Micro VMSに移植したREWMOLにおける日本語機能の実現

石原好宏*・刈谷丈治**

Implementation of the Facilities for Kanji Character Processing in the REWMOL Interpreter, Transported to Micro VMS

Yoshihiro ISHIHARA and Joji KARIYA

Abstract

The REWMOL processor (REal World MOdeling Language) is an interpreter written in C language. In the course of developing its original version, we had not paid any special attention to the processing of kanji characters. The reasons are, among others, 1) its development environment has shifted several times, and 2) no standard has yet been established for processing kanji characters by computer.

We transported recently the latest version V4.0r of the interpreter from UNIPLUS 〈Ustation〉 to Micro VMS 〈Micro VAX II〉. We took this opportunity to consider processing kanji characters, and made an effort to put such facilities in the transported version of the interpreter. The changes related mainly to character I/O and string data manipulation.

This report is an outline of this augmentaion—basic attitude of the original version toward these problems, our temporal lines of adding kanji character processing, our implementation of it, and some discussion.

1．まえがき

REWMOLは実世界モデルを記述するために開発した言語である．その処理系はインタープリタとして実現し，C言語で記述した．この処理系とその機能[1]，及びオペレーティングシステム（OSと略記）VAX/VMS（ハードウェアはVAX-11/730，以下同様）への処理系の移植と自然語処理のための機能拡張[2]，については既に概要を報告した．

この処理系は，VAX/VMS（VAX-11/730）へ最後に移植したバージョン4.0（V4.0と略記）が更にV4.0rに更新された．処理環境の方にもMicro VMS（Micro VAXII）が新しく加わった．これを機会に我々は，更新された処理系V4.0rを，当時の開発環境UNIPLUS（Ustation）から，新しくなった処理環境Micro VMS（Micro VAXII）の上に改めて移植した（昭和62年5月）．

この作業が完了すると，引続いて，移植後の処理系V4.0rに日本語機能を組込む作業に着手した．ここで日本語機能とは，漢字とASCII文字が混在する文字／文字列の入出力，及びそうした文字／文字列に対する操作のことである．なお，処理系V4.0rは，処理系V4.0のプログラムをわすがに手直ししたもので，機能的には両者ほとんど変わらない．

本稿では，この日本語機能の追加について報告する．まず，文字入出力と文字列操作に関するREWMOL処理系の原状と，移植先の処理環境Micro VMS（Micro VAXII）が提供する日本語機能の概要をまとめる．ついで，日本語機能の組込みに関する当面の方針と，それに基づいた実現方法を整理する．更に，これを他の日本語化の例と比較し，このたび組込んだ日本語機能の使い勝手についても言及する．

なお，以下の記述に現れる関数には，C言語のそれとREWMOL言語のそれとがある．文脈から明かな場合を除き，その都度区分を明記する．

*山口大学工業短期大学部情報処理工学科
**山口大学情報処理センター

2．REWMOL処理系と日本語処理

ここでは，REWMOL処理系の原状と移植先の処理環境を日本語処理の観点から整理し，日本語機能の組込みに関する問題点を明らかにする．

2.1 REWMOL処理系の原状

REWMOL処理系の開発環境は，最初がACOS-6 (ACOS) で，後にはUNIPLUS (Ustation)，AEGIS (Domain) と変わった．しかし，処理系の記述言語は初めからCで，これは変わっていない．

ところで，Cには最近，日本語対応の言語仕様を作る動きがあり，コンパイラも試作されている[3]．しかし，我々の手元にあるCは，いずれもKernighan，B.W.らの仕様[4]に準拠している．その言語仕様によれば，データやコメントの中では漢字が使えるものの，日本語処理に対する制約は多い．例えば文字列操作関数では，2バイトの漢字コードの場合も，外部表現の文字単位で字数を数えるわけではない．又，文字種判定用マクロ関数 (isdigit，isspace，など) では，下位7ビットだけに注目して文字を調べるので，そのままでは漢字コードを正しく識別できない．このように，現有のCで漢字を扱うのはそれほど容易ではない．

加えて，計算機環境が提供する日本語機能には今のところ統一規格がないので，各OSはそれぞれ独自の方式で処理機能を提供しているのが実状である．このため，日本語機能を処理系に組込んでしまうと，その部分の移植性は保証されなくなる．

