บรรณานุกรม

ภาษาไทย

คณะทำงานร่างข้อกำหนดร่วม เพื่อการเขียนโปรแกรมซึ่งแสดงผลเป็นภาษาไทย (TAPIC). มาตรฐานซอฟแวร์สำหรับภาษาไทย (วทท.2.0). กรุงเทพมหานคร: กระทรวงวิทยาศาสตร์ เทคโนโลยีและการพลังงาน, 2534.

คอลฟิลด์ เพื่อคนตาบอด, ห้องสมุด. เปิดโลกกว้างให้คนตาบอด. กรุงเทพมหานคร: มูลนิธิช่วยคนตาบอดแห่งประเทศไทย ในพระบรมราชินูประถัมภ์, ม.ป.ป. (อัดสำเนา)

ชัยยงค์ วงศ์ชัยสุวัฒน์. เพิ่มความจุของดิสค์ ด้วยการบีบอัดข้อมูล (1). วารสารไมโครคอมพิวเตอร์. ฉบับที่ 80 (มีนาคม 2535): 340-345.

ณรงค์ ปฏิบัติสรกิจ. แบบฝึกพิมพ์เบรลล์ไทย. กรุงเทพมหานคร: มูลนิธิช่วยคนตาบอดแห่งประเทศไทย ในพระบรมราชินูประถัมภ์, ม.ป.ป.

พระวรเวทย์พิสิฐ. หลักภาษาไทย. กรุงเทพมหานคร: ศูนย์ภาษาและวรรณคดีไทย คณะอักษรศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2534. (พิมพ์เนื่องในวโรกาสที่สมเด็จพระเทพรัตนราชสุดาฯ สยามบรมราชกุมารี ทรงมีพระชนมายุครบ 3 รอบ).

เยเนวีฟ คอลฟิลด์. อาณาจักรภายใน. กรุงเทพมหานคร: ดอนบอสโกการพิมพ์, 2532.

ยืน ภู่วรวรรณ. เทคนิคพัฒนาการ ภาษาไทยบนไมโคร ตอนที่ 7. วารสารไมโครคอมพิวเตอร์. ฉบับที่ 27 (มกราคม 2530): 97-105.

ภาษาอังกฤษ

Telesensory System Inc. The VersaPoint Brailler Embosser. Mountain View, CA94043: Telesensory Systems Inc., 1987.

Wilai Tanaprakob. Automatic syllable separation for thai word processing. Master's Thesis, University of Electro-communication, 1984.

ภาคผนวก

ภาคผนวก ก

รหัสอักษรเบรลล์สำหรับคอมพิวเตอร์

Dec	Braille	Sym	Name	Description
00	(---4--)	( )	NUL	Delay or pad character*
01	(1-----)	(A)	SOH	Start of heading*
02	(12----)	(B)	STX	Start of text*
03	(1--4--)	(C)	ETX	End of text*
04	(1--45-)	(D)	EOT	End of transmission
05	(1---5-)	(E)	ENQ	Enquire or who?*
06	(12-4--)	(F)	ACK	Acknowledge or yes*
07	(12-45-)	(G)	BEL	Bell or beep*
08	(12--5-)	(H)	BS	Backspace*
09	(-2-4--)	(I)	HT	Horizontal tab*
10	(-2-45-)	(J)	LF	Line feed*
11	(1-3---)	(K)	VT	Vertical tab*
12	(123---)	(L)	FF	Form feed*
13	(1-34--)	(M)	CR	Carriage return*
14	(1-345-)	(N)	SO	Shift out*
15	(1-3-5-)	(O)	SI	Shift in*
16	(1234--)	(P)	DLE	Data link escape*
17	(12345-)	(Q)	DC1	X-ON continue*
18	(123-5-)	(R)	DC2	Device control 2*

Dec	Braille	Sym	Name	Description
19	(-234--)	(S)	DC3	X-OFF or stop*
20	(-2345-)	(T)	DC4	Device control 4*
21	(1-3--6)	(U)	NAK	Negative acknowledge*
22	(123--6)	(V)	SYN	Synchronize*
23	(-2-456)	(W)	ETB	End of text block*
24	(1-34-6)	(X)	CAN	Cancel*
25	(1-3456)	(Y)	EM	End media*
26	(1-3-56)	(Z)	SUB	Substitute*
27	(-2-4-6)	([)	ESC	Escape*
28	(12--56)	(\)	FS	Field separator*
29	(12-456)	(])	GS	Group separator*
30	(---45-)	(^)	RS	Record separator*
31	(---456)	(_)	US	Unit separator*
32	(------)	( )		Space
33	(-234-6)	(!)		Exclamation point
34	(----5-)	(")		Double quote
35	(--3456)	(#)		Number sign
36	(12-4-6)	($)		Dollar sign
37	(1--4-6)	(%)		Percent
38	(1234-6)	(&)		And sign
39	(--3---)	(')		Apostrophe
40	(123-56)	(()		Left parenthesis
41	(-23456)	())		Right parenthesis
42	(1----6)	(*)		Asterisk or star

Dec	Braille	Sym	Name	Description
43	(--34-6)	(+)		Plus sign
44	(-----6)	(,)		Comma
45	(--3--6)	(-)		Hyphen
46	(---4-6)	(.)		Period
47	(--34--)	(/)		Slash
48	(--3-56)	(0)		Zero
49	(-2----)	(1)		One
50	(-23---)	(2)		Two
51	(-2--5-)	(3)		Three
52	(-2--56)	(4)		Four
53	(-2---6)	(5)		Five
54	(-23-5-)	(6)		Six
55	(-23-56)	(7)		Seven
56	(-23--6)	(8)		Eight
57	(--3-5-)	(9)		Nine
58	(1---56)	(:)		Colon
59	(----56)	(;)		Semicolon
60	(12---6)	(<)		Less than
61	(123456)	(=)		Equal sign
62	(--345-)	(>)		Greater than
63	(1--456)	(?)		Question mark
64	(---4--)	(@)		At sign, upper case**
65	(1-----)	(A)		Upper case**
66	(12----)	(B)		Upper case**

Dec	Braille	Sym	Name	Description
67	(1--4--)	(C)		Upper case**
68	(1--45-)	(D)		Upper case**
69	(1---5-)	(E)		Upper case**
70	(12-4--)	(F)		Upper case**
71	(12-45-)	(G)		Upper case**
72	(12--5-)	(H)		Upper case**
73	(-2-4--)	(I)		Upper case**
74	(-2-45-)	(J)		Upper case**
75	(1-3---)	(K)		Upper case**
76	(123---)	(L)		Upper case**
77	(1-34--)	(M)		Upper case**
78	(1-345-)	(N)		Upper case**
79	(1-3-5-)	(O)		Upper case**
80	(1234--)	(P)		Upper case**
81	(12345-)	(Q)		Upper case**
82	(123-5-)	(R)		Upper case**
83	(-234--)	(S)		Upper case**
84	(-2345-)	(T)		Upper case**
85	(1-3--6)	(U)		Upper case**
86	(123--6)	(V)		Upper case**
87	(-2-456)	(W)		Upper case**
88	(1-34-6)	(X)		Upper case**
89	(1-3456)	(Y)		Upper case**
90	(1-3-56)	(Z)		Upper case**

Dec	Braille	Sym	Name	Description
91	(-2-4-6)	([)		Left square bracket, upper case**
92	(12--56)	(\)		back slash, upper case**
93	(12-456)	(])		Right square bracket, upper case**
94	(---45-)	(^)		Up arrow or caret, upper case**
95	(---456)	(_)		Underline, upper or lower case
96	(---4--)	(`)		Grave accent, lower case
97	(1-----)	(a)		Lower case
98	(12----)	(b)		Lower case
99	(1--4--)	(c)		Lower case
100	(1--45-)	(d)		Lower case
101	(1---5-)	(e)		Lower case
102	(12-4--)	(f)		Lower case
103	(12-45-)	(g)		Lower case
104	(12--5-)	(h)		Lower case
105	(-2-4--)	(i)		Lower case
106	(-2-45-)	(j)		Lower case
107	(1-3---)	(k)		Lower case
108	(123---)	(l)		Lower case
009	(1-34--)	(m)		Lower case
110	(1-345-)	(n)		Lower case
111	(1-3-5-)	(o)		Lower case
112	(1234--)	(p)		Lower case
113	(12345-)	(q)		Lower case
114	(123-5-)	(r)		Lower case

Dec	Braille	Sym	Name	Description
115	(-234--)	(s)		Lower case
116	(-2345-)	(t)		Lower case
117	(1-3--6)	(u)		Lower case
118	(123--6)	(v)		Lower case
119	(-2-456)	(w)		Lower case
120	(1-34-6)	(x)		Lower case
121	(1-3456)	(y)		Lower case
122	(1-3-56)	(z)		Lower case
123	(-2-4-6)	({)		Left brace, lower case
124	(12--56)	(|)		Vertical bar, lower case
125	(12-456)	(})		Right brace, lower case
126	(---45-)	(~)		Tilde, lower case
127	(---456)	(DEL)		Delete, upper or lower case

