

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 with 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

𐎶

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 𐎶𐎵𐎶𐎵

Glyph	Output
66::98	𐎶
74::0	𐎵
86::97	𐎶
65::0	𐎶

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 about 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 𐎶𐎵𐎶𐎵, *Gamal* has two diacritical marks. The glyph object structure, however, can only store one. SSM fails in

A Dynamic Storage Model For Assyrian Computer Text
Peter BetBasoo

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 diacritical 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

A Dynamic Storage Model For Assyrian Computer Text
Peter BetBasoo

	Location Property	Cursor Effect
<i>Ptakha</i>	previous position	no motion
<i>Zqapa</i>	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 $\mathbf{\text{ܐܠܟܝܢ}}$, is stored as follows

Glyph	Output
85	ܐ
70	ܠ
103	ܝ
67	ܟ
103	ܝ
110	ܢ
76	ܐ
97	ܝ
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

A Dynamic Storage Model For Assyrian Computer Text
Peter BetBasoo

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.
- DeKelaïta, 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 (*Serio*), 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.

40	(Qishia simmalatia (ܩܝܫܝܐ ܣܝܡܡܠܬܝܐ)
41)	Qishia yamecaita (ܩܝܫܝܐ ܝܡܥܝܬܝܐ)
42	:	Makspana (ܡܟܣܦܢܐ)
43	+	Mazyiddana (ܡܙܝܝܕܢܐ)
44	*	Neeshangq dnoohara (ܢܝܫܢܟܐ ܕܢܘܘܚܪܐ)
45	-	Mapserana (ܡܦܣܪܢܐ)
46	.	Pasoqa (ܦܫܘܩܐ)
47	/	Palce'ana (ܦܠܥܝܢܐ)
48	0	Saeqar (ܣܝܩܐܪ)
49	1	Kha (ܟܗ)
50	2	Tre (ܦܚܝܢ)
51	3	Tlata (ܦܠܬܐ)
52	4	Arb'a (ܦܘܪܒܐ)
53	5	Khamsha (ܦܚܡܫܐ)
54	6	Ishia (ܦܫܝܐ)
55	7	Shaw'a (ܫܘܘܥܐ)
56	8	Imanya (ܦܝܡܢܝܐ)
57	9	Tish'a (ܦܝܫܬܐ)
58	:	Zawga (ܘܘܓܐ)
59	!	Pasoqa kirya (ܦܫܘܩܐ ܕܟܝܪܝܐ)
60	<	Soora min (ܣܘܘܪܐ ܡܝܢ)
61	=	Dma (ܕܡܐ)
62	>	Goora min (ܘܘܪܐ ܡܝܢ)
63	?	Neeshangq dshooola (ܢܝܫܢܟܐ ܕܫܘܘܠܐ)
64	~	Napsa (ܢܦܫܐ)
65	⌈ (ܦܚܝܢ) ⌋	Allap
66	⌈ (ܦܚܝܢ) ⌋	Bet
67	⌈ (ܦܚܝܢ) ⌋	Ganunal
68	⌈ (ܦܚܝܢ) ⌋	Dallai

