論文 78-15-2-2

論 文

마이크로·프로셋서를 이용한 한글문자 입출력시스템

(A HANGEUL Character Input Output Terminal Controlled by Microprocessor)

姜 哲 熙*, 富 永 英 義**

(Kang, Chul Hee, and Tominaga, Hideyoshi)

要 約

33자의 한글 기본자소(字素)의 크기를 가변시키면서, 9종류의 합성 패턴중에 해당되는 패턴을 찾아내어 한자를 구성하는 프로세스를 마이크로 프로셋서를 사용하여 실현시킬 때의 제반문제들에 관해 논하고 있다.

Abstract

This paper discusses topics, which can be considered to generate a HANGEUL character pattern with general micro-processor.: A character must be composed by combining the input elements at their proper sizes and positions based on algorithm proposed here.

1. 머 릿 말

가까운 과거 만해도 한글 입출력장치에 한글문자 모아쓰기를 가능케 하는 회로를 전용으로 정착시키는 것은 불합리적이라고 생각되었었고, 또한 활자형의 임팩트·프린터(impact printer)밖에 없었기 때문에 더욱 실현하기 힘든 것으로 알려졌었다. 이러한 문제점을 해결하기 위해(디스플레이(display)에 관한 연구이긴 하지만.) 문헌 1), 2)와 같은 귀중한 연구가 행해진 바 있다.

그러나, 최근의 반도체기술의 눈부신 발전, 특히 마이크로·프로셋서(micro-processor)의 출현은 저속도 분야에서 어떤 디지털·시스템의 논리회로를 하-드와이어드·랜덤·로직(hard-wired random logic)에서 프로그래머블·로직(programmable logic)으로의 변환을 용이하게 실현 가능케 함으로 인해, 예를 들어 한글문자 모아쓰기를 실현하는 비교적 복잡한 논리회로의 경우에도 아주 간단히 구성할 수 있게 되었다.

이 보고에서는 마이크로·콤퓨터를 사용하여 한글의 기본 자소만을 갖고 크기를 변화시키면서 글자를 합성시켜 일정한 크기의 모아쓰기 글자로 출력시킬 때의 문제점,

○한글문자의 부호화
○한글문자의 기본닫·패턴(dot pattern)의 결정
○한글문자의 패턴 발생 앨고리즘(algorithm)
○한글문자 합성 프로그램
○입출력 결과
○남은 과제

등에 관해 논할 것이다.

2. 한글문자의 부호화

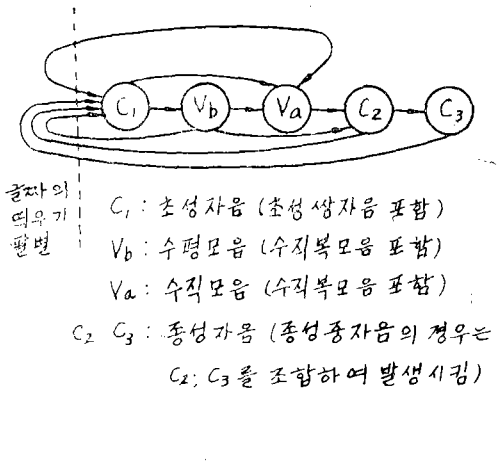
한글문자의 기본자소를 24자로 했을 경우, 5비트(bit)로 표현될 수 있어 리던던시(redundancy)가 적은 부호화가 가능하나, 합성되는 글자의 패턴수는 30종류나 된다.

한편, 33자(초성 쌍자음 5자와 수직복모음 4자를 포함)의 경우는 6비트라야 표현될 수 있는 반면, 합성 패턴수가 9종류로 줄어들어 합성 앨고리즘을 간략화 할 수 있다. 두 가지 방법 중 어느 것을 택하느냐는 설계자의 센스에 의해 결정되는 문제라고 생각되지만 이 보고에서는 후자를 택했다.

*正會員 **非會員
早稻田大學理工學部電子通信學科
(Dept. of Electronics and Communication Engineering WASEDA University Tokyo, Japan)
接受日字 : 1977年 8月 24日

후자를 택한 이유는,

1) 그림 1.에 표시한 바와 같이 글자와 글자 사이의 구별이 자동적으로 되기 때문에.



