A SANSKRIT USER INTERFACE

by

DAVID GEORGE NOHLE

B.S. Ohio State University, 1979

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements of the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1984

Approved by:

Rodney M. Bates
Major Professor

CONTENTS

1. **Introduction**

This paper describes a master's project involved with the Grammar Engine*, an experiment in natural language understanding with the Sanskrit language. The author developed a user interface that has already been used in word processing applications and is intended to be the user interface for the Grammar Engine system.

The Sanskrit language and its script, Devanagri, pose some interesting user interface and translation problems. The ability to use standard English keyboards to input Devanagri was desired. Normally, Devanagri syllables containing multiple consonants and a vowel are written with one symbol and it is desirable to enter them as separate alphabetic characters. This user interface was designed to allow its use for a variety of input character sets and any output character set and is presently capable of allowing for printing of any output character set whose characters can be depicted with an 11x7 dot matrix. Phonetic, numeric and keyboard input character sets are able to be printed presently. Although hardware is not available to allow printing of the complex Devanagri characters, the

---

* Grammar Engine is a registered trademark of Grammar Engine Inc.

translation to a numeric print code has been achieved. Translation by combining Devanagri characters into numeric codes that represent syllables has been planned for and is expected to be developed with little additional effort.

Chapter 1 is this introduction. Chapter 2 contains a description of the entire Sanskrit Grammar Engine project and the user interface portion of it developed by the author. In order to do this properly, Sanskrit and the Devanagri script will be described and the problems associated with their translation outlined. Chapter 3 outlines the requirements and design of the user interface. Chapter 4 relates the development history and describes the adapting of the software for use in a word processor to print drafts of books containing Sanskrit text. Chapter 5 concludes the paper with a description of the current status of the work and the results obtained. Further work planned for the project is outlined and related work in this area is discussed.

A list of references, the figures and the attachments follow chapter 5.

2. Description of the Problem

2.1 Sanskrit

Sanskrit is an ancient, Asian Indian language that originally was a commonly spoken language several thousand years ago. In this period, scholars realized that languages evolved and they wished to preserve the true meaning of important religious works written in Sanskrit. Thus, the language was studied extensively and many grammars were produced, debated and refined. Panini wrote the definitive Sanskrit grammar [1] about 2,000 years ago. This was recognized as a remarkable piece of work and efforts to produce other grammars ceased while attention was turned to the study of Panini's grammar. Sanskrit means "made smooth" or "polished" and Panini's grammar is considered to be a complete description of this structured and refined language.

Panini's grammar has been taught to young Indian boys by having them memorize the 35 or so pages of 4,000 sutras (or rules) by the time they are 10 years old. The grammar is still studied this way today.

The grammar is essentially a set of 4,000 rules divided into 8 books of 4 chapters each that is known as the "sutrapath". In Sanskrit, words are not pre-ordained, they are derived by applying the rules to a set of "2000 roots ("dhatupath") and

some associated lists ("ganapath"). This is the type of regularity that contributes to the preservation and standardization of the language.

Many treatises [2][3][7] have been written about the grammar. Panini is recognized as having used all of the techniques identified by Noam Chomsky (recognized as the greatest linguist of our time) in the production of this grammar. Some of these techniques provide means of incorporating semantics into grammatical descriptions and afford the grammar a depth unparalleled in any other natural language.

2.2 Devanagri Script

Sanskrit is written in the Devanagri script. The basic Devanagri alphabet with the phonetic representation of the pronunciation is given in Figure 1 [4][5][6]. This script is oriented towards consonants and syllables. Syllable symbols basically consist of a symbol for a consonant or group of consonants with a vowel marking appended to the top, bottom, or either side. Symbols for consonant groups are called compound consonants and typically look like a shorthand version of the separate consonants. Consonant groups are sometimes written separately, but compounding is preferred. Hundreds of unique syllable symbols are possible.

The shaping of Devanagri symbols reflect the type of writing implements that were traditionally used to write them. The symbols have not evolved for use with printing presses as Roman symbols have. Although some typewriters for printing Devanagri script have been fabricated, computer terminals are not known to be available.

A basic alphabet consisting of 49 symbols and a few diacratic marks exists and has been used to type Devanagri script, after a fashion, thus avoiding the problem of printing hundreds of unique symbols of different sizes. When this is done, even the vowels are presented as separate forms.

2.3 The Grammar Engine Project

Grammar Engine Inc. was founded in 1982 by Arvind Rana, a computer scientist, and his father, Jag Deva Singh, a professor of linguistics and Sanskrit scholar, with the express purpose of experimenting with Sanskrit and computing. A long term project was envisioned to take advantage of the extraordinary structure of Panini's grammar in implementing Sanskrit on a computer system. They saw the S-nskrit language as having a concise, static, complete and therefore less ambiguous, more manageable description than other natural languages. From this attempt, insight into the nature of language and the process of making intelligent

machines would be obtained.

The computer implementation of Panini's grammar (the Grammar Engine) was to be approached from two directions:

1. A linguistic database of rules was to be accumulated with some type of classifications or labels placed on the data items. Essentially, this process would involve entering and manipulating the text of the grammar. The classifications (ex. rule, definition, exception, etc.) would evolve as more knowledge was accumulated. This would result in a rule base that might later prove useful.

2. Parts of rules and eventually sections of the grammar were to be modeled in order to experiment with different knowledge representation strategies. This process would first involve studying the structure of the grammar in order to infer how Panini devised it. Once models were constructed, they could be checked for consistency and perhaps tested on text.

In the beginning stages of the project, logistical problems of working with Sanskrit and Devanagri script would be tackled. The logistical problems were mainly associated with input and output.

The first piece of software needed for the project was a

user interface for Sanskrit/Devanagri input and output so that files containing rules and other text could be entered and dealt with effectively. Most of the remainder of this paper focuses on the author's development of a flexible user interface designed to function as part of the Grammar Engine and later adapted to other applications.

3. User Interface

3.1 Requirements

Roman script based languages use the same set of symbols for both input and output. Devanagri script has far too many unique symbols to put them on a normal keyboard as single keys. Oriental languages have similar problems and the solutions have ended up being awkward and expensive.

Input could be managed with the simple 49 alphabetic character symbols, but many more symbols, approximately 300, are required for high quality output. Limiting input to the 49 simple alphabetic characters can be considered as a viable solution because Devanagri typewriters only have these characters. The input symbols could be translated in a computer system by grouping them into syllables and replacing them with complex symbols to form all possible output symbols.

If an English keyboard was to be used, a way to print inputted text back out as the actual keystrokes* entered

---

* Throughout this paper, the term "keystroke" will be used to indicate the ASCII code which stands for the Roman character pictured on a given key on the keyboard. When a "keystroke" is viewed with an editor after being stored in a file, it will, of course, be again in the form of a symbol and not a number.

might be useful. Indeed, if the translator were made general enough, printing of phonetic and internal numeric representations might also be permitted.

