香川大学経済論叢
第76巻 第1号 2003年5月 59−87

日本語・満州語の辞書作成のための補助システム（Ⅲ）

本田道夫

Ⅰ はじめに

日本語・満州語および満州語・日本語の辞書作成のための補助システムは，従来は日本電気のPC-9801シリーズのパソコンで，マイクロソフトのMS-DOS上，あるいはWindows 95，Windows 98上でDOS窓を利用する環境でのみ利用可能であった。しかし，次節以降で述べるように利用できるパソコンの機種，OSなどの状況の変化に対応して，DOS/V互換機上のWindows 95，Windows 98，Windows Me，Windows NT，Windows 2000，Windows XPなどの上でWin 32 APIを利用したシステムを開発した。本論文では，その開発に当たっての手法・方法について説明する。

日本語・満州語および満州語・日本語の辞書作成について相談を受け，そのための補助システムを開発し始めたのが，1993年であり，それ以降，辞書作成が具体的に進むに従って，辞書作成者と相談しながら，いろいろな機能を実現してきた［本田 1995］［本田 1998］。

開発開始時の1993年当時は，パソコンのOSはマイクロソフトのMS-DOS

であり，パソコンの機種としては日本電気のPC-9801シリーズの利用者が多く，辞書作成者も筆者も，そのようなシステムを利用していた。したがって，本補助システムも，日本電気のPC-9801シリーズのパソコンで，OSはMS-DOSということで開発を始めた。英文字や日本語の漢字などの通常の文字以外に，満州文字と通常には用意されていない漢字（以後，拡張漢字という）を扱う必要があったが,その入力と表示については,日本電気のPC-9801シリーズ固有のハードウエアに依存した方法を採用していたが,特に問題はなかった。

その後，マイクロソフトのWindows 95が販売され，パソコンの利用もグラフィカル・ユーザ・インターフェース（GUI）となり，利用者もMS-DOSよりはWindowsの方が便利に操作できるため，パソコンの主流のOSはマイクロソフトのMS-DOSからWindows 95さらにWindows 98に移った。ただし，本補助システムは，ハードウエアが日本電気のPC-9801シリーズであれば，Windows 95，Windows 98とも，DOS窓を利用すればMS-DOS上でと同様に利用できた。

しかし，日本電気の主力の製品もPC-9801シリーズから，いわゆるDOS/V互換機的なものへと移り，そちらの機種の方が処理速度の速いCPUを搭載したものが多く発表されるようになり，PC-9801シリーズのハードウエア固有の機能を利用していた本補助システムを利用するためのパソコンの選択肢も狭くなってきた。他方で，日本電気のパソコンにこだわらなければDOS/V機の選択は広くなってきた。

また，DOS窓での利用の場合，基本的には操作はMS-DOSでの操作と同じであり，したがってWindowsでのGUIを利用した便利なものに慣れると，DOS窓での操作では不便に感じることもあった。そこで，1999年4月頃から，本補助システムも，日本電気のPC-9801のハードウエア固有の機能を利用した同シリーズだけで利用できるものではなく，他社のDOS/V互換機でも利用できるように，Windowsに対応したものの開発を開始し，1999年7月に新システムとして利用できる状態になった。その後も実際に利用しながら改良を進めてきたが，現時点で辞書作成の作業もある程度見通しがつき，したがっ

て，本システムもある程度は固まったものとなったこともあり，ここでいったん報告することにした。

Ⅱ　新システムにおける文字フォントおよび文字コード

2.1　文字フォント

2.1.1　満州文字フォント

新補助システムはWindows上に開発するので，辞書作成ではTruetypeフォントを用いるのが自然である。もちろん，旧システムで用いていた半角文字，全角文字の2つの固定サイズのドット・イメージのフォントをそのまま用いることも選択肢の一つにはあった。しかし，本来の満州文字は，半角サイズと全角サイズの2種類だけの大きさの文字から構成されるものではなく，その間の大きさの文字，通常の全角サイズよりも大きな文字など，いろいろな大きさの文字から構成されるものである。Truetypeフォントとすることにより，そのような本来のサイズの文字として扱うことができるという利点があるので，本システムで用いるフォントは，Truetypeフォントとすることにした。なお，旧システムのドット・イメージのフォントの場合，表示に用いる半角文字8＊16ドット，全角文字16＊16ドットで構成されるフォントと，印刷で用いる半角文字12＊24ドット，全角文字24＊24ドットの2種類のフォント（4ファイル)を作成・利用していたが,本システムでも，満州文字については固定サイズのフォントと本来の適切な文字サイズのフォントの2種類のTruetypeフォント（2ファイル）を用意している。固定サイズのフォントは，辞書データ作成作業で用いる編集機能のエディタの開発を簡単にするために，画面表示用に用意したものである。つまり，画面では，固定サイズのフォントを用い，印刷および画面でのプレビュー（TexとDVIOUT）においては，文字毎に適切な大きさで表示・印刷できるフォントを用いている。ただし，TexとDVIOUTでは，満州文字はTruetypeから作成したPKフォントを用いている。なお，拡張漢字については，漢字であるので固定サイズの文字として作成されるので，画面と印刷の両方で同じものを用いている。

Windows 上での新システムのためには，満州文字と拡張漢字のフォントを Truetype フォントとして用意する必要があった。旧システムの場合，ドット・イメージのフォントを作成するためのプログラムで適切なものがなかったので筆者が開発したが，Truetype フォントの場合は，プログラムを開発するために必要な Truetype の仕様の情報が入手できなかった。また，既にシェアウエアあるいは製品としてフォント作成プログラムが存在していたこともあり，既存のソフトウエアを利用する方向で，Truetype フォント作成用のソフトウエアを何種類か検討・試用した。しかし，ドット・イメージでフォントを作成することと比べて，Truetype フォントを作成することは，どのソフトウエアを用いても，容易なことではなかった。

一方，全く何も無い白紙の状態から文字を作成するのではなく，字形としては十分なものではないにしても，元になる Truetype フォントがあれば，フォント作成ソフトウエアを用いて修正することにより，文字作成はかなり容易になることも分かった。そのために，満州文字の元となったというモンゴル文字の Truetype フォントを入手して，それを元に満州文字の Truetype フォントを作成する方法も調べた。このフォントは，東北大学の文学部だと思われるところで立ち上げていたホームページ（http://www.ling.is.tohoku.ac.jp/）で提供されていたものであり，それを元にして満州文字を作成する許可も得ていた。なお，現在，そのホームページは無くなっているようである。しかし，実際に作業を開始すると，モンゴル文字と満州文字は字数も異なり，満州文字を作成する元となるような文字も少なく，実際に利用することにはならなかった。さらに，いろいろな方法を調べたが最終的には，旧システムで利用していたドット・イメージでのフォントを Truetype に変換する方法をとることとした。この方法については，2.1.3 で説明する。

2.1.2　拡張漢字フォント

Truetype フォントの作成について調べている途中で，文字鏡研究会（http://www.mojikyo.org/html/index.html）という組織から，8 万字以上の漢

字を含む今昔文字鏡という文字フォントのセットが無償で提供されており，本システムでの拡張漢字の多くが含まれていることがわかった。さらに，文字の字形とその出典が明らかとなる資料を，文字鏡研究会に提示して文字の作成を依頼すれば，その文字を作成して文字鏡のフォント・セットに追加してくれることも分かった。したがって，拡張漢字の Truetype フォントについては，全面的に文字鏡研究会にお願いすることとした（現時点では，文字鏡以外にも，東京大学多国語処理研究会（http://www.l.u-tokyo.ac.jp/GT/）が提供する同様な文字フォントセット GT 書体も公開されているが，本システムでは，今昔文字鏡のフォントを利用している）。

2.1.3 ドット・イメージのフォントから Truetype フォントへの変換

2.1.1 で述べたように，いろいろな方法を調べたが，不十分な形の Truetype フォントであっても，何もないところから Truetype フォントを作成するよりも，それを修正することの方がかなり容易であったので，変換の方法を調べた。そして，以下の手順で，旧システムのドット・イメージのフォントから，新システムの元となる字形の Truetype フォントを作成することができた。この作業は，とにかく一度だけ作業を行えばよいことであるので，もっと良い方法があるかどうかは検討していない。