2) 기본 자소의 확대배율이 수평 1배, 2배, 수직 1배, 1.5배, 3배, 5종류만으로 한글 전부를 표현할 수 있게 되어 합성 앨고리즘이 간단하게 될 뿐더러 깨끗한 글자를 합성할 수 있기 때문에. (그림 2. 참조)

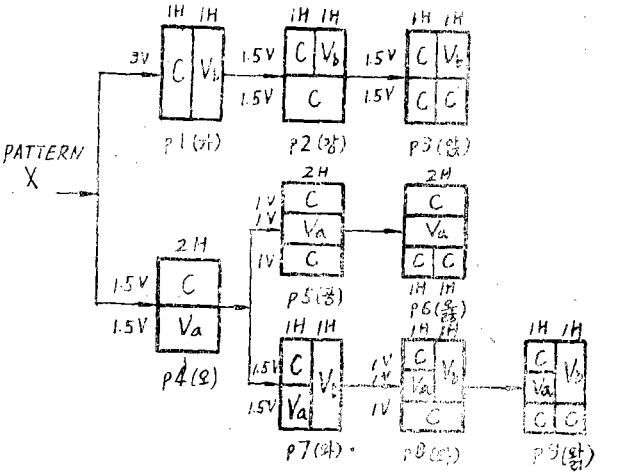


그림 1. 한글문자 발생의 상태 변이도
Fig. 1. State diagram of HANGEUL character.

그림 2. 9종류의 패턴 합성법
Fig. 2. Combining character forms.

3) 데이터(data) 전송의 경우는 자소의 정보를 보내기 보다 합성된 글자에 부호화를 행해 전송하는 것이 효율적이라고 생각되기 때문에.

예) 욱, 100100010010

짝수 Parity bit 11 bit (2048자까지 표현 가능)

	Parity Bit								
b_6	0	0	0	0	1	1	1	1	
b_5	0	0	1	1	0	0	1	1	
b_4	0	1	0	1	0	1	0	1	
b_3 b_2 b_1 b_0	0	1	2	3	4	5	6	7	
0 0 0 0	0 NUL	DLE	SP	0	@	P	ㅏ	ㄱ	
0 0 0 1	1 SOH	DC_1	!	1	A	Q	ㅐ	ㄴ	
0 0 1 0	2 STX	DC_2	"	2	B	R	ㅑ	ㄷ	
0 0 1 1	3 ETX	DC_3	#	3	C	S	ㅒ	ㄹ	
0 1 0 0	4 EOT	DC_4	$	4	D	T	ㅣ	ㅁ	
0 1 0 1	5 ENQ	NAK		5	E	U	ㅓ	ㅂ	
0 1 1 0	6 ACK	SYN	&	6	F	V	ㅔ	ㅅ	
0 1 1 1	7 BEL	ETB	'	7	G	W	ㅕ	ㅇ	
1 0 0 0	8 BS	CAN	(	8	H	X	ㅖ	ㅈ	
1 0 0 1	9 HT	EM	)	9	I	Y		ㅊ	
1 0 1 0	10 NL	SUB	*	:	J	Z		ㅋ	
1 0 1 1	11 VT	ESC	+	;	K	ㄲ	ㅗ	ㄹ	
1 1 0 0	12 FF	FS	,	<	L	ㄸ	ㅛ	ㅍ	
1 1 0 1	13 CR	GS	-	=	M	ㅃ	ㅜ	ㅎ	
1 1 1 0	14 SO	RS	.	>	N	ㅆ	ㅠ	ㅉ	
1 1 1 1	15 SI	US	/	?	O	ㅉ	ㅡ	DEL	

$C = b_6\, b_4\, (b_5 + \bar{b}_5\, b_3\, b_2)$
$V_a = b_6\, b_5\, \bar{b}_4\, (\bar{b}_3 + b_3\, \bar{b}_2\, \bar{b}_1\, \bar{b}_0)$
$V_b = b_6\, b_5\, \bar{b}_4\, b_3\, (b_2 + \bar{b}_2\, b_1\, b_0)$
$V = V_a + V_b = b_6\, b_5\, \bar{b}_4$

그림 3. 한글문자의 코드 예
fig. 3. Coded HANGEUL characters.

