A Dynamic Storage Model For Assyrian Computer Text

Peter (Jasim) BetBasoo
Nineveh Software Corporation

off-print from

SyrCOM-95

Proceedings of the First International Forum On Syriac Computing
(In Association wit Syriac Symposium II)

June 8, 1995
The Catholic University of America
Washington, D.C.

Edited by
George Anton Kiraz
University of Cambridge
(St. John's College)

Published by the Syriac Computing Institute

A Dynamic Storage Model for Assyrian Computer Text

Peter Jasim
Nineveh Software Corporation

In this paper I shall propose a model of representing Assyrian text in computer memory, and discuss proposed standards for an Assyrian keyboard and data interchange code. I shall also discuss the recent developments in the Unicode standard.

In his paper *On the Design of an Assyrian Word Processing System* (JAAS, Volume V, No. 2), Sargon Hasso proposes what I shall term a Static Storage Model (SSM) for representation of Assyrian text in a computer. The fundamental properties of SSM are:

> A glyph object structure composed of a character and a diacritical mark is used to represent a glyph (character+diacritical mark). A character requires 1 byte of storage, as does a diacritical mark; the minimum storage for a glyph is, therefore, 2 bytes. The glyph object structure can be visualized as follows
>
> Character::Diacritical mark
>
> A lookup table is used to render each glyph. This implies that *all possible combinations of characters and diacritical markings have been defined and placed in this lookup table*. It is for this reason that I call this the Static Storage Model.

In the Static Storage Model there is a many-to-one relationship between what is internally stored in the computer and what is rendered on an output device (such as a monitor or printer). For example, the following

ܐ̇

is stored internally as 65::97 (glyph codes are defined in Appendix A). A computer would use these codes to find the predefined glyph *Alap+Zqapa* in a lookup table. Each glyph object will have a unique entry in the lookup table. Here is an example for the word ܒܹܝܬ݂ܐ

Glyph	**Output**
66::98	ܒ݂
74::0	ܝ
86::97	ܬ̇
65::0	ܐ

Zero indicates no diacritical mark. Eight bytes are used to represent this word, two of which (the zeroes) are unused. It is important to realize that under SSM the computer has every possible glyph predefined in the lookup table. For this reason, the computer cannot represent any new combination of character and diacritical mark. Assuming there are 22 letters and abouty 20 diacritical marks, the lookup table would contain at least 22*20, or 440 glyphs. This assumes that a letter can have only one diacritical, which is not the case; the actual size of the lookup table will, therefore, be larger.

Another limitation of SSM is that it cannot represent multiple diacritical marks on the same character in an efficient way. For example, in the word ܓܡ̈ܠܐ, *Gamal* has two diacritical marks. The glyph object structure, however, can only store one. SSM fails in this case. This problem can be solved by special processing, but this comes at the expense of generality and complex algorithms.

In SSM the glyph object structure is a character and a diacritical mark. This leads to unusual and undesirable editing operations. If a user presses the delete key, what should be deleted, the character or the diacritical mark? Separate keys must be used to delete characters and diaciritical marks.

To summarize, the Static Storage Model makes inefficient use of memory, and it cannot handle characters with multiple diacritical marks. SSM also has a many-to-one relationship between internal storage and external representation, which forces the development of very complex rendering algorithms. In addition, many unusual and undesirable effects arise, all because of a poorly designed data structure. There is a far simpler alternative to SSM.

A Dynamic Storage Model

The Dynamic Storage Model (DSM) has the following fundamental properties:

Each letter or diacritical mark is stored as a unique, 1 byte code, separately and independently of its neighbors.

Each character or diacritical mark has a location property, which tells the computer where it should be placed: at the previous position, at the current position, or at the next position.

Each character or diacritical mark has a cursor effect property, which tells the computer how to move the cursor: backward, no motion, or forward.

A lookup table, which is called a font, is defined to contain only atomic glyphs; i.e., individual characters and diacritical marks. *The computer dynamically combines these to produce various combinations of characters and diacritical marks.* The font will contain, at most, 223 glyphs.

The Dynamic Storage Model has a glyph object structure which is 1 byte in length. Here is the previous example using DSM

Glyph	Output
66	
98	
74	
86	
97	
65	

The following properties are true of DSM

1. DSM requires less storage space. Only six bytes are required to store this word, whereas SSM requires eight bytes -- a 25% reduction in storage space.
2. Each glyph is stored consecutively in memory.
3. There is a one-to-one relationship between internal and external representation.
4. The diacritical marks *Ptakha* and *Zqapa* have the following properties

	Location Property	Cursor Effect
Ptakha	previous position	no motion
Zqapa	previous position	no motion

The remaining diacritical marks are similarly defined (Appendix E).

