ローマ字-カナ変換用サブプログラムRKHKNについて

石原, 好宏
山口大学工業短期大学部情報処理工学科

https://doi.org/10.15017/1472555

出版情報：九州大学大型計算機センター広報. 12 (1), pp.20-28, 1979-03-25. 九州大学大型計算機センター
バージョン：
権利関係：

ローマ字ーカナ変換用サブプログラムRKHKNについて

石原好宏*

1. まえがき

日本語を電子計算機で処理する必要性は最近ますます増えている．しかし現在のところ本センターでは漢字の入出力ができない．従って日本語はカナ文字かローマ字で表記せざるを得ない．その場合，出力に関してはセンター内にカナ文字付きライプリンターが備えられているので問題はないが，入力に関しては問題がある．入力装置として端末タイプライターかカナ文字付きカード穿孔機を使えば確かにカナ文字入力が可能ではある．しかし多くのユーザーにとってこれらの装置はそれほど身近かに使えるわけではなく，又，日本語を機械処理する場合ローマ字で表記してある方が好都合であることも少なくない.[1] 一方，処理された日本語を出力する場合は一般にカナ文字の方が読み易い．従って現在のところ日本語処理の場合の使用文字は入力から内部処理までをローマ字表記，出力をカナ文字表記とすることが少なくないであろう．そのために本センターではFACOM230－75の時代にローマ字ーカナ変換プログラムKANAがプログラムライブラリーに登録され，ユーザーに開放されていた.[2] しかし，このプログラムはFORTRANで書かれていたために本センターのCPUがM－190に変更されたことに伴い若干の書き直しが必要となったようであるが，現在までのところが実施されていない．

一方筆者は日本語終端表現生成システム[3]の一部として同じくローマ字ーカナ変換用サブプログラムとしてRKHKNを作成した.[4] これは，ローマ字表記が撥音（ン）と促音（つまる音ッ）を例外としてすべて〃子音字部＋母音字部〃※の構造をもっていることを利用して日本語音の単位に相当するローマ字列を文頭側から順次切り出し，ローマ字ーカナ変換表を参照して対応するカナ文字表記を求めていくことを特色とする．この方法は，変換規則を一つの有限オートマトンで表現し，それを適用する手続きにAhoらのアルゴリズム[5]を応用するKANAの方法とは大きく異なる.[2] RKHKNも最初はFORTRANで書いたが，ローマ字ーカナ変換表の初期設定，プログラムの互換性などを考慮してCOBOLで書き直した．このたびこれが本センターのプログラムライブラリーの一つとして登録されたので，その概要と使い方を御紹介する．

2. ローマ字による日本語文の表記

一般の日本語文の構成要素の中にはカナ文字で表記すべきものの外に，数字，句読点，その外の特殊記号，更には英語その外の綴りなど，カナ変換後も原文のままにしておきたいものが含まれる．そこでRKHKNが想定しているローマ字文の表記方式を二群に分けて整理する．

2.1 ローマ字の表記方式

＊山口大学工業短期大学部情報処理工学科

※A，I，U，E，Oを母音字，それ以外の英文字を子音字と呼ぶ．

現在ローマ字の表記方式としては訓令式の外に，日本式，ヘボン式も相当に実用されている．従って変換プログラムは，任意の表記方式で綴られたローマ字文をできる限り区別なく処理してくれるほど実用性が高いと言える．一方，人間向きに作られている表記方式をそのままで機械処理しようとするとあいまいさが生じて簡単な変換手続きを作ることは難しい．従ってここでは，長音の表記など一部で慣用をまげて機械向きに修正することによって表記の解釈に一意性を与え，変換手続きを簡単化する．以上の立場に立って，RKHKNが許す日本語音のローマ字表記方式を次のように整理する．

A1）日本語音は促音と撥音を除きいずれも〃子音字部＋母音字部〃の構造をもつ．このうち子音字部は英文字0～2文字で，母音字部は常に英文字1文字で表記される．一方，これらの日本語音はカナ文字1～3文字で表記される．但しこの中には濁点や半濁点も1文字として数えられている．

A2）促音は後続音を表す英文字列の先頭文字をもう一つ重ねることによって表記する．但し後続音の先頭文字がCのときは，Cを重ねてもよいし，Cの前にTを置いてもよいものとする．