A Dynamic Storage Model For Assyrian Computer Text
 Peter BetBasoo

69	𐎠 (𐎠 𐎠)	Ilea	90	Msha'tana (𐎠𐎠𐎠𐎠)
70	𐎡 (𐎡 𐎡)	Wow	91	Samka (𐎠𐎠𐎠)
71	𐎢 (𐎢 𐎢)	Zen	92	Mnakhia (𐎠𐎠𐎠𐎠)
72	𐎣 (𐎣 𐎣)	Kheih	93	Rahia dkarie (𐎠𐎠𐎠𐎠𐎠𐎠𐎠𐎠)
73	𐎤 (𐎤 𐎤)	Telh	94	Marmana (𐎠𐎠𐎠𐎠)
74	𐎥 (𐎥 𐎥)	Yood	95	Seria dkhoyada (𐎠𐎠𐎠𐎠𐎠𐎠𐎠𐎠)
75	𐎦 (𐎦 𐎦)	Kap	96	?
76	𐎧 (𐎧 𐎧)	Lamnad	97	Zqapa (𐎠𐎠𐎠, Western)
77	𐎨 (𐎨 𐎨)	Meem	98	Pakha (𐎠𐎠𐎠 Western)
78	𐎩 (𐎩 𐎩)	Noon	99	Zlame pshege (𐎠𐎠𐎠𐎠𐎠𐎠)
79	𐎪 (𐎪 𐎪)	Sinkei	100	Zlame qishye (𐎠𐎠𐎠𐎠𐎠, Western)
80	𐎫 (𐎫 𐎫)	'e	101	Khwasa (𐎠𐎠𐎠𐎠 Western)
81	𐎬 (𐎬 𐎬)	Pe	102	Rwakha (𐎠𐎠𐎠𐎠 Western)
82	𐎭 (𐎭 𐎭)	Sade	103	Rwasa (𐎠𐎠𐎠𐎠𐎠, 𐎠𐎠𐎠𐎠 in Eastern and Western)
83	𐎮 (𐎮 𐎮)	Qop	104	Syame (𐎠𐎠𐎠)
84	𐎯 (𐎯 𐎯)	Resh	105	Rinkha (𐎠𐎠𐎠𐎠 below letter)
85	𐎰 (𐎰 𐎰)	Sheen	106	Qishia (𐎠𐎠𐎠𐎠 below letter)
86	𐎱 (𐎱 𐎱)	Tow	107	Majlecyana (𐎠𐎠𐎠𐎠 below letter)
87	reserved for 23rd Mandiac letter		108	Seria khiera (𐎠𐎠𐎠𐎠𐎠 below letter)
88	Mai'ana (𐎠𐎠𐎠𐎠)		109	Seria 'elayia (𐎠𐎠𐎠𐎠 above letter)
89	Rima (𐎠𐎠𐎠𐎠)		110	Talqana (𐎠𐎠𐎠𐎠𐎠 for silent or accented letters)
			111	'elaye (𐎠𐎠𐎠 rests on either side of letter)
			112	Takhtaye (𐎠𐎠𐎠𐎠 rests on either side of letter)
			113	Kikhwa (𐎠𐎠𐎠𐎠)
			114	Sleeva (𐎠𐎠𐎠𐎠)
			115	Mhagyana 2 (𐎠𐎠𐎠𐎠 below letter)
			116	Mhagyana (𐎠𐎠𐎠𐎠 below letter)
			117	Gneezi (𐎠𐎠𐎠𐎠)
			118	Stoona goonya (𐎠𐎠𐎠𐎠𐎠)

A Dynamic Storage Model For Assyrian Computer Text

Peter BetBasoo

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	Ⲁ	49	<i>Kha</i> (Ⲁ)
2	ⲁ	50	<i>Tre</i> (ⲁ)
3	Ⲃ	51	<i>Tiata</i> (Ⲃ)
4	ⲃ	52	<i>Arb'a</i> (ⲃ)
5	Ⲅ	53	<i>Khamsha</i> (Ⲅ)
6	ⲅ	54	<i>Ishia</i> (ⲅ)
7	Ⲇ	55	<i>Shaw'a</i> (Ⲇ)
8	ⲇ	56	<i>Tmanya</i> (ⲇ)
9	Ⲉ	57	<i>Tish'a</i> (Ⲉ)
0	ⲉ	48	<i>Seeper</i> (ⲉ)
.	Ⲋ	45	<i>Mapserana</i> (Ⲋ)
=	ⲋ	61	<i>Dna</i> (ⲋ)
Q	Ⲍ	83	<i>Qop</i>
W	ⲍ	80	ⲍ
E	Ⲏ	101	<i>Khwasa</i> (Ⲏ) Western Ⲏ
R	ⲏ	100	<i>Zlame qishye</i> (ⲏ) Western ⲏ

- 119 Ⲑ *Stoona khaya* (Ⲑ)
- 120 ⲑ *Stoona 'elaya* (ⲑ)
- 121 [left bracket
- 122] right bracket
- 123 unused
- 124 Ⲓ *Toopra simmalaya* (Ⲓ)
- 125 ⲓ *Toopra yameenaya* (ⲓ)
- 126 Ⲕ *Sharree malkhamia* (Ⲕ)
- 127 ⲕ *Maktee malkhamia* (ⲕ)
- 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)