รหัสภาษาไทยสำหรับเครื่องพิมพ์ มอก.988–2532

TAPIC Code ID **11**

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	DLE	SP	0	@	P	`	p	่	๑	เ	ฐ	ภ	ะ	ๆ	๐
1	SOH	DC1	!	1	A	Q	a	q	้	๒	ก	ฑ	ม	ั	แ	๑
2	STX	DC2	"	2	B	R	b	r	๊	๓	ข	ฒ	ย	า	โ	๒
3	ETX	DC3	#	3	C	S	c	s	๋	๔	ฃ	ณ	ร	ำ	ใ	๓
4	EOT	DC4	$	4	D	T	d	t	์	๕	ค	ด	ฤ	ิ	ไ	๔
5	ENQ	NAK	%	5	E	U	e	u	๎	๖	ฅ	ต	ล	ี	ๅ	๕
6	ACK	SYN	&	6	F	V	f	v	๏	๗	ฆ	ถ	ฦ	ึ	ๆ	๖
7	BEL	ETB	'	7	G	W	g	w	๐	๘	ง	ท	ว	ื	็	๗
8	BS	CAN	(	8	H	X	h	x	๑	๙	จ	ธ	ศ	ุ	่	๘
9	HT	EM	)	9	I	Y	i	y	๒	๚	ฉ	น	ษ	ู	้	๙
A	LF	SUB	*	:	J	Z	j	z	๓	๛	ช	บ	ส	ฺ	๊	ฯ
B	VT	ESC	+	;	K	[	k	{	๔	เ	ซ	ป	ห	฿	๋	๏
C	FF	FS	,	<	L	\	l	|	๕	แ	ฌ	ผ	ฬ	์	๐	๑
D	CR	GS	-	=	M	]	m	}	๖	โ	ญ	ฝ	อ	ญ	๎	๑
E	SO	RS	.	>	N	^	n	~	๗	ใ	ฎ	พ	ฮ	ฯ	๏	๒
F	SI	US	/	?	O	_	o	DEL	๘	ไ	ฏ	ฟ	ฯ	ๆ	๐	

▦ =ไม่มีตัวอักษร ▨ =ต่างกับมาตรฐาน

ภาคผนวก ค

รูปและ เสียงในภาษาไทย

ในภาษาไทยมีการใช้รูปและ เสียงต่าง ๆ พอสรุปได้ดังนี้

รูปและ เสียงสระภาษาไทย

รูปสระภาษาไทย มี 21 รูป ดังนี้

1. ะ เรียก วิสรรชนีย์ สำหรับประหลัง เป็นสระ อะ และ ประสมกับรูปอื่น เป็นสระ เอะ แอะ โอะ เอาะ เอียะ เออะ อัวะ

2. ั เรียก ไม้หันอากาศหรือไม้ผัด สำหรับ เขียนข้างบน เป็นสระ อะ เมื่อมีตัวสะกด และ ประสมกับรูปอื่น เป็น สระ อัวะ อัว

3. ็ เรียก ไม้ไต่คู้ สำหรับ เขียนข้างบนแทนวิสรรชนีย์ ในสระบางตัวที่มีตัวสะกด เช่น เอ็น แอ็น อ็อน และ ใช้ประสมกับ ก เป็น สระ เอาะ มีไม้โท คือ ก็ อ่านว่า เก้าะ

4. า เรียก ลากข้าง สำหรับ เขียนข้างหลัง เป็นสระ อา และประสมกับรูปอื่น เป็น สระ เอาะ อำ อา

5. ิ เรียก พินทุอิ สำหรับ เขียนข้างบน เป็นสระ อิ และประสมกับรูปอื่น เป็น สระ อี อึ อือ เอียะ เอีย เออะ เออ และใช้แทนตัว อ ของสระ เออ เมื่อมีตัวสะกด เช่น เกอน เขียน เป็น เกิน

6. ี เรียก ฝนทอง สำหรับ เขียนข้างบนพินทุอิ เป็น สระ อี และ ประสมกับรูปอื่น เป็น สระ เอียะ เอีย

7. ึ เรียก นฤคหิต หรือหยาดน้ำค้าง สำหรับ เขียนข้างบนพินทุอิ เป็น สระ อึ

8. " เรียก พันหนู สำหรับเขียนข้างบนพินทุอิ เป็น สระ อือ และประสมกับรูปอื่นเป็น สระ เอือะ เอือ

9. ุ เรียก ตีนเหยียด สำหรับเขียนข้างล่าง เป็น สระ อุ

10. ู เรียก ตีนคู้ สำหรับเขียนข้างล่าง เป็น สระ อู

11. เ เรียก ไม้หน้า สำหรับเขียนข้างหน้ารูปเดี่ยวเป็น สระ เอ สองรูปเป็น สระ แอ และประสมกับรูปอื่นเป็น สระ เอะ แอะ เอา เออะ เออ เอียะ เอีย เอือะ เอือ เอา

12. ใ เรียก ไม้ม้วน สำหรับเขียนข้างหน้าเป็น สระ ใอ

13. ไ เรียก ไม้มลาย สำหรับเขียนข้างหน้าเป็น สระ ไอ

14. โ เรียก ไม้โอ สำหรับเขียนข้างหน้าเป็น สระ โอ เมื่อประวิสรรชนีย์เป็น สระ โอะ

15. อ เรียก ตัวออ สำหรับเขียนข้างหลังเป็น สระ ออ และ ประสมกับรูปอื่นเป็นสระ อือ เออะ เออ เอือะ เอือ

16. ย เรียก ตัวยอ สำหรับประสมกับรูปอื่นเป็น สระ เอียะ เอีย

17. ว เรียก ตัววอ สำหรับประสมกับรูปอื่นเป็น สระ อัวะ อัว

18. ฤ เรียก ตัวรึ สำหรับเขียนเป็น สระ ฤ

19. ฤา เรียก ตัวรือ สำหรับเขียนเป็น สระ ฤา

20. ฦ เรียก ตัวลึ สำหรับเขียนเป็น สระ ฦ

21. ฦา เรียก ตัวลือ สำหรับเขียนเป็น สระ ฦา

เสียงของสระภาษาไทยประกอบด้วยเสียงสระเดียว และผสม ทำให้เกิดเสียงได้ 24 เสียง ดังนี้

1. สระเดี่ยว

/ ิ / อย่างในคำ กิน บิน ลิ้น

/ ี / อย่างในคำ ปีน รีบ ดีด

/ เ–ะ / อย่างในคำ เก็ง เร่ง เม่น

/ เ / อย่างในคำ เกรง เอม เก้ง

/ แ–ะ / อย่างในคำ แก่ง แว่น แพะ

/ แ / อย่างในคำ แต่ แรม แม้ว

/ ึ / อย่างในคำ ยึด ซึม คึก

/ ื / อย่างในคำ ปลื้ม ดื้อ จืด

/ เ–อะ / อย่างในคำ เงิน เปิ่น เยอะ

/ เ–อ / อย่างในคำ เธอ เดิน เสริม

/ ะ / อย่างในคำ งำ สังข์ จะ

/ า / อย่างในคำ งา จาน ยาก

/ ุ / อย่างในคำ จุ ซุง คุด

/ ู / อย่างในคำ หมู่ ตูม หูด

/ โ–ะ / อย่างในคำ โต๊ะ นม บด

/ โ / อย่างในคำ โต โรง โหล

/ เ–าะ / อย่างในคำ เมาะ บ่อน ก็

/ อ / อย่างในคำ บ่อ ชอบ นอน

2. สระประสม

เอียะ อย่างในคำ เดียะ เผียะ เชียะ

เอีย อย่างในคำ เมีย เรียน เบียด

เออะ อย่างในคำ เดือด เกือก เผือก

เอือ อย่างในคำ เรือ เดือน เยื้อน

อัวะ อย่างในคำ ผัวะ ยัวะ

อัว อย่างในคำ กลัว บวม ด้วย

<u>รูปและ เสียงของพยัญชนะไทย</u>

พยัญชนะไทย มี 44 รูป และคิดรวมเป็นเสียงได้ 21 เสียง โดยแยกกลุ่มเสียงได้ดังนี้

1. ก
2. ข ฃ ค ฅ ฆ
3. ง
4. จ
5. ฉ ช ฌ
6. ซ ศ ษ ส
7. ญ ย
8. ฎ ด ท (บางคำ)
9. ฏ ต
10. ฐ ถ ฑ (บางคำ ฒ ท ธ)
11. ณ น
12. บ
13. ป
14. ผ พ ภ
15. ฝ ฟ
16. ม
17. ร
18. ล ฬ
19. ว
20. ห ฮ
21. อ

วรรณยุกต์

สำหรับรูปวรรณยุกต์ ที่กำหนดโทนเสียงนั้นมี 4 รูป และ ผันตามโทนของเสียง เป็น 5 เสียง คือ

กา ก่า ก้า ก๊า ก๋า

มาตราตัวสะกด

ตามหลักภาษาไทย มีมาตราตัวสะกดได้ 9 แม่ โดยเรียงลำดับตัวอักษรดังนี้

1. แม่ ก กา คือแม่อักษรที่ไม่มีตัวสะกด ได้แก่ อักษรที่ประสมด้วยสระ 28 ตัว ดังตัวอย่างที่ใช้ ก ประสม ดังนี้ กะ กา กิ กี กึ กือ กุ กู เกะ เก แกะ แก โกะ โก เกาะ กอ เกอะ เกอ เกียะ เกีย เกือะ เกือ กัวะ กัว กฤ กฤา

2. แม่ กก คือแม่อักษรที่สะกดด้วยตัว ก ข ค ฆ

3. แม่ กง คือแม่อักษรที่สะกดด้วยตัว ง

4. แม่ กด คือแม่อักษรที่สะกดด้วยตัว จ ฉ ช ซ ฌ ฆ ฐ ฑ ฒ ด ต ถ ท ธ ศ ษ ส

5. แม่ กน คือแม่อักษรที่สะกดด้วยตัว ญ ณ น ร ล ฬ และ ที่ใช้ ร หัน โดยไม่มีตัวสะกด หรือมีตัวการันต์ เช่นคำว่า (เลือก)สรร (สร้าง)สรรค์

