論文79-16-4-1

Korean Character Processing
Part Ⅱ. Terminal Design and History
(한글문자의 컴퓨터처리 : Ⅱ. 터미날 설계와 역사)

鄭 元 良 *
(Chung, Won Lyang)

要 約

이 논문은 "한글문자의 컴퓨터 처리 : I. 이론"의 후편으로서 동일 subject의 실질적, 역사적 측면을 취급한다. 논문의 전반부에서는 다음 문제들을 논한다. : 한글 입출력 터미날의 기능적 설계, 모아쓰기 algorithm과 dot matrix fonts에 의존한 한글 character generator, 입력 keyboard 구성(key-set 와 key-stroke 수 사이의 관계), binary code의 설계를 위해 고려되어야 할 조건 등이다. 후반부는 개인적 관점에서 본 한글문자의 컴퓨터 처리의 역사론에 할당되었다. 기록화된 업적들을 주요 내용에 따라 네그룹으로 분류하였고, 시대순으로 나열된 참고문헌들의 비판적 개론을 위해 문제점들을 하나씩 거론하였다. 입력(문자의 컴퓨터인식)과 출력(모아쓰기의 처리)의 문제들을 분별하여 토론하였다.

Abstract

This article is a sequel to "Korean Character Processing: Part I. Theoretical Foundation" and deals with the practical and historical aspects of the same subject. We discuss, in the first half, the functional design of Korean I/0 terminals, Korean character generators based on the conversion algorithm and dot matrix fonts, input keyboard configuration (trade-offs between a key set and the number of key-strokes), and the conditions to be considered for binary code design. The second half of the article is devoted to the history of Korean Character processing which is seen from the personal viewpoints. The recorded works are classified into 4 groups according to their major contents. Then we bring up each problematic issue to give a critical review of articles. Issues related to output (conversion process) and input (character recognition) are separated. The bibliography is given in a chronological order.

1. Design of a Korean Input Output Terminal

One immediate application of the Korean character conversion algorithm is to the input/output of Korean. It is the purpose of this section to discuss a design of a Korean I/0 terminal system based on the theory that we developed in part I[27] and to consider design parameters and alternatives.

By a "Korean I/0 terminal" we mean a ter-

*正會員, 韓國科學技術硏究所
(Korea Institute of Science and Technology)
接受日字 : 1979年 6月 1日

minal system, which accepts Korean symbol string input from a keyboard device or from the main CPU system and then applies the conversion algorithm to obtain 2-D character output being sent to main CPU or printed (or displayed). Whether a terminal is an intelligent one or not is immaterial in the context of present discussion but will be a factor when the terminal is to be implemented.

A. Functional Structure of a Korean I/0 Terminal

We will consider the input and output units of a terminal separately.

The "output unit" of a terminal receives a symbol string input from an applications program (in the main CPU system) or from a display file (residing in the display terminal). Its output (2-D character pattern) is directed to the print module (i.e., printer drive mechanism) or to the video generation module (in case of a display terminal). To maintain the device-indepen-dence nature of our presentation device-specific parameters will be ignored. The "character conversion module" within the output unit implements the algorithm of part I[27] and draws the character form data from the "symbol font memory module". Figure 1 shows the block diagram of an output unit.

The "input unit" gets its input symbol string from a keyboard-like device and converts it to 2-D character pattern for display (i.e., echoeing) or for transmission to an applications software. Fig 2 shows the functional structure of an input unit.

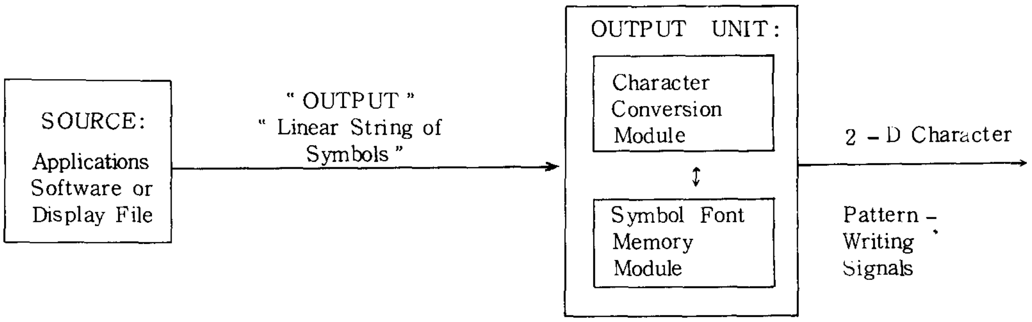
SOURCE:
Applications Software or Display File
" OUTPUT "
" Linear String of Symbols "
OUTPUT UNIT:
Character Conversion Module
Symbol Font Memory Module
2-D Character
Pattern-Writing Signals

Fig. 1. Block diagram of output unit in Korean I/0 terminal.