4) 인텔리전트·터미날(intelligent terminal)로서 한글 입출력장치를 상정(想定)하고 있기 때문에 터미날 자체가 마이크로·프로세서를 갖어, 한글문자의 합성은 물론 3)에서 논한 코드(code)변환, 통신제어등을 행해 준다고 가정하고 있기 때문 등이다.

한글문자의 표준화는 3),4)에서 논한 문제와 급격히 발전하는 집적회로 기술등을 염두에 두고 행해질 필요가 있다고 생각되어진다. 그림 3.에 표시하는 코드 예는 ASC Ⅱ 코드의 소문자 부분에 한글을 적용시켰을 때의 한 예이다. 수직복모음과 초성중자음의 코드를 디코딩(decoding)하는 회로가 비교적 복잡하다고 생각되지만 IC를 이용한다면 간단히 실현시킬 수 있다.

3. 한글문자의 패턴발생 앨고리즘

3-1. 기본자소의 닽·패턴 결정

기본자소의 닽·패턴은 합성했을 때의 패턴의 크기를 기본으로 하여, 필요에 따라 크기를 축소시키는 방법과, 최소한의 크기를 기본으로 하여 확대시키면서 합성시키는 방법이 있다. 여기서는 패턴·메모리(memory) 용량을 최소한으로 하기위해 후자를 택했다.

한글 기본자소 33자 전부와 균형 잡힌 글자 패턴을 표현할 수 있는 최소의 닽수가 5×5인 것을 경험적 수법에 의해 구했다. 따라서, 합성된 한글 한자 10×15 닽수로 표현되어진다.

그림 4.에 한글 기본자소의 닽·패턴을 되자인 한 예를 제시한다.

3-2. 직선의 확대

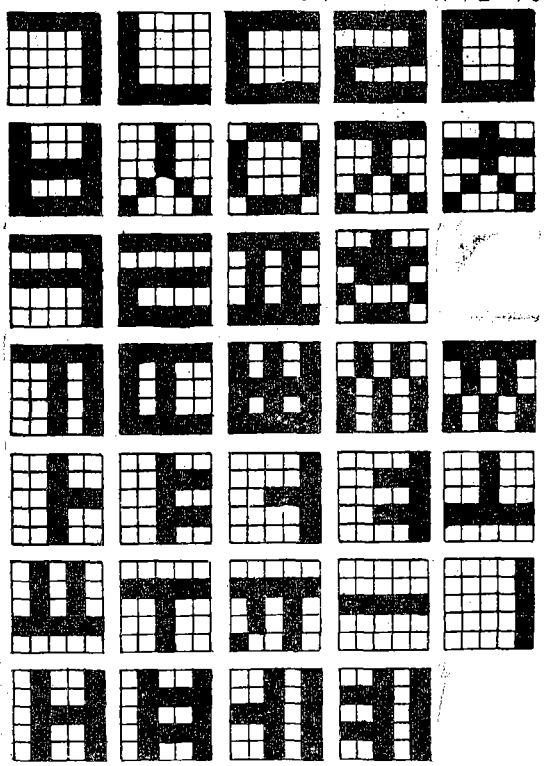


그림 4. 한글문자의 닫·패턴
Fig. 4. Dot pattern of HANGEUL characters.

닫·패턴을 그대로 확대하면 스트로크(stroke)의 굵기 까지 확대되어 균형잡히지 못한 글자가 되어버린다 따라서, 패턴·데이터는 스트로크 표현법을 이용해, 스트로크의 좌표치(시점과 종점) 데이터에 필요한 배율을 곱해 확대된 좌표치를 구하도록 했다.

3-3. 빗금의 확대

빗금을 단순히 확대하면 그림 5-a와 같이 되어 보기 싫은 글자가 되어 버린다. 그래서, 그림 5-b와 같이 45°방향과 직선으로 표현함으로써 글자 모양의 부자연스러움을 없앨 수 있다는 것을 발견했다. 또한, 그림 6.에 표시한 것과 같이(단, E_1, E_4, E_7, E_{10}는 제외) 빗금을 확대할 때는 시점에서 종점을 향해 반시계 방향으로 선분이 항상 바깥 쪽으로 볼록 나오도록 연결시킨다.