An input encoding was desired that would allow normal English keyboards to be used immediately, as Devanagri keyboards are not available. It was desired to be able to output Devanagri symbols on both video terminals and line printers, although suitable hardware might not be purchased for some time. In the mean time, simple terminals and dot matrix printers that were at hand were to be used and the design and software made so that it could be extended for better hardware.

3.2 **Design**

3.2.1 High Level Design This section will present the high level design of the User Interface Subsystem by examining its inputs and outputs. The keyboard input language is the single type of input and printing instructions are the output.

3.2.1.1 Selection of the Keyboard input language for English keyboards Keycaps were not going to be replaced on the terminals, so a keyboard input language was designed in such a way as to strike a balance between human factors considerations and ease of translation. Selection of keystrokes was done to allow persons familiar with Roman

characters and their English pronunciations and Devanagri characters and their Sanskrit pronunciations to relate one with the other. Control characters were not used for entering Sanskrit as they were needed for their normal functions. Some of the 49 input characters would be required to be entered with multiple keystrokes to allow full use of all control characters.

In an input encoding made up of one and two character codes, the "first set" could be considered to be those characters allowed to be alone or to stand as the first of a two character code. The "follow set" could then be considered to be those character codes allowed to be second in a two character string. By selecting a "follow set" disjoint from a "first set" for input tokens representing the alphabetic characters, the program to translate between input keystrokes and output printing codes could be built as a simple finite state machine [8]. Pronunciations are frequently represented with Roman characters by linguists. This was taken advantage of in choosing the input keyboard keystrokes that were to represent alphabetic characters. A small "follow set" consisting of the "h" and "~" characters was settled on and capitals were used to represent some symbols like long vowels. It was recognized that after using the keyboard awhile, users might like some changes made in the input keystrokes (within limits), so a mechanism

to permit this was desired.

The input codes are given in figure 2. This table has three columns:

- the phonetic representation
- the Devanagri script symbol
- the keyboard input keystrokes

3.2.1.2 Selection of output codes Because of the dependence on printer hardware, it was decided to have a numeric code that would eventually be looked up in a table where an associated sequence of printing instructions would be located. The codes themselves are arbitrary (except that the symbols in the input domain need the same encoding as in the output domain).

The needed table would have the following two columns and it will be given later in this document as part of another table.

- the sequence of printing instructions
- the numeric code

3.2.2 Detailed Design of User Interface Subsystem The detailed design of the User Interface Subsystem will be discussed in this section by first defining the central data

structure involved, the internal numeric codes. The subsystem will then be broken down into functional pieces (in the same way that it was designed) and each piece (or module) described in detail.

3.2.2.1 Selection of Internal Numeric Codes Like any computer language, Sanskrit was expected to have an internal numeric representation. As described above, 49 Devanagri characters were to comprise the input domain and required a corresponding number of numeric codes. This set was selected for use as the internal encoding for the Grammar Engine. Output (printing) codes were to be produced by the user interface from translation of the internal numeric codes as needed.

The encoding scheme is given in figure 3. This table has three columns:

- the numeric code (which might change based on advantages that certain encodings might have for other subsystems in the Grammar Engine)
- the Devanagri script symbol
- the phonetic representation

3.2.2.2 Incoming Translations The incoming translation was always from keyboard type input to internal numeric code and was made straightforward by the simplification of the

"follow set".

3.2.2.3 Outgoing Translations Outgoing Translations would always be from numeric code to a set of printing or displaying instructions. This makes it heavily dependent on the hardware. To allow for the eventual use of various devices, the output codes table described above would eventually be expanded to hold these sequences of printing or displaying instructions. (Note that different tables would exist for different devices.) At present, it holds a one character print instruction for the reasons outlined below.

An Okidata Microline 92 Serial Dot Matrix Printer [6] was initially available for use with the Grammar Engine project. The printhead allowed up to 64 11x7 dot matrix symbols to be downline loaded and printed via host software. In this context, downline loading refers to the process of sending information to the printer that programs it as to how to print an alternate character set. This process is accomplished by sending a sequence of codes that include a normal ASCII code and eleven numbers, each representing a column in the dot matrix. The ASCII code is then sent to the printer when it is in the "print alternate character set" mode to request printing of the downline loaded symbol. A full dot addressing (graphics) mode is also available in the device. As characters of the same size as the normal

Roman fonts were desired, the former capability was selected to be the basis for the initial implementation. Extension to other printers and printing methods was to be a consideration in the software design.

Functions were defined to allow:

- fabrication of the output symbols by users who knew the scripts and were not computer programmers
- translation to printer command sequences via commands
- downline loading of these symbols via commands

With several hundred symbols needed for Devanagri output and only 64 allowable downline loaded characters, software would be required to change the loaded characters occasionally to make them all available as needed.

In examining some of the symbols of even average complexity, it was realized that an 11x7 matrix was not nearly fine-grained enough to allow recognizable representations of many of the characters in the basic alphabet, much less the far more complex compound characters. It was then clear that eventually a better printer would be needed to print normal size Devanagri symbols.

Thus, the initial implementation would involve printing of one character symbols that could be defined with an 11x7

matrix and subsequent implementations would allow more complex printing instructions either by:

- sequences of characters and backspaces to allow overprinting of several primitives to form one symbol,
- use of XY addressable graphics capabilities,
- or a printer with more dots on the printhead that could print more complex characters.

It was recognized that different fonts or the use of different, but equivalent syllable symbols might cause changes to be made in this encoding.

3.2.2.4 Subsystem Decomposition Once the input, output and central data structures had been defined, an overall picture of the inside of the subsystem could be generated. The printing symbols would include all of the input symbols as well as several hundred combination symbols. It seemed reasonable to eventually derive one master encoding that included all numeric codes and use it for both purposes. The following activities needed to be integrated into the subsystem:

1. User defined pictures (from normal editable files) must be translated into sequences suitable to command the printer as to how to print the user definable symbols. (CSBUILDER)

2. The printer command sequences must be sent to the printer (downline load the character definitions). (CSLOADER)

3. The alternate character set must be demonstrated for debugging purposes. (CSDEMO)

4. Keyboard input must be parsed and translated into the central alphabetic numeric code. (XALL)

5. The simple alphabetic numeric code must be translated into complex numeric code. If printing of complex symbols was not desired, then this step could be skipped. (STOC)

6. Numeric code must be translated into some print codes for some type of output (numeric, keyboard, phonetic, simple alphabetic Devanagri or complex Devanagri with combined syllable symbols). (PALL)

Figure 4 depicts a subsystem decomposition that has all language specific data outside of the modules and in user changeable files.

