CANADIAN THESES

THÈSES CANADIENNES

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

THIS DISSERTATION HAS BEEN MICROFILMED EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ MICROFILMÉE TELLE QUE NOUS L'AVONS REÇUE

The University of Alberta

A Chinese Character Software System

by

Samuel Yin Lun PUN

A thesis
submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree
of Master of Science

Department of Computing Science

Edmonton, Alberta
Spring, 1986

ISBN 0-315-30300-X

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Samuel Yin Lun PUN

TITLE OF THESIS: A Chinese Character Software System

DEGREE FOR WHICH THIS THESIS WAS PRESENTED: Master of Science

YEAR THIS DEGREE GRANTED: 1986

(Signed) .....Samuel Pun.....

Permanent Address:
22c, 15/Fl., Broadway,
Mei Foo Sun Chuen,
Kowloon,
Hong Kong.

Date Jan. 13, 1986

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled **A Chinese Character Software System** submitted by **Samuel Yin Lun PUN** in partial fulfillment of the requirements for the degree of **Master of Science.**

T. H. Merrett
Supervisor

Wu Xuanzhi

Date Jan. 13, 1986

TO MY MOTHER

ABSTRACT

Developments in computer technology in the past few decades have led to rapid advances in the machine processing of information. Native Chinese speakers account for about a quarter of the world's population. The need for Chinese information processing among Chinese is not questioned.

This thesis is concerned with the design and implementation of two major components of a *Chinese Character Software System* (CCSS). These components are the *Logical Chinese Terminal* (LCT) and the *Pattern Editor*. The CCSS provides preliminary capabilities for Chinese information processing. The LCT serves as an interface between users and the UNIX system, and allows the input and output of both Chinese and conventional characters. The primary function of the pattern editor is to allow users to create interactively the display patterns of Chinese characters.

In the design of the components, the following issues have been investigated.

(1) The internal representation of Chinese characters in UNIX systems.

(2) A method of displaying (outputing) Chinese characters.

(3) A method of On-line Hand-written Chinese character recognition.

(4) An interactive approach for designing the display patterns of Chinese characters, and automatic generation of recognition schemes for designated characters.

The work reported here represents an attempt to establish a research tool for the study of Chinese information processing using UNIX systems.

ACKNOWLEDGEMENTS

I wish to thank Professor T. A. Marsland, my supervisor, for his advice, criticism and guidance throughout the research and preparation of this thesis. I am also grateful to Professor J. Penney for his helpful comments and suggestions on this thesis. Further thanks are due to the members of my examination committee, Dr. E. Chan, Dr. M. Green and Professor Xuanzhi Wu for their helpful comments.

I must thank Mr. J. Lennon who has read the draft of this thesis and improved its readability.

Financial support from the Department of Computing Science in the form of teaching and research assistantships and the Natural Sciences and Engineering Research Council of Canada for supporting this work under the grant NSERC A7902 are gratefully acknowledged.

Special thanks to all my friends, especially H. C. Chan, C. W. Cheng, S. C. Lau, and H. K. Yim for the various form of help and encouragement that they have given to me.

Finally, I am deeply grateful to my wife, Clara, whose contribution in terms of encouragement, patience and understanding is immeasurable. Her love made everything possible.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1. The Goal

Developments in computer technology in recent decades have led to rapid advances in the machine processing of information. To increase the accessibility of computer technology in Chinese speaking society the computer input and output of Chinese characters is now a subject of intensive research around the world. A good introduction to the field has been given by Chu [Chu79a].

Chinese information processing is often encumbered at the input and output stages owing to the enormous set of characters involved. The language consists of more than 50,000 characters (49030 characters are listed in the Kang-hsi dictionary) of which 3,000 to 5,000 are in common use. Without machine processing of Chinese, many important computer applications in business, management and information retrieval cannot be developed for Chinese speaking society, which composes a quarter of the world's population. The need for Chinese information processing is not questioned.

This thesis is concerned with the design and implementation of a software system, specifically a *Chinese Character Software System* (CCSS), on UNIX[1] that provides a basic capability for Chinese information processing. The following issues, raised by the design and implementation of the system, are presented (investigated) in this thesis:

(1) The design of an internal representation of Chinese characters;

(2) The design of an efficient and effective method for displaying Chinese characters;

[1] UNIX is a trademark of AT & T BELL Laboratories.

(3) The design of a method for reconizing hand-written square style Chinese characters;

(4) The design and implementation of a logical Chinese terminal; and

(5) The design and implementation of a pattern editor.

The primary goal is to establish a research tool for the study of Chinese information processing using UNIX systems; that is to explore UNIX as a tool for research into Chinese information processing. The effort is justifiable because UNIX is a popular operating system. Moreover, many UNIX machines have been developed and are used extensively in universities and research institutes. The secondary goal is to produce a system that allows for growth and for experiments on various system components.

1.2. Formation of Chinese Characters

Chinese, being a hieroglyphic language, is very different from western languages such as English. It is necessary, therefore, to describe the formation of the characters. The development of Chinese ideographics took place in five stages over 6,000 years as described by Suen [Sue80]. The first stage was the creation of the characters, the Bone-and-Shell script that indeed were pictures. The next stage occurred during the Chou dynasty. The Bone-and-Shell script was developed into the six-fold classification that is called the Six Scripts. The standardization of the characters during the Qin dynasty led to today's unified set of characters. The fourth stage saw the development of the cursive script into a rectilinear form and the final stage occurred when the rectilinear characters were codified into the present square ideographic characters.

The ideographic characters currently in use have remained the same for about 2,000 years (a simplified version[2] has been under development in mainland China since 1956) and still can be classified into the following categories, the Six Scripts:

[2] Current version has 2,236 simplified characters [Yip85].

(1) Symbolic Characters : symbols of abstract concepts.

example:	modern form	meaning
i)	一	one
ii)	二	two
iii)	上	up
iv)	下	down

Table 1.1 Sample characters of the 'symbolic' script

(2) Pictograph : pictorial representation of objects.

example:	modern form	ancient form	meaning
i)	木		wood (tree)
ii)	水		water

Table 1.2 Sample characters of the 'pictograph' script

(3) Ideograph : two or more pictographs put together to form a character with a meaning extended from the pictorial combination.

example:	modern form	meaning
i)	林	grove(two trees)
ii)	休	rest(man resting beside a tree)

Table 1.3 Sample characters of the 'ideograph' script

(4) Extended Meaning : although most characters involve extended meaning, there are some cases where the meaning has been extended so far from the original meaning that the connection is obscure.

example:	modern form	meaning
i)		thing was originally an ideograph made up of ox and knife representing the sacrifice is the ancient ritual. A sacrifice is the offering of a 'thing' to the gods.

Table 1.4 Sample character of the 'extended meaning' script

(5) Phonetic Complex : one part of the character gives a clue on meaning, the other part gives a clue as to sound.

example:	modern form	meaning
i)		grass
ii)		flower
		radical standing for vegetation
		represents the sound

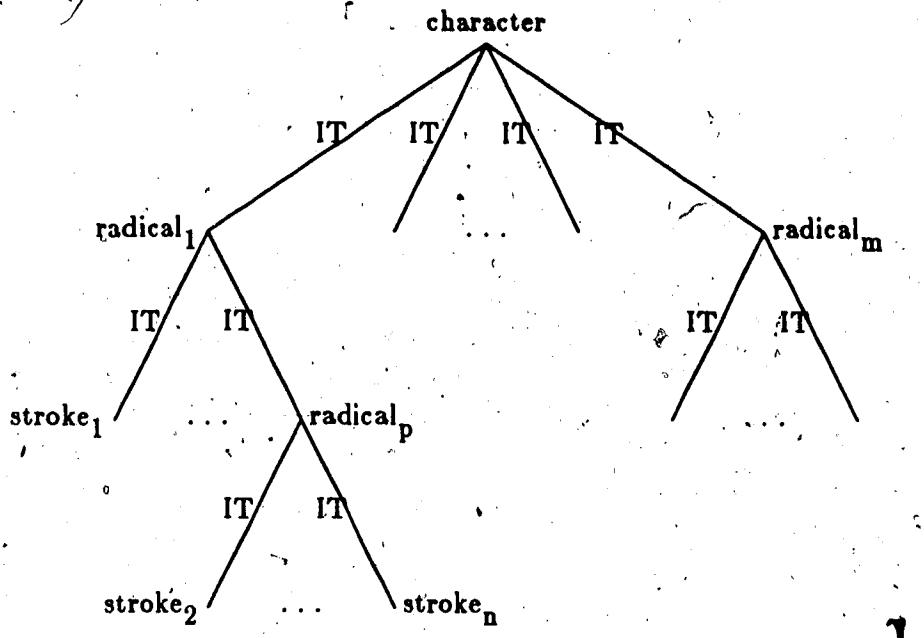
Table 1.5 Sample characters of the 'phonetic' script

(6) Borrowed Forms : This involves taking the original meaning away from a character and replacing it with an arbitrary meaning that has a similar sound to the original character.

example:	modern form	meaning
𠂤	自	nothingness was originally a pictograph of a dancer with fluttering sleeves which was borrowed to represent nothingness

Table 1.6 Sample character of the 'borrowed forms' script

Each of the ideographic patterns is composed of basic strokes confined in a square shape. Naturally, one would decompose the pattern into subparts that themselves are legal characters or subparts that are called radicals. Radicals can be decomposed further into basic strokes. Lines are used to make the strokes in modern writing. The hierarchical structure shown in Figure 1.1 illustrates the decomposition process. An ideograph may be decomposed into only one level or into multiple levels. Figure 1.2 shows an example of the decomposition process for the Chinese word *language*.



