ローマ字カナ文字変換プログラムの作成

石 原 好 宏*

Programming a Procedure for the Transliteration from Roman Alphabet to Japanese Syllabary (Kana)

Yoshihiro ISHIHARA

Abstract

Roman alphabet is often adopted as a means of writing for input data and for their internal processing in the case of mechanical processing of Japanese language. And yet it is hoped that the final results of processing are written at least in Japanese syllabary (kana), which is more familiar to Japanese than Roman alphabet. Then a program which transliterates Roman alphabet into kana is required.

First, specific features of Japanese strings written in Roman alphabet and writing rules of special symbols and strings which are not to be transliterated are collected. It is followed by an outline of the transliteration procedure based on the specific features and the writing rules. Then the directions for use of the actually constructed transliteration program RKHKN are given. An example of using the RKHKN and examples of the processing by the program are also presented. Finally, RKHKN is briefly compared with the KANA, which is also a transliteration program written by Takeya. RKHKN, written in COBOL, describes transliteration rules rather procedurally. KANA, on the other hand, has transliteration rules in the form of finite automaton, or transition network, and the procedure which drives the automaton. RKHKN is implemented on FACOM230-28 and also on FACOM230—38S.

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．ところで，日本語を表記する文字の体系には，漢字，カナ文字，ローマ字，数字などの外，句読点その他の特殊記号が含まれる．このうち漢字については最近機械処理の必要性が次第に認識され，処理方式の検討や実験も幾つか試みられてはいるが，まだ十分に普及しているとは言えない．従って今のところ日本語を機械処理する際には漢字以外の文字を使うことが大部分であると言える．一方，数字，句読点，その他の特殊記号は日本語をカナ文字で表現するにしてもローマ字で表現するにしても共通に必要とするものであり，又既にそれらのほとんどは現在供用されている入出力装置に一応備えられているので特に問題はない．しかし日本語の表記にカナ文字を用いるかローマ字を用いるかについては幾つかの点で検討が必要である．すなわち，通常の日本人にとってはローマ字よりカナ文字の方が取り扱いに慣れている．ところが現状ではカナ文字付きカード穿孔機はそれほど普及していないためカナ文字を穿孔するには多くの場合，従来の通常のカード穿孔機を用いたマルチパンチによらざるを得ない．しかしこの方法は操作が非常に面倒で，多くのデータを取り扱うような場合実用的とは言えない．更に，日本語の機械処理では活用処理を含むことが多く，その場合音節の単位を表示するカナ文字より，音素の単位を表示するローマ字の方が内部処理を行うときにははるかに有利である[1]．一方，出力装置としてのラインプリンター，キャラクターディスプレー装置などには，事務処理への計算機導入の増加に伴って，カナ文字を備えたものが次第に普及してきている．そうなると日本語の入力および内部処理ではローマ字表記を用い，一方処理結果の出力にはカナ文字を用いることが自然であり，当然ローマ字からカナ文字への変換を自動的に行う必要性も増してくる．

このような背景のもとに，このたび筆者は日本語終端表現生成システム[2]の一部としてローマ字カナ文字変換プログラム RKHKN を作成した．ローマ字から

* 工業短期大学部情報処理工学科

カナ文字への変換プログラムの考え方や作成例が従来からなかったわけではないが，それらに関する詳細な報告は少ない．身近かなものとしては九州大学大型計算機センターに登録されている変換プログラムについての武谷のもの[3]があるだけである．武谷の変換プログラムはローマ字からカナ文字への変換規則を一つの有限オートマトンで表現し，それをローマ字の入力文に適用しカナ文字列を出力する手続きの部分にAhoらが与えたアルゴリズム[4]を利用していることを特色とする．これに対して筆者のものは，ローマ字文が撥音（ン）と促音（つまる音ッ）を例外としてすべて子音字部十母音字部* の構造をもっていることに着目して，これを文頭から順次切り出し，ローマ字からカナ文字への変換表を参照して対応するカナ文字列を求めていくことを特色とする，この方法について武谷は，“……子音字部の長さが0から3（例えばPPYA）まで変化するため，子音字部の同定が難しい．特に誤った綴りなどがある場合を考えると，子音字部の同定は非常に複雑なものになると予想される．……”[3]としている．しかし実際に作成してみると，処理手続きもそれほど複雑となることはなく，変換表の作成・管理が有限オートマトンによる変換規則の作成・管理よりはるかに容易なことを考慮すると，筆者が採用した方法が有利な点も少なくないと言えよう．