지금, 확대된 좌표치의 시점의 x좌표, y좌표를 S_x, S_y, $|S_x-E_x|=\Delta x$, $|S_y-E_y|=\Delta y$라고 놓으면,

$$\begin{cases} S_x-E_x=0 \begin{cases} S_y-E_y=0 ; \text{이동하지 않음} \\ |S_y-E_y|>0 ; \pm y\text{방향으로 이동} \end{cases} \\ S_x-E_x>0 \begin{cases} S_y-E_y=0 ; -x\text{방향으로 이동} \\ |S_y-E_y|>0 ; \text{빗금방향으로 이동} \end{cases} \\ S_x-E_x<0 \begin{cases} S_y-E_y=0 ; +x\text{방향으로 이동} \\ |S_y-E_y|>0 ; \text{빗금방향으로 이동} \end{cases} \end{cases}$$

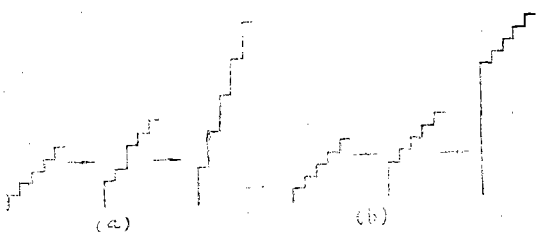


그림 5. 빗금확대의 두가지 방법
Fig. 5. Two methods of magnifying oblique-line.

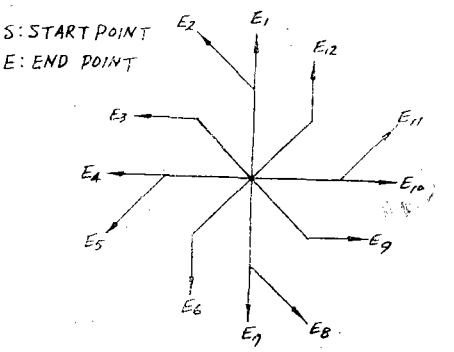


그림 6. 직선, 빗금의 연결법
Fig. 6. Connecting method of point-to-point.

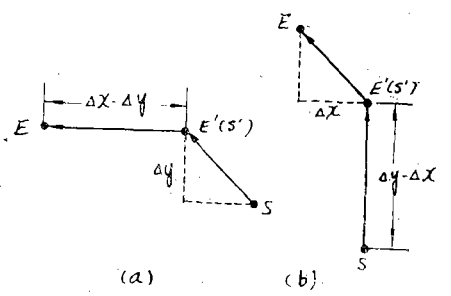


그림 7. 빗금 연결법
Fig. 7. Connecting method of oblique-line.

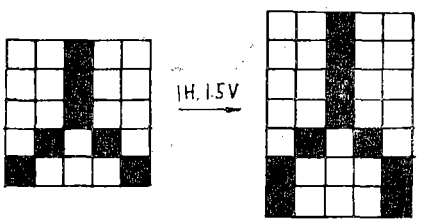


그림 8. 빗금을 확대시킨 보기
Fiig. 8. Example of magnified oblique-line.

과 같은 관계식으로 전 방향을 표현할 수 있다.

그리고, 실제로 확대된 빗금을 그릴 때는 다음과 같은 제한조건을 부과시킬 필요가 있다. 예를 들어,

$S_x - E_x > 0$, $S_y - E_y < 0$일 경우에는

$\begin{cases} \Delta x - \Delta y = 0 \text{이면 } 45° \text{의 빗금} \\ \Delta x - \Delta y > 0 \text{이면 그림 7-a} \\ \Delta x - \Delta y < 0 \text{이면 그림 7-b} \end{cases}$

와 같이 직선과 빗금을 연결시킨다. 이와 같은 알고리즘에 따라서 확대시킨 예가 그림 8. 이다.

