朝鮮語テクストのコンピュータ処理について

―― 中期朝鮮語のKWIC索引作成の場合 ――

趙　　義成

On Computer Processing of Korean Text

―― In Case of KWIC Index Composing of Middle Korean ――

CHO, Eui-sung

(2000年1月17日受理)

県立新潟女子短期大学研究紀要

別刷 2000 No. 37

朝鮮語テクストのコンピュータ処理について
── 中期朝鮮語のKWIC索引作成の場合 ──

趙　義成

On Computer Processing of Korean Text
── In Case of KWIC Index Composing of Middle Korean ──

CHO, Eui-sung

0. はじめに

0.1 目的

本稿は中期朝鮮語（以下「中期語」と呼ぶ）テクストのコンピュータ処理に関して，その方法を模索し提示することを目的とする．現在，国際的に，あるいは韓国国内的に通用しているハングルのコードは現代朝鮮語の処理ために用意されており中期語の処理は前提とされていない．従って，中期語の処理は一般のテキストファイルが利用できず，韓国の研究者の多くは中期語が表示できるワードプロセッサを利用して中期語テクストの処理をしている．しかしながら，ワードプロセッサの機能だけでは中期語テクストをデータとして処理するには役不足であり，中期語のデータベース化は少なからぬ困難を伴っているのが実情である．

本稿では中期語のKWIC索引（文脈つき語彙索引）を，韓国国内のワードプロセッサである「アレアハングル」とC言語を用いて作成しようと思う．

0.2 中期語語彙索引

中期語の語彙索引は，過去に日本および韓国でいくつか作成されている．日本では「月印千江之曲KWIC索引」（福井玲，1985），「諺解三綱行實圖索引」（志部昭平，1992），「釈譜詳節第六KWIC索引」（野間秀樹・伊藤英人，1994）などがあるが，日本のコンピュータ事情からハングルそれ自体を操作するのが困難であったため，これらはハングルをローマナイズして処理をしている[1]．韓国では近年，文獻研究會（1994），서상규（1997）など，コンピュータを用いた語彙索引が世に現われている．いずれの場合にも自作ソフトウェアなどを開発して索引を作成しており，その性能は一様ではない．とりわけキーワードの並べ替えのプログラミングをいかに行なうかによって，できあがる索引の善し悪しが左右されるといっても過言ではない．従って，本稿においても最も核心となる部分はキーワードの切り出し方，およびその並べ替えのプログラムである．

本稿で構想するKWIC索引は，以下のような形を目指す：

（1）正順と逆順の索引の作成．語彙索引は，単にる単語を検索するのに留まるべきではない．言語研究においては，単語のみならず語尾など，特定の形態に着目する場合も少なくない．従って逆順によって特定の語尾などが一目で観察できるような索引が要求される．

（2）傍点情報の整理．過去の中期語語彙索引では，傍点情報はことごとく無視されてきたと言っ

1）志部昭平の「諺解 三綱行実図研究 索引」はコンピュータ処理の段階ではハングルをローマナイズし，印刷の段階でハングル化している．

国際教養学科

てもよい．甚だしくは，傍点を全く記入しないテクストを用いた索引もあるが，中期語研究おいてアクセント研究は非常に重要な分野の１つであるのだから，傍点の情報を何らかの形で索引に反映させる必要があろう．従って，本稿では傍点をも順序だてて並べる索引の作成を模索する．

1. ハングルのコード体系

1.1 完成型と組み合わせ型

ハングルのコードは過去にさまざまな試みがなされ，３バイトコード・多バイトコードなどが提起されたが，現在では日本語のコードと同じく２バイトコード体系が一般的に採用されている．２バイトコード体系はさらに「完成型」と「組み合わせ型」という２つの体系に分けられ，お互い異なった原理でコードが編成されている．

「完成型」は「KS完成型」とも称され，国際規準にのっとったKS C 5601で採用された標準コード体系である．このコード体系では，ハングル１文字に対して１つの２バイトコードを割り当てる．制御文字などのコードとの衝突を避けるため，上位バイト・下位バイトともにA1hからFEhまでの領域のみを使用し，うちハングルに2350字が割り当てられている．例えば，가はB0A0h，다はB4D9hのごときである(巻末資料１参照)．

このコード体系は国際的標準となっているにも拘らず，韓国国内での評価はあまり芳しくない．現代語においてありうべきハングルの文字数は総計11172字であるが，完成型ではコード総数の制約上，日常的に頻繁に使用するハングル2350字のみをコードに収めているため，完成型からあぶれたハングルはコンピュータで表示することができないからである．

巻末資料２は完成型におけるハングルの割り当ての一部であるが，これには 갂，갃，갅，갆 といった文字に対してコードが割り当てられていないのが一目瞭然である．

これに対して「組み合わせ型」は文字どおりハングル字母を分解して，それぞれの字母を「組み合わせ」てコードを作る．組み合わせ型では２バイトのうち，まず先頭の１ビットをハングルか英数文字かを判別するマーカーとしている．すなわち，先頭ビットが０ならば英数文字，１ならばハングル・漢字と判定するのである．そして残りの15ビットを５ビットずつ等分し，上位５ビットを初声に，中位５ビットを中声に，下位５ビットを終声にそれぞれ割り当てる．

＜図１＞組み合わせ型コードの構造

上位バイト								下位バイト							
1	0	0	0	1	0	1	1	0	1	0	0	0	1	0	1
	初声 ㄱ					中声 ㅠ				終声 ㄴ					

＜図１＞は「큔」という文字の組み合わせ型におけるコードであるが，初声・中声・終声のおのおのの字母にコードが割り当てられ，全体として8B45hが「큔」のコードとなっている（組み合わせ型での各字母のコード割り当てについては巻末資料３を参照）．なお，組み合わせ型では初声・中声・終声のそれぞれに「Fill」と呼ばれるものが１つずつある．これは該当する字母が空白であるときに用いられるコードで，例えば「가」は終声がないので，終声のコードに「Fill」が当てられる．また「ㄱ」は初声がないので，初声のコードに「Fill」が当てられる．

組み合わせ型はありうべきハングルの全ての文字に対してコードが用意されているので，完成型のように「ハングルが出ない」という不都合は一掃される．しかし，字母の組み合わせによっては制御文字のコードとの衝突が起きるため，国際的には認められていないという問題がある．

1.2. アレアハングルのコード体系

韓国のワープロソフト「アレアハングル」は現代朝鮮語においてありうべき全ての ハングルと，中期語を含め古典に現われたハングルが表示できるが，それを可能たらしめているのはこのソフトで独自に定めた「アレアハングル２バイトコード(以下「アレアコード」と呼ぶ）である．アレアコードでは従来の１バイト文字も全て２バイトコードで表現しているため，英数文字もハングルも全ての文字が２バイトとして処理される．すなわち，既存の１バイト文字はコードの頭に00hを付加することによって２バイト化される．例えば

