JSY\$GTBL\_TO\_NEC

- NEC コードに変換します。
- ただし、カタカナ・コードは' ¥' (%X'5C') に置換します。
- %X'60', %X'7B' ~ %X'7E' ( '{ | } ~ ) は' ¥' (%X'5C') に置換します。
- 英小文字は大文字に変換します。
- %X'80' ~ %X'A0' の制御コードは' ¥' (%X'5C') に置換します。

図 6-3 JSY\$GTBL\_TO\_NEC テーブル

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	10	20	30	40	50	*1	50	*1	*1	*1	*1	*1	*1	*1	*1
1	01	11	21	31	41	51	41	51	*1	*1	*1	*1	*1	*1	*1	*1
2	02	12	22	32	42	52	42	52	*1	*1	*1	*1	*1	*1	*1	*1
3	03	13	23	33	43	53	43	53	*1	*1	*1	*1	*1	*1	*1	*1
4	04	14	24	34	44	54	44	54	*1	*1	*1	*1	*1	*1	*1	*1
5	05	15	25	35	45	55	45	55	*1	*1	*1	*1	*1	*1	*1	*1
6	06	16	26	36	46	56	46	56	*1	*1	*1	*1	*1	*1	*1	*1
7	07	17	27	37	47	57	47	57	*1	*1	*1	*1	*1	*1	*1	*1
8	08	18	28	38	48	58	48	58	*1	*1	*1	*1	*1	*1	*1	*1
9	09	19	29	39	49	59	49	59	*1	*1	*1	*1	*1	*1	*1	*1
A	0A	1A	2A	3A	4A	5A	4A	5A	*1	*1	*1	*1	*1	*1	*1	*1
B	0B	1B	2B	3B	4B	5B	4B	*1	*1	*1	*1	*1	*1	*1	*1	*1
C	0C	1C	2C	3C	4C	5C	4C	*1	*1	*1	*1	*1	*1	*1	*1	*1
D	0D	1D	2D	3D	4D	5D	4D	*1	*1	*1	*1	*1	*1	*1	*1	*1
E	0E	1E	2E	3E	4E	5E	4E	*1	*1	*1	*1	*1	*1	*1	*1	*1
F	0F	1F	2F	3F	4F	5F	4F	7F	*1	*1	*1	*1	*1	*1	*1	FF

※\*1 は%X'5C'

---

## JSY\$GTBL\_TO\_NEC\_K

- NEC コードに変換します。
- %X'60' , %X'7B' ~ %X'7E' ( '{ | } ~ ) は ' ¥ ' (%X'5C') に置換します。
- 英小文字は大文字に変換します。
- %X'80' ~ %X'A0' の制御コードは ' ¥ ' (%X'5C') に置換します。
- 未定義カタカナ・コードは ' ¥ ' (%X'5C') に置換します。

図 6-4 JSY\$GTBL\_TO\_NEC\_K テーブル

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	10	20	30	40	50	*1	50	*1	*1	*1	B0	C0	D0	*1	*1
1	01	11	21	31	41	51	41	51	*1	*1	A1	B1	C1	D1	*1	*1
2	02	12	22	32	42	52	42	52	*1	*1	A2	B2	C2	D2	*1	*1
3	03	13	23	33	43	53	43	53	*1	*1	A3	B3	C3	D3	*1	*1
4	04	14	24	34	44	54	44	54	*1	*1	A4	B4	C4	D4	*1	*1
5	05	15	25	35	45	55	45	55	*1	*1	A5	B5	C5	D5	*1	*1
6	06	16	26	36	46	56	46	56	*1	*1	A6	B6	C6	D6	*1	*1
7	07	17	27	37	47	57	47	57	*1	*1	A7	B7	C7	D7	*1	*1
8	08	18	28	38	48	58	48	58	*1	*1	A8	B8	C8	D8	*1	*1
9	09	19	29	39	49	59	49	59	*1	*1	A9	B9	C9	D9	*1	*1
A	0A	1A	2A	3A	4A	5A	4A	5A	*1	*1	AA	BA	CA	DA	*1	*1
B	0B	1B	2B	3B	4B	5B	4B	*1	*1	*1	AB	BB	CB	DB	*1	*1
C	0C	1C	2C	3C	4C	5C	4C	*1	*1	*1	AC	BC	CC	DC	*1	*1
D	0D	1D	2D	3D	4D	5D	4D	*1	*1	*1	AD	BD	CD	DD	*1	*1
E	0E	1E	2E	3E	4E	5E	4E	*1	*1	*1	AE	BE	CE	DE	*1	*1
F	0F	1F	2F	3F	4F	5F	4F	7F	*1	*1	AF	BF	CF	DF	*1	FF

※\*1 は%X'5C'

---

## JSY\$GTBL\_TO\_MSDOS

- MSDOS コードに変換します。
- ただし、カタカナ・コードは' ¥' (%X'5C') に置換します。
- %X'80' ~ %X'A0' の制御コードは' ¥' (%X'5C') に置換します。

図 6-5 JSY\$GTBL\_TO\_MSDOS テーブル

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	10	20	30	40	50	60	70	*1	*1	*1	*1	*1	*1	*1	*1
1	01	11	21	31	41	51	61	71	*1	*1	*1	*1	*1	*1	*1	*1
2	02	12	22	32	42	52	62	72	*1	*1	*1	*1	*1	*1	*1	*1
3	03	13	23	33	43	53	63	73	*1	*1	*1	*1	*1	*1	*1	*1
4	04	14	24	34	44	54	64	74	*1	*1	*1	*1	*1	*1	*1	*1
5	05	15	25	35	45	55	65	75	*1	*1	*1	*1	*1	*1	*1	*1
6	06	16	26	36	46	56	66	76	*1	*1	*1	*1	*1	*1	*1	*1
7	07	17	27	37	47	57	67	77	*1	*1	*1	*1	*1	*1	*1	*1
8	08	18	28	38	48	58	68	78	*1	*1	*1	*1	*1	*1	*1	*1
9	09	19	29	39	49	59	69	79	*1	*1	*1	*1	*1	*1	*1	*1
A	0A	1A	2A	3A	4A	5A	6A	7A	*1	*1	*1	*1	*1	*1	*1	*1
B	0B	1B	2B	3B	4B	5B	6B	7B	*1	*1	*1	*1	*1	*1	*1	*1
C	0C	1C	2C	3C	4C	5C	6C	7C	*1	*1	*1	*1	*1	*1	*1	*1
D	0D	1D	2D	3D	4D	5D	6D	7D	*1	*1	*1	*1	*1	*1	*1	*1
E	0E	1E	2E	3E	4E	5E	6E	7E	*1	*1	*1	*1	*1	*1	*1	*1
F	0F	1F	2F	3F	4F	5F	6F	7F	*1	*1	*1	*1	*1	*1	*1	FF

※\*1 は%X'5C'