(1) ドット・イメージのファイルから BMP 形式のファイルを作成する。

旧システムでは，印刷用の満州文字は，半角文字の場合 12＊24 ドット，全角文字は 24＊24 ドットでの構成であったが，それを単純にそれぞれ 96＊192（半角文字），あるいは 192＊192（全角文字）に拡大した字形の BMP 形式の文字毎のファイルに変換する。このための，プログラムは筆者が作成した。なお，BMP 形式のファイルの場合，ファイルの先頭部分に，図形に関する情報が記されているが，その情報の十分な資料が入手できなかったので，Windows 付属ソフトの「Paint」を用いて，テスト用の文字フォントを作成し，そのファイルの先頭部分を解析して用いた。

(2) BMP 形式のファイルから Truetype フォントのファイルに変換する。

文字毎のBMP形式のファイルを，TTEDITというプログラムで読み込み，Truetypeフォントに変換してファイルに保存する。TTEDITはシェアウエアのソフトウエアであるが，この作業だけの比較的短時間での利用であったので，作者には申し訳ないが無料で利用させていただいたままである。

(3) 文字毎のTruetypeファイルを1つのファイルにまとめる。

Font Creator Program (FCP 2) を2つ起動しておいて，一方のFCP 2でTTEDITで作成した1文字毎のファイルを読み込み，コード位置に注意しながら，「コピー」と「貼り付け」を利用して，他方のFCP 2に全角・半角の満州文字全部含むものを作成し，ファイルに保存する。ドット・イメージでのフォントの場合に半角満州文字用と全角満州文字用の2つのファイルであったものを，Truetypeフォントでは1つのフォントにまとめた(つまり，全角・半角および画面用・印刷用と4つであったファイルを，画面用と印刷用の2つにした)。

FCP 2には，BMP形式のファイルを読み込んでTruetypeフォントに変換する機能もあるが，全角・半角の満州文字で，自動的にサイズが変更されるということが生じたので，Truetypeフォントへの変換にはTTEDITを用いた。なお，FCP 2もシェアウエアであったが，これも一時的に用いるだけであったので，無料で利用させていただいた。このソフトウエアは，筆者の記憶では，「窓の杜 (http://www.forest.impress.co.jp/)」に掲載されており，ダウンロードしたが，現在は掲載されていないようである。

なお，満州語文字については，各文字のデザインで文字枠（これは文字毎に設定する）を文字の左右の端までとしておけば，1単語中の文字と文字は繋がって表示された。

2.1.4 Truetypeから Tex用の PKフォント・ファイルの作成

最終的な印刷ではTexシステムを用いることにしている。最近のTexおよびDVIOUTでは，Truetypeフォントを用いることができるし，実際に文字

鏡の文字（以後，文字鏡文字という）については，Truetype フォントを用いた印刷ができている。しかし，上記のようにして作成した満州文字の Truetype フォントについては，メトリック情報の入った TFM ファイルは TTF2TFM.EXE プログラムで作成でき，Tex での処理は可能であるが，DVIOUT で直接 Truetype フォントを用いて印刷することはまだ成功していない（現在も調査中である）。そこで，現在は，TTF2PK.EXE プログラムを用いて Truetype フォントのファイルから Tex 用の PK フォント・ファイルを作成する方法をとっている。この方法で作成した TFM ファイルと PK ファイルを用いた印刷で，1 単語中の文字と文字は繋がって印刷された。

なお，この方法で PK フォントを作成した場合，Truetype フォントは，16 進数表現で 21～9D（半角満州文字）と A1～DE（全角満州文字）であるが，作成された PK フォントでは，01～7D（半角満州文字）と 81～BE（全角満州文字）になっている。とにかくまず利用できればよいので，文字コードを一致させる方法については調べていない。

2.2　文字コード

Truetype フォントを用いる場合，単一の文字フォントだけではなく，いろいろな文字フォントを混在した文書としての情報も入れられる RTF 方式を採用する方法もあるが，本システムでは，RTF 方式はとっていない。

本システムを用いての辞書作成は，満州語・日本語辞書，および日本語・満州語辞書の両方を作成するための元となるデータファイルを作成し，そのデータファイルからそれぞれの辞書として印刷するための Tex 形式のファイルをプログラムで作成し，さらに Tex システムで処理し，印刷するというものである。そして，最も多くの時間を費やす作業は，元となるデータファイルを作成する作業であり，この作業では，通常の文字フォント以外に必要なものは，満州文字フォントと拡張漢字のフォントの 2 種類だけである。また，MS-DOS のデフォルトの日本語文字コードである Shift-JIS コード，および拡張文字コード領域は，Windows にもそのまま引き継がれているので，従来と同様な拡

張文字コード領域に満州文字と拡張漢字の文字コードを割り当てる方が，新システムを構成する処理プログラムの開発も容易であったので，RTF 方式とはしなかった。

なお，プログラムでの処理中は拡張文字領域のコードを用い，ファイルを保存する時に RTF 方式で保存する方法もあり，その場合には，作成した辞書データファイルは，マイクロソフトのワープロである Word や Windows のシステムにデフォルトで入っている WordPad でも扱える（ただし，キーボードからの入力方法の検討が必要ではある)。しかし，本システムでの現時点での辞書作成の過程では，Word などを利用する必要はなく，特に TRF 方式を採用するメリットは少ないと判断し，満州文字と拡張漢字の文字コードとしては，プログラムでの処理中および辞書データのファイル中の両方とも，従来の Shift-JIS の拡張文字領域を用いることとした。満州文字およびそのコードを表 2-1 に，拡張漢字の文字およびコードの一部を表 2-3 に示す（現時点での拡張文字数は 1,500 文字弱である。表の空いているところは重複があったのを除いたために生じたものである)。なお，満州全角文字の最後（コード FABD）の文字は，[本田 1995] 以降に追加されたものである。

ところで，2.1.3 で説明した手順でドット・イメージから変換した Truetype フォントの満州文字は，ソフトウエアを初期設定（デフォルト設定）のまま用いた結果，1 バイトコードのフォントとして作成された（表 2-2)。また，文字鏡文字のフォントの場合は，文字鏡研究会で定めた文字コード(複数のフォントセットからなる)のまま用いた。つまり，満州文字と文字鏡文字の Truetype フォントの文字コードと，本システム（旧および新の両方システム）の文字コードは異なるものとなった。上記で述べたように，処理プログラムの開発などからは，本システムの文字コードの方が都合がよいので，満州文字および文字鏡文字の文字コードを変更して本システムの文字コードに合わせることも検討した。しかし，満州文字フォントは数も少なくて合わせることも可能ではあったが，文字鏡フォントは 8 万字以上の個数であるために，全部の文字を本システムの拡張漢字コードに入れるわけにはいかず，そのためには，拡張漢字に対

応する文字を文字鏡フォントから選び出して，新たな拡張漢字フォントセットを作成する必要があり，その手間および著作権のことを考慮するとあまり良い方法とは思われなかった。そこで，満州文字および文字鏡文字のフォントでの文字コードはそのままとし，本システムの処理プログラムで表示するとき，あるいは印刷するときに，Truetype フォントの文字コードに変換する方法を採用した。