そこで本稿では，まず変換手続きの中で利用するローマ字列の特徴と特殊文字列について整理する．次いで，これに基づく変換処理手続きの概要を述べ，作成した変換プログラムの使用方法，使用上の注意などを述べ，実際の変換処理の例を示す．最後に武谷の変換プログラムとの比較検討を簡単に行う．

2. ローマ字の表記方式と特徴

日本語の表記に用いられるローマ字の綴り方は1954年12月の内閣訓令第1号でいわゆる訓令式に統一されたが，現実には日本式，ヘボン式も相当に実用されている[5]．従って変換プログラムは，任意の表記方式でつづられたローマ字文をできるだけ区別なく処理してくれるほど実用性は高いと言えよう．又，長音の表記など一部で少し不自然な点が生ずることを容認すれば，音素の長い並びや意味内容を調べなくても十分に実用的なローマ字カナ文字変換を行うことができる．以上のことからここでは長い音素列や意味を考慮せずに変換できる限りにおいて任意の表記方式を許す立場に立って，日本語のローマ字表記の特徴を次のように整理する．

A1) 日本語音は促音（つまる音ッ）と撥音（ン）を除いていずれも“子音字部十母音字部”の構造をもつ．このうち子音字部はローマ字0～2文字で，母音字部は常にローマ字1文字で表記される．なお，これらの音をカナ文字で表記するときはいずれも1～3文字で表記できる．但しこのとき濁点や半濁点も1文字として数えている．

A2) 促音は後続の音を表す文字列の先頭文字をもう1つ重ねることによって表す．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〕 ICCHI（イッチ），ITCHI（イッチ）
KEKKA（ケッカ），NIPPON（ニッポン），…

A3) 撥音はそれの後続文字位置に何が置かれるかによって次のように4種類の表記法を使い分ける．これは変換処理を一意的かつ機械的に行うための配慮である．

a) 後続文字位置が空白のとき；N又はN-.

b) 後続文字位置がY以外の子音字のとき；N，但し後続文字位置にB, P, Mのいずれかが来るときに限ってNの代りにMを使ってもよい．

* \ * \ * \ * \	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゛	キ゛	ク゛	ケ゛	コ゛
Z	サ゛	シ゛	ス゛	セ゛	ソ゛
D	タ゛	チ゛	ツ゛	テ゛	ト゛
B	ハ゛	ヒ゛	フ゛	ヘ゛	ホ゛
P	ハ゜	ヒ゜	フ゜	ヘ゜	ホ゜
F	ファ	フィ	フ	フエ	フォ
J	シ゛ャ	シ゛	シ゛ュ	シ゛エ	シ゛ョ
KY	キャ	*	キュ	*	キョ
GY	キ゛ャ	*	キ゛ュ	*	キ゛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	シ゛ャ	*	シ゛ュ	シ゛エ	シ゛ョ
JY	シ゛ャ	*	シ゛ュ	シ゛エ	シ゛ョ
TS	*	*	ツ	*	*

Fig. 1 Transliteration table between Roman alphabet (Romaji) and Japanese syllabary (Kana)

* A,I,U,E,O を母音字，それ以外の英文字を子音字と呼ぶ．

c) 後続文字位置がY又は母音字のとき；N-．もちろん拗音やナ行の音を表すときはNYA, NAのように，Nの直後に-（ハイフン）は付けない．

A4) 長音を表記するときは適当な母音字を重ねるか（例えば OOKINA OUJI; 大きな王子），又はハイフンを母音字の直後に付けるか（例えばO-KE-; オーケー）のいずれかとする．

以上のようにローマ字の表記法を整理すれば，A1)に述べた促音と撥音を除く各日本語音は，子音字部32種類と母音字部5種類から各1種類を組み合せた文字列で表記できる．この文字列は計算上は160個となるが，必ずしもすべての文字列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のこのような立場での対応表を Fig.1 に示す．この表を以後ローマ字カナ文字変換表，又は単に変換表と呼ぶ．なお，変換表で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．

