香川大学経済論叢
第67巻 第3・4号 1995年2月 127-141

日本語・満州語の辞書作成のためのシステム（1）

本田道夫
今井慈郎

I はじめに

英語，日本語以外の言語について計算機を用いて研究する場合，従来は，その言語を英文字を用いて表記する方法，すなわち翻字での扱いがなされていたことが多かった。パソコンが比較的安価になり，そのような言語の研究に用いられるようになってからも，しばらくはそのような翻字表現での処理がなされてきた。しかし，研究対象としている言語の文字をパソコンで入力・表示したり，紙に印刷できることにより，視覚的にも，データ処理的にも，効率化がはかれることはいうまでもない。

そのような観点から，われわれは，英文字や日本文字以外の，フランス語，ドイツ語，スペイン語，ロシア語など，主としてヨーロッパ系の言語のアルファベット，また，言語研究には欠かせないギリシャ文字とそれにアクセント等の文字修飾がついたもの，さらには，古いロシア文字である教会スラブ文字などが扱えるパソコンシステムの開発を行ってきた[4][5]。また，その上で，言語研究のためのソフトウエアの開発も行ってきた[4][5]。

さらに，少人数での日本語・ブルガリア語の辞書作成のための準備も進めてきた。パソコンなどの計算機の能力を用いた辞書作成では，ブルガリア語から日本語への辞書のためのデータから，日本語からブルガリア語への辞書も比較的簡単に作成できるなどの利点も考えられる。しかし，そのためには，単なるワープロ機能だけではなく，見出し語をキーワード管理できるデータベース的

な機能等も必要であり，また，辞書作成を効率的に進めるためには，単語の綴りからの発音記号の生成や，単語の文節への区切りなどの作業で，補助的なソフトウエアの作成も重要であり，それらのアルゴリズムの検討を重ねてきた。

このようなシステム開発状況のなかで，日本語・満州語辞書作成について相談を受けた。われわれの考えていた辞書作成システムは，日本語・ブルガリア語辞書に限ったものではなく，一般的に少人数での辞書作成システムとして考えていたし，日本語・満州語辞書の方が仕様が簡単そうであったので，それまでの検討結果を，まず日本語・満州語辞書作成システムで実現することとした。

ただし，実際にシステム作成を始めてみると，当初の計画以外に，満州文字の文字種が多くキーボードから入力するのは大変であることから，英字での翻字表現を満州文字表現に変換する機能などの要望がでてきたこと，われわれの目的としていた日本語・ブルガリア語の辞書とは，編集方針がかなり異なるので別の独立したシステムとした方がよいと判断したこと等から，完成に時間がかかることが判明し，まず，満州文字を扱える入力・表示・印刷システムの制作を行うことにした。以下では，主としてこのサブシステムについて述べる。

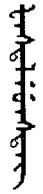
なお，最近では，Macintosh のパソコン，あるいは MS-Windows などでは，英文字，日本語文字以外の文字も扱えるが，それらで用意されている文字種は，われわれのシステムで扱える文字種よりもかなり少なく，それらを用いるためには文字フォントの登録が必要であるし，また，各種ユーティリティの開発の方法がまだ十分理解できていないため，当分は，MS-DOS の上での現システムの方式を拡張していく方針としている。ただし，将来は，それらの上での文字種の作成を行い，それまでのデータを移していくことも考えている。

II 満州文字入力・表示システム

1. 文字種および文字コードの割り当て

満州文字は，表 1 および表 2 に示すように半角文字 119 字，全角文字 61 字からなる。ただし，満州文字は縦書きの文字体系であるが，これらの表の文字は，横向きに設計している。英文字を用いた翻字表現も入れるために，日本語・満