「A」のコードは41hであるが，アレアコードでは00hを冠して0041hのように2バイト化しているのである．ハングル・漢字は組み合わせ型と同様に先頭ビットが1で8000h以降に配置されている．これによって「アレアハングル」では既存の1バイト文字と2バイト文字を衝突させることなく共存させている．

また，既存の組み合わせ型コードでは，各字母コードごとに字母の割り当てられていない箇所が散在するが，「アレアハングル」ではその 空白部分に 古典ハングル字母の 一部を 割り当てている．これによって古典ハングルの一部が組み合わせ型によって表現されることになる．しかし，それだけでは 古典ハングルを 網羅することはできない．そこで「アレアハングル」では，中声に当てている中位5ビットのうち，始めの2つを空け，そこに完成型方式で古典に現われるハングルを割り当てている（このコードを完成型古典コードと呼ぶことにする）．単純計算で，初声32個×中声2個×終声32個分，合計2048個のコードを古典ハングル用に割り当てていることになる[2]（巻末資料3参照）．

本稿で試みられる中期語KWIC索引は，このアレアコードを用いてコンピュータ処理を行なう．

2. 中期語テクストの入力

2.1 テクストと付随情報

中期語のKWICを作成するためには，何よりもまず中期語のテクストを作成する必要がある．テクストの書式をどのようなものにするかは，作成者の感性に任されるところであるが，テクストそれ自体以外にも必要と思われる情報として，以下のようなものが考えられる．

まず，テクストの帳・行の情報は欠かすことができない．これはキーワードが原文のどこにあるかを 検索するために 必要不可欠であるからである．

また，中期語のテクストとして重要なのは，傍点を必ず記入することである．韓国国内で作成された中期語のコーパスには傍点が全て省かれたものが少なくないが，これでは中期語テクストとしての資料価値が半減してしまう．

しかして，巻末資料5のようなテクストが作られることになる．ここでは1つの文が1行で入力されている．カッコ内の数字と記号は帳・行の情報である．初めの2字が巻数を表し，次の3字が帳数を表し，a・bが帳の表・裏を表し，その次の数字が行数を表す．従って「01002a5」は，月印釈譜巻一，第2帳表面，第5行を表す．上のテクストではその後ろにさらに s（釈譜詳節の本文），w（月印千江之曲の本文），n（割注）」という付加情報をオプションとして加えている．付加情報としては，例えば会話部分を「 」で括るとか，割注のテクストを［ ］で括るなどが考えられる．これらは必要に応じて入力すればよい．

なお，テクストの保存は，アレアコードによる処理を行なうため，通常のテキストファイルでなく，アレアコードのファイルで行なう．

2.2 分かち書きの問題

テクスト入力の段階で問題となることは，原文でなされていない中期語の分かち書きを，入力の際にどのようにするかということである．分かち書きの問題は単語をいかに認定するかという文法論の問題でもあるが，KWIC作成においてはむしろコンピュータの処理の問題として浮上する問題である．正順と逆順の索引があれば検索したい形式はほぼ求められるが，2つ以上の形態素が合わさった形の場合，語中に入り込んだ形式は検索が不可能である．例えば，「ᄒᆞ늘ᄃᆞᆨ」という語は「ᄒᆞ늘」と「ᄃᆞᆨ」から成るが，これを分かち書きしないで入力するとコンピュータは1つのキーワードと認識し ㅍ の項に入れられる．ところが，例えば「ᄃᆞᆨ」という単語を検索しようとすると「ᄒᆞ늘ᄃᆞᆨ」は ㅍ の項にないため検索することができないという事態に陥る．

このような不都合を解消するには2つの方法をとるしかない． 1つは 「ᄒᆞ늘 ᄃᆞᆨ」のように，2つの形態に分かち書きすることである．このようにすればコンピュータ処理の段階ではそれぞれ別個のキーワードと認識して処理される．いま1

2) 実際には完成型古典コードには1865文字が収められている．

つは「풍류=ᄒᆞ다」のように，形態の境界に何らかの分離記号を入れ，コンピュータ処理の段階でそれを認識させ別個のキーワードとして立てさせることである．最終的にコンピュータでは双方とも各形態を別個のキーワードとして立てることになるわけだが，分かち書きをすると入力されたテクストを見たときに，全く別個の単語という意識が働く．従って，合成された形態が1単語であると判断されるような場合には，分離記号を入れる方法のほうが妥当かも知れない[3]．

3. KWIC索引の作成

3.1 作業全般の流れ

KWIC索引を完成させるまでのプログラムの手順は，おおよそ以下のようになる．

(1) テクストから単語をキーワードとして切り出し，その前後に文脈をつけ，KWIC(文脈付きキーワード)を作成する

(2) キーワードを가나다順にソートする

(3) 同一キーワード内で傍点順にソートする

(4) 切り出したKWICを가나다順に並べかえる

まず，(1)の段階で，並べ替えのなされていないKWICの原ファイルが作成される(巻末資料6参照)．この原ファイルが最終的にKWIC索引となるわけである(巻末資料12参照)．(2)の段階では，キーワードのみをソートするわけだが，この作業のために (1) とは別個にキーワードの情報のみを記述したテンポラリファイルを作成しなくてはならない．つまり，キーワードのみのソートの結果を受けて，その情報をもってKWIC全体を並べかえるわけである[4]．また，(3)の作業は過去の索引にないはじめての試みである．

3.2 キーワードの切り出し

テクストからキーワードを切り出し，その前後に文脈をつけるのは，市中に流布しているソフトウェアによっても可能であり，そのノウハウは特殊なものではないのでここでは割愛し，ここではキーワードの情報のみを収めたテンポラリファイルに関して述べることにする．

上に見たとおり，このプログラムはKWICの並べ替えのためにこのテンポラリファイルを直接操作することになるので，キーワード情報が収められたこのテンポラリファイルはこのプログラムの心臓部である．キーワードを가나다順にソートするためには，キーワードの가나다順に合わせて，文字コードが整然と並んでいなければならない．しかし，1.で見たように，アレアコードは組み合わせ型と完成型を併用しており文字の順番とコードの順番が一致していないので，文字の順番に合わせてコードを割り当て直す必要がある．最も簡便な方法は，古典ハングルを含めた全てのハングルを字母に分解し，それぞれに対して順番にコードを割り当てるいわば「完全組み合わせ型コード」を作ることである．しかし，古典ハングルを含めた全てのハングルに現われる各字母の数は，子音(初声・終声)が134個，母音(中声)が66個もある．従来の2バイト組み合わせ型では，各字母に割り当てうる数が最大32個であるため，2バイトではハングルを表現できない．そこで，初声・中声・終声のそれぞれに1バイトを割り当て，合計3バイトによってハングルを表現する(字母のコードは巻末資料7参照)．この方法によれば，「사」は490201h，「훈」は812207hと表現される．このコード体系を便宜上「趙コード」と呼ぶことにする．なお，趙コードではハングルは3バイトで表現できるが，コンピュータで処理するときは long型変数(4バイト)として扱われるため，プログラミングにおいては4バイトコードとして扱う[5]．アレアコードの趙コードへの変換に際して，アレアコードのうち，組み合わせ型の