These data files are the interfaces between modules and are of the following types:

1. Character Picture Files -

contain user-editable, character per-dot-representations of print symbols with an associated ASCII character must be unique and correlate with the master translation table. An example is given in Attachment I.

2. Loadable Numeric Files -

contain the output from translation of the character picture files which are the associated ASCII character and numbers that must be sent to the printer to downline load (define) character in its memory. An example is given in Attachment II.

3. Numeric Code Files -

contain the version of what would correspond to ASCII for Roman script which are the internal representations of symbols. An example is given in Attachment III.

4. Master Translation Table - is a three column table mapping input keystrokes to internal code and also to an associated ASCII character used as a printing label that must correspond to that used in the character picture files. An example is given in Attachment IV.

5. Keyboard Input Files -

contain keystrokes as they would entered at the terminal in an editable format. An example is given in Attachment V.

3.2.2.5 Module Descriptions The development strategy was to devise each module as a separate program that used files as input and output. This was to make the testing easier and allow pieces to be incorporated into other software in a variety of ways. The development of the simple to complex numeric code translator (STOC) was postponed until acquisition of a printer capable of printing complex symbols.

3.2.2.5.1 CSBUILDER The Character Set Builder performs the following steps:

1. asks the user for the input and output filenames
2. reads the input file which contains character pictures and translates to loadable numeric codes
3. writes the loadable numeric codes into the output file

3.2.2.5.2 CSLOADER The Character Set Loader performs the following steps:

1. asks the user for the input filename
2. reads the input file which contains loadable numeric codes and translates to printer instructions to load

characters

3. writes the printer instructions to the printer

3.2.2.5.3 CSDEMO The Character Set Demonstrator performs the following steps:

1. asks the user for the input filename

2. reads the input file which contains loadable numeric codes and translates to printer instructions to print samples of the characters

3. writes the printer instructions to the printer

3.2.2.5.4 XALL The Translate All program performs the following steps:

1. asks the user for the master translation table, keyboard input and numeric code output filenames

2. reads and stores the master translation table file

3. reads the input file which contains the keystrokes to be translated and translates to numeric code by looking them up in the master translation table (a linear search is presently used, but this may change if more speed is required)

4. writes the numeric codes to the designated output file

3.2.2.5.5 PALL The Print All program performs the following steps:

1. asks the user for the master translation table and numeric code input filenames

2. reads and stores the master translation table file

3. reads the input file which contains numeric codes and translates to printer instructions to print the characters by looking them up in the master translation table (once again, a linear search was used and may change if more speed is required)

4. writes the printer instructions to the printer

4. Development History

4.1 Environment

An IBM Personal Computer [9][10] with a 10 megabyte hard disk and one floppy disk drive was made available for use with the user interface portion of the project. This system was directly connected to the Okidata printer and was set up to run IBM PC DOS operating system [12]. IBM Advanced BASIC [11] was the only high level language available and was therefore selected for use. A visual editor, VEDIT, was available to allow entry of user defined keystroke codes and printing matrices.

4.2 Development of Prototypes and Independent Software Modules

Experiments were conducted to try out the printer capabilities for downline loading of characters. Attachment VI contains pictures and code used in an experiment to print a picture of the Grammar Engine Inc. company logo with 12 downline loaded characters. This experiment was successful, but resulted in horizontal blank lines in the logo because the bottom (eighth) row of dots is never defined in downline loaded characters.

Modules were developed one by one and tested individually with handmade data and eventually with input that was output

from other modules.

4.3 Installation in a Word Processor

During the spring of 1983, Jag Deva Singh was finishing work on a book written mainly in English that frequently used Sanskrit examples. A way to use word processing for this book to produce a version to be sent to India for publication was desired. Although no printer capable of printing full Devanagri script was available to us, the Indian publisher could typeset the book if the Devanagri portions were represented in phonetic symbols.

This application sized up as the perfect place to try out the software. Integration of the individual modules into this software was accomplished by removing the instructions to request and read file names from the user and changing to internal data structures to pass the information from one module to the next. The integrated source code versions are in Attachment VII.

This use of the user interface software proved to be a success when some of the typists using the word processor who were not familiar with software, had no major problems.

At one point, a number of chapters were typed with the wrong keystrokes used consistently for one of the alphabetic input characters because they better fit the pattern of the

keystroke definitions. Another typist preferred the original way of entering the symbol. The "wrong" set of keystrokes was added to the table as a second way of entering that alphabetic symbol and work proceeded with each typist using the keystrokes that they preferred. The format of the table allows multiple input strings to map to the same numeric code. Reference 7 is the book as produced by the word processor and user interface software and as sent to India for publication.

This experience clearly demonstrated the functionality, extensibility, modularity and flexibility of the software design.

4.4 General Problems

The following is a list of problems that were encountered during the design, implementation and subsequent usage of the software that and are still unresolved:

- The complexity of some characters does not allow them to be printed with an 11x7 dot matrix.

- Only 64 downline loadable characters are allowed in the particular printer that was used. Several hundred would allow much greater flexibility in the implementation of combined character printing.

- The translation to combined characters has not been developed yet. (It is, however, designed and will not be a major effort.)

- No mechanism exists in the Advanced BASIC language to allow files to be included or libraries to be used so that commonly used routines could be readily shared.

5. Conclusion

The Sanskrit user interface is now ready to be used by the Grammar Engine software when it becomes available. A generalized translation mechanism based on user defined input tables formed the heart of the software. This mechanism simplified the problem of doing several translations by generalizing to one case that not only includes a range of possible uses besides those involved in the original problem.

The code was easily adapted to do translation for a word processor and performed well when used to assist in the publication of a book containing Sanskrit text. Dr. Singh's book is now in publication in India.

Methods of printing complex Devanagri symbols via the use of primitives are being examined. In particular, current work at Stanford University was found [16] that relates to typography [15] and Indian scripts [14]. One of the techniques that may be adapted from this work involves the definition of a set of primitive strokes of symbols that allow printing of any complex Devanagri symbol when used in various combinations.

Aside from the printing of full Devanagri script, the user interface may soon be extended in a number of ways:

- A standard numeric code may be agreed upon for Sanskrit. If a standard is selected, the user interface may be altered to use it.

- Either an actual Sanskrit keyboard will be developed or a graphics display with light pen selection will be implemented.

- Graphics capabilities may be used to provide information displayed in color. An example of the use of this capability would be color coding the grammar rules according to type.

Members of the Grammar Engine project are proceeding with the study of the grammar. A very large project is now underway involving Stanford and a number of other institutions dealing with situated language in general [18]. Directors of the project are interested in sharing information with Grammar Engine project members [17] and intend to do so.

