日本語・満族語の辞書作成のための補助システム（Ⅶ）

本　田　道　夫

Ⅰ．はじめに

2009年3月の「日本語・満族語の辞書作成のための補助システム（Ⅵ）」［本田2009］から3年近くになるが，その間にも本システムの変更やエラー修正は行ってきた。その主なものは，編集サブシステムであるJMMan. exeについての機能追加に伴う変更やエラー修正などであったが，文字鏡文字の追加登録の場合は，編集サブシステムだけでなく，印刷サブシステムなどの変更・再構築も必要であった。編集サブシステムについては，その回数はバージョン番号でみると2009年3月17日の1.84から2011年12月6日の2.03まで合計20回に達していた。これまで，［本田1995］，［本田1998］，［本田2003］，［本田2006］，［本田2006］などでは，満州文字と文字鏡文字の文字コードの位置を変更するなど，大きな仕様変更についてだけ記してきたが，ここで一度，システム開発の様子を紹介する意味でマイナーな変更なども含めて，これらの変更・修正について説明する。

辞書作成作業における新規データ入力段階の頃は，結構頻繁に扱える文字鏡文字の追加登録作業を行っていたために手順を忘れることもなく，追加登録要請があればスムーズに問題なく対応できていたが，データ入力がある程度進ん

できた段階から，また特に入力したデータの見直し・校正の段階に入ってからは，それほど追加登録の要請が頻繁ではなくなってきたために，追加登録要請には，作業手順を思い出しながら作業を進めた。実は，扱える文字鏡文字を追加登録するということは，編集サブシステムのプログラム変更だけではなく，簡易印刷および最終の Tex を用いた印刷の両印刷サブシステムのプログラム変更など他のサブシステムのプログラム変更，さらには本システムで扱える文字鏡文字のコード順と画数順の一覧表の作成など，データ入力や校正作業をする上での便宜のためのものも作成しており，かなりの作業手順であった。そこで，追加登録の要請に対して迅速に対応するために，それらの作業手順をドキュメント化しておくことにした。まとめてみると結構な段階の作業であったので，これらについても本論文で説明する。

Ⅱ．編集サブシステムの変更・修正

ここでは，2009 年 3 月の「日本語・満族語の辞書作成のための補助システム（Ⅵ）」［本田 2009］以降の編集サブシステムの変更・修正について記す。なお，本編集サブシステムは，

- JMacs. exe　：英数字と日本語文字での文書・プログラム等の編集用
- JMMan. exe：満族語・日本語辞書作成のための原稿編集用
- JMSlav. exe：スラブ文字，教会スラブ文字，ギリシャ文字，特殊なラテン文字など西欧・東欧の多くの文字の言語研究で用いられる特殊な文字を利用した文書作成，辞書作成などでの編集用

の 3 つの編集システム（エディタ）のプログラムとして開発している。つまり，プログラムのソースコードは，３つの編集システムの共通部分と上記３つの各システム固有の部分からなり，固有の部分はコンパイルフラッグを切り替えることにより，目的のものを作成するようにしている。そのために，以下に記す編集サブシステムの変更・修正においては，満族語・日本語辞書作成用に関わるものだけではなく，共通的な処理，あるいは JMSlav. exe に関するものも含んでいる。各バージョンの後ろに関係する編集システムを記している。

○バージョン 1.84（2009/03/17）JMacs. exe，JMMan. exe，JMSlav. exe

Windows でクリップボードを利用したカット＆ペースト，あるいはコピー＆ペーストができるような機能として，キル（Kill）とヤンク（Yunk）のコマンドがある。コマンド C-Y（コントロールキーを押下したまま Y のキーを押下，以下 C-○については同様）で最後にキルしたテキストをカーソル位置に復元できる。C-Y に続いて ESC-Y（Escape キーの押下に続いて Y キーを押下，以下，ESC-○については同様）を入力すると，C-Y で復元したテキストに代わり，最後から 2 番目にキルしたテキストが復元される。さらに続けて ESC-Y を入力するごとに，最後から 3 番目にキルしていたテキストが復元されるというように遡ってキルされていたテキストを復元することができる。以前は正しく機能していたが，最近，2 回目の ESC-Y から機能しなくなっていたのに気がつき，これを修正した。

また，キーボードから入力する一連の複数のコマンドをキーボードマクロとして登録しておき，C-X E のコマンド 1 つを入力することにより，それら一連のコマンドを実行する機能がある。ところが，検索のコマンド C-S に引き続き，検索対象の文字列を入力し，そのあとに C-B C-P のようにカーソル移動する一連のコマンドを直接キーボードから入力したときは問題なく実行できたが，その一連のコマンドをキーボードマクロとして登録しておき，実行すると，最初の C-B のコマンドが実行されなくなっていたので，それを修正した。

これら 2 つについては，以前は問題なく機能していたが，その後のプログラム変更の時に機能しなくなっていたと思われるが，いつからそのようになっていたのかは不明である。

○バージョン 1.85（2009/09/23）JMSlav. exe

JMSlav. exe において，ラテン小文字の上下に一つあるいは複数のアクセントやウムラウトなどの記号が付く特殊文字が正しく表示されなかったのを修正した。

○バージョン 1.86（2009/09/24）JMacs. exe，JMMan. exe，JMSlav. exe
バッファを変更した後，最小化ボタンをクリックしたときにプログラムが終了してしまうエラーを修正。今まで筆者自身では，このような操作を行ったことがなかったので，エラーに気がついていなかったが，あるユーザから指摘された。

○バージョン 1.87（2010/01/14）JMMan. exe
扱える文字鏡文字を追加登録した。

○バージョン 1.88（2010/01/30）JMacs. exe，JMMan. exe，JMSlav. exe
Windows プログラミングには，メモリデバイスコンテキストというものがある。Windows の画面に相当する大きさのメモリ領域を，プログラムが開いたウインドウ画面と同じように扱えるメモリデバイスコンテキストとして確保しておき，ウインドウに直接表示する代わりに，確保したメモリデバイスコンテキストに出力し，さらにそのメモリデバイスコンテキストから実際のウインドウ画面に出力するというように用いることができるものである［シルト 2000］。
Windows ではキーボード操作，マウス操作などの情報は Windows からメッセージとして，その操作が行われたウインドウを開いているプログラムに送られてくる（図 1 の①）。そして，各プログラムは自分に送られてくるメッセージを常時監視しており，メッセージを関数 GetMessage() で受け取り，関数 TranslateMessage() で変換し，さらに関数 DispatchMessage() で Windows に返す（図 1：メッセージの処理の②，③）。そして Windows は，その返されたメッセージを，さらにそのプログラムのウインドウ関数にメッセージとして送る（図 1 の④），各プログラムは①，②，③のメッセージを処理する決まり切った部分と，メッセージ④を受けて実際の処理を行うウインドウ関数の部分からなるように作ることになっている［シルト 2000］。