3) 筆者が実際に入力したテクストでは，「풍류ᄒᆞ다」のような分離用言は「풍류=ᄒᆞ다」のように「=」で分け，「손발」のような合成語は「손-발」のように「-」で分けた．

4) テンポラリファイルによってキーワードのみをソートするのは，アレアコードによるたKWICファイルではコードの問題上가나다順のソートが不可能であるということと，KWICには傍点も記入されているため，そのままでは가나다順のソートができないからである．詳しくは3.2を参照．

5) その際，上位1バイトは空き(00h)となる．

コードは機械的に趙コードにコンバートすればよいが，完成型で割り当てられている古典ハングルはテーブルを用いて変換しなければならない（巻末資料８参照）．

また，キーワードのソートに際しては，傍点の情報も反映させる．例えば「:가, 가, ·가」というハングルを並べ替えるとき，韓国国内で開発された既存のハングルソートプログラムでは傍点のソートは全く無視されてきたのだが，これを「가, ·가, :가」のように，傍点ゼロ・傍点１つ・傍点２つの順番に並べ替える．

以上のことを踏まえて，テンポラリファイルの中にキーワードは以下のような形式で収めることにした．

(1) １つのキーワードに対して40バイトを当てる．ハングル１文字が４バイトで表現されるので，ハングル10文字分である．

(2) 傍点情報に10バイトを当てる．キーワードがハングル10文字分なので，１文字分の傍点に対して１バイトを当てる計算になる．

(3) 上の２つを合わせた50バイトを１つのキーワードの情報とする．

なお，傍点情報10バイトは，40バイトのハングル情報の後ろに置くことになる（<図２参照>）．なぜならば，ハングル自体のソートをまず行なわねばならず，傍点によるソートは２次的であるからである．

<図２>キーワードの情報

ハングル (40バイト)	傍点 (10バイト)

テンポラリファイル作成のルーチンは巻末資料９のごときである．配列 Hangul[0], Tone[0] は正順のキーワード，傍点情報が収められており，配列 Hangul[1], Tone[1] は逆順のキーワード，傍点情報が収められている．逆順の情報は，下の<図３>(2)のように，文字列を字母レベルでまるまるひっくりかえしてテンポラリファイルに収める．こうすることによって，キーワードの処理が後ろからなされることになり，並べ替えが逆順で行なわれるわけである．もちろん文字列がひっくり返っているのはテンポラリファイルだけであり，本体のKWICはそのままである．

逆順索引を作成する場合，<図３>(3)のように文字レベルでのみ逆順にし，字母レベルでは逆順にしない方法もあるが[6]，ここではその方法を取らない．逆順索引の利点は，例えば「간, 흔, 업슨」などの単語から語末の連体形語尾「-ㄴ」を検索することができる点である．字母レベルの逆順にすると，「간, 흔, 업슨」は「ㄴ」の部分に固まって配置されるので，連体形の検索が可能となるが，もし文字レベルのみの逆順にしてしまうとこの３つの単語はそれぞれ全く別個の箇所に配置されてしまい，連体形の検索が不可能となってしまう．

<図３>正順と逆順

(1) 正順

ㅁ	ㅏ	ㄹ	ㅆ	ᆞ	ㅁ
말			씀		

(2) 字母レベルの逆順

ㅁ	ᆞ	ㅆ	ㄹ	ㅏ	ㅁ
씀			말		

(3) 文字レベルの逆順

ㅆ	ᆞ	ㅁ	ㅁ	ㅏ	ㄹ
씀			말		

3.3 文脈の切り出し

前後の文脈を切り出す際に問題となることは，一定の長さの文脈をいかに切り出すかということである．これは印刷の問題とかかわる．従前のように，半角文字のフォント幅がどれも一定で，全角文字のフォント幅がどれも常に半角文字の２倍ならば，機械的に何文字か切り出してくればよい．しかし，アレアハングルのフォントは実際には傍点や英数字，括弧などのフォントの幅がどれも一定でないので，印刷時に版面の文脈の長さを一定にするには，それぞれのフォント幅を考慮に入れて文脈を切り出さねばならない．

巻末資料１０は後半の文脈を切り出すルーチン

6) 例えば서상규(1997)などがこの方法を用いて逆順の索引を作成している．

を簡略化したものである．hwpban，hwp2byteは文字種を調べる関数で，それぞれ半角文字，全角文字か否かを調べる．半角文字の場合は，さらにcharwidthという関数を通す．これはフォントの幅の値を返す関数で，プログラムの中では全角文字を16として，それぞれのフォントの幅を相対値で示している．返された値はclenに加えられていき，clenの値が定められた値Jisuを超えたら文脈の切り出しを中止する．例えば切り出す字数を全角10文字分とするとJisuの値は160であり，clenが160を超えた時点で文脈の切り出しが中止される．

キーワードと前後の文脈が切り出されたら，それを1行にしてKWICの原ファイルに書き出す．書式は「行番号: 前文脈 |キーワード| 後文脈」のようにする(巻末資料6参照)．

3.4 キーワードとKWICの並べ替え

キーワードのソート自体は，ソートプログラムに沿って行なうので，ここでは詳しく触れない．1つだけ言及するならば，より速いソートのためにはクイックソートなどの方法でソートする必要があるが，既存のソートプログラムのほとんどはテキストファイルを ソートするものであるため，4バイトデータをソートするためには独自にソートプログラムを組まねばならない．

キーワードのソートが終了したら，その結果に基づいて同じ順番でKWICの本体を並べ替える．だが，ここでキーワードがどのように並べ替わったかが分からなければ，KWICの本体を並べ替えることができない．そこで，キーワードのテンポラリファイルを作成する際に，キーワード情報とともに，キーワードの一連番号を付加しておく必要がある．この一連番号はKWIC本体の行番号である．キーワードのテンポラリファイルは，作成当初は巻末資料１１の左図ようにテクストに現われる順番で収められているが，ソートの結果，右図のように並べ替えられる．ここで一連番号を参照し，一連番号と同一数値のKWIC本体の行を拾い出して新たにKWICを並べ替えるのである．例えば，最初の単語「ᄆᆞᄆᆡ」は一連番号が７なので，KWICの７行目を読み取り別のファイルに書き出す．次の「ᄀᆞᄅᆞ치라」はKWICの9行目を読み取る．このようにして全てのKWICを並べ替えるわけである．