6. แม่ กบ คือแม่อักษรที่สะกดด้วยตัว บ ป ผ ฝ พ ฟ ภ

7. แม่ กม คือแม่อักษรที่สะกดด้วยตัว ม และที่ประสมด้วยสระ อำ

8. แม่ เกย คือแม่อักษรที่สะกดด้วยตัว ย และที่ประสมด้วยสระ ใอ ไอ

9. แม่ เกอว คือแม่อักษรที่สะกดด้วยตัว ว และที่ประสมด้วยสระ เอา

<u>อักษรนำ</u>

อักษรนำ ได้แก่พยัญชนะ 2 ตัว รวมกันอยู่ในสระเดียวกัน ตัวอย่างโครงร่างของอักษรนำได้แก่

1. อักษรกลางนำอักษรเดี่ยว

1.1 กณ กากณิก

1.2 กน กนก

1.3 กร ศักราช ธมกรก กรก กรอด กรีส

1.4 กรว จักรวรรดิ

1.5 กล กลาบ กล่ำ

1.6 จน ปัจนึก

1.7 จม จมูก

1.8 จร จรด จรวด จรัส จริต

1.9 จว จวัก เจว็ด

1.10 ดง ดงิด ดงุ่น

1.11 ดน ดนุ โดนด

1.12 ตร อุตริ

1.13 ตล ตลก ตลอด ตลาด ตลับ

1.14 ตว ตวัก ตวัด ตวาด

1.15 ปน กัปนก

1.16 ปร ปรอด ปรอท ปริตร

1.17 ปล ปลก ปลัด

1.18 อง องุ่น

1.19 อน อนาถ อเนก อนึ่ง

1.20 อม อมาตย์

1.21 อร อร่อย อร่าม อริ แอร่ม

1.22 อล อล่องฉ่อง อล่างฉ่าง อะลุ้มอล่วย

2. อักษรสูงนำอักษรเดี่ยว

2.1 ขณ ขณะ ลักขณะ

2.2 ขน ขนด ขนม ขนาน ขนาด เขนย โขนง

2.3 ขม ขมวด ขมีขมัน เขมร เขม่า

2.4 ขย ขย่ม ขยัน

2.5 ขร ขรม อักขระ

2.6 ขล เมขลา

2.7 ขฟ กักขฬะ

2.8 ฉง ฉงน ฉงาย

2.9 ฉน ฉนวน แนาก ไฉน ฉนำ

2.10 ฉม ฉมวก ฉมัง

2.11 ฉล ฉลวย ฉลอง ฉลาด

2.12 ฉว ฉวัดเฉวียน ฉวาง ฉวี

2.13 ฐย ปัฐยาววัตรฉันท์

2.14 ถง ถงาด แถง ไถง

2.15 ถน ถนน ถนัด ถนัน ถนิม

2.16 ถม ถมึง ปถมัง

2.17 ถล ถลก ถลน ถลอก ถลา

2.18 ถว ถวัลย์ ถวาย ถวิล

2.19 ผง ผงก ผงม ผงะ

2.20 ผน ผนวก ผนวช ผนัง

2.21 ผย ผยอง เผยอ

2.22 ผล ผลิต ผลึก

2.23 ผว ผวา ผวาด

2.24 ฝร ผรั่ง ผรั่น

2.25 ศน ปริศนา

2.26 ศย แพศยา

2.27 ศร ศฤงคาร

2.28 ศล โศลก ไศล

2.29 ศว เศวต พิศวง

2.30 ษณ กฤษณา ลักษณะ

2.31 ษย ริษยา

2.32 สง สงบ สงวน สง่า เสงี่ยม

2.33 สน สนุก สนาน

2.34 สม สมอ เสมอ สมาน

2.35 สย สยาม แสยะ สยบ

2.36 สร สระ แสรก

2.37 สล สลด สลบ สลาก

2.38 สว สวะ สว่าง

2.39 สวย สวยม สวยมพร สวยมภู

ภาคผนวก ง

พยัญชนะควบกล้ำที่ใช้กับสระผสม

สระผสมในภาษาไทยที่มีรูปสระในการเขียนอักษรเบรลล์ ซึ่งมีเพิ่มเติมจากอักษรตาดีภาษาไทย และมีการใช้ที่แตกต่างกัน คือมีการเรียงลำดับไม่เหมือนกัน ดังนั้นในการแปลงคำในแบบอักษรตาดีเป็นอักษรเบรลล์ ในคำที่เป็นสระผสมนี้ จะต้องตรวจสอบพยัญชนะควบกล้ำที่สามารถนำมาใช้กับสระผสมเหล่านี้ให้ถูกต้อง ก่อนที่จะสรุปว่าคำคำนั้นเป็นคำที่ใช้พยัญชนะควบกล้ำร่วมกับสระผสม วิไล ธนประกอบ (2527) ได้วิจัยในวิทยานิพนธ์ เรื่องหลักการแบ่งพยางค์ภาษาไทยโดยคอมพิวเตอร์ สามารถสรุปพยัญชนะควบกล้ำที่ใช้กับสระผสมเป็นตารางได้ดังนี้

1. ตารางสำหรับพยัญชนะควบกล้ำที่ใช้กับสระผสม เ-า

พยัญชนะควบกล้ำตัวที่สอง	พยัญชนะควบกล้ำตัวแรก
ร	ป ค พ ศ
ง	ห
ม	ห ข ช
ย	ห ข
ว	ห
ล	ก ห ป ค พ ข

2. ตารางสำหรับพยัญชนะควบกล้ำที่ใช้กับสระผสม เ-ะ

พยัญชนะควบกล้ำตัวที่สอง	พยัญชนะควบกล้ำตัวแรก
ร	ป
ล	ข ฝ

3. ตารางสำหรับพยัญชนะควบกล้ำที่ใช้กับสระผสม เ-าะ

พยัญชนะควบกล้ำตัวที่สอง	พยัญชนะควบกล้ำตัวแรก
ร	ก ป ค พ
ม	ห
ย	ห
ล	ห ป พ ผ
พ	ฉ

4. ตารางสำหรับพยัญชนะควบกล้ำที่ใช้กับสระผสม เ-อ

พยัญชนะควบกล้ำตัวที่สอง	พยัญชนะควบกล้ำตัวแรก
น	ส
ร	ก ห ป

ม	ห ส
ย	ห
ล	ก ห ผ

5. ตารางสำหรับพยัญชนะควบกล้ำที่ใช้กับสระผสม เ-ิ

พยัญชนะควบกล้ำตัวที่สอง	พยัญชนะควบกล้ำตัวแรก
ร	ก ส จ พ
อ	ผ
ย	ห ผ
ล	ต ห ค พ ถ
ด	ผ
ช	ผ

6. ตารางสำหรับพยัญชนะควบกล้ำที่ใช้กับสระผสม เ-อะ

พยัญชนะควบกล้ำตัวที่สอง	พยัญชนะควบกล้ำตัวแรก
ร	ก ป

7. ตารางสำหรับพยัญชนะควบกล้ำที่ใช้กับสระผสม เอีย

พยัญชนะควบกล้ำตัวที่สอง	พยัญชนะควบกล้ำตัวแรก
น	ห ส พ
ร	ก ต ห ค พ ป
ม	ห ส
ย	ส
ว	ก ห ฉ
ล	ก ห ป พ ฉ
บ	ส
ษ	ก

8. ตารางสำหรับพยัญชนะควบกล้ำที่ใช้กับสระผสม เอียะ

พยัญชนะควบกล้ำตัวที่สอง	พยัญชนะควบกล้ำตัวแรก
ร	ป
ล	พ ฝ

9. ตารางสำหรับพยัญชนะควบกล้ำที่ใช้กับสระผสม เอือ

พยัญชนะควบกล้ำตัวที่สอง	พยัญชนะควบกล้ำตัวแรก
น	ห
ร	ป ค พ
ง	ห
ม	ห ส
ย	ห
ล	ก อ ม ห ป ค ถ

10. ตารางสำหรับพยัญชนะควบกล้ำที่ใช้กับสระผสม เอือะ

พยัญชนะควบกล้ำตัวที่สอง	พยัญชนะควบกล้ำตัวแรก
ร	ป

11. ตารางสำหรับพยัญชนะควบกล้ำที่ใช้กับสระผสม แ–ะ

พยัญชนะควบกล้ำตัวที่สอง	พยัญชนะควบกล้ำตัวแรก
ร	ค
ม	ห
ย	ห ส
ว	ห ช
ล	ห พ ผ

12. ตารางสำหรับพยัญชนะควบกล้ำที่ใช้กับสระผสม โ–ะ

พยัญชนะควบกล้ำตัวที่สอง	พยัญชนะควบกล้ำตัวแรก
น	ห ฉ
ร	ท ห ป ค พ ศ
ม	ห ช
ย	ห พ ช
ว	ห
ล	ห ห ค พ ผ