---

## JSY\$GTBL\_TO\_MSDOS\_K

- MSDOS コードに変換します。
- %X'80' ~ %X'A0' の制御コードは ' ¥ ' (%X'5C') に置換します。
- 未定義カタカナ・コードは ' ¥ ' (%X'5C') に置換します。

図 6-6 JSY\$GTBL\_TO\_MSDOS\_K テーブル

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	10	20	30	40	50	60	70	*1	*1	A0	B0	C0	D0	*1	*1
1	01	11	21	31	41	51	61	71	*1	*1	A1	B1	C1	D1	*1	*1
2	02	12	22	32	42	52	62	72	*1	*1	A2	B2	C2	D2	*1	*1
3	03	13	23	33	43	53	63	73	*1	*1	A3	B3	C3	D3	*1	*1
4	04	14	24	34	44	54	64	74	*1	*1	A4	B4	C4	D4	*1	*1
5	05	15	25	35	45	55	65	75	*1	*1	A5	B5	C5	D5	*1	*1
6	06	16	26	36	46	56	66	76	*1	*1	A6	B6	C6	D6	*1	*1
7	07	17	27	37	47	57	67	77	*1	*1	A7	B7	C7	D7	*1	*1
8	08	18	28	38	48	58	68	78	*1	*1	A8	B8	C8	D8	*1	*1
9	09	19	29	39	49	59	69	79	*1	*1	A9	B9	C9	D9	*1	*1
A	0A	1A	2A	3A	4A	5A	6A	7A	*1	*1	AA	BA	CA	DA	*1	*1
B	0B	1B	2B	3B	4B	5B	6B	7B	*1	*1	AB	BB	CB	DB	*1	*1
C	0C	1C	2C	3C	4C	5C	6C	7C	*1	*1	AC	BC	CC	DC	*1	*1
D	0D	1D	2D	3D	4D	5D	6D	7D	*1	*1	AD	BD	CD	DD	*1	*1
E	0E	1E	2E	3E	4E	5E	6E	7E	*1	*1	AE	BE	CE	DE	*1	*1
F	0F	1F	2F	3F	4F	5F	6F	7F	*1	*1	AF	BF	CF	DF	*1	FF

※\*1 は%X'5C'

---

## JSY\$GTBL\_TO\_EBCDIK

- EBCDIK コードに変換します。
- ただし、カタカナ・コードは '<SUB>' (EBCDIC: %X'3F') に置換します。
- 英小文字は大文字に変換します。
- %X'80' ~ %X'A0' の制御コードは '<SUB>' (EBCDIC: %X'3F') に置換します。

図 6-7 JSY\$GTBL\_TO\_EBCDIK テーブル

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	10	40	F0	7C	D7	79	D7	*2	*2	*2	*2	*2	*2	*2	*2
1	01	11	4F	F1	C1	D8	C1	D8	*2	*2	*2	*2	*2	*2	*2	*2
2	02	12	7F	F2	C2	D9	C2	D9	*2	*2	*2	*2	*2	*2	*2	*2
3	03	13	7B	F3	C3	E2	C3	E2	*2	*2	*2	*2	*2	*2	*2	*2
4	37	3C	5B	F4	C4	E3	C4	E3	*2	*2	*2	*2	*2	*2	*2	*2
5	2D	3D	6C	F5	C5	E4	C5	E4	*2	*2	*2	*2	*2	*2	*2	*2
6	2E	32	50	F6	C6	E5	C6	E5	*2	*2	*2	*2	*2	*2	*2	*2
7	2F	26	7D	F7	C7	E6	C7	E6	*2	*2	*2	*2	*2	*2	*2	*2
8	16	18	4D	F8	C8	E7	C8	E7	*2	*2	*2	*2	*2	*2	*2	*2
9	05	19	5D	F9	C9	E8	C9	E8	*2	*2	*2	*2	*2	*2	*2	*2
A	25	3F	5C	7A	D1	E9	D1	E9	*2	*2	*2	*2	*2	*2	*2	*2
B	0B	27	4E	5E	D2	4A	D2	C0	*2	*2	*2	*2	*2	*2	*2	*2
C	0C	1C	6B	4C	D3	E0	D3	6A	*2	*2	*2	*2	*2	*2	*2	*2
D	0D	1D	60	7E	D4	5A	D4	D0	*2	*2	*2	*2	*2	*2	*2	*2
E	0E	1E	4B	6E	D5	5F	D5	A1	*2	*2	*2	*2	*2	*2	*2	*2
F	0F	1F	61	6F	D6	6D	D6	07	*2	*2	*2	*2	*2	*2	*2	FF

※\*2 は%X'3F'

---

## JSY\$GTBL\_EBCDIK\_TO\_ASCII

- EBCDIK コードを ASCII コードに変換します。
- ただし、カタカナ・コードは '<SUB>' (EBCDIC: %X'3F') に置換します。

図 6-8 JSY\$GTBL\_EBCDIK\_TO\_ASCII テーブル

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	10	*1	*1	20	26	2D	*1	*1	*1	*1	*1	7B	7D	*1	30
1	01	11	*1	*1	*1	*1	2F	*1	*1	*1	*1	*1	7B	4A	*1	31
2	02	12	*1	16	*1	*1	*1	*1	*1	*1	*1	*1	41	4B	53	32
3	03	13	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1	42	4C	54	33
4	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1	43	4D	55	34
5	09	*1	0A	*1	*1	*1	*1	*1	*1	*1	*1	*1	44	4E	56	35
6	*1	08	17	*1	*1	*1	*1	*1	*1	*1	*1	*1	45	4F	57	36
7	7F	*1	1B	04	*1	*1	5C	5C	5C	5C	5C	5C	46	50	58	37
8	5C	18	5C	5C	5C	5C	5C	5C	5C	5C	5C	5C	47	51	59	38
9	5C	19	5C	5C	5C	5C	5C	60	5C	5C	5C	5C	48	52	5A	39
A	5C	5C	5C	5C	5B	5D	7C	3A	5C	5C	5C	5C	5C	5C	5C	5C
B	0B	5C	5C	5C	2E	24	2C	23	5C	5C	5C	5C	5C	5C	5C	5C
C	0C	1C	5C	14	3C	2A	25	40	5C	5C	5C	5C	5C	5C	5C	5C
D	0D	1D	05	15	28	29	5F	27	5C	5C	5C	5C	5C	5C	5C	5C
E	0E	1E	06	5C	2B	3B	3E	3D	5C	5C	5C	5C	5C	5C	5C	5C
F	0F	1F	07	1A	5D	5E	5C	22	5C	5C	5C	5C	5C	5C	5C	FF

※\*1 は%X'5C'

