Appendix B: Vietnamese Characters under VISCII and VIQR by Encoding Order (continued)

VISCII	Char	VIQR	Descriptive Name
164	ấ	a^´	a circumflex acute
165	ầ	a^`	a circumflex grave
166	ẩ	a^?	a circumflex hook-above
167	ậ	a^.	a circumflex dot-below
168	ẽ	e~	e tilde
169	ẹ	e.	e dot-below
170	ế	e^´	e circumflex acute
171	ề	e^`	e circumflex grave
172	ể	e^?	e circumflex hook-above
173	ễ	e^~	e circumflex tilde
174	ệ	e^.	e circumflex dot-below
175	ố	o^´	o circumflex acute
176	ồ	o^`	o circumflex grave
177	ổ	o^?	o circumflex hook-above
178	ỗ	o^~	o circumflex tilde
179	Ỡ	O+~	O horn tilde
180	Ớ	O+	O horn
181	ộ	o^.	o circumflex dot-below
182	ờ	o+`	o horn grave
183	ở	o+?	o horn hook-above
184	ị	i.	i dot-below
185	Ự	U+.	U horn dot-below
186	Ứ	U+´	U horn acute
187	Ừ	U+`	U horn grave
188	Ử	U+?	U horn hook-above
189	ơ	o+	o horn
190	ớ	o+´	o horn acute
191	Ư	U+	U horn
192	À	A`	A grave
193	Á	A´	A acute
194	Â	A^	A circumflex
195	Ã	A~	A tilde
196	Ả	A?	A hook-above
197	Ă	A(	A breve
198	ả	a(?	a breve hook-above
199	ã	a(~	a breve tilde
200	È	E`	E grave
201	É	E´	E acute
202	Ê	E^	E circumflex
203	Ẻ	E?	E hook-above
204	Ì	I`	I grave
205	Í	I´	I acute
206	Ĩ	I~	I tilde
207	ỳ	y`	y grave
208	Đ	DD†	D bar
209	ứ	u+´	u horn acute

VISCII	Char	VIQR	Descriptive Name
210	Ò	O`	O grave
211	Ó	O´	O acute
212	Ô	O^	O circumflex
213	ạ	a.	a dot-below
214	ỷ	y?	y hook-above
215	ừ	u+`	u horn grave
216	ử	u+?	u horn hook-above
217	Ù	U`	U grave
218	Ú	U´	U acute
219	ỹ	y~	y tilde
220	ỵ	y.	y dot-below
221	Ý	Y´	Y acute
222	ỡ	o+~	o horn tilde
223	ư	u+	u horn
224	à	a`	a grave
225	á	a´	a acute
226	â	a^	a circumflex
227	ã	a~	a tilde
228	ả	a?	a hook-above
229	ă	a(	a breve
230	ữ	u+~	u horn tilde
231	ẫ	a^~	a circumflex tilde
232	è	e`	e grave
233	é	e´	e acute
234	ê	e^	e circumflex
235	ẻ	e?	e hook-above
236	ì	i`	i grave
237	í	i´	i acute
238	ĩ	i~	i tilde
239	ỉ	i?	i hook-above
240	đ	dd	d bar
241	ự	u+.	u horn dot-below
242	ò	o`	o grave
243	ó	o´	o acute
244	ô	o^	o circumflex
245	õ	o~	o tilde
246	ỏ	o?	o hook-above
247	ọ	o.	o dot-below
248	ụ	u.	u dot-below
249	ù	u`	u grave
250	ú	u´	u acute
251	ũ	u~	u tilde
252	ủ	u?	u hook-above
253	ý	y´	y acute
254	ợ	o+.	o horn dot-below
255	Ữ	U+~	U horn tilde

† VIQR also allows "Đ" to be represented by "Dd" or "dD". See Sec. 4.2.1.

Appendix B: Vietnamese Characters under VISCII and VIQR by Encoding Order

VISCII	Char	VIQR	Descriptive Name
002	Ả	A(	A breve hook-above
005	Ã	A(	A breve tilde
006	Ẫ	A^~	A circumflex tilde
020	Ỷ	Y?	Y hook-above
025	Ỹ	Y~	Y tilde
030	Ỵ	Y.	Y dot-below
065	A	A	A
066	B	B	B
067	C	C	C
068	D	D	D
069	E	E	E
070	F	F	F
071	G	G	G
072	H	H	H
073	I	I	I
074	J	J	J
075	K	K	K
076	L	L	L
077	M	M	M
078	N	N	N
079	O	O	O
080	P	P	P
081	Q	Q	Q
082	R	R	R
083	S	S	S
084	T	T	T
085	U	U	U
086	V	V	V
087	W	W	W
088	X	X	X
089	Y	Y	Y
090	Z	Z	Z
097	a	a	a
098	b	b	b
099	c	c	c
100	d	d	d
101	e	e	e
102	f	f	f
103	g	g	g
104	h	h	h
105	i	i	i
106	j	j	j
107	k	k	k
108	l	l	l
109	m	m	m
110	n	n	n
111	o	o	o

VISCII	Char	VIQR	Descriptive Name
112	p	p	p
113	q	q	q
114	r	r	r
115	s	s	s
116	t	t	t
117	u	u	u
118	v	v	v
119	w	w	w
120	x	x	x
121	y	y	y
122	z	z	z
128	Ạ	A.	A dot-below
129	Ắ	A(	A breve acute
130	Ằ	A(	A breve grave
131	Ặ	A(	A breve dot-below
132	Ấ	A^	A circumflex acute
133	Ầ	A^	A circumflex grave
134	Ẩ	A^?	A circumflex hook-above
135	Ậ	A^.	A circumflex dot-below
136	Ẽ	E~	E tilde
137	Ẹ	E.	E dot-below
138	Ế	E^	E circumflex acute
139	Ề	E^	E circumflex grave
140	Ể	E^?	E circumflex hook-above
141	Ễ	E^~	E circumflex tilde
142	Ệ	E^.	E circumflex dot-below
143	Ố	O^	O circumflex acute
144	Ồ	O^	O circumflex grave
145	Ổ	O^?	O circumflex hook-above
146	Ỗ	O^~	O circumflex tilde
147	Ộ	O^.	O circumflex dot-below
148	Ợ	O+.	O horn dot-below
149	Ớ	O+	O horn acute
150	Ờ	O+	O horn grave
151	Ở	O+?	O horn hook-above
152	Ị	I.	I dot-below
153	Ỏ	O?	O hook-above
154	Ọ	O.	O dot-below
155	Ỉ	I?	I hook-above
156	Ủ	U?	U hook-above
157	Ũ	U~	U tilde
158	Ụ	U.	U dot-below
159	Ỳ	Y	Y grave
160	Õ	O~	O tilde
161	ắ	a(	a breve acute
162	ằ	a(	a breve grave
163	ặ	a(	a breve dot-below