そのようなメッセージの一つに，ウインドウを書き換える命令の WM_PAINT というメッセージがある。ただし，WM_PAINT メッセージは自身のウインド

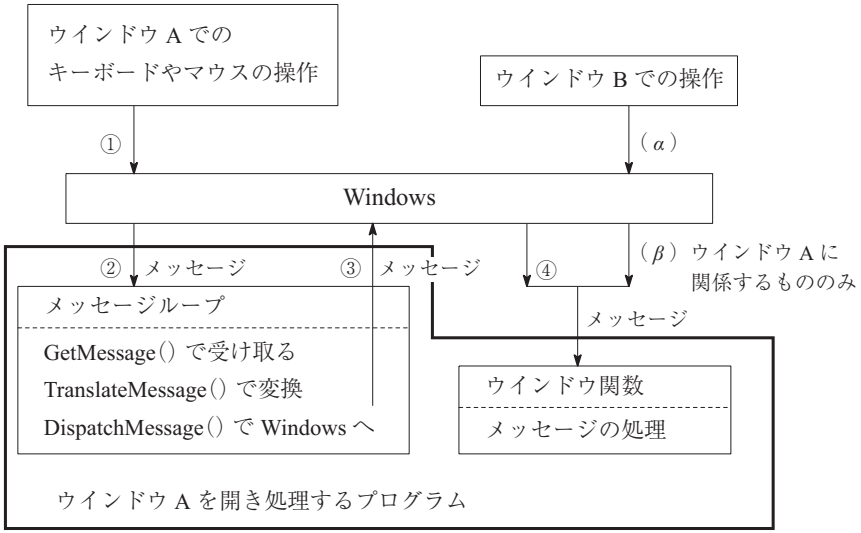

図１：メッセージの処理

ウの操作だけでなく，他のウインドウを操作したときにも Windows から送られてくる。たとえば，プログラム A が開いたウインドウ A にプログラム B が開いたウインドウ B が上に重なって表示されたあと，プログラム B のウインドウ B が閉じられた場合に，それまでウインドウ B の下に隠れていたウインドウ A が表示されなければならない。しかし，これはプログラム A は全然関係ないところで行われた操作によるものであり，プログラム A では再表示の必要性が分からない。そこで，プログラム A にウインドウ A を再表示せよという命令として Windows から WM_PAINT というメッセージを送るようになっている（図１の（α）と（β））。Windows が下に隠れるウインドウ表示を保存しておき，再表示が必要になったときに，保存していたものを使うという方法も考えられるが，Windows ではこのような方法はとらずに，再表示が必要なプログラムに WM_PAINT メッセージを送り，再表示はそのプログラムに任せている。これは，その画面を再表示するもっとも適切な方法は各プログラムが

知っているはずというように考えているからとのことである。そして，各プログラムは，WM_PAINT に対する処理方法として，

⑷　画面に表示する内容を再度計算・作成しながら表示する，
⑸　自身のウインドウ画面を書き換えるたびにその内容をメモリデバイスコンテキストに保存しておき，送られた WM_PAINT に対しては，そのメモリデバイスコンテキストを表示する。

という 2 つの方法がとれるようになっている。

編集サブシステムで WM_PAINT のメッセージに対応する方法としては，⑸の方法が適切と思われるので，以前に一度，⑸の方法でプログラムを作成したことがあった。しかし，そのときは，画面のちらつきが激しく，方法⑷を採ることにした。デバッガで調べて分かったことであるが，Windows から各プログラムに対して，WM_PAINT は非常に頻繁に送られており，そのたびに書き換えがなされるので画面がちらつき，あまりよい状態とは思えなかったので，その時点では採用していなかった。しかし，ある Web ページで，ウインドウ関数 FrmWProc()中で，背景色を白で塗りつぶす指令の WM_ERASEBKGND メッセージの処理を

```
case WM_ERASEBKGND:break;
```

のように処理し，デフォルトの処理に任さなければちらつかないという情報を入手したので，再度，メモリデバイスコンテキストを用いる方法を試みた。そして，確かにちらつきがなくなったので，この方法を採用することにした。この方法で各プログラムが自身の画面表示を変更するときは，まず表示内容をメモリデバイスコンテキストに書き出す。このことはメモリデバイスコンテキストに保存することにもなる。その後，関数 InvalidatRect()を呼び出し，Windows に自分自身のプログラムに WM_PAINT を送るように要請する。送られた WM_PAINT に対しては，そのメモリデバイスコンテキストを表示するようにして

いる。この方法を採用することにより，次のような点でプログラムも簡単にすることができた。

・プログラムで画面に表示するには画面に対応したデバイスコンテキストを取得して,そのデバイスコンテキストに出力するようにする。ただし,デバイスコンテキストは，他のプログラムが出力するときに，そのプログラムでも取得するので，できるだけ表示直前に取得し，表示が終了すれば直ちに解放するようにプログラムを作成するのが Windows でのプログラミング作法のようである。直接表示する場合には GetDC() と ReleaseDC() という関数でデバイスコンテキストの取得・解放を行うのに対して，WM_PAINT の処理としてデバイスコンテキストの取得・解放には BeginPaint() と EndPaint() を用いるというように，取得と解放に用いる関数が異なる。画面への表示処理はどちらから呼ばれても処理は同じであるので，プログラムを簡潔にするために表示に関係する関数はどちらにも対応する共通のものとして作成し，プログラムから直接か WM_PAINT の処理としてか，のどちらから呼ばれたのかに応じた取得方法とする必要があったが，その必要はなくなった。つまり，これまでは，表示に関係する関数に，GetDC() を使うのか，BeginPaint() を使うのかの情報を渡していたが，メモリデバイスコンテキストを用いるようにすることにより，WM_PAINT メッセージの処理だけでよいので，渡す必要がなくなった。