---

## 6.4 DEC 漢字コード変換ライブラリ・ルーチン

DEC 漢字コード変換ライブラリ・ルーチンは、DEC 漢字セット中の漢字コードを、同じDEC 漢字セット中の他の漢字コードに変換するためのルーチン群です。

---

## JLB\$KCV\_BEGIN\_CONV

変換指定ファイルを読み込みます。JLB\$KCV\_FIND, JLB\$KCV\_CONVERT に先立ってコールしなければなりません。複数の変換指定ファイルを同時に使用する場合には ident の値を変えてください。

---

### フォーマット

```
ret-status.wlc.v = JLB$KCV_BEGIN_CONV ( file-spec.rt.dx [,err_type.wl.r [,err_line_no.wl.r [,ident.rl.r]]] )
```

---

### 引数

file-spec		
	JLB usage	変換指定テーブルのファイル名
	mechanism	Descriptor 渡し
err_type		
	JLB usage	戻り値が JSY\$_INVFORMAT のときにエラーの種類をセットします。 1: フォーマット・エラー 2: 変換ファイルに重複したコードがある。 3: 変換ファイルのデータが 65,536 以上ある (重複したコードを指定しない限りありえない)。
	mechanism	Reference 渡し
err_line_no		
	JLB usage	戻り値が JSY\$_INVFORMAT のときにエラーのあるライン番号をセットします。
	mechanism	Reference 渡し
ident		
	JLB usage	変換指定ファイルを一意にするための番号を指定します。 省略時には 0 とします。
	mechanism	Reference 渡し

---

### 戻り値

ret-status

JLB\$KCV\_BEGIN\_CONV

SS\$_NORMAL	正常終了
JSY\$_INVARG	引数エラー
JSY\$_INVCALL	ルーチンの呼び出し順序エラー
JSY\$_INVFORMAT	変換指定ファイルにエラー
RMS\$_...	RMS エラー
LIB\$_...	LIB\$GET_VM()エラー
	LIB\$FREE_VM()エラー



---

## JLB\$KCV\_END\_CONV

ident で指定された番号の変換指定ファイルによる漢字コード変換を終了します。

---

### フォーマット

```
ret-status.wc.v = JLB$KCV_END_CONV ([ident.rl.r]#)
```

---

### 引数

ident	JLB usage	JLB\$KCV_BEGIN_CONV で指定した番号を指定します。 省略時には0 とします。
	mechanism	Reference 渡し

---

### 戻り値

ret-status	
SS\$_NORMAL	正常終了
JSY\$_INVARG	引数エラー
JSY\$_INVCALL	ルーチンの呼び出し順序エラー

---

## JLB\$KCV\_FIND

変換指定テーブルの変換指定行および通知指定行で指定された漢字コードがあるかどうかを検査して通知します。コード変換は行いません。

---

### フォーマット

```
ret-status.wlc.v = JLB$KCV_FIND ( src-str.rt.dx, index.wl.r, match-len.wl.r [,com-type.wl.r [,mode.rwl.r
[,rep-str.wt.dx [,rep-len.wl.r [,ident.rl.r]]]] )
```

---

### 引数

src-str		
	JLB usage	入力文字列
	mechanism	Descriptor 渡し
index		
	JLB usage	一致した文字列の始まる位置を入力文字列の先頭からのオフセットをセットします。一致する文字列がなければ入力文字列の長さをセットします。
	mechanism	Reference 渡し
match-len		
	JLB usage	一致した文字列の長さをセットします。
	mechanism	Reference 渡し
com-type		
	JLB usage	一致した文字列が変換指定であるか通知指定であるかをセットします。 1 : 通知指定 2 : 変換指定
	mechanism	Reference 渡し
mode		
	JLB usage	指定されたモードで検索を開始します。 0 : 漢字 1 : 半角カタカナ 省略時は漢字モード
	mechanism	Reference 渡し
rep-str		

	JLB usage	一致した文字列が変換指定のときに、変換指定ファイルの変換後文字列をセットします。
	mechanism	Descriptor 渡し
rep-len	JLB usage	一致した文字列が変換指定のときに、変換指定ファイルの変換後文字列の長さをセットします。
	mechanism	Reference 渡し
ident	JLB usage	JLB\$KCV_BEGIN_CONV で指定した番号を指定します。 省略時には0とします。
	mechanism	Reference 渡し

---

## 戻り値

ret-status	
SS\$_NORMAL	正常終了
JSY\$_KEYNOTFND	一致する文字列がなかった
JSY\$_INVARG	引数エラー
JSY\$_INVCALL	ルーチンの呼び出し順序エラー
LIB\$_...	LIB\$SCOPY_R_DX()エラー

---

## JLB\$KCV\_CONVERT

変換指定テーブルの変換指定行にしたがって、漢字コード変換を行います。

---

### フォーマット

```
ret-status.wc.v = JLB$KCV_CONVERT ( dst-str.wt.dx, src-str.rt.dx [,mode.rwl.r [,out-len.wl.r [,ident.rl.r]]] )
```

---

### 引数

dst-str	JLB usage	出力文字列
	mechanism	Descriptor 渡し
src-str	JLB usage	入力文字列
	mechanism	Descriptor 渡し
mode	JLB usage	指定されたモードで検索・変換を開始します。 0 : 漢字 1 : 半角カタカナ 省略時は漢字モード。
	mechanism	Reference 渡し
	out-len	
out-len	JLB usage	変換結果のバイト長
	mechanism	Reference 渡し
ident	JLB usage	JLB\$KCV_BEGIN_CONV で指定した番号を指定します。 省略時には 0 とします。
	mechanism	Reference 渡し

---

### 戻り値

ret-status

SS\$_NORMAL	正常終了
JSY\$_INVARG	引数エラー
JSY\$_INVCALL	ルーチンの呼び出し順序エラー
LIB\$_...	LIB\$SCOPY_R_DX()エラー
	LIB\$GET_VM()エラー

## 変換キー配列

ここでは、漢字変換入カルーチン (JLB\$GET\_COMMAND, JLB\$GET\_INPUT, JLB\$GET\_SCREEN) およびそれを使用している日本語ユーティリティ (KINQUIRE, JMAIL など) における変換キーの配列を示します。

日本語処理ライブラリの漢字変換入カルーチン (JLB\$GET\_COMMAND, JLB\$GET\_INPUT, JLB\$GET\_SCREEN) およびそれを使用している日本語ユーティリティ (KINQUIRE, JMAIL など) では変換キーを次のように割り当てています。

表 A-1 コントロール・キーを使用した変換キー一覧

使用するキー	機能
Ctrl/スペース	漢字変換/文節次候補
Ctrl/F	全角変換
Ctrl/K	カタカナ変換/文節カタカナ変換
Ctrl/L	ひらがな変換/文節ひらがな変換
Ctrl/N	無変換
Ctrl/P	次文節移動
Ctrl//	文節縮小
Ctrl/]	記号変換/コード入力
Ctrl/G + Ctrl/スペース	文節前候補
Ctrl/G + Ctrl/F	半角変換
Ctrl/G + Ctrl/P	前文節移動
Ctrl/G + Ctrl//	文節拡大
Ctrl/G + Ctrl/K	漢字変換/文節次候補
Ctrl/G + Ctrl/L	文節前候補