Appendix A: Vietnamese Characters under VISCII and VIQR by Collating Order

Char	VIQR	VISCII	Char	VIQR	VISCII	Char	VIQR	VISCII	Char	VIQR	VISCII
A	A	065	N	N	078	a	a	097	n	n	110
Á	A´	193	O	O	079	á	a´	225	o	o	111
À	A`	192	Ó	O´	211	à	a`	224	ó	o´	243
Ả	A?	196	Ò	O`	210	ả	a?	228	ò	o`	242
Ã	A~	195	Ỏ	O?	153	ã	a~	227	ỏ	o?	246
Ạ	A.	128	Õ	O~	160	ạ	a.	213	õ	o~	245
Ă	A(	197	Ọ	O.	154	ă	a(	229	ọ	o.	247
Ắ	A(´	129	Ô	O^	212	ắ	a(´	161	ô	o^	244
Ằ	A(`	130	Ố	O^´	143	ằ	a(`	162	ố	o^´	175
Ẳ	A(?	002	Ồ	O^`	144	ẳ	a(?	198	ồ	o^`	176
Ẵ	A(~	005	Ổ	O^?	145	ẵ	a(~	199	ổ	o^?	177
Ặ	A(.	131	Ỗ	O^~	146	ặ	a(.	163	ỗ	o^~	178
Â	A^	194	Ộ	O^.	147	â	a^	226	ộ	o^.	181
Ấ	A^´	132	Ơ	O+	180	ấ	a^´	164	ơ	o+	189
Ầ	A^`	133	Ớ	O+´	149	ầ	a^`	165	ớ	o+´	190
Ẩ	A^?	134	Ờ	O+`	150	ẩ	a^?	166	ờ	o+`	182
Ẫ	A^~	006	Ở	O+?	151	ẫ	a^~	231	ở	o+?	183
Ậ	A^.	135	Ỡ	O+~	179	ậ	a^.	167	ỡ	o+~	222
B	B	066	Ợ	O+.	148	b	b	098	ợ	o+.	254
C	C	067	P	P	080	c	c	099	p	p	112
D	D	068	Q	Q	081	d	d	100	q	q	113
Đ	DD†	208	R	R	082	đ	dd	240	r	r	114
E	E	069	S	S	083	e	e	101	s	s	115
É	E´	201	T	T	084	é	e´	233	t	t	116
È	E`	200	U	U	085	è	e`	232	u	u	117
Ẻ	E?	203	Ú	U´	218	ẻ	e?	235	ú	u´	250
Ẽ	E~	136	Ù	U`	217	ẽ	e~	168	ù	u`	249
Ẹ	E.	137	Ủ	U?	156	ẹ	e.	169	ủ	u?	252
Ê	E^	202	Ũ	U~	157	ê	e^	234	ũ	u~	251
Ế	E^´	138	Ụ	U.	158	ế	e^´	170	ụ	u.	248
Ề	E^`	139	Ư	U+	191	ề	e^`	171	ư	u+	223
Ể	E^?	140	Ứ	U+´	186	ể	e^?	172	ứ	u+´	209
Ễ	E^~	141	Ừ	U+`	187	ễ	e^~	173	ừ	u+`	215
Ệ	E^.	142	Ử	U+?	188	ệ	e^.	174	ử	u+?	216
F	F	070	Ữ	U+~	255	f	f	102	ữ	u+~	230
G	G	071	Ự	U+.	185	g	g	103	ự	u+.	241
H	H	072	V	V	086	h	h	104	v	v	118
I	I	073	W	W	087	i	i	105	w	w	119
Í	I´	205	X	X	088	í	i´	237	x	x	120
Ì	I`	204	Y	Y	089	ì	i`	236	y	y	121
Ỉ	I?	155	Ý	Y´	221	ỉ	i?	239	ý	y´	253
Ĩ	I~	206	Ỳ	Y`	159	ĩ	i~	238	ỳ	y`	207
Ị	I.	152	Ỷ	Y?	020	ị	i.	184	ỷ	y?	214
J	J	074	Ỹ	Y~	025	j	j	106	ỹ	y~	219
K	K	075	Ỵ	Y.	030	k	k	107	ỵ	y.	220
L	L	076	Z	Z	090	l	l	108	z	z	122
M	M	077				m	m	109			

† VIQR also allows "Đ" to be represented by "Dd" or "dD". See Sec. 4.2.1.

like subscribing to a collection of electronic magazines. These "magazines," called *newsgroups*, are devoted to particular topics. The "Soc.Culture.Vietnamese" newsgroup is very popular among both Vietnamese and non-Vietnamese worldwide.

Viet-Std: A non-profit group of overseas Vietnamese and other professionals working on software & hardware standards for the Vietnamese language. Members of the group exchange ideas via electronic mail and meetings.

Vowel: In this text, a generic term applying to all Vietnamese vowels and their various combining forms, e.g., a, ă, and á. See *Base Vowel*.

ASCII: American Standard Code for Information Interchange, a 128-character code used almost universally by computers for representing and transmitting characters data, in which each character corresponds to a decimal number between 0 and 127. Eight- or nine-bit codes of which the first 128 characters correspond to ASCII are called Extended ASCII; the additional characters are used to provide graphic characters for roman alphabets with diacritics, non-roman alphabets, special screen effects, etc.

Base Vowel: In this text, the unaccented Vietnamese vowels: a ă â e ê i o ô ơ u ư y (and their capitals). Contrast this with *Vowel*.

C0 Space: "Control characters" at code positions with hex values 00 through 1F.

C1 Space: "Control characters" at code positions with hex values 80 through 9F.

Code: In data communication, the numeric or internal representation for a character, e.g., in ASCII.

Code Page: Name used to denote glyph sets on the IBM PC. Abbreviated as CP. CP 850 is the multilingual code page, CP860 is for Portugal, CP863 is for French Canada, CP865 is for Norway.

Control Character: An ASCII character in the range 0 to 31, plus ASCII character 127, contrasted with the printable, or graphic, characters in the range 32 to 126. It is produced on an ASCII terminal by holding down the CTRL key and typing the desired character.

EBCDIC: Extended Binary Coded Decimal Interchange Code. The character code used on IBM mainframes. Not covered by any formal standards but described definitively in [15] and discussed at length in [16].

Floating Diacritics: A multiple-unit encoding approach for Vietnamese that treats the vowel and its diacritics as separate units. The diacritics may either precede or follow the vowel, or even the word. Contrast this with *Precomposed Character*.

Glyph: The physical appearance of a character as displayed on the screen or printed on paper.

G0 Space: "Graphic characters" at code positions with hex values 20 through 7F.

G1 Space: "Graphic characters" at code positions with hex values A0 through FF.

ISO: International Organization for Standardization. A voluntary international group of national standards organizations that issues standards in all areas, including computers, information processing, and character sets.

ISO 646: The standard 7-bit code set, equivalent to ASCII [12].

ISO Standard 8859: An ISO standard specifying a series of 8-bit computer character sets that include characters from many languages. These include ISO Latin Alphabets 1-9, which cover most of the written languages based on Roman letters, plus special character sets for Cyrillic, Greek, Arabic, and Hebrew [5].

ISO 8859/1: ISO Standard 8859 Latin Alphabet Number 1. Supports at least the following languages: Latin, Danish, Dutch, English, Faeroese, Finnish, French, German, Icelandic, Irish, Italian, Norwegian, Portuguese, Spanish, and Swedish [5].

ISO 2022 and ISO 4873: ISO standards for switching code pages [13].

ISO DIS 10646: The prospective 16- and 32-bit Universal Coded Set, (Draft International Standard) [4].

Latin: Referring to the Latin, or Roman, alphabet, comprised of the letters A through Z, or to any alphabet based upon it.