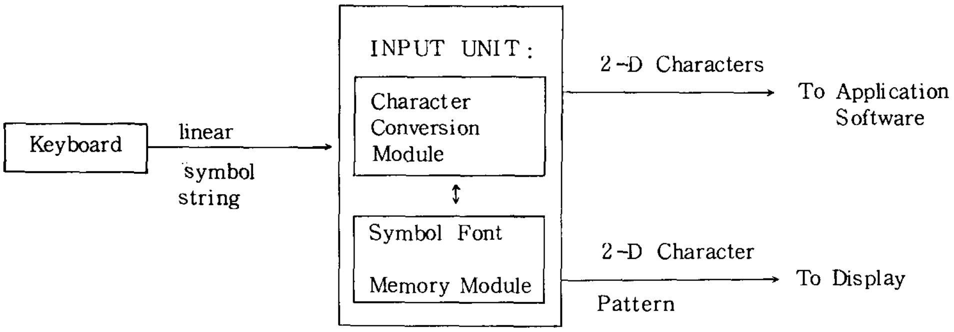

Fig. 2. Block diagram of input unit in Korean I/0 terminal.

We may observe from Fig. 1 and 2 that the combination of the character conversion and symbol font memory modules can be viewed as a (rather complicated) "character generator"in an interactive terminal.[26]

B. Character Generation Scheme

Once the character conversion module carries out the lexical and syntactic analysis algorithms its output can be used to obtain the proper combination of symbol fonts from the symbol font momory module. This is possible because its output consists of two pieces of data:" symbol code string denoted by $a_1a_2.(a_3)$.(parentheses denoting an optional symbol) and the " patternal structure code" denoted by $C_iV_j(C_k)$. (Based on the cardinal symbol set, the symbol code string may be of length up to 5 due to compound symbols. But they may all be transformed to the code of the form $a_1a_2.(a_3)$ very easily).

Now that the form of the output from the character conversion module is known we only have to discuss the organization of the symbol font memory module. For the reason of consistency, we will assume that all compound symbols (to be represented by two cardinal symbols) will be represented within the font memory as single patterns. We should note that there is no logical necessity for this./ There is a good reason for this, i.e., that of speed of character generation. Another assumption is, without sacrificing generality, that each font will be stored as a pattern on a dot -matrix of a prefixed size (See Section 1.4 of Ref.[26])

We will consider two general methods of symbol font memory organization :

(i) variable font scheme and (ii) fixed font scheme.

The basic idea of the "variable font scheme" is to use the minimum number of dot - matrix fonts. The mininum font set can be constructed out of 42 vowel and 30 consonant dot - matrices as follows : one dot - matrix for each long or short vowel corresponding to the elementary patterns of Fig. 2 of part I and one for each consonant. (For better shape, two dot - matrices can be used for each consonant, doubling the required number of fonts (i.e., to 60.) For character pattern generation, two or three vowel and consonant fonts should be superimposed (i.e.,"logical OR").This requires the availability of 5 different fonts for each consonant corresponding to 5 elementary patterns of Fig. 3 of part I. Suppose that the dot - matrix font for each consonant corresponds to the elementary pattern (i) of Fig. 3 of Part I (i.e., C_1 type). Then, the four remaining fonts can be produced by single translation transformations. This dynamic generation of consonant fonts can be done rather simply by a fixed set of translations but still requires additional time, resulting in the lower speed. This together with extra logic needed for dynamic font generation is the main argument against the variable scheme.

The "fixed font scheme"is based on the use of a large number of symbol patterns to attain high generation speed. Character pattern generation is done by static row and column - addressing of an array of symbol fonts. This necessitates the storage of vowel and consonant fonts for all elementary patterns: 42 vowel fonts (same as in variable font scheme), and 117 consonant fonts (5 for each cardinal consonant and 2 (C_4 and C_5)for each compound consonant). Fig. 3 shows a possible configuration of a font memory for the fixed font scheme. Division of font memory into 3 modules (for initial and terminal consonants, and for vowels) shall increase the speed of character generation by parallel read -out. Row and column selection operations are facilitated by the pattern structure and symbol codes respectively as shown in the figure.