4. 한글문자 합성 프로그램과 입출력 결과

4-1. 합성프로그램

합성프로그램을 다음과 같이 5두가지로 크게 나눌 수 있다.

i) 기본자소가 입력장치로부터 입력되면 기억하면서 글자와 글자사이의 띄우기를 검출한다.

ii) 입력된 기본자소의 집합이 그림 2.에서 표시한 합성패턴의 P_1에서 P_9까지의 어느 것에 속하는 지를 찾아 낸다.

iii) 찾아 낸 패턴을 다음과 같은 순서로 처리한다.

a) 입력된 자소의 집합이 33기본자중의 어느 자인가를 찾아내어 그 자의 달·패턴이 기억되어 있는 메모리·어드레스(address)를 검색한다.

b) 선택한 메모리의 내용을 앞에서 설명한 확대알고리즘에 따라서 필요한 크기로 확대한다. 또한 합성된 글자 한 자를 구성할 때까지 위의 프로세스(process)를 되풀이 한다.

iv) 다음에 입력될 자소집합을 위해 미리 필요한 준비를 행한다.

그림 9에 이상과 같은 알고리즘에 의한 흘로·챠어(flow-chart)를 나타냈다.

이 프로그램은 INTEL 8008의 쎔블러프(assembler)

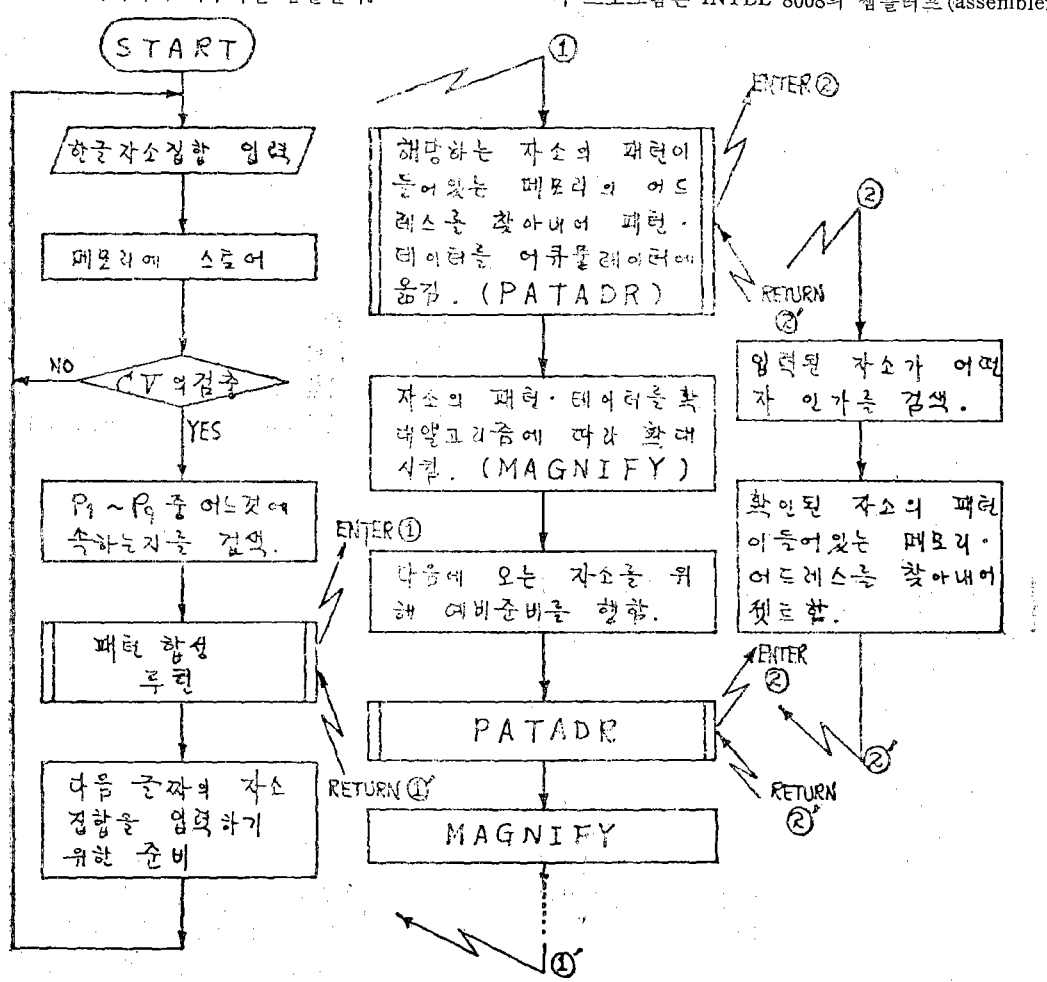


그림 9. 한글문 패턴발생 프로그램의 흘로·챠트

Fig.9. The flow-chart of HANGEUL character pattern generating program.