〔例〕MACCHI（マッチ），　MATCHI（マッチ），
MATTI（マツチ），NIPPON（ニッポン），…

A3）撥音はその後続音の先頭文字&の種類によって次の三つの表記法を使い分ける．

a）&が空白のとき；N又はN－．

b）&がY又は母音字のとき；N－．
これは拗音やナ行音との区別のための措置である．

c）&がY以外の子音字のとき；N．
但し，&がB，P，Mのいずれかであればそのときに限ってNの代りにMを使ってもよい．

A4）長音は適当な母音字（例えば，OO̲KINA　O̲UJI；オオキナ　オウジ），又は－（ハイフン，例えば，GAKKO－；ガッコー）を長音化すべき母音字の直後に付けることによって表す．従って，例えばTÔKYÔ（トウキョウ：東京）など長音部にハを付けたり，KYOTO（キョウト：京都）の片方だけを慣習的に長音として読むようなことはしない．※

以上のようにすれば，A1に述べた日本語音は，子音字部が32種類，母音字部が5種類の英文字列で表記できる．これを組合せると160種類の日本語音が表記できることになるが，必ずしもすべての組合せに対応して日本語音が存在するわけではない．日本語音のローマ字表記とカナ文字表記とのこのような立場での対応表を表1に示す．この表がローマ字－カナ変換表であり，以下では単に変換表と呼ぶ．なお，子音字部と母音字部との組合せに対応する日本語音がないところはカナ文字表記欄に＊を入れて示している．

2.2　特殊記号と特殊文字列の表記

ここで特殊記号とは，英文字26種類を除くすべての文字・記号である．このうちカナ表記のた

※このため，東京はTOUKYOU，京都はKYOUTOなどと表記することになる．

めの濁点と半濁点を除けばいずれもローマ字表記，カナ表記の中で共通に使われると考えてよい．一方，特殊文字列とは，ローマ字としては読めない文字列（例えばFORTRAN，PROGRAMなど），及びローマ字として読める文字列（例えばMIKE，MADEなど）であっても原文中の綴りをそのまま残したい文字列である．変換処理の一意性を保つためにこれらは次のように表記する．

B1） 特殊記号はそのまま書いてよい．特殊記号の連続（例えば数字列）の場合も同様である．

B2） ローマ字列中にある綴り $\alpha_1\alpha_2\cdots\alpha_m$ をそのままカナ文字列の中に残したければ，綴りの両端に＃を付けて＃$\alpha_1\alpha_2\cdots\alpha_m$＃と表記する．このとき$\alpha_i$は＃を除く任意の特殊記号や英文字でよい．なお，綴り$\alpha_1\alpha_2\cdots\alpha_m$がローマ字として読めなければ両端に＃を付ける必要はない．しかし，例えばTSARINAのように日本語音にない表記（ここではTSA）を含んでいても，それの子音字部と見なされる綴り（ここではTS）が変換表に登録されており，それに母音（ここではA）が後続しておれば，変換表の対応欄に＊が格納されており通常の処理が続けられるので注意が必要である．

	A	I	U	E	O
	ア	イ	ウ	エ	オ
K	カ	キ	ク	ケ	コ
S	サ	シ	ス	セ	ソ
T	タ	チ	ツ	テ	ト
N	ナ	ニ	ヌ	ネ	ノ
H	ハ	ヒ	フ	ヘ	ホ
M	マ	ミ	ム	メ	モ
Y	ヤ	＊	ユ	＊	ヨ
R	ラ	リ	ル	レ	ロ
W	ワ	イ	ウ	エ	オ
G	ガ	ギ	グ	ゲ	ゴ
Z	ザ	ジ	ズ	ゼ	ゾ
D	ダ	ヂ	ヅ	デ	ド
B	バ	ビ	ブ	ベ	ボ
P	パ	ピ	プ	ペ	ポ
F	ファ	フイ	フ	フエ	フォ
J	ジャ	ジ	ジュ	ジェ	ジョ
KY	キャ	＊	キュ	＊	キョ
GY	ギャ	＊	ギュ	＊	ギョ
SH	シャ	シ	シュ	シエ	ショ
SY	シャ	＊	シュ	シエ	ショ
DY	ヂャ	＊	ヂュ	ヂエ	ヂョ
CH	チャ	チ	チュ	チエ	チョ
NY	ニャ	＊	ニュ	ニエ	ニョ
HY	ヒャ	＊	ヒュ	ヒエ	ヒョ
BY	ビャ	＊	ビュ	ビエ	ビョ
PY	ピャ	＊	ピュ	ピエ	ピョ
MY	ミャ	＊	ミュ	ミエ	ミョ
RY	リャ	＊	リュ	リエ	リョ
ZY	ジャ	＊	ジュ	ジエ	ジョ
JY	ジャ	＊	ジュ	ジエ	ジョ
TS	＊	＊	ツ	＊	＊