3. 特殊記号，特殊文字列の表記

ローマ字で日本語を書く場合，すべての事柄が 2. に述べた表記方式に基づく文字だけで表記できるわけではない．数字，句読点，その外の特殊記号も必要であり，更には英語などの原語をそのままのつづりで用いることが必要な場合もある．これらの特殊記号や特殊文字列は，濁点と半濁点を除けば，ローマ字表記の中であってもカナ文字表記の中であっても共通に使用されるものと考えてよい．しかも本変換プログラムでは，2. でも述べたように，原則として長い文字列や意味などを考慮しない．そうなるとローマ字としても読めるような原語のつづり（例えば英語の MAKE, MIKE,…）をカナ文字文の中にもそのまま残したければ，何らかのマークをそのつづりの前後に付けるなどの措置が必要となる．こうした観点から本変換プログラムで許す特殊記号，特殊文字列の表記法を以下にまとめる．

B1) ローマ字列中にあるつづり $\alpha_1\ \alpha_2\ \cdots\alpha_m$ をそのままカナ文字列の中にも残したければ，つづりの両端に#を付けて # $\alpha_1\ \alpha_2\ \cdots\alpha_m$ # と表記する．このときα_iは#を除く任意の特殊記号や文字であってよい．

B2) ローマ字としては読めない文字列や数字，特殊記号などはB1)の規則にかかわらず#を付けずにそのままローマ字列の中に書いてよい．但し日本語音に該当するものがなくても，2. の最後に述べたように，子音字部と母音字部の組合せ160組の中に入る文字列の場合には，外に未知語がない限り，* がカナ文字の中に出力されるだけで通常の処理が続けられるので注意が必要である（4. 2 参照）．

4. 変換処理の概要

ここではまず，2. と3. の表記規則に従い任意の表記方式に基づいて書かれた日本語のローマ字文をカナ文字文に変換するための変換規則をまとめる．次いでそれに基づく変換のアルゴリズムを述べる．

4.1 変換規則

C1) 特殊記号は#と-を除き，記号ごとにカナ文字部へ転送する．

a) 2つの#で囲まれる文字列はそのままカナ文字部への転送する．但し#は取り除く．

b) -（ハイフン）はNの直後にある場合に限りN-全体をンに変換する．その外の場合は-をそのままカナ文字部へ転送する．

C2) 数字は一般の特殊記号と同等に取り扱う．

C3) 英文字列については次の処理を行う．

a) 子音字部については通常2文字以内の連鎖だけが許される．但し次のような場合にはそれぞれ個別の処理を行う．

イ) N以外の子音字重複および文字列TCの場合には左側の子音字を促音のッと読む．

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

子音字列は上記の場合を除き，次に母音字1個を検出するまでカナ文字への変換処理は行わない．なお，上記のイ）とロ）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直ちに未知語処理を行う（4.2 参照）．

b) 母音字は1文字を検出するたびにカナ文字への変換処理を行う．その際にそれまでに検出されて未処理のままの子音字列があるかどうかによって処理は次のように分かれる．

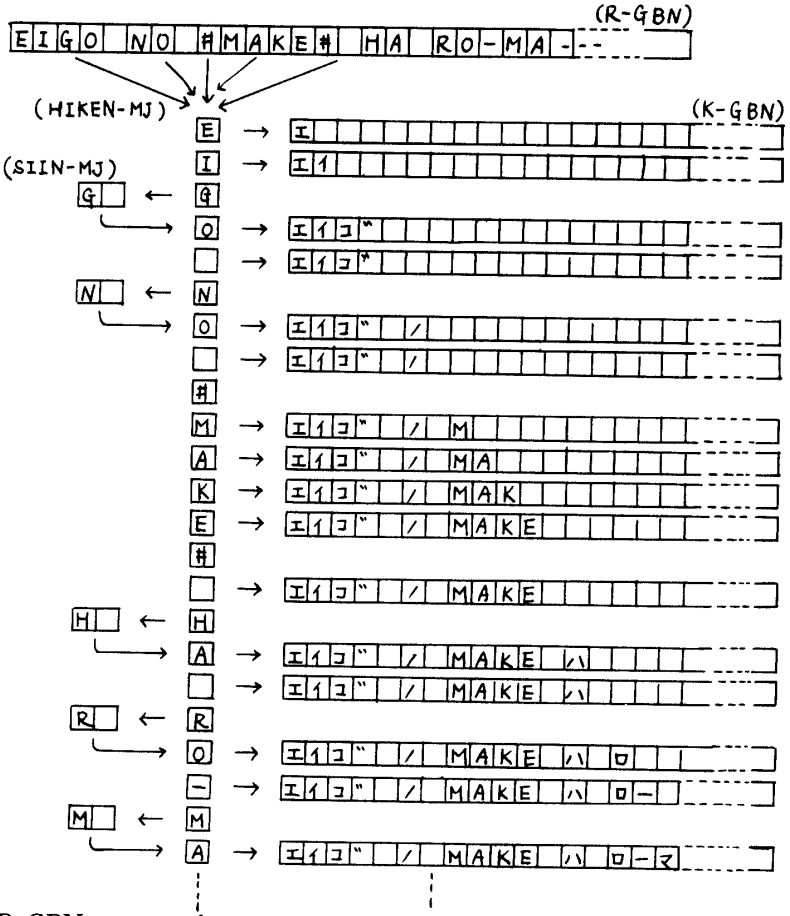
イ) 未処理の子音字列があれば，それが変換表のどの行に該当するかを調べ，次いで母音字がどの段に該当するかを調べる．こうして対応するカナ文字列を変換表から読み取ってカナ文字部へ転送する．