ภาคผนวก จ

โปรแกรมจัดพิมพ์เอกสารอักษรเบรลล์ไทย/อังกฤษ

```
/*********************************************************************
   PROGRAM : TBRL.C
    MODEL   : LARGE
    AUTHOR  : MR. SUKHUM MAHITDHIHARN
    TITLE   : BRAILLE EMBOSSED PRINTING FROM THAI/ENGLISH WORD PROCESSOR
    DATE    : JANUARY 20,1992.

   ** Aposthophe must be double charactor ('') **
    ** Pintoo in CU-Writer press shift-B **
 *********************************************************************/

#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include "thai.h"
#include "tbrl.h"

/*---------- THAI CODE TRANSLATION TABLE ----------*/
Byte ku_to_tis[] = {
                0x98, 0x99, 0x9a, 0x9b, 0x96, 0x95, 0x93, 0x92,
                0x90, 0x91, 0x8f, 0x20, 0x20, 0x20, 0x20, 0x20,
                0xf0, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7,
                0xf8, 0xf9, 0xa3, 0xa5, 0x20, 0x20, 0x20, 0x20,
                0xa0, 0xa1, 0xa2, 0xa4, 0xa6, 0xa7, 0xa8, 0xa9,
                0xaa, 0xab, 0xac, 0xad, 0xae, 0xaf, 0xb0, 0xb1,
                0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9,
```

```
        0xba, 0xbb, 0xbc, 0xbd, 0xbe, 0xbf, 0xc0, 0xc1,
        0xc2, 0xc3, 0xc4, 0xc5, 0xc7, 0xc8, 0xc9, 0xca,
        0xcb, 0xcc, 0xcd, 0xce, 0xd0, 0xc6, 0xd2, 0xd3,
        0xe0, 0xe1, 0xe2, 0xe3, 0xe4, 0xe6, 0xcf, 0xd8,
        0xd9, 0xd4, 0xd5, 0xd6, 0xd7, 0xd1, 0xed, 0xe7,
        0xe8, 0xe9, 0xea, 0xeb, 0xec, 0xda, 0x80, 0x81,
        0x82, 0x83, 0x84, 0x85, 0x86, 0x87, 0x88, 0x89,
        0x8a, 0x8b, 0x8c, 0x8d, 0x8e, 0x9e, 0x9f, 0xdf,
        0x20, 0x20, 0xee, 0xef, 0xfa, 0xfb, 0xfe, 0xff
};

Byte tis_to_ku[] = {
        0xe6, 0xe7, 0xe8, 0xe9, 0xea, 0xeb, 0xec, 0xed,
        0xee, 0xef, 0xf0, 0xf1, 0xf2, 0xf3, 0xf4, 0x8a,
        0x88, 0x89, 0x87, 0x86, 0x80, 0x85, 0x84, 0x81,
        0x80, 0x81, 0x82, 0x83, 0x88, 0x83, 0xf5, 0xf6,
        0xa0, 0xa1, 0xa2, 0x9a, 0xa3, 0x9b, 0xa4, 0xa5,
        0xa6, 0xa7, 0xa8, 0xa9, 0xaa, 0xab, 0xac, 0xad,
        0xae, 0xaf, 0xb0, 0xb1, 0xb2, 0xb3, 0xb4, 0xb5,
        0xb6, 0xb7, 0xb8, 0xb9, 0xba, 0xbb, 0xbc, 0xbd,
        0xbe, 0xbf, 0xc0, 0xc1, 0xc2, 0xc3, 0xcd, 0xc4,
        0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xca, 0xcb, 0xd6,
        0xcc, 0xdd, 0xce, 0xcf, 0xd9, 0xda, 0xdb, 0xdc,
        0xd7, 0xd8, 0xe5, 0x20, 0x20, 0x20, 0x20, 0xf7,
        0xd0, 0xd1, 0xd2, 0xd3, 0xd4, 0x20, 0xd5, 0xdf,
        0xe0, 0xe1, 0xe2, 0xe3, 0xe4, 0xde, 0xfa, 0xfb,
        0x90, 0x91, 0x92, 0x93, 0x94, 0x95, 0x96, 0x97,
        0x98, 0x99, 0xfc, 0xfd, 0x20, 0x20, 0xfe, 0xff
};

void main(int argc, char *argv[])
{
    struct tbraille *tb;        /* system variable */
    Byte ch, *strptr;
    char *fptr, fname[81];      /* input file 80 column max */
```

```
int  i, ccount, opt;
extern  char *optarg;       /* current getopt argument pointer */
extern  int optind;
FILE *infile;

if ((tb = (struct tbraille *)malloc(sizeof(struct tbraille))) == NULL)  {
    fprintf(stderr, "Not enough memory!\n");
    exit(1);
}
initsysvar(tb);             /* initial system variable */
tb->charcode = TIS;         /* Default TIS. thai code */
tb->editing = TRUE;         /* Default create editing */

/*--------- SET CONTROL VALUE TO OPTION ---------*/
while ((opt = getopt(argc, argv, "kenr:")) >= 0)    {
    switch (opt)    {
        case 'k' : tb->charcode = KU;               break;
        case 'e' : tb->editing = TRUE;              break;
        case 'n' : tb->editing = FALSE;         break;
        case 'r' : tb->maxbcol = atoi(optarg);      break;
        default  : fprintf(stderr,"Invalid option!\n");
            exit(1);
    }
}

if (argc < 2 || argc <= optind) {
    printf("Enter file name :");
    scanf("%s", fname);
}
else
    strcpy(fname, argv[argc-1]);

if ((infile = fopen(fname, "r")) == NULL)   {
    fprintf(stderr, "%s not found!\n", fname);
    exit(1);
```

```
}

if ((fptr = (char *)strchr(fname, '.')) != NULL)
    *fptr = NULL;
strcat(fname, ".emb");

if ((tb->embfile = fopen(fname, "w")) == NULL)  {
    fprintf(stderr, "Cann't open file %s\n", fname);
    exit(1);
}

if (tb->editing)    {           /* create editing file */
    if ((fptr = (char *)strchr(fname, '.')) != NULL)
        *fptr = NULL;
    strcat(fname, ".prf");
    if ((tb->prffile = fopen(fname, "w")) == NULL)  {
        fprintf(stderr, "Cann't open file %s\n", fname);
        exit(1);
    }
}

/*------ Get ASCII string and convert it to braille ------*/
while((tb->workptr = fgets(tb->workline, MAXCOL, infile)) != NULL)  {
    if (*(tb->workptr) == '.')      /* filter CW dot command */
        continue;

    if (tb->charcode == KU)
        ku2tis(tb->workline);       /* Use TIS code in program */

    ccount = strlen(tb->workline);  /* chars column count */
    if (tb->workline[ccount-1] == '\n')
        ccount--;                   /* not count '\n' */
    tb->bcount = 0;                 /* braille column count */
    tb->bcol_error = FALSE;         /* out of right margin error */
    strptr = tb->workptr;           /* current pointer for consider */
```

```
while (*strptr) {               /* NULL terminated string */
    if (isupper(*strptr))
        do_uppercase(strptr, tb);
    /*  else if (iscntrl(*strptr))  */

    else if (*strptr == '\'')
        do_singlequot(strptr, tb);

    else if (*strptr == '"')
        do_doublequot(strptr, tb);

    else if (isdigit(*strptr) || istdigit(*strptr))
        do_numberchar(tb);

    else if (*strptr == SaraA || *strptr == SaraAir || *strptr == Sara0)
        /* maybe เ, เ-ะ, เ-า, เ-าะ, เ-ิ, เ-ิะ, เ-ี, เ็, เ็ะ, เ็ด, เ็อ, เ็อะ */
        /* maybe แ, แ-ะ */
        /* maybe ๆ, ๆ-ะ */
        do_sara(strptr, tb);

    else if (*strptr == HunAkad)
        /* maybe ่, ่า, ่าะ */
        do_hanakad(strptr, tb);

    else if (istonal(*strptr) && strptr[1] == SaraUm)
        do_saraum(strptr, tb);      /* if folow with SaraUm : swap it */

    else    {
        tb->capital = FALSE;        /* reset status */
        tb->numberchar = FALSE;
        if (tb->editing)
            put_editing(*strptr);
    }
    put_braille(*strptr);   /* strptr may be change by above function */
```

```
            (tb->workptr)++;        /* next char */
            strptr = tb->workptr;
        }

        if (tb->editing)
            fprintf(tb->prnfile,
                "==> There are %2d charactor and %2d braille cell\n\n",
                ccount, tb->bcount);
    }

    fclose(infile);
    fclose(tb->embfile);
    if (tb->editing)
        fclose(tb->prnfile);        /* close editing file */
    free(tb);
}

/*-------------------------------------------------------*

    UPPER CASE CHARACTOR:
        put CAPITAL SIGN before charactor.
        put CAPITAL SIGN DOUBLE before upper case word.

 *-------------------------------------------------------*/
void do_uppercase(Byte *strptr, struct tbraille *tb)
{
    if (tb->editing)
        put_editing(*strptr);
    if (!(tb->capital))  {   /* previous is not uppercase */
        if (isupper(strptr[1]))  {
            tb->capital = TRUE;     /* group of uppercase */
            put_braille(CAPITALSIGN);
            put_braille(CAPITALSIGN);
        }
        else    {
```

```
            tb->capital = FALSE;    /* single uppercase */
            put_braille(CAPITALSIGN);
        }
    }
}

/*-------------------------------------------------------*

    SINGLE QUOT:
        if double single quot
            put apostrophe
        if never open single quot
            put single quot open
        else
            put single quot close

*-------------------------------------------------------*/
void do_singlequot(Byte *strptr, struct tbraille *tb)
{
    if (tb->editing)
        put_editing(*strptr);
    if (strptr[1] == '\'')  {           /* double apostrophe */
        *strptr = APOSTROPHE;
        tb->workptr += 1;
    }
    else if (tb->singleQuotOpen)    {   /* have been opened? */
        tb->singleQuotOpen = FALSE;
        put_braille(CLOSEQUOT);
        *strptr = CLOSESINGLE;
    }
    else    {       /* never open quot */
        tb->singleQuotOpen = TRUE;
        put_braille(OPENSINGLE);
        *strptr = OPENQUOT;
    }
```

```
}

/*-------------------------------------------------------*

    DOUBLE QUOT:
        if never open double quot
            put double quot open
        else
            put double quot close

*-------------------------------------------------------*/
void do_doublequot(Byte *strptr, struct tbraille *tb)
{
    if (tb->editing)
        put_editing(*strptr);
    if (tb->doubleQuotOpen) {           /* have opened? */
        tb->doubleQuotOpen = FALSE;
        *strptr = CLOSEQUOT;
    }
    else    {       /* never open quot */
        tb->doubleQuotOpen = TRUE;
        *strptr = OPENQUOT;
    }
}

void do_numberchar(struct tbraille *tb)
{
    if (tb->editing)
        put_editing(*(tb->workptr));
    if (!(tb->numberchar))  {       /* previous is not number */
        tb->numberchar = TRUE;
        put_braille(NUMBERSIGN);
    }
}
```

```
/*-------------------------------------------------------*

    SARAUM (ั):
        if folow by tonal
            swap tonal and SaraUm

*-------------------------------------------------------*/
void do_saraum(Byte *strptr, struct tbraille *tb)
{
    Byte ch;

    ch = *strptr;
    *strptr = strptr[1];        /* swap tonal and SaraUm */
    strptr[1] = ch;
    if (tb->editing)
        put_editing(*strptr);
}

/*-------------------------------------------------------*

    PART OF VOWEL:
    maybe เ, เ-ะ, เ-า, เ-าะ, เ-ิ, เ-ิะ, เ-, เ็, เ็ะ, เ็ิ, เ็ิะ, เ-ะ, เ-ะ
    There are 3 pattern of vowel
    V C [C] [t] V [V] [G] :     เ-ะ, เ-า, เ-าะ, เ-ิ, เ-ิะ, เ-ะ, เ-ะ
    V C [C] V [t] S [S] [G] :   เ-
    V C [C] V [t] V [S[S]] [G] : เ็, เ็ะ, เ็ิ, เ็ิะ
    Arrange consonant and part of vowel to braille sequent.

*-------------------------------------------------------*/
void do_sara(Byte *strptr, struct tbraille *tb)
{
    int i,
        vowel_len,                  /* part of vowel lenght */
        single_lead_conson  = FALSE,    /* status */
        double_lead_conson  = FALSE,
```

```
    tonal_include       = FALSE,
    conson_error        = FALSE,
    tonal_error         = FALSE,
    editing             = FALSE;
Byte first_lead_conson  = 0,        /* charactor for output */
     second_lead_conson = 0,
     tonal = 0;
Byte *Sara_test,        /* table selected */
     part_vowel;        /* part of vowel name */
/*-------- related vowel table --------*/
static Byte SaraA_test[] = {SaraAh, SaraR, OrAng, SaraIe, SaraE, SaraUee, NULL};
static Byte SaraAir_test[] = {SaraAh, NULL};
static Byte SaraO_test[] = {SaraAh, NULL};

switch (*strptr)    {   /* Select table */
    case SaraA :
        Sara_test = SaraA_test;
        break;
    case SaraAir :
        Sara_test = SaraAir_test;
        break;
    case SaraO :
        Sara_test = SaraO_test;
        break;
    default :
        return;     /* call error */
}

for (i=1; i<=4; i++)    {       /* fine related vowel */
    if ((strchr(Sara_test, strptr[i]) != NULL))
        break;
}
/*------ related vowel at position of i ------*/

/*------ Check for a part of vowel name ------*/
```

```
part_vowel = whatvowel(strptr, i, &vowel_len);

if (part_vowel && i > 1 && i <= 4)  {   /* position must at 2,3,4 */
    if (isconson(strptr[1]))    {       /* next of 1st-vowel must conson */
        first_lead_conson = strptr[1];
        switch (i)  {       /* fine first, secont leading consonant */
            case 4 :        /* for pattern: VC[C][t]V[V][G] */