DSM handles multiple diacritical marks on the same letter in a natural and intuitive way. For example, the word ܚܘܟܠܐ, is stored as follows

Glyph	Output
85	ܚ
70	ܘ
103	.
67	ܟ
103	.
110	́
76	ܠ
97	·
65	ܐ

DSM does not impose unusual editing operations on the user. For example, a delete operation would delete the glyph currently pointed to, be it a character or a diacritical mark. Hence, one key would be used for deletion, thus maintaining complete generality.

I have touched upon only a few of DSM's properties. There are many technical issues which arise in implementing DSM in a software system; it is beyond the scope of this paper to discuss these in detail. Appendix E contains the DSM specification for Eastern Assyrian. As can be seen from Appendix E, there is very little, aside from the script, that is specific to Eastern Assyrian (and not to Western Assyrian or Estrangelo). DSM transparently handles all three cases.

Four Essential Standards

Uniform standards are crucial for the development of hardware and software systems. The two most basic standards are a standard keyboard layout and a standard data interchange code, as well as a font standard and a contextual analysis standard. These four standards work conjunctively; it is not possible to omit one without effecting the system.

Data Interchange Code

A Data Interchange Code allows one computer to communicate with another. For example, it would be undesirable to have one computer store the letter *Alap* as 65, and another to store it as 100. Documents written on one machine would display garbage when shown on the second. In addition, a standard code is necessary for proper lexical operations, such as searching and sorting. Once again, I present the standard that was developed at the **First Ashurbanipal Library Computer Conference**, but slightly modified for improvement. This standard is called **SACII, Standard Assyrian Code for Information Interchange**. Please refer to Appendix A.

Keyboard Layout

It is important to have a standard Assyrian keyboard layout so that, once having learned the layout, a person can sit and use any Assyrian keyboard without retraining. The Assyrian Standard Keyboard Layout (ASKL) was developed at the **First Ashurbanipal Library Computer Conference**. The layout is based on a computer analysis of the frequency of use of each Assyrian letter. The most often used letters are placed near the center of the keyboard, and the least used are placed to either side (refer to the *Proceedings of the First Ashurbanipal Library Computer Conference* for more details). I have modified ASKL slightly since the original standard was published, mainly to make it compatible with modern operating systems (i.e., OS/2, Windows, Macintosh), and to remove the reliance on special shift keys. ASKL is shown in Appendix B.

Contextual Analysis

It is not possible to have a practical keyboard layout standard without contextual analysis, since letters in the Assyrian alphabet change shape depending on their position in a word. Appendix C specifies a standard method of contextual analysis.

Font Standard

Every Assyrian font, be it Eastern, Western, *Estrangelo*, or a new, modern creation, must conform to the font standard prescribed in Appendix D. The font standard is a corollary of SACII, and it is stated explicitly for emphasis.

Application of the model

Appendix E contains a specification for the Eastern Assyrian font based on the concepts developed in this paper. As can be seen, the combination of DSM and the proposed standards provides a robust approach to the problem of computerizing the Assyrian language.

Unicode and the Assyrian Language

There are two prevailing standards for information interchange codes, ASCII (American Standard Code for Information Change), which is used by all personal computers, and EBCDIC (Extended Binary Coded Data Interchange Code), which is used mainly by IBM mainframe computers. Both ASCII and EBCDIC define 256 codes for data interchange. For example, in ASCII the letter A is code 65, the letter B is code 66, and so on. Because ASCII and EBCDIC are limited to 256 codes, they cannot handle a language that has more than 256 characters (such as Japanese). Unicode was developed to solve this problem; it provides 65,536 codes for use, which is enough to encode all of the world's languages. Unicode will, it is pleasing to know, support Assyrian as well. The author and Sargon Hasso have submitted the Assyrian Unicode Standard to the Unicode Consortium, which has accepted the Assyrian Standard and is in the process of ratifying it.

Conclusions

In this paper I have presented a powerful storage model for representing Assyrian text in computer memory. I have also proposed standards for keyboard layout and data interchange codes. It is important to understand that DSM, ASKL, and SACII are dialect independent, i.e., they work with Eastern Assyrian, Western Assyrian (*Serto*), and *Estrangelo*. Indeed, if a computer system implements DSM and the proposed standards, a user will be able to switch from one font (Eastern, Western, or *Estrangelo*) to another at will, or to convert text written in one font to another with one hundred percent accuracy, or to type text in any font in a uniform way.

References

Becker, Joseph. *Arabic Word Processing*. Communications of the ACM. July, 1987.

Becker, Joseph. *Multilingual Word Processing*. Scientific American. July, 1984.

DeKelaita, Joseph. *Grammar of the Aramaic Language*. Assyrian Church of the East Press. 1929

Hasso, Sargon. *On the Design of an Assyrian Word Processor*. Journal of the Assyrian Academic Society, Volume 4, No. 2.