6. References

[1] Vasu, Srisa Chandra The Ashtadhyayi of Panini. Delhi, India: Motilal Banarsidass, 1891, reprinted 1962.

[2] Bohtlingk, Otto Panini's Grammatik. Germany: Georg Olms Verlagsbuchhandlung, 1964.

[3] Faddegon, Barend Studies on Panini's Grammar. Amsterdam: Uitgave Van De N. V. Noord-Hollandsche Uitgeversmaatschappi, 1936.

[4] Lanman, Charles Rockwell A Sanskrit Reader. Cambridge, Massachusetts: Harvard University Press, 1967.

[5] Macdonell, Arthur A. A Sanskrit Grammar for Students. London: Oxford University Press, 1927.

[6] Perry, Edward Delavan A Sanskrit Primer. USA: Columbia University Press, 1927.

[7] Singh, J. D. Panini - His Description of Sanskrit, An Analytical Study of Astadhyayi. In publication.

[8] Barrett, William A. and John D. Couch Compiler Construction: Theory and Practice. USA: Science Research Associates, 1979.

[9] International Business Machine Corporation. Personal

Computer XT Hardware Reference Library: Technical Reference International Business Machine Corporation, 1983.

[10] International Business Machine Corporation. Personal Computer XT Hardware Reference Library: Guide to Operations International Business Machine Corporation, 1983.

[11] International Business Machine Corporation. Personal Computer Hardware Reference Library: BASIC International Business Machine Corporation, 1982.

[12] International Business Machine Corporation. Personal Computer Computer Language Series: Disk Operating System by Microsoft, Inc. International Business Machine Corporation, 1983.

[13] Okidata. Microline 92 Serial Dot Matrix Printer. Okidata.

[14] Ghosh, Pijush K. An Approach to Type Design and Text Composition in Indian Scripts. Stanford, California: Department of Computer Science, Stanford University, 1983.

[15] Ghosh, Pijush K and Charles A. Bigelow. A Formal Approach to Lettershape Description for Type Design Stanford, California: Department of Computer Science,

Stanford University, 1983.

[16] Knuth, Donald E. Private Communication, Stanford University, November, 1983.

[17] Winograd, Terry. Private Communication, Stanford University, November, 1983.

[18] "Center for the Study Of Language and Information Research Program on Situated Language." AI Magazine, Vol. 5, No. 2, Summer 1984, 65-70.

7. Acknowledgements

I would like to thank my advisor, Dr. Rod Bates for his advice and encouragement as well as the other members of my masters' committee, Dr. Rich McBride and Dr. Roger Hartley, for theirs. The Computer Science Department staff assisted me extensively in producing this paper. Arvind Rana provided a wonderful opportunity to do a project that was both interesting and useful and made the project quite rewarding. His wife and family welcomed me into their home and made the entire experience most enjoyable. My fellow graduate students (in the Summer-On-Campus Program) have encouraged me not only during this project and the production of this paper, but during the entire course of my graduate studies. I am deeply indebted to you all.

8. Figures

8.1 Figure 1 - The Devanagri Alphabet

THE DEVANĀGARI LETTERS

Vowels.			Consonants.					
Initial.	Medial.	Equivalent.		Equivalent.			Equivalent.	
अ / अ	—	a	क	k	Gutturals	प	p	Labials
आ / आ	ा	ā	ख	k-h		फ	p-h	
इ	ि	i	ग	g		ब	b	
ई	ी	ī	घ	g-h		भ	b-h	
उ	ु	u	ङ	ṅ		म	m	
ऊ	ू	ū	च	c	Palatals	य	y	Semivowels
ऋ	ृ	ṛ (or ṛi)	छ	c-h		र	r	
ॠ	ॄ	ṝ (or ṛī)	ज	j		ल	l	
ऌ	ॢ	ḷ (or ḷi)	झ / ञ	j-h or ñ		व	v	
ए	े	e	ट	ṭ	Cerebrals	श	ś (or ç)	Spirants
ऐ	ै	ai	ठ	ṭ-h		ष	ṣ	
ओ	ो	o	ड	ḍ		स	s	
औ	ौ	au	ढ	ḍ-h		ह	h	
			ण	ṇ				
			त	t	Dentals	:	ḥ (Visarga)	
			थ	t-h				
			द	d		ं	ṃ or ṁ (Anusvāra)	
			ध	d-h				
			न	n				

8.2 Figure 2 - Keyboard Input Codes

Phonetic	Keyboard	Devanagri
a	a	अ
ā	A	आ
i	i	इ
ī	I	ई
u	u	उ
ū	U	ऊ
ṛ	R	ऋ
ṝ	r~	ॠ
ḷ	l	ऌ
e	e	ए
ai	E	ऐ
o	o	ओ
au	O	औ
k	k	क
kh	kh	ख
g	g	ग
gh	gh	घ
ṅ	G	ङ

8.2 Figure 2 - Keyboard Input Codes (cont.)

Phonetic	Keyboard	Devanagri
c	c	च
ch	ch	छ
j	j	ज
jh	jh	झ
ñ	Y	ञ
ṭ	T	ट
ṭh	Th	ठ
ḍ	D	ड
ḍh	Dh	ढ
ṇ	N	ण
t	t	त
th	th	थ
d	d	द
dh	dh	ध
n	n	न
p	p	प
ph	ph	फ
b	b	ब

8.2 Figure 2 - Keyboard Input Codes (cont.)

Phonetic	Keyboard	Devanagri
bh	bh	भ
m	m	म
y	y	य
r	r	र
l	l	ल
v	v	व
ś	sh	श
ṣ	S	ष
s	s	स
h	h, H	ह
ṃ	M	ं
ḥ	:	ः

8.3 Figure 3 - Numeric Encoding Scheme

Phonetic	Numeric	Devanagri
a	1	अ
ā	-1	आ
i	2	इ
ī	-2	ई
u	3	उ
ū	-3	ऊ
ṛ	4	ऋ
ṝ	-4	ॠ
ḷ	5	ऌ
e	6	ए
ai	8	ऐ
o	7	ओ
au	9	औ
k	38	क
kh	30	ख
g	27	ग
gh	22	घ
ṅ	17	ङ