                if (part_vowel >= SARAUO && part_vowel <= SARAUEH)  {
                    conson_error = TRUE;
                    break;              /* pattern VC[C]V[t].. (i<4) */
                }
                double_lead_conson = TRUE;
                if (istonal(strptr[3])) {
                    tonal_include = TRUE;
                    tonal = strptr[3];
                }
                else if (isconson(strptr[3]))
                    second_lead_conson = strptr[3];
                else    {
                    conson_error = TRUE;
                    break;
                }
            case 3 :        /* for other pattern */
                if (istonal(strptr[2])) {
                    if (tonal_include)  {
                        tonal_error = TRUE;
                        break;
                    }
                    else    {
                        tonal_include = TRUE;
                        tonal = strptr[2];
                    }
                }
                else if (isconson(strptr[2]))   {
                    if (i == 4 && isconson(strptr[3]))   {
```

```
                conson_error = TRUE;    /* Pos.1-2-3 are conson! */
                break;
            }
            else    {
                double_lead_conson = TRUE;
                second_lead_conson = strptr[2];
                break;
            }
        }
        else    {
            conson_error = TRUE;
            break;
        }
    case 2 :
        if (!double_lead_conson)
            single_lead_conson = TRUE;
            if (part_vowel == SARAUOE && first_lead_conson == ToreTaham)
                conson_error = TRUE;        /* except */
        break;
}
if (!conson_error && !tonal_error)   {      /* convert & output */
    if (strptr[i] == SaraE || strptr[i] == SaraUee) {
        if (istonal(strptr[i+1]))   {   /* may follow by tonal */
            tonal_include = TRUE;
            tonal = strptr[i+1];
        }
    }
    if (single_lead_conson) {
        if (tb->editing)    {
            put_editing('\\');
            puts_editing(strptr, vowel_len, tb);
            put_editing('\\');
            editing = TRUE;
        }
        put_braille(first_lead_conson);
```

```
                *strptr = expand_vowel(part_vowel, tonal_include, tonal, tb);
                tb->workptr += vowel_len;       /* skip to last column of vowel */
            }
            else if (double_lead_conson && posible_double_conson( \
                first_lead_conson, second_lead_conson, part_vowel))  {
                if (tb->editing)    {
                    put_editing('\\');
                    puts_editing(strptr, vowel_len, tb);
                    put_editing('\\');
                    editing = TRUE;
                }
                put_braille(first_lead_conson);
                put_braille(second_lead_conson);
                *strptr = expand_vowel(part_vowel, tonal_include, tonal, tb);
                tb->workptr += vowel_len;       /* skip to last column of vowel */
            }
        }
    }
    if (tb->editing && !editing)
        put_editing(*strptr);
}

/*-------------------------------------------------------*

    PART OF VOWEL:
    maybe ◌ฺ, ◌ฺ◌ะ
    Pattern is :    C [C] V [t] V [V] [G]
    Arrange consonant and part of vowel to braille sequent.

*-------------------------------------------------------*/
void do_hanakad(Byte *strptr, struct tbraille *tb)
{
    int i,
        vowel_len,          /* part of vowel lenght */
```

```
    tonal_include   = FALSE;
    Byte tonal = 0;         /* charactor for output */
    Byte part_vowel;        /* part of vowel name */

    for (i=1; i<=2; i++)    {       /* fine related vowel */
        if (strptr[i] == WoreWaan)
            break;
    }
    /*------ related vowel at position of i ------*/

    /*--- position must at 1 | 2 && Check for a part of vowel name ---*/
    if (i <= 2 && (part_vowel = whatvowel(strptr, i, &vowel_len)) > FALSE) {
        if (istonal(strptr[1])) {       /* follow HanAkad may be tonal */
            tonal_include = TRUE;
            tonal = strptr[1];
        }
        if (tb->editing)    {
            put_editing('\\');
            puts_editing(strptr, vowel_len, tb);
            put_editing('\\');
        }
        *strptr = expand_vowel(part_vowel, tonal_include, tonal, tb);
        tb->workptr += vowel_len;       /* skip to last column of vowel */
    }
    else if (tb->editing)
        put_editing(*strptr);
}

/*-------------------------------------------------------------*

    Fine name of part of vowel.
    str  = string of input text begin with first vowel of part of vowel.
    i    = index of string for first related vowel.
    last = return last position of the vowels.
    Return: number of the part of vowel name.
```

```
        FALSE, if not a part of vowel.

*-------------------------------------------------------*/
Byte whatvowel(Byte *str, int i, int *last)
{
   int sara_name, j;

   *last = 0;   /* initial */

   switch (*str)  {
      case SaraA :            /* 'เ' */
         switch (str[i]) {       /* first related vowel */
            case SaraAh :        /* 'ะ' */
               sara_name = SARAAEH;    /* เ-ะ */
               *last = i;
               break;
            case SaraR :         /* 'า' */
               if (str[i+1] == SaraAh) {
                  sara_name = SARAAOH;
                  *last = i+1;
               }
               else   {
                  sara_name = SARAAO;
                  *last = i;
               }
               break;          /* เ-าะ : เ-า */
            case OrAng :         /* 'อ' */
               if (str[i+1] == SaraAh) {   /* Check for char follow Orang */
                  sara_name = SARAUOH;    /* เ-อะ */
                  *last = i+1;
               }
               else if (!iskaran(str[i+1]) && !istonal(str[i+1]) \
                  && !isvowel1(str[i+1])) {
                  sara_name = SARAUOE;    /* เ-อ  */
                  *last = i;
```

```
    }
    else
        sara_name = FALSE;
    break;
case SaraIe :       /* 'เ' */
    sara_name = SARAUO;     /* เ */
    *last = i;
    break;
case SaraE :        /* 'แ' */
    j = (istonal(str[i+1]))? i+2: i+1;  /* Position of YoreYak */
    if (str[j] == YoreYak)  {
        if (str[j+1] == SaraAh) {   /* Check char follow YoreYak */
            sara_name = SARAIAH;    /* เ็ย */
            *last = j+1;
        }
        else if (!iskaran(str[j+1]) && !istonal(str[j+1]) && \
            !isvowel1(str[j+1]))    {
            sara_name = SARAIAE;    /* เ็ */
            *last = j;
        }
        else
            sara_name = FALSE;
    }
    else
        sara_name = FALSE;
    break;
case SaraUee :      /* 'เ' */
    j = (istonal(str[i+1]))? i+2: i+1;  /* Position of OrAng */
    if (str[j] == OrAng)  {
        if (str[j+1] == SaraAh) {   /* Check char follow OrAng */
            sara_name = SARAUEH;    /* เ็ว */
            *last = j+1;
        }
        else if (!iskaran(str[j+1]) && !istonal(str[j+1]) && \
            !isvowel1(str[j+1]))    {
```

```
                    sara_name = SARAUER;    /* เ◌ื */
                    *last = j;
                }
                else
                    sara_name = FALSE;
            }
            else
                sara_name = FALSE;
            break;
        default :
            sara_name = FALSE;
    }
    break;
case SaraAir :
    sara_name = (str[i] == SaraAh)? SARAAAH : FALSE;    /* แ-ะ */
    *last = i;      /* if FALSE , not use *last */
    break;
case Sara0 :
    sara_name = (str[i] == SaraAh)? SARAOOH : FALSE;    /* โ-ะ */
    *last = i;
    break;
case HunAkad :
    if (str[i+1] == SaraAh) {
        sara_name = SARAUVH;    /* ◌ัะ */
        *last = i+1;
    }
    else    {
        sara_name = SARAUV;     /* ◌ั */
        *last = i;
    }
    break;
default :
    sara_name = FALSE;
}
return sara_name;
```

```
}

/*-------------------------------------------------*

Expand part of vowel to it's braille code and output it.
They are SARAAEH, SARAAO, SARAAOH, SARAUOE, SARAUOH, SARAUO, SARAIAE,
  SARAIAH, SARAUER, SARAUEH, SARAAAH, SARAOOH ,SARAUV, SARAUVH
(ิ-ี, ิ-ำ, ิ-ำ่, ิ-ึ, ิ-ึ่, ิ-ื, ิ-ื่, ิ-ุ, ิ-ุ่, ิ-ู, ิ-ู่, เ-็, เ-่, ็-่, ็ำ, ็ำ่ )

*-------------------------------------------------*/
Byte expand_vowel(int vowel, int tonal_include, Byte tonal, struct tbraille *tb)
{
    Byte ch;

    if (tonal_include) {
        put_braille(vowel);
        ch = tonal;
    }
    else
        ch = vowel;

    return ch;
}

/*-------------------------------------------------*

Check for posible double leading consonant for part of vowels:
ิ-ี, ิ-ำ, ิ-ำ่, ิ-ึ, ิ-ึ่, ิ-ื, ิ-ื่, ิ-ุ, ิ-ุ่, ิ-ู, ิ-ู่, เ-็, เ-่

*-------------------------------------------------*/
posible_double_consan(Byte first, Byte second, Byte vowel)
{
    /*------ Table of posible double leading consonants for vowels ------*/
    struct double_consan    {
        Byte    second;     /* Char of second lead consonant */
```

```
    Byte    *first;     /* String of group of first lead consonant */
}conson_tab[][10] = {   /* Maximum number of second lead = 10 */
    {{'ร', "ป"}, {'ล', "ผ"}     },
    {{'ร', "ปคพศ"}, {'ง', "ห"}, {'ม', "หงฆ"}, {'ย', "หช"}, {'ว', "ห"},
        {'ล', "กหปคพผ"}},
    {{'ร', "กปคพ"}, {'ม', "ห"}, {'ย', "ห"}, {'ล', "หปพผ"}, {'พ', "ฉ"}},
    {{'น', "ส"}, {'ร', "กหป"}, {'ม', "หส"}, {'ย', "ส"}, {'ล', "กหผ"}},
    {{'ร', "กป"}},
    {{'ร', "กสจพ"}, {'อ', "ผ"}, {'ย', "หผ"}, {'ล', "ดหคพศ"}, {'ค', "ผ"},
        {'ฟ', "ผ"}},
    {{'น', "หสพ"}, {'ร', "กดหคพป"}, {'ม', "หส"}, {'ย', "ห"}, {'ว', "กหผ"},
        {'ล', "กหปพฉ"}, {'บ', "ส"}, {'ย', "ก"}},
    {{'ร', "ป"}, {'ล', "ผ"}},
    {{'น', "ห"}, {'ร', "ปคพ"}, {'ง', "ห"}, {'ม', "หส"}, {'ย', "ห"},
        {'ล', "กฮมหปคถ"}},
    {{'ร', "ป"}},
    {{'ร', "ค"}, {'ม', "ห"}, {'ย', "หส"}, {'ว', "หช"}, {'ล', "หผพ"}},
    {{'น', "ห"}, {'ร', "ป"}, {'ม', "ห"}, {'ย', "ห"}, {'ว', "ห"},
        {'ล', "กหผพ"}}
};
int i, double_lead_conson;

vowel -= SARAAEH;

for (i=0; i<12; i++)    {       /* find second lead consonant in table */
    if (second == conson_tab[vowel][i].second)
        break;
}

if (i < 12) {       /* find first lead consonant in string in table */
    if ((strchr(conson_tab[vowel][i].first, first)) != NULL)
        double_lead_conson = TRUE;
    else
        double_lead_conson = FALSE;
}
```

```
    else
        double_lead_conson = FALSE;
    return double_lead_conson;
}

/*-------------------------------------------------*

    Initila system variable

*-------------------------------------------------*/
void initsysvar(struct tbraille *tb)
{
    tb->capital         = FALSE;
    tb->singleQuotOpen  = FALSE;
    tb->doubleQuotOpen  = FALSE;
    tb->numberchar      = FALSE;
    tb->charcode        = TIS;
    tb->maxlcol         = MAXHCOL;
    tb->bcount          = 0;
}

/*-------------------------------------------------*

    Put braille charactor from ASCII charactor

*-------------------------------------------------*/
void _put_braille(Byte ch, struct tbraille *tb)
{
    /*------ ASCII Thai/English to Braille Translation table ------*/
    static Byte table1[] = {
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x09, 0x0a, 0x00, 0x0c, 0x0d, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x20, 0x36, 0x38, 0x23, 0x00, 0x00, 0x00, 0x27, 0x37, 0x37, 0x00, 0x00, 0x31, 0x2d, 0x34, 0x00,
    0x6a, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69, 0x33, 0x32, 0x00, 0x00, 0x00, 0x38,
    0x00, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4a, 0x4b, 0x4c, 0x4d, 0x4e, 0x4f,