IT refers to a general sequence of one or more Instance Transformations (IT); the IT in general are all different. [FoD83].

Figure 1.1 The hierarchical structure of a Chinese character

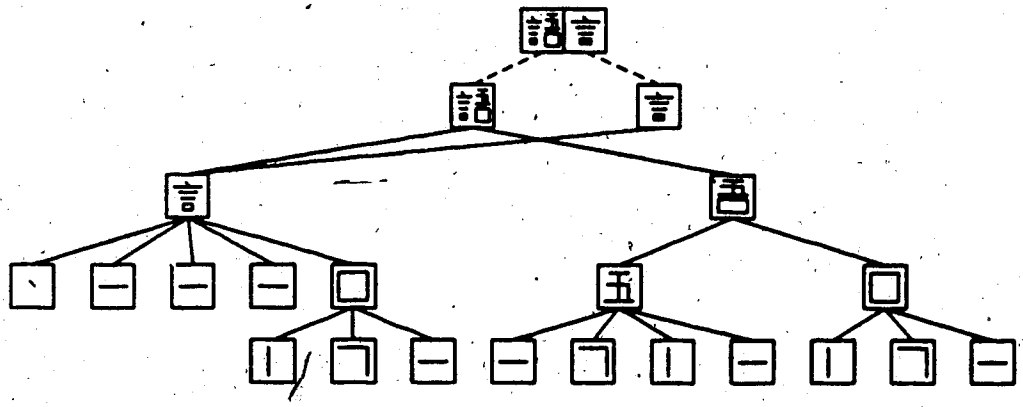


Figure 1.2 The decomposition of a Chinese word

1.3. The Fundamental Problems

The mapping from English text to its internal representation is straightforward, once the set of codes to represent each character has been determined. For Chinese, however, the mapping is more complicated because of the following fundamental problems involved in designing Chinese input methods.

(a) Large character set:

Chinese language consists of a large set of characters that is more than 50,000. Excluding the less frequently used characters, there remain at least 5,000 commonly used characters. This is the major problem in designing Chinese input methods.

(b) Complexity of characters:

The second problem is that Chinese ideographics have complex structures. There are more than seventy-five different combinations of relative position of radicals [HuS83]. Table 1.7 shows some of the basic structures. As described in Chapter two, the primitives of Chinese characters are strokes. Lee and Chou found that about fifteen per cent of the characters have more than twenty strokes and the average number of strokes per character is about fourteen [LeC80].

example:	structure	modern form	meaning
i)	left to right	加	plus
ii)	top to bottom	另	another
iii)	mixed	話	speech

Table 1.7 Some basic structures of Chinese characters

(c) Non-singular stroking order:

The problem is even more complicated owing to the lack of unique stroking sequence[3] among all writers of Chinese [MaS80]. A variety of writing sequences of a given character from individual to individual means that the characters cannot be uniquely identified by the sequence of their strokes.

(d) Similarity between characters:

Homonyms also exist in Chinese. Many characters look similar to each other but are completely different in meaning and vice versa. The relative position and the length of the strokes are often the keys that identify the individual characters. Some examples are shown in the table below.

example:	modern form	meaning
i)	日 / 曰	day / say
ii)	土 / 士 / 工	ground / gentlemen / work
iii)	大 / 犬	big/dog

Table 1.8 Sample characters that look alike

(e) Existence of regional dialects:

Many regional dialects exist in different parts of China. These dialects are so different from each other that it is virtually impossible to understand a dialect without intimate knowledge of that dialect. To complicate the problem, although ninety per cent of the characters are derived from the phonetic method, not all of these characters are pronounced according to their sound components owing to evolution and other reasons [Sue79].

[3] Major rules for stroking Chinese characters are: 1) stroke from top to bottom and 2) from left to right.

Chapter 2

Background

The main obstacle to machine processing of Chinese material is the lack of an effective input scheme owing to the problems inherent in the characters themselves. As Huang said "more than a hundred methods have been proposed in the past decade for solving the input problem [Hua85]." These methods can be classified into the following three categories:

(1) Direct input methods:

Input is via a special design keyboard, which normally requires one to three key strokes per character.

(2) Encoding methods:

Input is via a conventional keyboard or numerical pad using some encoding methods. These encoding methods are based on properties of the characters, such as the stroking sequences, and the structural or the phonetic information.

(3) Character recognition methods:

Input is via a tablet or an optical reader using a character recognition method to identify the hand-written or printed characters.

In this chapter, example systems in each of the above categories, together with their advantages and disvantages are discussed. Toward the end of this chapter, an overview (summary) of the Chinese character software system is presented.

2.1. Direct Input Methods

The main advantages of these input methods are their ease of implementation and the unique methods of coding. The Chinese systems in the early 70's were dominated by this approach. The designed keyboard is similar to a Chinese typewriter machine and the operation is like a conventional keyboard but with a much larger set of keys.

An example system is the "IPX 5486 Automatic Send/Receive (ASR) Telecommunications Terminal" by IPX Ideographix, Inc., U.S.A. The input of the characters in this system requires no more than three key strokes for each of the standard characters (9,600 characters). These characters are grouped by frequency of occurrence with 2,400 characters per group. Within each group, characters are listed in sequences of 37 Chinese phonetic symbols for easy identification [IPX].

In general, the direct input methods cannot be expanded easily and are incapable of allowing for the addition of new characters. They also make it difficult to locate a given character and thus require a long training period for memorizing the layout of the keyboard.

2.2. Special Encoding Methods

The methods in this category can be roughly divided into four types. The first type uses the stroking sequences of the characters, the second type uses the structural (radical) information while phonetic information is the basis of the third encoding method. The mixed strategy — using the phonetic and structural information — becomes the fourth type of encoding method.

The basic idea of using the stroking sequences is similar to the spelling of an English word. The characters are described by a string of stroke symbols. The experimental system by Yim et al [YPA83] is an example of a system using the stroking

sequences to design the input method. In their report, they pointed out that:

> "... A person with 12 years of education in Chinese can only stroke thirty to seventy percent of the characters correctly. ..." [YPA83] (page 22)

A consequence of this statement is that a more complicated parsing technique is required, allowing for multiple stroking sequences for a character. Nevertheless, the system is easily used by anyone who can write Chinese.

The "Tsang-chi Chinese character input method" [Chan] and the "Three Corner Coding Method (TCCM)" [HCH79] are the most popular encoding methods used in the commercial products (mini- and micro- computer systems). These methods use the structural information. The "ASL208 Chinese terminal[4] [Autom]" by Automated Systems (HK) Ltd. and the "IBM 5550 [IBM]" by IBM are terminals using the Tsang-chi encoding method. The Tsang-chi method has the merit of being implemented easily on an English keyboard. Twenty-five symbols (Tsang-chi alphabet[5]) are represented by the English alphabet from 'A' to 'Y'. A set of non-systematic rules are used in the process of selecting the alphabet sequences that represent the characters. The major problem of this method is that the rules are hard to follow. The length of the codes vary from one to five letters, thus making the manipulation of the characters more complicated.

TCCM is adopted and promoted by the Wang Co. Ltd., U.S.A., on WANG's Chinese Systems. This method is a more systematic coding method than others; a fundamental symbol chart (ten by ten) is used and a set of 100 major fundamental symbols and their minor counterparts are ordered on the chart[6]. Two principles, corner coding and shape coding, guide the selection of the three appropriate symbols appearing at each of the three corners of a given character. Each of these symbols has a two-

[4] In English mode, the terminal functions as a full VT100 terminal.
[5] see appendix A1
[6] see appendix A2

digit code number (determined by the row and the column values of that symbol in the table) which yields a six-digit code corresponding to the character being coded.

The third and fourth types of encoding methods are the Pinyin (phonetic encoding) and the Pinxxiee (phonetic + ideographic encoding) methods respectively. Both methods allow Chinese to be entered phonetically on a standard keyboard. The drawback to both methods is that the user has to know how to pronounce the characters.

A Pinyin system was implemented at Academia Sinica in China 1973. This system resolves a primary problem of the phonetic method — that is distinguishability among homonyms — by allowing the user to type in an additional word indicating the desired character. Thus a large vocabulary list was used in the system[7].

Tien has developed a Chinese word processor based on Pinxxiee [Tie82, Tie85]. The Pinxxiee formula defines each character precisely:

$$\text{Pinxxiee} = \text{Pinyin} + \text{tone}^{8} + \text{radical}$$

so that a Pinxxiee word exists for each character. A Pinxxiee word contains both the phonetic and radical information, that is, it consists of a Pinyin syllable with a radical suffix. The homonyms are differentiated by the radical suffixs. The disadvantage of this system is that the operator must know both Pinyin and composition radicals of the characters.