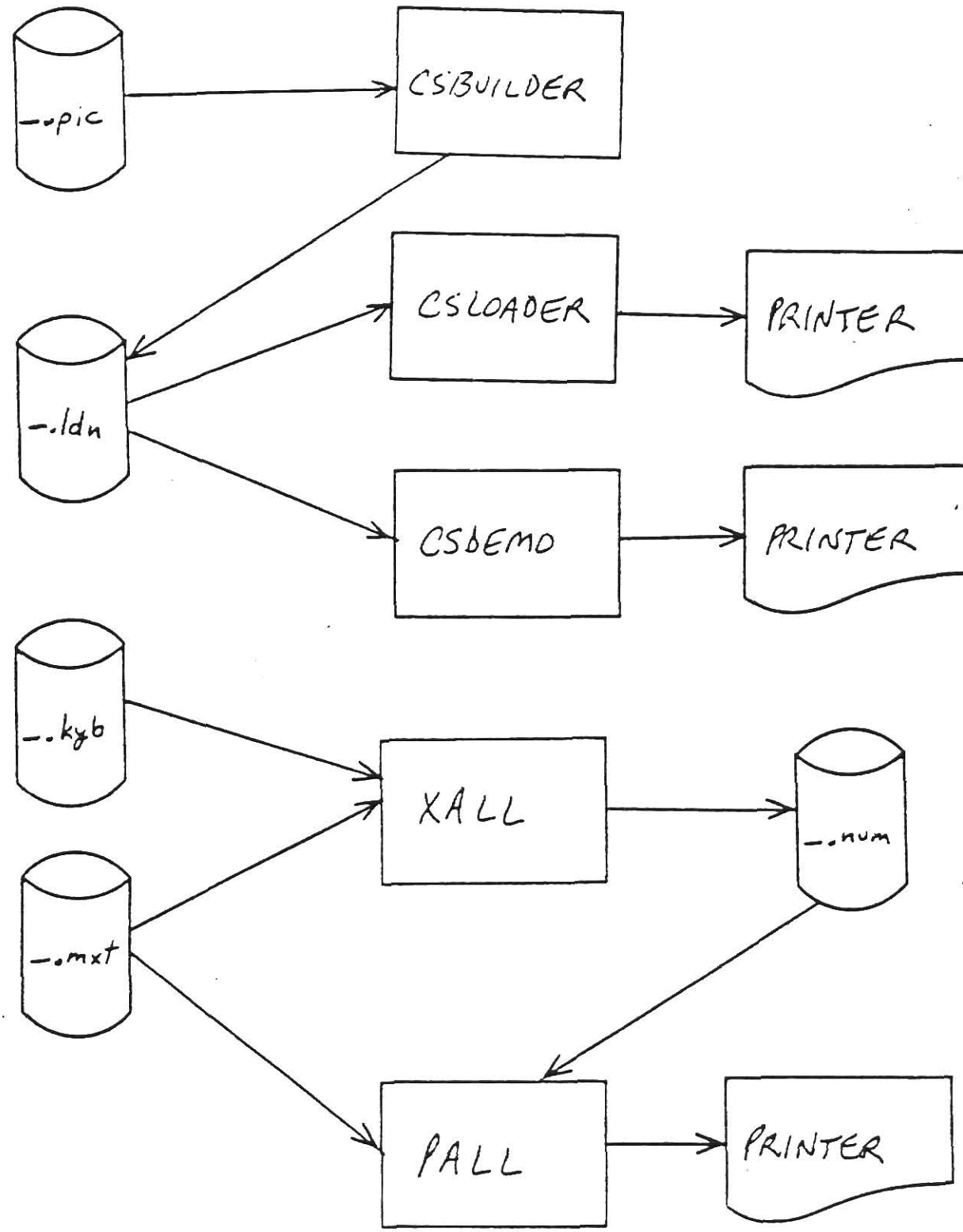
8.3 Figure 3 - Numeric Encoding Scheme (cont.)

Phonetic	Numeric	Devanagri
c	35	च
ch	32	छ
j	25	ज
jh	20	झ
ñ	15	ञ
ṭ	36	ट
ṭh	33	ठ
ḍ	28	ड
ḍh	23	ढ
ṇ	18	ण
t	37	त
th	34	थ
d	29	द
dh	24	ध
n	19	न
p	39	प
ph	31	फ
b	26	ब

8.3 Figure 3 - Numeric Encoding Scheme (cont.)

Phonetic	Numeric	Devanagri
bh	21	भ
m	16	म
y	11	य
r	13	र
l	14	ल
v	12	व
ś	40	श
ṣ	41	ष
s	42	स
h	10	ह
ṃ	44	ं
ḥ	45	ः

8.4 Figure 4 - Subsystem Decomposition



Attachment I - Sample Character Picture File

```
"a",D
"   xxxxxxxx "
"            "
"   xx xxx   "
"     x    xx "
"     x       "
"             "
"     xx      "
"b",A
"   xxxxxxxx "
"            "
"  x         x "
"  x         x "
"  x         x "
"  x         x "
"   xxxxxxx  "
"c",A
"   xxxxxxxx "
"            "
"     xxx    "
"      x     "
"      x     "
"      x     "
"     xxx    "
"d",A
"   xxxxxxxx "
"            "
"     xxxx   "
"    x     x "
"   x       x "
"   xxxxxxxx "
"   x       x "
"e",A
"            "
"            "
"     xxxx   "
"    x     x "
"   x       x "
"   xxxxxxxx "
"   x       x "
"f",A
"            "
"            "
"    xxx     "
"     x      "
"     x      "
"     x      "
"    xxx     "
```

Attachment I - Sample Character Picture File (cont.)

```
"g",A
"               "
"               "
" xxx    xxx "
"  x      x  "
"  x      x  "
"  x      x  "
"   xxxxx   "
"h",D
"  xxxxxxx  "
"  x      x "
"  xxxxxxx  "
"  xxxx     "
"  x  xxxx  "
"               "
"    XX     "
"i",D
"    x      "
"    x      "
"    x      "
"    x      "
"    xxxxxx  "
"               "
"    XX     "
"J",A
"               "
"               "
"  xxxxxxx  "
"  x        "
"  xxxxx    "
"  x        "
"  xxxxxxx  "
"K",A
"               "
"               "
"  xxxxxxx  "
" x       x "
" x       x "
" x       x "
"  xxxxxxx  "
"l",A
"               "
"               "
"  xx    xxx "
" x  x    x  "
"x    x   x  "
"xxxxxx   x  "
"x    x xxx "
```

Attachment I - Sample Character Picture File (cont.)

```
"m",A
"                "
"                "
"   xx    x   x"
"  x  x   x   x"
"x     x  x   x"
"xxxxxx x   x"
"x     x   xx  "
"n",A
"                "
"                "
"   x       x    "
"   x       x    "
"   xxxxxxx      "
"   x       x    "
"   x       x    "
"o",A
"                "
"                "
"   x       x    "
"    x     x     "
"     x x        "
"      x         "
"      x         "
"p",A
"                "
"                "
"  x          x  "
"   X       x    "
"    x     x     "
"     x x        "
"      x         "
"q",A
"                "
"                "
"   xxxxxx       "
"   x       x    "
"   xxxxxx       "
"   xxxx         "
"   x    xxx     "
"r",A
"                "
"                "
"   x            "
"   x            "
"   x            "
"   x            "
"   xxxxxxx      "
```