3.5 その他のこと

本稿では正順と逆順のKWIC索引作成について言及したが，１つ付け加えるならば，接尾辞など完全に語中に入り込んだ形態は，正順・逆順どちらの索引を用いても検索できない．よって，それとは別に語中の特定の形態を抜き出すプログラムを作成する必要がある．手順としては，キーワードの語中に該当する形態が含まれるか否かを調べればよいわけだが，例えば接尾辞「-거-」を抜き出す場合，この接尾辞とは関係なしにたまたま形態が一致するようなもの（例えば「머거도」のようなもの）も無条件に抽出されてしまう．従ってこのように語中に入った形態の索引の作成は，最終的に手作業による選別を行なわなければならない．

4. おわりに

中期語研究は書かれた言語を分析するものであるので，テクストを綿密に分析することから始まる．今までの研究では分析の対象となる語形や形態を テクストから手作業で 見つけ出していたが，コンピュータを用いて作成したKWIC索引は，分析対象をもれなく検索することができるので，中期語研究においては多大な威力を発揮するといえる．そのような意味で，今後もコンピュータによるKWIC索引の作成があちこちで行なわれるものと予想される．

附．ユニコードにおける中期語

ユニコードでは，完成型の最大の欠点であったハングル総字数の制限が撤廃され，現代語においてありうべきハングル11172文字全てに 完成型方式でコードが割り当てられた．これにより現代語におけるありうべきハングルは，全てコンピュータ上で処理が可能になったわけである．しかし，ユニコードのこの完成型コードには，古典ハングルが一切含まれていないため，中期語の処理にお

いては，依然問題がある．한국어정보처리연구소(1999) によると，ユニコードには ハングル字母コードが割り当てられており，これを用いた「ユニコード組み合わせ型」があるという．これを用いれば中期語をコンピュータで処理することが可能となるが，巻末資料１３を見て分かるように，古典ハングル字母は現代語字母の後に割り当てられており，字母の順番は가나다順ではない．従って，ユニコード組み合わせ型を 用いたとしても，字母を가나다順に並べ替える作業をしなければならず，この点においてユニコードの利点はあまりないというのが実情である．

参考文献

志部昭平(1992) 「諺解三綱行實圖研究 索引」，汲古書院

三田典玄(1990) 「実習C言語」，アスキー出版局

安岡孝一, 安岡素子(1999)「文字コードの 世界」，東京電気大学出版局

文獻研究會(1994) “釋譜詳節 文法形態 索引集”，太學社

서상규(1997) “飜譯老乞大 語彙索引”，도서출판 박이정

이준희, 정내권(1991) “컴퓨터속의 한글”, 정보시대

임인건(1990) “터보C정복”, 가남사

한국어정보처리연구소(1999) “C로 구현한 한글 코드 시스템 프로그래밍 가이드”, 도서출판 골드

資料1：KS C 5601 のコード割り当て

	A1 … FE
A1	特殊文字領域(1128字)
AD	
B0	ハングル領域(2350字)
C9	
CA	漢字領域(4888字)
FD	
FE	

資料2：完成型でのハングルの割り当て(抄)

	가	각	간	갇	갈	갉	갊
B0A0	B0A1	B0A2	B0A3	B0A4	B0A5	B0A6	B0A7
감	갑	값	갓	갔	강	갖	갗
B0A8	B0A9	B0AA	B0AB	B0AC	B0AD	B0AE	B0AF
같	갚	갛	개	객	갠	갤	갬
B0B0	B0B1	B0B2	B0B3	B0B4	B0B5	B0B6	B0B7
갭	갯	갰	갱	갸	갹	갼	걀
B0B8	B0B9	B0BA	B0BB	B0BC	B0BD	B0BD	B0BF

資料3：組み合わせ型コード

	初声	中声	終声
00			
01	(Fill)		(Fill)
02	ㄱ	(Fill)	ㄱ
03	ㄲ	ㅏ	ㄲ
04	ㄴ	ㅐ	ㄳ
05	ㄷ	ㅑ	ㄴ
06	ㄸ	ㅒ	ㄵ
07	ㄹ	ㅓ	ㄶ
08	ㅁ		ㄷ
09	ㅂ		ㄹ
0A	ㅃ	ㅔ	ㄺ
0B	ㅅ	ㅕ	ㄻ
0C	ㅆ	ㅖ	ㄼ
0D	ㅇ	ㅗ	ㄽ
0E	ㅈ	ㅘ	ㄾ
0F	ㅉ	ㅙ	ㄿ
10	ㅊ		ㅀ
11	ㅋ		ㅁ
12	ㅌ	ㅚ	
13	ㅍ	ㅛ	ㅂ
14	ㅎ	ㅜ	ㅄ
15		ㅝ	ㅅ
16		ㅞ	ㅆ
17		ㅟ	ㅇ
18			ㅈ
19			ㅊ
1A		ㅠ	ㅋ
1B		ㅡ	ㅌ
1C		ㅢ	ㅍ
1D		ㅣ	ㅎ
1E			
1F			

資料4：アレアコード

(太字はアレアハングル独自のコード)

	初声	中声	終声
00	**ㅂㄱ**	**古典**	**ㄿㅎ**
01	(Fill)	**古典**	(Fill)
02	ㄱ	(Fill)	ㄱ
03	ㄲ	ㅏ	ㄲ
04	ㄴ	ㅐ	ㄳ
05	ㄷ	ㅑ	ㄴ
06	ㄸ	ㅒ	ㄵ
07	ㄹ	ㅓ	ㄶ
08	ㅁ		ㄷ
09	ㅂ		ㄹ
0A	ㅃ	ㅔ	ㄺ
0B	ㅅ	ㅕ	ㄻ
0C	ㅆ	ㅖ	ㄼ
0D	ㅇ	ㅗ	ㄽ
0E	ㅈ	ㅘ	ㄾ
0F	ㅉ	ㅙ	ㄿ
10	ㅊ		ㅀ
11	ㅋ	**ㅚㅣ**	ㅁ
12	ㅌ	ㅚ	**ㅱ**
13	ㅍ	ㅛ	ㅂ
14	ㅎ	ㅜ	ㅄ
15	**ㅃㄷ**	ㅝ	ㅅ
16	**ㅃㅅ**	ㅞ	ㅆ
17	**ㅃㅈ**	ㅟ	ㅇ
18	**ㅱ**	**ㅞㅔ**	ㅈ
19	**ㅅㄱ**	**ㅟㅣ**	ㅊ
1A	**ㅅㄷ**	ㅠ	ㅋ
1B	**ㅆㅐ**	ㅡ	ㅌ
1C	**ㅆㅈ**	ㅢ	ㅍ
1D	**ㅿ**	ㅣ	ㅎ
1E	**ㅇ**	.	**ㅿ**
1F	**ㆆㆆ**	**.ㅣ**	**ㅇ**

「古典」は完成型古典コードに
割り当てられる部分

資料5：中期語テクスト入力の一例（「月印釈譜第一」の冒頭部分）

(01001a1w) 月印千江之曲第一 (01001a2w) ㊟

(01001a2n) [부:톄 百億 世界·예 化身·ᄒᆞ·야 敎化 ·ᄒᆞ샤·미 ·ᄃᆞ리 ·즈믄 (01001a3n) ᄀᆞᄅᆞ·매 비·취요·미 ·ᄀᆞᆮᄒᆞ·니·라.