The possibility of collision in coding is unavoidable in most encoding methods and the interpretation of the encoding rules may differ from person to person, thus ambiguity may occur. Since none of the methods proposed are unanimously accepted and because of the lack of standardization, most of the commercial systems employ more than one input method.

---

[7] The source of information is from the paper by Fu et al [FuL79].

[8] Each Chinese character has four tones in pronunciation. These tones are (a) neutral, (b) rising, (c) falling-rising, and (d) falling tones [HCD82].

2.3. Character Recognition Methods

Chinese characters are rich in texture. To recognize them, six types of features can be extracted from Kanji and used as proposed by Suen [Sue83], they are (1) distribution of points, (2) transformation, (3) physical measurements, (4) edges and lines, (5) outline of character and (6) center-line of character. Usually, some of these features are combined to produce more distinctive features. A brief summary of some on-line character recognition methods is presented in Table 2.1.

Authors	*Features used/Classification Scheme*	*Results & Data Tested*
[ChH79] Chong et al.	Stroking order is imposed. Strokes are classified into four basic groups. The stroke (group) sequence is used to generate a key to be searched against a dictionary.	9529 characters recognition speed is 66 seconds for 211 characters (simulated study result).
[HaY80] Hanaki et al.	Character is coded into a symbol string using binary relation between stroke and reference zone. Direct Matching Recognition is used for characters that have up to five strokes. The Unit (radical) Structure Recognition is used for multi- stroke characters.	DMR method : 210 characters correctly recognized (rate about 95%). USR method : 400 characters.
[IYM81] Ikeda et al.	Stroke number is used as primary parameter. Character is coded into a stroke vector sequence and a positional vector sequence. One of the five type recognition (Matching) procedures is employed depending on how many strokes are in a character.	2000 characters, recognition rate from 90 to 98%
[Ara83] Arakawa	Fourier coefficients of pen-point movement loci relating to strokes are used as feature vectors of characters. Classification method is Bayesian Decision Rule. This method is effective for alphanumerics and Japanese characters only.	99.2% for Alphanumerics, 99.5% for Hiragana, 99.3% for Katakana, 95% for Chinese.
[TNM83] Takahashi et al.	A *priori* knowledge-based approach is formulated for stroke extraction. Each stroke is extracted precisely with reference to a *priori* knowledge about the stroke.	A 99.1% recognition rate was obtained by a recognition experiment in 2074 hand-printed characters by 30 writers.
[WaU83] Wakahara et al.	Stroke-number and Stroke-order free Recognition method by selective stroke linkage. Inter-stroke distances for all stroke pairs between an input character and a reference pattern are calculated.	1977 characters, 98.9% for both data including both variations, 99.8% for data including only stroke-order variation.

Table 2.1 Features used in on-line recognition schemes

2.4. Overview of the Chinese Character Software System

The *Chinese Character Software System* (CCSS) is designed and implemented on a VAX[9] 11/780 machine running the UNIX operating system (4.2 BSD). The CCSS system consists of two major components, a *Logical Chinese Terminal* (LCT) and a *pattern editor*. Both components (programs) are written in the C programming language. The supporting packages, WINDLIB[10] and FDB[11] are extensively used to ease the implementation.

The function of the LCT component is to serve as an interface between the user and the UNIX system. A graphics terminal (Jupiter), a keyboard and a tablet are needed for display and input respectively. The internal representation of Chinese characters in the CCSS system is designed in such a way that the internal codes for Chinese characters can be mixed with the ASCII codes for conventional characters. That in turn simplifies the editing requirements, for example pattern matching of Chinese and/or English characters (or words).

An *on-line hand-written character recognition* scheme is proposed and used for input of Chinese characters. The method allows novice users to input the square style characters by writing (drawing) via an input tablet. Another input method, the *Three Corner Coding Method* (TCCM)[12], is employed. In this method, a higher input speed can be achieved, but users have to memorize the TCCM encoding rules.

The display of Chinese characters is based on the hierarchical structure of the characters themselves. A Chinese ideographic *pattern dictionary* is implemented by using the FDB. The characters are regarded as graphic patterns. Each character is

---

[9] VAX is a trademark of Digital Equipment Corporation.

[10] WINDLIB is a resource based, general purpose, device independent graphics package by M. Green and N. Bridgeman [GrB84].

[11] FDB is a Frame Based Database System oriented towards scientific and engineering applications by M. Green [Gre82].

[12] The TCCM is invented by L. R. Hu, Y. W. Chang, and K. T. Huang in the late 70's [HCH79].

composed of radicals, and a radical consists of strokes and/or other radicals. The basic primitives of the patterns are strokes that are formed from the straight line segments. With the decreasing cost of memory and increasing CPU power, the approach of storing the hierarchical structure information, rather than the bit-map information of the characters, has become feasible and cost-effective.

The pattern-editor allows a user to create and add new characters into the pattern dictionary. Another function of the editor is to generate the recognition scheme for the newly formed pattern. The scheme is stored in a database called the *recognition dictionary,* that is also implemented by using the FDB.

To create/compose a new character pattern, editing is done by interactively positioning and selecting existing radicals and/or strokes[13]. A menu is employed for the selection of strokes because there are not many (less than 30) basic strokes. For the selection of radicals, a hand-written character recognition method is used to retrieve the desired radicals. Once the design of a new pattern has been completed, the pattern will be added to the pattern dictionary. A recognition scheme for that pattern is then automatically generated and stored in the recognition dictionary.

[13] Radical/stroke that is already created and stored in the dictionary.

Chapter 3

The Design

This chapter is concerned with the design of the following issues:

(1) The internal representation of Chinese characters;

(2) A display module;

(3) A recognition scheme for restricted hand-written square-style Chinese character;

(4) A logical Chinese terminal; and

(5) A pattern editor.

3.1. The Basic Design Requirements

In seeking a solution to a problem, one needs to identify the requirements. From these requirements, we can describe a possible solution to the problem in terms of a set of processes. Each process corresponds to a module that supports the corresponding capability. In this section, a set of requirements for the CCSS system is presented. They are:

(1) Modularity/Expandability

The CCSS system should be designed and implemented in such a way that it allows further functional expansion and change of individual components (modules). That is the expansion has to be feasible. To achieve expandability, the input, output and manipulation of data by the CCSS system should be considered as three separate information processing problems. In this way, testing and evaluation of ideas (methods) can be localized to individual modules.

(2) Dual-languages

The system should be able to handle both conventional characters as well as Chinese characters; otherwise the facilities provided by the UNIX operating system could not be directly used since the command language is expressed in terms

of English. The dual-languages system is also important because the translation from one language to another is not always practical and/or possible.

(3) Ease of use and User-friendly

Any system that is accessible only to highly trained experts may never become popular in today's competitive market. *Ease of use* becomes a major issue in designing systems. The CCSS system should therefore be easy enough to operate by both expert and novice[14] users. The system should be capable of using different input methods so that users will have various input methods to choose from. The term *user-friendliness*, in software design, is always a criterion and therefore it is also a requirement of the system.

(4) Incremental Expansion of the Character Set

The last but not the least important requirement is that the system should allow users to create new characters and develop his/her own character set (if desired) in an incremental base.

The processes involved to achieve these requirements are formulated and finally the software modules needed to support them are designed and implemented. These software modules are described in the sections that follow.

[14] For example, people who do not know Chinese, and/or people who know Chinese but are not familiar with input encoding methods.

3.2. Internal Representation of Chinese Characters

In this section, the specifications as well as the structure of the *Chinese-code* (internal codes of Chinese characters) are presented.

3.2.1. The Specifications

With the term *expandability* in mind, it is natural to specify the internal Chinese-code to be independent of any input and/or output method (I/O-independent) since no one I/O method can yet be considered as perfect, and requiring no further modification. As a result of I/O-independent internal coding, the stored data need not be changed when the input and/or output method is changed.

For Practical reasons, documents from business applications in Chinese speaking society are likely to be in both Chinese and English. No system can satisfy the needs of users in this domain (and others) if it can process only either Chinese or English but not both simultaneously. Therefore, the Chinese-codes have to be distinguished from the internal codes (ASCII) of conventional characters, so that they can be mixed within a file. In other words, a file can contain both Chinese and English characters.

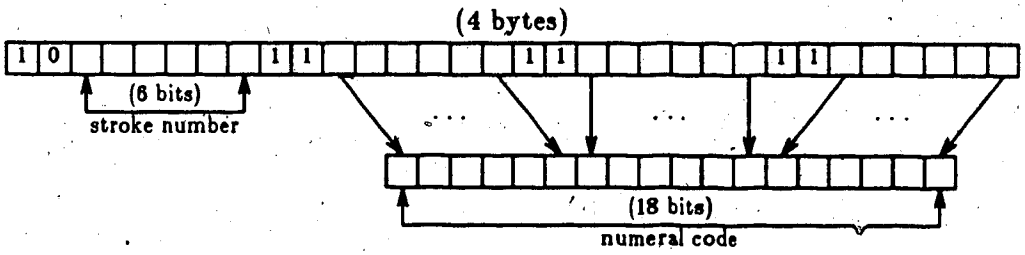
Additionally, the Chinese-codes should also support editing requirements. The issues such as sorting/dictionary-ordering and searching should be fulfilled. Processing of English text has been evolving for thirty years and we have learned what functions are wanted. In principle, for processing of Chinese text, these functions remain the same.