・これまでは，編集対象のファイル内容を表示している部分と，メッセージや検索中の入力された文字列，マクロ定義途中の入力などの表示を行うメッセージ行と，カーソルが位置している行位置とカラム位置の表示や対象ファイル名，キャップスモード，括弧対応モードなどを表示するモード行は，表示が必要になったときにプログラム中で直接表示していた。そして，再表示に際して，それらの再度表示内容を個々に再構成して出力していたが，メモリデバイスコンテキストを用いれば，それらすべてを保存できているので，その必要がなくなった。

・なお，処理時間も速くなるはずであるが，最近のパソコンの CPU の能力が十分高いためか，処理方式の変更による高速化は実感できなかった。

○バージョン 1.89（2010/01/30）JMacs. exe，JMMan. exe，JMSlav. exe

これまで Windows から呼び出される主関数である WinMain()中で，CreateWindow()関数で作成したウインドウのウインドウハンドルを WinMain()の局所変数 hwnd に入れておき，ウインドウハンドルを必要とする関数に引数として渡していたが，これを大域変数 hWnd とすることにより，全体として引数の個数を減らすことにした。引数の数を減らすということはその引数を受け取る関数実行時のスタックへの引数の積み上げと，終了後のスタックの積み下しの操作が減ることになる。

○バージョン 1.90（2010/02/01）JMacs. exe，JMMan. exe，JMSlav. exe

初期フォントのサイズ設定は，関数 FWmCreate()中で既に設定しているので，それ以降に呼ばれる WinMain()中での再度設定する必要はないので，コメント化した。

プログラムの変更ではないが，これまでは変更が間違っていた場合に備えて，変更前のプログラム部分をコメント化して残しておいたが，それらがかなりの量になってきたし，安定して動いている部分は，コメント化して残していたプログラム部分を削除した。これにより，ソースプログラムのファイルサイズも少し小さくなった。

○バージョン 1.91（2010/02/17）JMSlav. exe

スラブ文字は文字数が多いために，通常のキー入力，シフトキーを併用したキー入力以外に，ALT キーを併用したキー入力，ALT キーとシフトキーを併用したキー入力による入力も用いている。また，テンキーを用いての入力も採用している。入力に用いるキーの組み合わせが異なると図 1 の④としてウインドウ関数に送られてくるメッセージも異なる。たとえば，通常キー入力とシフトキーを併用したキー入力，コントロールキーを併用したキー入力は WM_

CHAR メッセージ，あるいは WM_KEYDOWN メッセージとして文字コードが送られてくるが，ALT キーを併用した通常キー入力は MW_MENUCHAR メッセージとしても文字コードが送られてくる。テンキーの数字，あるいは ALT キーと併用したテンキーの数字は WM_CHAR メッセージ，あるいは WM_KEYDOWN メッセージとして文字コードが送られるが，テンキーの記号は WM_CHAR メッセージだけが送られてくる。スラブモードで入力文字コードが 0 以上かつ 9 以下の場合は，ALT キーを併用したテンキーであると処理していたが，GetKeyState(VK_MENU)の値をチェックし，ALT キー押下を確認するようにした。

○バージョン 1.92（2010/03/18）JMMan. exe，JMSlav. exe

スラブ文字系のキー入力関係を大幅に修正・変更した。テンキーの数字と記号の入力については，シフトキーと ALT キー併用の入力は，ALT キーだけの併用入力とプログラムでは区別できないので，シフトキーの代わりにコントロールキーを用いることにした。つまり，テンキー部についてだけは，単なる数字キーと記号キーの入力，コントロールキーと併用したキー入力，ALT キー併用のキー入力，ALT キーとコントロールキー併用のキー入力の 4 通りの入力とした。

最初のスラブ系言語用のシステムは NEC の PC-9800 シリーズの MS-DOS 上でのソフトウエアとして開発し，さらに PC-9800 上の Windows システムに移植した。そのときは，テンキー入力についてもシフトキーと NFER キーの併用が可能であったので，DOS/V 機の Windows に移植したときも，そのままにしておいた。スラブ系言語用として利用していたユーザは，DOS/V 機のパソコンが普及したあとも PC-9800 シリーズが販売されている間は PC-9800 を利用し，DOS/V 機を利用するようになったときは，テンキーによる入力もあまり複雑なものを用いていなかったので，結局かなりの間，シフトキーと ALT キー併用のテンキー入力は，できないことに気がついていなかった。

なお，PC-9800 シリーズのときは ALT キーはなかったので，ALT キーではなく，NFER キーを用いていた。DOS/V 機には NFER キーに相当する「無変

換」キーがあるが，現在の Windows 上でジャストシステムの日本語変換システム ATOK をインストールしている場合，「無変換」キーは ATOK の入力文字の切り替えに用いているため，日本語変換をオフにした状態で「無変換」キーを用いるか，あるいは，上記で説明したように ALT キーを用いるかの両方が可能なようにした。なお，このことは，満族語用についても同様であり，ALT キーあるいは「無変換」キーを同様な機能のキーとして併用することができるようにした。さらに，スラブ系文字入力で最上段キーのシフト併用のキーの処理が間違っていたのを修正した。

○バージョン 1.93（2010/05/10）JMMan. exe

扱える文字鏡文字の追加登録を行った。この追加登録により，先頭バイトが 16 進数（以後も文字コードについては 16 進数で表記する）で FA の文字コード部分がいっぱいになり（つまり，FA00 から FAFF まで使ってしまった），FB の文字コード部分を利用するようになった。編集サブシステムと印刷サブシステムのソースプログラムの変更が必要であった。

○バージョン 1.96（2010/05/17）JMMan. exe

文字鏡文字のうち，文字鏡文字番号が 200000 程度を超えている文字は，これまでは JMMan. exe で表示ができなかった文字コードのものが，JmML2MJ. tbl の作成時に特別の処理をして，表示できるように修正した。

ただし，JMMan. exe では表示できるようになったが，この時点では Tex では表示できていなかった。ただし，2011 年 11 月 14 日に Tex でも表示可能となった。

○バージョン 1.97（2010/06/14）JMSlav. exe

スラブ系言語での特殊文字（コード F6F0－F6F9）のフォントは，以前には，TrueType フォント YSpcRF. ttf に作成して入れていたが，いつの間にか，YSpcRF. ttf からはなくしていた。おそらく，教会スラブ文字 TrueType フォント YOldBLRF. ttf に入れるように考えて YSpcRF. ttf から削除していたのであろ