州語の辞書は横書きのものとすることから，パソコンでは横書きのものとして扱うことにしたためである。なお，満州文字での単語は，たとえば，



のように，縦に中心線がつながったように表記されるものである。（この文字列は何の意味もないものである。）

なお，文字のサイズは，正確には，単純に半角文字と全角文字のサイズに分けられるものではないが，パソコンのMS-DOSの外字の登録機能を利用する現在のシステムで扱う都合上，半角文字119字と全角文字61字に分類して実現したが，実用上は差し支えないとのことであった。

満州文字ではないが，作成する辞書中での翻字表現のために，通常の英文字以外の文字も5文字必要である[6]。なお，文字コード表の最後のコードFB9Cの《‘》は，正確には文字ではなく，翻字表現において，kやhなどと合成され，k‘やh‘のように利用されるものであるが，本システムでは1文字として扱うことにした。

さらに，辞書中では，中国語との関係から，現在日本では利用されていない中国文字（全角の漢字）も2000字程度用いる必要がある。ただし，これらの文字については，まだフォントの作成が終了していない。

結局，本システムで扱わなければならない文字としては，半角文字124文字，全角文字61字＋2000文字程度ということになる。

これらに対する文字コードとしては，MS-DOSでユーザ定義文字用として，許されている16進数でF000～FEFFまでのうち，半角満州文字にはFB21～FB97を，翻字用の半角文字にはFB98～FB9Cを割り当て，全角満州文字には，FA80～FABCを割り当てた。中国の漢字用としては，F020～FA7Fまでを割り当てる予定である。

キーボードの配列については，図1のようにしている。ただし，中国語のための文字については，2000文字のファイルを用意しておき，マルチファイル編集の機能により，そこから切り出すようにする予定である。

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
FB20		﹁	﹂	﹃	﹄	﹅	﹆	﹇	﹈	﹉	﹊	﹋	﹌	﹍	﹎	﹏
FB30	﹐	﹑	﹒	﹓	﹔	﹕	﹖	﹗	﹘	﹙	﹚	﹛	﹜	﹝	﹞	﹟
FB40	﹠	﹡	﹢	﹣	﹤	﹥	﹦	﹧	﹨	﹩	﹪	﹫	﹬	﹭	﹮	﹯
FB50	ﹰ	ﹱ	ﹲ	ﹳ	ﹴ	﹵	ﹶ	ﹷ	ﹸ	ﹹ	ﹺ	ﹻ	ﹼ	ﹽ	ﹾ	ﹿ
FB60	ﺀ	ﺁ	ﺂ	ﺃ	ﺄ	ﺅ	ﺆ	ﺇ	ﺈ	ﺉ	ﺊ	ﺋ	ﺌ	ﺍ	ﺎ	ﺏ
FB70	ﺐ	ﺑ	ﺒ	ﺓ	ﺔ	ﺕ	ﺖ	ﺗ	ﺘ	ﺙ	ﺚ	ﺛ	ﺜ	ﺝ	ﺞ	ﺟ
FB80	ﺠ	ﺡ	ﺢ	ﺣ	ﺤ	ﺥ	ﺦ	ﺧ	ﺨ	ﺩ	ﺪ	ﺫ	ﺬ	ﺭ	ﺮ	ﺯ
FB90	ﺰ	ﺱ	ﺲ	ﺳ	ﺴ	ﺵ	ﺶ	ﺷ	ﺸ	ﺹ	ﺺ	ﺻ	ﺼ	ﺽ	ﺾ	ﺿ
FBA0	ﻀ	ﻁ	ﻂ	ﻃ	ﻄ	ﻅ	ﻆ	ﻇ	ﻈ	ﻉ	ﻊ	ﻋ	ﻌ	ﻍ	ﻎ	ﻏ
FBB0	ﻐ	ﻑ	ﻒ	ﻓ	ﻔ	ﻕ	ﻖ	ﻗ	ﻘ	ﻙ	ﻚ	ﻛ	ﻜ	ﻝ	ﻞ	ﻟ
FBC0	ﻠ	ﻡ	ﻢ	ﻣ	ﻤ	ﻥ	ﻦ	ﻧ	ﻨ	ﻩ	ﻪ	ﻫ	ﻬ	ﻭ	ﻮ	ﻯ
FBD0	ﻰ	ﻱ	ﻲ	ﻳ	ﻴ	ﻵ	ﻶ	ﻷ	ﻸ	ﻹ	ﻺ	ﻻ	ﻼ			