図 A-1 LK401-JJ , LK401BJ の変換キーの機能一覧

使用するキー	機能
前候補 変換 (次候補)	漢字変換/文節次候補
Shift + 前候補 変換 (次候補)	文節前候補
カタカナ ひらがな	ひらがな変換/文節ひらがな変換
Shift + カタカナ ひらがな	カタカナ変換/文節カタカナ変換
無変換	無変換
Shift + ←	文節縮小
Shift + →	文節拡大
Shift + ↑	前文節移動
Shift + ↓	次文節移動

図 A-2 数字キーパッドを使用した変換キー配置





## 変換キー配列

### 変換キーの機能

漢字変換	ローマ字/かな次候補	漢字
ひらがな変換	ローマ字/かな	ひらがな
カタカナ変換	ローマ字/かな	カタカナ
全角変換	ローマ文字半角	ローマ文字全角
半角変換	ローマ文字全角	ローマ文字半角
記号変換	ローマ文字半角	特殊記号
無変換	変換無し	
左/右移動	変換対象文字列で一文字左/右に移動します。 文字列修正後に再変換が可能です。	
先頭/最後移動	変換対象文字列の先頭/最後に移動します。 文字列修正後に再変換が可能です。	
左/右字削除	現在のカーソル位置より左/右の文字を削除します。	

# B

---

## 変換対応表

ここでは、日本語ユーティリティおよび日本語処理ライブラリ内の変換ルーチンで行う各種変換の対応表を示します。

ローマ字/かな変換対応表

ローマ字半角/全角変換コード対応表

記号変換対応表

---

### B.1 ローマ字/かな変換対応表

変換ルーチン

LIB\$TRA\_ROM\_KANA

JSY\$TRA\_ROM\_KANA

変換対応表  
B.1 ローマ字/かな変換対応表

図 B-1 ローマ字/かな変換対応表

ローマ字					ひらがな				
a	i	u	e	o	あ	い	う	え	お
ka	ki	ku	ke	ko	か	き	く	け	こ
qa	qi	qu	qe	qo	く <small>あ</small>	く <small>い</small>	く	く <small>え</small>	く <small>お</small>
sa	si	su	se	so	さ	し	す	せ	そ
ta	ti	tu	te	to	た	ち	つ	て	と
na	ni	nu	ne	no	な	に	ぬ	ね	の
ha	hi	hu	he	ho	は	ひ	ふ	へ	ほ
fa	fi	fu	fe	fo	ふ <small>あ</small>	ふ <small>い</small>	ふ	ふ <small>え</small>	ふ <small>お</small>
ma	mi	mu	me	mo	ま	み	む	め	も
ya	yi	yu	ye	yo	や	い	ゆ	え	よ
ra	ri	ru	re	ro	ら	り	る	れ	ろ
*la	li	lu	le	lo	ら	り	る	れ	ろ
wa	wi	wu	we	wo	わ	ゐ	う	ゑ	を
nn					ん				
xa	xi	xu	xe	xo	あ	い	う	え	お
xka			xke		カ			ケ	
		xtu					っ		
		xtsu					っ		
xwa		xwu			わ		う		
xya	xyi	xyu	xye	xyo	や	い	ゆ	え	よ
ga	gi	gu	ge	go	が	ぎ	ぐ	げ	ご
za	zi	zu	ze	zo	ざ	じ	ず	ぜ	ぞ
ja	ji	ju	je	jo	じ <small>ゃ</small>	じ	じ <small>ゅ</small>	じ <small>え</small>	じ <small>ょ</small>
da	di	du	de	do	だ	ぢ	づ	で	ど
ba	bi	bu	be	bo	ば	び	ぶ	べ	ぼ
pa	pi	pu	pe	po	ぱ	ぴ	ぷ	ぺ	ぽ
va	vi	vu	ve	vo	ヴァ	ヴィ	ヴ	ヴェ	ヴォ

(次ページへつづく)

\* **TARO** キーパッドで **SET ROMKANA TARO** コマンドを実行した場合、あいうえお(小さな文字)になります。

図 B-2 ローマ字/かな変換対応表（つづき）

ローマ字					ひらがな				
kya	kyi	kyu	kye	kyo	きゃ	きい	きゅ	きえ	きよ
qya	qyi	qyu	qye	qyo	くゃ	くい	くゅ	くえ	くよ
kwa	kwi	kwu	kwe	kwo	くわ	くい	く	くえ	くお
sya	syi	syu	sye	syo	しゃ	しい	しゅ	しえ	しよ
sha	shi	shu	she	sho	しゃ	し	しゅ	しえ	しよ
tya	tyi	tyu	tye	tyo	ちゃ	ちい	ちゅ	ちえ	ちよ
cya	cyi	cyu	cye	cyo	ちゃ	ちい	ちゅ	ちえ	ちよ
cha	chi	chu	che	cho	ちゃ	ち	ちゅ	ちえ	ちよ
tsa	tsi	tsu	tse	tso	つあ	つい	つ	つえ	つお
tha	thi	thu	the	tho	てゃ	てい	てゅ	てえ	てよ
nya	nyi	nyu	nye	nyo	にゃ	にい	にゅ	にえ	によ
hya	hyi	hyu	hye	hyo	ひゃ	ひい	ひゅ	ひえ	ひよ
fya	fyi	fyu	fye	fyo	ふゃ	ふい	ふゅ	ふえ	ふよ
mya	myi	myu	mye	myo	みゃ	みい	みゅ	みえ	みよ
rya	ryi	ryu	rye	ryo	りゃ	りい	りゅ	りえ	りよ
lya	lyi	lyu	lye	lyo	りゃ	りい	りゅ	りえ	りよ
gya	gyi	gyu	gye	gyo	ぎゃ	ぎい	ぎゅ	ぎえ	ぎよ
gwa	gwi	gwu	gwe	gwo	ぐわ	ぐい	ぐ	ぐえ	ぐお
zya	zyi	zyu	zye	zyo	じゃ	じい	じゅ	じえ	じよ
jya	jyi	jyu	jye	jyo	じゃ	じい	じゅ	じえ	じよ
dya	dyi	dyu	dye	dyo	ぢゃ	ぢい	ぢゅ	ぢえ	ぢよ
dha	dhi	dhu	dhe	dho	でゃ	でい	でゅ	でえ	でよ
bya	byi	byu	bye	byo	びゃ	びい	びゅ	びえ	びよ
pya	pyi	pyu	pye	pyo	ぴゃ	ぴい	ぴゅ	ぴえ	ぴよ