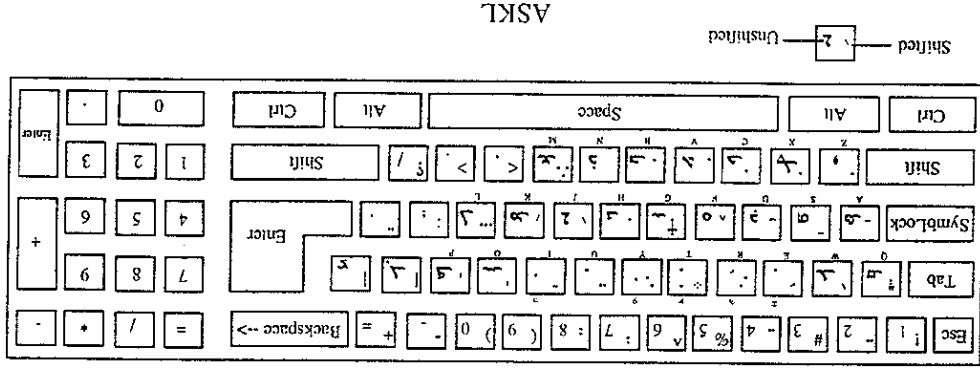
Codes 101 and 103 require clarification. In Eastern Assyrian Ⲏ = ee (as in sheet), in Western Assyrian it is the diacritical mark Ⲏ which is ee. As far as the computer is concerned, Ⲏ is just a letter with a dot under it, as are ⲏ Ⲑ ⲑ Ⲓ. One key, therefore, serves all these purposes. However, when one switches to a Western font the Ⲏ becomes a Ⲏ, and there no longer is a key for Ⲏ, 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 Ⲍ, is not desired -- code 127 would force the software to show Ⲍ in this case.

A Dynamic Storage Model For Assyrian Computer Text

Peter BetBasoo

SHIFT L	..	117	Greezi (ܓܪܝܘܘܘܬܐ)
SHIFT :	:	58	Zawga (ܙܘܘܓܐ)
SHIFT '	"	34	Manrana (ܡܢܪܢܐ)
SHIFT Z	.	88	Mzi'ana (ܡܙܝܥܢܐ)
SHIFT X	.	89	Ritma (ܪܝܬܡܐ)
SHIFT C	.	90	Msha'hana (ܡܫܗܢܐ)
SHIFT V	.	91	Samka (ܣܡܟܐ)
SHIFT B	.	92	Mnakhta (ܡܢܚܬܐ)
SHIFT N	.		
SHIFT M	:	93	Rakha dkarie (ܪܚܝܩܐ ܕܟܪܝܐ)
SHIFT ,	<	60	Soora min (ܣܘܪܐ ܡܝܢ)
SHIFT .	>	62	Goora min (ܓܘܪܐ ܡܝܢ)
SHIFT /	!	63	Neeshangpa dshooola (ܢܝܫܢܗܘܩܦܐ ܕܫܘܘܠܐ)



Assyrian Standard Keyboard Layout

Note the following equivalences Eastern = Western

ܡܫܝܘܩܐ	.	ܡܫܝܘܩܐ
ܠܝܫܘܩܐ	.	ܠܝܫܘܩܐ
ܡܫܝܘܩܐ	.	ܡܫܝܘܩܐ
ܠܝܫܘܩܐ	.	ܠܝܫܘܩܐ
ܡܫܝܘܩܐ	.	ܡܫܝܘܩܐ
ܠܝܫܘܩܐ	.	ܠܝܫܘܩܐ

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 **𐎶𐎵𐎶** requires the following Contextual Analysis:

space **𐎶** **𐎵** **𐎶** Key pressed
space **𐎶** **𐎵** **𐎶** Computer shows

After pressing space, the computer changes the final **𐎶** in the word to **𐎶**. 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)
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 SSCII (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. SSCII defines the following codes for each of these forms.

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

A Dynamic Storage Model For Assyrian Computer Text

Peter BetBasoo

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). SACH 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 SACH, 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.

SACH Symbol	Connections		Location Property		Cursor Effect	
	Left, Right	None	Previous, Current	Next	Backward, None	Forward
32-35	N	N	C	C	F	F
36	N	N	P	P	N	N
37	N	N	C	C	F	F
38	N	N	P	P	N	N
39-41	N	N	C	C	F	F
42	N	N	P	P	N	N
43-63	N	N	C	C	F	F
64	N	N	P	P	N	N
65	R	R	C	C	F (middle forms, 65-87)	F
66	RL	RL	C	C	F	F
67	RL	RL	C	C	F	F
68	R	R	C	C	F	F
69	R	R	C	C	F	F
70	R	R	C	C	F	F
71	R	R	C	C	F	F
72	RL	RL	C	C	F	F
73	RL	RL	C	C	F	F
74	RL	RL	C	C	F	F
75	RL	RL	C	C	F	F