```

```
0x00, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58, 0x59, 0x5a, 0x00, 0x00, 0x00, 0x00, 0x80,
0x00, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69, 0x6a, 0x6b, 0x6c, 0x6d, 0x6e, 0x6f,
0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78, 0x79, 0x7a, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x81, 0x36, 0x82, 0x25, 0x83, 0x25, 0x28, 0x84, 0x71, 0x85, 0x86, 0x87, 0x65, 0x88, 0x2c, 0x2e,
0x20, 0x67, 0x6b, 0x89, 0x75, 0x8a, 0x8b, 0x5d, 0x6a, 0x2f, 0x2b, 0x21, 0x8c, 0x8d, 0x8e, 0x8f,
0x90, 0x91, 0x92, 0x93, 0x64, 0x5c, 0x74, 0x29, 0x94, 0x6e, 0x76, 0x26, 0x70, 0x78, 0x3f, 0x24,
0x95, 0x6d, 0x79, 0x72, 0x96, 0x6c, 0x97, 0x77, 0x98, 0x99, 0x73, 0x68, 0x9a, 0x6f, 0x3d, 0x00,
0x61, 0x3e, 0x2a, 0x7a, 0x62, 0x32, 0x5b, 0x35, 0x63, 0x33, 0x34, 0x00, 0x00, 0x00, 0x00, 0x00,
0x66, 0x3c, 0x69, 0x9b, 0x3a, 0x2a, 0x31, 0x27, 0x39, 0x34, 0x37, 0x38, 0x30, 0x22, 0x00, 0x00,
0x6a, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

/*---------- Braille extend charactors table ----------*/
static Byte table2[][2] = {
{0x2c, 0x2c}, {0x66, 0x61}, {0x6f, 0x61}, {0x25, 0x61}, {0x28, 0x61},
{0x71, 0x61}, {0x3c, 0x61}, {0x69, 0x61}, {0x65, 0x61}, {0x39, 0x6b},
{0x2d, 0x75}, {0x2c, 0x75}, {0x2c, 0x2b}, {0x2c, 0x79}, {0x2c, 0x64},
{0x2c, 0x5c}, {0x2c, 0x74}, {0x2c, 0x29}, {0x2d, 0x29}, {0x2c, 0x6e},
{0x30, 0x29}, {0x2c, 0x63}, {0x72, 0x31}, {0x6c, 0x31}, {0x2c, 0x73},
{0x2d, 0x73}, {0x2c, 0x6c}, {0x3a, 0x31}
};

Byte brl;

brl = table1[ch];
if (brl)    {                   /* ignore code 0x00 */
    if (brl < 0x80)
        put_b(brl, tb);
    else    {                   /* double code */
        put_b(table2[brl-0x80][0], tb);
        put_b(table2[brl-0x80][1], tb);
    }
}
}
```

```
void put_b(Byte ch, struct tbraille *tb)
{
    if (!iscntrl(ch))                   /* not count control code */
        tb->bcount++;                   /* braille column count */
    if (!tb->bcol_error && tb->bcount > tb->maxbcol)    {   /* over maximum column */
        if (tb->editing)
            puts_editing("||  ** ERROR\n", 12, tb);
        putc('4\n', tb->embfile);       /* cariate return & newline */
        tb->bcol_error = TRUE;
    }
    putc(ch ,tb->embfile);
}

/*-------------------------------------------------------*

    Put charactor to make editing file.

*-------------------------------------------------------*/
void _put_editing(Byte ch, struct tbraille *tb)
{
    if (tb->charcode == KU && ch > 0x7f)
        ch = tis_to_ku[ch-0x80];        /* convert back to KU */
    putc(ch,tb->prnfile);
}

void puts_editing(Byte *start, int len, struct tbraille *tb)
{
    int i;

    for (i=0; i<=len; i++)
        put_editing(start[i]);
}

/*-------------------------------------------------------*
```

```
   Convert Thai code from KU to TIS

*-------------------------------------------------------*/
void ku2tis(Byte *str)
{
    Byte *ptr;

    ptr = str;
    while (*ptr)    {
        if (*ptr > 0x7f)
                                *ptr = ku_to_tis[*ptr-0x80];

        ptr++;
    }
}

/*-------------------------------------------------------*

   Convert Thai code from TIS to KU

*-------------------------------------------------------*/
void tis2ku(Byte *str)
{
    Byte *ptr;

    ptr = str;
    while (*ptr)    {
        if (*ptr > 0x7f)
                                *ptr = tis_to_ku[*ptr-0x80];

        ptr++;
    }
}
```

```
/*****************************************************************
  thai.h — thai characters
*****************************************************************/

#define KoreGai         (unsigned char) 0xA1   /* ก */
#define KorKai          (unsigned char) 0xA2   /* ข */
#define KorKuad         (unsigned char) 0xA3   /*   */
#define KoreKwai        (unsigned char) 0xA4   /* ค */
#define KoreKon         (unsigned char) 0xA5   /*   */
#define KoreRakung      (unsigned char) 0xA6   /* ฆ */
#define NgorNgoo        (unsigned char) 0xA7   /* ง */
#define JoreJarn        (unsigned char) 0xA8   /* จ */
#define ChorChing       (unsigned char) 0xA9   /* ฉ */
#define ShoreChang      (unsigned char) 0xAA   /* ช */
#define SoreSoe         (unsigned char) 0xAB   /* ซ */
#define ShoreKracher    (unsigned char) 0xAC   /* ฌ */
#define YoreYing        (unsigned char) 0xAD   /* ญ */
#define DoreChada       (unsigned char) 0xAE   /* ฎ */
#define TorePatak       (unsigned char) 0xAF   /* ฏ */
#define ThorSantan      (unsigned char) 0xB0   /* ฐ */
#define ToreMontoe      (unsigned char) 0xB1   /* ฑ */
#define TorePootao      (unsigned char) 0xB2   /* ฒ */
#define NoreNane        (unsigned char) 0xB3   /* ณ */
#define DoreDek         (unsigned char) 0xB4   /* ด */
#define ToreTao         (unsigned char) 0xB5   /* ต */
#define ThorToong       (unsigned char) 0xB6   /* ถ */
#define ToreTaharn      (unsigned char) 0xB7   /* ท */
#define ToreTong        (unsigned char) 0xB8   /* ธ */
#define NoreNoo         (unsigned char) 0xB9   /* น */
#define BoreBaimai      (unsigned char) 0xBA   /* บ */
#define PorePla         (unsigned char) 0xBB   /* ป */
#define PorPeng         (unsigned char) 0xBC   /* ผ */
#define ForFa           (unsigned char) 0xBD   /* ฝ */
#define PoreParn        (unsigned char) 0xBE   /* พ */
#define ForeFun         (unsigned char) 0xBF   /* ฟ */
#define PoreSumpao      (unsigned char) 0xC0   /* ภ */
#define MoreMar         (unsigned char) 0xC1   /* ม */
#define YoreYak         (unsigned char) 0xC2   /* ย */
```

```
#define RoreReo         (unsigned char) 0xC3   /* ร */
#define RoreRu          (unsigned char) 0xC4   /* ฤ */
#define LoreLing        (unsigned char) 0xC5   /* ล */
#define WoreWaan        (unsigned char) 0xC7   /* ว */
#define SoreSala        (unsigned char) 0xC8   /* ศ */
#define SoreRusi        (unsigned char) 0xC9   /* ษ */
#define SoreSeo         (unsigned char) 0xCA   /* ส */
#define HorHeeb         (unsigned char) 0xCB   /* ห */
#define LorChula        (unsigned char) 0xCC   /* ฬ */
#define OrAng           (unsigned char) 0xCD   /* อ */
#define HorNokHook      (unsigned char) 0xCE   /* ฮ */

#define PaiYarnnoy      (unsigned char) 0xCF   /* ฯ */
#define SaraAh          (unsigned char) 0xD0   /* สระอะ */
#define HunAkad         (unsigned char) 0xD1   /* ไม้หันอากาศ */
#define SaraR           (unsigned char) 0xD2   /* สระอา */
#define SaraUm          (unsigned char) 0xD3   /* สระอำ */
#define SaraIe          (unsigned char) 0xD4   /* สระอิ */
#define SaraE           (unsigned char) 0xD5   /* สระอี */
#define SaraUe          (unsigned char) 0xD6   /* สระอึ */
#define SaraUee         (unsigned char) 0xD7   /* สระอือ */
#define SaraU           (unsigned char) 0xD8   /* สระอุ */
#define SaraUU          (unsigned char) 0xD9   /* สระอู */
#define ThaiDot         (unsigned char) 0xDA   /* . */
#define SaraA           (unsigned char) 0xE0   /* สระเอ */
#define SaraAir         (unsigned char) 0xE1   /* สระแอ */
#define SaraO           (unsigned char) 0xE2   /* สระโอ */
#define MaiMuan         (unsigned char) 0xE3   /* ไม้ม้วน */
#define MaiMalai        (unsigned char) 0xE4   /* ไม้มลาย */
#define MaiYamok        (unsigned char) 0xE6   /* ไม้ยมก ๆ */
#define MaiTaiKuu       (unsigned char) 0xE7   /* ไม้ไต่คู้ ก็ */
#define MaiEk           (unsigned char) 0xE8   /* ไม้เอก */
#define MaiToe          (unsigned char) 0xE9   /* ไม้โท */
#define MaiTri          (unsigned char) 0xEA   /* ไม้ตรี */
#define MaiJattawa      (unsigned char) 0xEB   /* ไม้จัตวา */
#define Karan           (unsigned char) 0xEC   /* การันต์ */

#define _Exclamation    (unsigned char) 0x21   /* ! */
```

```
#define _Dollar         (unsigned char) 0x24    /* $ */
#define _Percent        (unsigned char) 0x25    /* % */
#define _OpenParen      (unsigned char) 0x28    /* วงเล็บเปิด*/
#define _CloseParen     (unsigned char) 0x29    /* วงเล็บปิด */
#define _Hyphen         (unsignec char) 0x2D    /* - */
#define _FullStop       (unsigned char) 0x2E    /* . */

#define _Colon          (unsigned char) 0x3A
#define _SemiColon      (unsigned char) 0x3B
#define _Question       (unsigned char) 0x3F    /* ? */
#define _LeftBracket    (unsigned char) 0x5B    /* [ */
#define _RightBracket   (unsigned char) 0x5D    /* ] */
#define _LeftBrace      (unsigned char) 0x7B    /* { */
#define _RightBrace     (unsigned char) 0x7D    /* } */
#define _Baht           (unsigned char) 0xDF    /* ฿ */
```

```
/*************************************************************
 TBRL.H  header file for TBRL.C
 *************************************************************/

#ifndef TRUE
#define TRUE        1
#define FALSE       0
#endif
#define ERROR       -1

#define MAXCOL      266         /* Maximum length of input buffer string */
#define MAXBCOL     40          /* default maximum length of braille outupt */
#define TIS         0           /* Thai charactor code */
#define KU          1

#define put_braille(x)      _put_braille(x, tb);    /* write file macro */
#define put_editing(x)      _put_editing(x, tb);
#define isconson(x)         (x >= KoreGai   && x <= HorNokHook)
#define istonal(x)          (x >= MaiEk     && x <= MaiJattawa)
#define isvowel1(x)         (x >= SaraAh    && x <= SaraUU)
#define isvowel2(x)         (x >= SaraA     && x <= MaiMalai)
#define isvowel3(x)         (x >= MaiYamok  && x <= MaiTaiKuu)
#define iskaran(x)          (x == Karan)
#define istdigit(x)         (x >= 0xf0      && x <= 0xf9)

#define CAPITALSIGN 0x9e    /* Spacial braille charactor */
#define APOSTROPHE  0x27
#define OPENQUOT    0xeb
#define CLOSEQUOT   0xec
#define OPENSINGLE  0x9e
#define CLOSESINGLE 0x27
#define NUMBERSIGN  0x23
#define MARKSIGN    0x9f    /* Mark word */
#define PINTOO      0xda    /* Pali-Sansadrit accent */

#define SARAAEH     0x90        /* เ-ะ (Part of vowel) */
#define SARAAO      0x91        /* เ-า */
```

```
#define SARAAOH     0x92        /* เ-าะ */
#define SARAUOE     0x93        /* เ-ือ  */
#define SARAUOH     0x94        /* เ-ือะ */
#define SARAUO      0x95        /* ิ    */
#define SARAIAE     0x96        /* ีย   */
#define SARAIAH     0x97        /* ียะ */
#define SARAUER     0x98        /* ือ   */
#define SARAUEH     0x99        /* ือะ */
#define SARAAAH     0x9a        /* ุ-ะ  */
#define SARAOOH     0x9b        /* ู-ะ  */
#define SARAUV      0x9c        /* ั ว   */
#define SARAUVH     0x9d        /* ั วะ  */

#define UPPERCASE   0           /* Charactor type case for chartype()  */
#define CONTROLCHAR 1
#define SINGLEQUOT  2
#define DOUBLEQUOT  3
#define NUMBERCHAR  4
#define THAINUMBER  5
#define SARAA       6           /* 'เ' (Thai vowel) */
#define SARAAIR     7           /* 'ุ' */
#define SARAO       8           /* 'ู' */
#define HANAKAD     9           /* ์  */

typedef unsigned char Byte;

struct tbraille {               /* System variable structure          */
    Byte    workline[MAXCOL];   /* Working line input buffer          */
    Byte    *workptr;           /* Pointer to current char in workline */
    FILE    *embfile;           /* Output file for embrosser          */
    FILE    *prnfile;           /* Output file for editing            */
    int     capital;            /* Capital char status                */
    int     singleQuotOpen;     /* Single Quot Open status            */
    int     doubleQuotOpen;     /* Double Quot Open status            */
    int     numberchar;         /* Number char status                 */
    int     charcode;           /* Thai charactor code                */
    int     editing;            /* Create editing file status         */
```