3.2.2. The Structure of Chinese-code

A file[15] in the UNIX system is a sequence of bytes [KeP84, WaC83]. Depending on interpretation, the meaning of the bytes in a file can be treated as either Chinese or English. As described by Kernighan and Pike:

> "... No structure is imposed on a file by the system, and no meaning is attached to its contents — the meaning of the bytes depends solely on the programs that interpret the file. Furthermore, as we shall see, this is true not just of disc files but of peripheral devices as well. Magnetic tapes, mail messages, characters typed on the keyboard, line printer output, data flowing in pipes — each of these files is just a sequence of bytes as far as the system and the programs in it are concerned." [KeP84] (page 41)

To separate the Chinese and ASCII codes, a special character, or a string of characters, can be used to serve as the *delimiter*. The major difficulty associated with this method is that the *file access* must be sequential (to maintain correct interpretation of the data). To support a file access such as *random*, the codes themselves must be differentiable. One appropriate method to obtain differentiable codes is by using the seventh bit of the bytes (counting from right to left); setting this bit to '1' or '0' the bytes (data) can be interpreted as Chinese or ASCII codes respectively.



stroke number: range from 1 to 63
numeral code: range from 1 to 262143

Figure 3.1 The format of the Chinese-code

[15] Meaning those ordinary data files such as text files or binary files.

The internal code for each of the Chinese characters uses/needs four bytes. Figure 3.1 shows the format of the Chinese-code. To identify Chinese characters, a numerical I/O-independent an encoding scheme is adopted; each character is assigned a unique number. Since the number for each character is arbitrarily assigned and normally does not contain any information for dictionary-ordering, a *prefix* can be added for dictionary-ordering purposes. In the proposed structure, the prefix-byte contains the stroke number of the encoded character, so the codes can easily be sorted in stroke-number ordering (a method of Chinese dictionary ordering).

To distinguish the starting position of Chinese-codes, the sixth bit is set to '0' and '1', for the first and subsequent bytes respectively. Hence, by examining the highest two bits of a byte, the data can be immediately identified.

The characteristics of the structure (Chinese-codes) are summarized below:

(1) The codes are I/O-independent;

(2) Equal length internal codes (four bytes);

(3) The codes themselves are distinguishable from ASCII codes;

(4) The starting and subsequent bytes within a Chinese-code are distinguished by a prefix code;

(5) The codes can be sorted according to the stroke number from one to sixty-three;

(6) The codes have the capability of identifying 262,143 characters (patterns).

3.3. The Display Module

In this section, a *display module* is described. This module uses the structural information of Chinese characters to generate and output the vector format (coordinate of the end points of the line segments that formed the pattern) of the desired character.

Chinese characters are ideographs, and each can be regarded as a graphic pattern. The whole character set can be represented by a *geometric model* that describes entities with inherent geometrical properties, and thus lend themselves naturally to graphical representation. Ghinese characters have a hierarchical structure (shown in Figure 1.1) that is influenced by a bottom up construction process. The pattern primitives are strokes, that can be combined into various radicals, which in turn can be combined into different characters.

Three databases are employed in the display module: the *stroke database*, the *radical database* and the *character database*. Different information is stored in these databases. Figure 3.2 shows the basic block diagram of the module.

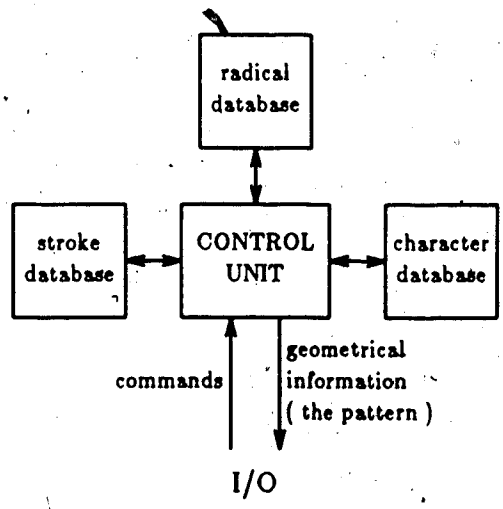


Figure 3.2 The block diagram of the display module

The stroke database stores information on how strokes are generated from display

primitives (in our case, the display primitives are straight lines). The radical database stores information on how radicals are generated from strokes, and/or other radicals. Finally, the information on how characters are generated from radicals is stored in the character database. Appendix A3 shows an example of information stored in these databases for the Chinese word *language,* displayed in Figure 1.2.

The stroke, radical and character patterns are all defined in their own co-ordinate system (the co-ordinate system used is a 64 by 64 matrix space). A function of the "control unit" does the necessary retrieval and geometric transformation of the information for building the needed character pattern, and to output the corresponding pattern to the desired receiver.

The reason for the decision of using three separate databases is that, in principle, changes in the style (font) of the strokes will automatically be propagated to all higher level components, the radicals and the characters. Therefore, to change the style of the character set, only the stroke and/or the radical databases need to be changed.

The main advantage of the module is that the generation of Chinese characters in different output forms becomes much easier. Much redundant work can be reduced; for example, by using the appropriate transformation matrices applied to the final master co-ordinate system, different character styles and sizes can be obtained successfully. No additional pattern storage capacity is required here and this will be specifically beneficial for type-setting purposes [Kia82]. Figure 3.3 shows some examples of different styles and sizes of Chinese characters. These patterns can be used directly in CRT display. They also can be easily transformed into dot-matrices (but not vice versa!) for applications to matrix printers.

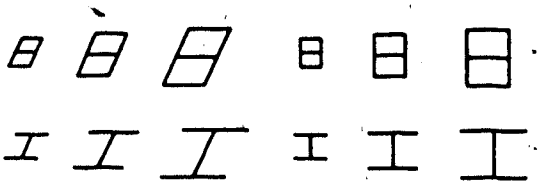


Figure 3.3 Examples of different styles and sizes of Chinese characters

Another advantage is that the construction process of the character set is systematic. A *syntactic description language* describes the geometric relationships of the strokes and the radicals of the characters. The languages proposed by Naganashi et al. [NNN83], or by Huang and Suen [HuS83] can be used for this purpose. However, in this thesis, an interactive approach is proposed; it is presented in Section 3.5. The approach reduces the required user memory, for storing the construction rules when using the syntactic description language method. The immediate feedback of user actions by this method eases the overall construction process.

3.4. Hand-written Chinese Character Recognition Scheme

In this section, a character recognition scheme is described. It is designed to recognize the restricted hand-written square-style Chinese characters. The word "restricted" is used here to emphasize that the characters have to be properly written. The current scheme cannot recognize rough and distorted characters. A block diagram of the recognition scheme is illustrated in Figure 3.4.

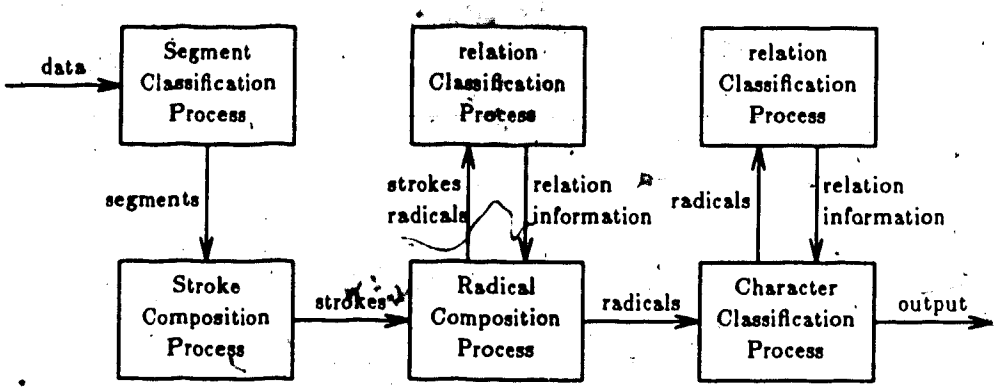


Figure 3.4 The block diagram of the recognition scheme

The recognition process is systematically divided into four stages:

(1) Segment classification;

(2) Stroke composition;

(3) Radical composition; and finally,

(4) Character classification.

The basic idea of the scheme is to compose the segments into strokes[16], strokes into radicals and finally into the appropriate character according to the information (positional relationships) stored in the character recognition dictionary. Both the stroke connecting point and the stroke relative position are important keys for identifying characters.

[16] In this thesis, a stroke is divided into segments at the point where direction of pen motion changes.

3.4.1. Strategy of Character Recognition

For hand-written character data acquisition, a data input tablet can be used. By tracing the data (pen-point movement loci) from the tablet during the writing of a character, the number of composite strokes/segments with their starting and ending positions can be determined. In the proposed recognition scheme, an assumption is made that data acquisition is started from the two end points of each segment. The preprocessing (to separate the data into segments), such as the smoothing operation and the turning motion detection of the pen-point movement loci, is excluded from the scheme. The reason for the exclusion is because the technique or algorithm for segment identification is device dependent. In the CCSS system, the technique to obtain the two end point positions of each segment is the interactive method provided by the graphics package: WINDLIB.