(01001a3n) 第·ᄂᆞᆫ 次第·라.]

(01001a4s) 釋譜詳節第一

(01001a5w) 其一

(01001a6w) 巍巍 釋迦佛 無量 (01001a7w) 無邊 功德·을 劫劫·에 (01001b1w) 어·느 :다 ᄉᆞᆯ·ᄫᅳ·리. ㊟

資料6：KWIC索引の原ファイル（資料5のテクストを用いた場合）

```
2: (01001a2n)                          [|부:톄      | 百億 世界·예 化身·ᄒᆞ·야
2: (01001a2n)              [부:톄 百億 世界|·예      | 化身·ᄒᆞ·야 敎化·ᄒᆞ샤·미
2: (01001a2n)       부:톄 百億 世界·예 化身|·ᄒᆞ·야   | 敎化·ᄒᆞ샤·미 ·ᄃᆞ리 ·즈믄
2: (01001a2n)        世界·예 化身·ᄒᆞ·야 敎化|·ᄒᆞ샤·미 | ·ᄃᆞ리 ·즈믄 ᄀᆞᄅᆞ·매 비
2: (01001a2n)       化身·ᄒᆞ·야 敎化·ᄒᆞ샤·미 |·ᄃᆞ리   | ·즈믄 ᄀᆞᄅᆞ·매 비·취요·미
2: (01001a2n)  ·ᄒᆞ·야 敎化·ᄒᆞ샤·미 ·ᄃᆞ리 |·즈믄     | ᄀᆞᄅᆞ·매 비·취요·미 ·ᄀᆞᆮ
2: (01001a3n)  敎化·ᄒᆞ샤·미 ·ᄃᆞ리 ·즈믄 |ᄀᆞᄅᆞ·매    | 비·취요·미 ·ᄀᆞᆮᄒᆞ·니·라.
2: (01001a3n)   ·미 ·ᄃᆞ리 ·즈믄 ᄀᆞᄅᆞ·매 |비·취요·미 | ·ᄀᆞᆮᄒᆞ·니·라.
2: (01001a3n)  ·즈믄 ᄀᆞᄅᆞ·매 비·취요·미 |·ᄀᆞᆮᄒᆞ·니·라 |.
3: (01001a3n)                          第|·ᄂᆞᆫ      | 次第·라.]
3: (01001a3n)                    第·ᄂᆞᆫ 次第|·라    |.]
6: (01001a7w)    釋迦佛 無量 無邊 功德|·을        | 劫劫·에 어·느 :다 ᄉᆞᆯ·ᄫᅳ
6: (01001a7w)    無量 無邊 功德·을 劫劫|·에       | 어·느 :다 ᄉᆞᆯ·ᄫᅳ·리. ㊟
6: (01001b1w)  量 無邊 功德·을 劫劫·에 |어·느      | :다 ᄉᆞᆯ·ᄫᅳ·리. ㊟
6: (01001b1w)  邊 功德·을 劫劫·에 어·느 |:다      | ᄉᆞᆯ·ᄫᅳ·리. ㊟
6: (01001b1w)   德·을 劫劫·에 어·느 :다 |ᄉᆞᆯ·ᄫᅳ·리 |. ㊟
```

資料7：趙コード

(1) 子音コード

Fill	ㄱ	ㄲ	ㄳ	ㄴ	ㄵ	ㄶ	ㄷ	ㄸ	ㄹ	ㄺ	ㄻ	ㄼ	ㄽ	ㄾ	ㄿ
01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10
ㅀ	ㅁ	ㅂ	ㅃ	ㅄ	ㅅ	ㅆ	ㅇ	ㅈ	ㅉ	ㅊ	ㅋ	ㅌ	ㅍ	ㅎ	ㅏ
11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20
ㅐ	ㅑ	ㅒ	ㅓ	ㅔ	ㅕ	ㅖ	ㅗ	ㅘ	ㅙ	ㅚ	ㅛ	ㅜ	ㅝ	ㅞ	ㅟ
21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	30
ㅠ	ㅡ	ㅢ	ㅣ	ㅥ	ㅦ	ㅧ	ㅨ	ㅩ	ㅪ	ㅫ	ㅬ	ㅭ	ㅮ	ㅯ	ㅰ
31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	40
ㅱ	ㅲ	ㅳ	ㅴ	ㅵ	ㅶ	ㅷ	ㅸ	ㅹ	ㅺ	ㅻ	ㅼ	ㅽ	ㅾ	ㅿ	ㆀ
41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50
ㆁ	ㆂ	ㆃ	ㆄ	ㆅ	ㆆ	ㆇ	ㆈ	ㆉ	ㆊ	ㆋ	ㆌ	ㆍ	ㆎ	0	0
51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	70
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	80
0	0	0	0	0	0	ㅇ									
81	82	83	84	85	86	87									

(2) 母音コード

Fill 01	ㅏ 02	ㅐ 03	ㅑ 04	ㅒ 05	ㅓ 06	ㅔ 07	ㅕ 08	ㅖ 09	ㅗ 0A	ㅘ 0B	ㅙ 0C	ㅚ 0D	ㅛ 0E	ㅜ 0F	ㅝ 10
ㅞ 11	ㅟ 12	ㅠ 13	ㅡ 14	ㅢ 15	ㅣ 16	ㅥ 17	ㅦ 18	ㅧ 19	ㅨ 1A	ㅩ 1B	ㅪ 1C	ㅫ 1D	ㅬ 1E	ㅭ 1F	ㅮ 20
ㅯ 21	ㅰ 22	ㅱ 23	ㅲ 24	ㅳ 25	ㅴ 26	ㅵ 27	ㅶ 28	ㅷ 29	ㅸ 2A	ㅹ 2B	ㅺ 2C	ㅻ 2D	ㅼ 2E	ㅽ 2F	ㅾ 30
ㅿ 31	ㆀ 32	ㆁ 33	ㆂ 34	ㆃ 35	ㆄ 36	ㆅ 37	ㆆ 38	ㆇ 39	ㆈ 3A	ㆉ 3B	ㆊ 3C	ㆋ 3D	ㆌ 3E	ㆍ 3F	ㆎ 40
ㆍㅣ 41	ㆍㅡ 42	ㆍㅜ 43													