こうした事情により，REWMOL処理系の原版の場合は，ASCIIコード(1バイト)である英数字・特殊記号・制御コードの処理だけに注目し，漢字コード（2バイト）の処理には特別の注意を払ってこなかった．このため，REWMOL処理系の改修をしないままで漢字を読込んでも，漢字が正しく処理される保証はない．

従って，REWMOL処理系V4.0rに日本語機能を組込むためには，漢字コードとASCIIコードに対する移植先OSの取扱い方に基づいて，漢字入力の機能を追加し，併せて，関連するREWMOL基本関数の機能を拡張するために実行時ルーチンの一部を改修する必要がある．

2.2 移植先の日本語処理環境

新しい処理環境で日本語機能を提供するのは，日本語Micro VMS[5]である．これは標準版のMicro VMSに日本語機能が追加されたOSで，ユーティリティ(日本語エディタ，漢字コード変換，漢字プリントなどの日本語処理を行う)，日本語ライブラリ(日本語処理のアプリケーションを作るときに有用な日本語の文字列操作，文字列変換，ローマ字・かな／漢字変換などの機能をオブジェクトライブラリと共有イメージで提供する)，辞書（単語辞書・個人辞書）等を含む[6]．

一方，こうした日本語処理用のソフトウェアを稼動させるハードウェアとしては，CPUのMicro VAX II (主記憶容量16Mb) の他，固定磁気ディスク装置〈RA81〉(456Mb)，2台の漢字端末〈VT282〉，1台の漢字プリンタ〈LA84〉がある．ちなみに，その他の入出力機器として，ASCII端末〈VT220-J〉27台，オープンリール磁気テープ装置〈TSV05〉(1600bpi)，カセット磁気テープ装置〈TK50〉(95Mb) 各1台等もある．

なお，現在の処理環境には実行時の漢字入力を支援するフロントエンドプロセッサの機能がない．ユーザプログラムへの漢字入力は，日本語ライブラリの機能をユーザ側でうまく使って表現する外ない．

3．日本語機能の組込みに関する基本方針

ここでは，新しい環境Micro VMS (Micro VAX II) に移植したREWMOL処理系V4.0rに日本語機能を組込むに当たって，我々が採った基本方針を示す．併せて，そうした方針の意図ないし意味，それを具体化するときに考慮すべき事項，などをまとめる．

(1) REWMOL処理系には，原則として高い移植性を持たせたい．しかし，我々が利用する日本語機能は，当分の間，日本語Micro VMSの処理環境が標準に提供するものに限る．

=〉日本語機能には今のところ統一的な規格がないので，この措置はやむをえない．しかし，日本語Micro VMS以外の環境で処理系V4.0rを動かす場合も想定して，そんなときには日本語機能を除いた従来の機能だけを持つREWMOL処理系が立上がるようにしてあれば便利である．

ところで，手軽に日本語入力を実現する一つの方法に，フロントエンドプロセッサの利用がある．しかし，そのような手段で入力した漢字データを未改修のユーザプログラムが正しく処理できるためには，漢字処理の機能があらかじめ組込まれている必要がある．今回の場合，事前にはそうした配慮がしてなかった．

どうせREWMOL処理系のソースプログラムに手を加えるのであれば，文字入出力と文字列操作を併せて考えても，それほど負担増とはなるまい．従ってここでは，余分の出費を抑えるためにも，

日本語ライブラリが標準に提供する機能の範囲でこれらの実現を図ることにする.

(2) プログラミング言語としてのREWMOLでは,漢字とASCII文字とを区別せず,両者が混在する文字列でも正しく取扱えること.漢字ないしASCII文字専用の関数は特に設けない.

=> 漢字を,英数字と同様に,シンボルあるいはストリング(文字列)の要素として使うためにこの方針を掲げた.但し,丸括弧( ),かぎ括弧[ ],及び空白はASCIIコードの場合に区切り記号とするのと違って,漢字コードの場合は単なる句読点として扱う.

この項は,シンボルや文字列の計数,挿入,削除,切出しなどを扱うREWMOL関数に関連がある.

(3) REWMOLで書かれた既存のプログラムが,書換えなしで,支障なく動くこと.

=> 従来の文字入出力関数と文字列操作関数が,漢字とASCII文字を同等の文字として扱えるようになれば,既存のプログラムを書換える煩わしさが回避できる.又,特別な意味を持つ関数を無闇に増やすことも避けられる.