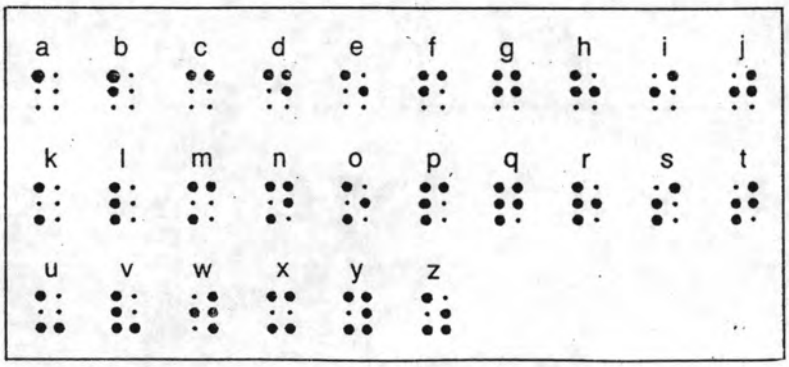
```
    int     bcount;             /* Braille column count                 */
    int     maxbcol;            /* Maximum braille column for print     */
    int     bcol_error;         /* Broille column out of maximum        */
};

/*------- Prototype ----------*/
void do_uppercase(Byte *strptr, struct tbraille *tb);
void do_singlequot(Byte *strptr, struct tbraille *tb);
void do_doublequot(Byte *strptr, struct tbraille *tb);
void do_numberchar(struct tbraille *tb);
void do_sara(Byte *strptr, struct tbraille *tb);
void do_hanakad(Byte *strptr, struct tbraille *tb);
void do_saraum(Byte *strptr, struct tbraille *tb);
Byte whatvowel(Byte *str, int i, int *last);
Byte expand_vowel(int vowel, int tonal_include, Byte tonal, struct tbraille *tb);
void initsysvar(struct tbraille *tb);
void _put_braille(Byte ch, struct tbraille *tb);
void put_b(Byte ch, struct tbraille *tb);
void _put_editing(Byte ch, struct tbraille *tb);
void puts_editing(Byte *start, int len, struct tbraille *tb);
void ku2tis(Byte *str);
void tis2ku(Byte *str);
```