## 変換対応表

### B.1 ローマ字/かな変換対応表

- \* ローマ字は大文字でもかまいません。
- \* つまる音（“っ”）は次にくる子音を重ねます。はねる音（“ん”）はnです。
- \* 小文字は図 B-1 のように x を使って示すこともできます。
- \* “ん” の次に母音がくるときには “ ’ ”（アポストロフィ）を間にいれます。
- \* 例えばこのように変換できます。

nippon	にっぽん
an'an	あんあん
Kemma	けんま
itchi	いっち
nonno	のんの
rombun	ろんぶん

ローマ字・かな漢字変換ルーチンでは、次の特殊文字変換も行います。

図 B-3 特殊文字変換対応表

変換前	漢字・ひらがな・カタカナ	全角変換
；（セミコロン）	、（読点）	；
．（ピリオド）	。（句点）	．
＋（正符号）	・（中点）	＋
－（負符号，ハイフン）	ー（長音記号）	－
＼（バック・スラッシュ） または ¥（円記号）	＼（逆斜線）	¥
‘（左一重引用符） <sup>1</sup>	‘（左一重引用符）	、
～（ティルダ）	～（波ダッシュ）	
@（単価記号）	（二重丸）	@
*（星印）	（米印）	*
=（等号）	＝（げた記号）	=
[（始め大カッコ）	「（始めかぎ括弧）	[
]（終り大カッコ）	」（終りかぎ括弧）	]
<（不等号，より小）	（始め山括弧）	<
>（不等号，より大）	（終り山括弧）	>

## 変換対応表

### B.1 ローマ字/かな変換対応表

- TARO キーパッドの省略時設定では、このキーがHENKAN MODEコマンドに定義されています。左一重引用符を入力したい場合は、以下のようにUNDEFINE KEYコマンドで定義を取り消してください。

Command: UNDEFINE KEY

キーを押してください: ( `キーを押す)

キー定義が取り消されました

再びHENKAN MODEコマンドをキーに定義したい場合は、DEFINE KEYコマンドで別のキーに定義してください。以下の例では、`[GOLD]+[F7]`に定義しています。

Command: DEFINE KEY

JEVE command: HENKAN MODE

定義するキーを押してください: (`[GOLD]`キーを押してから、`[F7]`キーを押す)

キーが定義されました

---

## B.2 ローマ字半角/全角コード変換対応表

ここでは、ローマ字半角/全角コード変換対応表を示します。

変換ルーチン

JLB\$TRA_ROM_HALF	JSY\$TRA_ROM_HALF	JSY\$CHG_ROM_HALF
JLB\$TRA_ROM_FULL	JSY\$TRA_ROM_FULL	JSY\$CHG_ROM_FULL
JLB\$TRA_ROM_SIZE	JSY\$TRA_ROM_SIZE	JSY\$CHG_ROM_SIZE

表 B-1 ローマ字半角/全角コード変換対応表

半角	全角	半角	全角	半角	全角
20	A1A1	@ 40	@ A1F7	‘ 60	` A1AE
! 21	! A1AA	A 41	A A3C1	a 61	a A3E1
" 22	" A1C9	B 42	B A3C2	b 62	b A3E2
# 23	# A1F4	C 43	C A3C3	c 63	c A3E3
\$ 24	\$ A1F0	D 44	D A3C4	d 64	d A3E4
% 25	% A1F3	E 45	E A3C5	e 65	e A3E5
& 26	& A1F5	F 46	F A3C6	f 66	f A3E6
' 27	' A1C7	G 47	G A3C7	g 67	g A3E7
( 28	( A1CA	H 48	H A3C8	h 68	h A3E8
) 29	) A1CB	I 49	I A3C9	i 69	i A3E9
* 2A	* A1F6	J 4A	J A3CA	j 6A	j A3EA
+ 2B	+ A1DC	K 4B	K A3CB	k 6B	k A3EB
, 2C	, A1A4	L 4C	L A3CC	l 6C	l A3EC
- 2D	- A1DD	M 4D	M A3CD	m 6D	m A3EE
. 2E	. A1A5	N 4E	N A3CE	n 6E	n A3ED
/ 2F	/ A1BF	O 4F	O A3CF	o 6F	o A3EF
0 30	0 A3B0	P 50	P A3D0	p 70	p A3F0
1 31	1 A3B1	Q 51	Q A3D1	q 71	q A3F1
2 32	2 A3B2	R 52	R A3D2	r 72	r A3F2
3 33	3 A3B3	S 53	S A3D3	s 73	s A3F3
4 34	4 A3B4	T 54	T A3D4	t 74	t A3F4
5 35	5 A3B5	U 55	U A3D5	u 75	u A3F5
6 36	6 A3B6	V 56	V A3D6	v 76	v A3F6
7 37	7 A3B7	W 57	W A3D7	w 77	w A3F7
8 38	8 A3B8	X 58	X A3D8	x 78	x A3F8
9 39	9 A3B9	Y 59	Y A3D9	y 79	y A3F9
: 3A	: A1A7	Z 5A	Z A3DA	z 7A	z A3FA
; 3B	; A1A8	[ 5B	[ A1CE	{ 7B	{ A1D0
< 3C	< A1E3	¥ 5C	¥ A1EF	7C	A1C3
= 3D	= A1E1	] 5D	] A1CF	} 7D	} A1D1
> 3E	> A1E4	^ 5E	^ A1B0	~ 7E	A1B1
? 3F	? A1A9	_ 5F	_ A1B2		



---

## B.3 記号変換対応表

ここでは、記号変換対応表を示します。

変換ルーチン

JLB\$TRA\_SYMBOL

JSY\$TRA\_SYMBOL

次のような変換があります。

- 1文字変換
- 2文字変換
- 3文字変換
- 区点コード変換
- 16進コード変換

### B.3.1 1文字変換

図 B-4 1文字変換

変換前	1	2	3	4	5	6	7	8	9	0	-
変換後											

### B.3.2 2文字変換

T1, T3, T7, T9, T- は、縦横共に太い罫線に変換されます。Y1, Y3, Y7, Y9, Y0 は、縦横共に太い罫線に変換されます。

图 B-5 2 文字变换

变换前	f 1	f 2	f 3	f 4	f 5	f 6	f 7	f 8	f 9	f 0	f -
变换後											
变换前	t 1	t 2	t 3	t 4	t 5	t 6	t 7	t 8	t 9	t 0	t -
变换後											
变换前	y 1	y 2	y 3	y 4	y 5	y 6	y 7	y 8	y 9	y 0	y -
变换後											

変換対応表  
B.3 記号変換対応表

図 B-6 2文字変換(つづき)

変換前	SS	><	( )	C)	<>	[ ]	<\	</	=	>-	<-	^	V
変換後	§								〒				
変換前	^i	/]	/^	/_	l_	~ ~	o)	~v	=-	=:	*<	*>	v~
変換後	全	々	ノ										
変換前	cc	oc	;.	s1	s2	(- )-	(_ )_	(( ))	cu	ca			
変換後													
変換前	/ \	\ /	~	=>	==	v-	] -	.1	'1	'2	.a	%%	##
変換後			ㄣ					°				%o	
変換前	.	.^	+1	+2	]								
変換後			†	‡	¶								