うが，教会スラブ文字の方に入れていなかったことが問題だったようである。文字フォントの TrueType ファイルの変更は，筆者の担当ではないためにいきさつは分からない。なお，YSpcRF.ttf を以前のように，先頭部分 0x20－0x09 にスラブ文字用特殊記号，0x41－0x5E に上下につける記号を入れるようにし，それに伴ってプログラムも変更した。

○バージョン 1.98（2010/07/01）JMMan. exe

扱える文字鏡文字として文字コード FA31 と FB85 に登録していた文字が，同じ文字鏡文字 232638 であったが，別の文字が表示されていた。原因は，JmMl2Mj. TBL が古いものであった。最新のものにすることにより同じ文字が表示されるようになった。なお，このように同じ文字鏡文字に対して JMMan. exe では別の文字コードとして登録されているものを調べたところ複数あった。このような文字鏡文字の二重追加登録は起こるとは思っていなかったのでチェックすることはしていなかったが，追加登録する文字が多くなると起こりうることと考えて，最初からチェックするようにしておけばよかった。システムとしては，二重登録が見つかった場合，どちらかに統一すべきであるが，既に入力されているファイルでは両方用いているため，当面は両方とも使えるようにした。最終的にどちらか一方に統一するかどうかは決めていない。

○バージョン 1.99（2010/11/30）JMMan. exe

キーボードから入力した英文字列（ローマ字列）を満族語文字列に変換するための規則が変更（バージョンアップ）されたので，それに対応した変換規則に書き換えたのち JMMan. exe を作り直した。

○バージョン 2.00（2011/01/05）JMacs. exe，JMMan. exe，JMSlav. exe

英文字列を大文字の英文字列に変換するコマンド ESC-U が正しく動作しなかったのを修正した。また，半角空白コードを改行コードで置き換えるときに発生するエラーも修正した。改行への置換が，ページ最下行で行われるようなときに発生しているらしいので，調べたところコメント化していたプログラム

部分を生かすことにすれば問題が解決しそうであるので，そのようにした。なぜ，コメント化したのかは不明であり，これにより別の問題が発生する可能性があるかも分からないが，さしあたりの対処として行った。ただし，今のところ問題は発生していない。

○バージョン 2.01（2011/11/13）JMMan. exe

シフト JIS での文字コード 89F6 の文字は JMMan. exe 中で入力したときは正しく表示されているが，一旦ファイルに保存し，そのファイルを再度読み込むと F5F1 のコードになってしまうことを修正した。これは C プログラムの処理アルゴリズムのエラーではなく，内部で用いている Unicode での文字コードを保存するときにシフト JIS のコードに変換するための変換テーブルの記述にエラーがあったことによる。その変換テーブルのファイルを修正することにより解決した。

○バージョン 2.02（2011/11/14）JMMan. exe

文字コード入力による文字入力において，通常のシフト JIS，扱える文字として登録されている文字鏡文字，さらに満族語文字，のいずれにも該当しない文字コードの入力をチェックするようにした。JMMan. exe を用いて入力作業を主として行っているユーザが入力したファイルに，そのような文字コードのものが入力されていたので，チェック機能の必要性を感じて実現した。

○バージョン 2.01（2011/11/13）JMMan. exe

扱える文字鏡文字として 1 文字を，文字コード FB9A の位置に追加登録した。

Ⅲ．文字鏡文字の追加登録の作業手順の整理

まず，本システムにおける文字鏡文字の扱いについて説明しておく。文字鏡文字は現在 16 万文字を超える文字が表現できるようになっている（最初に文字鏡文字を採用するように決めたときは 10 万文字程度であった）が，その画面への表示には，文字フォントグループの番号と，そのグループ内の文字コー

ドを指定することにより行う。これらの情報のために，3バイトあるいは4バイトが必要である。したがって，文字鏡文字を本システムに取り込むときに，文字フォントグループの番号とグループ内の文字コードというようにすると，この文字のために3または4バイト必要ということになる。このことは，ファイルを編集するときにメモリに読み込んだときに，UTF-16BE の Unicode 文字として全文字2バイトのものとして扱うことに反する。なお，NEC PC-9800 シリーズ用に開発していた最初のころは，英数字は ASCII コードの1バイト，日本語文字は2バイトとしていた。この方法であれば，カーソルを逆方向に移動する場合，カーソル位置の前の文字が1バイトの文字か2バイトの文字かを判断して前に移動する。しかし，その判断をするには，その行の先頭からバイト数を判断しながら一つ前の文字位置を決めるか，あるいは現在の行に対して，その中の各文字の位置を管理しておき，その管理情報を用いて，戻る位置を決める方法のどちらかとする方法が考えられるが，後者を採用していた。どちらにしても面倒な処理が必要であった。そこで，Windows でデータ領域として利用できるメモリ領域が大きくなり，かつ Unicode が扱えるようになったので，内部で保持する文字コードは UTF-16BE を用いて，すべて2バイトとすることにしたという経緯がある。

文字鏡文字は非常に多くの文字を扱えるが，目的である満族語・日本語辞書に必要な文字鏡文字は3,000文字程度ということもあり，本システムで扱う文字鏡文字を，シフト JIS コード体系で用いていない文字コード領域，F100 から FFFD までに割り当てることにすれば，2バイト文字として扱えることになると考え，そのような方法を採用した。F000 から F0FF までは満族語文字に用いているので，文字鏡文字には F100 からの割り当てとした。最初に本システムを開発したときは，文字鏡文字には F000 から F9FF まで，満族語文字用には FA00 から FAFF までは半角文字と同じ横幅の小さい文字，FB00 から FBFF までは全角と同じ横幅の大きい文字としていた。これは，当初は半角文字の大きさは小文字，全角文字の大きさは大文字と考えていたためであるが，満族語文字にははっきりした大文字小文字の対応がないことと，文字鏡文字の登録数

が増えてきたことから，上記のような文字コードに変更した［本田 2008］。この範囲であれば今後の満族文字の追加が多くなっても対応できる。そして，扱う文字鏡文字が生じれば，それに F100 から順に文字コードを割り当てることにした。その割り当てた文字コードの文字を表示するときには，その文字コードに対してあらかじめ求めていた文字鏡のフォントグループの番号とそのグループ内の文字コードを用いて表示するようにした。

このようなことから，必要になるたびに文字鏡文字に対して，本システムでの文字コードを割り当てるという作業が必要であり，さらにそれに伴って以下のようなプログラムの変更なども必要になってくる。