ロ) 未処理の子音字列がなければ，母音字がどの段に該当するかだけを調べて，変換表の

第一行（ア行）で該当の段のカナ文字を読み取ってカナ文字部へ転送する．

4.2 変換処理の手続き

この処理手続きは，4.1で述べた変換規則C1)～C3)を手続き的に記述するもので外に必要な辞書類がFig. 1に示した変換表だけであることを特色とする．その処理手続きは概略次のとおりである．まず入力されたローマ字文の文頭側から順次1文字ずつ切り出しては1文字分の領域 HIKEN-MJ へ転送してその字種を調べる．その結果，HIKEN-MJ の内容が子音字ならばそれを更に子音字格納領域 SIIN-MJ（容量2文字）へ転送しておく．一方 HIKEN-MJ の内容が母音字であれば，その前に SIIN-MJ に格納している未処置の子音字列といま HIKEN-MJ の中にある母音字とで表す日本語音を表記するカナ文字を変換表から求めて，その都度カナ文字部 K-GBN へ転送し文頭側から順次詰めていく．但し撥音または促音を表記するローマ字列の場合は変換規則C1) のb) 又はC3) のa) に基づいて，変換表を参照することなく処理手続きの中で適切なカナ文字が指定され K-GBN へ転送し格納される．なお HIKEN-MJ の内容が特殊記号，数字あるいは＃で指定された特殊文字列の場合は変換規則 C1) と C2) に基づいて，原則として原文字列のままでカナ文字部へ転送される．以上のいずれにも該当せずローマ字として読めない文字列が1ケ所でもあった場合には，その文字列を含む単語（2つの空白にはさまれた非空白の文字列）は未知語としてそのままカナ文字部へ転送される．未知語となるのは，子音字列の長さが撥音や促音の表記と関係がないのに3文字以上となる場合，及び変換表の子音字部に該当の子音の組合せがない場合である．なお，変換表からカナ文字列の代りに＊を得る



R-GBN: storage for source sentences written in Roman alphabet.
K-GBN: storage for transliterated sentences into Kana.
——→ : indicates the direction of transferring string of characters after having been processed.
HIKEN-MJ: storage for a character transferred one by one from R-GBN.
SIIN-MJ: storage for at most two characters which represent consonant.

Fig. 2 Progress of a transliteration from Roman alphabet into Kana

ような場合（例えばローマ字列がTSAのとき；Fig.1参照）には正しくカナ文字へ変換される場合と同じ処理が行われる．Fig.2に変換処理の経過の例を示す．

5. 作成した変換プログラム

5.1 作成の経過とプログラミング言語

本変換プログラムRKHKNははじめ日本語終端表現生成システム[2]の一部として作成した．その際は本工業短期大学部のMELCOM 9100/30Fにインプリメントし全システムをFORTRANで記述したので，RKHKNも当然FORTRANで記述された．この種のプログラムでは通常いわゆる文字操作が避けられず，多くの場合その部分だけはアセンブラー言語で別に副プログラムが作られる．但しMELCOM 9100/30FのFORTRANにはこのような処理のためにマスキング代入文が準備されているのでこれを用いた．しかしいずれにしても，FORTRAN的な処理方式のもとで文字操作を行うことに変りはない．FORTRANでは単位のデータ格納領域の大きさが，データの型ごとに固有の容量2～数文字分に固定されている．これは文字操作上は大きな障害となり，本来の処理目的のための手続きの外に，その固定の単位領域内での文字操作のための余分の手続きが必要となる．従って本稿の変換方式をFORTRANでプログラムした場合，目的の異なる2種類の手続きが入り混じり，プログラムは少々分かりにくくなる．