表 2-1　満州文字の本システムでの文字コード

満州文字半角

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
FB20		᠁	᠂	᠃	᠄	᠅	᠆	᠇	᠈	᠉	᠊	᠋	᠌	᠍	᠎	᠏
FB30	᠐	᠑	᠒	᠓	᠔	᠕	᠖	᠗	᠘	᠙	᠚	᠛	᠜	᠝	᠞	᠟
FB40	ᠠ	ᠡ	ᠢ	ᠣ	ᠤ	ᠥ	ᠦ	ᠧ	ᠨ	ᠩ	ᠪ	ᠫ	ᠬ	ᠭ	ᠮ	ᠯ
FB50	ᠰ	ᠱ	ᠲ	ᠳ	ᠴ	ᠵ	ᠶ	ᠷ	ᠸ	ᠹ	ᠺ	ᠻ	ᠼ	ᠽ	ᠾ	ᠿ
FB60	ᡀ	ᡁ	ᡂ	ᡃ	ᡄ	ᡅ	ᡆ	ᡇ	ᡈ	ᡉ	ᡊ	ᡋ	ᡌ	ᡍ	ᡎ	ᡏ
FB70	ᡐ	ᡑ	ᡒ	ᡓ	ᡔ	ᡕ	ᡖ	ᡗ	ᡘ	ᡙ	ᡚ	ᡛ	ᡜ	ᡝ	ᡞ	ᡟ
FB80	ᡠ	ᡡ	ᡢ	ᡣ	ᡤ	ᡥ	ᡦ	ᡧ	ᡨ	ᡩ	ᡪ	ᡫ	ᡬ	ᡭ	ᡮ	ᡯ
FB90	ᡰ	ᡱ	ᡲ	ᡳ	ᡴ	ᡵ	ᡶ	ᡷ	ṅ	š	ū	ž	ᡸ			

満州文字全角

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
FA80	᡹	᡺	᡻	᡼	᡽	᡾	᡿	ᢀ	ᢁ	ᢂ	ᢃ	ᢄ	ᢅ	ᢆ	ᢇ	ᢈ
FA90	ᢉ	ᢊ	ᢋ	ᢌ	ᢍ	ᢎ	ᢏ	ᢐ	ᢑ	ᢒ	ᢓ	ᢔ	ᢕ	ᢖ	ᢗ	ᢘ
FAA0	ᢙ	ᢚ	ᢛ	ᢜ	ᢝ	ᢞ	ᢟ	ᢠ	ᢡ	ᢢ	ᢣ	ᢤ	ᢥ	ᢦ	ᢧ	ᢨ
FAB0	ᢩ	ᢪ	᢫	᢬	᢭	᢮	᢯	ᢰ	ᢱ	ᢲ	ᢳ	ᢴ	ᢵ	ᢶ		

表 2-2　満州文字の Truetype フォントでの文字コード

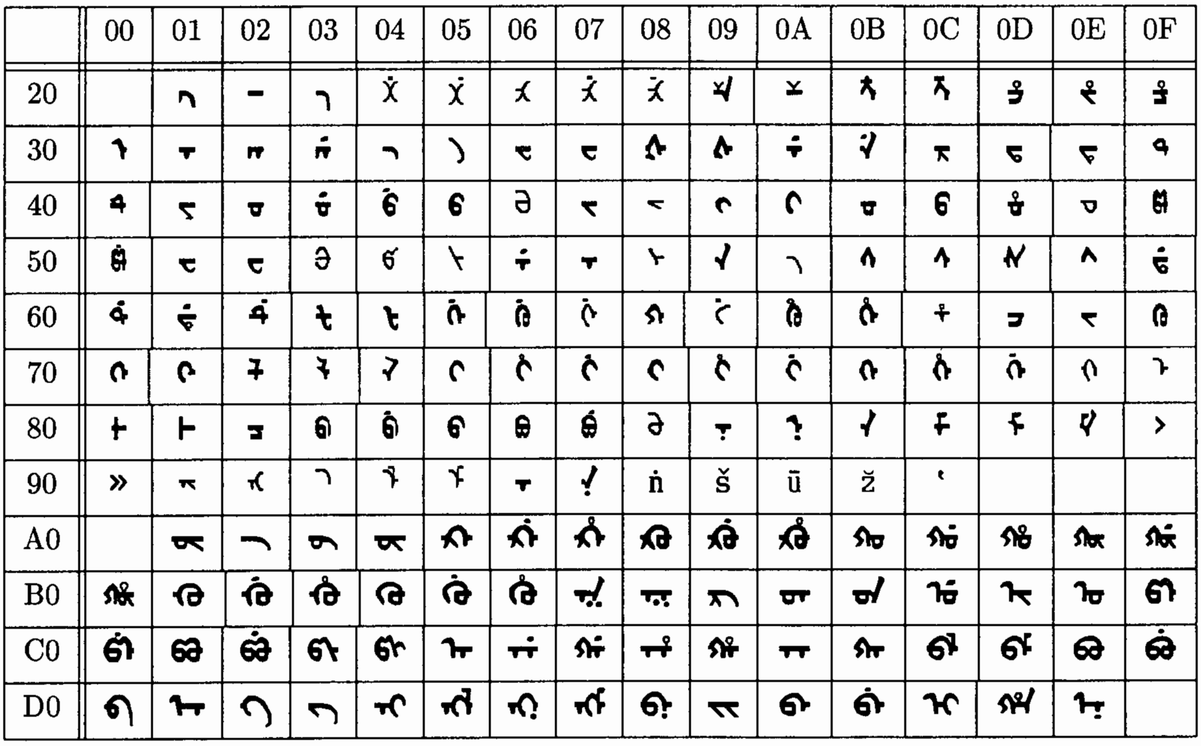

表 2-3　拡張漢字の本システムでの文字コード

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
F020	刁	么	丫	勻	卡	另	夯	奶	扒	氐	犰	扗	丟	伙	仵	仿
F030	划	刖	吆			妃		鑲	汉	[illegible]	犳	穵	佔	你	刨	吧
F040	囤	囫	妗	坍	弝	忐	扭	扯	忑	杈	机	沉	牤	[illegible]	肐	虿
F050	虬	阧	您	盼	汇	滙	宷	寘	寮	岦	岺	崧	嵂	嵭	嶸	嶹
F060	巕	弫	弴	[illegible]	戓	忞	恝	悊	愓	惞	惲	㑞	愰	愷	憘	抦
F070	拼	揵	涒	侉	兕	刹	刽	刵	呢	呲	哂	咕	坫	坯	妯	宨
F080	屉	帔	弁	[illegible]	怵	怔	戕	抿	拖	杭	杴	杻	泎	沫	泔	沱
F090	炁	炕	炖	狍	㺃	狖	玫	疙	秈	芤	芰	芡	衩	缸	畬	佔
F0A0	俏	咧	咵	咱	哆	垛	垡	姤	帢	骈	挂	拴	挎	昏	柯	柤
F0B0	柷	毡	洼	洿	[illegible]	狗	[illegible]	狨	狪	玷	祇	祉	眏	砍	砑	疤
F0C0	籽	紁	紉	胗	芘	紓										

Ⅲ　新システムにおけるデータのフィールド構成

3.1　フィールド構成の変更

最初に述べたように，旧システムの開発からは10年程になっているが，その間に，日本語・満州語および満州語・日本語の両辞書の元データとなるファイルのフィールド構成については，何回かの変更があった。一例であるが，例文については次のように変更を重ねてきた。当初は，満州語・日本語の辞書の作成がメインであり，そのときには，例文を入れる話はなかった。しかし，辞書作成者が作業を進める途中で，例文を入れることとし，各満州語の単語に対する日本語訳の部分（複数の場合がある）に，満州語での例文とその訳を入れていた。しかし，日本語・満州語辞書とする場合は，処理プログラム作成担当の筆者から見た場合，日本語訳と例文の区分が明確でないために，適切に例文が抜き出せない懸念が生じた。相談した結果，満州語の単語1語について例文は1つだけであろうということであったので，例文の箇所は1カ所としていた。しかし，実際に，さらに例文を入れる作業を始めると，満州語単語1つに対して日本語訳が複数ある場合は，そのうちの複数の訳にそれぞれ例文をつけることがあり得るということになり，現在は，そのようなフィールド構成となっている。これらのことは，筆者が満州語および満州語辞書について作成者と十分な相談をし知識を得ていれば，避けられた問題ではあったかもしれないが，とにかく仕様変更としての対処が必要であった。

