ローマ字入力方式のモードに関する検討

大野　邦夫
（ＮＴＴ横須賀電気通信研究所）

１．まえがき

ローマ字かな漢字変換による日本語入力方式は，ASCIIキーボードに慣れた一般のプログラマにとっては使い易い日本語入力方式である．[Takenaka81] ところで，現在ワードプロセッサやパーソナルコンピュータで用いられているローマ字かな漢字変換方式の多くは，先にひらがな，カタカナ，英数，等のモードを指定してローマ字で入力する方式，すなわちPREFIX方式を用いている．ローマ字入力でない，かなキーボードによるかな漢字変換の場合は，これが自然な方法と考えられるが，ローマ字かな漢字変換方式の場合は必ずしも最適な方式とは言えない．　また，一定量の文書を，先にローマ字で一括入力し，後に漢字，ひらがな，カタカナ，英数，等を指定する方式も考えられる．（ここではバッチ入力変換方式と呼ぶことにする．）　ここでは，PREFIX方式とPOSTFIX方式およびバッチ入力変換方式についての操作性の比較を行なった結果について報告する．

２．基本的考え方

2.1 PREFIXとPOSTFIX

検討に先立ってPREFIX方式とPOSTFIX方式について若干の説明を加えておく．PREFIXとは，先ず行なおうとする操作の指示があり，その後にその操作の対象が続く．POSTFIXはその逆で，先ず対象となる要素が示されて，その後に操作が続く．[Meyrowitz82]

一般に，POSTFIXの方が操作時のモードを少なくすることができ，[Smith82] モードを少なくすることがユーザインタフェースの向上につながると考えられる．[Tesler81] また，計算機との会話において，システムの内部状態に大きな変化を及ぼすのは，操作対象ではなく操作指示である．従って操作誤りに対する取り消し機能(UNDO)は，POSTFIXの場合の方が実現が容易である．[Stelovsky84]

2.2 必要とされる変換機能

ローマ字かな漢字変換方式により作成するドキュメントの文字コードは，本来ここで問題にするユーザインタフェース等とは無関係なのであるが，装置へのインプリメント上考慮する必要がある．一般の日本語文書の文字コードは，JIS-C6226で定められた２バイトのコード，またはシフトJISや，最近定められたUNIXの日本語コード等，JIS-C6226をベースとして部分的に改良を加えたものである．一方，日本語文書以外に，日本語を含むコマンドやソースプログラム等の計算機環境の場合を考えると，コマンドやプログラム言語は通常はASCIIやJIS-C6220の１バイトコードの英数字であり，その中に変数名やコメント等の形で２バイトの日本語コードが入ることになる．従って，ローマ字かな漢字変換方式により作成されるドキュメントやコマンド列等の文字コードは，１バイト系と２バイト系の混在したものとなるのが一般的である．

一方，表示されたり，印刷されたりした文字は，２バイト系は全角で，１バイト系は半角で示すのが合理的である．キーボードからの英数字入力は１バイト系なので，ローマ字入力時に英数字モードであれば半角で表示されることになる．（２バイト系でも英数字は含まれているが１バイトの英数字を優先することになる．）

以上から，ローマ字かな漢字変換入力時には，基本的には以下の４つの変換機能が必要と考えられる．

(1)かな漢字変換（熟語変換）
(2)ひらがな変換
(3)カタカナ変換
(4)英数変換

このほか，半角のカタカナモード，先に述

べた全角の英数モード，横倍角モード等があるがこれらのモードについては検討の対象としない．必要であれば，別の形でサポートすればよい．

2.3 PREFIX方式

PREFIX方式でローマ字かな漢字変換を行なう操作を図１に示す．この方式では，先に英数モードか，かなモードかを英数シフトキー等によって決めておき，英数モードの場合は，キーボード入力文字列がそのままテキストバッファに送られる．かなモードの場合はキーボード入力文字列がローマ字かな漢字変換テーブルによってかな文字列に変換される．この時にカタカナモードであれば，変換テーブルによって得られたカタカナの文字列がテキストバッファに送られる．ひらがなモードであれば，かな漢字変換処理が行なわれ，文節等の区切りにおいて無変換キーが押されれば，ひらがながテキストバッファに送られ，変換キーが押されればかな漢字変換された文字列がテキストバッファに送られる．（なお，変換・無変換についてはPREFIXでなくPOSTFIXとなっている．同音語の選択だけはPOSTFIXでなければできないからである．）