### B.3.3 3文字変換

図 B-7 3文字変換

変換前	k ><	k ( )	k < >	k [ ]	k < \	k < /
変換後						

### B.3.4 区点コード変換

“J” に続けて区番号 (1 ~ 3 けた) , 点番号 (2 けた)

表 B-2 区点コード変換

区番号	1 ~ 94	DEC 漢字セット
	101 ~ 194	拡張領域
点番号	1 ~ 94	

例)

J125                      “々”  
 区番号                  1  
 点番号                  25

### B.3.5 16進コード変換

“X” に続けて 16 進 2 バイト・コード

表 B-3 16 進コード変換

---

第 1 バイト	A1 ~ FE	
第 2 バイト	A1 ~ FE	DEC 漢字セット
	21 ~ 7E	拡張領域

---

例)

XA1B9                    “々”  
第 1 バイト    A1  
第 2 バイト    B9

# 索引

## A

ASCII コード . . . . . 6-33, 6-34, 6-45

## D

DEC 漢字コード

変換ライブラリ・ルーチン一覧 . . . . . 6-3

DEC 漢字コードへの変換 . . . . . 4-49

DEC 漢字コード変換

ライブラリ・ルーチン . . . . . 6-46

JLB\$KCV\_BEGIN\_CONV . . . . . 6-47

JLB\$KCV\_CONVERT . . . . . 6-52

JLB\$KCV\_END\_CONV . . . . . 6-49

JLB\$KCV\_FIND . . . . . 6-50

DEC コード . . . . . 6-4

## E

EBCDIK コード . . . . . 6-43, 6-45

## J

JIS コード . . . . . 6-4

JLB\$ROM\_SIZE . . . . . 3-32

JLB\$CNV\_CLAUSE\_DELETE . . . . . 5-37

JLB\$CNV\_CLAUSE\_FULL . . . . . 5-29

JLB\$CNV\_CLAUSE\_HALF . . . . . 5-31

JLB\$CNV\_CLAUSE\_HIRAGANA . . . . . 5-25

JLB\$CNV\_CLAUSE\_KATAKANA . . . . . 5-27

JLB\$CNV\_CLAUSE\_NOCONVERT . . . . . 5-35

JLB\$CNV\_CLAUSE\_SYMBOL . . . . . 5-33

JLB\$CNV\_CLOSE\_DICTIONARY . . . . . 5-6

JLB\$CNV\_CONVERT . . . . . 5-7

JLB\$CNV\_DELETE\_WORD . . . . . 5-43

JLB\$CNV\_EXTEND\_CLAUSE . . . . . 5-23

JLB\$CNV\_GET\_CLAUSE . . . . . 5-17

JLB\$CNV\_GET\_KANJI . . . . . 5-11

JLB\$CNV\_IO\_ERROR . . . . . 5-45

JLB\$CNV\_LEARN . . . . . 5-39

JLB\$CNV\_NEXT\_WORD . . . . . 5-13

JLB\$CNV\_OPEN\_DICTIONARY . . . . . 5-4

JLB\$CNV\_PREV\_WORD . . . . . 5-15

JLB\$CNV\_REGISTER\_WORD . . . . . 5-41

JLB\$CNV\_ROM\_CONVERT . . . . . 5-9

JLB\$CNV\_SHORTEN\_CLAUSE . . . . . 5-21

JLB\$DATE\_TIME . . . . . 3-69

JLB\$DEL\_TANGO . . . . . 5-53

JLB\$DEV\_KANJI . . . . . 3-72

JLB\$DEV\_KANJI\_IN . . . . . 3-73

JLB\$ENT\_TANGO . . . . . 5-51

JLB\$GET\_COMMAND . . . . . 3-50

JLB\$GET\_INPUT . . . . . 3-52

JLB\$LOCC . . . . . 3-22

JLB\$NOF\_BYTE . . . . . 3-4

JLB\$POS\_CURR . . . . . 3-7

JLB\$POS\_NEXT . . . . . 3-8

JLB\$POS\_PREV . . . . . 3-10

JLB\$POSITION . . . . . 3-11

JLB\$RCHAR . . . . . 3-25

JLB\$RMS\_USER\_VTF7 . . . . . 3-55

JLB\$RMS\_VTF7\_USER . . . . . 3-57

JLB\$RNEXT . . . . . 3-26

JLB\$SKPC . . . . . 3-24

JLB\$STR\_EQUAL . . . . . 3-12

JLB\$STR\_SEARCH . . . . . 3-16

JLB\$STR\_START . . . . . 3-14

JLB\$TRA\_DICCLS . . . . . 5-50

JLB\$TRA\_DICINI . . . . . 5-49

JLB\$TRA\_KANA\_DAKU . . . . . 3-42

JLB\$TRA\_KANA\_FULL . . . . . 3-45

JLB\$TRA\_KANA\_HALF . . . . . 3-43

JLB\$TRA\_KANA\_HIRA . . . . . 3-39

JLB\$TRA\_KANA\_KANA . . . . . 3-41

JLB\$TRA\_KANA\_KATA . . . . . 3-40

JLB\$TRA\_KANA\_TANGO . . . . . 5-56

JLB\$TRA\_ROM\_CASE . . . . . 3-35

JLB\$TRA\_ROM\_FULL . . . . . 3-31

JLB\$TRA\_ROM\_HALF . . . . . 3-30

JLB\$TRA\_ROM\_KANA . . . . . 3-36

JLB\$TRA\_ROM\_LOWER . . . . . 3-33

JLB\$TRA\_ROM\_TANGO . . . . . 5-54

JLB\$TRA\_ROM\_UPPER . . . . . 3-34

JLB\$TRA\_SYMBOL . . . . . 3-47

JLB\$TRA\_TANGO\_DONE . . . . . 5-62

JLB\$TRA\_TANGO\_NEXT . . . . . 5-58

JLB\$TRA\_TANGO\_PREV . . . . . 5-60

JLB\$TRIM . . . . . 3-18

JLB\$TRUNC . . . . . 3-20

JLB\$WCHAR . . . . . 3-27

JLB\$WNEXT . . . . . 3-28

JSY\$CH\_CURR . . . . . 4-7

JSY\$CH\_GCHAR . . . . . 4-16

JSY\$CH\_GNEXT . . . . . 4-17

JSY\$CH_NBYTE	4-22	JSY\$RMS_USER_VTF7	3-64