表1 ローマ字－カナ変換表

3. 変換処理の概要

ここでは2.1及び2.2の表記方式に則って書かれた日本語のローマ字文をカナ変換するための変換規則をまとめ，それに基づく変換処理の手続きを述べる．

3.1 変換規則

C1） 英文字列については次の処理を行う．

C1.1） 子音字部としては通常2文字以内の連鎖だけを許す．但し次のような場合にはそれぞれについて個別処理を行う．

ィ）N又はM以外の同一子音字の重複および文字列TCの場合には左側の子音字を促音のツと読む．

ロ） NN，MB，MP，MMの子音字列の場合には左側の子音字を撥音のンと読む．

子音字列は上記ィ，ロの場合を除き後続の母音字が検出されるまでカナ文字への変換処理を保留する．なお，上記ィ，ロ以外で子音字が3文字以上連鎖していることを検出したらその文字列は特殊文字列と見なして処理する（3.2参照）．

C1.2） 母音字は1文字検出されるたびに，先に保留されていた子音字列（0～2文字の長さをもつ）と合せてカナ文字への変換処理を行う．この処理は保留されている子音字列の長さが0か否かによって次のように分かれる．

イ）子音字列の長さが0のときは，今検出された母音字が属する段（母音の字種）だけを調べ，変換表の第1行（ア行）の該当段からカナ文字列を読み取ってカナ文字文領域※へ転送する．

ロ）子音字列の長さが1か2のときは，まずそれが変換表のどの行に該当するかを調べ，次いで母音字がどの段に該当するかを調べる．こうして対応したカナ文字列を変換表から読み取ってカナ文字文領域へ転送する．

C2）特殊記号，特殊文字列については次の処理を行う．

C2.1）特殊記号は，＃と－を除いて，それが検出されるたびに直ちにそのままカナ文字領域へ転送する．

C2.2）二つの＃で囲まれた部分は特殊文字列と見なし，＃を取り除いた上で，そのままカナ文字文領域へ転送する．

C2.3）－（ハイフン）はNの直後にあるとき（N－）に限りNと合せてンと読む．その外の場合は通常の特殊記号と見なしてそのままカナ文字文領域へ転送する．

3.2　変換処理の手続き

この処理手続きは3.1に述べた変換規則を手続き的に記述し，辞書類としては表1に示した変換表だけを用いることを特色とする．処理手続きの概略は次のとおりである．なお，ローマ字原文はあらかじめローマ字文領域R－GBNに格納され，その文字数も分っているものとする．

まず，変換処理された結果を格納するカナ文字文領域K－GBNの全体を空白に，又カナ文字カウンターを0に初期化する．このカウンターはK－GBNに文字列が転送されるたびにその文字数を数え，累計を行う．