なお，フィールド構成の変更を考えるときに，フィールド間の区切りについても検討した。これまでの区切り方法は，日本語訳や例文などの長い行があることを考慮して，それぞれのフィールドの区切りにデータとしては絶対に出現しないものとしてControl-Nと「改行」のコードを用いることとし，フィールド中（内容）には単なる「改行」があってもよいことにしていた。しかし，辞書データを保存したファイル中にControl-Nのコードがあると，ファイルをそのまま印刷することができないために，ファイルからControl-Nのコードを除くプログラムを作成しておいて，印刷のたびに適用する必要があるなど

の問題があった。そこで，現在は，Control-Nと「改行」のコードに代えて，辞書データのテキスト中には現れることがない＄と「改行」をフィールドの区切りとして用いることにしている。また，満州語単語間の区切り（レコード間区切り）は，＄と2つの「改行」を用いることにしている。なお，フィールド区切りについては，3.2で述べるようにXML的な方法も考慮したが，現時点では＄を用いる方法をとることとした。

現時点でのデータの構造は次の表3-1のようになっている。なお，日本語訳から例文までの3つのフィールドは日本語訳の数に対応して繰り返すことができる。「番号」のフィールドが2カ所あるが，それぞれ異なる情報としての番号である。

図3-1　辞書の元データのフィールド構成(1)

番号　＄
満州語　＄
ローマ字　＄
品詞　＄
中国語　＄
中国語読み　＄
番号　＄
日本語訳　＄　　「日本語訳　＄」から「例文　＄」までは繰り返し可
日本語読み　＄
例文　＄

3.2　XML的なフィールド構成の記述

現在，インターネット上でのデータの有効利用実現の1つの方法として，フィールド識別のタグを用いたXML（Extensible Markup Language）が注目されている。本システムのデータについても，入力時およびファイル保存時のデータの形式として，次の表3-2のようなフィールド構成のタグを用いることも考慮し，実際にデータファイルを変換・作成してみた。

図3-2　辞書の元データのフィールド構成(2)

〈Word〉
　〈No〉　番号　　〈/No〉

```
〈Man〉満州語　　〈/Man〉
〈Rom〉ローマ字　〈/Rom〉
〈Art〉品詞　　〈/Art〉
〈Chn〉中国語　　〈/Chn〉
〈ChY〉中国語読み〈/ChY〉
〈NoA〉番号　　〈/NoA〉
〈Jpn〉　　〈Jpn〉〈/Jpn〉の部分は繰り返し可
  〈JpW〉日本語訳　〈/JpW〉
  〈JpY〉日本語読み〈/JpY〉
  〈Exm〉例文　　〈/Exm〉
〈/Jpn〉
〈/Word〉
```

XML 的な構成の記述方法は，データを入力する時に入力しているフィールドが明確であるので，その点では単に＄を用いる方法よりもよいと思われる。しかし，XML 的な記述にしても，現時点では「満州語」および「中国語」の部分で用いられる満州文字と文字鏡文字については，フォント本来の文字コードではなく，本システムの文字コードであるので，そのままでは，別のソフトウエアで扱えるものとはならない。さらに，実際にこの形式でデータを作成した場合にタグの文字列で，データサイズがかなり大きくなることとなった。Windows では 4 G バイト程度の仮想メモリ方式を採用しており，新システムの編集機能のためのエディタで扱えるデータの大きさは非常に大きく，上記のタグの部分として増えるものが実際に問題となることはないが，上記のメリット・デメリットから，XML 的な記述方法を現時点で積極的に利用するほどのものではないと判断した。ただし，将来的には，たとえば電子辞書にする場合には，XML 的な記述と，フォントにあわせたフォントセットと文字コードでファイルに保存することも検討している。

Ⅳ　新システムにおける編集機能のプログラム

旧システムでは，編集機能のエディタは，アセンブリ言語で記述したプログラムであった。しかし，Win 32 API をアセンブリ・プログラムから呼び出す

方法についての資料・情報が入手できなかった。一方，言語Cを用いた方法については，資料・情報も比較的多くのものが入手できたので，新システムでは新たに言語Cで編集機能のプログラムを作成することとした。なお，編集機能としては，[本田 1998] で述べたようなデータベース機能を含んだ1単語毎に画面に表示され編集するものとする方法もあったが，これまでのデータ作成の作業で，エディタの方が便利であるとの辞書作成者の意見もあったので，エディタで実現することとした。なお，このエディタは，辞書作成の作業だけでなく，通常のテキスト・ファイルの作成にも利用できるものである。

4.1 エディタの新機能・新コマンド

全く新しく言語Cのプログラムとして作成するので，エディタのコマンド体系・操作方法等は,新規に決めることもできたが,コマンド体系がこれまで利用してきたEmacs likeなエディタと同様な方が，慣れていることもあり適切と考えた。ただし，Windowsの上のソフトウエアであり，Win 32 APIを利用することから，旧システムのものに比べて，マウス操作でのコマンド入力など，次のような機能・仕様を追加することとした。

- 全てのコマンドはキーボードからだけではなく，エディタのウィンドウのメニューバーから，マウスでコマンドを選択・指定することができる。もちろん，例えば，文字列検索コマンドの場合，コマンドに続く文字列の入力はキーボードから行う必要がある。しかし，メニューバーからコマンドを選択できるので，これまでのように，すべてのコマンドを覚えている必要はなく，その点では初心者でも利用が容易なものとなった。なお，キーボードからは利用できず，メニューバーからだけしか利用できないコマンドも追加されている。
- マウス操作でエディタのウインドウの縦横のサイズを変更することができる。これまでのMS-DOS上では，縦24行，横80文字であったが，Windowsの解像度に依存するが，通常のテキスト・ファイルの作成では十分である縦，横サイズのウインドウを用いた作業ができる。

・長い行については，折り返して表示する，あるいは折り返し無しで表示する，の両方の表示が可能。なお，折り返し無しの表示では，カーソル移動とともに，カーソル位置を含む横幅分の部分を表示する。
・縦スクロールバーのマウスでの操作により，表示テキストをスクロールすることが可能。
・マウスのホイール･ボタンで,表示テキストをスクロールすることが可能。
・マウスの左クリックで，カーソルを移動することが可能。
・Windows のクリップボードとエディタの間でテキストのやりとりが可能。本システムの処理プログラム間であれば，クリップボード経由で全ての文字列やりとりが可能である。もちろん，クリップボードを利用できる他のソフトウエアとの間でも，クリップボードを経由しての文字列のやりとりは可能ではあるが，満州文字と拡張漢字については，本システムのソフトウエア以外では表示が適切に行えないこととなる。

メニューバーからのエディタのコマンド呼び出しの階層構成を次に示す。ポップアップメニューの項目の右側に括弧書きしたもの（例えばC-X C-F）はキーボードからも利用できる機能であり，そのコマンド入力を示している。括弧書きがないものは，新システムで追加されたコマンドであり，キーボードからは呼び出す方法を提供していないものである。

表 4-1　メニューバーからのコマンド階層構成

◇ファイル
- 新規作成・開く：新バッファ（C-X C-F）
- 新規作成・開く：現バッファ（C-X C-V）
- 上書き保存（C-X C-S）
- 別名で保存
- カーソル位置に読み込み（C-X C-I）
- 終了（C-X C-C）

◇編集
- マーク設定（C-Space/@）
- コピー（リージョンをコピー）
- 貼り付け

◇Buffer
- バッファ情報表示（C-X C-B）
- バッファを切替（C-X B）
- バッファを閉じる（C-X K）

◇設定
- タブ幅
  2
  4
  6
  8
- 英大小文字同一視

・リージョン操作
　切り取り
　長方形削除（C-＾ C-K）
　長方形復元（C-＾ C-Y）
　Kill（C-W）
　Kill バッファへ保存（M-W）
　Yank（C-Y）
・検索
　順方向検索（C-S）
　順方向次を検索（C-S）
　逆方向検索（C-R）
　逆方向次を検索（C-R）
・置換（M-%）
・英文字列先頭大文字化（M-C）
・英文字列小文字化（M-L）
・英文字列大文字化（M-U）