MS-DOS: Microsoft's Disk Operating System for microcomputers based on the Intel 80x86 family of CPU chips.

Modifier: A phonetic diacritical mark. The Vietnamese modifiers are: breve (trăng, ˘), circumflex (mũ, ˆ), horn (móc, ˒).

PC: Personal Computer. In this text, the term PC refers to the entire IBM PC and PS/2 families and compatibles, which includes the AT, 286, 386, and 486 PC's.

PostScript: A page description language with graphics capabilities designed for electronic printing. The description is high-level and device-independent. PostScript is a trademark of Adobe Systems Incorporated.

Precomposed Characters: An encoding approach for Vietnamese that treats all vowel combinations as single units. Contrast this with *Floating Diacritics*.

TEX: A computerized typesetting system developed by Donald Knuth [17], providing nearly everything needed for high-quality typesetting of mathematical notations as well as of ordinary text. TEX is a trademark of the American Mathematical Society.

Tone Mark: A tonal diacritical mark that indicates the tone/accent. The Vietnamese tone marks are: acute (sắc), grave (huyền), hook above (hỏi), tilde (ngã), dot below (nặng).

Unicode: A 16-bit multilingual character code proposed by the Unicode Consortium [3].

Unix: A popular operating system developed at AT&T Bell Laboratories and noted for its portability.

Usenet: A worldwide network available to users for sending messages (or "news articles") that can be read and responded to by other users. Participating in Usenet is

of CONTROL or FUNCTION keys. Any enhancement in compliant applications is a bonus for the user, so long as such enhancements do not adversely conflict with the minimum expected behavior described here.

5.3 ADAPTING EXISTING VIETNAMESE APPLICATIONS

A realistic approach to standardization provides for the inertia against change in existing software applications. While it is desirable that the standard 8-bit encoding described here be fully supported, an alternative exists which is more amenable to rapid adoption. All applications should provide a means for importing and exporting data encoded using the VISCII 8-bit encoding table. At the same time, the VIQR keyboard interface should be implemented, at least as an optional entry method. Such moves are highly desirable both for the user and the vendor alike. The user will be able to use the software immediately because of the uniform keyboard interface, as well as process the same data in different applications and on different platforms, with increased productivity and interactivity among users. This ease of use means greater acceptance and a correspondingly larger customer base for the vendor.

6 SUMMARY & CONCLUSIONS

This paper has presented a proposal for standardization of Vietnamese information processing. A case has been made for the necessity of standardization; we hope to have encouraged vendors and users of Vietnamese alike to work together toward this goal to benefit everyone involved. Various encoding approaches were discussed, leading to the choice of the VISCII 8-bit encoding proposal. A single encoding table was presented that has been shown in actual practice to work well for Vietnamese including editing, processing, storage, transfer, font encoding, and printing. Where 8-bit data handling was not available or reliable, e.g., electronic mail transport, the Vietnamese Quote-Readable specification (VIQR) was introduced to provide a seamless filtering gateway. VIQR was defined to be input-source-independent and hence has been designed to be applicable to Vietnamese keyboard input as well as machine data filters. All of this was shown to have been integrated into existing environments facilitating the use of existing tools and applications—a major strength of the encoding. Finally, these specifications have been linked together seamlessly to include every point in the input-process/transfer-output cycle of data handling and provide for a truly unified framework for Vietnamese information processing.

References

[1] Bạch Hưng Khang. "Institute of Informatics,". Hà Nội, Việt Nam, February 1991.

[2] B. Jerman-Blažič, "Will the Multi-octet Standard Character Set Code Solve the World Coding Problems for Information Interchange?," *Computer Standards & Interfaces*, vol. 8, pages 127–136, 1988.

[3] The Unicode Consortium. *The Unicode Standard: Worldwide Character Encoding Version 1.0.* Addison-Wesley, Reading, MA, first edition, October 1991.

[4] ISO Technical Committee, "Universal Multiple-Octet Coded Character Set (UCS), ISO/IEC DIS 10646-1.2," Draft standard, International Organization for Standardization, 1992.

[5] International Organization for Standardization. *ISO 8859/x: 8-bit International Code Sets.* ISO, 1977.

[6] Famjxuæn Thais. *Việt Ngữ Cải Cách.* Tứ Hải, Hà Nội, Việt Nam, March 1948.

[7] Phạm Xuân Thái. *Chữ Việt Hợp Lí.* Tín-Đức Thư-Xã, Sài Gòn, Việt Nam, April 1958.

[8] J. Postel, "Simple Mail Transfer Protocol," RFC 822, USC Information Sciences Institute, August 1982.

[9] J. C. Klensin et al., "SMTP Extensions for Transport of Text-Based Messages Containing 8-bit Characters," Internet draft, Massachusetts Institute of Technology, July 1991.

[10] K. Simonsen, "Character Mnemonics & Character Sets," Internet draft, Danish Unix Users Group, January 1992.

[11] K. Simonsen, "Mnemonic Text Format," Internet draft, Danish Unix Users Group, August 1991.

[12] International Organization for Standardization. *ISO 646: 7-bit Coded Character Set for Information Interchange.* ISO, third edition, 1991.

[13] International Organization for Standardization. *ISO 2022: 7-bit and 8-bit Coded Character Sets—Code Extension Techniques.* ISO, third edition, 1986.

[14] E. M. van der Poel, "Multilingual Character Encoding for Internet Messages," Internet draft, Software Research Associates, Japan, January 1992.

[15] IBM. *System/370 Reference Summary–GX20-1850-5*, sixth edition, 1984.

[16] C.E. Mackenzie. *Coded-Character Sets: History and Development.* Addison-Wesley, Reading, MA, 1980.

[17] D.E. Knuth. *The TEXbook.* Addison-Wesley, Reading, MA, 1984.

Glossary of Terms

Announcer: A character or sequence of characters appearing in the data that signifies the start of some special sequence. In this text, it announces a Vietnamese composition sequence.

data stream, such as a terminal application. It is useful because mail headers do not adhere to the VIQR, and they are more adversely affected when interpreted in non-Literal states.

5.2 VIETNAMESE KEYBOARDING

Keyboards are becoming increasingly internationalized. As mentioned in the 8-bit specification, this is the major reason for using the same code positions for those Vietnamese characters already present in ISO 8859/Latin-1. A Vietnamese keyboard driver designed to work in the 7-bit-only environment can assume that it will not encounter Vietnamese base vowels residing in G1. Keyboard drivers for the 8-bit environments, like 8-bit electronic mail agents (Section 5.1), must be prepared to accept any base vowel, including those encoded in G1.

The real-time echoing behavior of keyboard input during composition requires further specification. The options are to report the character only after the composition sequence has finished, or to report all intermediate forms and backspacing over them. Each has its own useful context as described below.

5.2.1 Immediate Echo for Implicit Composition

Implicit composition is designed to be convenient for a user processing data that is mostly Vietnamese. As such it is desirable for the keyboarding user to get immediate feedback on typed keys. With implicit composition, the keyboard works in *immediate-echo* mode. Keypresses immediately generate key events. If a character is subsequently composed with a diacritical mark, a *backspace* (typically `BS`, ASCII 0x08) is sent followed by the new composed character. This cycle continues as long as composition is possible. The sequence of events for the key sequence `"a^´n"` under immediate echo is:

1. user types `a`, `a` is sent to the application
2. user types `^`, `BS` and `â` are sent
3. user types `´`, `BS` and `ấ` are sent
4. user types `n`, the single key `n` is sent

The actual *backspace* character code may vary depending on the system, application, and user settings. The keyboard interface should use the appropriate code, and/or allow the user to specify the preferred *backspace* character.