次に，R－GBNから1文字だけ被験文字領域HIKEN-MJ（容量1文字）へ転送しその字種を調べる．この転送はローマ字原文の文頭側から順次1文字ずつ行う．HIKEN-MJの内容が子音字ならば変換規則C 1.1に基づいて処理する．すなわち一般には子音字は更に子音字格納領域SIIN-MJ（容量2文字）に退避させる．但し，撥音または促音を表記するローマ字列となっていないか常に監視※※しておき，もしC 1.1のイ，ロ，又はC 2.3に示す文字列であれば直ちに該当のカナ文字がK－GBNへ転送される．一方，HIKEN-MJの内容が母音字ならば，先にSIIN-MJに退避させていた子音字列も参照し，変換規則C 1.2に基づいて変換表から該当するカナ文字列を求めてK－GBNへ転送し，文頭側から順次詰めていく．なお，HIKEN-MJの内容が特殊記号，あるいは＃で囲まれた特殊文字列の構成要素である場合は変換規則C 2に基づいてHIKEN-MJの内容をそのままK－GBNへ転送する．以上のいずれにも該当せずローマ字として読めない部分文字列が1ケ所でも発見されたら，その部分を含む単語（二つの空白にはさまれた非空白文字の連鎖）も特殊文字列と見なしてそのままK－GBNへ転送する．これは途中まで変換されてきたこの単語のカナ文字列に取って代る．このような特殊文字列扱いとなるのは，子音字列の長さが撥音や促音の表記と関係がなくて3文字以上となる場合，及び変換表の子音字部に

※生成されたカナ文字列を格納する作業領域である．

※※SIIN-MJの第1文字とHIKEN-MJを注目しておけばよい．

該当する子音字列が登録されていない場合である．なお，変換表からカナ文字列ではなく＊を得る場合（表1とB2参照）にはそのまま変換処理は続行される．図1に変換処理の経過例の一部を示す．

4. RKHKNの使い方

以上に基づいて作成したローマ字ーカナ変換サブプログラムRKHKNはCOBOLで書かれているが，このことはあまり意識せずにFORTRANで書かれたプログラムからもCALL文で呼び出すだけで，外に特別の制御文を必要とせずに使うことができる．以下にその使い方を述べる．

〔機能〕ローマ字文領域に格納されているローマ字文を，指定された文字数だけ文頭から順次カナ文字領域に頭から順次詰め，変換されたカナ文字数も数えて呼んだプログラムにもどす．

〔プログラム名〕　RKHKN

〔使用言語〕　COBOL

〔占有領域〕　約3740バイト

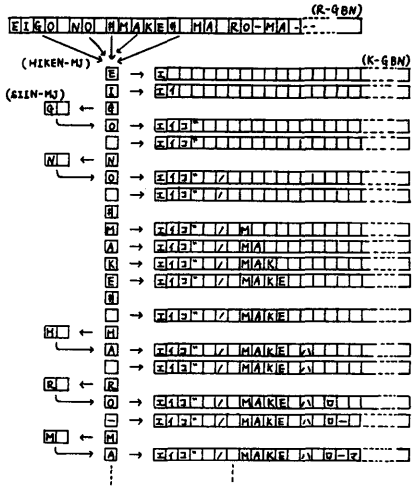

図1　変換処理の経過例（一部）

〔呼び出し法〕

（ｉ）呼ぶプログラムがFORTRANの場合；

CALL RKHKN(RGBN,RMJSU,KGBN,KMJSU)

（ⅱ）呼ぶプログラムがCOBOLの場合；

CALL 'RKHKN' USING RGBN,RMJSU,KGBN,KMJSU.

〔パラメーター〕以下の説明において〝F；〟は呼ぶプログラムがFORTRAN，〝C；〟はCOBOLであるときの注意事項である．

RGBN…ローマ字原文領域（容量800バイト以内で任意）．

F；整数型配列，但し，各配列要素には左詰めでむだなく文字が格納されているものとする．

C；英数字項目の表とする．

RMJSU…RGBNに格納された文字列の長さ（文字数）を示す変数，800≧RMJSU≧0．

F；整数型変数，但し4バイト長．

C；PIC S9(5) COMP SYNC で確保する．

KGBN…変換されたカナ文字列の格納領域（容量800文字）．領域内には左詰めでむだなく文字が格納される．なおこの領域はRKHKNの中で変換処理に先立って全体が空白で初期化される．従って正確に800文字分の領域確保が必要である．

F；整数型配列で800／n語の大きさ．但しnは配列要素のバイト長．

C；PIC X OCCURS 800で確保する．

KMJSU…KGBNに格納されたカナ文字列の長さを示す変数．この変数の確保の仕方はRMJSUに同じ．

以上四つのパラメーターのうちRGBNとRMJSUはRKHKNを呼ぶ前に定義されていなければならないが，KGBN，KMJSUは不定のままでよい．なお，RGBN，RMJSUの内容はRKHKNから呼んだプログラムに制御がもどされた時点において，呼ぶ直前のものが保持されている．