(4) 日本語機能の組込みによってREWMOL処理系の処理効率(メモリ効率,実行速度)が低下しても,ここでは無視する.

=> 処理効率を下げずに新しい機能を追加しようとすれば,処理系の中のデータ構造や処理の方式などまでかなりの手直しが必要である.又,長さの違う文字コードをREWMOL処理系で一様に扱おうとすれば,全部の文字コードを2バイトに正規化するなどの措置が必要となる.

しかし,そうした改良には相当の手間がかかるので,将来の検討課題としたい.ここでは,日本語機能をREWMOL上に実現することがまず目標である.処理効率の低下は問題にしない.

(5) 処理系の中で使う文字コードとしては,漢字用にDEC漢字コード[7],英数字・特殊記号・制御コード用にASCIIコード,を用いる.

=> DEC漢字コードは,JIS漢字コードに8080(16進数)を加えたものである.但しこのコードは,8ビットのJISカタカナコードとは一部で衝突するので,混ぜては使えない.

その代わり,ASCIIコードと漢字コードとの境界には目印にシフトコードなどを入れなくても,先頭バイトの最上位ビットに1があれば,無条件にそれに続く1バイトと合わせた2バイトを漢字コード,その他は1バイトごとをASCIIコード,として扱うことができる.

4. 日本語機能の実現方法

ここでは,前章の方針に基づいて試みた日本語機能組込みの具体的方法を,日本語の入出力処理と文字列操作に分けて述べる.

4.1 日本語の入出力処理

日本語処理を特に意図していなかったREWMOL処理系に日本語の入出力機能を組込もうとしたとき,次の2点が問題となった.1つは入力時のローマ字・かな/漢字変換,及び入力バッファの作成とその管理,に関する機能追加の問題,もう1つはREWMOLの入出力関数の実行時ルーチンで文字コードを処理する仕方の変更の問題である.

まず漢字変換入力については,日本語Micro VMSの汎用ライブラリの漢字変換入力ルーチンJLB$GET_INPUTを利用する.一方,入出力関数の実行時ルーチンの機能変更については,DEC漢字コードがASCIIコードと境を接しシフトコードなどを介さずに混在することを考慮して,処理系の部分的手直しをすることで対処した.これらの詳細は以下に述べる.

4.1.1 漢字変換入力と入力バッファの管理

(1) 漢字変換を伴う入力

ASCII文字を扱うだけであれば,REWMOL処理系の従来の方法,つまりOSに既存の入力バッファを使い,必要に応じてCのgetc( )関数で1文字ずつ読込むことで問題はない.しかし,日本語を扱おうとすれば,ローマ字・かな/漢字変換に加えて,同音異字の選択,変換対象の変更といった煩雑な処理も行う必要がある.

日本語Micro VMSには,日本語入力に付随するこうした処理を支援するルーチン群が,基本/汎用ライブラリに相当の種類揃えてある.これらをうまく使えば,必要な機能は比較的簡便に実現できる.但しその場合,漢字端末の制御等に関する知識もかなり要求され,誰でも気軽に使いこなすというわけにはいかない.

ところがこの汎用ライブラリの中には,漢字変換入力ルーチンとしてJLB$GET_INPUT[6]も入っている.これは,ローマ字/かなでキー入力される文字列(複文節)を順次漢字かな混じり文に変換し,その文字列と長さ(バイト長)を指定領域に返す,という一連の仕事を1つにまとめたものである.漢字種の選択と確定,変換対象の変更などは補助キーパッドキーの操作

で簡単にできる．なお，このルーチンで取扱う入力の単位は，リターンキーが押されるまでの1レコード分である．

ところでこのルーチンは，プロンプタ指定，入力したデータと文字数の受取り，実行終了状態値の参照，などを独自の方式で行うので，エンドユーザが常時使うのは少々煩わしい．そこで我々は，1レコード分の日本語文字列とその字数を指定領域(前者は文字配列，後者は数字変数)に返す汎用ユーザ関数yrkstr( )を定義し，JLB$GET_INPUTはそこから間接的に呼ぶようにした．この方がユーザにとっては使い易いと考えたからである．

なお，このルーチンはASCIIコードだけの読込みもむろん正しく行う．

(2) 入力バッファの管理

ところで，JLB$GET_INPUTを使って外部データを読込む場合は，REWMOLの入力関数でデータを読む前に，一定のタイミングで，まとまった（例えば1レコード分）文字列を処理系の中のどこかに一括して取込んでおく必要がある．そのために，REWMOL側で入力バッファを設け，それを管理する必要も出てくる．我々は次の要領でこれを実現した．