こうしたことから筆者は本変換プログラムを試みにCOBOLでもプログラムした．COBOLは本来事務データ処理を目的としたプログラム言語であって，本稿のような文字操作の場合もそれほど余分の処理手続きを混入させる必要がない．従って文書としてもFORTRANによるものよりはるかに読み易い．COBOLで書いた本変換プログラムは山口大学電子計算機室のFACOM 230-28で処理実験を行った．なおこのプログラムは大分大学工学部組織工学科電子計算機室のFACOM 230-38Sへもサポートプログラムの一つとして登録され，図形解釈システムなどの出力結果のローマ字カナ文字変換処理に既に実用されている．

このように本変換プログラムはCOBOLで記述した方が汎用性が高いので，以下の説明は特に断らない限りCOBOLで書いたものについて行う．

5.2 使用方法

本変換プログラムはCOBOLで書かれているが，そのことをほとんど意識せずに，FORTRANで書かれたプログラムからもCALL文で呼び出して使うことができる．以下にその使い方を述べる．

〔機能〕　指定された文字データ領域に格納されたローマ字文を，指定された文字数だけ文頭から順次カナ文字に変換し，別に指定された文字データ領域に文頭側から順次詰め，変換されたカナ文字の数と共に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1760語，但し1語＝16ビット
〔呼び出し法〕　CALL RKHKN (RGBN, RMJSU, KGBN, KMJSU)

〔パラメータ〕　FORTRANのプログラムから呼ぶ場合とCOBOLのプログラムから呼ぶ場合の両方について領域の確保の仕方を述べる．前者をF，後者をCとして示す．

RGBN……ローマ字文格納領域（容量800文字）．
F; 整数型一次元配列で800/n語分．但しnは配列要素当りの格納文字数．
C; PIC X OCCURS 800で確保．

RMJSU …RGBNへ格納されたローマ字文の文字数を数えてセットしておく．
F; 整数型変数，但し32ビット長．
C; PIC S9(5) COMP SYNC.

KGBN……変換されたカナ文字文の格納領域（容量800文字）．配列要素内には左詰めでむだなく文字を格納する．配列内は変換処理に先だって空白でクリアされる．なお領域の確保の仕方はRGBNと同じ．

KMJSU…変換後KGBNに格納されたカナ文字文の文字数がセットされる．なお変数の確保の仕方はRMJSUと同じ．

〔使用上の注意〕　ここではRKHKNを使用するときに留意すべき事項を個条書きにして示す．

(1) ローマ字の表記法はヘボン式，訓令式，日本式のいずれか1つの方式を単独に用いても，又これらを混用しても結果に特に影響はない．なお，一般的な表記法については2.のA1)～A4)とFig.1及び3.のB1)～B2)を参照されたい．

(2) ローマ字で長音を表すためにHが使われることがある．しかしRKHKNではこれを許していない．

(3) 促音の次の文字が空白となるような場合の促音のローマ字表記法は想定されていない．

(4) カナ文字表記の際，拗音を表す小さなャ，ュ，ョ及び促音のッは通常のラインプリンターにないので

大きなヤ，ユ，ヨ，ツが出るようにしている．

(5) 文脈または意味処理をしないので，助詞の“は”と“へ”を発音に忠実にWA, Eとローマ字表記しておくと，変換の結果は“ワ”，“エ”となるので注意を要する．

(6) WA, WI, WU, WE, WOをそれぞれワ，イ，ウ，エ，オと読むことにしている．これは日本語終端表現の生成処理のとき，一般にワ行の四段活用をする動詞（例えばIW—言うの語幹）の活用処理をその外の行の四段活用と統合して子音動詞として取り扱い[1]，特別処理をしなかったために必要となったものである．