To achieve a better performance, a stable and compact representation of the characters must be found. The previous work tells that the hierarchical structure (used in the display module) of the characters is a stable and compact representation. With the raw data gathered during the writing of an input character, the recognition process performs something like the inverse operation on the hierarchical structure. That is, it composes the primitive data into intermediate data (strokes and radicals) and finally into the appropriate character.

In on-line Chinese character recognition systems [ChH79,IYM81,NMA83], the positional information and the stroking sequence are the important keys used to determine the input character. When the stroking sequence is used as the primary identification key, the recognition process will be similar to the searching process of a spelled English word from a dictionary. But, because of the existence among writers of multiple stroking sequences within a character, the recognition process becomes more difficult and users are greatly restricted. Figure 3.5 shows an example of the different

writing sequences of the Chinese character *day*.

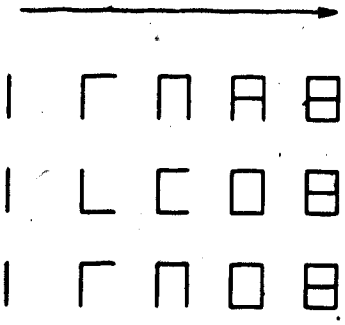


Figure 3.5 Some writing sequences of the Chinese character: day

To complicate the problem, the existence of the stroke number variation, that is the splitting of a single stroke and concatenation of successive strokes, makes the recognition harder and it may even fail as a result. Owing to the above problems, the information of the stroking sequence is dropped and the positional information is used to determine the input character. The positional relationship between strokes and radicals is stable among characters written by different writers.

In the proposed recognition scheme, a four stage recognition method, based on a syntactic approach, is adopted. A block diagram of this scheme is shown in Figure 3.4. The input data are segments and the total number is static. A character is entered into the recognition process by a sequence of X-Y co-ordinates, that is the two end point positions of the segments. This sequence is classified into the four segment-types described in Section 3.4.2 using the regional classification method.

In the second stage, the segment sequence is first sorted by segment-type order, then according to the position of the segments within the same group. This sorting operation is used to maintain consistency in the composition process. That is, the appropriate segments can be composed into the same stroke-types regardless of the writing sequence. The segments are then converted into strokes. With reference to *a priori* knowledge stored in the recognition dictionary, the process is independent of the

users' knowledge about the strokes. Hence, the desired stroke number variation and stroke order variation are achieved. The user only needs to write/draw the characters, with or without intimate knowledge of the formating strokes.

Once the strokes of an input character are determined, a sequence of them is sought in the recognition dictionary for radicals. A digital search tree structure is employed in the dictionary. The composing sequence is the searching path of the tree. By matching the positional relationship of successive strokes, they can be grouped into radicals. Ambiguity arises (and is unavoidable!) when there is more than one radical made up of the same sequence, with identical positional relationships, of successive strokes. A difference in positional relationship between a pair of strokes within the two ambiguous radicals is used as the branching condition when searching the tree for the target radical; hence, the ambiguity is resolved.

In the final stage, the input character is classified. The process is similar to the radical composition. A digital search tree is employed and the relative positions of the successive radicals are used as the branching conditions in searching the tree to determine the input character.

3.4.2. Segment Classification

A Chinese character can be viewed as a two dimensional arrangement of strokes. In modern writing, a stroke can be decomposed into one or more segments; they are horizontal, vertical, diagonal or anti-diagonal (shown in Figure 3.6). The writing direction of the strokes is not used in the proposed recognition scheme, to relax the restriction on proper stroking direction of Chinese characters. Hence, non-Chinese writers can also use the system by viewing the characters as a two dimensional arrangement of lines.

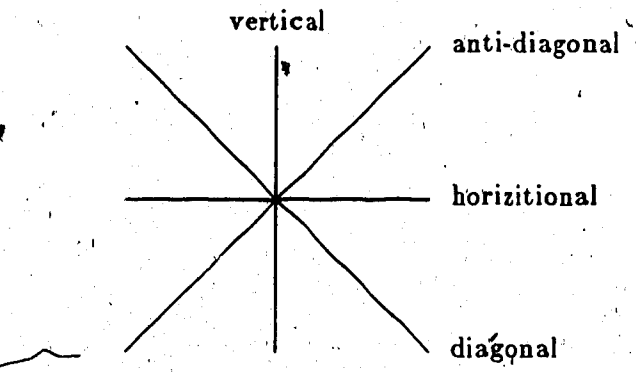
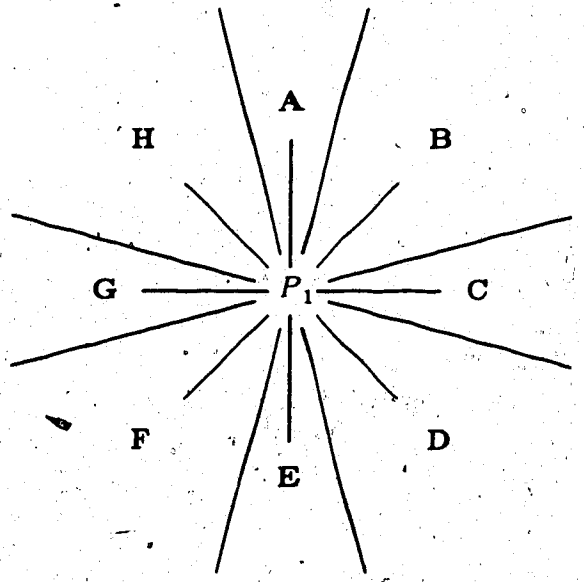


Figure 3.6 The basic segment types



Region {A, C, E, G} are 30° sectors

Region {B, D, F, H} are 60° sectors

Figure 3.7 Regional classification of segments

The character *day*, shown in Figure 3.5, can be decomposed into three horizontal segments and two vertical segments regardless of the writing sequences. To determine the type of the segments, the regional determination method is employed. Given the two end point positions, P_1 and P_2, of a segment, to determine the *segment-type*, the

following test statements are used: (with the point P_1 at the center, see Figure 3.7)

- if ($P_1 = P_2 \pm \delta$) then segment_type = VERTICAL /* segment is too short */
- else if $P_2 \in$ (region A or E) then segment_type = VERTICAL
- else if $P_2 \in$ (region B or F) then segment_type = ANTI-DIAGONAL
- else if $P_2 \in$ (region C or G) then segment_type = HORIZONTAL
- else if $P_2 \in$ (region D or H) then segment_type = DIAGONAL

When writing a vertical or horizontal segment, the variation of angle measured from the axis (vertical or horizontal) is expected to be smaller than the angle variation from the writing of a diagonal/anti-diagonal segment. Therefore, instead of dividing into equal regions, the vertical and horizontal segment regions are set to 30° sectors, and the diagonal and anti-diagonal segment regions are set to 60° sectors.

3.4.3. Stroke Composition

Once the segments are classified, the *stroke composition* process is started. The function of the process is to compose the segments into basic strokes according to a *priori* knowledge about the strokes stored in the recognition dictionary. A digital search tree (trie) structure is employed; an example of the basic strokes is shown in Figure 3.8 and the tree for these strokes is shown in Figure 3.9.

The first step of the process is to perform the sorting operation. The segments are sorted into segment-type order, and into an order based on the starting position of the segments within the same segment-type. The starting position of the vertical, anti-diagonal and diagonal segments are from top to bottom while the horizontal segment is from left to right. After the sorting operation, the conversion is begun. The composing sequence is the searching path of the digital search tree.

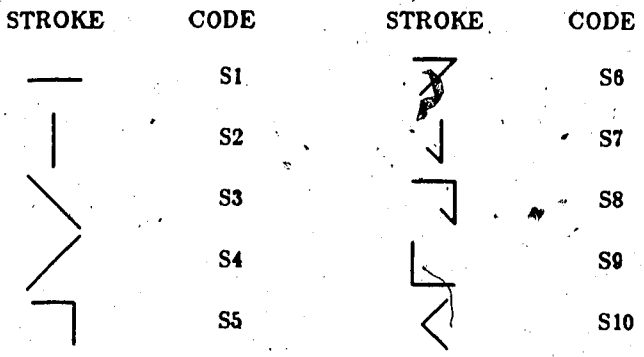


Figure 3.8 Some examples of basic strokes

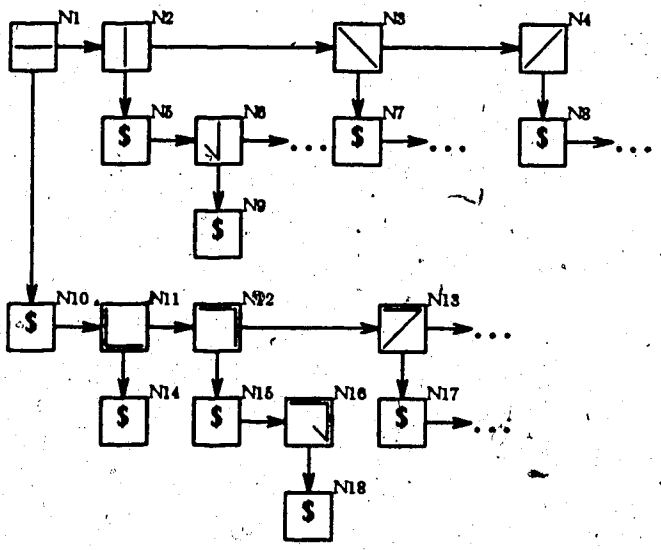


Figure 3.9 The digital search tree for stroke composition

To explain the conversion process, we will describe how to search the tree (Figure 3.9). Suppose we are given the stroke "S8" as the object of the search, see Figure 3.8. We consider each of the segments *horizontal, vertical, diagonal* in turn, starting at the first level of the tree and proceeding as follows. Follow the node labeled 'N1' and going down one level to node-'N10'; at this node the segment is set to stroke-type 'S1' then follow the node-'N11'. At this node, a *vertical* segment with the end point connected to the starting point of the stroke is searched from the vertical-segment list and

not found. We then proceed to node-'N12'. Again the search is repeated and this time we found the required segment, the process is advanced to node-'N15'. The stroke is set to 'S4' and branched to node-'N16'; at this node, the search is again repeated and looking for the diagonal segment with the end point connected to the end point of the stroke. Since we found the segment the search therefore goes down a level to node-'N18'. The process is continuous and finally terminated with the segments grouped into stroke type 'S8'.