〔制御文のパラメーター〕センターバッチでカナ文字付きラインプリンターを使うためには，EXEC文，DD文の中でパラメーター　SYSOUT=Kを指定しなければならない．

〔ローマ字表記上の注意〕ここにRKHKNで処理されるローマ字の表記上の留意事項をまとめておく．なお，一般的な表記法については2.1及び2.2の表記方式，その具体列については図3の処理例を合せて参照して欲しい．

(1) ローマ字の表記法としては訓令式，ヘボン式，日本式のいずれか一つの方式を単独に用いても，これらを一つの文中で混用しても変換結果に影響はない．

(2) 促音の次の文字が空白となるようなローマ字表記は想定していない．従って例えば「アッ　トサケブ」は「アット　サケブ」というカナ文字列であると考えて　ATTO SAKEBU と書く．

(3) カナ文字で拗音，促音を表記するときは通常小文字のヤ，ユ，ヨ，ツを使う．しかしラインプリンターには通常小文字がないので，RKHKUではそれぞれ対応する大文字で代用する．小文字が備わる端末タイプライターのことは考慮しなかった．

(4) 長い文脈や意味を考慮した処理は行っていない．従って発音と表記が異なるものの表記には注意を要する．例えば助詞の〃は〃は発音通りにWAと書くとワと変換されるので，ハと変換して欲しければHAと書く必要がある．

(5) WA，WI，WU，WE，WOをそれぞれワ，ィ，ウ，エ，オと読むようにしている．これは日本語文を生成する際のワ行四段活用の動詞の活用処理を簡単化するために設けた.[1] なお，現在のラインプリンターにはワ行のヲの活字がなく日本語の表記を不自然にしている．何とか増設してもらえないものだろうかと思う．

5. 使用例と処理結果

図2にRKHKNを使ったプログラム例の主プログラムだけを示している．これはまず副プログラムRDGBNで穿孔カードからRGBNへローマ字文を読み込み，その文字数を数えてRMJSUにセットする．次にRKHKNでカナ変換を行い，次いで副プログラムPTRKでは図3に示したような印刷を行う．RKHKNによる処理時間は図3の各例文の変換結果の後に示している．例文ごとの総時間がJIKAN，ローマ字文の一文字毎の平均がHEIKINとしてミリ秒単位で示してある．これによるとM－190で処理した場合ローマ字文の一文字当り約0.04ミリ秒である．

```
      IMPLICIT INTEGER (A-Z)
      INTEGER*4 ITIME,JTIME,JIKAN
      REAL      ARMJSU,AJIKAN,HEIKIN
      DIMENSION RGBN(200),KGBN(200)
   10 CALL  RDGBN (RGBN,RMJSU,TEST)
      IF(TEST.EQ.1) STOP
      CALL CLOCKM(ITIME)
      CALL  RKHKN (RGBN,RMJSU,KGBN,KMJSU)
      CALL CLOCKM(JTIME)
      CALL  PTRK  (RGBN,RMJSU,KGBN,KMJSU)
      JIKAN=JTIME-ITIME
      ARMJSU=RMJSU
      AJIKAN=JIKAN
      HEIKIN=AJIKAN/ARMJSU
      WRITE(6,1) RMJSU,JIKAN,HEIKIN
    1   FORMAT (/1H ,T21,'(RMJSU =',I5,' MJ,',
     1           2X,'JIKAN =',I6,' MSC,',
     2           2X,'HEIKIN =',E13.5,' MSC)')
      GO TO 10
      END
```