表1　モード1（コントロール＋1）での入力

（全角および半角満州文字）

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
FA80	ﮀ	ﮁ	ﮂ	ﮃ	ﮄ	ﮅ	ﮆ	ﮇ	ﮈ	ﮉ	ﮊ	ﮋ	ﮌ	ﮍ	ﮎ	ﮏ
FA90	ﮐ	ﮑ	ﮒ	ﮓ	ﮔ	ﮕ	ﮖ	ﮗ	ﮘ	ﮙ	ﮚ	ﮛ	ﮜ	ﮝ	ﮞ	ﮟ
FAA0	ﮠ	ﮡ	ﮢ	ﮣ	ﮤ	ﮥ	ﮦ	ﮧ	ﮨ	ﮩ	ﮪ	ﮫ	ﮬ	ﮭ	ﮮ	ﮯ
FAB0	ﮰ	ﮱ	﮲	﮳	﮴	﮵	﮶	﮷	﮸	﮹	﮺	﮻	﮼			

表2　モード2（コントロール＋2）での入力

（翻字のための文字）

2.　システムの実現と問題点

本システムでは，全角文字と半角文字を併せて，185＋2000文字程度を扱う必要がある。一方，最近のNEC-PC9801で表示できるユーザ登録文字は，マニュアル[3]では，16進数のJISコードで7621～767Eおよび7621～777Eの118文字となっている。（ただし，実際には，7601～7680までと，7701～7780までの，合計256文字までは登録できるようである[1][2]。EPSONのPC 9801互換機では，7621～767Fまでと，7721～777Fまでと若干少ない文字しか扱えないようであ

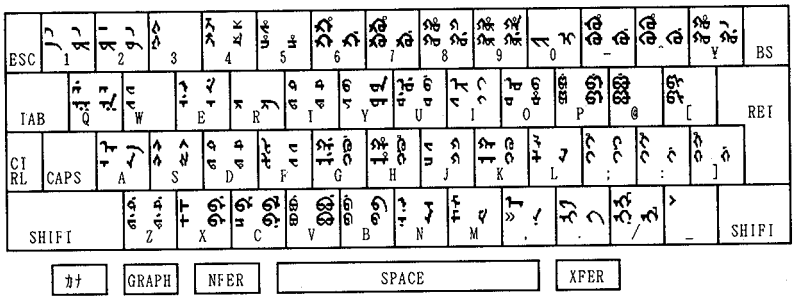


図1　満州文字のキー配列（モード1）

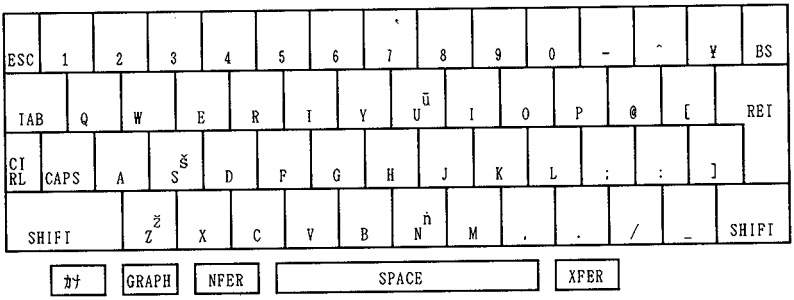


図2　満州文字翻字表現用文字（モード2）

る。）

登録可能な文字数よりも，登録しなければならない文字数の方が多く，これに対処するために，既に開発・利用していた欧米系の文字表示のためのシステムと同じような，以下の方法をとることにした。[5]