It is important to notice that the composition only depends on the *shape* of the strokes and the segments will be grouped into the compound strokes made up of maximum segments. For example, the stroke "S8" could be grouped into two stroke types, "S3"+"S5" or "S1"+"S7". To reduce the possible multiple composition sequences, the resulting stroke will always be "S8", instead of the above mentioned stroke combinations.

The result of the stroke composition process is a set of stroke-types and each stroke-type contains the following information: (1) the stroke-code, (2) the starting and ending points of the stroke; and finally, (3) the bottom left and the top right corner positions (the maximum boundaries) of the stroke.

3.4.4. Radical Composition and Character Classification

A radical is made up of one or more strokes. Figure 3.10 shows some of the basic radicals. The method for the radical composition is similar to the stroke composition. The positional relationship between successive strokes is computed by the "positional relation classification" process; the starting and ending positions of the strokes are used as the reference points when computing the relationships. This sequence of relationships is stored in the dictionary and used as the branching conditions during the conversion process. Some of the positional relationships between strokes and radicals are shown in Figure 3.11.

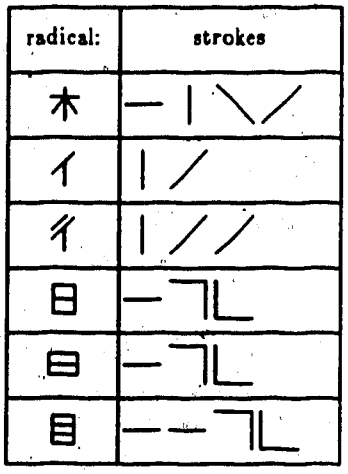


Figure 3.10 Some of the basic radicals

Stroke relations

relation	example
cross	
touch	
longer	
apart	

--- first stroke

— second stroke

Radical relations

relation	example
left of	A B
above	A / B
inside	A B

A: first radical

B: second radical

Figure 3.11 Some positional relationships of strokes and radicals

Ambiguity arises when more than one radical composed by the same sequence with identical positional relationship of successive strokes. To resolve the ambiguities, the (downward) branching when searching the tree, is resolved by the relationship between two different strokes which gives a difference between the two ambiguous radicals. An assumption is made in the ambiguity resolution process; a difference in positional relationship, computed by the proposed relation classification method, existed between at least a pair of strokes within any two ambiguous radicals. The above

assumption is valid for most, if not all, ambiguous characters.

The result of the radical composition process is a set of radical-types and each radical-type contains the following information: (1) the radical-code, (2) the bottom left and the top right corner positions (the maximum boundaries) of the radical.

A Chinese character is made up of one or more radicals. The "character classification" process is similar to the radical composition process. A digital search tree is employed, in which the branching conditions are the positional relationship between successive radicals. For relationship between radicals, the maximum boundaries, that is the bottom left and the top right corner positions of the radicals are used as the reference points. A simple but effective method is devised for the "positional relation classification" for both strokes and radicals. This method is described in the section that follows.

3.4.4.1. Positional Relation Classification Scheme

The features (connective relationships and relative location of strokes) are important information needed to identify a radical or a character. In this section, a classification scheme based on the above features is presented. For the sake of extracting sufficient information but avoiding excessive computation, the positional relationships are classified into eight relation types shown in Figure 3.12. To explain the scheme, we will describe how to perform the classification process.

INPUT:

two units U_1 and U_2; each unit has the following information, where $i \in \{ 1, 2 \}$.

- T_i:

  represents the type of the unit, either stroke or radical;

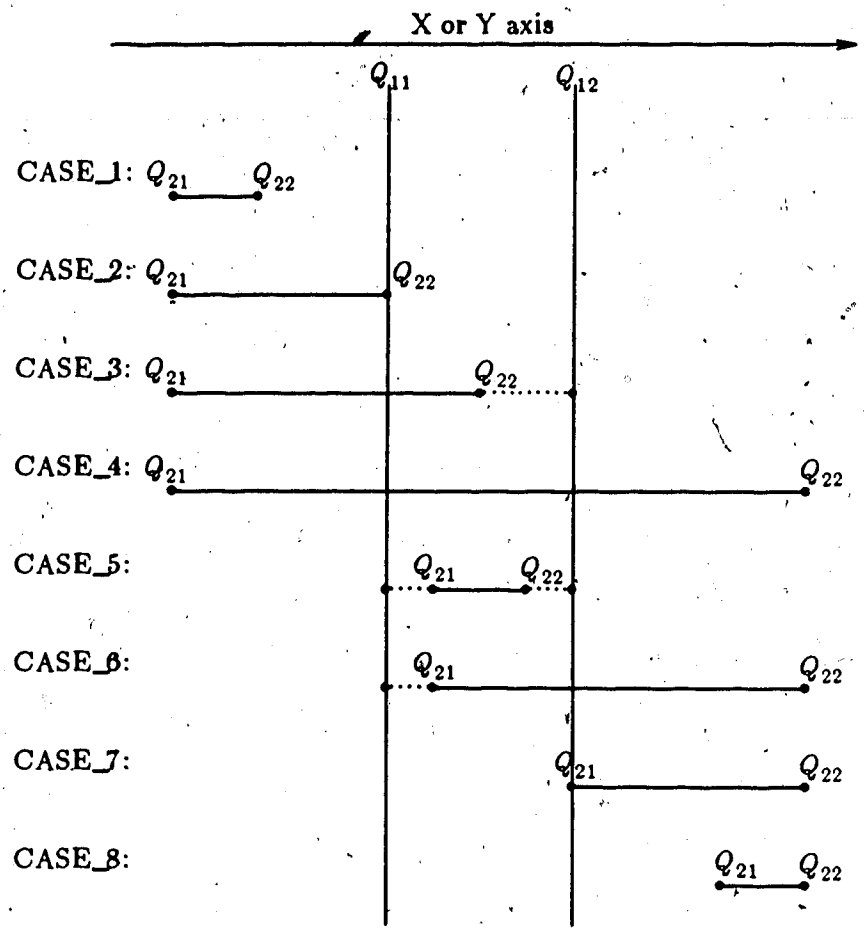
- P_{i1}:

  is the starting point of the stroke, or the bottom left corner position of the

radical.

- P_{i2}:

  is the ending point of the stroke, or the top right corner position of the radical.



N.B. The POINT can be in anywhere along the dotted line
$Q_{i1} \leq Q_{i2}$, where $i \in \{1, 2\}$.

For the X projection:	For the Y projection:
$Q_{i1} = \min(P_{i1}(x), P_{i2}(x))$	$Q_{i1} = \min(P_{i1}(y), P_{i2}(y))$
$Q_{i2} = \max(P_{i1}(x), P_{i2}(x))$	$Q_{i2} = \max(P_{i1}(y), P_{i2}(y))$

Figure 3.12 The eight types of positional relationships

To compute the positional relationship between these two units (strokes or radicals), we perform the following steps:

- Step one:

  compute the relationship between U_1 and U_2 in the X-projection dimension. That is, the values $P_{ij}(X)$ are used as the reference points in computing the relationship, where i, j $\in$ { 1, 2 }.

- Step two:

  compute the relationship between U_1 and U_2 in the Y-projection dimension. That is, the values $P_{ij}(Y)$ are used as the reference points in computing the relationship.

- Step three:

  if (either T_1 or T_2) is NOT a radical-type

  then RETURN the relationships computed in steps one and two.

- Step four:

  compute the radical relationship (see Figure 3.11 for radical relationships) using the results of steps one and two and RETURN the relationship.