2.4 POSTFIX方式

POSTFIX方式の場合は，図２に示すように，キーボード入力のASCII文字列は２つのフローとなり，一方は，英数字バッファにそのまま蓄積され，他方はローマ字かな変換テーブルを経てかなに変換される．かなは更に３つのフローとなり，ひらがなバッファ，カタカナバッファ，およびかな漢字変換処理を経た熟語バッファに送られる．操作者は，英数字，カナ，かな，変換の４つのキーのうちの一つを押下し，入力された文字列を選択しテキストバッファに送る．このように，POSTFIX方式だと，英数，かな，カナ等のモード設定せずに入力することができる．

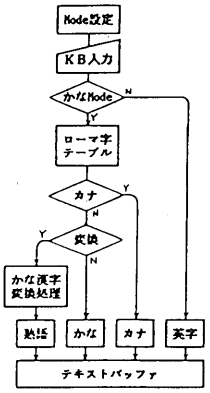

図 1. PREFIXの場合の操作手順

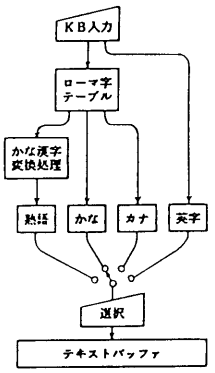

図2. POSTFIXの場合の操作手順

2.5 バッチ入力変換方式

この場合は，よりマクロな視点からのPOSTFIXで，入力の操作を大きく２つのフェイズに分け，最初のフェーズで，一定量の文章をローマ字で一括入力し，後のフェーズで再び文頭にカーソルを戻して１語づつ変換してゆく方式である．そのため，前項で述べたPREFIX，POSTFIX両方式の場合に較べ，ローマ字入力時にモードを設定したり変換対象を意識したりせず，単に変換単位の語に区切るだけで良い．そのため入力時の心理的な負担は小さくなると考えられる．また，この方法の付随的メリットとして，変換辞書に対する応答がリアルタイムでなくなる点が挙げられる．ローマ字入力中に先に入力された語の変換候補文字を辞書から主記憶上のバッファに抽出しておき，後に一括して変換するフェーズでバッファ上の候補文字を選択するようにできるからである．

3 実験システム

実験システムとしては， Symbolics-3600 Lispマシン[Weinreb81]を用いた．本研究目的のためには，かな漢字変換機能が必要なため，熟語単位の変換機能をSymbolics-3600の入力エディタ（Lispトップレベルで使用する）およびテキストエディタ，ZMACS[Wechsler81]上に構築した．このかな漢字変換機能はローマ字入力方式により実現され PREFIX，POSTFIX双方の方式で変換可能とした．またZMACSエデ

ィタの２ウインドウモードを使用して，バッチ入力変換方式も可能とした．

3.1 PREFIX方式

Symbolics-3600では，MODE-LOCKキーおよびCAPS-LOCKキーを用い，表１に示すような方式のローマ字入力機能を有している．
この機能を用いPREFIX方式のローマ字かな漢字変換機能を実現した．変換キーのためには，hyper-space（hyperキーとspaceキーの並押下），無変換キーのためには，hyper-control-space（hyperキー，controlキー，spaceキーの並押下）で対処した．従って，状態遷移は図３のようになる．変換処理において，同音語の候補文字がある場合は，ポップアップメニュで表示し，マウスで選択できるようにした．なお，参考のためにSymbolics-3600のキーボードレイアウトを図４に示す．

表１　Symbolics-3600 のローマ字入力機能

	MODE-LOCK OFF	MODE-LOCK ON
CAPS-LOCK OFF	英数モード（小文字）	ひらがなモード
CAPS-LOCK ON	英数モード（大文字）	カタカナモード