5.2.2 Delayed Echo for Explicit Composition

When a composition sequence is started, the keyboard interface must not send any key events to the application expecting keyboard input until the sequence is terminated. Composition may end either naturally when the interface receives a character that cannot be composed into the sequence, or when the closure character `<CLS>` is received. A single key event for the composed character is then sent to the application above. Subsequent processing can proceed naturally. Consider what happens when the user types the sequence `"\a^´n"` under delayed echo:

1. user types \, no key is sent to the application
2. user types `a`, no key is sent
3. user types `^`, no key is sent
4. user types `´`, the single key `ấ` is sent
5. user types `n`, the single key `n` is sent

Or an example involving closure, `"t\o+<CLS>"`:

1. user types `t`, the key `t` is sent
2. user types \, no key is sent
3. user types `o`, no key is sent
4. user types `+`, no key is sent
5. user types `CTRL-A`, the single key `ơ` is sent

Note that without the closure key the keyboard interface would still be left hanging after the `"+"` key has been pressed, because the user can still enter a tone mark as part of the composition sequence.

This *delayed-echo* behavior for explicit composition is designed to ensure compatibility with applications expecting single key events for each character, particularly in the English state where only explicit composition is available.

While it is certainly possible to have immediate-echo in explicit composition or delayed-echo in implicit composition, these options are not useful and serve only to confuse the user learning how to use a Vietnamese keyboard. It is therefore simplest to associate delayed-echo with explicit composition, and immediate-echo with implicit composition. These options make natural sense.

This standard defines the minimal "look-and-feel" behavior a user can expect from a compliant Vietnamese software package. A standardized interface decreases the required learning time for each new application. This standard does not preclude other input mechanisms to improve user-friendliness, e.g., intelligent menu-driven diacritics, or to assist in speed typing, e.g., through the use

4.2.5 Vietnamese State

The data stream state is set to Vietnamese when the sequence <COM>V or <COM>v is encountered. In Vietnamese mode, both explicit and implicit compositions are in effect. The following examples assume that the data stream is initially in English state:

```
\vCh\u+~ Vi\e^.t   →   Chữ Việt
\vChu+~ Vie^.t     →   Chữ Việt
Chu+~ \vVie^.t     →   Chu+~ Việt
```

The availability of implicit composition in Vietnamese state ensures that the text is not cluttered with unnecessary <COM>s, as would be the case in Vietnamese text using explicit composition. Explicit composition is included to maintain compatibility with the English state so that there is no need to define additional meanings for the <COM> sequences. Also, the real-time keyboard compatibility mentioned previously is also available in Vietnamese state through explicit composition.

4.2.6 Character Literals in English and Vietnamese States

Consider the following example:

```
\vDu~ng, how are you?   →   Dũng, how are yoủ
```

In this example, the sequence "you?" was interpreted as "yoủ" because the data stream was still in Vietnamese state. Thus it is sometimes desirable to suppress composition altogether without having to switch states. The *literal* property of the <COM> character conveniently accomplishes this. In either Vietnamese or English state, whenever <COM> is followed by a non-combining character *c* the result is the literal character *c* itself. The <COM> is discarded from the data stream. To get the <COM> character literally, use <COM><COM>. Consider the following examples:

```
\vddi dda^u?     →   đi đâủ
\vddi dda^u\?    →   đi đâu?
\vddi v\o^?      →   đi vổ
\vddi v\o^\?     →   đi vô?
\\               →   \
\\V              →   \V
\\M              →   \M
\\L              →   \L
```

4.2.7 Closure

The data stream supports another special character used to generate *explicit closure*. The closure character is CTRL-A (ASCII 0x01), known here as <CLS>. When <CLS> is encountered in the data stream, it immediately terminates any ongoing composition sequence. The <CLS> itself is always discarded, unless it appears in the literal sequence <COM><CLS>.

Explicit closure is useful in real-time character applications such as keyboard entry, when it is necessary to specify that a composition sequence has in fact ended and the input engine should not stay hanging and wait for more data.

5 SPECIFIC APPLICATIONS

This section outlines application-specific guidelines and conventions that have evolved in the software development community. It is intended to be a live and growing documentation of such discussions as more experience is gathered. Readers are welcome to participate in these discussions and contribute to the development of these guidelines in particular, and to the standards in general.

5.1 ELECTRONIC MAIL OVER 7-BIT CHANNELS

Many of the available channels for electronic mail currently still enforce the 7-bit limitation. The 8-bit character set defined in Section 3 cannot be transported verbatim over these channels. VIQR plays an important role here, as it provides for 7-bit transport of Vietnamese text without the ambiguity problem of deciding what to do with the double usage of a diacritical/punctuation mark, e.g., the hook-above or question mark, "?". Because of the 7-bit nature of these communications channels, mail agents will typically not encounter those Vietnamese-specific base vowels that are encoded in the G1 area, namely: ă, Ă, â, Â, ê, Ê, ô, Ô, ơ, Ơ, ư, and Ư. However, mail agents designed to work with 8-bit channels are still expected to handle the occurrence of these characters according to the complete VIQR, namely to combine base vowels and diacritical marks as appropriate, for example:

```
ă´   →   ắ
```

In order to be correctly interpreted, electronic mail messages must explicitly set the language state either in the headers or text body. One cannot assume what state the receiving input engine is in at the start of the message, since messages are not always read in message units, e.g., when a file containing multiple mail messages is scanned.

Furthermore, if a language state specification (\L, \V or \M) is present in a mail message, it is highly recommended that the message end in the Literal state. This helps applications reading multiple mail messages in one

4.2.1 Implicit Composition

Implicit composition is useful for data containing a large percentage of Vietnamese characters.

With implicit composition, a sequence of a base vowel followed by one or two diacritical marks is combined into one Vietnamese letter as long as it is grammatically legal. This is best illustrated by examples:

```
a^      →  â
o+?     →  ở
ơ?      →  ở
Vie^.t  →  Việt
Viê.t   →  Việt
la´^n   →  lá^n (not lấn)
lá^n    →  lá^n (not lấn)
```

Note in the last two example that the sequence `"a^´"` is not grammatically equivalent to `"a´^"` or `"á^"`. In general a modifier (`"("`, `"^"`, `"+"`) must immediately follow the appropriate vowel in order to be combined.

The special sequence `"dd"` is composed into `"đ"`; `"DD"`, `"dD"`, and `"Dd"` all represent `"Đ"`.

The base vowels are: `a, ă, â, e, ê, i, o, ô, ơ, u, ư, y,` and their corresponding capitals. The encoding values are those listed in Table 3, the 8-bit VISCII proposed standard.

The diacritical marks are represented by ASCII characters having correspondingly similar appearances. Table 4 lists the 7 ASCII characters used as mnemonic replacements for the Vietnamese diacritics; the first three are modifiers, and the remaining five are tone marks.