まず入力バッファは次のような構造体として定義する．これは3つのメンバinput，len，flgで構成し，それぞれ文字列<最大IBUF_SIZEバイト(現在，200)>，有効文字長，バッファの使用状態，を格納する．なお，この入力バッファは，取扱いの便宜から，ローカル領域ではなく，グローバル領域に置く．但しこれは，情報隠蔽の立場から言えば，決して好ましくない．なお，この入力バッファは最大IBUF_LMT（現在，20）個を同時にオープンすることができる．

入力バッファには，それぞれ固有の識別番号(0～IBUF_LMT-1)を付けておく．そして入力ファイルが1つオープンされるたびに，未使用の入力バッファ1つを探し，その入力バッファのflgに，使用中を示す1を書込んでおく(処理はCの関数ysskbf( )で行う，以下関数名だけを示す)．併せて，その入力バッファの識別番号を入力ファイルのファイルセルのメンバbufnに記入し，対応関係を明示しておく．入力ファイルがクローズされると，その入力バッファも解放しておく(flgに未使用を示す0を記入，yrlkbf( ))．

ところで，REWMOL側で管理する入力バッファは，磁気ディスクなどのファイルから確定したデータを一挙に入力するだけの場合は不要である．例えば，実行状態を保存(凍結)したファイルからREWMOLの処理系を立上げるとき，この入力バッファを経由させる必要はない．しかし，REWMOL関数FLOOKCで先読みして検査しながら漢字を入力するような場合には不可欠である．そのときは，たとえ磁気ディスクファイルからの入力であっても，入力バッファを経由させる必要がある．

それは次の理由による．REWMOL関数FLOOKCの仕様によれば，この関数は入力中に1文字読んだ後，直ちにその1文字，すなわち2バイトの文字コード，を入力バッファに返しておく必要がある．ところがOSの標準機能としては，1バイトのバッファリングしか保証していない．従って，REWMOL側が管理する入力バッファを使わざるをえない．

なお，入力バッファのflgは，処理系を立上げるときに全部0に初期化しておく(yiskbf( ))．又，入力バッファは使ってもローマ字・かな／漢字変換ルーチンを使う必要がなければ，入力ファイルのファイルセルのbufnには－1を記入しておく．

4.1.2 REWMOLの入力関数[9]

REWMOLの基本入力関数はFGETC/FGETA/FGETX／FLOOKCの4つである．これらはそれぞれ，文字／アトム／S式／文字を入力バッファから読出してその値を返す．

この機能を実現するための補助関数として，処理系の原版ではCの関数jrgetc( )を定義した．この関数は，起動されるたびに1バイトのデータを，OS標準の入力バッファからCの関数getc( )を使って読取る．但し，入力バッファが空のときに入力データを読取ろうとすると，jrgetc( )はCの関数jrslno( )に制御を移し，レコードの先頭の数字列と空白だけは読飛ばして，有意の1バイトをデータとして読取る．

原版のjrgetc( )が行う上述の処理内容を，ここでは以下のように改める．まず，入力操作が現在漢字端末から行われていることをjrslno( )の中で確認すると，ユーザ定義の入力バッファY_KBUFinput[kbfn]の中へ1レコード分のデータを，Cの関数yrkstr( )を使い，一括して取込むように変更する．ここで，kbfnは入力バッファの識別番号である．又，Y_KBUF.input[kbfn]は識別番号kbfnを持つ入力バッファであって，それ自身1次元の文字型配列である．なおこの入力バッファには，jrslno( )の中で，yrkstr( )を使って取込んだ原データを逆順に詰めておく．

次に，jrgetc( )では，従来OS標準の入力バッファからgetc( )を使って読込んでいたデータを，ユーザ定義の入力バッファから読込むように変更した．そのため

の補助関数としてygetc( )を定義した．更に，Cの標準関数ungetc( )に対応する関数としてyungetc( )も定義した．これはユーザ定義の入力バッファへ1バイト分のデータを戻す．

REWMOLの入力関数FGETAとFGETXは，上記の外はまったく変更せずに，前者がアトムを，後者がS式を，漢字ASCII文字混じりで入力できるようになった．しかし，FGETCとFLOOKCについては，更に以下の手直しが必要であった．元来FGETCとFLOOKCでは1バイトの文字コードの入力だけを想定していたからである．