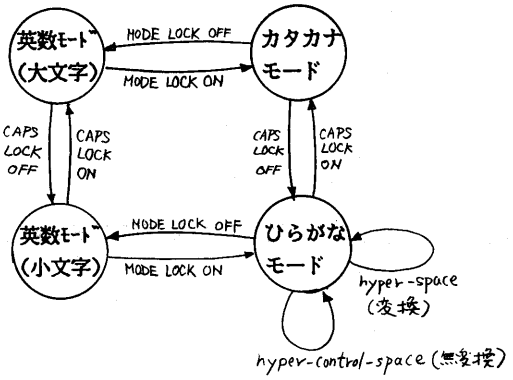

図３　PREFIX方式における状態遷移図

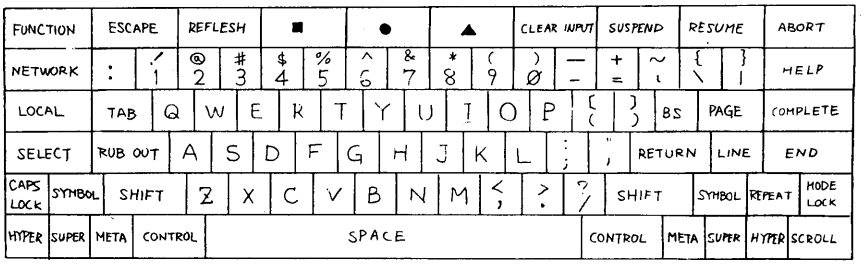

図４　Symbolics-3600のキーボードレイアウト

3.2 POSTFIX方式

POSTFIX方式の場合のローマ字かな漢字変換法のキー操作を以下に示す．

熟語変換：　HYPER-SPACE
ひらがな変換：　HYPER-CONTROL-SPACE
カタカナ変換：　HYPER-META-SPACE
英数変換：　CONTROL-SPACE

変換される文字列は，現在のポインターの前の単語，又はマークされた文字列である．熟語変換処理における同音語の選択は，やはり前項と同様，マウスによるポップアップメニュー選択とした．

リスプトップレベルにおける使用例を図５に示す．ここでは階乗という関数を定義し，この関数を用いて階乗計算を行なっている．図５の画面は，20の階乗を計算した後，さらに計算を行なうために，(kaijyouと入力し，HYPER-SPACEを押下し，ポップアップメニューを表示させたところである．

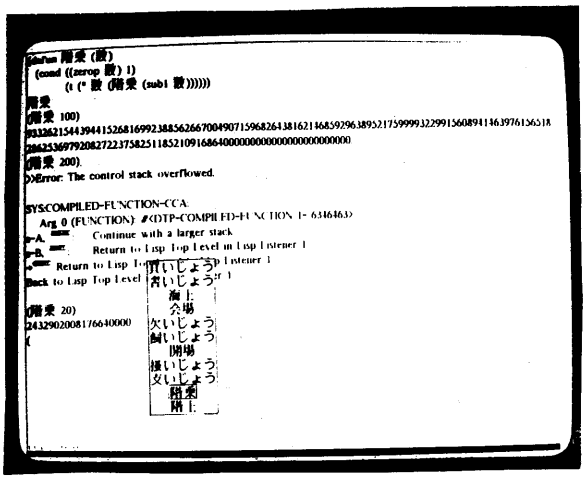

図5　POSTFIX方式によるローマ字入力画面例

3.3　バッチ入力変換方式

3.3.1 キーボードのキーで変換する場合

ZMACSエディタの２ウインドウモードにおいて，上のウインドウに１語ずつスペースで区切ってローマ字で一括入力した後，カーソルを先頭に移動させ，１語ずつ変換し，変換結果を下のウインドウに表示させるようにした．なお日本語文においてスペースが必要な場合はスペースを連続して入力する．変換のためのキー操作は以下の通りである．