図2　RKHKNの使用例

```
** RGBN :  IKA NO SHORI REI HA, RO-MA JI WO KANAMOJI NI HENKANSURU TETSUDUKI WO COBOL DE KAITA PUROGURAMU R
          KHKN NI YORU MONO DESU.  PUROGURAMU NO KINOU HA FORTRAN NO SUBROUTINE TO ONAJI DESU GA, FORTRAN
          NI YORU BAAI YORI YOMIYASUKU KAKETEIRU YOUDESU.  SONO RIYUU HA, COBOL NI YORU HOU GA MOJI TAN-I
          NO SYORI TETSUDUKI GA YOUINI KAKERU KOTO DA TO OMOWAREMASU.
   KGBN => ｲｶ ﾉ ｼｮﾘ ﾚｲ ﾊ, ﾛｰﾏ ｼﾞ ｦ ｶﾅﾓｼﾞ ﾆ ﾍﾝｶﾝｽﾙ ﾃﾂﾂﾞｷ ｦ COBOL ﾃﾞ ｶｲﾀ ﾌﾟﾛｸﾞﾗﾑ RKHKN ﾆ ﾖﾙ ﾓﾉ ﾃﾞｽ.  ﾌﾟﾛｸﾞﾗﾑ
          ﾉ ｷﾉｳ ﾊ FORTRAN ﾉ SUBROUTINE ﾄ ｵﾅｼﾞ ﾃﾞｽ ｶﾗ, FORTRAN ﾆ ﾖﾙ ﾊﾞｰｲ ﾖｳ ﾖﾐﾔｽｸ ｶｹﾃｲﾙ ﾖｳﾃﾞｽ.  ｿﾉ ﾘﾕｳ ﾊ, C
          OBOL ﾆ ﾖﾙ ﾎｳ ｶﾞ ﾓｼﾞ ﾀﾝｲ ﾉ ｼｮﾘ ﾃﾂﾂﾞｷ ｶﾞ ﾖｳｲﾆ ｶｶﾙ ｺﾄ ﾀﾞ ﾄ ｵﾓﾜﾚﾏｽ.
          (RMJSU =  348 MJ,  JIKAN =   15 MSC,  HEIKIN =  0.43103E-01 MSC)

** RGBN :  NIHONGO NO "HATUON" NO HYOUKI NIHA #N# NO HOKA NI "M" WO TSUKAU BAAI GA ARIMASU.  TATOEBA, #IMBA
          NUMA# (IMBANUMA), #KAMPAI# (KAMPAI), #SINMAI# (SINMAI) TO ITTA GUAI DESU.  KORERA HA, #INBANUMA#
          (INBANUMA), #KANPAI# (KANPAI), #SINMAI# (SINMAI) TO KAITEMO HENKAN NO KEKKA HA MATTAKU KAWARIMA
          SEN.
   KGBN => ﾆﾎﾝｺﾞ ﾉ "ﾊﾂｵﾝ" ﾉ ﾋｮｳｷ ﾆﾊ N ﾉ ﾎｶ ﾆ "M" ｦ ﾂｶｳ ﾊﾞｰｲ ｶﾞ ｱﾘﾏｽ.  ﾀﾄｴﾊﾞ, IMBANUMA (ｲﾝﾊﾞﾇﾏ), KAMPAI (ｶﾝﾊﾟ
          ｲ), SIMMAI (ｼﾝﾏｲ) ﾄ ｲｯﾀ ｸﾞｱｲ ﾃﾞｽ.  ｺﾚﾗ ﾊ, INBANUMA (ｲﾝﾊﾞﾇﾏ), KANPAI (ｶﾝﾊﾟｲ), SINMAI (ｼﾝﾏｲ) ﾄ ｶｲ
          ﾃﾓ ﾍﾝｶﾝ ﾉ ｹﾂｶ ﾊ ﾏｯﾀｸ ｶﾜﾘﾏｾﾝ.
          (RMJSU =  292 MJ,  JIKAN =    9 MSC,  HEIKIN =  0.30822E-01 MSC)

** RGBN :  EIGO NO #MAKE# HA RO-MA JI YOMI DE "MAKE" TO YOMEMASU.  SHITAGATTE, #MAKE# NO RYOUTAN NI SYA-PU
          KIGOU WO TUKENAKEREBA, KONO HENKAN PUROGURAMU DEHA MAKE TO YONDE SHIMAIMASU.  TOKOROGA, RO-MA JI