まず，両関数とも，最初に読込んだ1バイトが漢字コードの一部か否か調べる．そうであれば次の1バイトも続けて読込み，両者を合わせて1つの漢字コード（10進数）とし，それを関数値として返す．さもなければ，最初の1バイトだけで1つの文字コードとし，それを関数値として返す．

FLOOKCの場合は更に，一度読んだ文字コードを入力バッファへ戻して，再入力に備える必要がある．そこで，読込んだ文字がASCIIコードか漢字コードかに応じて正しく戻すように改修した．

上記以外の入力関数(READ, READ-SENT, など)は，いずれも改修済みないし改修の不要なREWMOLの基本関数を使って定義してあるので，直接には手を加えることなく，正しく動くようになった．

4.1.3 REWMOLの出力関数[8)]

REWMOLの基本出力関数はFPUTC/FPUTXの2つである．このうちFPUTCは，1文字分のASCIIコードを10進数で引数として受取り，対応する文字パターンを出力する．この改修の要点は，漢字の場合に2バイトのコードを1つの10進数(例えば，'漢'=b4clならば46273)で受取り，処理系の中の実行ルーチンによって2バイトに復元し出力ルーチンに出すようにしたことである．なお，ASCIIコードの扱いは従来と変わらない．

FPUTXは，元来S式を引数として受取り，そのまま出力する関数である．このため，漢字とASCII文字が混在するS式を受取る場合も，まったく手を加えることなく正しい処理ができている．

上記以外の多くの出力関数(PRINT, PP, pp, MSG, MSG_CCG, PRIN-SENT, PRIN-TEXT, など)[8,9)]は，いずれも他の多くの基本関数（実行時ルーチンがCが書かれ，REWMOL本体に組込まれた関数)やブートファイル関数（REWMOL関数自体で定義された関数）を使って定義してある．これらの場合は，基本的な出力関数と文字列操作関数を上述のように改修しただけで，直接の改修はせず，いずれも漢字ASCII文字混じりの文字データを正しく扱えるようになった．

4.2 文字列操作

文字列操作を行うREWMOLの基本関数には，STRLEN, SUBSTR, NTHCHAR, STRTOCHAR, CHARTOSTRの5つがある．これらについては，引数として渡される文字数と文字位置が，漢字かASCII文字かを問わず，外部表現の字数で指定できるようにすることが，使い易さの点から要求される．

そこでまず，文字数（長）／文字位置を外部表現の数値から内部表現の数値に変換する関数（ykstrlen, ycvstrlen_ei)，及び2バイトの漢字コードを1つの10進数にまとめる関数（ycvknjcd_ps）を，次の仕様のように定義した．

1) 漢字ASCII文字混じりの文字列strを引数として受取り，その外部表現の字数を関数値として返す．
```
int ykstrlen (str)
char *str;
```
2) 漢字ASCII文字混じりの文字列strと，文頭を0とした文字位置を外部表現の数nで受取り，それを内部表現の数に変換し，関数値として返す．
```
int ycvstrlen_ei (str, n)
char *str;
int n;
```
3) 2バイトの漢字コードcd1とcd2を別々の引数として受取り，(cd1 * 256)+cd2の計算をして，その結果を関数値として返す．
```
int ycvknjcd_ps (cd1, cd2)
int cd1, cd2;
```

次に，これらを補助関数として使い，文字列操作のためのREWMOL基本関数の実行ルーチンを漢字ASCII文字混じりデータ対応に改修した．

この結果，REWMOLの入出力関数で漢字コードを入出力するとき，及び文字列操作関数で漢字コードを外部へ出力するときは，5桁の正整数（10進数）を使うことになる．その場合でも，内部表現はむろん従来どおり2バイトのままである．

5．検　討

上に述べた方針に基づいてREWMOLの処理系V4.0rに対する改修作業を実施した．そして，機能拡張を行ったREWMOLの各基本関数が単独でも他の関数との組合せでも仕様どおりに動作するかどうかを，種々の動

作実験によって調べた．その結果，所期の機能が実現されていることが確認できた．この処理系をMicro VMS版日本語REWMOL（V4.0r）と名付ける．

ここでは，今回の改修に関連する二三の事項について若干の検討を試み，他の類似システムにおける日本語化の例と比較する．なお，それに先立って，今回の作業量の目安を示すために，処理系のソースプログラムの規模と，改修の途中で加除変更した行数等を簡単に示しておく．