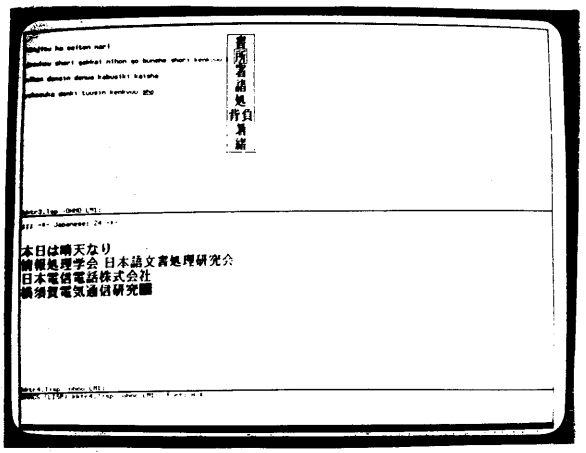

図6　バッチ変換入力方式による画面例

熟語変換：　SUPER-CONTROL-META-SPACE
ひらがな変換：　SUPER-CONTROL-SPACE
カタカナ変換：　SUPER-META-SPACE
英数変換：　SUPER-SPACE

変換操作を行なうと，カーソルが次の変換単位の語に進むように設計したため，最初に先頭に移動させる場合以外はカーソル操作は不要である．

図６にこの場合の画面例を示す．上の画面はローマ字入力画面で，入力文字列の最後の sho にアンダラインが引かれ，この部分が変換対象になっている．ここでは，この部分を漢字に変換するために SUPER-META-CONTROL-SPACEが押下されポップアップメニューが表示されている．下の画面は，日本語の出力画面で，すでに変換された文字列が表示されている．

3.3.2 マウスのキーで変換する場合

この変換方式において，試作した５個のキーを有するマウスを用いて操作することを試みた．このマウスの外観を図７に示す．このマウスの作成意図・機能等について，およびZMACSへのインプリメントについては，他の文献を参照されたい．[Ohno84],[Ohno85] ここでは，上で述べた変換のためのキー操作を，以下の通り親指キーシフト時のマウスのキーに割り付けた．

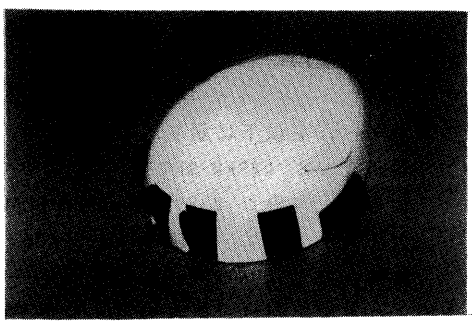

図7　バッチ入力変換方式で使用したマウスの外観

熟語変換：　　親指キー
ひらがな変換：　中指キー
カタカナ変換：　薬指キー
英数変換：　　小指キー

なお人差指キーはメニュー選択のために使用する．

4．操作実験

4.1 実験方法

操作実験のために以下の文例を用いた．

EX-1)エキスパートシステムの開発を効率化するAIツールの商用化が進んでいる．
EX-2)ローマ字かな漢字変換方法にはPREFIX方式とPOSTFIX方式とがある．
EX-3)Xerox社のAltoコンピュータのマウスのボタンは3個付いていた．
EX-4)INSモデル実験システム
EX-5)AI用ワークステーション
EX-6)本日は晴天なり

EX-7)

```
(defun 階乗　(数)
  (cond ((zerop 数) 1)
        (t (* 数 (階乗 (sub1 数))))))
```

以上の７つの文例について，操作実験を行なった．実験は，筆者により，各例文について，PREFIX，POSTFIX各々について５回行ない，操作時間をストップウォッチで測定した．

4.2 キーストロークレベルモデル

操作時間を予測するために，Card 等が提案したキーストロークレベルモデル[Card80]を用いることを試みた．もしこの方法が有効であれば，操作実験を行なわなくとも，操作手順を明らかにすることにより，操作時間の予測が可能となる．

このモデルの場合，マンマシンインタラクション操作を，以下の６つの場合にカテゴリ分類する．

(1) K ・・・・・・ Keystroking
(2) P ・・・・・・ Pointing
(3) H ・・・・・・ Homing
(4) D ・・・・・・ Drawing
(5) M ・・・・・・ Mental Operator
(6) R ・・・・・・ System Response

キーストロークレベルモデルは，ユニットタスクにおける実行時間を問題とする．実行時間をTeとすると，

$$Te = Tk + Tp + Th + Td + Tm + Tr$$

ここで，Tk，Tp，Th，Td，Tm，Trは，各々，Keystroking，Pointing，Homing，Drawing，Mental operation，System response に要する時間である．ユーザがエキスパートであるとすると