資料 8 ：アレアコードの趙コードへの変換テーブル

(1) アレア組み合わせ型コードの字母コード変換テーブル

```
unsigned char Choseong[32] = {   // 初声
   0x37, 0x01, 0x02, 0x03, 0x07, 0x11, 0x13, 0x15,  // ㅂ, Fill, ㄱ, ㄲ, ㄴ, ㄷ, ㄸ, ㄹ
   0x2c, 0x36, 0x3b, 0x49, 0x51, 0x5f, 0x70, 0x72,  // ㅁ, ㅂ, ㅃ, ㅅ, ㅆ, ㅇ, ㅈ, ㅉ,
   0x77, 0x7c, 0x7d, 0x7e, 0x81, 0x39, 0x3c, 0x42,  // ㅊ, ㅋ, ㅌ, ㅍ, ㅎ, ㅱ, ㅲ, ㅳ,
   0x47, 0x4a, 0x4c, 0x4f, 0x54, 0x5e, 0x6d, 0x86   // ㅸ, ㅺ, ㅼ, ㅾ, ㆀ, ㆁ, ㆆ, ㆆㆆ
};

unsigned char Jungseong[32] = {   // 中声
   0x00, 0x00, 0x01, 0x02, 0x05, 0x06, 0x09, 0x0a,  // Null, Null, Fill, ㅏ, ㅐ, ㅑ, ㅒ, ㅓ,
   0x00, 0x00, 0x0e, 0x0f, 0x12, 0x13, 0x14, 0x15,  // Null, Null, ㅖ, ㅕ, ㅘ, ㅙ, ㅚ, ㅛ,
   0x00, 0x21, 0x1b, 0x1c, 0x22, 0x25, 0x27, 0x2a,  // Null, ㅵ, ㅶ, ㅷ, ㅹ, ㅻ, ㅽ, ㅿ,
   0x2f, 0x32, 0x2b, 0x33, 0x36, 0x38, 0x3f, 0x42   // ㆄ, ㆅ, ㅸ, ㅡ, ㅢ, ㅣ, ㆍ, ㆍㅣ
};

unsigned char Jongseong[32] = {   // 終声
   0x2b, 0x01, 0x02, 0x03, 0x05, 0x07, 0x0e, 0x10,  // ㄻ, Fill, ㄱ, ㄲ, ㄳ, ㄴ, ㄵ, ㄶ,
   0x11, 0x15, 0x16, 0x1c, 0x1f, 0x23, 0x28, 0x29,  // ㄷ, ㄹ, ㄺ, ㄻ, ㄼ, ㄽ, ㄾ, ㄿ,
   0x2a, 0x2c, 0x47, 0x36, 0x3c, 0x49, 0x51, 0x5f,  // ㄶ, ㅁ, ㅱ, ㅂ, ㅲ, ㅅ, ㅆ, ㅇ,
   0x70, 0x77, 0x7c, 0x7d, 0x7e, 0x81, 0x5e, 0x6d   // ㅈ, ㅊ, ㅋ, ㅌ, ㅍ, ㅎ, ㅿ, ㆁ
};
```

(2) 完成型古典コードの字母コード変換テーブル

```
long Hcomp[0x749] = {
   0x020101, 0x030101, 0x050101, 0x070101, 0x0e0101, 0x100101, 0x110101, 0x130101,
   0x150101, 0x160101, 0x1c0101, 0x1f0101, 0x230101, 0x280101, 0x290101, 0x2a0101,
                                          :
   0x87325f, 0x873f01, 0x873f07, 0x874201, 0x874202, 0x87425f, 0x87426d, 0x873001,
   0x87305f
};
/* ㄱ, ㄲ, ㄳ, ㄴ, ㄵ, ㄶ, ㄷ, ㄸ,
```

```
   ㄹ, ㄺ, ㄻ, ㄼ, ㄽ, ㄾ, ㄿ, ㅀ,
                    ⋮
   ㅎ, ㅇ, ㅈ, ㅊ, ㅋ, ㅌ, ㅍ, ㅎㅎ,
   ㅏㅏ */
```

資料 9 ：キーワードのテンポラリファイル生成関連ルーチン

(1) 基本ルーチン

```
void gener_key(void)
{
   int ini, cen, fin, i;
   char tone=1;
   unsigned short code;

   Count = 0; // 行番号を表す変数の初期化
   // キーワードの行番号を書き出す
   fwrite(&Kwicline, 4, 1, Fp_k1); // Fp_k1 = 正順用テンポラリファイル
   fwrite(&Kwicline, 4, 1, Fp_k2); // Fp_k2 = 逆順用テンポラリファイル

   for(i=0; Keyword[i] != NULL; i++) {
      /* アレアコードによる文字列 Keyword から 1 文字ずつ取得し文字種を判定する */
      if(Keyword[i] == 0x85) {   //「·(傍点)」のとき
         tone = 2;
      } else if(Keyword[i] == ':') {   //「:(傍点)」のとき
         tone = 3;
      } else if(hwpnatja(Keyword[i])) {   // ハングル字母のとき
         gener_natja(Keyword[i], tone);
         tone = 1;
      } else {   // ハングル 1 文字のとき
         // ハングルを字母に分解する
         ini = (Keyword[i] >> 10) & 0x1F;   // 初声
         cen = (Keyword[i] >> 5) & 0x1F;    // 中声
         fin = Keyword[i] & 0x1F;           // 終声
         if (cen == 0 || cen == 1) {   // 完成型古典コードのとき
            gen_cpl(ini, cen, fin, tone);
         } else {                      // 組み合わせ型コードのとき
            gen_cbn(ini, cen, fin, tone);
         }
         tone = 1;
      }
   }
   write_key(); // キーワードをテンポラリファイルに書き出すルーチンへ
}
```

(2) 完成型古典コードの趙コードへの変換ルーチン

```
void gen_cpl(int ini, int cen, int fin, char tone)
{
   long code=0;
   unsigned short hwpcode;

   hwpcode = (cen << 10) | (ini << 5) | fin;  // アレアコードを生成する
   code = Hcomp[hwpcode];  // 変換テーブルを用いて趙コードに変換する

   // キーワード・傍点情報を配列 Hangul, Tone に一時的に書き出す
   // 正順
   Hangul[0][Count] = code;
   Tone[0][Count] = tone;
   // 逆順
   Hangul[1][Count] = (code >> 16) | (code & 0xff00) | ((code & 0xff) << 16);
   Tone[1][Count] = tone;

   Count++; // 行番号を1つ進める
}
```

(3) アレア組み合わせ型コードの趙コードへの変換ルーチン

```
void gen_cbn(int ini, int cen, int fin, char tone)
{
   unsigned long a, b, c, code=0;

   // 変換テーブルを用いて各字母を趙コードに変換する
   a = Choseong[ini];
   b = Jungseong[cen];
   c = Jongseong[fin];
   code = (a << 16) | (b << 8) | c;  // 1文字の趙コードを生成する

   // キーワード・傍点情報を配列 Hangul, Tone に一時的に書き出す
   // 正順
   Hangul[0][Count] = code;
   Tone[0][Count] = tone;
   // 逆順
   Hangul[1][Count] = (c << 16) | (b << 8) | a;
   Tone[1][Count] = tone;

   Count++; // 行番号を1つ進める
}
```