Jasim, Peter, ed. *Proceedings of the First Ashurbanipal Library Computer Conference, Topic: Assyrian Word Processing*. Ashurbanipal Library Press. 1989.

Knuth, Donald. *Art of computer Programming, Volume 1: Fundamental Algorithms*. Addison Wesley Publishing Company. 1973

Knuth, Donald. *Art of computer Programming, Volume 2: Sorting and Searching*. Addison Wesley Publishing Company. 1973

Segal, J. B. *Diacritical Points and the Accents in Syriac*. Oxford University Press. 1953.

Appendix A

SACII
Standard Assyrian Code for Information Interchange

This appendix lists the *Standard Assyrian Code for Information Interchange* (SACII), from 0 to 255.

To insure that data can be transferred from one computer to another, all vendors should use SACII faithfully. SACII provides a broad, fundamental foundation upon which all applications will be developed, such as database management, lexical analysis, spelling checkers, and so forth.

SACII defines the letters, symbols, and punctuation marks used in writing Assyrian. This includes Eastern Assyrian, Western Assyrian (*Serto*), and *Estrangelo*. As a matter of fact, no distinction is made between these three fonts at this level.

SACII fully supports contextual analysis. It reserves codes for the free, initial, middle, and final forms of each letter.

In the following table each letter is shown in Eastern, *Estrangelo*, and Western, in that order.

Code	Symbol	Name
------	----------	-------
0-31		same as ASCII control codes
32		*Shopa speeqa* (ܫܘܦܐ ܣܦܝܩܐ)
33	!	*Neeshanqa dpooqdana* (ܢܝܫܢܩܐ ܕܦܘܩܕܢܐ)
34	"	*Mamrana* (ܡܐܡܪܢܐ)
35	#	*Minyana* (ܡܢܝܢܐ)
36	¨	*Rahta* (ܪܗܛܐ)
37	%	*Immoona* (ܐܡܘܢܐ)
38	:	*'aseer* (ܥܣܝܪ)
39	'	*Mkhayiddana* (ܡܚܝܕܢܐ)

40	(	*Qishta simmalaita* (ܩܫܬܐ ܣܡܠܝܬܐ)
41	)	*Qishta yameenaita* (ܩܫܬܐ ܝܡܝܢܝܬܐ)
42	:	*Mtakspana* (ܡܬܟܣܦܢܐ)
43	+	*Mazyiddana* (ܡܙܝܕܢܐ)
44	،	*Neeshanqa dnoohara* (ܢܝܫܢܩܐ ܕܢܘܗܪܐ)
45	‐	*Mapserana* (ܡܦܣܪܢܐ)
46	.	*Pasoqa* (ܦܣܘܩܐ)
47	/	*Palee'ana* (ܦܠܝܓܢܐ)
48	0	*Seepar* (ܣܝܦܪ)
49	1	*Kha* (ܚܕ)
50	2	*Tre* (ܬܪܝܢ)
51	3	*Tlata* (ܬܠܬܐ)
52	4	*Arb'a* (ܐܪܒܥܐ)
53	5	*Khamsha* (ܚܡܫܐ)
54	6	*Ishta* (ܐܫܬܐ)
55	7	*Shaw'a* (ܫܒܥܐ)
56	8	*Tmanya* (ܬܡܢܝܐ)
57	9	*Tish'a* (ܬܫܥܐ)
58	:	*Zawga* (ܙܘܓܐ)
59	+	*Pasoqa kirya* (ܦܣܘܩܐ ܟܪܝܐ)
60	<	*Soora min* (ܙܘܪܐ ܡܢ)
61	=	*Dma* (ܕܡܐ)
62	>	*Goora min* (ܓܘܪܐ ܡܢ)
63	؟	*Neeshanqa dshooala* (ܢܝܫܢܩܐ ܕܫܘܐܠܐ)
64	¨	*Napsa* (ܢܦܫܐ)
65	ܐ (ܐ א)	*Allap*
66	ܒ (ܒ ב)	*Bet*
67	ܓ (ܓ ג)	*Gammal*
68	ܕ (ܕ ד)	*Dallat*