Tm＝1.35sec
Tk＝0.2sec
Th＝0.4sec
Tp＝1.1sec

となり，この値を用いて操作時間を積算することが出来る．なおTdは本検討においては不要であり，Trは実測する必要がある．なおこのモデルにおいては，Tm，Th，Tpの値については，以上の数値を用いたが，Tkについては，実測に基づき，0.3secとした．Mオペレータは Card等の設定したガイドラインに沿って，操作の区切り毎に入れるようにした．従って，テキスト入力の前，変換キー押下の前，マウスで選択する前等には必ず挿入されている．しかし，マウスでメニュー項目を指示した後のボタン操作のように，無意識に機械的に行なえる場合は省いている．より具体的な適用方法については別の文献で述べているので参照していただきたい．[Ohno85]

4.3 実験結果

実験結果を表２に示す．またこの実験結果とキーストロークレベルモデルによる計算値との比較を図８に示す．図からわかる通り，実験値と計算値との一致はかなり良好であり，キーストロークレベルモデルを用いて操作時間の評価が可能なことを意味していると考え

表2　実験結果(PREFIXとPOSTFIXとの比較)

	MAX	MIN	MEAN	K.S.L.Model
EX-1 PREFIX	137.0	126.0	132.8	126.0
POSTFIX	130.0	115.0	123.4	121.0
EX-2 PREFIX	123.0	90.0	102.8	97.7
POSTFIX	82.0	70.0	76.0	93.6
EX-3 PREFIX	67.4	63.5	65.9	67.8
POSTFIX	62.8	54.2	59.0	59.5
EX-4 PREFIX	29.5	23.1	25.8	28.2
POSTFIX	24.4	21.0	22.6	24.1
EX-5 PREFIX	23.3	21.6	22.3	23.35
POSTFIX	18.4	15.9	17.5	17.0
EX-6 PREFIX	29.0	25.7	27.6	29.6
POSTFIX	28.6	24.3	26.0	29.6
EX-7 PREFIX	132.0	99.0	111.0	98.25
POSTFIX	82.0	66.0	74.0	75.0

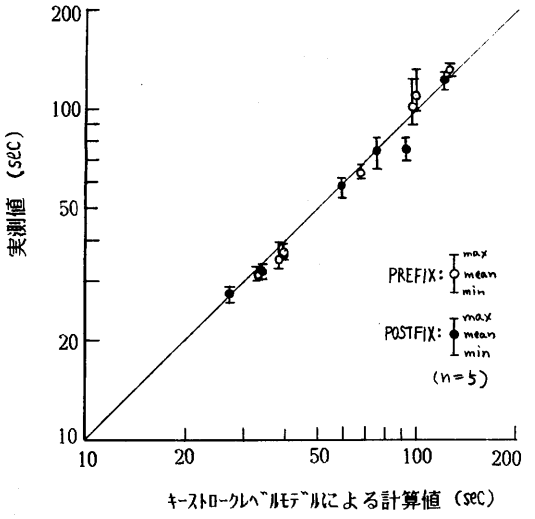

図8　実験結果(PREFIXとPOSTFIXとの比較)

られる．

表2の結果から，POSTFIX方式の方がPREFIX方式よりも操作時間が短いことが判る．これは実験結果全ての場合について言えており，かつEX-6以外はキーストロークレベルモデルによる計算でも裏付けられている．

EX-6の場合は，キーストロークレベルモデルにおいてはPREFIX方式とPOSTFIX方式との操作時間が一致しており，かつ実験結果の方も大きな差は見られない．この理由は，変換対象が熟語とひらがなであるため，PREFIX方式の場合でもモード切り替えが不要であったためである．それにひきかえ，英数字やカタカナが混在する場合はPOSTFIX方式が有効なことが明らかである．特に，EX-7のようにLISPのソースプログラムではその差が顕著である．