JSY\$CH_NCHAR	4-23	JSY\$RMS_VTF7_USER	3-66
JSY\$CH_NEXT	4-4	JSY\$SKPC	4-34
JSY\$CH_PCHAR	4-18	JSY\$STR_EQUAL	4-25
JSY\$CH_PNEXT	4-19	JSY\$STR_SEARCH	4-29
JSY\$CH_PREV	4-6	JSY\$STR_START	4-27
JSY\$CH_RCHAR	4-12	JSY\$TRA_DICCLS	5-50
JSY\$CH_RNEXT	4-13	JSY\$TRA_DICINI	5-49
JSY\$CH_SIZE	4-21	JSY\$TRA_KANA_DAKU	4-63
JSY\$CH_WCHAR	4-14	JSY\$TRA_KANA_FULL	4-65
JSY\$CH_WNEXT	4-15	JSY\$TRA_KANA_HALF	4-64
JSY\$CHG_GENERAL	4-48	JSY\$TRA_KANA_HIRA	4-60
JSY\$CHG_JIS_KUTEN	4-49	JSY\$TRA_KANA_KANA	4-62
JSY\$CHG_KANA_DAKU	4-45	JSY\$TRA_KANA_KATA	4-61
JSY\$CHG_KANA_FULL	4-47	JSY\$TRA_KANA_TANGO	5-56
JSY\$CHG_KANA_HALF	4-46	JSY\$TRA_ROM_CASE	4-57
JSY\$CHG_KANA_HIRA	4-42	JSY\$TRA_ROM_FULL	4-53
JSY\$CHG_KANA_KANA	4-44	JSY\$TRA_ROM_HALF	4-52
JSY\$CHG_KANA_KATA	4-43	JSY\$TRA_ROM_KANA	4-58
JSY\$CHG_KEISEN	4-50	JSY\$TRA_ROM_LOWER	4-55
JSY\$CHG_ROM_CASE	4-41	JSY\$TRA_ROM_SIZE	4-54
JSY\$CHG_ROM_FULL	4-37	JSY\$TRA_ROM_UPPER	4-56
JSY\$CHG_ROM_HALF	4-36	JSY\$TRA_SYMBOL	4-67
JSY\$CHG_ROM_LOWER	4-39	JSY\$TRA_TANGO_DONE	5-62
JSY\$CHG_ROM_SIZE	4-38	JSY\$TRA_TANGO_NEXT	5-58
JSY\$CHG_ROM_UPPER	4-40	JSY\$TRA_TANGO_PREV	5-60
JSY\$CNV_CLAUSE_DELETE	5-37	JSY\$TRIM	4-31
JSY\$CNV_CLAUSE_FULL	5-29	JSY\$TRUNC	4-32
JSY\$CNV_CLAUSE_HALF	5-31		
JSY\$CNV_CLAUSE_HIRAGANA	5-25	<b>L</b>	
JSY\$CNV_CLAUSE_KATAKANA	5-27	LIB\$GET_COMMAND	3-50
JSY\$CNV_CLAUSE_NOCONVERT	5-35	LIB\$GET_INPUT	3-52
JSY\$CNV_CLAUSE_SYMBOL	5-33		
JSY\$CNV_CLOSE_DICTIONARY	5-6	<b>M</b>	
JSY\$CNV_CONVERT	5-7	MSDOS コード	6-39, 6-41
JSY\$CNV_DELETE_WORD	5-43		
JSY\$CNV_EXTEND_CLAUSE	5-23	<b>N</b>	
JSY\$CNV_GET_CLAUSE	5-17	NEC コード	6-35, 6-37
JSY\$CNV_GET_KANJI	5-11		
JSY\$CNV_GET_PHONETIC_CLAUSE	5-19	<b>R</b>	
JSY\$CNV_IO_ERROR	5-45	RMS ファイル名コンバータ	
JSY\$CNV_LEARN	5-39	状態	3-59
JSY\$CNV_NEXT_WORD	5-13	無効	3-62
JSY\$CNV_OPEN_DICTIONARY	5-4	有効	3-62
JSY\$CNV_PREV_WORD	5-15		
JSY\$CNV_REGISTER_WORD	5-41	<b>V</b>	
JSY\$CNV_SHORTEN_CLAUSE	5-21	VTF-7 コードセット	3-55, 3-57, 3-64, 3-66
JSY\$DEL_TANGO	5-53		
JSY\$ENT_TANGO	5-51		
JSY\$LOCC	4-33		
JSY\$POS_CURR	4-11		
JSY\$POS_NEXT	4-8		
JSY\$POS_PREV	4-9		
JSY\$POSITION	4-24		
JSY\$RMS_GET_ENCODING	3-59		
JSY\$RMS_LIST_ENCODING	3-60		
JSY\$RMS_SET_ENCODING	3-62		

## イ

- 1文字変換 . . . . . B-8
- インデックスの更新 . . . . . 3-26, 3-28

## オ

- 大文字/小文字の相互変換 . . . . . 4-41, 4-57
- 大文字の相互変換 . . . . . 3-35
- 大文字への変換 . . . . . 3-34, 4-40, 4-56
- オブジェクト・ライブラリ・ファイル . . . . . 1-1

## カ

- カタカナへの変換 . . . . . 3-36, 3-45
- かな漢字変換 . . . . . 5-7
  - 辞書 . . . . . 5-63
  - ライブラリ . . . . . 1-1, 5-1
    - ルーチン一覧 . . . . . 5-1
- かな単語変換 . . . . . 5-56
- 漢字コード変換 . . . . . 6-49, 6-50, 6-52
  - CP/M . . . . . 6-1
  - HITACHI . . . . . 6-1
  - IBM . . . . . 6-1
  - JEF . . . . . 6-1
  - JIS . . . . . 6-1
  - MSDOS . . . . . 6-1
  - NEC . . . . . 6-1
  - ライブラリ . . . . . 1-1, 6-1
    - ルーチン一覧 . . . . . 6-2
  - ルーチン . . . . . 6-2, 6-3
    - JLB\$TRA\_KANJI\_DEC\_HITACHI . . . . . 6-8
    - JLB\$TRA\_KANJI\_DEC\_JIS . . . . . 6-4
    - JLB\$TRA\_KANJI\_DEC\_NEC . . . . . 6-20, 6-22, 6-24, 6-26, 6-28, 6-30
    - JLB\$TRA\_KANJI\_HITACHI\_DEC . . . . . 6-10, 6-12, 6-14, 6-16, 6-18
    - JLB\$TRA\_KANJI\_JIS\_DEC . . . . . 6-6
- 漢字端末 . . . . . 3-72
- 漢字入力端末 . . . . . 3-73
- 漢字変換入力ルーチン . . . . . 3-2, 3-49, A-1
- 漢字文字列要求ルーチン . . . . . 5-11

## キ

- 記号変換 . . . . . 3-47, 4-67
  - 対応表 . . . . . B-8
- 基本ライブラリ . . . . . 1-1, 4-1
  - ルーチン . . . . . 4-1
- 共有イメージ・ファイル . . . . . 1-1