(a) F100 から FFFD に割り当てた文字鏡文字を表示するために必要な文字鏡のフォントグループとグループ内の文字コードの対応表（JmMl2Mj. tbl）の再作成を行う必要がある。このファイルは，編集サブシステム，および印刷サブシステムの両方の C のソースプログラム中に読み込まれるものとして共通したものである。

(b) 編集サブシステム，および印刷サブシステムの C のソースプログラム中で文字鏡文字に関係する部分，特に登録されている文字鏡文字の文字コード範囲を記している部分の変更。

(c) 追加登録する文字鏡文字によっては，表示するために関数 CreateFont() による必要な新しい文字鏡フォントグループのフォント作成が必要となる場合がある。その場合には，両システムとも，その C のソースプログラムのフォント作成部分の変更（追加）が必要である。最近は，既に多くの文字鏡文字が登録されているために，新しい文字鏡文字を追加登録するときに，新しい文字鏡文字フォントグループのフォント作成が必要なことはあまりなくなっているが，頻繁ではなくときどき文字鏡文字を追加登録することになった頃は，このことを忘れていたために正しい文字表示ができずに，原因を突き止めるのにかなり時間がかかることもあった。

これら(a)，(b)，(c)を行ったあと，編集サブシステムおよび印刷サブシステム

のCプログラムのコンパイル・リンク（以後,「ビルド」という）を行い，JMMan. exe および PPMMan. exe を作成する。

また，文字鏡文字の追加登録に伴い，上記のプログラムに関する変更の作業だけでなく，編集サブシステムで文字鏡文字を入力するときのために，文字鏡文字の本システムにおける文字コード順と画数順の一覧表の作成などの作業も行う。これらの表は見やすさを考えて，Tex で処理して印刷できるものとして作成するために，Tex 用のファイルの作成を行っている。これらの文字鏡文字を追加登録する作業手順を以下に説明する。なお，以下では，ファイル名が説明の文と区別しにくい場合は「ファイル名」のように鍵括弧で囲むこともある。

【手順１】

満族語・日本語辞書作成のために，編集サブシステムを用いて入力を行っているユーザが，追加登録する文字鏡文字が必要となったときに，ファイル「JM コード対照表. jmm」の最後の方の未使用コードの部分（JM-Code は，記入済み）に，追加したい文字鏡文字について，画数，本システムでの文字コード（JM-Code），Unicode，文字鏡文字としての文字番号（文字鏡-Code）を調べて記入する。このファイルの形式は以下に示すようなものである。「漢字」の欄の下は，最終的には追加登録する文字鏡文字が表示されるが，この時点ではまだ表示できないので，括弧内に画数を記しただけのものを作成する。Unicode は分からないときは，０としておいてもよい。

漢字	JM-Code	Unicode	文字鏡-Code	備考
(12)	FB9A	9ED1	48038	// 2011/11/13 追加

【手順２】

以下の（2-1）～（2-3）を順に実行して Unicode 部分が 0 である行に対して，Unicode を求めて記入する。ただし，プログラム 02 TblAddUni. c 作成に際して参考にした「Unicode－Shift-JIS－文字鏡（大漢和）」に記載されていたものは正しいようではあるが，漏れがあり完全ではないようである。そこで，別の手

段で対応する Unicode が正しく分かるのであれば，それを優先することにしている。

(2-1) 02 TblAddUni. c をビルドする。ただし，「A>」の部分はプロンプトであり，開発フォルダに依存した表示がなされる（以下，同様)。

```
A>cl /J 02TblAddUni.c
```

(2-2) 02TblAddUni. exe を実行し，「JM コード対照表. jmm」の文字鏡文字に対する Unicode があれば，それを Unicode 欄に記入したものを ZAA に作成する（ZAA はプログラム中で指定)。

```
A>02TblAddUni
```

(2-3)「JM コード対照表. jmm」を Old フォルダに移動し，ZAA の Unicode が正しいことを確認し，「JM コード対照表. jmm」にコピーし，保存する。

【手順３】

03MlSort. exe を実行し，ファイル ZAA の各行を本システムでの文字コードの順に並べた出力をファイル ZBB に作成する。このためのコマンドは

```
A>03MlSort /＋13 ZAA ZBB
```

とする。「＋13」は，ZAA の各行を比較するのに 13 カラム目から行うという指定である。13 カラム目から本システムでの文字コード，すなわち JM-Code が開始している。

【手順４】

ファイル ZBB を現在の JMMan. exe で開き，ファイルの最初と最後にある不要な行（複数）を削除する。現在のという意味は，ここでの作業により，次のバージョンの JMMan. exe を作成するからである。「JM コード対照表. jmm」には，手順１での追加登録の記入を行いやすいように，本システムでの文字コー

ドだけで，その右の対応する Unicode と文字鏡文字が 0 となっている部分がある。今後の利用のためであり，まだ利用していない部分を，その右が「0　0」であっても FxFF の行まで残す。この「FxFF まで残す」とは，次のような意味である。たとえば，現時点で最後に追加したものは，本システムでの文字コード（JM-Code）は FB9A であるが，この場合，FB9B，FB9C，…，FBFF までは右側が「0 0」のままで，記入されていなくても残すということである。ファイル ZBB は次のようになっている。

```
芙（ 7） F100   0   72229
肩（ 8） F101   0   29346
杭（ 8） F102   0   14547
罕（ 8） F103   0   28215
  :      :     :     :
```

【手順 5】

ファイル ZBB をファイル ZCC にコピーする。ZBB は手順 11 で満族語・日本語辞書用のデータファイルを Tex 用のファイルに変換するプログラムで用いるので変更せずに置いておくためである。

【手順 6】

ファイル ZCC を次の（6-1），（6-2）の手順により，各行が「本システムでの文字コード　Unicode」のように変更する。

（6-1）JMacs. exe の長方形削除の機能を用いて，最初の「/*」と空白の部分を除き，本システムでも文字コードが先頭から始まるようにする（手順 7 のため）。

（6-2）JMacs. exe のキーボードマクロを用いて Unicode より右の部分を削除する。ファイル ZCC は次のようになっている（ただし，F100 の F が第 1 カラムである）。

```
F100        0
F101        0
 :          :
F11D        0
F11E     9D19
F11F        0
F120     5201
F121     4E48
 :          :
```

【手順７】

通常のソートプログラム（Sort. exe）を次のように用いて，ファイル ZCC を Unicode の順にソートし，結果をファイル ZDD にとる。

```
A>Sort /+8 <ZCC >ZDD
```