C. Input Keyboard Configuration

A typewriter - like keyboard as a computer system component is used on keypunch machines and as an input device on coversational terminals (both TTY's and graphic terminals). Design of a good keyboard concerns ergonomists and computer scientists.

The former is interested in human engineering

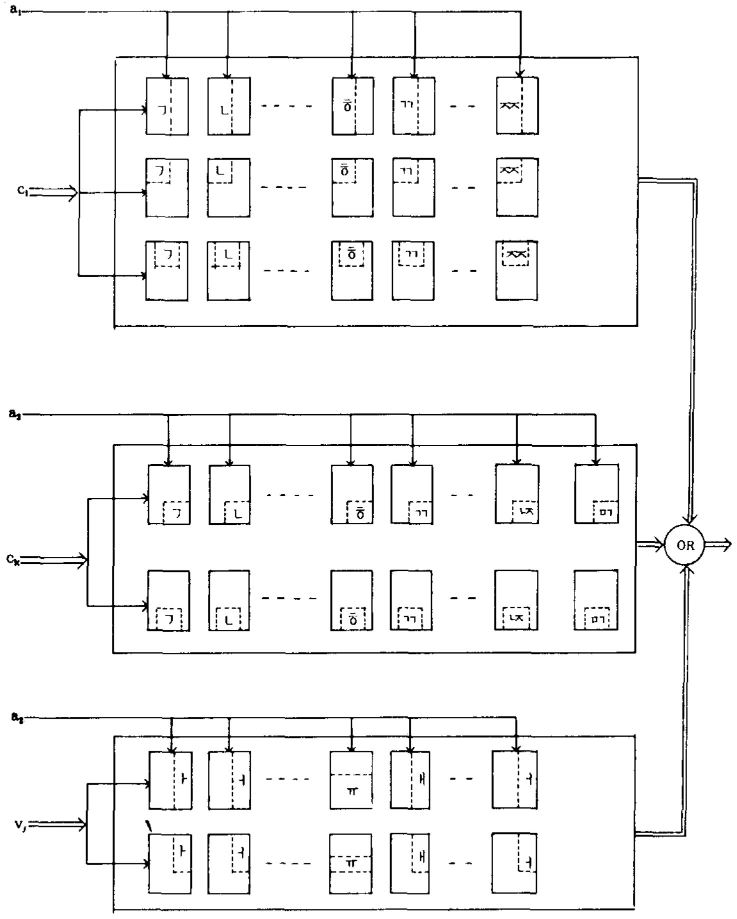

Fig. 3. Font memory for character generator.

aspects such as size and positional layout of keys and assignment of symbols to keys. The lätter is concerned with the contents and size of the character set (key set) and its encoding scheme. Of course, we take the position of the latter. We will consider the first problem here and the second in the next section.

A keyboard as a computer input device, unlike a typewriter keyboard, requires only one key or font for each Korean alphabet symbol because the linear representation of Korean characters suffices for unambiguous input. Inherent assumption for this argument is that a character conversion algorithm exists within the keyboard or in the computer system in the form of hardware, software ,or firmware, and that no special symbol is used as a character separator (i.e., syllabus separator).

We have shown constructively in the discussion of character conversion algorithm in part I the following.

PROPOSITION 1. The maximum number of keys for Korean keyboard is " 51". For this , only one key - stroke is needed for input of each Korean symbol (consonant or vowel).

We all know that the difficulty of operating a keyboard increases as the number of keys increases. Therefore, we tend to minimize the size of the key set. Furthermore, some of the existing keyboards do not support that many keys if we want to include English alphabet as well. For example, KS teletypewriter keyboard contains only 26 character keys as shown in Fig. 4. As a counterpart to proposition 1 we now have the following

PROPOSITION 2. The minimum number of keys for Korean keyboard is "25". This minimum key set consists of the basic Korean alphabet symbols (simple consonants and vowels) plus one special symbol designating the double consonants. In this case, compound vowels require up to 3 key-strokes.

D. Design of Code

It is necessary to design a code for alphabet symbols for internal representation. There are a number of codes being used by different machines and manufacturers. This is a situation somewhat similar to that for English alphabet (BCD, EBCDIC, ASCII) and may be worse.