Attachment I - Sample Character Picture File (cont.)

```
"s",A
"     xx       "
"              "
"  x xxxxx     "
"   x      x   "
"   x      x   "
"   x      x   "
"   x      x   "
"t",A
"              "
"              "
"  xx      xx  "
"  x x    x x  "
"  x   x x   x  "
"  x    x    x  "
"  x         x  "
"u",A
"   xxxxxxx    "
"              "
"  x xxxxx     "
"   x      x   "
"   x      x   "
"   x      x   "
"   x      x   "
"v",D
"  x xxxxx     "
"   x      x   "
"   x      x   "
"   x      x   "
"   x      x   "
"              "
"    xxxxx     "
"w",A
"              "
"              "
"   xx     x   "
"   x x    x   "
"   x  x   x   "
"   x    x x   "
"   x     xx   "
"x",A
"        x     "
"        xxxx  "
"      x x    x"
"      x x    x"
"      x       "
"x     x       "
"  xxx         "
```

Attachment I - Sample Character Picture File (cont.)

```
"y",A
"       x     "
"x      xxxx  "
"x      x    x"
"xxxx   x    x"
"x    x        "
"x    x        "
"xxxx          "
"z",D
"          x   "
" xxxXX xxx    "
"x     X x   x"
"x     X x   x"
" XXXXx        "
"       x      "
" xxxxX        "
"A",D
"       x x    "
"       x xxx  "
" xxxxx x   x"
"x     x x   x"
" xxxxx        "
"              "
"  xX          "
"B",A
"          x   "
"          xxx "
"       x x   x"
"       x x   x"
" xxxxx        "
"x     x       "
" xxxxx        "
"C",A
"              "
"              "
"       xxx    "
"        x     "
"        x     "
" x      x     "
"  xxxx        "
"D",A
"              "
"              "
" xxxxxxx      "
" x       x    "
" xxxxxxx      "
" x       x    "
" xxxxxxx      "
```

Attachment I - Sample Character Picture File (cont.)

```
"E",A
"              "
"              "
" xxxxxxxx     "
" x            "
" x     xxx    "
" x       x    "
" xxxxxxxx     "
"F",D
" xxxxxxx      "
" x        x   "
" x        x   "
" x        x   "
" xxxxxxx      "
"              "
"     xx       "
"G",A
"              "
"              "
" xxxxxxx      "
" x        x   "
" x        x   "
" x        x   "
" xxxxxxx      "
"H",A
"         x    "
"x        xxx  "
"x    x   x   x"
"x   x    x   x"
"xxx           "
"x   x         "
"x    xx       "
"I",D
"         x    "
"xxxxx   xxx   "
"x     x x   x"
"x     x x   x"
"xxxxx         "
"x             "
"x             "
"J",A
"         x    "
"         xxx  "
" xxx     x   x"
"x    x   x   x"
"x             "
"x    x        "
" xxx          "
```

Attachment I - Sample Character Picture File (cont.)

```
"K",D
"x        x    "
"xXXX     xxx  "
"x        x   x"
"x        x   x"
" XXXX         "
"              "
"   Xx         "
"L",A
"         x    "
"x        xxx  "
"x        x   x"
"xxxx     x   x"
"x             "
"x             "
" xxxx         "
"M",A
"              "
"              "
"   xxxxxxx    "
" x            "
" x            "
" x            "
"   xxxxxxx    "
"N",D
"   xxxxxxx    "
"       x      "
"       x      "
"       x      "
"       x      "
"              "
"       xx     "
"O",A
"              "
"              "
"   xxxxxxx    "
"       x      "
"       x      "
"       x      "
"       x      "
"P",A
"              "
"              "
" x      xx    "
" x    x       "
" xXXX         "
" x    x       "
" x      xx    "
```

Attachment I - Sample Character Picture File (cont.)

```
"Q",D
"  xxxxxxx    "
"  x       x  "
"  x       x  "
"  x       x  "
"  xxxxxxx    "
"  x          "
"  x          "
"R",A
"  xx   xx    "
"    xx       "
"   xxxxxxx   "
"  x          "
"   xxxxxx    "
"         x   "
"  xxxxxxx    "
"S",D
"   xxxxxx    "
"  x          "
"   xxxxxx    "
"         x   "
"  xxxxxxx    "
"             "
"     x       "
"T",A
"             "
"             "
"   xxxxxxx   "
"  x          "
"   xxxxxxx   "
"          x  "
"   xxxxxxx   "
"U",A
"     xx      "
"             "
"  xx      xx "
"  x x    x x "
"  x   x x   x "
"  x    x    x "
"  x         x "
"V",D
"   x       x "
"   x       x "
"   xxxxxxx   "
"   x       x "
"   x       x "
"             "
"      xx     "
```