各文字には表1，表2で示した文字コードを割り当てておき，それらの文字を表示するときに，ダイナミックに，表示できる文字として，7601～7680 あるいは 7701～7780 までのどれかのコードを割り当てて登録する。表示の必要がない場合には，その表示のためのコードは別の文字に利用される。

つまり，満州文字のためのシステムとしては，既に開発・利用していた欧米系の文字表示のためのシステムを元に，半角文字については，従来の方法と同様の方法で，全角文字はMS-DOSの外字登録の機能をそのまま用いることにした。もともと，NECのPC-9801のMS-DOSの外字登録は，全角文字についてのみサポートされているものであるが，それを半角文字登録のために，PC-9801に固有な特殊な性質を用いて，半角文字の登録表示を行っていた。つまり，半角文字の表示が特殊な処理をしていた。したがって，全角文字については，ごく普通に文字登録機能を用いるだけであり，問題なく処理できると考えていた。

しかし，この方法ではつぎの問題が生じた。

問題点1　奇数個の半角文字に全角文字が続くと，全角文字の表示が正しくおこなわれない。

この問題点は，欧米系の文字システムの場合も，同様な現象が通常の全角文字との間では生じていたので，予期できないわけではなかった。欧米系の半角文字のみの場合には，全角の日本語文字などと同時に使われる場合にも，単語の前後に半角空白を用いることにしていたので，実用上は問題とはならなかった。しかし，満州文字の場合には，空白をあけることのできない単語中に半角文字と全角文字が入り混じるので，解決しなければならない問題となった。

さまざまな方法を検討・テストしたが，最終的には，《全角文字を左右半分に分けて，それぞれを従来の方法と同様に半角文字として登録し，全角文字に対しては，連続した半角文字2つで表示する》という解決方法をとることにした。しかし，次の点が依然として気に懸かる点としては残ってしまった。

・登録できる文字数が250個程度と制限されている一方で，全角1文字に対して2文字分の登録を行うことになる。満州文字の場合には，全角文字も比較的よく出現するので，1画面に表示できる半角文字2000字に対

して，登録文字数が不足するということが考えられる。

なお，満州語辞書を扱う場合，現在の日本で用意されている JIS 第一および第二水準の漢字だけでは足りず，2000 字程度の漢字の外字も扱わなければならないので，この点でも登録文字数が不足するということが考えられる。

この問題は，実際の文書等での全角文字と半角文字の使用状況が分からないので，非常に気に懸かる点ではあるが，さしあたりこの方法での使用で様子を見て，実際に問題となった時点で，別の対処方法——たとえば，満州語の単語の前後では，半角空白を開けるということにすれば，中国漢字のための登録については，全角を 1 文字分で登録できる——を考えることにした。

このような，《全角文字を左右半分づつの 2 つの半角文字として扱う》という解決方法をとることにより，つぎのような新たな問題が生じることとなった。

問題点 2　1 つの全角文字に対する，表示のためのコードが 4 バイトとなる。これまでのシステムは，文字コードに対して，表示のためのコードをダイナミックに決定して返す機能を C 言語等のプログラムから呼び出して利用できるようにしていた。これまでは，当然 2 バイトのコードのみを返す仕様としていたが，4 バイトのコードも混じることになる。

本システムの元となった欧米系文字のためのシステムは，言語研究用の各種処理の C 言語でのプログラムや，辞書作成のデータベースなどで利用できるように，通常の標準文字出力に対する表示処理の BIOS (Basic I/O System) に対して，トラップ処理するデバイスドライバとして組み込む方式で実現していた。ただし，キーボード入力と表示の機能だけでなく，つぎのような機能も実現していた[5]。

(1) 拡張文字コードに対し，表示のためのコードを返す。
(2) 現在の入力モードを保存しモード0（通常入力）とする。
(3) 保存していた入力モードに復帰する。
(4) カーソルの形状設定および表示を行う。
(5) 画面の再表示を行う。
(6) 入力モードを直接指定したモードに設定する。
(7) 全角文字，半角文字等の文字コードの範囲を返す。
(8) スラブ系文字等の文字コードの範囲を返す。
(9) 大文字の文字コードに対して，小文字の文字コードを返す。
(10) 小文字の文字コードに対して，大文字の文字コードを返す。
(11) 2つの文字列を辞書式順序で比較した結果を返す。