69	ܗ (ܗ ܗ)	*Hea*
70	ܘ (ܘ ܘ)	*Wow*
71	ܙ (ܙ ܙ)	*Zen*
72	ܚ (ܚ ܚ)	*Kheth*
73	ܛ (ܛ ܛ)	*Teth*
74	ܝ (ܝ ܝ)	*Yood*
75	ܟ (ܟ ܟ)	*Kap*
76	ܠ (ܠ ܠ)	*Lammad*
77	ܡ (ܡ ܡ)	*Meem*
78	ܢ (ܢ ܢ)	*Noon*
79	ܣ (ܣ ܣ)	*Simket*
80	ܥ (ܥ ܥ)	*'e*
81	ܦ (ܦ ܦ)	*Pe*
82	ܨ (ܨ ܨ)	*Sade*
83	ܩ (ܩ ܩ)	*Qop*
84	ܪ (ܪ ܪ)	*Resh*
85	ܫ (ܫ ܫ)	*Sheen*
86	ܬ (ܬ ܬ)	*Tow*
87	reserved for 23rd Mandaic letter	
88	•	*Mzi'ana* (ܡܙܝܥܢܐ)
89	•	*Ritma* (ܪܬܡܐ)

90	·	*Msha'lana* (ܡܫܐܠܢܐ)
91	.	*Samka* (ܣܡܟܐ)
92	.	*Mnakhta* (ܡܢܚܬܐ)
93	∴	*Rahta dkarte* (ܪܗܛܐ ܕܟܪܬܗ)
94	^	*Marmana* (ܡܪܡܢܐ)
95	-	*Serta dkhooyada* (ܣܪܛܐ ܕܚܘܝܕܐ)
96	‡	?
97	˙	*Zqapa* (ܙܩܦܐ Western ܿ)
98	·	*Ptakha* (ܦܬܚܐ Western ܰ)
99	··	*Zlame psheeqe* (ܙܠܡܐ ܦܫܝܩܐ)
100	·.	*Zlame qishye* (ܙܠܡܐ ܩܫܝܐ Western ܶ)
101	:	*Khwasa* (ܚܒܨܐ Western ܼ)
102	·	*Rwakha* (ܪܒܚܐ Western ܼ)
103	.	*Rwasa* (ܪܒܨܐ ، ܪܒܨܐ in Eastern *and* Western)
104	··	*Syame* (ܣܝܡܐ)
105	^	*Rimkha* (ܪܡܚܐ below letter)
106	ᵕ	*Qishta* (ܩܫܬܐ below letter)
107	ᵕ	*Majleeyana* (ܡܓܠܝܢܐ below letter)
108	-	*Serta khteta* (ܣܪܛܐ ܬܚܬܝܬܐ below letter)
109	-	*Serta 'elayta* (ܣܪܛܐ ܥܠܝܬܐ above letter)
110	ˏ	*Talqana* (ܡܪܗܛܢܐ ܐܘ ܛܠܩܢܐ for silent or accented letters)
111	.·	*'elaye* (ܥܠܝܐ rests on either side of letter)
112	·.	*Takhtaye* (ܬܚܬܝܐ rests on either side of letter)
113	⁘	*Kikhwa* (ܟܘܟܒܐ)
114	†	*Sleewa* (ܨܠܝܒܐ)
115	\	*Mhagyana 2* (ܡܗܓܝܢܐ ܒ below letter)
116	/	*Mhagyana* (ܡܗܓܝܢܐ below letter)
117	...	*Gneezi* (ܓܢܝܙܐ)
118	\	*Stoona goonya* (ܣܛܘܢܐ ܓܘܢܝܐ)

119	⁝	*Stoona khtaya* (ܣܛܘܢܐ ܚܬܝܐ)
120	⁚	*Stoona 'elaya* (ܣܛܘܢܐ ܥܠܝܐ)
121	[	left bracket
122	]	right bracket
123		unused
124	܁	*Toopra simmalaya* (ܛܘܦܪܐ ܣܡܠܝܐ)
125	.	*Toopra yameenaya* (ܛܘܦܪܐ ܝܡܝܢܝܐ)
126	start rendering	*Sharee malkhamta* (ܫܪܝ ܡܠܟܡܬܐ)
127	stop rendering	*Maklee malkhamta* (ܡܟܠܝ ܡܠܟܡܬܐ)
128 - 150		reserved for free forms
151 - 173		reserved for initial forms
174 - 196		reserved for final forms
197 - 255		font specific (such as ligatures)

Codes 101 and 103 require clarification. In Eastern Assyrian ܝ݂ = *ee* (as in sh*ee*t), in Western Assyrian it is the diacritical mark ܺ which is *ee*. As far as the computer is concerned, ܝ݂ is just a letter with a dot under it, as are ܒ݂ ܘ݂ ܠ݂. One key, therefore, serves all these purposes. However, when one switches to a Western font the ̣ becomes a ܺ , and there no longer is a key for ̣ , which is still needed. Two codes must be defined, therefore, to guarantee a one-to-one relationship between Eastern and Western text. Code 101 has the same meaning in both Eastern and Western (*ee*); code 103 always means a dot under a letter, regardless of the font.