We can identify the following factors to be taken into consideration when we design a code for Korean alphabet :(i) we should use the minimum-length code, (ii) The code should facilitate ef-

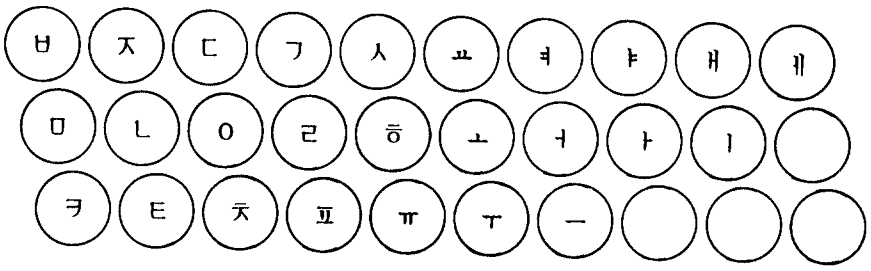

Fig. 4. Korean standard keyboard set.

The special symbol mentioned in the above was proposed in[24] where it was named "△ parameter." We will call it "△ symbol " from now on. We may see here that there is a tradeoff between the size of a key set and the required number of key-strokes. From the results of part I we have

PROPOSITION 3. The "cardinal symbol set" constitutes a minimal key set that requires at most 2 key-strokes per alphabet symbol, without using the △ symbol.

In proposition 3 if we allow the use of △symbol then the key set size may be reduced to "27" by deleting 5 double consonants from the cardinal symbol set. As we may see from Fig. 4 the Korean standard keyboard incorporates this feature except for the △ symbol which can easily be added. It should also be pointed out that widely-used keyboards (IBM and CDC for instance) are based on the "proper extension" of this standard key set.

ficient "conversion algorithm", (iii) the expansion or contraction of an alphabet set should be systematically done by appending or removing one or two bits, and (iv) the code should allow efficient lexicographic sorting of characters.

It is rather difficult to have a code that will satisfy all of the above conditions for Korean alphabet. However, a few sensible comments can be made about each factor. Korean alphabet requires at least 5 bits, either for cardinal symbol set or minimum key set. Moreover, 6-bit code can accommodate all 51 symbols.

But it is necessary to use 7 or 8 bits for mixed Korean / English alphabets. For the design of efficient conversion algorithms (described in part I) it should be easy to distinguish between consonants and vowels, between initial, middle, and terminal sounds, and between vertical (V_1), horizontal (V_2), and mixed-shape (V_3) vowels. As to the third factor, suppose that a 5-bit code is used for the car-

b₃ b₂ b₁ b₀ \ Decimal	b₇ b₆ b₅ b₄	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
0 0 0 0	0		ㄸ		
0 0 0 1	1	ㄱ	ㅃ		
0 0 1 0	2	ㄴ	ㅆ		ㅔ
0 0 1 1	3	ㄷ	ㅉ		ㅑ
0 1 0 0	4	ㄹ	ㄳ		ㅖ
0 1 0 1	5	ㅁ	ㄵ		ㅒ
0 1 1 0	6	ㅂ	ㄶ		ㅣ
0 1 1 1	7	ㅅ	ㄺ		ㅕ
1 0 0 0	8	ㅇ	ㄻ		ㅓ
1 0 0 1	9	ㅈ	ㄼ	ㅢ	ㅑ
1 0 1 0	10	ㅊ	ㄽ	ㅟ	ㅏ
1 0 1 1	11	ㅋ	ㄾ	ㅞ	ㅡ
1 1 0 0	12	ㅌ	ㅀ	ㅝ	ㅠ
1 1 0 1	13	ㅍ	ㅁㄱ	ㅚ	ㅜ
1 1 1 0	14	ㅎ	ㅄ	ㅙ	ㅛ
1 1 1 1	15	ㄲ		ㅘ	ㅗ

Fig. 5. Code design example.

dinal symbol set. Then, the code for complete alphabet symbols can be designed by adding one more bit. This also makes it possible to partition the alphabet set by an examination of single bits -- for example, into Korean and English alphabet sets. Regarding the fourth condition, character sorting can be made efficient by making the numerical order of a code compatible with the lexicographic order of symbols such as in EBCDIC or ASCII. One such code is being used in an interactive terminal made by KORONIX.

We now present a design example of a code which satisfies the first three conditions but not the last one. The code is shown in Fig. 5. It is easily seen that the code can be expanded or contracted by single bits (expansion from the cardinal symbol set is accomplished by appending "0" for consonants and "1" for vowels as the 6-th bit), and that the tests required in the conversion algorithm (lexical and syntactic analyses) for separation of consonants / vowels, V_1 / V_2 / V_3, and C_1 / C_2 / ··· / C_5 can be implemented by simple switches (i.e., comparison of numerical values). The minimum 5-bit code can also be extracted without rewriting the code. Meaningful divisions are depicted by broken lines between adjacent groups of alphabets in Fig. 5.