Attachment II - Sample Loadable Numeric File

```
%D,a, 0 , 0 , 5 , 5 , 89 , 69 , 5 , 5 , 9 , 9 , 0
%A,b, 0 , 60 , 65 , 65 , 65 , 65 , 65 , 65 , 65 , 61 , 0
%A,c, 0 , 0 , 1 , 1 , 69 , 125 , 69 , 1 , 1 , 1 , 0
%A,d, 0 , 0 , 113 , 41 , 37 , 37 , 37 , 37 , 41 , 113 , 0
%A,e, 0 , 0 , 112 , 40 , 36 , 36 , 36 , 36 , 40 , 112 , 0
%A,f, 0 , 0 , 0 , 68 , 124 , 68 , 0 , 0 , 0 , 0 , 0
%A,g, 0 , 4 , 60 , 68 , 64 , 64 , 64 , 68 , 60 , 4 , 0
%D,h, 0 , 0 , 31 , 13 , 77 , 93 , 21 , 21 , 21 , 2 , 0
%D,i, 0 , 0 , 0 , 31 , 80 , 80 , 16 , 16 , 16 , 0 , 0
%A,j, 0 , 0 , 124 , 84 , 84 , 84 , 84 , 68 , 68 , 0 , 0
%A,k, 0 , 56 , 68 , 68 , 68 , 68 , 68 , 68 , 68 , 56 , 0
%A,l, 112 , 40 , 36 , 36 , 40 , 112 , 0 , 68 , 124 , 68 , 0
%A,m, 112 , 40 , 36 , 36 , 40 , 112 , 0 , 60 , 64 , 64 , 60
%A,n, 0 , 0 , 124 , 16 , 16 , 16 , 16 , 16 , 124 , 0 , 0
%A,o, 0 , 0 , 4 , 8 , 16 , 96 , 16 , 8 , 4 , 0 , 0
%A,p, 0 , 4 , 8 , 16 , 32 , 64 , 32 , 16 , 8 , 4 , 0
%A,q, 0 , 0 , 124 , 52 , 52 , 52 , 84 , 84 , 72 , 0 , 0
%A,r, 0 , 0 , 124 , 64 , 64 , 64 , 64 , 64 , 64 , 0 , 0
%A,s, 0 , 4 , 120 , 4 , 5 , 5 , 4 , 4 , 120 , 0 , 0
%A,t, 0 , 124 , 4 , 8 , 16 , 32 , 16 , 8 , 4 , 124 , 0
%A,u, 0 , 4 , 121 , 5 , 5 , 5 , 5 , 5 , 121 , 0 , 0
%D,v, 0 , 1 , 30 , 65 , 65 , 65 , 65 , 65 , 30 , 0 , 0
%A,w, 0 , 0 , 124 , 4 , 8 , 16 , 32 , 64 , 124 , 0 , 0
%A,x, 32 , 64 , 64 , 64 , 60 , 0 , 15 , 2 , 2 , 2 , 12
%A,y, 126 , 72 , 72 , 72 , 48 , 0 , 15 , 2 , 2 , 2 , 12
%D,z, 12 , 82 , 82 , 82 , 82 , 126 , 0 , 15 , 2 , 2 , 12
%D,A, 8 , 20 , 84 , 84 , 20 , 31 , 0 , 15 , 2 , 2 , 12
%A,B, 32 , 80 , 80 , 80 , 80 , 124 , 0 , 15 , 2 , 2 , 12
%A,C, 0 , 32 , 64 , 64 , 64 , 68 , 60 , 4 , 0 , 0 , 0
%A,D, 0 , 124 , 84 , 84 , 84 , 84 , 84 , 84 , 40 , 0 , 0
%A,E, 0 , 124 , 68 , 68 , 68 , 68 , 84 , 84 , 116 , 0 , 0
%D,F, 0 , 31 , 17 , 17 , 81 , 81 , 17 , 17 , 14 , 0 , 0
%A,G, 0 , 124 , 68 , 68 , 68 , 68 , 68 , 68 , 56 , 0 , 0
%A,H, 126 , 16 , 16 , 40 , 68 , 64 , 0 , 15 , 2 , 2 , 12
%D,I, 126 , 18 , 18 , 18 , 18 , 12 , 0 , 15 , 2 , 2 , 12
%A,J, 56 , 68 , 68 , 68 , 40 , 0 , 0 , 15 , 2 , 2 , 12
%D,K, 15 , 18 , 82 , 82 , 16 , 0 , 0 , 15 , 2 , 2 , 12
%A,L, 62 , 72 , 72 , 72 , 64 , 0 , 0 , 15 , 2 , 2 , 12
%A,M, 0 , 56 , 68 , 68 , 68 , 68 , 68 , 68 , 68 , 0 , 0
%D,N, 0 , 0 , 1 , 1 , 1 , 95 , 65 , 1 , 1 , 0 , 0
%A,O, 0 , 0 , 4 , 4 , 4 , 124 , 4 , 4 , 4 , 0 , 0
%A,P, 0 , 124 , 16 , 16 , 16 , 40 , 68 , 68 , 0 , 0 , 0
%D,Q, 0 , 127 , 17 , 17 , 17 , 17 , 17 , 17 , 14 , 0 , 0
%A,R, 0 , 73 , 85 , 86 , 86 , 85 , 85 , 84 , 36 , 0 , 0
%D,S, 0 , 18 , 21 , 21 , 85 , 21 , 21 , 21 , 8 , 0 , 0
%A,T, 0 , 8 , 84 , 84 , 84 , 84 , 84 , 84 , 84 , 32 , 0
%A,U, 0 , 124 , 4 , 8 , 17 , 33 , 16 , 8 , 4 , 124 , 0
%D,V, 0 , 0 , 31 , 4 , 4 , 68 , 68 , 4 , 31 , 0 , 0
%A,W, 0 , 48 , 84 , 84 , 84 , 84 , 84 , 84 , 84 , 56 , 0
```

Attachment III - Sample Numeric Code File

```
 1  1  2  1  1  5  1  1  10  9  1  5  9  1  2  3  4
 5  6  7  8  9  10  11  12  1  0  5  0  9  1  0  5  0
 9  0  0  1  0  0  0  0  5  0  9  0  0
```

Attachment IV - Sample Master Translation Table

-4,"r~",a
-3,"U ",b
-2,"I ",c
-1,"A ",d
1,"a ",e
2,"i ",f
3,"u ",g
4,"R ",h
5,"L ",i
6,"e ",J
7,"o ",k
8,"E ",l
9,"O ",m
10,"H ",n
10,"h ",n
11,"y ",o
12,"v ",p
13,"r ",q
14,"l ",r
15,"Y ",s
16,"m ",t
17,"G ",u
18,"N ",v
19,"n ",w
20,"jh",x
21,"bh",y
22,"gh",z
23,"Dh",A
24,"dh",B
25,"J ",C
26,"b ",D
27,"g ",E
28,"D ",F
29,"d ",G
30,"kh",H
31,"ph",I
32,"ch",J
33,"Th",K
34,"th",L
35,"c ",M
36,"T ",N
37,"t ",O
38,"k ",P
39,"p ",Q
39,"P ",Q
40,"sh",R
41,"S ",S
42,"s ",T
44,"M ",U
45,": ",V

Attachment V - Sample Keyboard Input File

```
?aeiou
AEIOU
r~
ar~e
r~r~
r~ aR e
a
A
"r~ aR e"
" a "
```

Attachment VI - Logo Building Experiment

```
130 OPEN "I",#1,"c:GEILCS"
140 OPEN "O",#2,"lpt1:"
150 FOR CHRCTR% = 1 TO 12 STEP 1
160     INPUT #1,NAM$
170     PRINT "loading next character: "; NAM$
180     PRINT #2,CHR$(27);"%A";NAM$;
190     FOR COL% = 1 TO 11 STEP 1
200             INPUT #1,COLCODE%
210             PRINT COLCODE%;
220             PRINT #2,CHR$(COLCODE%);
230     NEXT COL%
240     PRINT "";     REM put a line feed out after each character
245     PRINT #2, ""
250 NEXT CHRCTR%
255 REM put a line feed out after all done (clears garbage out)
260 PRINT #2, ""
280 PRINT #2, CHR$(27); "1"; "GRAMMAR"; CHR$(27); "2"; "1234"
290 PRINT #2, CHR$(27); "1"; " ENGINE"; CHR$(27); "2"; "5678"
300 PRINT #2, CHR$(27); "1"; "    INC"; CHR$(27); "2"; "9ABC"
310 PRINT #2, CHR$(27); "0"
```

```
"1",0,0,127,127,127,127,7,7,7,7,7
"2",7,7,7,7,7,7,7,7,7,127,127
"3",127,127,7,7,7,7,7,7,7,7,7
"4",7,7,7,7,127,127,127,127,0,0,0
"5",0,0,127,127,127,127,28,28,28,28,28
"6",28,28,28,28,28,28,28,28,28,127,127
"7",127,127,60,124,124,92,92,28,28,28,28
"8",28,28,28,28,127,127,127,127,0,0,0
"9",0,0,127,127,127,127,112,112,112,112,112
"A",112,112,112,112,112,112,112,112,112,127,127
"B",127,127,112,112,112,113,113,115,115,118,118
"C",124,124,120,120,127,127,127,127,0,0,0
```