Codes 126 and 127 instruct the software to control rendering. This is useful in cases where a ligature, such as ܠܐ , is not desired -- code 127 would force the software to show ܐܠ in this case.

Appendix B

ASKL
Assyrian Standard Keyboard Layout

This appendix lists the complete definition of the Assyrian Standard Keyboard Layout (ASKL). ASKL was designed based on the frequency of use of each Assyrian letter; the most frequently used letters are placed in the center of the keyboard, and the letters least frequently used are placed on either side.

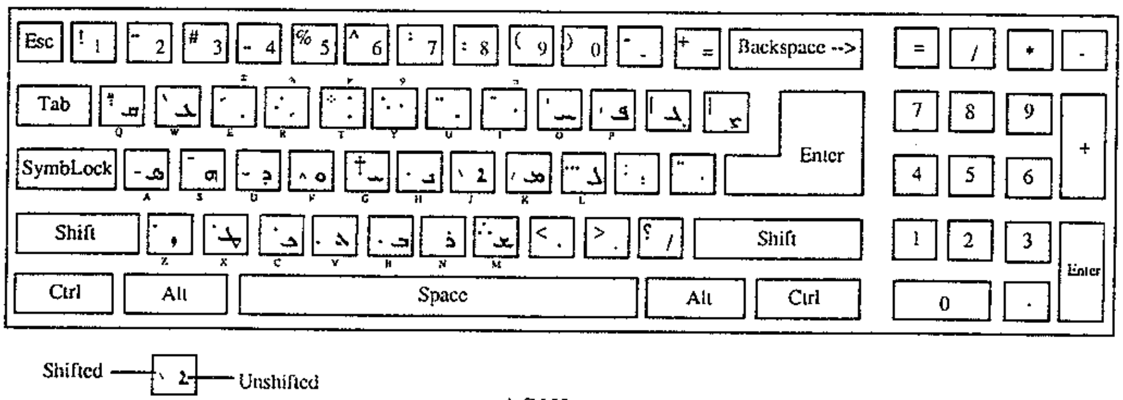
The following specification assumes contextual analysis (appendix C); keys are listed from top row to bottom row, from left to right as seen on a QWERTY (English) keyboard. The following key combinations are defined in SSKL.

QWERTY Key	ASKL	SACII	Description
1	1	49	*Kha* (ܚܕ)
2	2	50	*Tre* (ܬܪܝܢ)
3	3	51	*Tlata* (ܬܠܬܐ)
4	4	52	*Arb'a* (ܐܪܒܥܐ)
5	5	53	*Khamsha* (ܚܡܫܐ)
6	6	54	*Ishta* (ܐܫܬܐ)
7	7	55	*Shaw'a* (ܫܒܥܐ)
8	8	56	*Tmanya* (ܬܡܢܝܐ)
9	9	57	*Tish'a* (ܬܫܥܐ)
0	0	48	*Seepar* (ܨܝܦܪ)
-	-	45	*Mapserana* (ܡܦܣܪܢܐ)
=	=	61	*Dma* (ܕܡܐ)
Q	ܩ	83	*Qop*
W	ܥ	80	*'e*
E	.	101	*Khwasa* (ܚܒܨܐ Western)
R	.,	100	*Zlame qishye* (ܙܠܡܐ ܩܫܝܐ Western)

T	ܵ	98	*Ptakha* (ܦܬܵܚܵܐ Western ܲ)
Y		97	*Zqapa* (ܙܩܵܦܵܐ Western ܵ)
U		99	*Zlame psheeqe* (ܙܠܵܡܹܐ ܦܫܝܼܩܹܐ)
I		102	*Rwakha* (ܪܘܵܚܵܐ Western ܸ)
O	ܚ	72	*Kheth*
P	ܦ	81	*Pe*
[	ܓ	67	*Gammal*
]	ܨ	82	*Sade*
A	ܣ	79	*Simkat*
S	ܗ	69	*Hea*
D	ܕ	68	*Dallat*
F	ܘ	70	*Wow*
G	ܝ	74	*Yood*
H	ܢ	78	*Noon*
J	ܐ	65	*Allap*
K	ܡ	77	*Meem*
L	ܠ	76	*Lamnad*
;	܆	59	*Pasoqa kirya* (ܦܣܘܩܐ ܟܪܝܐ)
'		39	*Mkhayiddana* (ܡܚܝܕܢܐ)
Z	ܙ	71	*Zen*
X	ܛ	73	*Teth*
C	ܟ	75	*Kap*
V	ܬ	86	*Tow*
B	ܒ	66	*Bet*
N	ܪ	84	*Resh*
M	ܫ	85	*Sheen*
,	،	44	*Neeshanqa dnoohara* (ܢܝܫܢܩܐ ܕܢܘܗܪܐ)
.	.	46	*Pasoqa* (ܦܣܘܩܐ)
/	/	47	*Palee'ana* (ܦܠܝܥܢܐ)
SHIFT 1	!	33	*Neeshanqa dpooqdana* (ܢܝܫܢܩܐ ܕܦܘܩܕܢܐ)
SHIFT 2		36	*Rahta* (ܪܗܛܐ)