◇表示
・再表示（C-L）
・バッファ情報表示（C-X C-B）
・現画面分割（C-X 2）
・次ウインドウへ（C-X 0）
・前ウインドウへ（C-U -1 C-X 0）
・現ウィンドウだけ表示（C-X　1）
・現ウインドウを消す（C-X　0）

◇カーソル移動
・先頭行へ（M-〈）
・最後行へ（M-〉）
・行頭へ（C-A）
・行末へ（C-E）
・英文字列分順方向移動（M-F）
・英文字列分逆方向移動（M-B）
・カーソルとマーク位置交換（C-X C-X）
・マーク位置を遡る（C-U C-@）
・次ページへ（C-V）
・前ページへ（M-V）
・Help 表示開始・終了
・バージョン情報
　する
　しない
・括弧対応表示
　する
　しない
・括弧対応表示時間
　2秒
　3秒
　5秒
・全角空白の変換
　半角空白に変換
　全角空白のまま
・保存時の改行コード
　CRLF
　LF
　CR
・折り返し表示
　する
　しない
・Read/Write　属性
　Read Only
　Read Write
・マウス Wheel で送る行数
　1
　2
　3
　4
　5

◇MLing モード
・通常（NFER-0）
・満州文字列変換（NFER-1）
・特殊英文字（NFER-2）
・満州文字：キー対応（NFER-3）
・文字コード入力（NFER-4）

◇ヘルプ

また，マウスの右クリックから利用できるコマンドとしては，比較的よく利用するであろう以下のものを設けている。

表 4-2 マウスの右クリックから利用できるコマンド

- コピー
- 貼り付け
- マーク設定（C-Space/@）
- カーソルとマーク位置交換（C-X C-X）
- マーク位置を遡る（C-U C-@）

4.2 エディタの開発

エディタを開発するに当たり，すでにソース・プログラムが公開されていたMule，石原忠（elrond@gol.com）氏が日本語を扱えるように手を加えたElrond's MicroEMACS 4.00（J 1.02）などのソース・プログラムを入手し，それらのどれかを満州文字や文字鏡文字が扱えるように変更したものを作成することも考えた。しかし，私自身でWindows用のプログラムを開発する経験を持ちたいということもあり，最終的にはそれらのプログラムの処理方式を参考にして，最初から作成することとした。

エディタは，これまでのエディタとの継続性からEmacs likeとすることにしており，参考にしたMuleなどはEmacs Lispのインタープリタを含んでおり，Emacs Lispでプログラムを記述して機能を拡張することができるものであった。しかし，新編集サブシステムであるエディタの開発当時の1999年頃のパソコンでは，最初の起動に時間がかかるものであった。また，そのような機能を実現していなかった旧システムのエディタでもこれまでの利用では十分であったこともありLispインタープリタとLispによる機能の追加は実現しないこととした。

開発のためのプログラム言語としてはC/C++を用いることは決めていたが，Windowsの機能を呼び出すのに，Win 32 APIを用いるかMFC（Microsoft Foundation Class）を用いるかの選択があった。両方ともマイクロソフト社が

決めた仕様であるが，Windowsの機能呼び出しとしては，Win 32 APIの方がより基本的なものであること，Win 32 APIでの関数はBorland社など，マイクロソフト社以外の会社から提供されている言語Cの開発システムの関数とかなり互換性があることなどから，Win 32 APIを利用することとした。なお，Win 32 APIを利用したプログラム作成のために，【プログラミングにおいて参考にした書籍】としてあげているものを参考にした。筆者がWin 32 APIを利用するプログラミングについて，ほとんど知らない状態からの開発であったので，これだけの書籍を入手し参考にすることとなった。一応のプログラムが完成した時点で振り返ってみると，特に参考になったものは「Schildt, Herbert（細井一雄，豊田光智衣訳）『Windows 95 C/C＋＋プログラミング入門』日経BP出版センター，1995」，同じ著者の「Windows 98プログラミング」，「Windows 2000プログラミング標準講座」，「Microsoft Corporation(アスキー書籍編集部他　訳)『公式Win 32 APIレファレンス　Volume 1』アスキー，1999」であった。

エディタのソース・プログラムのうち言語Cのプログラム・ファイルはコメント行，プログラムを見やすくするための空白行なども含み，現時点で9,000行余り，それ以外にヘッダ・ファイル，リソース・ファイルなどもあり，全部で1万行弱である。したがって，プログラムの詳細を説明することは省くが，プログラム作成開始以前に，Windows上でのプログラミングとして多少気になっていた点に対する処理について簡単に記しておく。

○メニューバーからのマウスによるコマンド指定の処理：

これについては，Windowsプログラミングの多くの書籍で解説があったので，方針を決めれば簡単に処理できた。

○キーボード入力の処理：

Windowsでは，キーボードからの入力は，プログラムの実行の流れとは別に，ウインドウ関数（エディタのプログラム中で定義）が，割り込み的に呼び出されて渡される。そこで，ウインドウ関数では，それをエディタのプログラム中に確保している領域（以後，キーボード・バッファという）

に保存する。一方では，「そのキーボード・バッファを調べて，入力があった場合は，取り出して処理し，再度キーボード・バッファを調べる。入力が無かった場合は，再度キーボード・バッファを調べる。」ことを終了コマンドが入力されるまで，繰り返している。

○満州文字，文字鏡文字のエディタの画面（ウインドウ）への表示処理：

英数字や通常の日本語文字の表示は，参考にした書籍に書かれていた方法で可能であった。満州文字および文字鏡文字の表示については，関数SelectObjectでフォントを切り替えてからTruetypeでの文字コードを関数TextOutで出力することにより可能であった。

数値型の変数に代入されている値を表示する場合は，関数TextOutは数値型の変数を引数とはできないので，あらかじめ関数sprintfで数値をテキストに変換しておかなければならない。このようなことも含めて，参考にしたWindowsプログラミングの多くの書籍で解説があった。

○ウインドウ操作に対する処理：

ウインドウのサイズ変更，ウインドウの移動などのユーザによるウインドウの操作に対しては，Windowsが対応するが，そのとき，Windowsからウインドウ関数に再表示（再描画）の要請がなされる。したがって，その要請に対応したプログラム作成が必要である。なお，この再表示・再描画の要請は，非常に頻繁に出されており，当初は高速な表示速度実現を心がけたプログラムとしていた。しかし，現在ではパソコンの能力が十分高いので，あまり複雑な画面表示高速化の工夫よりもプログラムの明快さを優先している。

○エディタでの編集対象のテキストの管理：

MS-DOSベースの時は，テキストの管理は全テキストを入れるだけの十分な連続領域を確保する方法であったが，今回は，各行毎に連続領域を確保し，それをポインタで行の順にリンクする方法とした（この方法は，参考にした多くのプログラムで採用されていた方法でもあった）。

基本となるテキストの管理方法を変更したために，各コマンドの処理方

法も変更する必要があった。MS-DOS ベースの場合は，CPU の処理能力が低かったので，処理高速化の工夫をいろいろなところでしていたが，現在は CPU の処理能力が高くなっているので，プログラムの明快さを優先している。例えば，基本的な１文字分の前へのカーソル移動の処理（コントロール F に対応）は，移動後の位置が，現在行内，現在行外，現在画面内，現在画面外の場合を含んだ処理となっている。そこで，次の行への移動は，行が替わるまで１文字分の前方向へのカーソル移動を繰り返して呼び出し，同様に，次画面への移動は，表示が現在画面外になるまで，文字分の前方向へのカーソル移動を繰り返して呼び出すというような処理にしている。このように処理することにより，１文字分の前後へのカーソル移動の処理を正しく作成しておけば，他の部分は簡単になり，それだけ誤りは少なくなる。

○キル・バッファの管理：

テキストの管理と同様に，MS-DOS ベースの時の連続領域を確保する方法ではなく，固定長の領域をポインタでリンクする方法とした。キル・バッファの固定長領域には，行の長さが短い場合には，改行を示すものとして CR のコード（16 進数で０D）を用いて複数行をいれ，逆に，行が長い場合には，１行が２つ以上の固定長の領域にまたがることもある。