76	Λ	RL	C	F	N
77	⋈	RL	C	F	N
78	⋈	RL	C	F	N
79	⋈	RL	C	F	N
80	⋈	RL	C	F	N
81	⋈	RL	C	F	N
82	⋈	R	C	F	N
83	⋈	RL	C	F	N
84	⋈	R	C	F	N
85	⋈	RL	C	F	N
86	⋈	R	C	F	N
87	reserved				
88	⋈	N	P	F	N
89	⋈	N	P	F	N
90	⋈	N	P	F	N
91	⋈	N	P	F	N
92	⋈	N	P	F	N
93	⋈	N	P	F	N
94	⋈	N	P	F	N
95	⋈	RL	C	F	N
96	⋈	N	P	F	N
97	⋈	N	P	F	N
98	⋈	N	P	F	N
99	⋈	N	P	F	N
100	⋈	N	P	F	N
101	⋈	N	P	F	N
102	⋈	N	P	F	N
103	⋈	N	P	F	N
104	⋈	N	P	F	N
105	⋈	N	P	F	N
106	⋈	N	P	F	N
107	⋈	N	P	F	N
108	⋈	N	P	F	N
109	⋈	N	P	F	N
110	⋈	N	P	F	N
111	⋈	N	P	F	N
112	⋈	N	P	F	N
113	⋈	N	P	F	N
114	⋈	N	P	F	N
115	⋈	N	P	F	N
116	⋈	N	P	F	N
117	⋈	N	P	F	N
118	⋈	N	P	F	N
119	⋈	N	P	F	N
120	⋈	N	P	F	N

A Dynamic Storage Model For Assyrian Computer Text

Peter BetBasoo

121	I	N	C	F	165	𐎠	L	P	N
122	J	N	C	F	166	𐎡	L	P	N
123	unused	R	P	N	167	𐎢	L	P	N
124	.	L	P	N	168	𐎣	L	P	N
125	.	N	P	N	169	𐎤	L	P	N
126-127		N	C	N	170	𐎥	L	P	N
128	𐎠	N	P	N	171	𐎦	L	P	N
129	𐎡	N	P	N	172	𐎧	L	P	N
130	𐎣	N	P	N	173	reserved	L	P	N
131	𐎤	N	P	N	174	𐎨	R	P	N (final forms, 174-196)
132	𐎥	N	P	N	175	𐎩	R	P	N
133	𐎦	N	P	N	176	𐎪	R	P	N
134	𐎧	N	P	N	177	𐎫	R	P	N
135	𐎨	N	P	N	178	𐎬	R	P	N
136	𐎩	N	P	N	179	𐎭	R	P	N
137	𐎪	N	P	N	180	𐎮	R	P	N
138	𐎫	N	P	N	181	𐎯	R	P	N
139	𐎬	N	P	N	182	𐎰	R	P	N
140	𐎭	N	P	N	183	𐎱	R	P	N
141	𐎮	N	P	N	184	𐎲	R	P	N
142	𐎯	N	P	N	185	𐎳	R	P	N
143	𐎰	N	P	N	186	𐎴	R	P	N
144	𐎱	N	P	N	187	𐎵	R	P	N
145	𐎲	N	P	N	188	𐎶	R	P	N
146	𐎳	N	P	N	189	𐎷	R	P	N
147	𐎴	N	P	N	190	𐎸	R	P	N
148	𐎵	N	P	N	191	𐎹	R	P	N
149	𐎶	N	P	N	192	𐎺	R	P	N
150	reserved	N	P	N	193	𐎻	R	P	N
151	𐎠	N (initial forms, 151-173)	P	N	194	𐎼	R	P	N
152	𐎡	N	P	N	195	𐎽	R	P	N
153	𐎣	N	P	N	196	reserved	R	P	N
154	𐎤	N	P	N	197	𐎿	ligature for 𐎶𐎺		
155	𐎥	N	P	N	198	𐎻	ligature for 𐎺𐎻		
156	𐎦	N	P	N	199	𐎼	ligature for 𐎻𐎼		
157	𐎧	N	P	N	200	𐎽	ligature for 𐎼𐎽		
158	𐎨	N	P	N	201-255	unused	unused		
159	𐎩	N	P	N					
160	𐎪	N	P	N					
161	𐎫	N	P	N					
162	𐎬	N	P	N					
163	𐎭	N	P	N					
164	𐎮	N	P	N					

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: 𐎠 𐎡 𐎢 𐎣. Letters which accept a left tail are:

𐎤 𐎥 𐎦 𐎧 𐎨 𐎩 𐎪 𐎫 𐎬 𐎭 𐎮 𐎯 𐎰 𐎱 𐎲 𐎳 𐎴 𐎵 𐎶 𐎷 𐎸 𐎹 𐎺 𐎻 𐎼 𐎽 𐎾 𐎿.

R2 If 𐎠 appears at the end of a word and is not preceded by 𐎡 or 𐎢, then 𐎠 is replaced by 𐎠.

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

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

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

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

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

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