表3および図9にバッチ入力変換方式についての実験結果をとキーストロークレベルモデルとの比較を示す．表においてKBは変換操作にキーボード上のキーを，MOUSEは試作したマウスのキーを使用する場合の処理時間を示している．この結果より，マウスのキーによって変換操作を行なう方が操作が早いことがわかる．

一方，表2の結果と較べると，キーボードによって変換操作を行なう限り，バッチでない場合のPOSTFIX方式に較べ，実測値，モデル

表3　実験結果（ﾊﾞｯﾁ変換方式）

	MAX	MIN	MEAN	K.S.L.Model
EX-3 KB	75.5	66.7	71.1	67.75
MOUSE	66.5	56.0	60.1	55.75
EX-5 KB	22.3	18.9	20.5	19.35
MOUSE	17.5	16.4	16.8	16.4
EX-6 KB	32.8	29.4	30.7	32.85
MOUSE	27.6	24.7	25.9	28.0

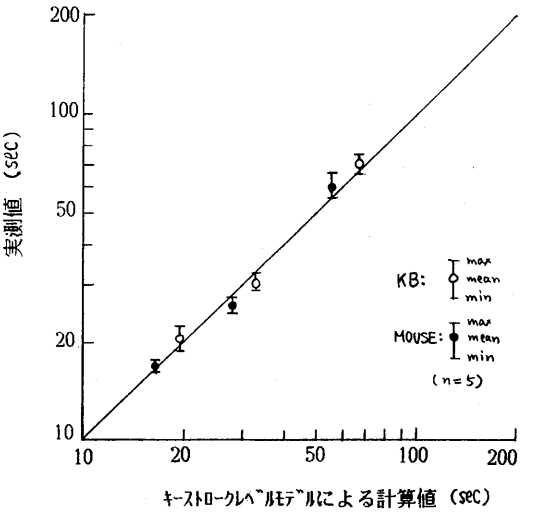

図9　実験結果(ﾊﾞｯﾁ入力変換方式)

による計算値とも操作時間は増大している．場合によっては，PREFIX方式の場合を上回っているものもある．従って，キーボードのみを使用して変換操作を行なう場合は，バッチ方式は操作時間の短縮には結び付かないと言える．

一方，試作したマウスを用いて変換操作を行なう場合には，操作時間はかなり短縮され，POSTFIX方式の場合と同等か，或は若干上回る結果が得られている．

4.4 既存の商品によるPREFIX,POSTFIX方式の比較

以上はSymbolics-3600上の試作したシステムによる検討例である．このシステムによる検討だけでなく，さらに既存のシステムによっても評価を試みた．PREFIX方式としては日本語ワードプロセッサ,OASYS(100J形),POST-FIX方式として，パーソナルコンピュータ,PC-9800上のJS-WORD（変換辞書はハードディスクに入っている）を使用した．

OASYSの場合は，モード切り替えに”英字／英大文字”キー，”カタカナ／英小文字”キーと”無変換”キーを用いる方式になっている．状態遷移図を図10に示す．

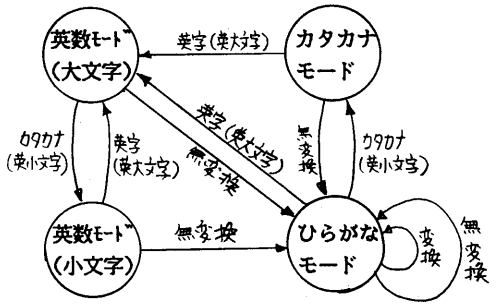

図10 OASYSの入力モードの状態遷移図

一方,JS-WORDの場合は，以下のキー操作で変換処理を行なう．

熟語変換：	SPACE
ひらがな変換：	RETURN
カタカナ変換：	F・4
英字変換：	F・5

なお，OASYS,JS-WORDとも英字の場合は，全角とした．半角にすると共に操作が複雑になるからである．実験で用いた文例は，EX-7を除きSymbolics-3600の場合と同様である．実験結果およびキーストロークレベルモデルによる計算の結果を表４に，両者の比較を図11に示す．今回は，３回の入力操作を行ない，平均値を求め，キーストロークレベルモデルで算出した計算値と比較した．

表4 実験結果(OASYSとJS-WORDとの比較)