## ク

- 空白の切り捨て . . . . . 3-18, 4-31
- 区点コード変換 . . . . . B-8, B-11

## ケ

- 罫線文字への変換 . . . . . 4-50
- 元号 . . . . . 3-70

## コ

- 個人辞書 . . . . . 5-63
  - 学習モード . . . . . 5-64
  - 学習モードの選択 . . . . . 5-65
  - 共有モード . . . . . 5-64
  - 供用した場合の変換結果 . . . . . 5-68
  - 参照モード . . . . . 5-65
  - 使用モードの選択 . . . . . 5-64
  - 単語数 . . . . . 5-66
  - 不使用モード . . . . . 5-65
- 小文字の相互変換 . . . . . 3-35
- 小文字への変換 . . . . . 3-33, 4-39, 4-55
- コントロール・キーを使用した変換キー一覧 . . . . . A-1

## サ

- 3文字変換 . . . . . B-8, B-11

## シ

- 辞書
  - アクセス時のエラー . . . . . 5-68
  - オープン . . . . . 5-4, 5-49
  - クローズ . . . . . 5-6, 5-50
- 辞書 I/O ステータス・ルーチン . . . . . 5-45
- システム辞書 . . . . . 5-63
- 16進コード変換 . . . . . B-8, B-12
- 自立語
  - 削除ルーチン . . . . . 5-37
  - 次候補ルーチン . . . . . 5-13
  - 自立語前候補ルーチン . . . . . 5-15

## ス

- 数字キーパッド
  - 変換キー配置 . . . . . A-3

## セ

- 全角カタカナの相互変換 . . . . . 3-41
- 全角カタカナへの変換 . . . . . 3-40, 4-43, 4-61
- 全角の相互変換 . . . . . 3-32
- 全角/半角の相互変換 . . . . . 4-38, 4-54



全角ひらがな/カタカナへの変換	4-47, 4-58, 4-65
全角ひらがな/全角カタカナの相互変換	4-44, 4-62
全角ひらがなの相互変換	3-41
全角ひらがなへの変換	3-36, 3-39, 3-45, 4-42, 4-60
全角への変換	3-31, 4-37, 4-53

## タ

濁点/半濁点処理	3-42, 4-45, 4-63
単語	
候補の決定	5-62
削除	5-53
ルーチン	5-43
次候補	5-58
単位変換ルーチン	5-2, 5-48
登録	5-51
ルーチン	5-41
前候補	5-60

## チ

注意事項および制限事項	5-67
-------------	------

## ト

特殊文字変換対応表	B-5
-----------	-----

## ニ

日本語ファイル名ルーチン	3-2
日本語ファイル名ルーチン (Alpha のみ)	3-54
日本語文字列操作ルーチン	3-1, 3-3, 4-1, 4-3
日本語ライブラリの使用例	
C	2-2
FORTRAN	2-1
MACRO	2-2
2文字変換	B-8

## ハ

1 バイト・コード変換テーブル	6-2, 6-32
JSY\$GTBL_EBCDIK_TO_ASCII	6-45
JSY\$GTBL_TO_ASCII	6-33
JSY\$GTBL_TO_ASCII_K	6-34
JSY\$GTBL_TO_EBCDIK	6-43
JSY\$GTBL_TO_MSDOS	6-39
JSY\$GTBL_TO_MSDOS_K	6-41
JSY\$GTBL_TO_NEC	6-35
JSY\$GTBL_TO_NEC_K	6-37
バイト数の入手	3-4, 3-6, 4-21, 4-22
パラメータ指定による	
変換	4-48
半角カタカナへの変換	3-43, 4-46, 4-64

半角の相互変換	3-32
半角への変換	3-30, 4-36, 4-52
汎用ライブラリ	1-1, 3-1
ルーチン一覧	3-1

## ヒ

日付けおよび時間の入手	3-69
-------------	------

## フ

ファイル名コンバータの名前	
取得	3-60
複文節変換	
単語単位変換との混用	5-67
ルーチン	5-1, 5-3
文節	
位置情報ルーチン	5-17
学習辞書	5-63, 5-64
カタカナ変換ルーチン	5-27
記号変換ルーチン	5-33
縮小ルーチン	5-21
全角変換ルーチン	5-29
対応する入力文字列	
位置情報ルーチン	5-19
長伸長ルーチン	5-23
半角変換ルーチン	5-31
ひらがな変換ルーチン	5-25
無変換ルーチン	5-35

## ヘ

変換	
開始と終了	5-67
確定ルーチン	5-39
キー	
機能	A-4
カタカナ変換	A-4
漢字変換	A-4
記号変換	A-4
全角変換	A-4
先頭/最後移動	A-4
半角変換	A-4
左/右移動	A-4
左/右字削除	A-4
ひらがな変換	A-4
無変換	A-4
配列	A-1
指定ファイル	6-47
対応表	B-1
ルーチン	B-1, B-6, B-8
JLB\$TRA_ROM_FULL	B-6
JLB\$TRA_ROM_HALF	B-6
JLB\$TRA_ROM_SIZE	B-6
JSY\$CHG_ROM_FULL	B-6
JSY\$CHG_ROM_HALF	B-6
JSY\$CHG_ROM_SIZE	B-6
JSY\$TRA_ROM_FULL	B-6

変換	
ルーチン (続き)	
JSY\$TRA_ROM_HALF	B-6
JSY\$TRA_ROM_SIZE	B-6

## ホ

---

ポインタの更新	4-13, 4-15, 4-17, 4-19
---------	------------------------

## モ

---

### 文字

位置の入手	3-7, 3-8, 3-10, 4-4, 4-6, 4-7, 4-8, 4-9, 4-11
書き込み	3-27, 3-28, 4-14, 4-15, 4-18, 4-19
検索	3-22, 4-33
数の入手	4-23
飛び越し	3-24, 4-34
取り出し	3-25, 3-26, 4-12, 4-13, 4-16, 4-17
変換ルーチン	4-2
文字数の入手	3-6
文字変換ルーチン	4-35
文字列	
切り捨て	3-20, 4-32

検索	3-16, 4-24, 4-29
チェック	4-8, 4-9, 4-11, 4-16, 4-17, 4-18, 4-19
比較	3-12, 3-14, 4-25, 4-27
変換ルーチン	3-2, 3-29, 4-2, 4-51

## ユ

---

ユーザ定義	5-41
-------	------

## ラ

---

ライブラリ構成	1-1
---------	-----

## リ

---

リンク方法	1-2
-------	-----

## ロ

---

ローマ字・かな漢字変換	3-50, 3-52
ローマ字・かな単語変換	5-54
ローマ字/かな変換対応表	B-1
ローマ字漢字変換	5-9
ローマ字半角/全角コード変換対応表	B-6



日本語 OpenVMS  
日本語ライブラリ 利用者の手引き

---

2009年 11月 発行

日本ヒューレット・パカード株式会社

〒102-0076 東京都千代田区五番町 7 番地

電話 (03)3512-5700 (大代表)