SHIFT 3	#	35	*Minyana* (ܡܢܝܢܐ)
SHIFT 4	..	64	*Napsa* (ܢܦܣܐ)
SHIFT 5	%	37	*Immoona* (ܐܡܘܢܐ)
SHIFT 6	^	94	*Marmana* (ܡܪܡܢܐ)
SHIFT 7	:	38	*'aseer* (ܥܣܝܪ)
SHIFT 8	:	42	*Mtakspana* (ܡܬܟܣܦܢܐ)
SHIFT 9	(	40	*Qishta simmalaita* (ܩܫܬܐ ܣܡܠܝܬܐ)
SHIFT 0	)	41	*Qishta yameenaita* (ܩܫܬܐ ܝܡܝܢܝܬܐ)
SHIFT -	-	95	ܣܪܛܐ ܡܚܘܝܕܐ (connector)
SHIFT =	+	43	*Mazyiddana* (ܡܙܝܕܢܐ)
SHIFT Q	⁑	96	
SHIFT W	\	118	*Stoona goonya* (ܣܛܘܢܐ ܓܘܢܝܐ)
SHIFT E	˜	110	*Talqana* (ܬܠܩܢܐ ܐܘ ܡܪܗܛܢܐ for silent or accented letters)
SHIFT R	.˙	111	*'elaye* (ܥܠܝܐ)
SHIFT T	⁘	113	*Kikhwa* (ܟܘܟܒܐ)
SHIFT Y	˙.	112	*Takhtaye* (ܬܚܬܝܐ)
SHIFT U	..	104	*Syame* (ܣܝܡܐ)
SHIFT I		107	*Majleeyana* (ܡܓܠܝܢܐ below letter)
SHIFT O	'	120	*Stoona 'elaya* (ܣܛܘܢܐ ܥܠܝܐ)
SHIFT P	ı	119	*Stoona khtaya* (ܣܛܘܢܐ ܬܚܬܝܐ)
SHIFT [	[	121	left bracket
SHIFT]	]	122	right bracket
SHIFT A	-	108	*Serta khteta* (ܣܪܛܐ ܬܚܝܬܐ below letter)
SHIFT S	-	109	*Serta 'elayta* (ܣܪܛܐ ܥܠܝܬܐ above letter)
SHIFT D	ᵕ	106	*Qishta* (ܩܫܬܐ below letter)
SHIFT F	^	105	*Rimkha* (ܪܡܚܐ below letter)
SHIFT G	†	114	*Sleewa* (ܨܠܝܒܐ)
SHIFT H	.	103	*Rwasa* (ܪܒܨܐ ، ܪܘܒܨܐ in Eastern *and* Western)
SHIFT J	\	115	*Mhagyana 2* (ܡܗܓܝܢܐ ܒ below letter)
SHIFT K	/	116	*Mhagyana* (ܡܗܓܝܢܐ below letter)

SHIFT L	•••	117	*Gneezi* (ܓܢܝܙܘܬܐ)
SHIFT ;	:	58	*Zawga* (ܙܘܓܐ)
SHIFT '	"	34	*Mamrana* (ܡܐܡܪܢܐ)
SHIFT Z	˙	88	*Mzi'ana* (ܡܙܝܥܢܐ)
SHIFT X	˙	89	*Ritma* (ܪܗܛܡܐ)
SHIFT C	˙	90	*Msha'lana* (ܡܫܐܠܢܐ)
SHIFT V	.	91	*Samka* (ܣܡܟܐ)
SHIFT B	.	92	*Mnakhta* (ܡܢܚܬܐ)
SHIFT N			
SHIFT M	∴	93	*Rahta dkarte* (ܪܗܛܐ ܕܟܪܬܗ)
SHIFT ,	<	60	*Soora min* (ܙܥܘܪܐ ܡܢ)
SHIFT .	>	62	*Goora min* (ܓܘܪܐ ܡܢ)
SHIFT /	؟	63	*Neeshanqa dshooala* (ܢܝܫܢܩܐ ܕܫܘܐܠܐ)



ASKL
Assyrian Standard Keyboard Layout

Note the following equivalences	Eastern	=	Western
ܙܩܦܐ	˙		◌ܳ
ܦܬܚܐ	:		◌ܰ
ܙܠܡܐ ܦܫܝܩܐ	.		◌ܶ
ܚܒܨܐ	.		◌ܺ
ܪܘܨܐ	˙		◌ܽ
ܙܠܡܐ ܩܫܝܐ	-		none

Appendix C

Contextual Analysis

Contextual Analysis is a development made possible by the computer. Very simply put, Contextual Analysis is the ability of the computer to automatically place the correct shape of a letter into a word. For example, the word ܡܪܝܡ requires the following Contextual Analysis:

space	ܡ ܪ ܝ ܡ	Key pressed
space	ܡ ܪ ܝ ܡ	Computer shows

After pressing space, the computer changes the final ܡ in the word to a ܡ. Thus all a typist needs to type is one letter and the computer determines which shape of that letter to place in the word; this means that there would be only 22 letter keys on the Assyrian keyboard.