          TO SITE HA YOMENAI TUDURI WO MOTU GO HA SHA-PU KIGOU GA NAKUTEMO, GENGO NO MAMA WO KANA MOJI BU
          N NO NAKA NI NOKOSHITE KUREMASU.  TATOERA, COBOL, FORTRAN PROGRAM NADO DESU.  TADA, NIHON GO NO
          ON NIHA NAKUTEMO, SONO ON NO SIINBU TO ROINBU GA DOREKA NO NIHON GO ON NO KOUSEI YOUSO TO NATTE
          IRU TOKI HA * GA INSATU SARERU NODE CHUUI GA HITUYOU DESU.  TATOEBA #TSA# WO FUKUMU MOJIRETU #IT
          TSAI# HA "ITTSAI" TO NARIMASU.
   KGBN => ｴｲｺﾞ ﾉ MAKE ﾊ ﾛｰﾏ ｼﾞ ﾖﾐ ﾃﾞ "ﾏｲｸ" ﾄ ﾖﾒﾏｽ.  ｼﾀｶﾞｯﾃ, MAKE ﾉ ﾘｮｳﾀﾝ ﾆ ｼｬｰﾌﾟ ｷｺﾞｳ ｦ ﾂｹﾅｹﾚﾊﾞ, ｺﾉ ﾍﾝｶﾝ ﾌﾟ
          ﾛｸﾞﾗﾑ ﾃﾞﾊ ﾏｹ ﾄ ﾖﾝﾃﾞ ｼﾏｲﾏｽ.  ﾄｺﾛｶﾞ, ﾛｰﾏ ｼﾞ ﾄ ｼﾃ ﾊ ﾖﾒﾅｲ ﾂﾂﾞﾘ ｦ ﾓﾂ ｺﾞ ﾊ ｼｬｰﾌﾟ ｷｺﾞｳ ｶﾞ ﾅｸﾃﾓ, ｹﾞﾝｺﾞ ﾉ
          ﾏﾏ ﾅ ｶﾅ ﾓｼﾞ ﾌﾞﾝ ﾉ ﾅｶ ﾆ ﾉｺｼﾃ ｸﾚﾏｽ.  ﾀﾄｴﾊﾞ, COBOL, FORTRAN PROGRAM ﾅﾄﾞ ﾃﾞｽ.  ﾀﾀﾞ, ﾆﾎﾝ ｺﾞ ﾉ ｵﾝ ﾆﾊ
          ﾅｸﾃﾓ, ｿﾉ ｵﾝ ﾉ ｼｲﾝﾌﾞ ﾄ ﾛｰｲﾝﾌﾞ ｶﾞ ﾄﾞﾚｶ ﾉ ﾆﾎﾝ ｺﾞ ｵﾝ ﾉ ｺｳｾｲ ﾖｳｿ ﾄ ﾅｯﾃ ｲﾙ ﾄｷ ﾊ * ｶﾞ ｲﾝｻﾂ ｻﾚﾙ ﾉﾃﾞ ﾁｭｳｲ
          ｶﾞ ﾋﾂﾖｳ ﾃﾞｽ.  ﾀﾄｴﾊﾞ TSA ｦ ﾌｸﾑ ﾓｼﾞﾚﾂ ITTSAI ﾊ "ｲｯﾂｰｲ" ﾄ ﾅﾘﾏｽ.
          (RMJSU =  608 MJ,  JIKAN =   25 MSC,  HEIKIN =  0.41118E-01 MSC)

** RGBN :  MATCHI IPPON WO MOTTE, ANATA HA ITTAI DONNA TOKORO HE ITTE KITA NO DESU KA?
   KGBN => ﾏｯﾁ ｲｯﾎﾟﾝ ｦ ﾓｯﾃ, ｱﾅﾀ ﾊ ｲｯﾀｲ ﾄﾝﾃﾞﾓﾅｲ ﾄｺﾛ ﾍ ｲｯﾃ ｷﾀ ﾉﾃﾞｽ ｶ?
          (RMJSU =   76 MJ,  JIKAN =    3 MSC,  HEIKIN =  0.39474E-01 MSC)

** RGBN :  SHIN-YAKU WO NYUUIN KANJA NI TOUYO SHITE, GOKINA SEIKA WO AGETA.
   KGBN => ｼﾝﾔｸ ｦ ﾆｭｳｲﾝ ｶﾝｼｬ ﾆ ﾄｳﾖｳ ｼﾃ, ｺﾞｷﾝﾅ ｾｲｶ ｦ ｱｹﾞﾀ.
          (RMJSU =   64 MJ,  JIKAN =    3 MSC,  HEIKIN =  0.46875E-01 MSC)

** RGBN :  ITTAI DONNA TOKORO HE ITTEKITA NO DESU KA?
   KGBN => ｲｯﾀｲ ﾄﾝﾅ ﾄｺﾛ ﾍ ｲｯﾃｷﾀ ﾉ ﾃﾞｽ ｶ?
          (RMJSU =   44 MJ,  JIKAN =    2 MSC,  HEIKIN =  0.45455E-01 MSC)
```