(4) キーワード情報をテンポラリファイルに書き出すルーチン

```
void write_key(void)
{
   int i, a, b;
   long tmp_l;
   char tmp_c;

   // 逆順のキーワード情報をひっくり返す
   for(a=0, b=Count - 1; a<b; a++, b--) {
      // キーワード
      tmp_l = Hangul[1][b];
      Hangul[1][a] = Hangul[1][b];
      Hangul[1][b] = tmp_l;
      // 傍点
      tmp_c = Tone[1][a];
      Tone[1][a] = Tone[1][b];
      Tone[1][b] = tmp_c;
   }

   // キーワード情報の書き出し
   for(i=0; i<40; i++) {
      fwrite(&Hangul[0][i], 4, 1, Fp_k1);
      fwrite(&Hangul[1][i], 4, 1, Fp_k2);
   }
   // 傍点情報の書き出し
   for(i=0; i<40; i++) {
      fwrite(&Tone[0][3-(i%4)+(i/4)*4], 1, 1, Fp_k1);
      fwrite(&Tone[1][3-(i%4)+(i/4)*4], 1, 1, Fp_k2);
   }
}
```

資料１０：文脈切り出しルーチン（抄）

```
void backward(void) {
   int i, clen;

   for(i=0; i<MAXBAK-1; i++) {
      Backward[i] = Sentence[Backhead+i];
      if(Backward[i] == NULL) {
         break;
      } else {
         if(hwpban(Backward[i]))   // 半角文字のとき
            clen += charwidth(Backward[i]);   // フォント幅を算出
```

```
            else if(hwp2byte(Backward[i]))   // 全角文字のとき
                clen += Hgl;   // Hgl = 全角文字幅
            else
                clen += Ban;   // Ban = 半角文字幅
        }
        if(clen >= Jisu) break; // 指定した幅を超えたら文脈の切り出しを中止する
    }
}
```

資料１１：キーワード用テンポラリファイルの構成とその処理の模式図

一連番号	キーワード	傍点情報
1	부톄	02
2	예	1
3	ᄒᆞ야	11
4	ᄒᆞ샤미	101
5	ᄃᆞ리	10
6	ᄌᆞ믄	10
7	ᄀᆞᄅᆞ매	011
8	비취요미	0101
9	ᄀᆞᆮᄒᆞ니라	1011

一連番号	キーワード	傍点情報
7	ᄀᆞᄅᆞ매	011
9	ᄀᆞᆮᄒᆞ니라	1011
5	ᄃᆞ리	10
1	부톄	02
8	비취요미	0101
2	예	1
6	ᄌᆞ믄	10
44	ᄒᆞ샤미	101
3	ᄒᆞ야	11

資料１２：KWIC索引の完成ファイル（資料５のテクストを用いた場合）

(1) 正順

```
2: (01001a3n)  敎化·ᄒᆞ샤·미 ·ᄃᆞ리 ·ᄌᆞ믄 |ᄀᆞ·ᄅᆞ·매   | 비·취요·미 ·ᄀᆞᆮᄒᆞ·니·라.
2: (01001a3n)  ·ᄌᆞ믄 ᄀᆞ·ᄅᆞ·매 비·취요·미 |·ᄀᆞᆮᄒᆞ·니·라 |.
3: (01001a3n)                          第|·ᄂᆞᆫ        | 次第·라.]
6: (01001b1w)  邊 功德·을 劫劫·에 어·느 |:다        | 술·ᄇᆞᆯ·리. ㊟
2: (01001a2n)   化身·ᄒᆞ·야 敎化·ᄒᆞ샤·미 |·ᄃᆞ리     | ·ᄌᆞ믄 ᄀᆞ·ᄅᆞ·매 비·취요·미
3: (01001a3n)                  第·ᄂᆞᆫ 次第|·라      |.]
2: (01001a2n)                          [|부:톄     | 百億 世界·예 化身·ᄒᆞ·야
2: (01001a3n)  ·미 ·ᄃᆞ리 ·ᄌᆞ믄 ᄀᆞ·ᄅᆞ·매 |비·취요·미 | ·ᄀᆞᆮᄒᆞ·니·라.
6: (01001b1w)  德·을 劫劫·에 어·느 :다 |술·ᄇᆞᆯ·리   |. ㊟
6: (01001b1w)  量 無邊 功德·을 劫劫·에 |어·느       | :다 술·ᄇᆞᆯ·리. ㊟
6: (01001a7w)   無量 無邊 功德·을 劫劫|·에        | 어·느 :다 술·ᄇᆞᆯ·리. ㊟
2: (01001a2n)          [부:톄 百億 世界|·예      | 化身·ᄒᆞ·야 敎化·ᄒᆞ샤·미
6: (01001a7w)    釋迦佛 無量 無邊 功德|·을       | 劫劫·에 어·느 :다 술·ᄇᆞᆯ
2: (01001a2n)  ·ᄒᆞ·야 敎化·ᄒᆞ샤·미 ·ᄃᆞ리 |·ᄌᆞ믄     | ᄀᆞ·ᄅᆞ·매 비·취요·미 ·ᄀᆞᆮ
2: (01001a2n)   世界·예 化身·ᄒᆞ·야 敎化|·ᄒᆞ샤·미     | ·ᄃᆞ리 ·ᄌᆞ믄 ᄀᆞ·ᄅᆞ·매 비
2: (01001a2n)   부:톄 百億 世界·예 化身|·ᄒᆞ·야      | 敎化·ᄒᆞ샤·미 ·ᄃᆞ리 ·ᄌᆞ믄
```

(2) 逆順

```
6: (01001b1w) 邊 功德·을 劫劫·에 어·느 |        :다| 술·ᄇᆞᆯ·리. ㊟
3: (01001a3n)                  第·ᄂᆞᆫ 次第|      ·라|.]
2: (01001a3n) ·ᄌᆞ믄 ᄀᆞ·ᄅᆞ·매 비·취요·미 |·ᄀᆞᆮᄒᆞ·니·라|.
2: (01001a3n) 敎化·ᄒᆞ샤·미 ·ᄃᆞ리 ·ᄌᆞ믄 |   ᄀᆞ·ᄅᆞ·매| 비·취요·미 ·ᄀᆞᆮᄒᆞ·니·라.
2: (01001a2n)  부:톄 百億 世界·예 化身|     ·ᄒᆞ·야| 敎化·ᄒᆞ샤·미 ·ᄃᆞ리 ·ᄌᆞ믄
6: (01001a7w)   無量 無邊 功德·을 劫劫|       ·에| 어·느 :다 술·ᄇᆞᆯ·리. ㊟
```