The lexicographic ordering is maintained only within each group. (vowels in inverse lexicographic order) This code can be extended for the representation of 2-D form of symbols simply by establishing code values for additional 19 simple and double consonants used as terminal sounds. But, this does not describe the patternal structure of characters but just the position (initial, middle, and terminal) of symbols within each character. At least 8 bits are required for the representation of patternal structure (there are 159 symbols). In fact, such an 8-bit code can be readily constructed for use with a character generator of Fig. 3 by taking the first 5 bits of the code in Fig. 5 and adding 3 bits representing the patternal structure code.

Then, these 3 bits are used as signals for row selection and the other 5 bits for column selection in Fig. 3.

2. Historical Survey

The works on Korean character processing by computer during the last 10 years since 1969 can be classified into 4 categories according to their major contents:

(i) Conversion algorithms and theory,

(ii) Display or hardcopy output device design,

(iii) Information theoretic analysis,

(iv) Character recogniton.

The list of articles belonging to these categories is given in table 1. (The article number represents the chronological order.)

Reference to the present author's work (Part I[27] and Section 1 of Part II) will be excluded from the following discussion.

Table 1. Classification of articles.

Categories	Article Number
Conversion algorithms and theory	1, 5, 6, 10, 16, 18, 20, 23
Output device design	7, 8, 11, 13, 14, 17, 19, 22, 25
Information theoretic analysis	9, 24
Character recognition	2, 3, 4, 12, 15, 21

The framework of our survey is provided by the problematic issues in Korean character processing for each of which relevant works will be comparatively discussed. At the same time, we will try to bring forward the historical significance of each work and comment on the merits and methodological aspects.

The first computer output of Korean characters was realized on a line printer and on a microfilm plotter[1,5]. The practical application of this line printer output in Korea started in 1971 at KIST computer center[5]. The first incident of display output is reported in [4] although it was not really for obtaining output but for verifying the results of a character recognition algorithm. Later works of output device design are mainly in the form of graphic displays except for[13,14,17,22]. Among the hard copy output devices, the line printer[1,5] has been used at KIST since its installation and other devices have been largely experimental such as TTY[13], dot - matrix printer[14], and ink-jet printer[17,22].

It was necessary to clarify the 2-D structure of Korean characters from the start. We want to cite 3 works[1,5,6] which obtained the well-known 30 combinations of consonants (C) and vowels (V). These C-V combinations are essentially the systematic layouts of C and V within a 4 × 3 rectangular grid window as shown in fig.6. It was in[1,5] that this 2-D structure of Korean characters was first observed. As a matter of fact, the line printer of[1,5] uses this 4 × 3 - grid window to print each character using 3 lines. These 30 combinations were shown to have a cyclic network-like structure (fig. 6) in which each neighboring element is obtained by a systematic addition/deletion of a C or V and a shifting operation[6].

Fig. 6. 2-D structure of Korean characters as C-V patterns within a 4×3 grid.

CV	CV C	CV CC	CVV	CVV C	CVV CC
CCV	CCV C	CCV CC	CCVV	CCVV C	CCVV CC
CCV V	CCV V C	CCV V CC	CCVV V	CCVV V C	CCVV V CC
CV V	CV V C	CV V CC	CVV V	CVV V C	CVV V CC
C V	C V C	C V CC			
CC V	CC V C	CC V CC			

[6] also tried to analyze the combinatorial properties of these 2-D structures using matrix and graph formulation and concluded that there are 14364 possible Korean characters composed from this combination rule and the alphabet. However, this 14364-character set is somewhat larger than the exact 12369-character set primarily because the analysis method was not so precise. (The contents of[6] and later articles by the same author seem to be derived from his Ph.D. thesis.) By the theory of Part I by the present author (particularly the patternal structure theory) it is simple to obtain this number as follows : Let $|C_i V_j C_k|$ denote the total number of syntactically correct characters with the patternal structure code $C_i V_j C_k$. Then, we know that $|C_1| = |C_2| = |C_3| = 19$, $|C_4| = |C_5| = 30$, $|V_1| = 9$, $|V_2| = 5$, and $|V_3| = 7$. From this

we get $|C_1V_1| + |C_2V_2| + |C_3V_3| + |C_3V_1C_4| + |C_2V_2C_5| + |C_3V_3C_4| = |C_1| \times (|V_1| + |V_2| + |V_3|) + |C_1| \times (|V_1| + |V_2| + |V_3|) \times |C_4| = |C_1| \times |V| + |C_1| \times |V| \times |C_4| = 19 \times 21 + 19 \times 21 \times 30 = 399 + 11970 = 12369$. A similar approach to this was used in[9] which obtained a linear algebraic expression(C +CC) (V +VV +VVV) (1 +C +CC), to describe the combinatorial structure of characters. This algebraic expression became the basis of a lexical analyzer for linear input[16].