これらの機能は，それまでの主としてスラブ系の言語における言語研究のためのさまざまなソフトウエアの開発を行ってきた結果，それぞれのソフトウエアごとにそのような処理を行うよりも，本システムで共通的にサポートすべきであろうとして，設けた機能である。また，たとえば，(1)の機能のように，エディタ，あるいはデータベースなどでの画面表示などのように，画面の大部分の表示のし直しが，かなり頻繁に行われる場合，標準出力を介するよりも，表示登録文字コードを直接受け取って，テキストVRAMに置くようにした方が，画面表示が高速に行えるなどの，必要から設けたものもある。

これらの機能については，C言語あるいはアセンブラによるプログラムから，次のようにして，直接呼び出すことができるようにしており，その方法は，ただ1つの関数

```
unsigned int        MLingCall (int  機能指定, int  渡す値);
```

を用意し，引数の《機能指定》の値により，さまざまな機能を指定するようにしていた。たとえば，機能(1)を呼び出すには，

display_code = MLingCall（0x0100，スラブ文字などの文字コード）；

とし，この関数呼び出しに対する2バイトの値が，表示コードとなっていた。もちろん，機能(2)などのように，戻り値を利用することのない場合もあるが，戻り値を利用する場合には，それらはすべて，関数 MLingCall の2バイトの符号なし整数値を用いていた。

満州文字を扱えるシステムについても，そのような機能は継承するべきであると考えたことはもちろんであるし，ロシア文字等の為のシステムと，満州文字のためのシステムのほぼ同様なプログラムを2つ開発・維持することに伴う煩雑さを避けるため，キーボードに対する文字コードのテーブルと，文字フォントのみを切り替えることにより，これまでの欧米系の文字のためのシステムにも，満州文字のためのシステムにもなるように設計するつもりであった。

しかし，満州文字に対処するための変更では，このうちの《機能指定》が，16進数で0100のときに返す値についてのみ，戻る値として，2バイトの値も4バイトの値もあり得るということになった。この仕様の変更自体はさして煩雑な処理ではないが，この仕様変更に伴って，現在までに開発しているソフトウエアをすべて変更しなければならないということは，大変な作業となることが予想された。したがって，これまでのソフトウエアに対しては問題なく作動するような仕様変更が是非とも必要となった。

もちろん，従来の機能は残したまま，この機能だけのための，4バイトの値を返す「符号なしロング整数」型の関数を作成して対処する方法もあったが，全体として統一がとれないし，また，そのようなつぎはぎ的な対処は，今後のシステムの改良において問題となることも予想されるため，なんとか2バイトで返せる方法を考えることとした。

表示文字コードが16進数で7601～7680および7701～7780までのものであ

ることから，次のような処理を行うことにし，2バイトで文字コードを返すことができるようにした。

・《機能指定》として，16進数0101を設ける。

・この《機能指定》に対しては，まず，与えられた文字コードが，全角の満州文字であれば，左右半分ずつの半角文字コードしての2バイトの表示文字コードを2つ求める。これらのコードは，76XXあるいは77XXのものである。ただし，XXは01～80の範囲の値である。(半角の満州文字の場合には，2バイトのコードであり，後半の2バイトとしてはゼロとでもしておく)。

・もし，そのコードが76XXであればXX－0x01，

77XXであればXX－0x01＋0x80

の値を計算する。(ここで0x01，0x80は16進数での表現とする。)その値は，0から0xFFまでの値となり，1バイト分での値である。しかも，その値が0x7F以下であれば元のコードは76XX，0x80以上であれば元のコードは，77XXのコードであることがわかり，容易に2バイトの元のコードが復元できる。