Table 4: ASCII Mnemonics for Vietnamese Diacritics

Diacritic	Char	ASCII Code	Dấu
breve	(	0x28, left paren	trăng (˘)
circumflex	^	0x5E, caret	mũ (ˆ)
horn	+	0x2B, plus sign	móc (ʾ)
acute	´	0x27, apostrophe	sắc
grave	`	0x60, backquote	huyền
hook above	?	0x3F, question	hỏi
tilde	~	0x7E, tilde	ngã
dot below	.	0x2E, period	nặng

4.2.2 Explicit Composition

Explicit composition is associated with the concept of a leading character which explicitly announces the composition. The announcer character is the backslash (`"\"`, ASCII 0x5C), known here as `<COM>`. The subsequent combining characters are defined in the same way as those in implicit composition. Thus the examples given above would appear in explicit composition mode as:

```
\a^      →  â
\o+?     →  ở
Vi\e^.t  →  Việt
```

Explicit composition is useful for data containing mainly English text, as well as for maintaining real-time compatibility with keyboard character events, as will be discussed in Section 5.2 on Vietnamese keyboarding.

With the composition methods described, we are now ready to discuss how they are employed in each of the three states. The state of the data stream is specified by the two character sequence `<COM>x`, where `x` is specified below.

4.2.3 Literal State

The appearance of `<COM>L` or `<COM>l` in the data stream initiates the Literal state. This state is intended for near-perfect transparent literal data transfer. Neither implicit nor explicit composition is available here, nor is the `<COM>` character special, except when it is followed by one of the six characters `l, L, v, V, m` or `M` which initiates one of the three states.[9]

4.2.4 English State

The sequence `<COM>M` or `<COM>m` sets the data stream state to English. In English state, only explicit composition is supported. This means that in order to generate a Vietnamese letter, the announcer character `<COM>` must be used. A "composition" sequence not preceded by `<COM>` will be left uninterpreted. Examples:

```
\mD\u~ng, how are you?  →  Dũng, how are you?
\mKho\e?  kh\o^ng?      →  Khoẻ không?
```

As noted, the sequence `"you?"` above was not converted into `"youỏ"` because no composition was specified.

[9] To effect `<COM>L`, `<COM>M`, and `<COM>V` themselves, it is necessary to switch to either English or Vietnamese state and use the Character Literal feature available there.

use of mnemonics to the 83 invariant ISO-646 [12] graphic characters, which is a good idea in principle, but sacrifices readability in the process. For example, the counter-intuitive mnemonics for hook-above (dấu hỏi) and tilde (dấu ngã) are "2" and "?", respectively, in order to avoid "~" itself, which is not an invariant. The wide availability of ASCII keyboards to the great majority of Vietnamese users makes this too unreasonable a limitation in the context of Vietnamese processing. It should be noted that we are in fact arguing in favor of "readability for most" against "illegibility for all." Furthermore, with ongoing progress on keyboard and display internationalization, e.g., in graphical window environments where keyboard mapping and font switching are easily implemented, this availability is on the increase, further obsoleting the restriction.

The greater difficulty is that the two-character fixed-length encoding[7] cannot provide a readable or mnemonic representation of all Vietnamese characters, in particular those with 2 diacritical marks. The variable-length mnemonics[8] have been extended to include all Vietnamese characters, but this scheme is so cluttered with announcers and delimiters that readability and efficiency are near nil, keeping in mind that diacritics are heavily used in Vietnamese. While machine data translators will have little trouble with any "mnemonic" scheme, one that is directly accessible to human users, who are in many cases typing mail messages using 7-bit editors, needs to be more user-friendly. A Vietnamese user will not want to learn or remember among all possible combinations that, say, "a5" stands for "ắ", nor will she like typing sequences as long as "&_a(′_" for some letter in every word.

To satisfy the readability and flexibility requirements, a separate specification is necessary. It is better to adopt an approach like code-page switching under ISO-2022 [13] to switch the text into "Vietnamese" mode and optimize encoding according to the language state. Recently, van der Poel put forth a mnemonic proposal [14] which emphasizes language-specific conventions for these reasons. This proposal provides a means to specify the language state, each with its own (efficient) encoding method. Its strength lies in the flexible specification that conformant implementations "need not be able to display all of the character sets specified"; they have the option of stating messages such as "undisplayable Greek appeared here" for unsupported languages (for a more precise specification, see [14]). This allows networking communities to determine the best approach for encoding their own languages. The VIQR convention is compatible with this approach and should easily be incorporated into this framework.

The specification here encompasses all data streams including text transfer, file I/O, and keyboard entry. This principle has been the major reason for success in operating systems such as Unix, in which device-specific details are hidden as much as possible from the applications programmer, leaving a uniform interface above which tools such as common library routines can be shared. Indeed as the keyboard example above has implied, the characters actually typed by the user are often not different from the text data that is eventually stored or transmitted. It is therefore desirable to provide a common base on which to build data interpreters for all data streams, independent of the input source. In actual implementation, this has greatly facilitated development of the Vietnamese-capable software base.

In addition, the user stands to benefit tremendously from standardization of keyboard entry. One does not need to learn a different keyboard entry technique for each different Vietnamese application. If one standard keyboard model is fully supported by all Vietnamese software, a user familiar with the standard can sit down and start typing Vietnamese immediately. This standard defines the minimum expected behavior from compliant software; any additional input techniques can of course be incorporated as a superset of the standard behavior. This is discussed further in Section 5.2 on Vietnamese keyboarding.

4.2 QUOTED-READABLE SPECIFICATION (VIQR)

The mnemonic model from Viet-Net is fully employed in the specification. The Vietnamese QR comprises three major states: Literal, English, and Vietnamese. The Literal state is intended for completely transparent handling of literal data (except of course for the escape sequences into and out of Literal state). The English and Vietnamese states are designed for mixed use of English and Vietnamese, with each optimized in appearance as well as data size for texts containing mostly English and Vietnamese, respectively. In either state there exist methods for composing Vietnamese-specific characters, using a base vowel followed by one or two diacritics.

We first introduce the concept of implicit and explicit composition, then discuss how they are used in each of the states.

[7] The convention is "&xy", where x is a literal character and y represents some combining form.

[8] The convention is "&_xxxx_" where xxxx can be an arbitrary mnemonic sequence.