GRAMMAR
ENGINE
INC

Grammar
Engine
Inc

Attachment VII - Programs as Used in Word Processor

CSBUILDER

```
10 REM CS.BAS - build a character set
20 DIM R$(7)
30 INPUT "input file:",F1$ : INPUT "output file:",F2$
40 OPEN "I",#1,F1$ : OPEN "O",#2,F2$
50 FOR C%=1 TO 64 : IF EOF(1) THEN END
60 INPUT #1,N$,A$ : PRINT "translating : ";N$
70 FOR W%=0 TO 6 : IF EOF(1) THEN 170
80 INPUT #1,R$(W%) : PRINT R$(W%) : NEXT W%
90 PRINT #2,"%";A$;",";N$;
100 FOR K%=1 TO 11 : S%=0
110 FOR W%=0 TO 6
120 IF MID$(R$(W%),K%,1)()" " THEN S%=S%+(2^W%)
130 NEXT W%
140 PRINT #2,",";S%; : NEXT K%
150 PRINT #2,"" : NEXT C%
160 IF EOF(1) THEN END
170 PRINT "character partially defined" : BEEP : END
```

Attachment VII - Programs as Used in Word Processor (Cont.)

CSLOADER

```
3610 REM
3620 REM subroutine to load a character set
3630 REM
3640 OPEN "I",#6,CX$
3650 FOR CC=1 TO 64 : IF EOF(6) THEN 3700
3660 INPUT #6,AD$,NM$ : PRINT #2,CHR$(27);AD$;NM$;
3670 FOR C=1 TO 11 : IF EOF(5) THEN KK=0 ELSE INPUT #6,KK
3680 PRINT #2,CHR$(KK); : NEXT C
3690 NEXT CC
3700 CLOSE #6
3710 REM load master translate table.
3720 REM
3730 OPEN "I",#6,"san.mxt"
3740 ON ERROR GOTO 3760
3750 FOR M.C=0 TO 63 : INPUT #6,M.N(M.C),M.K$(M.C),M.M$(M.C) : NEXT
3760 CLOSE #6
3770 ON ERROR GOTO 3510 : RESUME 3780
3780 RETURN
```

Attachment VII - Programs as Used in Word Processor (Cont.)

CSDEMO

```
100 REM CSDEMO.BAS - demonstrate a character set
110 REM This program reads one line of input per character
120 REM containing the name that is to reference it and 11
130 REM numbers describing each column that has been
135 REM downline loaded into the lpt. The name is the
140 REM only field used to print the actual character.
150 INPUT "Loaded character set file? ", FILENAME$
160 OPEN "I",#1,FILENAME$
170 OPEN "O",#2,"lpt1:"
180 FOR CHRCTR% = 1 TO 64 STEP 1
190 IF EOF(1) THEN GOTO 310
200 INPUT #1, NAM$
210 PRINT "printing next character: "; NAM$
220 PRINT #2, CHR$(27); "1"; NAM$; " - "; CHR$(27); "2"; NAM$
230 FOR COL% = 1 TO 11 STEP 1
240 IF EOF(1) THEN GOTO 350
250 INPUT #1, COLCODE%
260 PRINT COLCODE%;
270 NEXT COL%
280 PRINT "": REM put a line feed out after each character
290 PRINT #2, ""
300 NEXT CHRCTR%
310 PRINT #2, CHR$(27); "1";
320 IF EOF(1) THEN END
330 PRINT "Can't load more than 64 characters..." : BEEP
340 END
350 PRINT #2, CHR$(27); "1";
360 PRINT "Partially defined character..." : BEEP
370 END
```

Attachment VII - Programs as Used in Word Processor (Cont.)

XALL/PALL

```
3790 REM XSTRING - Translate string from keyboard input
3800 REM
3810 TSX$="" : TKX$="" : LOCC=0
3820 FOR P=1 TO WL : TK$=MID$(WD$(WN),P,1)+" "
3830 IF P(WL THEN TKX$=MID$(WD$(WN),P+1,1)
3840 IF TK$="r " AND TKX$="~" THEN TK$="r~" : GOTO 3860
3850 IF TK$="s " AND TKX$="h" THEN TK$="sh" ELSE 3870
3860 P=P+1
3870 FOR X=0 TO M.C- 1
3880 IF M.K$(X)=TK$ THEN TSX$=TSX$+M.M$(X) : GOTO 3910
3890 NEXT X
3900 TSX$=TSX$+CHR$(27)+CHR$(49)+LEFT$(TK$,1)+CHR$(27)+CHR$(50)
3905 LOCC=LOCC+4
3910 NEXT P
3920 WD$(WN)=CHR$(27)+CHR$(50)+TSX$+CHR$(27)+CHR$(49)
3925 WL=LEN(TSX$)- LOCC
3930 RETURN
```

A SANSKRIT USER INTERFACE

by

DAVID GEORGE NOHLE

B. S., Ohio State University, 1979

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1984

A SANSKRIT USER INTERFACE

by David George Nohle

AN ABSTRACT OF A MASTER'S REPORT

This paper describes a master's project involved with the Grammar Engine*, an experiment in natural language understanding with the Sanskrit language. The author developed a user interface that has already been used in word processing applications and is intended to be the user interface for the Grammar Engine system.

The Sanskrit language and its script, Devanagri, pose some interesting user interface and translation problems. The ability to use standard English keyboards to input Devanagri was desired. Normally, Devanagri syllables containing multiple consonants and a vowel are written with one symbol and it is desirable to enter them as separate alphabetic characters. This user interface was designed to allow its use for a variety of input character sets and any output character set and is presently capable of allowing for printing of any output character set whose characters can be depicted with an 11x7 dot matrix. Phonetic, numeric and keyboard input character sets are able to be printed presently. Although hardware is not available to allow printing of the complex Devanagri characters, the translation to a numeric print code has been achieved. Translation by combining Devanagri characters into numeric codes that represent syllables has been planned for and is expected to be developed with little additional effort.

---

* Grammar Engine is a registered trademark of Grammar Engine Inc.