The efforts to make the 2-D output "beautiful"by considering the geometric variables such as the relative positions and shapes of characters within a window have been made as the understanding of 2-D syntax increased [7, 8, 11, 14, 17, 19, 22, 23, 25]. The main geometric features used in obtaining the beautiful output include the distinction between long and short vowels (depending on the absence and presence of a terminal consonant) and the position of consonants as a function of vowel type. A variety of character conversion algorithms were developed, and in most cases, the conversion algorithms were implemented by hardware, except that microprocessor codes were used in[7, 25] and that software implementation was used in[19, 23].

We may classify the conversion/output algorithms according to the type of character fonts used. Those algorithms for line printers and TTY's are based on the "fixed character fonts", i.e., one font per alphabet symbol[1, 5, 13, 14, 20]. Those based on stroke generation include[2, 8, 12, 18, 23], and the stroke pattern for each symbol consists of fixed line segments (most of them use 8 directional vectors (✳) plus "O"). The use of dot matrix symbol fonts seems to be most popular.[7, 11, 14, 17, 19, 22, 25] The organization of symbol font memory into initial, middle, and terminal sound groups and parallel read-out for speeding up the display were first mentioned in.[11] We may notice that different dot-matrices and symbol font sets have been used : dot matrices of the size 10×16[11], 12×14[17, 22], and 5×5[25] font sets containing 67[11], 85[17, 22], 33[25], dot patterns.[25] uses a method different from others in that the output character is formed from the proper positioning of component symbol dot matrices (5×5) within a 13×15 dot matrix. (In other words, it is a variable font scheme.).

Analysis of Korean character syntax in terms of the formal language theory has been done by a couple of works[7, 10, 19, 21] in developing a theory or algorithm. Context-free property of Korean character syntax was observed first in[7]. However, no one really tried to use this grammar in the design of lexical and syntactic analyzers for the conversion algorithm as in the compilers of programming languages. Algebraic theories have been used to describe the 2-D syntax and conversion algorithms : Boolean algebra[8, 9, 13, 16, 25] and group theory[12].

2-D structure of Korean character has been analyzed in terms of segmentation ("elementary patterns") and pattern combinations. Pattern forms were divided into a number of classes, and different numbers of pattern classes were used : The maximum was 30 as we mentioned earlier[1, 5, 6, 8], the minimum was 6 as in Part I[17, 22], also used were 18[2], 9[14, 25], and 7[16]. We may note that it is desirable to use the smaller number of pattern classes. This is because the small number of pattern classes implies the larger symbol font set but simpler con ersion algorithm and shorter code.

As to the binary code for the Korean alphabet, a scan through the literature indicates the popularity of ASCII, but without apparent reasons. Table 2 summarizes various codes.

The use of microprocessors to implement the conversion algorithm or as an input/output terminal controller can be found in a mumber of works.[7, 14, 17, 19, 22, 25]

A relatively large number of works appeared on character recognition for computer input (6 articles). They can be classified into two groups according to whether a recognition algorithm is for handwritten[15] or printed characters[2,3,4,12,21]. These algorithms are dealing with either 1-D characters[3,4,12,15] or 2-D characters [2,21] All of them are automatic algorithms,

Table 2. Summary of binary codes for Korean alphabet.

Article No. or Manufacturer	Base Code	#bits	# Korean-Symbols	Comment
4		15		
5	ASCII (BCD, EBCDIC)	8	33	K/B-oriented
11		15	67	distinction of initial/middle/terminal sounds
14	ASCII	7	33	
17	ASCII	7	55	
19	ASCII	7	57	
22	ASCII	7	34	K/B-oriented
24	ASCII	8	28	lexicographic order
25	ASCII	7	33	lexicographic order
IBM	EBCDIC	8	40	K/B-oriented
CDC	BCD	7	32	K/B-oriented
Koronix	ASCII	8	66	lexicographic order

but none possesses learning or training capabilities. Each work will be briefly discussed in the following.

Analysis of 2-D Korean characters as a line structure base on 8 directional strokes (✳) and ″O″ was done by[2] for the first time. [2] identified 18 areal structures and used them to recognize each character. That is, it determined the relative positions of each symbol within an areal pattern from the matrix encoding of each symbol in terms of stroke segment code.