5.1 改修の規模

REWMOL処理系を構成する関数群は，機能ないし使用目的で分類して，約130個のファイルに分散格納している．又そのソースプログラムの総行数は4,000行を超えている．

このうち，このたびの改修作業の対象となった既存ファイル数は11個，新しく追加したファイル数は3個であった．変更ないし追加したソースプログラムは，行数で言えば，既存ファイル分で約130行，追加ファイル分で約170行であった．

5.2 機能評価

ここでは，改修に伴う動作実験などで明らかになった，日本語機能にかかわる関数の性能やプログラミング言語としての使い勝手などを，項目別にまとめる．

(1)ローマ字・かな／漢字変換

ローマ字・かな／漢字変換機能の実装方式の主流は，現在，ドライバ組込み型，デーモン介在型，ライブラリ組込み型の3つである[10]．この分類に従えば，我々の方式はライブラリ組込み型に属する．

ライブラリ組込み型の場合，ユーザは，アプリケーションごとにライブラリの中から必要な機能を選んで組込む必要がある．その場合，ハードウェア，ソフトウェアの知識もかなり要求される．反面，ライブラリが充実していれば，目的に合った任意の機能を実現できる可能性もある．ただ，今回のように一度完成したプログラムに組込むことによって機能拡張するのは，それほど簡単でない．

そこで今回は，変換処理に必要な一連の仕事をまとめたライブラリルーチンJLB$GET_INPUTを利用した．これで当面の目標をひとまず達成することはできた．しかし，幾つかの点で不満が残る．

その最たるものは，入力データの長さが端末画面の1行を超えるとき，カーソルは改行されず，画面の右端に留まることである．それでも，バッファの長さが十分であれば，データ自体は失われない．ただ，1行を超えた分については入力文字を目で確認することができず，従って，漢字の種類が選べない．そのため，長い文を1レコードで入力するのは難しい．

更に，文字列の中から変換の対象を選ぶ／切換える操作が“日本語ワープロ”ほどにはうまくできない．そのため，長いローマ字列を一気にキー入力して連文節変換をすることは難しい．

こうした弱点を知った上でうまく使うならば，現在の機能でも種々の実験用に十分耐えうる，と考える．

(2)REWMOL関数による入出力および文字列操作

今度改修したREWMOLの基本関数，及びブートファイル関数は，いまのところいずれも異常動作などの問題を発生してない．

文字列操作関数については，漢字ASCII文字混じり文をSTRTOCHARに与え，その結果を続けてCHAR-TOSTRに渡した場合，最初の関数に与えた引数と同じ文字列を結果として返す．PACKとUNPACKについても同様である．更に，STRLEN，NTHCHAR，SUB-STRの場合も，漢字ASCII混じり又は単一いずれの文字列に対しても，正しく文字列操作をすることを確認した．

入出力関数については，4.1.2と4.1.3で既に言及したので，ここでは省略する．

(3)移植性の問題

3．の(1)に掲げた‘高い移植性’という方針にもかかわらず，当面我々が実現した処理系は，日本語Micro VMSの環境で動く日本語端末に適応した版だけである．

しかし，Cのプリプロセッサの条件付きコンパイル，及びMicro VMSのライブラリルーチンによる端末機種の自動検出の機能を組合わせると，それぞれの環境に適応した機能だけをうまく組込んだ処理系を立上げることが極めて簡単にできる．すなわち，条件付きコンパイルの機能によって，VMS環境と非VMS環境を識別できる．又，端末機種の自動検出の機能では，VAX環境の下で漢字端末環境とASCII端末環境を識別できる．

この点については，テストプログラムによる動作実験を既に完了している．従って，移植性の問題は一応解決済みと考えている．

5.3 他との比較

SNOBOL，LISPのような記号処理用言語に日本語機能をあとから組込む問題は，こうした言語の処理系が海外から持込まれることも少なくないなどの理由で，

最近でも関心を呼ぶ話題である[11,12,13]．このような場合，作業の眼目は，処理系の効率確保よりも，ユーザに文字コードの内部表現の違いを意識させず，いかに日本語を扱えるようにするかという点である．従って，漢字処理のために特別の関数を設けるのでなく，既存の関数の機能を拡張することによって目的を達成することが当面の目標となる．