Table 3: VISCII 8-bit Encoding Standard Proposal for Vietnamese.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x	NUL	SOH	Ả	ETX	EOT	Ẵ	Ẫ	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	Ỷ	NAK	SYN	ETB	CAN	Ỹ	SUB	ESC	FS	GS	Ỵ	US
2x	SP	!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8x	Ạ	Ắ	Ằ	Ặ	Ấ	Ầ	Ẩ	Ậ	Ẽ	Ẹ	Ế	Ề	Ể	Ễ	Ệ	Ố
9x	Ồ	Ổ	Ỗ	Ộ	Ợ	Ớ	Ờ	Ở	Ị	Ỏ	Ọ	Ỉ	Ủ	Ũ	Ụ	Ỳ
Ax	Õ	ắ	ằ	ặ	ấ	ầ	ẩ	ậ	ẽ	ẹ	ế	ề	ể	ễ	ệ	ố
Bx	ồ	ổ	ỗ	Ỡ	Ơ	ộ	ờ	ở	ị	Ự	Ứ	Ừ	Ử	ơ	ớ	Ư
Cx	À	Á	Â	Ã	Ả	Ă	ả	ã	È	É	Ê	Ẻ	Ì	Í	Ĩ	ỳ
Dx	Đ	ứ	Ò	Ó	Ô	ạ	ỷ	ừ	ử	Ù	Ú	ỹ	ỵ	Ý	ỡ	ư
Ex	à	á	â	ã	ả	ă	ữ	ẫ	è	é	ê	ẻ	ì	í	ĩ	ỉ
Fx	đ	ự	ò	ó	ô	õ	ỏ	ọ	ụ	ù	ú	ũ	ủ	ý	ợ	Ữ

Table 2: Vietnamese-specific characters already present in 8859/Latin-1.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Cx	À	Á	Â	Ã					È	É	Ê		Ì	Í		
Dx	Đ		Ò	Ó	Ô	Õ				Ù	Ú			Ý		
Ex	à	á	â	ã					è	é	ê		ì	í		
Fx	đ		ò	ó	ô	õ				ù	ú			ý		

(non-breaking space character on Macintosh), or 255. The list of potentially non-graphic characters in C1 and G1 can be quite large: nearly 30 characters in MS Windows 3.0 and roughly 25 characters in MS Windows 3.1. These positions must be populated with upper case characters in consistence with the above philosophy. In applications where font switching is allowed and upper case characters are blocked out, a solution is to supply fonts in pair: a normal font and a capital font. In the capital font all the positions that should be filled with lower case characters are actually filled with the corresponding upper case. When a capital letter in the normal font cannot be rendered, the user simply switches to the corresponding capital font and types in the corresponding lower case character.

With the above guidelines, the task is then to lay out the remaining Vietnamese characters in some fashion, perhaps even arbitrary. This has been done in such a way so as to provide some degree of symmetry simply for aesthetics. It turns out that all the above guidelines can be adhered to except for compatibility with the letter `Õ` (`O tilde`) in 8859/Latin-1. Note that the Vietnamese collating order cannot in any case be preserved, but this is not a major issue since collation for non-ASCII characters is well accepted to be a table-lookup problem.

The preceding guidelines have resulted in the VISCII 8-bit Vietnamese encoding proposal listed in Table 3. It is intended to be a single table that applies to Vietnamese data handling including storage, processing, transmission, and font encoding. This greatly simplifies the integration, implementation, and usage processes and is indeed one of the major strengths of the proposal.

4 VIQR: MNEMONIC ENCODING SPECIFICATION FOR VIETNAMESE

4.1 MOTIVATION

While the 8-bit specification attempts to standardize Vietnamese encoding in 8-bit environments, much remains to be addressed in important 7-bit environments such as electronic mail transport and other 7-bit data lines, as well as in keyboard entry applications where the interface for generating Vietnamese characters needs to be standardized.

Transporting more than 128 unique symbols over 7-bit data channels is not a problem specific to the Vietnamese language. Since its proposal in 1982, the Internet Simple Mail Transfer Protocol ("SMTP", [8]) has seen unrelenting efforts to extend it to accommodate 8-bit and wider-word data in European Latin scripts and Oriental ideographic characters (see, e.g., [9]). While clean 8-bit transport is highly desirable, all mail gateways are not going to be converted overnight. For the foreseeable future there is a need for unambiguous transport of Vietnamese text over existing 7-bit channels.

Indeed there is an *ad-hoc* standard in use on the Viet-Net mailing list and the Usenet newsgroup Soc.Culture.Vietnamese, where mnemonic use of appropriate characters to follow a vowel proves to be quite readable; for example, "Việt Nam" would be written as "Vie^.t Nam". However, this is troubled by the ambiguity in the multiple roles played by the mnemonic diacritical marks; for example, does "tha?" mean "tha?" or "thả"?

The Viet-Net convention is not far in concept from a quoted-readable format proposed by K. Simonsen [10, 11]. which disambiguates such texts by specifying text states at both the character and character set levels. Unfortunately, in its attempt to provide a universal solution to mnemonic encoding, the proposal does not provide a good answer for Vietnamese text. First, it restricts the

Table 1: A sampler of possible C0 usage conflicts. Codes selected for this standard proposal are noted with a †.

CODE	COMM.	CTRL-	GENERAL	PRINTER (PC)	PC	UNIX	VI (Unix)
0	NUL	@	C string			strings	
1	SOH	A					
2†	STX	B					back screen
3	ETX	C	INTR		INTR	INTR	INTR
4	EOT	D	EOF			EOF	back tab
5†	ENQ	E					
6†	ACK	F					forw.screen
7	BEL	G	BEL	BEL	BEL	BEL	
8	BS	H	BS	BS	BS	BS	BS
9	HT	I	HT	HT	HT	HT	HT
10	LF	J	LF	LF	LF	LF	LF
11	VT	K		VT			
12	FF	L	FF	FF		FF	redraw
13	CR	M	CR	CR	CR	CR	CR
14	SO	N		wide on(IBM)			
15	SI	O		comp.on(IBM)			
16	DLE	P			Prt.on/off		
17	DC1	Q	XOFF	XOFF	XOFF	XOFF	
18	DC2	R		comp.off(IBM)		retype	
19	DC3	S	XON	XON	XON	XON	
20†	DC4	T		wide off(IBM)			forw.tab
21	NAK	U		clr buf(IBM)		kill	kill
22	SYN	V				literal	literal
23	ETB	W				werase	werase
24	CAN	X				kill	
25†	EM	Y				suspend	
26	SUB	Z			EOF	suspend	
27	ESC	[	ESC	ESC sequence	ESC	ESC	ESC
28	FS	\				quit	
29	GS	]	Telnet ESC				
30†	RS	^					
31	US	_			Windows		

of C0 encoding. In any event, the option exists for data to be sent in some "binary" mode, or to employ the Vietnamese Quoted-Readable format to be described in Section 4.

The overwhelming advantage of this approach is that it is readily and easily integrated into existing environments without many of the problems plaguing the other alternatives, if they can at all be integrated. As a testimony to the approach's successful application, this document itself was prepared using the TEX system under Unix. The text source was edited in an 8-bit X terminal window, using a minimally modified[5] version of Elvis, a public-domain 8-bit version of Unix's Vi text editor. Both TEX (a document preparation system) and Dvi2ps (a PostScript generator) readily accepted and processed Vietnamese (8-bit) data transparently. Many other applications including a spreadsheet, various text viewers, PostScript and dot-matrix printing, DOS's WordPerfect, Word, PC Tools, etc., have been tested and seen to operate well with Vietnamese text. Modifications, if any, were primarily in making these applications accept 8-bit data. An educational teaching tool for Vietnamese has also been produced using the C programming language with 8-bit Vietnamese strings embedded in the source code. With increasing system internationalization, applications and tools are being made 8-bit "clean," further facilitating integration of this Vietnamese encoding.

3.2 ENCODING RATIONALE

A basic requirement is to preserve the 7-bit ASCII graphic characters (G0) layout, since the emphasis is on integration. G0 was therefore left unchanged. For the 6 C0 characters, we first lay out the code space and consider typical usage, a sampler of which is in Table 1. The codes selected, STX (2), ENQ (5), ACK (6), DC4 (20), EM (25), and RS (30) present the least possible problems with data communication and significant applications considered. The use of ACK, for example, is actually context-dependent. In those protocols we have reviewed, it is only considered a "control" character outside of a data frame; within a data frame it is transfered without special interpretation. To reduce the probability of conflict even further, the 6 least-often used Vietnamese capital letters, Ả, Ã, Ẫ, Ỷ, Ỹ, and Ỵ, are encoded into these slots.