[3] and [4] are based on the extraction of the pattern code of each symbol from the 3×5 grid sampling of the rectangular window containing the symbol. This 3×5 grid results in a 3×5 Boolean matrix which leads to a 15-bit code. This 15-bit code was used to generate a display output in[4].

In[12] the 24 Korean symbols were partitioned into 3groups (concave, line, and mixed structures), and the generating pattern sets were determined for these groups ({ㄱ, ㄴ, ┌ , ┘} and {✳, 0}). And then the algebraic properties of symbols were derived from this framework and the complexities of character structures were compared. It was shown that the basic vowels form a number of equivalence classes under the rotational transformation. It also emphasized the fact that geometric as well as topological properties were valuable in character recognition. It drew a conclusion that (✳, 0) is fundamental as a generating set and that connecting nodes play critical roles in the recognition process.

The concave structure of[12] became the basis of[15]. The notable feature of[15] is the determination of the minimum set of elementary pattern segments that can describe the geometric structure of Korean symbols uniquely. This minimum set consisted of 6 logic patterns and 4 phase properties. Character recognition was done by determining the segment structure from the extracted pattern segments. A logic circuit based on the phase property was desiged to perform the task.

An automatic recognition by a syntactic method was discussed in[21]. The algorithm consisted of 4 stages : (i) preprocessing -- digitization by a vidicon camera and the use of a thinning algorithm for segmenting basic strokes (✳, 0), (ii) graph construction -- detection of 4 feature points (nodes) and construction of matrix representation of the graph from the chain encoding of line segments, (iii) segmentation -- extraction of component symbols from the graph matrix, and (iv) pattern analysis -- tree construction from the extracted symbols and nondeterministic syntactic analysis based on a tree grammar.

There have been a few other developments that are not listed as references. In 1974, a KIST team attempted to develop an English-Korean translator and Korean-COBOL processor, which seems to be the first(and last) of such efforts. The first Korea line printer program[1,6] was said to be improved in its efficiency in 1977[20]. Around that time the KIST Software Development Center developed a Korean output routine for Tektronix 4014/15 graphic terminals[23]. Recently, a Korean output terminal based on a color T.V. and a microprocessor was designed and implemented by a KAIS team(Computer Science Department), the results of which were presented at a KISS Conference in spring of 1979. To the present author's knowledge, there are a couple of ongoing projects to develop Korean output terminals at KIST and KIET.

There have been also a number of commercial ventures on Korean I/O terminals, both successful and unsuccessful, since 1978. The impetus for competitive efforts came from the imminent needs for such terminals as computer applications began to penetrate into the public sectors. The potential of the market in Korea should not be underestimated. If we name some of them here, they are Gold Star Electric Co. (derived from the ink-jet printer of [17, 22]), Samsung Electronics(interactive and printer terminals), Korea Computer Co. (intelligent terminals based on DATA-100 systems), Koronix Corp.(KHD-1A display terminal, being used in a KIST project), and IBM. It remains to be seen who will dominate the domestic Korean I/O terminal market in the coming years. Manufacturers of large mainframe computers such as IBM have the advantages of having a large number of customers who are already using their computer systems. However, independent terminal distributers can become quite competitive by making their products compatible with these mainframes and by lower prices.

3. Summary

In this article(and Part I[27]), we tried to present a coherent discussion of how to make Korean an integral part of computer-processed data very much like English and number. The center of this effort is the conversion algorithm which is needed because of the unique characteristics of Korean characters. The linear representation of Korean is convenient for human operation of input devices and of course a natural form of computer-stored data. However, we are accustomed to the conventional 2-D form of Korean in reading and writing.

Viewed in the light of the total computer system, this conversion process provides for an essential man-machine interface. This lead us to the embodiment of the conversion algorithm in an input/output terminal. We discussed a few related problems: the design of input keyboard and coding method in addition to the terminal design.

A short historical review of the articles dealing with Korean character processing was presented in the second half. It is intended to provide the reader with a quick glimpse of the 10-year's works in the field.

As a concluding remark we should call for the organized efforts to establish the national standards in Korean character processing. Standards for the key set and binary code will be most fundamental. Standards for the Korean character syntax and conversion algorithm should also be considered. Furthermore, enforcing the standards will be as important as establishing it. (The KS key set seems to be in the danger of total ignorance.) The author tried to present a foundation and approaches that are viable in pursuing these goals.