The following is a basic algorithm for contextual analysis. Note, this algorithm does not include support for font-specific rendering (see Appendix E).

```
Step 1   Get keystroke
     Is it a space?
       Yes  (a space was typed)
         Beginning of the document?
           No (not beginning of document)
             Is previous character a letter?
               Yes (it's a letter)
                 Is letter preceded by a space?
                   Yes (preceded by a space)
                     Put free form
                   No (not preceded by a space)
                     Change it to final form
         Put space (type the keystroke)
       No (something other than space was typed)
         Is keystroke a letter?
           Yes
             Beginning of the document or previous character a space?
               Yes (beginning of document or space)
                 Put initial form
               No (not beginning of document or space)
                 Put middle form
           No
             Type the keystroke
     Go to step 1
```

Appendix D

Font Standard

An Assyrian font must define, at a minimum, the character set defined by SACII (Appendix A, codes 32-196). While the shape of each character will differ from font to font, the identity of the character will remain the same.

Every Assyrian font must have four forms for each letter

1.	free	letter is not connected on either side.
2.	initial	letter is not connected on right side and is connected on the left side.
3.	middle	letter is connected on both sides.
4.	final	letter is connected on right side and is not connected on the left side.

Contextual Analysis will automatically place the correct form of the letter into the word. SACII defines the following codes for each of these forms.

Form	SACII code
Middle forms	65 - 87
Free forms	128 - 150
Initial forms	151 - 173
Final forms	174 - 196

Appendix E
DSM Specification for Eastern Assyrian

This appendix uses the dynamic storage model and the standards developed in Appendix A, Appendix C, and Appendix D to define the properties of the Eastern Assyrian font. The following properties are defined

- P1. The four shapes of each letter
- P2. Connection property of each character
- P3. Location property of each character
- P4. Cursor effect property of each character
- P5. Ligatures
- P6. Rendering rules

Contextual analysis is font specific. Eastern Assyrian, Western Assyrian, and *Estrangelo* require different rules of rendering and different ligatures (P5 and P6). SACII supports contextual analysis by reserving codes for the free, initial, middle, and final forms of each letter (P1 and P2). These codes provide a standard, font independent method of rendering the three major Assyrian fonts. There are, however, differences in the fonts which are not encoded in SACII, and which must be handled algorithmically. These rendering rules (P5 and P6) must be specified for each Assyrian font.

The following is the specification for Eastern Assyrian. Specifications for Western Assyrian and *Estrangelo* remain to be developed.

The following table defines the first five properties, P1-P5, of Eastern Assyrian.

SACII	Symbol	Connections Left, Right None	Location Property Previous, Current Next	Cursor Effect Backward, None Forward
-------	----------	--------------	-------------------	------------------
32-35		N	C	F
36	**	N	P	N
37	%	N	C	F
38	:	N	P	N
39-41		N	C	F
42	:	N	P	N
43-63		N	C	F
64	-	N	P	N
65	ܐ	R	C	F (middle forms, 65-87)
66	ܒ	RL	C	F
67	ܓ	RL	C	F
68	ܕ	R	C	F
69	ܗ	R	C	F
70	ܘ	R	C	F
71	ܙ	R	C	F
72	ܚ	RL	C	F
73	ܛ	RL	C	F
74	ܝ	RL	C	F
75	ܟ	RL	C	F

76	[illegible]	RL	C	F
77	[illegible]	RL	C	F
78	[illegible]	RL	C	F
79	[illegible]	RL	C	F
80	[illegible]	RL	C	F
81	[illegible]	RL	C	F
82	[illegible]	R	C	F
83	[illegible]	RL	C	F
84	[illegible]	R	C	F
85	[illegible]	RL	C	F
86	[illegible]	R	C	F
87	reserved			
88	•	N	P	N
89	•	N	P	N
90	•	N	P	N
91	•	N	P	N
92	•	N	P	N
93	∴	N	P	N
94	^	N	C	F
95	.	RL	C	F
96	†	N	P	N
97	ˋ	N	P	N
98	:	N	P	N
99	.	N	P	N
100	,	N	P	N
101	.	N	P	N
102		N	P	N
103	.	N	P	N
104	¨	N	P	N
105	^	N	P	N
106	ˇ	N	P	N
107	-	N	P	N
108	-	N	P	N
109	-	N	P	N
110	ˊ	N	P	N
111	⁛	N	C	F
112	⁚	N	C	F
113	⁘	N	C	F
114	†	N	C	F
115	\	N	P	N
116	/	N	P	N
117	...	N	C	F
118	\	N	P	N
119	׀	N	P	N
120	׀	N	P	N