・関数MLingCallの値として，上記のように2バイトの値から計算した1バイトの2つの値v1，v2から，さらに値v1<<8＋v2を求めて返す。この値は，2バイトである。(v1<<8は，v1の値を8ビット左にシフトしたものを表す)

・受け取った側では，v1，v2を復元し，さらに元の表示コードを復元することができる。

このようにすることにより4バイトの文字コードを2バイトの文字コードにして，関数 MLingCall の値として返すことが可能となる。なお，MLingCall を呼び出すソフトウエア側では，表示文字コードが2バイトとなるか4バイトとなるかは分かっているので，2バイトとなる場合には，《機能指定》として従来からの 0x0100 を用いればよい。0x0101 を用いて，返された値の最初の2バイト分を用いてもよい。いずれにしても，これまで開発したソフトウエアは，変更なしに利用できるし，満州文字用に開発する今後のシステムに対しても，2バイトの値を返すという範囲内で対処できた。

III 満州文字印刷システム

印刷のための方法としては，従来の欧米系の文字システムで採用していた方法と同様の方法で扱うこととした。

この方法で特に問題はないが，満州文字での単語の場合，先に述べたように各文字の中心線は連続させる必要があるので，文字間の間隔をあけることはできない。一方，日本語文字などは，多少間隔をあけた方が見やすい。したがって，たとえば満州文字の間隔は0ドット，日本語文字の間隔は2ドット程度とするように，印刷プログラムで自動的に間隔調整を行うことが考えられる。

しかし，自動調整では，満州文字と日本語文字が混ざって含まれているテキストなどでは，CRT 画面上では合っている上下の文字が，印刷すると左右にずれるということも起こりうる。したがって，デフォルトの文字間隔は，満州文字，日本語文字とも0ドットとしている。これを変更する場合は，これまでのシステムでも用意していたコマンドを用いて，ユーザの責任で行うこととした。

IV 翻字表現の満州文字への変換について

一応，図1に示すようなキー配列で，モード1でキーボードから直接満州文字を入力することができるようにしている。しかし，実際にキーボードから直接入力することは，難しいものがある。慣れればそうではないかもしれないが，既開発のスラブ文字などの場合のように，よく用いる文字種はそれほど多くな

く，しかも，ある程度の標準キー配列が用意されすでにタイプライタなどで，そのキー配列に慣れている場合と比較すると，やはり取り掛かりに難しいものがあるようである。

また，満州文字システムのまず第一の目的である日本語・満州語の辞書においては，翻字表現も入れるということもあり，翻字表現を満州文字の文字列に変換する方法を考えることにした。ただし，現在進めている辞書用のデータの入力では，既に翻字表現での単語の入力がなされていることもあり，変換しながら入力する方法ではなく，すでに翻字表現で入力されているものを変換する方法をとることにした。

翻字表現の変換は，日本語のローマ字表現をひらがなどに直す場合と比べると，規則はかなり複雑である。日本語・満州語辞書の作成を考えておられる満州語に詳しい先生からは，約６０００行ほどの翻字の規則についての情報をいただいた。しかし，そのままでは，プログラムの作成に結びつけられないため，grep などのソフトウエアを用いて，なんとかプログラムの作成に結びつくような整理を行った結果，約1000行ほどに整理できた。翻字表現を提案した文献[6]もあるが，元々の印刷が鮮明でない上に，満州文字に不慣れなせいもあり，それから十分な情報が得られていない。ただし，どうも，その文献では，最初にいただいた約6000行分に相当するほどの情報は入っていないようである。