図－3　処理結果の例

6. 検　討

ここではRKHKNとKANAを二，三の項目について比較してみる．

(1) 変換方式：KANAでは，〃…子音部の長さが0から3（例えばPPYA）まで変化するため，子音部の同定が難しい．特に誤った綴りなどがある場合を考えると，子音部の同定は非常に複雑なものになると予想される．…〃[2]との立場から，Ahoらの手法[5]を応用して，ローマ字文の中の〃子音字部＋母音字部〃を部分列とする文を受理する有限オートマトンを遷移表として実現し，更にこのオートマトンを走らせて実際に検索を行うアルゴリズムを作成している．こうすれば検索のアルゴリズムは非常に簡単になるが，遷移表の作成や修正などは必ずしもそれほど容易ではないと思われる．

これに対してRKHKNの場合，変換表は表1に示すように簡単な対応表だけであり，その作成修正はきわめて容易である．一方，変換規則を手続き的に記述することは，確かにKANAの検索アルゴリズムよりかなり複雑である．しかし遷移表をもれなく作ることと比較した場合それほど複雑であるとは言えないであろう．

(2) 処理時間：FACOM230－75によりKANAで処理した場合，ローマ字文1文字当り約1 msを要していた[2]．これに対してM－190によりRKHKNで処理すると同じくローマ字文1文字当り約0.04 msを要する．これはもちろんKANAとRKHKNの処理方式の違いよりもFACOM230－75とM－190との処理速度の違いの影響の方が大きいものと思われる．

(3) 使い易さ：KANAの場合，拗音，促音を小文字で出力できるような準備があったがRKHKNではその配慮がない．又，KANAではローマ字文とカナ文字文の領域が整合寸法で取られていたが，RKHKNではCOBOLでコーディングしたこともあって固定長の領域を取っている．従って少々使いにくい面も出てくるかもしれない．ただ，RKHKNは既に大分大学工学部組織工学科のFACOM230－38Sにもライブラリーの一つとして登録され一年以上の実用経験をもっているが，特に支障は生じていないようである．

7. むすび

ローマ字ーカナ変換と言えば，すぐヘボン式かそれとも…と表記方式を意識しがちである．しかし2.1のA1～A4のような規則にまとめると，このような表記方式の違いは単に変換表の中の項目の違いに過ぎないと言えよう．又，促音，撥音，拗音，長音などの表記規則も単純である．従って変換の仕方を遷移表の形式で文字ごとに示すKANA方法と比べても，字種ごとにまとめて手続き的に変換の仕方を示し変換表を援用するRKHKNの方法はそれほど複雑なプログラムとはなっていないと言えよう．

最後に，KANAに関する詳細な資料を提供していただいた九州大学理学部基礎情報学研究施設武谷峻一講師に深く感謝する．

文　献

1）石原，田町：〃日本語終端表現の生成のための言語情報の分析と分類〃，電子通信学会論文誌，Vol. 61-D，No. 9，pp. 627-634（昭和53-09）

2）武谷峻一：〃ローマ字ーカナ変換プログラムについて〃，九大大型計算機センター広報，Vol. 9，No. 4，pp. 267-272，（1976）

3）石原，田町：〃日本語終端表現生成システムの作成と実験〃，電子通信学会論文誌，Vol. 61-D，No. 9，pp. 635-642，（昭53-09）

4）石原　好宏：〃ローマ字カナ文字変換プログラムの作成〃，山口大学部研究報告，Vol. 29, No. 1，pp. 135-141，（昭53-10）

5）A. V. Aho and M. J. Corasick：〃Efficient String Matching：An Aid to Bibliographic Search〃，Commu. of ACM，Vol. 18，No. 6，pp. 333-340（June 1975）