121	[	N	C	F	
122	]	N	C	F	
123	unused				
124	.	R	P	N	
125	.	L	P	N	
126-127		N	C	N	
128	ܐ	N	P	N	(free forms, 128-150)
129	ܒ	N	P	N	
130	ܓ	N	P	N	
131	ܕ	N	P	N	
132	ܗ	N	P	N	
133	ܘ	N	P	N	
134	ܙ	N	P	N	
135	ܚ	N	P	N	
136	ܛ	N	P	N	
137	ܝ	N	P	N	
138	ܟ	N	P	N	
139	ܠ	N	P	N	
140	ܡ	N	P	N	
141	ܢ	N	P	N	
142	ܣ	N	P	N	
143	ܥ	N	P	N	
144	ܦ	N	P	N	
145	ܨ	N	P	N	
146	ܩ	N	P	N	
147	ܪ	N	P	N	
148	ܫ	N	P	N	
149	ܬ	N	P	N	
150	reserved				
151	ܐ	L	P	N	(initial forms, 151-173)
152	ܒ	L	P	N	
153	ܓ	L	P	N	
154	ܕ	L	P	N	
155	ܗ	L	P	N	
156	ܘ	L	P	N	
157	ܙ	L	P	N	
158	ܚ	L	P	N	
159	ܛ	L	P	N	
160	ܝ	L	P	N	
161	ܟ	L	P	N	
162	ܠ	L	P	N	
163	ܡ	L	P	N	
164	ܢ	L	P	N	

165	ܣ	L	P	N
166	ܥ	L	P	N
167	ܦ	L	P	N
168	ܨ	L	P	N
169	ܩ	L	P	N
170	ܪ	L	P	N
171	ܫ	L	P	N
172	ܬ	L	P	N
173	reserved			
174	ܐ	R	P	N (final forms, 174-196)
175	ܒ	R	P	N
176	ܓ	R	P	N
177	ܕ	R	P	N
178	ܗ	R	P	N
179	ܘ	R	P	N
180	ܙ	R	P	N
181	ܚ	R	P	N
182	ܛ	R	P	N
183	ܝ	R	P	N
184	ܟ	R	P	N
185	ܠ	R	P	N
186	ܡ	R	P	N
187	ܢ	R	P	N
188	ܣ	R	P	N
189	ܥ	R	P	N
190	ܦ	R	P	N
191	ܨ	R	P	N
192	ܩ	R	P	N
193	ܪ	R	P	N
194	ܫ	R	P	N
195	ܬ	R	P	N
196	reserved			
197	[illegible]	ligature for [illegible]		
198	[illegible]	ligature for ܠܐ		
199	[illegible]	ligature for ܠܐ		
200	[illegible]	ligature for [illegible]		
201-255		unused		

Rendering rules (P5, P6)

A word begins with a space and ends with a space, and cannot contain a space.

If a user inserts the suspend rendering code (127) before a character then that character is printed as is, without special rendering -- the following rules would not apply.

Rules of rendering

R1 Left and right tails are attached to a letter that is preceded or followed by a space, or both. Letters which accept a right tail are: ܚ ܟ ܢ ܫ . Letters which accept a left tail are: ܒ ܓ ܚ ܠ ܢ ܡ ܠ ܣ ܝ ܦ ܟ ܩ ܫ ܥ .

R2 If ܬ appears at the end of a word and is not preceded by ܕ or ܪ, then ܬ is replaced by ܬ.

R3 If ܟ appears at the end of a word and is preceded by a letter that does not connect on its left, then ܟ is replaced by ܟ.

R4 If ܟ appears at the end of a word and is preceded by a letter that connects on its left, then ܟ is replaced by ܟ.

R5 If ܢ appears at the end of a word and is preceded by a letter that does not connect on its left, then ܢ is replaced by ܢ.

R6 If ܢ appears at the end of a word and is preceded by a letter that connects on its left, then ܢ is replaced by ܢ.

R7 If ܠܬ appear at the end of a word then ܠܬ are replaced by the ligature ܠܬ or ܠܬ, depending on a default set by the user.

R8 If the word ܗܘ appears then it is replaced by the ligature ܗܘ .