異なる文字種を統合した文字列の実現，そのメモリ効率，文字列の処理効率，といった問題は，処理系を最初から設計し作成するようなときに考慮して，初めて現実的な意味を持つ[14]．

我々の今回の試みは，日本語機能の組込みを処理系作成の当初はまったく意図していなかったので，結果的には一度完成したものにあとから追加する，という立場を採ることになった．従って，改修の基本方針などは，Franz Lispの日本語化で採られたもの[13]とほとんど変わらない．

ところで，同じVMSの環境で動くLISP処理系を日本語化した例としてVAX LISPがある[12]．この改修については詳細が不明ではあるが，文献12)から判断する限り，日本語入力にライブラリルーチンJLB$GET_INPUTを使う点は同じである．ただVAX LISPの場合，シンボル生成ルーチンを漢字用に改修している．この点で我々の場合は，文字コードとして8ビットコードを受入れるように変更した以外は，まったく手を入れていない．なお，文字列処理に関してどのような改修をしたかについては，記述がないので分からない．

6．むすび

Micro VMS版日本語REWMOL (V4.0r) を実現したことで，REWMOL関数のユーザは，漢字を，ASCII文字とまったく同様に，文字コードの内部表現を全然意識せず，外部表現のイメージだけで取扱えるようになった．

REWMOLは，並列処理の機能の外に，オブジェクト指向プログラミングの機能もある程度備えた言語である[1,15]．こうした環境で日本語が使えることは，プログラミング言語としてのREWMOLの有用性をさらに高めるものである．

今後の課題としては，一つにREWMOLにおける日本語機能の効率向上がある．しかし，我々はそれ以上に，現在のREWMOLを，マンマシンインターフェースの諸問題，自然語理解，自然語生成等のプログラミングで活用していくことにまず力点を置きたい．その過程で必要ならば，言語の改良も更に続けていくことにする．

参考文献

1) 刈谷丈治，石原好宏："実世界記述言語REWMOLの開発"，信学論(D)，J68-D，2，114-121 (Feb. 1985)

2) 石原好宏，刈谷丈治："REWMOLのVAX/VMSへの移植と自然語処理のための機能拡張"，山口大学工学部研究報告，37，2，297-303 (Mar. 1987)

3) 石田晴久："UNIXの日本語機能と日本語対応Cコンパイラ"，最新UNIX (bit臨時増刊，19，6) 132-138，共立出版 (May. 1987)

4) Kernighan, B.W. and Ritchie, D.M.："The C programming language", Prentice-Hall (1978) (邦訳，石田晴久 (訳)："プログラミング言語C"，共立出版 (1981))

5) 日本DEC㈱："日本語Micro VMS V4.4リリースノート" (Aug. 1986)

6) 日本DEC㈱："日本語Micro VMS日本語VAX/VMS日本語ライブラリ利用者の手引き" (Mar. 1986)

7) 日本DEC㈱："漢字コード表" (1984)

8) 刈谷丈治："REWMOL (V4.0r) 取扱説明書"，山口大学情報処理センター，p.52，(Apr. 1987)

9) 石原好宏："Micro VMS版日本語REWMOL (V4.0r) 取扱説明書"，山口大学工業短期大学部情報処理工学科，(Oct. 1987)

10) 鈴木幸市・今福幸治："日本語処理—コードとワープロソフト"，最新UNIX，bit (5月号臨時増刊)，19，6，873-883 (May. 1987)

11) 牛島和夫，吉田和幸："日本語テキスト処理機能を追加したSNOBOL4"，コンピュータソフトウェア，1，1，84-86 (Apr. 1984)

12) 前端克典，川合　進："日本DECのAI製品における日本語化について"，情報処理学会研究報告，86-SYM-36-5，(Mar. 10 1986)

13) 杉田研治："日本語用Franz Lisp"，最新UNIX，bit (5月号臨時増刊)，19，6，764-772 (May 1987)

14) 杉村利明，奥乃　博，竹内郁雄："TAOにおける日本語文字列処理"，情報処理学会研究報告，86-SYM-36-4 (Mar. 1986)

15) 刈谷丈治："能動的オブジェクトを用いたモデル構築による問題解決"，山口大学工学部研究報告，37，1，95-100 (Oct. 1986)

(昭和62年10月15日受理)