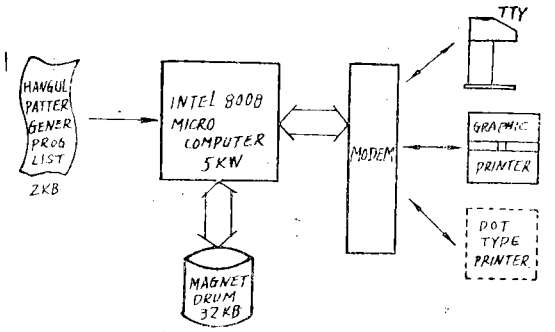


그림 10. 실험시스템의 구성도
Fig. 10. Block diagram of this system.

를 이용하여 작성했고, 그의 프로그램·스텝(step)수는 약 500(2KB)스텝이었다.

4-2. 시스템 구성

실험시스템 구성도를 그림 10.에 제시한다. 마이크로·콤퓨터의 주기억이 RAM이기 때문에 한글문자 합성프로그램은 소형 마그네틱·드럼(magnetic drum)에 저장되어있다.

실험시스템에는 포함되어 있지 않지만, 닽·프린터를 인텔리젼트 터미날로서 생각하여 마이크로·콤퓨터를 직접 프린터 속에 탑재 됐을때의 블럭·다이어그램(block-diagram)에(例)를 그림 11.에 표시한다.

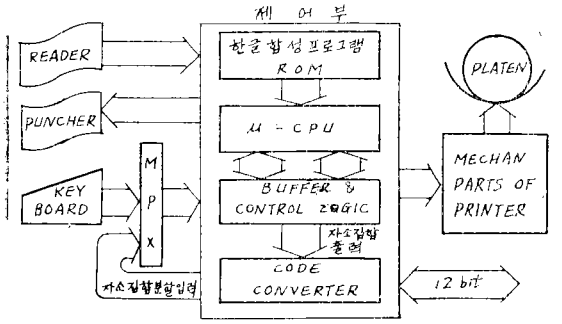


그림 11. 원거리통신에 한함 마이크로·콤퓨터를 내장한 닽·프린터.
Fig. 11. A block diagram of dot printer equipped with micro-computer.

4-3. 입출력 결과

입 력

실험결과로 부터 글자와 글자사이의 띄우기를 자동적으로 구별하는 것은 타자수인 인간에게는 부적합하다는 것을 알았다. 왜냐하면, 현재 출력할 글자의 띄우는 곳을 알기 위해서는 다음 자의 CV가 와야 판명되기 때문에 타자수가 항상 앞의 글자와 다음 글자의 두 자를 외우고 있어야 됨으로 틀리기 쉽다.

따라서, 인간이 계산기에 대해 입력할 때는 띄움을 알리는 다미(dummy)신호를 넣고 통신하고, CPU가 터미날과 통신할 때는 자동적으로 띄움을 판단할 수 있도록 시스템을 구성하는 것이 적합하리라 생각한다.

출 력

이 실험시스템의 출력장치는 그래픽(graphic)·프린터를 사용했다. 출력속도는 한글문자의 모든 패턴(P_1~P_9까지)에 대해 평균 10msec(=100 자소/sec ; INTEL 8008의 실행속도 20μsec)이내에 한자소를 발생시킬 수 있다. (인자 기계부의 기계적 속도는 포함되어 있지 않음.)

그림 12.에 출력 결과를 제시했다. 이 출력결과로 부터 다음과 같은 몇가지 문제점을 고찰한다.