5.3 使用例

カード上のローマ字文を読んで文字数を調べる作業を副プログラムRDGBNで行った後，RKHKNでカナ文字文への変換を行い，更に処理結果を副プログラムPTRKで印刷するプログラムの例をFig. 3に，そ

```
      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
```

Fig. 3 An example of the actual use of RKHKN

```
** RGBN :  KONO RO-MA JI KANA MOJI HENKAN PUROGURAMU HA SAISYO FORTRAN DE KAITE, MELCOM9100/30F DE SYORI SH
          ITA NO DESU GA, NOCHI NI COBOL DE KAKINAOSHI, FACOM230-28 DE SYORI SHIMASHITA.  COBOL DE KAITA P
          ROGRAM NO HOU GA HARUKANI YOMIYASUI YOUDESU.
   KGBN => ｺﾉ ﾛｰﾏ ｼﾞ ｶﾅ ﾓｼﾞ ﾍﾝｶﾝ ﾌﾟﾛｸﾞﾗﾑ ﾊ ｻｲｼｮ FORTRAN ﾃﾞ ｶｲﾃ, MELCOM9100/30F ﾃﾞ ｼｮﾘ ｼﾀ ﾉ ﾃﾞｽ ｶﾞ, ﾉﾁ ﾆ COB
          OL ﾃﾞ ｶｷﾅｵﾝ, FACOM230-28 ﾃﾞ ｼｮﾘ ｼﾏｼﾀ.  COBOL ﾃﾞ ｶｲﾀ PROGRAM ﾉ ﾎｳ ｶﾞ ﾊﾙｶﾆ ﾖﾐﾔｽｲ ﾖｳﾃﾞｽ.
         (RMJSU =  236 MJ,  JIKAN =  1654 MSC,  HEIKIN =  0.70085E+01 MSC)

** RGBN :  NIHON GO NO HATUON NO HYOUKI NI M WO TUKAU KOTO GA ARU YOUDESU.  TATOEBA, #IMBANUMA# (IMBANUMA),
           #KAMPAI# (KAMPAI), #SIMMAI# (SIMMAI) TO ITTA GUAI DESU.  KORERA HA #INBANUMA# (INBANUMA) NADO T
          O M NO KAWARI NI #N# TO KAITEMO KANA HENKAN NO KEKKA HA KAWARIMASEN.
   KGBN => ﾆﾎﾝ ｺﾞ ﾉ ﾊﾂｵﾝ ﾉ ﾋｮｳｷ ﾆ M ｵ ﾂｶｳ ｺﾄ ｶﾞ ｱﾙ ﾖｳﾃﾞｽ.  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) ﾅﾄﾞ ﾄ M ﾉ ｶﾜﾘ ﾆ N ﾄ ｶｲﾃﾓ ｶﾅ ﾍﾝｶﾝ ﾉ ｹﾂｶ ﾊ ｶﾜﾘﾏｾﾝ

         (RMJSU =  260 MJ,  JIKAN =  1517 MSC,  HEIKIN =  0.58346E+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.  TATOEBA, COBOL, FORTRAN PROGRAM NADO DESU.  TADA, NIHON GO NO
          ON NIHA NAKUTEMO, SONO ON NO SIINBU TO BOINBU 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 "ﾏｹﾞ" 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 ｵ ﾂｹﾅｹﾚﾊﾞ, ｺﾉ ﾍﾝｶﾝ ﾌﾟ
          ﾛｸﾞﾗﾑ ﾃﾞﾊ ﾏｹﾞ 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 ｵ 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 ｵ 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 ﾎﾞｲﾝﾌﾞ ｶﾞ ﾄﾞﾚｶ ﾉ ﾆﾎﾝ ｺﾞ ｵﾝ ﾉ ｺｳｾｲ ﾖｳｿ ﾄ ﾅﾝﾃ 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 ｵ ﾌｸﾑ ﾓｼﾞﾚﾂ ITTSAI ﾊ "ｲｯｻｲ" ﾄ ﾅﾘﾏｽ.
         (RMJSU =  608 MJ,  JIKAN =  3982 MSC,  HEIKIN =  0.65493E+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 ｵ ﾓｯﾃ, ｱﾅﾀ ﾊ ｲｯﾀｲ ﾄﾞﾝﾅ ﾄｺﾛ ﾍ ｲｯﾃ ｷﾀ ﾉ ﾃﾞｽ ｶ?
         (RMJSU =   76 MJ,  JIKAN =   480 MSC,  HEIKIN =  0.63158E+01 MSC)

** RGBN :  SHIN-YAKU WO NYUUIN KANJA NI TOUYO SHITE, OOKINA SEIKA WO AGETA.
   KGBN => ｼﾝﾔｸ ｵ ﾆｭｳｲﾝ ｶﾝｼﾞｬ ﾆ ﾄｳﾖ ｼﾃ, ｵｵｷﾅ ｾｲｶ ｵ ｱｹﾞﾀ.
         (RMJSU =   64 MJ,  JIKAN =   395 MSC,  HEIKIN =  0.61719E+01 MSC)
```