The remaining task is to encode the other 128 Vietnamese characters into the extended ASCII space (C1 and G1). Since no unique international encoding standard exists in this region, the philosophy is to be as much conservative as possible so that in the worst case the user can still use all of the lower case Vietnamese letters.

[5] The modifications provided the keyboard interface described in later sections.

The encoding of C1 is less troublesome, although in application-specific contexts it has been found that some C1 characters are employed with special meanings. A review of ongoing work on 8-bit mail transport standardization indicates that C1 characters will be fully supported as graphic characters without special interpretation. Nevertheless, it is prudent to encode only uppercase characters into the C1 space.

For G1, the aim is to accommodate the popular PC character set (code page 850) and to adhere, if possible, to the 8859/Latin-1 mapping where Vietnamese-specific characters are already encoded.

Experience in development of this encoding on the MS-DOS platform motivates the consideration of line-drawing glyphs in the PC character set. In many situations where both Vietnamese and line-drawing characters are desirable but font switching is impossible, the best we can do is to preserve all the lower case Vietnamese characters and all the single- and double-line drawing characters. This means that code positions occupied by single- and double-line drawing characters must be populated with upper case letters. With this provision, the MS-DOS user can be supplied with either code pages containing all Vietnamese glyphs or code pages where a number of upper case Vietnamese characters are replaced by PC line-drawing characters. For existing applications, the user can choose the code page most appropriate for her purpose. Where the code page with line-drawing characters must be used, the penalty from missing Vietnamese characters has been minimized by the choice of the infrequently used ones. For new applications, code page switching can easily be done on the fly, if it is desired.

Compatibility with the 8859/Latin-1 standard is merely for user friendliness and is not mandatory. It is natural and reasonable for a user in France to expect that the same keystrokes producing "é" on the screen for French will do the same for Vietnamese. The motivation for this compatibility is the predominant and increasing availability of 8859/Latin-1 keyboards and font sets, e.g., Digital's VT-terminal series, Xterm keymaps, and Microsoft's Windows. Table 2 lists the subset of 8859/Latin-1 characters in G1 that are also Vietnamese.[6] It can be concluded that all 8859/Latin-1 text that contains characters mostly from G0 (ASCII) and this table, French text for example, is highly readable in the Vietnamese environment.

Finally, certain characters in G1 are not renderable in a number of applications such as character codes 160 (non-breaking space character in 8859/Latin-1), 202

[6] Note that the "đ" in Table 2 is actually a similar-looking Icelandic "edh" in 8859/Latin-1; the Vietnamese rendering form is better reflected in 8859/Latin-2.

A3. Drop 6 of the "least-used"[4] Vietnamese characters, typically accented capitals such as Å̉, Ã̃, Ã̃, Ỷ, Ỹ, and Ỵ.

A4. Map accented "y" combinations into corresponding "i" combinations, e.g., "kỹ sư" is replaced with "kĩ sư."

A5. Encode into the ASCII control space C0.

Approaches A1 and A2 both satisfy the typical needs of the word processing environments in which rarely used ASCII characters can be avoided, or employed by font shifting. However they both eliminate prospects for integration of Vietnamese into existing ASCII environments where all graphic characters in G0 are needed. A character that already serves one purpose cannot be re-used for another. First, it makes rendering of the needed G0 character incorrect, as it would now look like a Vietnamese character. The frequency of use of G0 characters in an integrated environment is far too high for this conflict to be tolerable. Second, while font shifting may be employed to remedy this in some situations, a more serious problem occurs when the Vietnamese character is needed. The environment would typically have assigned some specific meaning to the G0 character, particularly with those in the NRC set. Consider, for example, using the backslash character "\" for a Vietnamese character under Unix. The backslash is used for many escape mechanisms under Unix so that the Vietnamese character cannot simply be used but must be escaped in one way or another. This is more than just an inconvenience; it means data interchange is complicated by the fact that the escape mechanism will not be understood on another platform, and data integrity has thus not been preserved. A standard employing this approach fails at its basic mission: to provide cross-platform transparency. A similar case can be made for the other G0 characters.

Both A3 and A4 propose to limit Vietnamese language data in one way or another. Most agree that elimination of some Vietnamese characters are simply unacceptable; indeed, this point is so fundamental that we have in the foregoing chosen to assume it as a technical requirement without elaboration. However, it must be said that A4 is not a proposal without rationale. A school of thought exists that believes y's existing in words as a single vowel should be mapped to corresponding i's, as their pronunciations are indeed identical. The concept dates as far back as 1948 [6, 7]. However, it is not the function of an encoding standard to settle a linguistic issue, and hence A4 is also a bad choice.

The immediate objection to A5 is primarily in data communication channels where many C0 characters are used as data control. In addition, it also presents problems for integration into environments where some C0 characters are used in the keyboard interface and in data format controls, similar to the problem facing A1 and A2. However, as will be discussed further, judicious choice of the 6 C0 characters to be used has in practice been shown successfully to avoid characters that are significant in data communication. Furthermore, most data channels provide for clean transfer of binary data, and there is no reason to worry that arbitrary data bits cannot be employed over these binary routes.

With those particular cases where C0 is used in the keyboard interface, judicious choice as well as remapping of keys can minimize conflict. Data format control is application-specific but is typically scattered in C0 and C1. It is therefore a universal problem for integration because C1 is necessarily densely encoded, but, again, conflict can be avoided by studying significant applications. Finally, the choice can be made for 6 least-used Vietnamese characters so that the probability of conflict is greatly reduced.

It should be noted here that the foregoing discussion has subjected the alternatives to the requirements of integration into existing applications and platforms, as outlined in Section 1. The importance of this goal cannot be overstated, and it does present complications that result in the following Pragmatism Principle: it is obviously impossible to define a standard that would operate seamlessly with all existing applications, therefore pragmatic considerations must be made to make a standard workable in as many important applications and on as many platforms as possible, with emphasis on the word "workable."

3 VISCII: 8-BIT ENCODING SPECIFICATION FOR VIETNAMESE

3.1 MOTIVATION

The available body of evidence shows that alternative A5 described in the previous section, encoding into 6 of the C0 characters, has the greatest chance of success in fulfilling the requirements outlined in Section 1. The choice of the 6 C0 codes and the 6 least-used Vietnamese capital letters to encode, when made carefully, greatly reduces the probability of conflict for all practical purposes. Concerns regarding data communications are well addressed by avoiding C0 codes that are in fact often used for data control. Indeed, data communication concerns are more applicable to C1 and G1 encoding; a prominent example is electronic mail transfer through 7-bit gateways and mail agents. Communication failure here has in most cases been due to the use of the eighth bit and not because

[4] Least-used because they (a) rarely begin words and therefore do not often get capitalized, and (b) appear in fewer words.

familiar "don't reinvent the wheel" rule is not only an advantage—but a necessity—if a meaningful application base is to be established in any reasonable length of time. Furthermore, it is known that overall efficiency both in time and space is greater in processing precomposed character units when compared with the floating-diacritic approach [2]. Floating diacritics therefore must be limited to only where they are necessary and inevitable, such as in keyboard entry or 7-bit data transmission. There is no reason to require that all applications must deal with the complexities and inefficiencies of floating diacritics, for example, in 8-bit data processing, storage, transmission, screen rendering, or printing.