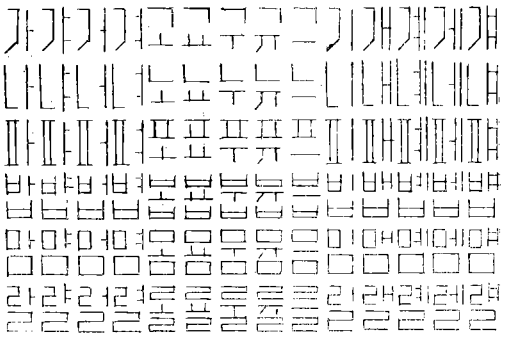


그림 12. 출력결과
Fig. 12. Printed result of this method.

i) 기본자소「ㅗㅛㅜㅠㅡ」를 5×5닽·패턴에 꽉 차게 하면, 그림 13-a와 같은 보기 흉한 글자가 되어 버린다. 반면, 이 문제를 해결하기 위해 패턴들의 맨 밑을 1라인 띄워서 패턴을 만들면, 그림 13-b와 같이 △만큼 크기가 적어져 가지런하게 되지 않는다.

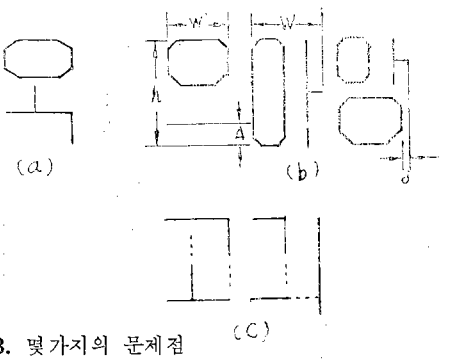


그림 13. 몇가지의 문제점
Fig. 13. Plobrems to be solved.

ii) 가로로 2배(2H) 확대시키면 닽수가 짝수가 되어 앞에서 말한 바와 같이 글자의 품질을 악화시킨다. 그래서, 2H의 경우에는 10×15닽(그림 13-b의 w×h)을 9×15 (w′×h)로 고쳐, 균형 잡히도록 출력시켰다. 단,「앙」과 같은 경우에는 받침「ㅇ」의

가로가 9비트로 줄어 들기 때문에 δ만큼의 오차가 필연적으로 생겼다.

iii) 그림 13-c에 제시한 것처럼 점선으로 표시한 곳까지 인쇄되는 편이 자질(字質)을 향상시킨다. 그러나, 기본 패턴에 어떤 정보를 첨가시키지 않는 한 실현될 수 없는 문제이다.

iv), ii)와 같은 문제의 해결 방법으로서 그림 14.와 같이 수평모음의 기본 패턴을 바꿈으로서 가능한 한 자연미를 띈 글자를 발생시키는 방법도 생각될 수 있다.

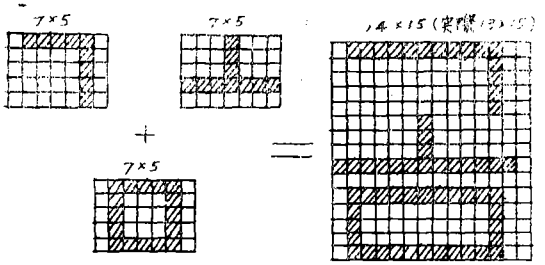


그림 14. 자질을 향상시키는 한 방법
Fig. 14. A method for improvement of good character quarity.

이상과 같이 전체적인 앨고리즘 속에 포함시킬 수 없는 예외적인 경우에는 케이스·바이·케이스로 해결해 나가야 될줄 믿으나, 이것으로 인해 발생하는 프로그램 효율의 저하를 어느 정도까지 억제시킬 것인가 하는 트레이드·오프(trade-off)점에 대해 금후 더욱 검토될 필요가 있다.

그림 15.에 타방식의 임팩트, 난임팩트(non-impact)·프린터의 출력 결과와의 비교예를 보고 있다.

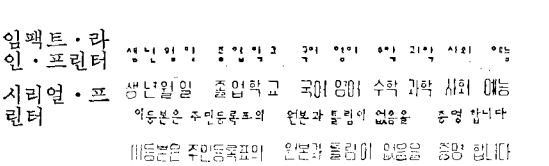


그림 15. 타방식과의 비교
Fig. 15. Comparison with other methods.