In step one or two, the relationship is computed by the following pseudo C function code:

```
compute_relation( Q11, Q12, Q21, Q22 )
    int Q11, Q12, Q21, Q22;
{
    if ( Q22 < Q11 )
        then return( CASE_1 );
    else if ( Q22 = Q11 )
        then return( CASE_2 );
    else if ( ( Q21 < Q11 ) && ( Q22 <= Q12 ) )
        then return( CASE_3 );
    else if ( ( Q21 < Q11 ) && ( Q22 > Q12 ) )
        then return( CASE_4 );
    else if ( Q22 <= Q12 ) )
        then return( CASE_5 );
    else if ( Q21 < Q12 )
        then return( CASE_6 );
    else if ( Q21 = Q12 )
        then return( CASE_7 );
    return( CASE_8 ); /* ( Q21 > Q12 ) */
```

}

The overall possible combined relation types between any pair of strokes is sixty-four. That is, the eight primary relation types in the X-projection dimension times the eight primary relation types in the Y-projection dimension. In step four, the radical-relation is computed by the following pseudo C function code:

```
radical_relation( R1, R2 )
    int R1, R2;
        /* R1 is the RELATION computed in step one;
           R2 is the RELATION computed in step two. */
{
    int Left_Right_Relation, Below_Above_Relation, In_Out_Relation,
        RELATION;

/* compute the Left_Right relation */
    if ( ( R1 = CASE_1 ) || ( R1 = CASE_2 ) )
        then Left_Right_Relation = LEFT_RELATION;
    else if ( R1 = CASE_7 ) || ( R1 = CASE_8 ) )
        then Left_Right_Relation = RIGHT_RELATION;
    else Left_Right_Relation = MID_RELATION;
/* compute the Below_Above relation */
    if ( ( R2 = CASE_1 ) || ( R2 = CASE_2 ) )
        then Below_Above_Relation = BELOW_RELATION;
    else if ( R2 = CASE_7 ) || ( R2 = CASE_8 ) )
        then BELOW_ABOVE_Relation = ABOVE_RELATION;
    else BELOW_ABOVE_Relation = MID_RELATION;
/* compute the Inside_Outside relation */
    if ( ( R1 = CASE_5 ) && ( R2 = CASE_5 ) )
        then In_Out_Relation = IN_RELATION;
    else if ( ( R1 = CASE_4 ) && ( R2 = CASE_4 ) )
        then In_Out_Relation = OUT_RELATION;
    else In_Out_Relation = MID_RELATION;
/* combining the *_Relations together */
    RELATION = Left_Right_Relation +
               10 * Below_Above_Relation +
               100 * In_Out_Relation;
    return( RELATION );
}
```

For relationships between pairs of radicals, the primary possible relation types are reduced to three, the left of (right of), the above (below), and the inside (outside) relation types. The overall possible combined relation types between radicals is eleven.

This reduction (in the overall possible combined relation types for radicals) increases the allowable variation in the writing of the characters.

3.4.4.2. Ambiguity Resolution

Problems arise because of the limited information used in the basic recognition scheme.

(1) In the basic recognition scheme, the positional relationships of successive strokes are used to identify radicals. It is possible that more than one radical is made up of the same sequence with identical positional relationships of successive strokes. Hence, an ambiguity can arise.

(2) The radicals/characters, shown in Table 1.8 example (i), are not distinguishable owing to the lack of relationship between X and Y dimensions. The positional relationship between every pair of strokes for these two radicals is the same. The only radicals found in this category (so far) are the above-mentioned examples.

These counter arguments may require changing the basic recognition scheme. However, the change is not necessary because the expected total number of the ambiguous radicals is small and the problems can therefore be solved effectively by using a *resolution table*. For problem (1), the relationship between another pair of strokes that provide a positive identification, is used as the (downward) branching condition. For problem (2), a routine could be added to examine the relationships between the X and Y dimensions of strokes within the radicals/characters that are not uniquely determined by the positional relationships. To avoid excessive computation, the ratio between the X and Y dimensions of the final radical is computed. That is, the ratio between (P_{i2}(X) − P_{i1}(X)) and (P_{i2}(Y) − P_{i1}(Y)) is used to determine the correct radical. The ratio is classified into the following three types: equal_to_one, greater_than_one and less_than_one, and it is used as the branching condition.

3.5. The Logical Chinese Terminal and the Pattern Editor

This section describes the functions and design principles of the *Logical Chinese Terminal* (LCT) and the *Pattern Editor*. The LCT is designed to input and output Chinese characters as well as conventional characters; its primary function is to serve as an interface between users and the UNIX system. The word "logical" is used here to emphasize that the LCT is a logical terminal, that is, it is not a physical Chinese terminal. The pattern editor is designed to allow users to create the display patterns of Chinese characters. It is also used to generate automatically the recognition schemes of the designated characters.

A graphical interactive approach to the design of both components has been adopted. When the user interface modules of both components are designed, emphasis is put on the design principles described in Chapter Six of the book entitled "Fundamentals of Interactive Computer Graphics" by Foley and Van Dam [FoD83].

(1) Give feedback to users' input

Feedback is an essential ingredient in human-computer interface. With feedback, the user knows what he/she (or the system) is currently doing, hence avoiding confusion.

(2) Help users to learn the system

This will increase the productivity since the user would then know how to act correctly with minimal effort.

(3) Accommodate errors

Everyone makes mistakes when working with computers, therefore the system has to allow for correcting mistakes whenever possible; thus not making things progressively worse.

(4) Design for consistency

Consistency eases the user in learning the system, and also eases implementation

in most cases.

(5) Structure the display

Structuring the display allows the user to locate relevant information and easily understand the types of information that are presented.

(6) Minimize memorization

This principle releases the user from thinking heavily about how to use the system; hence the user can concentrate on the actual work.

3.5.1. Designing the LCT

As a logical I/O device, the LCT provides basic functions such as cursor movement control and screen control functions. The cursor movement control functions, besides the backspace, carriage return and line-feed functions, include absolute positioning of the cursor and screen oriented (visual) display editing. The screen control functions include a command to allow the change of screen mode. Three screen modes, the scroll mode, the full page mode and the half page mode are provided. Page modes are provided because scrolling cannot be implemented efficiently in the selected hardware and software facilities (Jupiter and WINDLIB). The clear screen command and the local echo on/off command are also included in the screen control functions.

To increase the chance of serving a larger user population, two input methods, the Three Corner Coding Method (TCCM) and an on-line hand-written recognition method are currently used. Two advantages of the TCCM input method are its syntactic simplicity and its excellent systematic coding method. Furthermore, it is easy to implement. However, like other encoding methods, it requires substantial (amount of) memorization. Users have to memorize the codes corresponding to the symbols in the fundamental chart. This method is aimed only at professional (frequent) users; while the recognition method is aimed at novice users. The reason for choosing the

by novice users, but also because it requires no special memorization. In principle, users need only draw (write) the characters at an input tablet, with or without knowing the meaning of the formating strokes and/or radicals of the characters.

The LCT allows application programs to use its interaction techniques. Two functions are currently provided. The first function allows application programs to define commands. The second function allows them to display help/error messages. The techniques for selecting those commands and for displaying those messages are governed by the LCT. The basic idea is to let the LCT handle the interaction. By taking this approach it is feasible to unify the management and maintenance of user interfaces that are transparent to application programs.

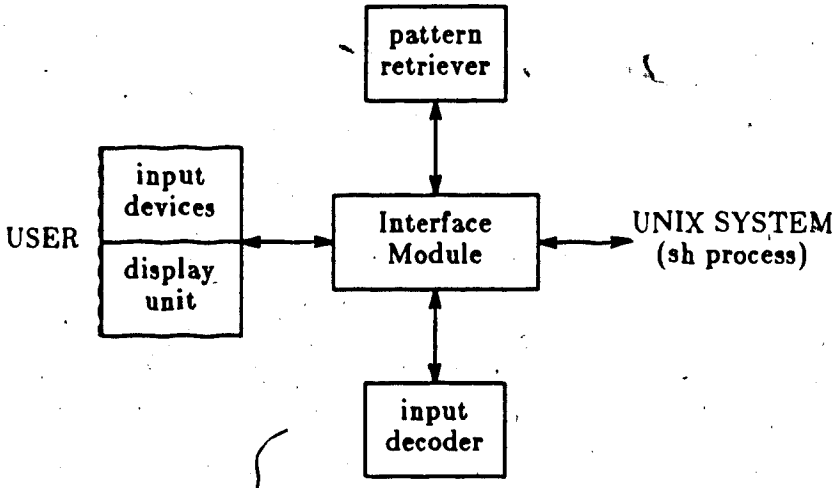


Figure 3.13 The block diagram of the LCT

A block diagram of the LCT is illustrated in Figure 3.13. The LCT is logically divided into modules, the *user interface* module, the *input decoder* module and the *pattern retriever* module. Each module corresponds to a specific subset of the required functions. By dividing the component into functional modules, the implementation phase can be carried out in a stepwise manner by establishing milestones during the

eases the maintenance of the component. Moreover, since all device dependencies are limited to the interface module, the other modules can be used in other applications without modification. For example, the pattern retriever can be used in Chinese text formating applications.

To round up this section and to provide an overview of the LCT, a summary of its general operations is presented as follows. The interface module controls the commands/data flow between the user, an application program and the functional modules. It performs functions such as converting raw input data into the required form and performing the dialog control. The pattern retriever functionally corresponds to the display module described in Section 3.3. It receives a Chinese-code from the interface module and returns the corresponding pattern. The input decoder is used to translate the TCCM codes into the corresponding Chinese-codes. It also performs the recognition process of Chinese characters that was presented in the previous section.