There still remains a lot to be done in the area of Korean character processing. Efforts should be mustered to develop computer languages in Korean. Computers are no longer a fancy but everybody 's tool. This includes kids and nonspecialists. For them, Korean will be the language of interaction with a computer system. In relation to this, we need efforts to move from character processing to language processing. Most of all we cannot be too early. So we better start now!

4. Bibliography

1. Sung, K.S. and Smith, D.L. Computer handling of Korean language texts. Battelle Memorial Res. Lab., 1969.
2. Kang, I.K. and Lee, H.S. A method of Korean Character pattern extraction. "J. KIEE" 6-2(Sept. 1969), 1-5.
3. Lee, J.K. A method for the recognition of printed Korean characters, I. "J.KIEE" 6-4 (Dec. 1969) 198-209.
4. Lee, J.K. and Lee, K.W. Recognition of printed Korean characters(II)."J.KIEE" 7-3 (Nov. 1970), 130-136.
5. Sung, K.S., et al. Research on Koreanization of EDPS. Rpt. NK6-184, KIST, Jan. 1971.
6. Lee, J.K. Mathematical analysis of the structure of Korean characters. "J.KIEE" 9-4 (Sept. 1972), 197-204.
7. Park, H.S. Korean language mechanization. Proc. of Symp. of Korean Scientists and Engineers in Korea-U.S.A. 1974, pp 232-258.
8. Lee, J.K. and Lee, K.H. A new Korean character display. "J.KIEE" 11-1(Feb.1974), 23.
9. Lee, J.K. and Choi, H.M. Statistical measurement of monosyllable entropy for Korean language. "J.KIEE" 11-3 (June 1974), 119-125.
10. Byun, H.S. Properties of the Korean alphabet and its computer applications. "J. KISS" 1-1 (Aug. 1974), 19-22.
11. Ann, S.G. A method for the display of Hangeul in its traditional combined form. "J.KIEE" 12-1(Feb. 1975), 27-33.
12. Lee, J.K. and Choo, H. Algebraic structure for the recognition of Korean characters. "J.KIEE" 12-2 (Apr. 1975), 44-50.
13. Kim, J.K., Song, K.H., and Ahn, S.S. On the control logic circuits for the platen controlled Korean teletypewriter. "J.KIEE" 12-4 (Aug. 1975), 116-121.
14. Lee, Y.T. and Goo, J.H. Study on the development of Korean printer terminal. Rpt. BSJ 96-752-7, KIST, March 1976.
15. Lee, J.K. and Kim, H.K. Automatic recognition of hand-written Hangeul by the phase rotation. "J.KIEE" 13-1(march 1976), 23-30.
16. Lee, J.K. and Nam Kung, J.C. Automatic discriminating of monosyllable in Korean Characters "J.KIEE" 13-5 (Dec. 1976), 155-159.
17. Goo, J.H. and Lee, M.J. Study on the development of Korean printer based on ink-jet technique. Rpt. BSE 304-933-7, KIST, Feb. 1977.
18. Kim, S.M. A study on organizing the least informations for vector drawing Korean characters. "J.KIEE" 14-2 (Aug. 1977), 32-38.
19. Lee, Y.T., et al. Development of Korean terminal using the commercial T.V. receiver. Rpt. BSE 359-992-7, KIST, Sept. 1977.
20. Cho, S.Y, Yoon, J.C., and Kwon, S.D. Study on Korean output software development. KIST, Dec. 1977.
21. Kim, J.K. and Agui, T. A study on the pattern recognition of Korean characters by syntactic method. "J.KIEE" 14-5 (Dec. 1977), 15-21.
22. Goo, J.H. and Lee, M.J. Study on the improvement of Korean printer based on ink-jet technique. Rpt. BSI 590-1044-7, KIST, Fed. 1978.
23. Chung, K.D. and Chung, W.H. How to use

the Korean output routine for graphic system. "KIST Computer Newsletter" 9-1 (March 1978), 15-16.

24. Lee, J.K., Park, J.W., and Kim, C.S. Some structural analysis of HANGEUL information source and its applications to the improved coding method. "J.KIEE" 15-2 (May, 1978), 1-17.

25. Kang, C.H. and Tominaga, H.A HANGEUL Character input output terminal controlled by microprocessor. "J.KIEE" 15-2 (May, 1978), 8-14.

26. Chung, W.L. "Interactive Graphic Computer Aided Design" Note for a professional seminar with the same title (Fed. 22-24, 1979), KIST, Jan. 1979, 241pp.

27. Chung, W.L. Korean character processing: Part I. Theoretical foundation. to appear "J.KIEE" 16-3 (July, 1979) 1-8.