2: (01001a2n) [부:톄 百億 世界| ·예| 化身·ᄒᆞ·야 敎化·ᄒᆞ샤·미
2: (01001a2n) [| 부:톄| 百億 世界·예 化身·ᄒᆞ·야
6: (01001b1w) 量 無邊 功德·을 劫劫·에 | 어·느| :다 ᄉᆞᆯ·ᄫᅳ·리. ㊟
2: (01001a2n) 化身·ᄒᆞ·야 敎化·ᄒᆞ샤·미 | ·드리| ·즈믄 ᄀᆞᄅᆞ·매 비·취요·미
6: (01001b1w) 德·을 劫劫·에 어·느 :다 | ᄉᆞᆯ·ᄫᅳ·리. ㊟
2: (01001a2n) 世界·예 化身·ᄒᆞ·야 敎化| ·ᄒᆞ샤·미| ·드리 ·즈믄 ᄀᆞᄅᆞ·매 비
2: (01001a3n) ·미 ·드리 ·즈믄 ᄀᆞᄅᆞ·매 | 비·취요·미| ·ᄀᆞᆮᄒᆞ·니·라.
2: (01001a2n) ·ᄒᆞ·야 敎化·ᄒᆞ샤·미 ·드리 | ·즈믄| ᄀᆞᄅᆞ·매 비·취요·미 ·ᄀᆞᆮ
3: (01001a3n) 第| ·ᄂᆞᆫ| 次第·라.]
6: (01001a7w) 釋迦佛 無量 無邊 功德| ·을| 劫劫·에 어·느 :다 ᄉᆞᆯ·ᄫᅳ

資料１３：ユニコード組み合わせ型コード

	110	111	112	113	114	115	116	117	118	119	11A	11B	11C	11D	11E	11F
0	ᄀ	ᄐ	ᄇᄃ	ᄉᄅ	ᄋ	ᄌ	HJF	게	데	레	ᅮ	ᄅᆡ	ᄐᆡ	ᄑᆡ	ᄑᆪ	ᄋᆡ
1	ᄁ	ᄑ	ᄇᄉ	ᄉᄆ	ᄋᆡ	ᄍ	ᅡ	ᄀᆡ	ᄃᆡ	ᄅᆡ	ᅴ	ᄅᆫ	ᄑ	ᄑᆫ	ᄑᆮ	ᄋᆺ
2	ᄂ	ᄒ	ᄇᆡ	ᄉᆡ	ᄋᆮ	ᄎ	ᅢ	ᄁᆞ	ᄉᆞ	ᄅᆡ	ᅟ	ᄅᆡ	ᄒ	ᄑᆺ	ᄒᆼ	ᄋᆸ
3	ᄃ	ᄂᆡ	ᄇᆮ	ᄊᆡ	ᄋᆫ	ᄎᆞ	ᅣ	ᅳ	ᅮ	ᄭᅮ		ᄅᆺ	ᄀᆨ	ᄅᆡᆨ	ᄇᆨ	ᄑᆨ
4	ᄃᆞ	ᄂᆞ	ᄇᆡᆫ	ᄊᆞ	ᄋᆯ	ᄎᆞᆫ	ᅤ	ᄀᆞ	싸	끼		ᄅᆮ	ᄆᆨ	ᄅᆡᆷ	ᄇᆫ	ᄑᆼ
5	ᄅ	ᄂᆞᆫ	ᄇᆞᆫ	ᄉᆞᆫ	ᄋᆷ	ᄎᆞᆮ	ᅥ	ᅵ	쌔	두		ᄅᆯ	ᄂᆡ	ᄅᆡᆫ	ᄇᆮ	ᄒᆡ
6	ᄆ	ᄂᆡᆫ	ᄇᆞᆫ	ᄉᆞᆨ	ᄋᆸ	ᄑᆡ	ᅦ	ᄀᆞᆫ	써	ᄂᆞᆫ		ᄅᆷ	ᄃᆮ	ᄅᆡᆨ	ᄇᆼ	ᄒᆫ
7	ᄇ	ᄃᆡ	ᄇᆨ	ᄉᆞᆫ	ᄋᆹ	ᄑᆼ	ᅧ	ᄀᆞᆫ	ᄭᅩ	둑		ᄆ	ᄂᆞ	ᄅᆸ	ᄉᆡ	ᄒᆷ
8	ᄇᆡ	ᄏᆡ	ᄇᆞᆨ	ᄉᆡ	ᄋᆺ	ᄒᆼ	ᅨ	ᄀᆞᆫ	씨	가	ᄀᆡ	ᄇ	ᄂᆞᆷ	ᄅᆡᆨ	ᄉᆮ	ᄒᆸ
9	ᄉ	ᄐᆫ	ᄇᆮ	ᄉᆞᆫ	ᄋᆺ	ᄋ	ᅩ	ᄀᆞᆯ	ᄀᆞ	가	ᄀᆡ	ᄇᆞᆨ	ᄂᆞᆫ	ᄅᆫ	ᄉᆨ	ᄋ
A	ᄊᆞ	ᄅᆫ	ᄇᆞᆫ	ᄉᆞᆫ	ᄋᆽ		ᅪ	ᄀᆞᆮ	개	ᄀᆞ	ᄀᆞᆺ	ᄉ	ᄃᆡ	ᄆᆡ	ᄉᆡᆨ	
B	ᄋ	ᄒᆼ	ᄇᆼ	ᄉᆞᆨ	ᄋᆾ		ᅫ	ᄀᆞᆫ	ᄁᆞ	ᄀᆞ	ᄂᆞ	ᄊᆞ	ᄃᆮ	ᄆᆮ	ᄉ	
C	ᄌ	ᄆᆡ	ᄇᆼᆼ	ᄉ	ᄋ		ᅬ	ᄀᆞᆯ	게	ᄀᆞᆯ	ᄂᆞᆺ	ᄋ	ᄅᆡᆺ	ᄆᆡ	ᄋᆡ	
D	ᄌᆞ	ᄒᆼ	ᄉᆡ	ᄊᆞ	ᄌᆞ		ᅭ	ᄀᆞᆸ	고	ᄂᆡ	ᄂᆞᆫ	ᄌ	ᄀᆫ	ᄆᆞ	ᄋᆷ	
E	ᄎ	ᄇᆡ	ᄉᆡ	ᄉᆞᆺ	ᄌ		ᅮ	ᄀᆞᆺ	구	과	ᅟ	ᄃ	ᄎ	ᄀᆫ	ᄆᆺ	ᄋᆼ
F	ᄏ	ᄇᆫ	ᄉᆫ	ᄊᆞ	ᄍ	HCF	거	고	궤	너	ᄅ	ᄏ	ᄒᆼ	ᄆᆞ	ᄋᆨ	