「/+8」の部分は ZCC 内での Unicode の開始カラム位置である（実行前に確認しておく）。ZCC において本システムでの文字コードが第１カラムからとなっているときは，「/+8」でよい。ここに現れるほとんどの文字の Unicode は４桁であり，その場合は９カラム目から始まるが，何個かは５桁のものがあるので「/+8」とする。

【手順８】

JMacs. exe でファイル ZDD を開き，先頭部分の Unicode 部分が０の行をすべて除いたあと，キーボードマクロを用いて，各行を次のように変更した後，ファイルの先頭に１行，最後に２行追加する。

```
WCHAR JmUni2M1Tbl[][2] = { <==追加
   {0xF93F, 0x34C2},          <== {ML 文字コード, Unicode}
   {0xF2A8, 0x34D4},
        :
   {      0,      0}          <==最後に入れる：追加
};                             <==；を忘れずに！：追加
```

【手順９】

09UniSJChk. c（ファイル ZDD を include している）をビルドしたあと，実行して ZDD の Unicode のうちシフト JIS コードに存在するものをチェックする。手順６から手順９までは，本システムに追加登録した文字鏡文字が，シフト JIS の文字として存在するものであるかどうかをチェックするためである。文字鏡文字について，直接シフト JIS の文字として存在するか調べる方法は現在のところないので，Unicode 文字を介して，文字鏡文字と Unicode の対応，Unicode 文字とシフト JIS 文字の対応ということでチェックしている。

09UniSJChk. exe でチェックしたときに，シフト JIS に存在するとしてコマンドプロンプト画面（DOS 窓）に表示されるものに対して，以下のことを行う。

・「S-JIS:Fxxx」のもの（F で始まる Fxxx のもの）は Shift-JIS での拡張文字であり，本システムで用いる満族語文字や文字鏡文字と文字コードが重なるので，変換テーブル ZDD に残す。
・重複しないものは，Shift-JIS を用いればよいので，手順１のファイル「JM コード対照表. jmm」からその文字コードの行は除き，手順２から続ける。なお，「JM コード対照表. jmm」と ZDD に残しておいてもよいが，その場合はシフト JIS にある文字に対して文字鏡文字を用いることになる。

09UniSJChk. exe でチェックして文字コードの先頭が F でない文字で，シフト JIS に存在すると分かった文字は，あえて文字鏡文字を用いる必要はなく，

ファイル「JM コード対照表. jmm」に入れる必要はないので，削除すればよい。しかし，既に辞書用のデータファイルに入力されているものであれば，削除すれば，データファイルの表示ができなくなるので，ファイル「JM コード対照表. jmm」からの削除と，データファイルの変更を同時に行う必要があるので，ここでは既に登録されている文字については削除は行わず，今回新たに追加登録する文字について，シフト JIS に文字があるのであれば，それは削除し，その旨を，文字鏡文字の追加登録を依頼してきたユーザに伝えることにしている。これは，文字鏡文字の追加登録の依頼は，シフト JIS の文字として存在しないものだけだと思っていたが，あるとき，必ずしもそうでなく，シフト JIS に存在しているのを見逃していることがあることに気がついたためである。登録したものを既に用いているものもあるので，新たに追加登録するものについてのみ，追加登録しないようにした。文字鏡文字を利用するようにした最初から 09UniSJChk. c を作成してシフト JIS にあるかどうかをチェックして，あるものであればシフト JIS の文字を用いればよいので登録しないとすればよかったが，そのようなことは想定していなかったので，あとからの開発となった。

【手順 10】

ファイル ZDD を JmUni2Ml. TBL に名前変更して編集サブシステム（JMMan. exe）開発用のフォルダに移動する。JmUni2Ml. TBl は編集サブシステムの C プログラムのソースファイル（Jm. c）中に，ビルド時に読み込まれるようになっている。

【手順 11】

ファイル ZBB を，本編集システム（JMMan. exe）で開き，Ml 2 Mj 例. TBL を参考に以下のように変更する。ただし，最初の行の先頭の UINT の U の位置が各行の先頭位置，すなわち第 1 カラムである。「/*」と「*/」に挟まれた本システムでの文字コード（ML-Code）と Unicode の部分は，C プログラムとしてはコメントであり，削除してもよいが，右端の文字鏡文字番号との対応確認のために残している。

```
UINT MlMjTbl[] = {
  // MLコード    Unicode    文字鏡文字番号,    備考
  /* F100          0 */        72229,
  /* F101          0 */        29346,
  /* F102          0 */        14547,
       :            :           :
  /* F123       5300 */         2497,        //= F920
       :            :           :
  /* FBFF          0 */           0,
};
```

このように変更するには，編集サブシステム JMMan. exe の長方形削除機能と，キーボードマクロ機能を用いれば簡単である。

【手順 12】

12Ml2Mj. c（ファイル ZBB を include）をビルド・実行し，リダイレクションで，その出力を ZEE にとる。このプログラムは，文字鏡文字番号から，編集サブシステムのプログラム JM.c で用いる「文字フォント名＋文字番号」の情報を作成するものである。ファイル ZEE の内容は次のようなものである。

```
EM_MLMJ EmMlMjFl[] = {
   {/* F100: 072229 */ 113, 0xE1C3},
   {/* F101: 029346 */ 106, 0x8EB0},
   {/* F102: 014547 */ 103, 0x9AE5},
   {/* F103: 028215 */ 106, 0x88AD},
        :       :        :      :
   {/* FBFF: 000000 */ 112, 0x88DC},
};
```

【手順 13】

ZEE を JmMl2Mj. TBL に名前変更し，編集サブシステム（JMMan. exe）開発用のフォルダに移す。なお，ファイル JmMl2Mj. TBL は，C のソースプログラムである JM. c 中に include されており，ビルド時に読み込まれる。その後，JM. c の文字鏡文字の最終コードを記している部分

```
#define EM_MOJCHARX1BGN 0xFB9B  // 文字鏡文字範囲外 1 開始
```

の最後の 16 進数で記されている部分（上記では 0xFB9B の部分）を適切に変更したのち，JM. c をビルドして，追加登録した文字鏡文字が入力・表示できる新バージョンの編集サブシステム（JMMan. exe）を作成する。