	MAX	MIN	MEAN	K.S.L.Model
EX-1 PREFIX	59.1	56.3	57.3	51.6
POSTFIX	54.7	54.0	54.3	48.0
EX-2 PREFIX	50.8	47.2	48.8	47.7
POSTFIX	46.6	43.7	44.7	42.75
EX-3 PREFIX	52.3	49.3	50.4	48.6
POSTFIX	40.5	36.5	38.5	38.7
EX-4 PREFIX	17.7	15.3	16.9	17.55
POSTFIX	17.6	14.5	15.6	13.95
EX-5 PREFIX	15.7	14.5	14.8	14.25
POSTFIX	15.8	12.5	13.9	11.3
EX-6 PREFIX	12.7	11.7	12.2	12.9
POSTFIX	12.7	11.5	12.1	12.9

(注) OASYSはPREFIX方式，JS-WORDはPOSTFIX方式である．

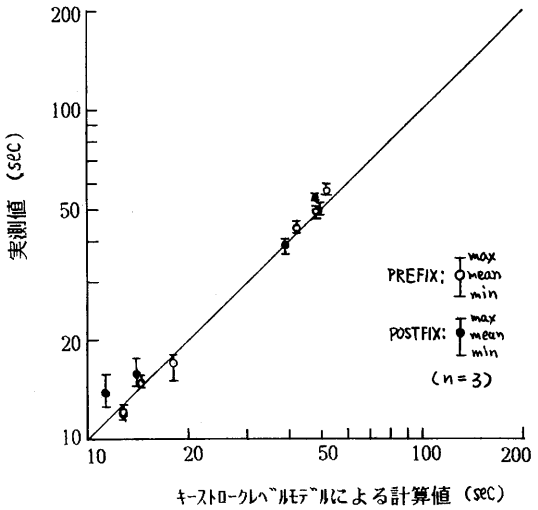

図11 実験結果(OASYSとJS-WORDとの比較)

OASYSは富士通(株)，PC-9800は日本電気(株)，JS-WORDはアスキー(株)の製品である．

表からわかる通り，EX-1からEX-5については，POSTFIX方式のJS-WORDの方が操作時間が短縮されている．EX-6については，操作時間はほぼ同等であるが，これは先に述べた通り，PREFIX方式でもモード切り替えが不要なためである．なおOASYS,JS-WORDともローマ字かな漢字変換時の応答速度は十分に早く，（共に0.5sec以下）この差は両者の差にはなっていないと考えられる．

表２の結果と比較すると，この場合の方が操作時間が圧倒的に短くなっている．その理由は，１つには，Symbolics-3600の場合辞書アクセスの応答に若干の時間を要したためであるが，表４の場合，変換時に同音語の第１候補が該当する語になるようにしたためである．

５．考察

5.1 PREFIX方式とPOSTFIX方式の比較

以上の実験結果より，POSTFIX方式の方が操作時間が短縮されることが明らかになった言える．その効果は，日本語文章中に，英数字やカタカナが混在している程顕著であると言える．

また，今回の実験では操作誤りを起こした場合は，実験を中止し，始めからやり直したが，誤操作を訂正する方法はPOSTFIX方式の方が容易であった．すなわち，PREFIX方式における操作誤りのかなりの部分をモード指定誤りが占めているが，これの訂正のためには，誤って指定したモードにおいて入力した文字をDELETEキーで全て削除し，正しいモードで再び入力し直さねばならない．それに対し，POSTFIX方式では，このようなモードが無いのでこのような誤操作は生じない．その代わり，ひらがな，カタカナ，英数字等への変換操作時に誤る可能性があるわけであるが，この場合は誤変換に気づいた後，正しい変換キーを押下し直すだけで，容易に訂正することができる．今回の検討では正確に測定しなかったが，操作誤りの比率は圧倒的にPREFIX方式の方が多かった．

以上から，ローマ字かな漢字変換方式においては，POSTFIX方式の方が操作性に優れていると言える．現在，多くの日本語ワードプロセッサがPREFIX方式を用いているが，これは，ローマ字ではなくカナ入力をかな漢字変換のベースとしたためと思われる．（アルファベットからかなへはローマ字によって変換できるが，かなからアルファベットへは変換ができないのでモード切り替えが必要になる．）