5. 맺 음

한글 기본자소를 33자로 함으로서 패턴발생과 합성앨고리즘이 간단화 됨을 보이고, 실제로 시스템을 구성하여 실험한 결과 비교적 양호한 결과를 얻을수 있었다. 따라서, 이 방법이 ASCⅡ폰트(font)에 의해 영문자를 발생시키는 것과 같이 한글을 5×5닻으로 기본자소를 발생시켜 그것을 10×15닻으로 한글을 합성시키는 유효한 수법임을 입증시켰다.

또한, 본 방법은 글자간의 자동 구별을 논의(論外)로 한다면, 한글 입력 키—보드(key-board)의 입력 키—수는 24개에서 33개까지 자유로 선택할 수 있다.

출력속도는 INTEL 8080정도의 CPU를 사용하면 1,000자소/sec이상으로 스피드·업시킬수 있으므로 중속 라인·프린터에도 충분히 합성 출력시킬 수 있다. 활자형의 라인·프린터의 경우는 인쇄 속도가 2,000lpm (line/min.)~수천 lpm이지만, 닻·프린터의 경우는 수십 lpm~250lpm정도 밖에 되지 않는다. 따라서, 한글문자를 이 논문에서 제안한 방법으로 출력시킬 경우에는 닻·프린터를 사용하지 않으면 안되기 때문에 라인·프린터의 출력속도에 필적하는 닻·프린터가 개발되지 않는 한, 라인·프린터에 비해 인쇄속도가 떨어지는 것은 숙명적인 문제가 될 것이다.

주 계산기가 직접 터미날을 관리하는 경우에는 여기서 제안한 코드를 사용하여 내부통신을 할 수 있으나, 원거리 통신의 경우는 전송선로를 유효하게 사용키 위해, 앞에서 논한, 전체적으로 부호화하는 방법을 들 수 있다. 전체적으로 부호화하는 방법으로서, 가변길이 부호화 방식(Huffman 부호, Wyle부호등)이 적합하리라 생각한다. 그러나, 가변길이 부호화 방식을 적용시키기 위해서는 한글의 정보 이론적인 데이터가 필요하다. 이와 같은 연구는 과거에 발표된[4] 바 있지만, 더욱 광범위하게 데이터를 수집할 필요가 있을 뿐더러 이것을 위해서는 한글문자 자동입력(계산기에 대해서)에 관한 연구가[5] 앞서야 됨으로, 필자가 현재 논할 대상이 되지 못한다.

이 연구는 원래 전자계산기를 대상으로 한 한글입출력장치에 관한 것이었지만, 일반 사무용의 닻·프린터나 그래픽·프린터가 개발된다면 충분히 일반 사무용 한글 타이프라이터로서 실현될 수 있다고 생각된다. 마이크로·콤퓨터가 마치 세탁기에 모터가 들어 있는 것처럼 하나의 부품으로서 프린터 속에 내장될 것이다.

감사의 말

와세다대학의 平山 博 교수에게 깊은 감사의 뜻을 표합니다.

필자의 한 사람인 강철희가 한양대학에 재학중 여러가지로 지도해주신 이영근교수, 임제탁교수, 임인칠교수께 사의를 표합니다.

본 연구에 관해 귀중한 조언을 해주신 인하대의 이주근 교수께 감사드립니다.

실험 시스템의 제작에 협력을 아끼지 않은 본 연구실의 4학년생 馬場 宏(현재東芝), 浜口 元彦(현재NEC)군에게 감사드립니다.

參 考 文 獻

1. Joo-Keun Lee; "Korean character display by variable combination method.", Keio Engineering Reports, Vol. 26, No.10 1973
2. 安秀桔 : "한글문자 모아쓰기 Display의 한 방안", 韓電誌, Vol. 12, No. 1, Feb. 1975
3. 이주근 : "한글문자 인식을 위한 대수적 구조", 韓電誌, Vol. 12, No. 2, April 1975
4. 이주근, 최홍문 : "한국어 음절의 Entropy에 관한 연구", 韓電誌, Vol. 11, No. 3, June 1974
5. 이주근, 김홍기 : "위상회전에 의한 필기체 한글의 자동인식,"韓電誌, Vol. 13, No. 1, pp. 23-30, March 1976
6. E.F. Yhap: "Keyboard method for composing chinese characters." Journal of IBM., Vol. 19, No. 1, pp. 60-70, Jan. 1975