The second major context points to the pragmatic and vital consideration of existing precedents set in the Vietnamese software base. Standardization necessarily requires adaptation, but it makes little sense to propose to change the world so significantly that the inertia against large changes greatly delays adoption of the standard. The trend towards 16-bit and wider data standards for multinational character sets has gained momentum with the recent works of Unicode [3] and ISO 10646 [4]. However, the need for an 8-bit Vietnamese standard is irreplaceable until these new standards are fully supported and completely dominate the computing world. An 8-bit Vietnamese standard must not ignore existing software precedents so that it can gain speedy acceptance before it becomes obsolete.

Thirdly, the standard must address the issue of user interface; if not defining it, then at least consider its possible effects on the end-user. This relates primarily to the 7-bit keyboarding and representation of Vietnamese—in both instances diacritics are necessarily floating, and represented mnemonically by existing 7-bit characters with similar appearance. With keyboarding, one must preserve where possible existing practices such as that defined for the Viet-Net mailing list and the Usenet newsgroup Soc.Culture.Vietnamese, both with members worldwide. For 7-bit readable representation, the keyword is "readable." The goals here are to maintain a short learning time and to promote a uniform interface so that it is not necessary for a user to re-learn the particulars of every software installation before being able to use it effectively.

Finally, to every extent possible, the standard must stay within the framework of international standards, e.g., ISO-8859/x [5], in order to ensure compatibility with existing environments. For example, this goal means preservation of the ASCII encoding. It should extend also to the encoding into the same 8859/Latin-1 slots those Vietnamese characters that are already defined, thus ensuring that 8859/Latin-1 keyboards will work transparently for those Vietnamese characters. However, there are many standards requirements that are obsolete from a practical viewpoint. For example, in recent Unicode/ISO-10646 decisions, the prohibition from use of the available control character space—those with encodings between xx00h and xx1Fh, except for C0 itself—was discarded on the grounds that it was a waste of encoding space. As will be discussed later, the encoding of Vietnamese into the existing 8-bit space presents some well-known trade-offs. Where trade-offs are made, they must be justified with good reason—pragmatic preferred over theoretical.

These primary requirements are summarized as follows:

- R1. Straightforward and direct integration into existing platforms.
- R2. Ease of adaptation for existing software.
- R3. User-friendly mnemonic encoding scheme and interface.
- R4. Adherence to international standards.
- R5. Trade-offs made only on practical usage considerations and with good reason.

In the following section we present a brief review of the strengths and weaknesses of different approaches to Vietnamese encoding. Section 3 will describe the proposed 8-bit encoding table in detail. A quoted-readable encoding scheme encompassing 7-bit data streams, including electronic mail and keyboard input, is presented in Section 4. Finally, Section 5 outlines the particular rules and conventions relevant in some application-specific contexts.

2 REVIEW OF CURRENT CONVENTIONS

A review of current conventions used by software vendors reveals one distinct feature: virtually all realize the strengths of a precomposed encoding and adopt it as a primary requirement. The complications arise from a familiar fact: apart from the alphabetics already available in the ASCII standard, Vietnamese requires an additional 134 unique characters. Of these, 128 can be coded in the C1 and G1 areas. The allocation of the remaining 6 characters in the lower C0 and G0 space is handled with differing approaches:

- A1. Encode into 6 of the "least-used" G0 characters in the context of Vietnamese data processing.
- A2. Encode into 6 slots of the National Replacement Character[3] (NRC) set.

[3] This set contains 12 country-specific characters at code positions corresponding to ASCII characters #, $, @, [, \,], ^, `, {, |, }, ~.

A Unified Framework for Vietnamese Information Processing

Vietnamese Standardization Working Group[1]
September 1992[2]

ABSTRACT

Increasing demand for Vietnamese electronic information processing has seen answer in a wide array of Vietnamese-capable applications. The inevitable need for integration of Vietnamese into existing environments and the exchange of data among them point to the necessity of standardization. This paper presents the strategic and pragmatic technical considerations that must go into such a standard, and reviews existing conventions/proposals in these important contexts. A full description of the Viet-Std proposal is presented, including 1) an 8-bit, fully precomposed encoding table for Vietnamese Standard Code for Information Interchange (known as VISCII), 2) a 7-bit Vietnamese Quoted-Readable (known as VIQR) standard for data interchange over 7-bit channels, with a seamless interface to the 8-bit encoding, and 3) a keyboard user-interface specification that works transparently with both 1 and 2. Together, these provide a truly unified framework for a Vietnamese information processing environment with simplicity, efficiency, and straightforward integration. The real-world construction of this framework has proven quite successful in an array of compliant applications from a number of group and individual developers across a number of platforms, including Unix and its variants, the X window system, MS-DOS, Windows, and with ongoing work elsewhere.

1 INTRODUCTION

With the growing Vietnamese population abroad and the proliferation of computer usage within Viet Nam, the Vietnamese language has seen rapidly increasing representation in electronic information processing. The concomitant growth in demand for Vietnamese-capable software has resulted in successful launches of myriad vendors in the U.S. and elsewhere, mainly in the area of Vietnamese word processing. In addition, individual and group efforts have also been productive in providing Vietnamese-language users with high-quality public-domain applications. In Viet Nam, centers such as the Institute of Informatics have reported impressive progress on many fronts, among which is the Vietnamization of standard software packages [1].

All of the above illustrate two important points: 1) There are growing market demands for Vietnamese-capable processing engines, and 2) There is no shortage of technical talent to fulfill those demands. Unfortunately, therein lies a large problem: most existing Vietnamese applications have been designed to operate in the exclusive framework or environment of the developer, and all are incompatible with one another. As long as this trend continues, the application base for Vietnamese can never keep reasonable pace with demand. Users want to do more with Vietnamese than mere word processing, and to expect one single vendor to provide all potential applications across all platforms is to dream the impossible. Technicians providing these applications are limited to the Vietnamese tools they must themselves learn and develop from the ground up. Standardization is necessary. Anyone who has had to deal with the incompatibility between ASCII and EBCDIC can try to imagine a world where every machine is using a different character set, and appreciate how limited that world would be in its application base and how cumbersome in its data interchange. A uniform framework will greatly benefit both the user and the technician alike.

The proposal for any Vietnamese data standardization must take several important points in the proper contexts. First and foremost, since this discussion is geared toward existing 7- and 8-bit environments, the prime goal is straightforward and direct integration onto current platforms. The standard must work here and now. This implies the use of precomposed Vietnamese characters, because the handling of floating diacritics will never see full or simple support outside of specific contexts. The standard must be designed so as to take advantage of existing applications as much as possible. The

[1] Postal address: Viet-Std, 1212 Somerset Dr., San Jose, California 95132, USA. E-mail address: Viet-Std@Haydn.Stanford.EDU

[2] This version 1.1 supersedes version 1.0 of January 1992. The only significant difference is the exchange of the positions of ạ (`a dot-below`) and Õ (`O tilde`) in the 8-bit table.