V　新システムにおける辞書用ファイル作成プログラム

本システムでは，編集用エディタで作成したデータ・ファイルを元に，満州語・日本語辞書および日本語・満州語辞書の２つの Tex（Latex）のファイルを，プログラムにより作成し，そのファイルを Tex システムで処理し DVIOUT などの DVI ウエアを用いて印刷するように考えている。この Tex 形式のファイル作成プログラムは文字コードを編集システムで用いているものから Tex 用のフォント・セットと文字コードに変更することも行っている。

現在入力している辞書データから，満州語・日本語辞書を作成する場合には，データの一部を除き，文字コードを Tex 用に変更する程度の作業でよく，

見出し語の順序を変更する必要はない見込みであるが，日本語・満州語辞書作成のためには，辞書データの日本語訳の読みの項目をキーにして辞書式順序に見出し語を並べ直し，かつその日本語訳項目，満州語ローマ字表現項目，満州語項目，例文項目などの適切な項目を取り出して，Tex 用に変更しなければならない。その際の見出し語の並べ直しには ISAM 方式のデータ管理方式を用いる予定である。

当初は，ISAM 方式のデータベース機能を備えたサブシステムを辞書データの入力用として考え，MS-DOS の上でその機能を実現していた［本田 1998］。しかし，既に述べたように辞書作成作業を実際に進める過程で，補助的な利用のために考えていたエディタでの入力の方が利用しやすいということになり，データベース機能を備えたサブシステムは入力用としては用いなくなった。しかし，そのサブシステムでは，ISAM 方式により満州語ローマ字項目でのキーワード管理，日本語訳項目でのキーワード管理を用いているので，日本語訳の順序で確認するなど，入力した辞書データの確認用としては有用であるので，MS-DOS 上でのものを Windows でのキーボード入力，画面表示に対応したものに変更して，新システムでも，同様なものを開発中である。

また，その MS-DOS 上でのプログラム中の ISAM 方式のキーワード管理のかなりの部分は，日本語を見出し語とする日本語・満州語辞書用の Tex 形式のデータファイルの作成にも利用することにしている。なお，Windows 上に作成中であるデータベース機能を備えたシステムについては，作成している辞書データを元に，電子辞書を作成する場合にも利用できると考えている。

Ⅵ　新システムにおける印刷

6.1　データファイルの確認のための印刷

すでに述べたように，最終的な辞書としての印刷形式は Tex システムを用いるが，辞書データの入力途中で，確認のためにデータの全部あるいは一部を印刷することができれば便利である。その場合，Tex 用に変換するプログラムを作成し，印刷したい部分に適用する方法もある。しかし，その場合，辞書

データをプログラムで変換し，Tex を適用して DVI ファイルを作成し，DVI ウエアで印刷するという手順となる。この手順はあまり複雑ではないが，変換することなく直接印刷プログラムがあれば，入力データの確認のための印刷はもっと手軽に行えると思われた。

そこで，本システムで用いている満州文字，文字鏡文字を含むファイルに対する印刷プログラムを作成することにし，印刷プログラム作成のための資料・情報を収集していたところ，曽田氏（http://www 2. biglobe. ne. jp/~sota/index. html）が印刷プログラム PPM をそのソースプログラムと一緒に公開されていたものを入手した。そのプログラムは，単なる印刷だけでなく，用紙の指定，1ページの行数と1行の文字数やページ番号などのページ・スタイルの指定，プレビューなどの機能を備えたものであり，同機能のものを私が作成するよりも，それに追加・変更した方がはるかに短時間にできそうであったので PPM を本システム用に追加・変更することを検討した。その結果，プログラム中でそれぞれ関数 CreateFont および CreateFontIndirect を呼び出して満州文字と文字鏡文字のフォントを作成することと，ファイル中の満州文字と文字鏡文字の文字コードを表示・印刷用のフォントと文字コードに変換することを追加・変更する程度でプレビューと印刷の両方とも実現できることが分かった。

そこで，曽田氏に，ソース・プログラムに対する追加・変更をお認めいただくよう依頼したところ，非常に快くご承認いただけたので，本システムでは，直接のファイル印刷用としては，それを用いることとした。

6.2　辞書ファイルの印刷

最終的な印刷には Tex を利用するが，その手順は次のように考えている。ただし，以下でのファイル名は確定したものではない（最終的には変更の可能性もある）。

(1)　次のファイルを用意する。

○ DicMJMk. c（DicMJMk. exe）：辞書用データから満州語・日本語辞書

の内容部分を Tex 形式のファイル（DicMJ. xm）を作成するプログラム。文字コード変換は行わない。

○DicJMMk. c（DicJMMk. exe）：辞書用データから日本語・満州語辞書の内容部分を Tex 形式のファイル（DicJM. xm）を作成するプログラム。文字コード変換は行わない。

○DicConv. c（DicConv. exe）：DicMJ. xm あるいは DicJM. xm 中の満州文字コードおよび文字鏡文字コードを Tex 用に変換し，DicMJ. tex あるいは DicJM. tex を作成するプログラム。

○DicMain. tex：辞書用 Tex ファイルの枠組み的なもので以下の項目を含む。

・用紙（Ａ４，Ｂ５）の設定値

・1段組，2段組の指定

・フォント（フォントファミリ，フォントスタイル，フォントシェイプ，フォントシリーズなど）の定義，満州語・日本語辞書および日本語・満州語辞書の１つの見出し語に対する記述（見出し語とその訳，例文など）のマクロ

・DicMJ. tex，DicJM. tex をインクルードする。

(2) 辞書データから DicMJ. xm あるいは DicJM. xm をプログラム DicMJMk. exe あるいは DicJMMk. exe により作成する。

(3) DicMJ. xm あるいは DicJM. xm からプログラム DicConv. exe で満州文字と文字鏡文字の文字コードを Tex 用の文字コードに変換し，DicMJ. tex あるいは DicJM. tex を作成する。

(4) DicMJ. tex あるいは DicJM. tex を Tex（pLatex）システムで処理し，DicMJ. dvi あるいは DicJM. dvi を作成する。

(5) DVI ウエア（DVIOUT を用いる予定）で DicMJ. dvi あるいは DicJM. dvi を印刷する（DVI ファイルから高精細密度で印刷するシステムを有した印刷会社があれば，依頼することも考えている）。

なお，(1)の DicMJMk. c あるいは DicJMMk. c では文字コードの変換まで

は行わないのは，以下の理由による。tex 用のファイルを作成するのに，どうしてもプログラムでは処理できないようなことがあるかもしれないが，その場合には，編集サブシステム（エディタ）を用いて手作業で修正・変更することになるが，そのときに，エディタで満州文字と文字鏡文字が見やすいように表示された方がよい。

以下，図 6-1，図 6-2 に辞書の印刷例を示す（例であり，見出し語が少ないので同じ見出し語を繰り返して用いている）。印刷例では 1 段組，2 段組，見出し語を大きくする，日本語訳を大きくするなどをテスト的に試しているが，現時点で最終的な形式は定まっていない。

Ⅶ　おわりに

満州語の辞書作成の補助システム作成について，辞書作成者からお話があり，最初のシステム（旧システム）の開発を開始してからは 10 年ほど，新システムの開発開始からでも 3 年ほどになり，システム開発としては非常に長期のものとなっている。本システムの場合には，辞書データの作成・入力作業にそれだけの時間がかかっていることが大きな要因である。もっとも，見出し語の数，例文の数などからみてもかなり充実した辞書となるようであり，そのようなものを 1 人で作成するには，この程度の時間が必要なのかもしれない。また，辞書作成の作業に伴い，データの変更，文字の追加（特に，拡張漢字を文字鏡研究会に依頼し，それをシステムに組み込む）など，システムの変更を行っており，最終的な仕様がすべて固まるのは，あるいは辞書が完成するときかもしれない。なお，辞書作成者によれば，辞書完成までは，さらに今後 2 年間ほどかかる見込みということである。