なお，満州語の翻字表現は，以下のようになっているようである。

母音：a, e, i, o, u, ū

子音：b, bg, bk, bl, bm, bn, c, d, dz,
f, g, g', h, h', j, jk, k, k',
l, m, n, ṅ, ng, ngg, ngk, ngl, ngm, ngn,
p, r, s, sy, ŝ, t, tsi, tsin, ts',

w, y, z̄

語末となることのある子音：

b, k, l, m, n, ng, r, s, t

変換の規則が複雑となる大きな要因は，どうも以下のような点にあるらしい。

(1) 同じ翻字の文字でも，たとえばつぎの文字 a の例のように，単独の場合，語頭の場合，語中の場合，語末の場合で異なる満州文字に対応する。このことは，母音に限らず，ほとんどの文字に該当する性質である。ただし，FB30＋FB59 という表記は，それら 2 つのコードの文字をつなげることを意味する。

単独の場合	FB30＋FB59
先頭の場合	FAA4
語中の場合	FB57
語末の場合	FB23 b, p, k', g', h' の後のみ
	FB59 上記以外

(2) つぎの例の k のように，前の母音と後ろの子音によって，対応する満州文字が異なる。

FA97
- a あるいは i あるいは o の次の語末の k。
- 語頭あるいは語中の buk, cuk, duk, fuk, juk, luk, muk, nuk, ruk, puk, suk, ŝuk, tuk, yuk の場合の k。
- tek の語中の k。

FD71 ・語頭の ekc, ekj, ūkc, ūkj の k。

・語頭あるいは語中の bek, cek, dek, fek, gek, hek, jek, kek, lek, mek, nek, pek, rek, sek, šek, wek, yek の場合の k。ただし，k のあとに c または j がつづくとき。

FB75 ・語頭の ek, ūk の場合の k。ただし，k のあとに c，j 以外の文字が続くとき。

・語中の ūk の場合の k。

・bek, cek, dek, fek, gek, hek, jek, kek, lek, mek, mek, pek, rek, sek, ŝek, wek, yek の場合の k。ただし，k のあとに c，j 以外の文字がつづくとき。

・語頭あるいは語中の guk, huk, kuk の場合の k。

(3) 語末に子音がくることがあるように，必ずしも，子音＋母音というようには区切れない。

最初に，約 6000 行ほどの規則をいただいたときに，まず気になったことは，(3)の性質に関連することであるが，『与えられた翻字表現が，必ずユニークに「子音＋母音」のように区切れるか』ということであった。たとえば，babga の場合，

(a) bab と ga と区切るべきなのか，

(b) ba と bga と区切るべきなのか

ということである。これについては，『先頭からできるだけ長くとれる文字の並びで区切る』という原則にしたがってよいことであった。(したがって，上記の場合には(a)の区切り方となる。)

この区切りの問題が片づけば，あとは，区切られた文字列を多少複雑かもしれないが，約1000行程にまとめなおした規則にしたがって満州文字に変換すればよいということになり，プログラム作成にとりかかった。

しかし，かなりプログラムが完成した時点で，実は，文字列の区切りが，上記の原則ではだめであるという連絡があり，変換プログラムの作成は一時中断しているのが現状である。

満州語に詳しい人はごく自然に変換ができるらしいので，おそらく何らかの規則があるものと思われるが，今後かなり話し合いながら，その規則を聞き出す以外に方法はないようである。

〔1〕 Igal Blumenreich「IBM PC ソフトを PC-9801 動かすポーティング手法入門」『インターフェース』，No 148, 1989, CQ 出版社

〔2〕 Igal Blumenreich「IBM PC ソフトを PC-9801 で動かすの補足：半角文字を外字登録する」『インターフェース』，No 148, 1989, CQ 出版社

〔3〕「MS-DOS 3．3 C ユーザズリファレンスマニュアル」日本電気

〔4〕 本田道夫，山田勇「言語学研究へのパーソナルコンピュータの応用」『香川大学経済論叢』第63巻第2号（31-64），1990

〔5〕 本田道夫，吉岡珠実，山田勇「スラブ系・ラテン系の言語研究のための基礎システム」『香川大学経済論叢』第64巻第2・3号（423-496），1991

〔6〕 P G. Von Mollendorff「MANCHU GRAMMAR with ANALYSED TEXTS」『Chinese Customs Sereice』American Presbyterian Missions Press, 1982