5.2 バッチ入力変換方式

本検討の結果，キーボードのみで操作する限り，操作時間の短縮は困難なことが判った．同様の指摘は他の報告においても見受けられる．[Kimura85] 但し，この方式は，2.4項で述べたように辞書アクセスに時間を要する場合には有効である．

一方，この方法でも，試作した５個のキーを有するマウスを使用すると操作時間の短縮をはかることができた．これは同音語選択のためにマウスとポップアップメニューを使用する場合，マウスのキー操作だけで変換操作と同音語選択操作が可能なためなためである．

6 あとがき

一般に，かな漢字変換機能の高度化とは，一括して変換できる文書のサイズが取り上げられることが多い．単漢字よりは熟語，熟語よりは文節，単文節よりは複文節，さらにはべた書きと言う形で進展してきている．そして正確に変換できる限り間違いではない．

しかし現実には，100%正しく変換できるシステムは実現されていない．従ってかな漢字変換機能の高度化のためには，変換可能な文書サイズと共に，変換が正しく行なわれなかった場合の訂正方法の操作性の良否が大きな問題となる．ところが，変換対象の文書サイズが大きければ大きい程，誤った同音語の訂正のためにカーソル操作が要求され，むしろ煩雑になる．従って，ローマ字かな漢字変換入力においては，POSTFIX方式で，逐次確認しながら変換する方法が１つの現実的な解であ

ろう．またそのためにはキーボードのキー配列にも改善の余地がある．現在POSTFIX方式で変換するパーソナルコンピュータのワープロソフトの場合，変換キーとしてキーボード最上段の機能キーを用いているが，使用頻度が高いこれらのキーはスペースキーの近傍に配置し，親指で容易に操作できるようにすべきであろう．

一方，理想的には，べた書き変換のように変換処理を自動化してしまう方式が今後望まれるわけであるが，先に述べた通り誤変換の訂正操作の方法を検討する必要がある．今回の検討におけるバッチ入力変換方式において，後半の変換フェーズの操作においてマウスが有効であったが，誤変換の訂正処理はカーソル移動を伴うのでマウスとポップアップメニューを用いるのが１つの方法かと思われる．

最後に，本検討を進めるに当たり，協力いただいた小橋調査役，杉村主任，討論いただいた白鳥調査員に感謝します．

文　献

[Card80]
Card, S. K. et al, :"The Keystroke-Level Model for User Performance Time with Interactive Systems", Comm. ACM, Vol.23, No.7, pp. 396-410 (1980)

[Kimura85]
木村，小沼，粕川，:"「松」とJWORDの比較論"，コンピュータソフトウエア，Vol.2 No.2 (Apr.1985)

[Meyrowtz82]
Meyrowitz, N. et al.,:"Interactive Editing Systems", ACM Computing Surveys, Vol.14, No.3, pp. 321-415 (1982)

[Ohno84]
大野，深谷，Nievergelt, J.,:"ユニバーサルコマンドを有するマウスの研究（その１）"，信学技報 IE 83-102 (1984)

[Ohno85]
大野，杉山，:"ユニバーサルコマンドを有するマウスの研究（その２）",信学技報 IE 85- (1985)

[Smith82]
Smith, D. C. et al., :Designing the Star User Interface", Byte, Vol.7, No.4, pp.242-282, (Apr. 1982)

[Stelovsky84]
Stelovsky, J., :"XS-2:The User Interface of an Interactive System", Doctor of Technical Science Thesis, ETH, Zurich, (1984)

[Takenaka81]
竹中，坂内，細川，:" 英文キーボードによる日本文入力について"，情報処理学会日本文入力方式研究会資料１－１(1981.10.21)

[Tesler81]
Tesler, L. :"The Smalltalk Environment", Byte, Vol.6, No.8, pp.90-147, (Aug. 1981)

[Wechsler81]
Welchsler, A. C.,:"Zmacs Manual", Symbolics Inc.(1981)

[Weinreb81]
Weinreb, D. et al. :"Lisp Machine Manual (4th ed.)", Symbolics Inc. (1981)