今後の大きなシステム開発としては，ローマ字表記から満州文字表記を生成する機能の実現がある。これについては，辞書作成者による著書［河内 1996］［河内 2000］や，国会図書館で「満州語」で検索した書籍［渡部 1930］［戸部 1987］［池上 1999］などで調べたが，満州文字表記をローマ字表記に変換することについては述べられているが，その逆は述べられていなかった。したがっ

図 6-1　1段組の例

afabumbi [ᠠᡶᠠᠪᡠᠮᠪᡳ] v.　Ⅰ. 交付する。寄託する。送付する。輸納する。解交する。交與する。言いつける。命ずる。諭す。返却する。おさめる。ゆだねる。任用する。bi emu wang de emu gargan i cooha be afabufi cuwan arafi 朕は一王に一支隊の兵を〈ゆだね〉船を造り (内. 崇 2. 正.24)。amban mini beye be,huwang di de afabufi 臣は躬を皇帝に〈ゆだね〉(内. 崇 2. 正.24)。juse, sargan, uksun mukun be meni meni bargiyabu seme afabuhabi 子、妻、宗族をおのおの保護せしめよと〈命じた〉(内. 崇 2. 正.24)。endurinnge hese, dorgi ku de afabufi baicame gaisu 聖旨を奉じたるに内庫に〈交與し〉察收せよ、と (禮史. 順 10.8.25)。amin beile — de amba cooha be afabufi— cooha unggihe 阿敏貝勒 — に大軍を〈統せしめ〉— 往征せしめた (滿太宗. 天聰元. 正.8)。meni cin tiyan jiyan yamun i leo ke　ko i juwe hafan be unggifi, erin be tuwara baita de afabumbihe 我が欽天監の漏刻科の二官員を差し、報時の事を〈監せしめていた〉。minggurun i buhe ejehe doron be afabume jihebi 明國の與えた勅印を〈おさめて〉前來す (禮史. 順 10.8.17)。baita de afabume tucibuhe hafan 帖 委供事の官員 (禮史, 順 10.8.17)。［政部 5・辦事 1, 交付］1769/1907。Ⅱ. 攻めさせる。

afaha [ᠠᡶᠠᡥᠠ] n. Ⅰ. 目録、物品名や人名等の列記表。「政部 5・事務 1. 単子］1667/1797。Ⅱ. 書物の一葉。丁。［. 樂部 7. 書 2. 篇子］2827/3044。Ⅲ.（紙一枚二枚の）枚［樂部 7. 文学什物 1. 一張紙］3079/3312。Ⅳ.v. 攻めた。戦った。.

afaha [ᠠᡶᠠᡥᠠ] n. Ⅰ. 目録、物品名や人名等の列記表。「政部 5・事務 1. 単子］1667/1797。Ⅱ. 書物の一葉。丁。［. 樂部 7. 書 2. 篇子］2827/3044。Ⅲ.（紙一枚二枚の）枚［樂部 7. 文学什物 1. 一張紙］3079/3312。Ⅳ.v. 攻めた。戦った。.

afaha [ᠠᡶᠠᡥᠠ] n. Ⅰ. 目録、物品名や人名等の列記表。「政部 5・事務 1. 単子］1667/1797。Ⅱ. 書物の一葉。丁。［. 樂部 7. 書 2. 篇子］2827/3044。Ⅲ.（紙一枚二枚の）枚［樂部 7. 文学什物 1. 一張紙］3079/3312。Ⅳ.v. 攻めた。戦った。.

afaha [ᠠᡶᠠᡥᠠ] n. Ⅰ. 目録、物品名や人名等の列記表。「政部 5・事務 1. 単子］1667/1797。Ⅱ. 書物の一葉。丁。［. 樂部 7. 書 2. 篇子］2827/3044。Ⅲ.（紙一枚二枚の）枚［樂部 7. 文学什物 1. 一張紙］3079/3312。Ⅳ.v. 攻めた。戦った。.

afaha [ᠠᡶᠠᡥᠠ] n. Ⅰ. 目録、物品名や人名等の列記表。「政部 5・事務 1. 単子］1667/1797。Ⅱ. 書物の一葉。丁。［. 樂部 7. 書 2. 篇子］2827/3044。Ⅲ.（紙一枚二枚の）枚［樂部 7. 文学什物 1. 一張紙］3079/3312。Ⅳ.v. 攻めた。戦った。.

afaha [ᠠᡶᠠᡥᠠ] n. Ⅰ. 目録、物品名や人名等の列記表。「政部 5・事務 1. 単子］1667/1797。Ⅱ. 書物の一葉。丁。［. 樂部 7. 書 2. 篇子］2827/3044。Ⅲ.（紙一枚二枚の）枚［樂部 7. 文学什物 1. 一張紙］3079/3312。Ⅳ.v. 攻めた。戦った。.

bithe [ᠪᡳᡨᡥᡝ] n. Ⅰ. 書物。本。書類［7 樂部. 書 1. 書］2752 /2965。Ⅱ. 手紙。書信。［7. 樂部. 書 3. 書札］2933 / 3158。duka tucire bithe gaisfi unggiki 應に出關の〈引〉を給し差すべ し (禮史. 順 10.8.25)。bithe hulame weceki〈祝文〉を読み致祭すべし (禮史. 順 10.8.29)。meni jurgan ci halame benju seme bithe unggiki sembi 臣が部より〈行文し〉それをして■換せしめたい (禮史. 順 10.8.25)。sansi goloi dzung du meng ciyoo fang de bithe unggifi sangnahabi　陝西省総督孟喬芳に〈行文し〉賞給した (禮史. 順.10.82 5)。suwe elcin takurafi bithe wesimbume 爾等は遣使進〈表〉し (禮史. 順 10.8.25)。

図6-2　2段組の例

afabumbi [ᠠᡶᠠᠪᡠᠮᠪᡳ] v. Ⅰ.交付する。寄託する。送付する。輸納する。解交する。交與する。言いつける。命ずる。諭す。返却する。おさめる。ゆだねる。任用する。bi emu wang de emu gargan i cooha be afabufi cuwan arafi 朕は一王に一支隊の兵を〈ゆだね〉船を造り (內. 崇 2. 正.24)。amban mini beye be,huwang di de afabufi 臣は躬を皇帝に〈ゆだね〉(內. 崇 2. 正.24)。juse, sargan, uksun mukun be meni meni bargiyabu seme afabuhabi 子、妻、宗族をおのおの保護せしめよと〈命じた〉(內. 崇 2. 正.24)。endurinnge hese, dorgi ku de afabufi baicame gaisu 聖旨を奉じたるに内庫に〈交與し〉察收せよ、と (禮史. 順 10.8.25)。amin beile — de amba cooha be afabufi— cooha unggihe 阿敏貝勒 — に大軍を〈統せしめ〉— 往征せしめた (滿太宗. 天聰元. 正.8)。meni cin tiyan jiyan yamun i leo ke ko i juwe hafan be unggifi, erin be tuwara baita de afabumbihe 我が欽天監の漏刻科の二官員を差し、報時の事を〈監せしめていた〉。minggurun i buhe ejehe doron be afabume jihebi 明國の與えた勅印を〈おさめて〉前來す (禮史. 順 10.8.17)。baita de afabume tucibuhe hafan 帖委供事の官員 (禮史, 順 10.8.17)。[政部 5・辦事 1, 交付] 1769/1907。Ⅱ. 攻めさせる。

afaha [ᠠᡶᠠᡥᠠ] n. Ⅰ. 目録、物品名や人名等の列記表。「政部 5・事務 1. 単子] 1667/1797。Ⅱ. 書物の一葉。丁。[. 樂部 7. 書 2. 篇子] 2827/3044。Ⅲ.（紙一枚二枚の）枚 [樂部 7. 文学什物 1. 一張紙] 3079/3312。Ⅳ.v. 攻めた。戦った。.