ภาคผนวก ฉ

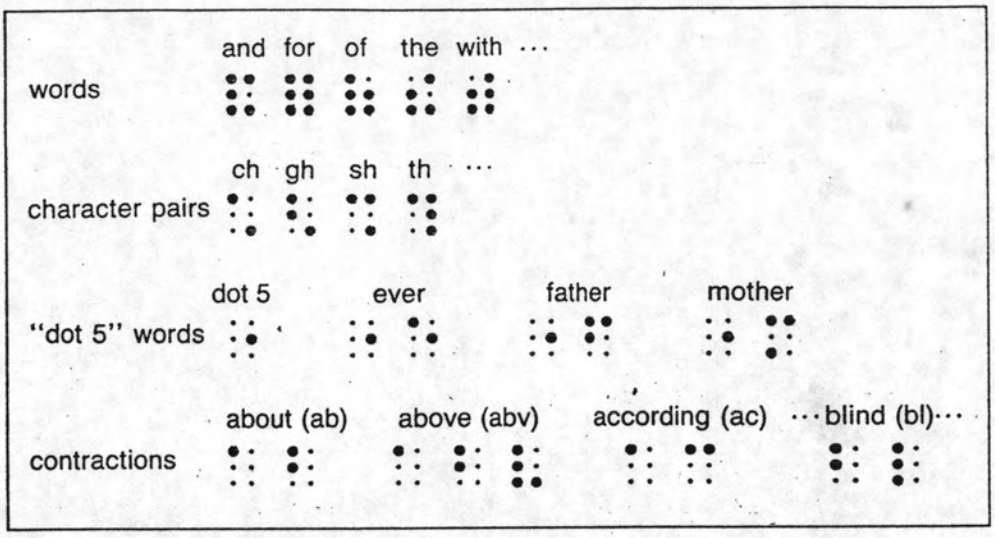
การนำอักษรเบรลล์ประยุกต์ใช้ในด้านการบีบอัดข้อมูล

เทคนิคการบีบอัดข้อมูลอาจแบ่งได้เป็นสองพวกคือ พวกที่อาศัยข้อมูลทางสถิติและดูความน่าจะเป็นของตัวอักษรต่าง ๆ (statistical) อีกพวกหนึ่งอาศัยหลักการของพจนานุกรม (Dictionary) โดยที่กลุ่มของตัวอักษรที่เรียงต่อกันตามพจนานุกรม อาจเกิดซ้ำ ๆ ได้บ่อยภายในเอกสารชุดหนึ่ง ๆ ดังนั้นวิธีพจนานุกรม จะเป็นการนำกลุ่มคำมาเข้ารหัส สร้างเป็นดัชนีของการชี้ค้นหาคำจริงในพจนานุกรม ซึ่งวิธีนี้จะช่วนลดจำนวนตัวอักษรและประหยัดพื้นที่ได้มาก

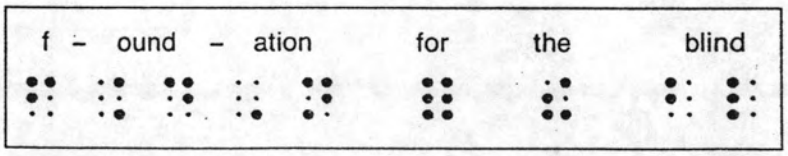
ตัวอย่างการบีบอัดข้อมูลที่จัดเป็นแบบวิธีพจนานุกรม โดยใช้รหัสตัวอักษรของระบบเบรลล์ ที่ใช้กับคนตาบอดในรูปที่ ก. มีการเข้ารหัสเป็นคำและกลุ่มตัวอักษรในรูปที่ ข. ดังนั้นข้อความ foundation for the blind สามารถจัดเป็นรหัสดังรูปที่ ค. ได้ ซึ่งช่วยลดจำนวนชุดของจุดที่ใช้แทนคำและตัวอักษรให้น้อยลง



รูปที่ ก. รหัสตัวอักษรเบรลล์



รูปที่ ข. การเข้ารหัสคำและกลุ่มตัวอักษร



รูปที่ ค. ตัวอย่างการบีบอัดข้อมูล

ประวัติผู้เขียน

นาย สุชุม มหิทธิหาญ เกิดเมื่อวันที่ 10 สิงหาคม พ.ศ. 2508 ที่เขตพระโขนง กรุงเทพมหานคร เป็นบุตรคนที่ 3 จากพี่น้องทั้งหมด 5 คน

การศึกษา

พ.ศ. 2532 เข้าศึกษาต่อระดับปริญญาโท สาขาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย

พ.ศ. 2531 สำเร็จการศึกษาอุตสาหกรรมศาสตร์บัณฑิต จาก ภาควิชา เทคโนโลยีไฟฟ้าอุตสาหกรรม คณะวิศวกรรมศาสตร์ สถาบัน เทคโนโลยีพระจอม เกล้าพระนคร เหนือ

พ.ศ. 2529 สำเร็จการศึกษาระดับประกาศนียบัตรวิชาชีพชั้นสูง จากภาควิชาไฟฟ้า วิทยาลัย เทคโนโลยีอุตสาหกรรม สถาบัน เทคโนโลยีพระจอม เกล้าพระนคร เหนือ

พ.ศ. 2527 สำเร็จการศึกษาระดับประกาศนียบัตรวิชาชีพ จากภาควิชาไฟฟ้า วิทยาลัย เทคโนโลยีอุตสาหกรรม สถาบัน เทคโนโลยีพระจอม เกล้าพระนคร เหนือ