【手順 14】

14MjTexTblMk. c（ZBB を include している）をビルド・実行し，リダイレクションで出力を MjTbl. tex（Tex 用の文字鏡一覧表）にとる。ただし，たとえば，今回の文字鏡文字の追加登録により，本システムでの文字コードが，FAFF を超えて，FB00 以上になったような場合には，次のように，ファイル 14MjTexTblMk. c　中の

```
MjTexTblMk('N', 0xFA00, 0xFAFF);
```

の直後に，

```
MjTexTblMk('N', 0xFB00, 0xFBFF);
```

の行を追加したあと，ビルドする。

【手順 15】

JmMojikyo. tex（MjTbl. tex を include）を Tex で処理し，JmMojikyo. dvi を作成し，確認する。確認のあと，ファイル名を「文字鏡文字コード表. dvi」に変更する。このファイルは次のように表示・印刷されるものである。

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
F100	羙	肴	杬	罕	玦	彨	傍	看	晌	挃	結	涵	愔	㺄	紓	飥
F110	啗	滾	挈	硜	罨	虧	鞊	鳲	橄	獖	䐬	緵	誫	䑑	鴂	觥
F120	刁	么	丫	匀	卡	另	夯	奶	扒	氐	犰	忖	丟	伙	仵	仿
F130	划	刖	吣	踏	鶤	妃	鷉	鑲	汉	犯	犳	空	佔	你	刨	吧
F140	囤	囫	妗	坍	弝	忐	扭	扯	忑	杈	杌	沉	牤	犽	肐	虸
F150	虬	阧	您	盼	汇	滙	宷	寘	脊	岦	岺	崧	嵂	嵭	嶸	嶹
F160	巋	弫	弴	彅	彧	忞	恝	悊	惕	愀	惲	愑	愰	愷	憘	抦
F170	拼	揵	涒	侉	兕	剁	剳	聊	呢	呲	哂	咕	坫	坯	妯	宺
F180	屉	帔	幷	怐	怵	怔	戕	抿	拖	杭	枚	杻	泎	沬	泔	沱
F190	炁	炕	炖	狍	狕	狖	玫	疙	秈	芤	芰	芡	衩	釭	沓	佶
F1A0	俏	咧	咵	咱	哆	垛	垡	姤	帢	弡	挂	拴	挎	昬	柯	相
F1B0	柷	毡	洼	洿	狟	狥	狢	狨	狪	玷	祇	祉	畎	砍	砑	疤
F1C0	籽	紁	紉	胗	茈	紆	鵚	靏	丧	潔	刅	咺	喋	嚆	墩	帽
F1D0	凢	丰	汋	邛	抗	壳	吽	扳	灶	姊	帔	戾	衲	芩	响	帲
F1E0	孩	羿	叚	盼	昐	袉	迨	痧	耔	耇	倮	倏	哼	捄	涂	帨
F1F0	袪	荔	逄	玼	胥	虒	荆	陡	啚	兗	減	埤	抪	掮	挏	帿

【手順 16】

dvipdfm. exe で「文字鏡文字コード表. pdf」を作成し確認する。なお，現在 PDF は作成できないこともある。憶測であるが，フリーで利用できるとして開発されていた文字鏡文字であるが，ある時期から管理者が代わったようであり，それ以降に追加された文字鏡文字によっては PDF に変換できないようである。また，PDF に変換できるが，ところどころ表示が変になることもある。したがって，PFD ファイルが作成できた場合もチェックが必要である。

【手順 17】

ファイル「文字鏡コード表. dvi」を適切なフォルダにコピーする。このファイルは Windows システムをインストールして，その上に，Tex の処理系を入れた場合に，確認用として用いる。

【手順 18】

18MjJmTblMk. c（ZBB を include している）をビルド・実行し，「JmMojikyo. jmm」を作成・確認する。このファイル「JmMojikyo.jmm」は，Windows システムをインストールして，その上に，本システムを入れた場合に，確認用として用いる。なお，文字が増えて FxFF を超えた場合には，14MjTexTblMk. c に

```
MjTblMk(fpo, 0xFB00, 0xFBFF);
fputc(0x0D, fpo); fputc(0x0A, fpo);
fputc(0x0D, fpo); fputc(0x0A, fpo);
```

のように，適切に 3 行を追加したあと，ビルドすること。

【手順 19】

ファイル 19 TblAddMlChar.c を次のように，ビルド・実行し，ファイル ZAA からファイル ZFF を作成する。

```
A>CL /J 19TblAddMlChar.c
A>19TblAddMlChar ZAA ZFF
```

手順 2 で作成したファイル ZAA には，今回追加登録した文字鏡文字については，その行の先頭部分の漢字の位置にまだ漢字が入力されていないので，このプログラムで，その行の右側で，16 進数の英数字で記された本システムでの文字コードに応じた漢字を入れている。その後,手順 13 で新たに作成した編集サブシステムを用いて，今回追加登録した文字鏡文字の行を画数に応じた適切な位置に移動させた後，ファイル名を，「JM コード対照表. jmm」に変更する。

【手順 20】

20Tbl2Tex. c を次のようにビルド・実行して，ZFF から Tex 用のファイル ZGG を作成したのち，ファイル「JmMjkKakusuu. tex」を Tex で処理し，文字鏡文字の本システムでの文字コードの画数順の表である TblKakusuu. dvi を作成し，さらにファイル名を「文字鏡コード画数表. dvi」に変更する。

```
A>CL /J 20Tbl2Tex.c
A>20Tbl2Tex ZFF ZGG
A>pLatex JmMjkKakusuu
```

なお，ファイル ZGG は次のような形式の内容であり，このファイルは「JmMjkKakusuu. tex」中に「¥input｛ZGG｝」として，「JmMjkKakusuu. tex」の Tex での処理時に読み込まれる。¥TMO ｛000121｝は文字鏡文字を Tex での処理用に変換するためのマクロであり，¥tt｛F 121｝は F 121 という文字列を等幅のタイプライタフォントとするという指定の Tex 用のコマンドである。

```
漢字          & JM-Code   & Uni-Code  & 文字鏡-Code  備考    ¥¥ ¥hline
¥TMO｛000121｝（3）& ¥tt｛F121｝ & ¥tt｛4E48｝ & ¥tt｛  121｝   & ¥tt｛｝ ¥¥ ¥hline
¥TMO｛000071｝（3）& ¥tt｛F122｝ & ¥tt｛4E2B｝ & ¥tt｛   71｝   & ¥tt｛｝ ¥¥ ¥hline
      :       &     :     &     :     &     :                ¥¥ ¥hline
```