afaha [ᠠᡶᠠᡥᠠ] n. Ⅰ. 目録、物品名や人名等の列記表。「政部 5・事務 1. 単子] 1667/1797。Ⅱ. 書物の一葉。丁。[. 樂部 7. 書 2. 篇子] 2827/3044。Ⅲ.（紙一枚二枚の）枚 [樂部 7. 文学什物 1. 一張紙] 3079/3312。Ⅳ.v. 攻めた。戦った。.

afaha [ᠠᡶᠠᡥᠠ] n. Ⅰ. 目録、物品名や人名等の列記表。「政部 5・事務 1. 単子] 1667/1797。Ⅱ. 書物の一葉。丁。[. 樂部 7. 書 2. 篇子] 2827/3044。Ⅲ.（紙一枚二枚の）枚 [樂部 7. 文学什物 1. 一張紙] 3079/3312。Ⅳ.v. 攻めた。戦った。.

afaha [ᠠᡶᠠᡥᠠ] n. Ⅰ. 目録、物品名や人名等の列記表。「政部 5・事務 1. 単子] 1667/1797。Ⅱ. 書物の一葉。丁。[. 樂部 7. 書 2. 篇子] 2827/3044。Ⅲ.（紙一枚二枚の）枚 [樂部 7. 文学什物 1. 一張紙] 3079/3312。Ⅳ.v. 攻めた。戦った。.

afaha [ᠠᡶᠠᡥᠠ] n. Ⅰ. 目録、物品名や人名等の列記表。「政部 5・事務 1. 単子] 1667/1797。Ⅱ. 書物の一葉。丁。[. 樂部 7. 書 2. 篇子] 2827/3044。Ⅲ.（紙一枚二枚の）枚 [樂部 7. 文学什物 1. 一張紙] 3079/3312。Ⅳ.v. 攻めた。戦った。.

afaha [ᠠᡶᠠᡥᠠ] n. Ⅰ. 目録、物品名や人名等の列記表。「政部 5・事務 1. 単子] 1667/1797。Ⅱ. 書物の一葉。丁。[. 樂部 7. 書 2. 篇子] 2827/3044。Ⅲ.（紙一枚二枚の）枚 [樂部 7. 文学什物 1. 一張紙] 3079/3312。Ⅳ.v. 攻めた。戦った。.

bithe [ᠪᡳᡨᡥᡝ] n. Ⅰ. 書物。本。書類 [7 樂部. 書 1. 書] 2752 /2965。Ⅱ. 手紙。書信。[7. 樂部. 書 3. 書札] 2933 / 3158。duka tucire bithe gaisfi unggiki 應に出關の〈引〉を給し差すべし (禮史. 順 10.8.25)。bithe hulame weceki〈祝文〉を読み致祭すべし (禮史. 順 10.8.29)。meni jurgan ci halame benju seme bithe unggiki sembi 臣が部より〈行文し〉それをして■換せしめたい (禮史. 順 10.8.25)。sansi goloi dzung du meng ciyoo fang de bithe unggifi sangnahabi　陝西省総督孟喬芳に〈行文し〉賞給した (禮史. 順.10.82 5)。suwe elcin takurafi bithe wesimbume 爾等は遣使進〈表〉し (禮史. 順 10.8.25)。

acabumbi [ᠠᠴᠠᠪᡠᠮᠪᡳ] v. niyalmai buyehede, abka urunakū acabumbi ṅšūž 人の欲する所に天は必ず〈従う〉(內. 崇 2. 正.24)。

て，辞書作成者から，さらに情報を得ることも必要ではあるが，プログラムとして完全なものが作成できるだけの情報とはならない恐れもある。一方，変換規則が適用できるように単語を区切ることができれば，プログラムで対応できそうなので，現実的な対処法としては，満州文字での単語に手作業で区切りを入れる方法を検討している。たとえ手作業で区切りをいれるにしても，キーボードから満州文字を入力するよりもはるかに楽な作業である。

なお，以前のMS-DOS上でのシステムの場合，満州文字のキーボード入力，文字表示などは，デバイスドライバとして実現していたので，例えばコマンドTypeで満州文字入りのファイルを画面に表示することなど，特に満州文字の表示に対応していない通常のソフトウエアで，満州文字を入力・表示することが可能であった。しかし，新システムでは，そのような方法が分からなかったこともあり，作成・開発した各処理プログラムで入力・表示に対応することとしているが，現時点では，編集機能のエディタがメインとなっているために，そのことで特に支障は生じておらず，最終的なものとしてもデバイスドライバとして実現することは必要ないと思っている。

本システムの開発途中の平成9年度から「少人数での辞書編纂のための支援システムの研究——日本・ブルガリア語の活用・学習辞典編纂を例にして——」という研究課題で，科学研究費補助金（基盤研究C）を受けた。本システムの開発は，その研究課題の一部でもあったが，科学研究補助金は3年間で平成11年度までであり，満州語に関する辞書作成は終了していないために，その後もシステム開発を続けている。

参 考 文 献

[池上 1999] 池上二良「満洲語研究」汲古書院，1999
[河内 1996] 河内良弘「満州語文語文典」京都大学学術出版会，1996
[河内 2002] 河内良弘「満州語文語入門」京都大学学術出版会，2002
[戸部 1987] 戸部実之「満州語入門」泰流社，1989
[本田 1987] 本田道夫，中村邦彦「複数の計算機システムにおける共通コマンド体系の画面

エディタの開発」，香川大学経済論叢第 60 巻第 3 号，1987
[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，1995
[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，1998
[渡部 1930] 渡部薫太郎（編）「満州語綴字全書」大阪東洋学会，1930

プログラミングにおいて参考にした書籍

Ezzell, Ben/Blaney, Jim（小暮明　監訳)「Windows 98 プログラミングバイブル」インプレス，1999
Chen, Steven「Windows 95 API ケーススタディ」翔泳社，1995
Microsoft Corporation（アスキーテクライト　訳)「Microsoft Windows 95 プログラマーズガイド」アスキー，1996
Microsoft Corporation (アスキー書籍編集部他　訳)「公式 Win 32 API レファレンス Volume 1，2，3」アスキー，1999
Petzold, Charles（「プログラミング Windows 第 5 版（上）（下)」アスキー，2000
Petzold, Charles（エー・ピー・ラボ，長尾高弘　訳)「プログラミング Windows 95」アスキー，1997
Rector, Brent/Newcomer, Joshph（株式会社クイック／和田なつみ　訳)「Win 32 プログラミング大全（上）（下)」アスキー，1998
Schildt, Herbert (細井一雄，豊田光智衣　訳)「Windows 95 C/C＋＋プログラミング入門」日経 BP 出版センター，1995
Schildt, Herbert（柏原正三　訳)「Windows 98 プログラミング」翔泳社，1998
Schildt, Herbert (山本信雄　監訳)「Windows 2000 プログラミング標準講座」翔泳社，2000
Simon, Richard/Gouker, Michael/Barnes, Brain（株式会社ロングテール，長尾高弘　訳)「Widnows 95 API バイブル(1)(2)(3)」翔泳社，1996，1997
Telles, Matthew/Gooke, Anderew（羽山博　監訳)「Windows 95 API パワフルテクニック大全集」インプレス，1997
エキスパートマグネティックス株式会社「VC＋＋による Win 32 プログラミング TIPS」技術評論社，1998
株式会社ガリバー「Windows 98 API バイブル」翔泳社，1999
北山洋幸「Windows 2000 時代の Win 32 API システムプログラミング」カットシステム，2000
北山洋幸「技術者のための Visual C＋＋実践プログラミング技法」技術評論社，1999
田中ひろゆき「Visual C＋＋6.0 の応用 50 例」ソフトバンク，1998
常岡伸二「実例で学ぶ Win 32 API 活用術」CQ 出版社，2000

プログラミングにおいて参考にしたホームページ

http : //www2. biglobe. ne. jp/~sota/index. html：曽田氏ホームページ
http : //www. forest. impress. co. jp/：窓の杜
http : //www. ling. is. tohoku. ac. jp/：現在は閉じられているようである
http : //www. l. u-tokyo. ac. jp/GT/：東京大学多国語処理研究会
http : //www. mojikyo. org/html/index. html：文字鏡研究会