Fig. 4 Examples of the result of processing by RKHKN

の処理結果をFig. 4に示す．処理時間はFACOM230-28で処理した場合の例をFig. 4の各例文の後に印刷しているが，ローマ字1文字当り約6.5msと見られる．

6. 検　討

ここでは本稿で述べた変換プログラムRKHKNと武谷の変換プログラムKANAとをいくつかの項目について比較検討する．

(1) 変換方式：KANAでは子音字部の同定や誤綴りの問題を解決するために，Ahoらが開発したパターンマッチングの手法[4]を用いている．これは入力文(KANAではローマ字文)の中にキーワード(KANAでは子音字部＋母音字の単位)を見い出すために，キーワードを部分列とする文を受理する有限オートマトン(KANAでは可能な子音字列＋母音字の組合せのすべてとそれに対応するカナ文字列を記述する遷移表)と，そのオートマトンの上を走らせる検索のアルゴリズムから成る．こうすれば，後者の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・修正などが相当に面倒であると思われる．

これに対してRKHKNでは変換表はFig.1に示したようにローマ字の子音字部と母音字との組合せとカナ文字列との単なる対応表に過ぎず，これの作成・修正などは至って簡単である．RKHKNではこの外に子音字部の切り出し，母音字の検出，撥音や促音を表す文字列の同定とカナ文字の出力，特殊記号・特殊文字列の同定とカナ文字部への転送などをすべて手続き的に記述していて，KANAの処理手続きと比べると相当に複雑になっている．しかしRKHKNでは処理手続きをCOBOLで書いているために読み易く，修正も実際にはそれほど面倒ではない．

(2) プログラムの占有領域と処理時間：KANAは現在M190用に書きかえ作業中ということであるが，FACOM230-75の時代はFORTRANで書かれ，占有領域は約1400語(1語＝36ビット)となっている．そして処理時間はローマ字1文字当り約1msと言う．RKHKNを直接FACOM230-75で処理する機会はなかった．しかしFACOM 230-75とFACOM230-28との処理速度の比が，文字処理に多く使われる置数・格納・分岐・転送などの場合，おおまかに1：10～20である．一方，実際の処理時間の比はおおよそ1：6.5である．即断はできないが，基本命令の処理速度の比よりかなり小さくなっているので，同じ機械で処理した場合RKHKNはKANAより早いことも考えられる．

7. むすび

本稿では日本語のローマ字表記法の特徴と，特殊記号・特殊文字列の表記の仕方について整理し，更にこれに基づいてローマ字カナ文字変換規則を整理した．次いでこれに基づく変換プログラムRKHKNをCOBOLで記述し，変換の実験も行った．更に，全く異なる方針のもとに作られた武谷の変換プログラムKANAとの簡単な比較検討も行った．RKHKNはFORTRANで書いたプログラムからも簡単に呼び出して使うことができるので，計算機室などでサポートプログラムの1つとしても利用できる．

最後に，本稿の作成に当って詳細な資料を提供していただいた九州大学理学部付属基礎情報学研究施設講師武谷峻一氏に深く感謝する．

参考文献

1) 石原好宏，田町常夫：信学論 **J61-D**, 627-634 (1978)
2) 石原好宏，田町常夫：信学論 **J61-D**, 635-642 (1978)
3) 武谷峻一：九大大型計算機センター広報, **9**, 267-272 (1976)
4) A.V.Aho and M.J. Corasick: Commun. ACM, **18**, 333-340 (1975)
5) 日下部文夫：岩波講座日本語, **8**, 341-383 (1977)

(昭和53年4月15日　受理)