ファイル「文字鏡コード画数表. dvi」は次のように表示・印刷されるものである。

漢字	JM-Code	Uni-Code	文字鏡-Code	備考
刁（2）	F120	5201	1846	

漢字	JM-Code	Uni-Code	文字鏡-Code	備考
么（3）	F121	4E48	121	
丫（3）	F122	4E2B	71	
亢（3）	F1D0	51E2	1740	
凡（3）	F4F0	0	76734	
从（3）	F51F	4E46	119	
囚（3）	F8E8	5166	1417	
义（3）	F9E2	4E49	124	
义（3）	FA32	4E49	124	//F9E2 と重複
叉（3）	FA83	53C9	3116	
犭（3）	FA99	72AD	20235	
扌（3）	FAC9	624C	11770	
氵（3）	FAFB	6C35	17085	
扌（3）	FB1E	624C	11770	//FAC9 と重複
艹（3）	FB7E	8279	54448	

【手順 21】

本システムのユーザに，JMMan. exe，文字鏡コード表. dvi，文字鏡画数表. dvi JmMojikyo. jmm を送付する。

【手順 22】

PPMX. exe をビルドし，さらに確認した後，本システムのユーザに送る。C のソースファイルである PPMX. c 中で，手順 13 で作成し，JMMan. exe 開発用のフォルダに移した JmMl2Mj. tbl を＃include 文で読み込むようになっている。

Ⅳ．終わりに

満族語・日本語の辞書作成を補助するための本システムの開発は，はっきりとは覚えていないが 1994 年あたりからであったと思う。そのころは，満族語の文字，あるいはシフト JIS には入っていない漢字などをパソコンで扱えるような市販あるいはフリーのソフトウエアがなく，辞書作成を考えている研究者から，相談を受けたのがはじまりであった。当時は，パソコンは NEC の PC-9800 シリーズ，OS は MS-DOS で，しかも編集サブシステムのプログラムは，アセンブリ言語で記述したものであった。そのために，満族語文字は文字フォントを自前で作成したものをユーザ登録文字として扱い，また，シフト JIS にない漢字についても同様に文字フォントを自前で作成しユーザ登録文字として扱うことで始めた。しかし，ユーザ登録文字として扱える文字数 256 よりも，本システムで必要な 3,000 程度の満族語文字および漢字の数が多く，それをどうして処理するかも難題であったが，それ以前の「言語学研究へのパーソナルコンピュータの応用」［本田 1990］，「スラブ系・ラテン系の言語研究のための基礎システム」［本田 1991］で用いていた方法を利用するなどで解決することにしていた。そのうちに，パソコンも DOS/V 機で OS は Windows となり，Windows も Windows 95，Windows 98 などを経て Windows 7 と変遷してきた。その間に，プログラムもアセンブリ言語で書いていたものを，言語 C でのプログラムとして書き換えた。なお，Windows がバージョンアップしたときに，Windows の機能を呼び出して処理していたものが動作しなくなるなど

のトラブルもあった。ただし，Windows になり，当初のシステム作成時に苦労していた通常文字以外の文字の表示のことは簡単に処理できるようになった。さらに，Winodws の上の TrueType フォントとして，シフト JIS 以外の漢字を 10 万文字程度含む文字鏡文字というものがフリーで提供されていることを知り，それを利用させていただくことにし，漢字のフォント作成の必要もなくなった。

このように，本システムの開発はパソコンや OS の変遷に伴っての変更や，プログラムそのものもアセンブリ言語で書いていたものを，言語 C で書き直すなどの変更を重ねながら，かなりの長期間に渡って行われてきているが，それは，辞書用のデータが非常に大きなものであり，しかも，最初は複数人数であったようであるが，途中からは一人で入力してきたために時間がかかり，その途中で文字鏡文字の追加登録やエラーの発見などがあり，そのたびに対応してきたためである。

最近になって，やっと満族語・日本語辞書の完成が視野にはいってきた。現在，すでにデータ入力は終了しており，第１回目の校正中であるとのことである。正確を期すために３回ほど校正をするとのことである。なお，辞書データはいくつかのファイルに分けて作成している。これはデータ入力にいくつかの資料を参考にしていたため，資料ごとに分けて複数人で入力していたことと，最初にシステムを開発したときの OS が MS-DOS で，プログラムで扱えるメモリ内のデータの大きさがかなり小さなものに制限されていたためである。

データの入力・校正の作業の終了のあと，辞書完成までには，さらに次のような作業が必要である。

⑴　それら複数のファイルを１つのファイルにをまとめる。

⑵　さらに各単語の記述をローマ字表現の部分をキーにして辞書式順序に並べ変える。

⑶　それを辞書としての形式で表現できるように Tex 用のファイルに変換する。

⑷　Tex で処理して，辞書としての形式のものを出力する。

⑸　その出力したものを元に，印刷・製本して辞書として完成する。

このうち，⑴から⑷が筆者の作業であり，⑸は印刷会社の作業となる。⑴から⑷の作業のためのプログラムはテスト的なものは作成しているが，まだ完成はしていない。いずれにせよ 2012 年中には完成できるものと期待している。

また，満族語・日本語の辞書が完成した後，上記⑴でまとめた同じファイルを，日本語の読みの部分をキーにして並べ替えて，日本語・満族語の辞書を作成する予定である。並べ替えたあと，多少の手作業は必要であろうが，比較的速く辞書を完成させることができると考えている。

参考文献

［シルト 2000］シルト，ハーバード「Windows 2000 プログラミング標準講座」翔泳社，2000 年
［本田 1990］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
［本田 1991］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
［本田 1995］本田道夫，今井慈郎「日本語・満州語の辞書作成のためのシステム（Ⅰ）」香川大学経済論叢第 67 巻第 3・4 号
［本田 1998］本田道夫「日本語・満州語の辞書作成のためのシステム（Ⅱ）」香川大学経済論叢第 71 巻第 3 号
［本田 2003］本田道夫「日本語・満州語の辞書作成のためのシステム（Ⅲ）」香川大学経済論叢第 76 巻第 1 号
［本田 2006］本田道夫「日本語・満州語の辞書作成のためのシステム（Ⅳ）」香川大学経済論叢第 78 巻第 4 号
［本田 2006］本田道夫「日本語・満族語の辞書作成のためのシステム（Ⅴ）」香川大学経済学部研究年報 46
［本田 2008］本田道夫「日本語・満族語の辞書作成のためのシステム（Ⅵ）」香川大学経済学部研究年報 48