3.5.2. Designing the Pattern Editor

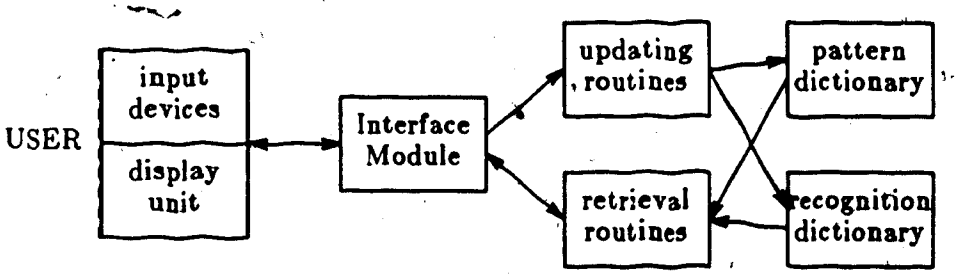


Figure 3.14 The block diagram of the editor

Figure 3.14 shows a block diagram of the *pattern editor*. A graphical interactive approach to the design of the editor has been adopted. This editor allows users to define the display patterns (strokes, radicals and characters) and to generate the

Both the *pattern* and *recognition* dictionaries are based on the hierarchical structure of the characters. When a display pattern is defined, it can be either a stroke, a radical or a character. This pattern is used as the reference pattern in the process of generating the recognition scheme. However, the order of creation for the sets of strokes, radicals and characters is not fixed. Allowing users to define the patterns (strokes, radicals and characters) in any order introduces a serious problem in the ordering of the recognition schemes. The problem could cause the failure to recognize some of the radicals/characters. This can be illustrated where some of the radicals that contain the stroke type "S8" (see Figure 3.8) are created before the creation of the stroke "S8". This will cause the composition of those radicals to be with the stroke types "S3" and "S5". The stroke types sent to the radical composition process will no longer be "S3" and "S5" after the creation of stroke "S8", hence those radicals will not be recognized.

To ensure consistency and integrity among data, it is necessary to perform checking on all radicals and characters when a new pattern is defined. However, we only perform such checking after each updating session or upon the user's request for performance considerations. A pseudo C code of the function that performs the checking and correction operations is shown below.

```
checking()
{
    PATTERN_PTR radical_pattern, char_pattern;
    int i, R_i, C_i;
    char modified = FALSE;

    /* checking the radicals */
    for ( i = 1; R_i ≠ LAST_RADICAL; i++ )
    {
        /* get the pattern from the pattern dictionary */
        radical_pattern = get_pattern( R_i );
        /* if we cannot recognize the R_i
           then perform the correction */
        if ( R_i ≠ rad_recognize( radical_pattern ) )
        {
            add_radical( radical_pattern );
```

```
                modified = TRUE;
            }
        }
        /* checking the characters */
        if ( modified ) for ( i = 1; C_i ≠ LAST_CHARACTER; i++ )
        {
            char_pattern = get_pattern( C_i );
            if ( C_i ≠ char_recognize( char_pattern ) )
                add_character( char_pattern );
        }
    }
```

To define the display pattern of a character, the user must specify the component radicals. To release users from the heavy burden of detailed specification of the internal radical-codes, an on-line hand-written character recognition technique for selecting the desired radicals is employed. The user positions the desired radical by specifying the opposing corners of its boundary rectangle. He/she may then select the radical by drawing (writing) it. As a result, the internal codes are transparent to users. A menu is used to display the available strokes because there is only a limited number of possible strokes (less than 30).

Chapter 4

The Implementation

4.1. Implementation Environment

The CCSS is implemented on a VAX 11/780 machine running the UNIX operating system (4.2 BSD) and is written in the "C" programming language. Both components, the *Logical Chinese Terminal* and the *Pattern Editor*, need a graphics terminal for display, a keyboard and a tablet for input. To ease the overall implementation, the supporting packages, WINDLIB and FDB, are used extensively.

WINDLIB is an event driven window based device independent graphics package. A "window" in WINDLIB defines an abstract co-ordinate space and a set of attributes. These attributes determine how and where the picture (data) is displayed. An event handler can be associated with each window and is used to handle the interaction with the user and controls all the operations within that window. This basic window management idea behind WINDLIB provides a powerful tool for designing a good user interface. The detailed description of WINDLIB can be found in [GrB84].

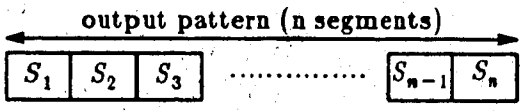
FDB is a Frame Based database, oriented towards scientific and engineering applications. Its structure is similar to a pointer type data structure. In FDB the entities are stored in frames and a frame is a collection of slots. Each slot has a name and a value. The value of a slot can be the name of another frame (slot becomes a pointer) or a primitive value such as an integer, a real number, a string or a "C" data structure. The data structure that has been designated for the pattern dictionary and the character recognition dictionary can be used within FDB without modification. The detailed description of FDB can be found in [Gre82].

4.2. Implementation of the LCT

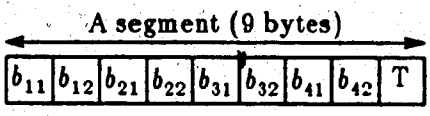
The LCT is divided into three modules as described in the last chapter. Each module corresponds to a specific subset of the required functions. The implementation of these modules is presented in the following three sections.

4.2.1. The Pattern Retriever Module

The function of this module is to retrieve the display patterns of Chinese characters from the pattern dictionary by giving their internal codes. The current version performs only the standard transformations, that is, to generate the normal patterns. Standard I/O channels are used. The module receives a Chinese-code from the input channel and outputs the corresponding pattern to its output channel. Figure 4.1 shows the structure of the output data.



S_k : the k^{th} segment of the output pattern



where: $b_{ij} \in \{0..9\}$

$b_{11}b_{12} = P_1(x)$
$b_{21}b_{22} = P_1(y)$
P_1 = starting position of the output segment

$b_{31}b_{32} = P_2(x)$
$b_{41}b_{42} = P_2(y)$
P_2 = ending position of the output segment

for intermediate segments: T = ';'
for the final segment: T = '.'

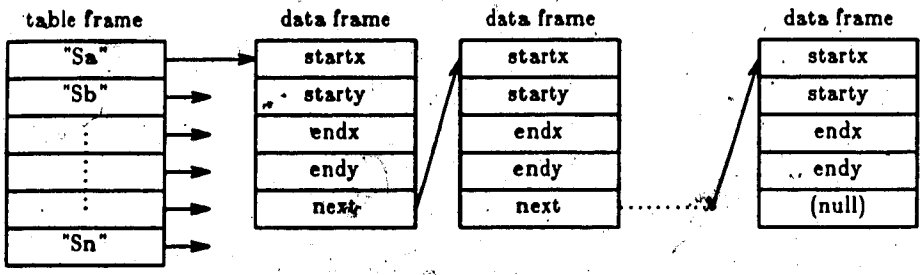
Figure 4.1 The structure of output patterns

Whenever an error is encountered during the retrieval process, the process will be stopped. A *cross* (the output data are: 00006464;00646400.) is sent as the output pat-

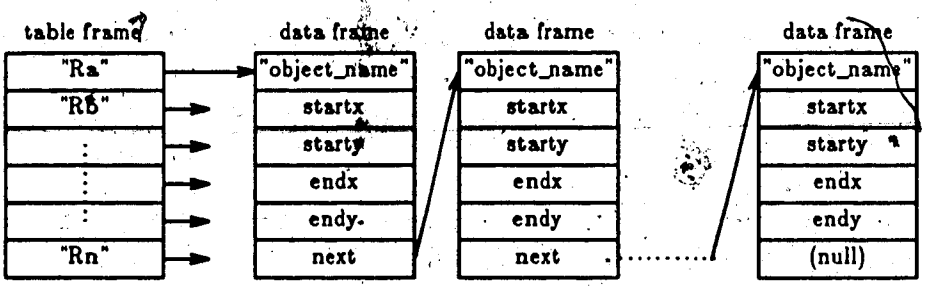
tern to indicate that an error occurred.

4.2.1.1. Data Structure of the Pattern Dictionary

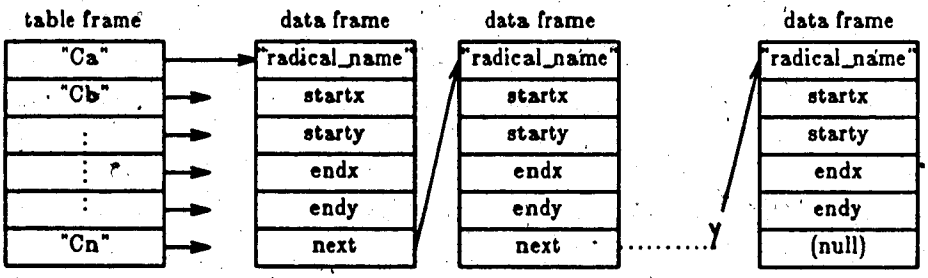
The pattern dictionary is implemented using the FDB. The basic organization of the dictionary is divided into three separate databases, the stroke, radical and character databases. Figure 4.2(a,b,c) show the organization of these databases respectively.



(a): The organization of the stroke database



(b): The organization of the radical database



(c): The organization of the character database

Figure 4.2 Organization of the pattern dictionary

The names of the slots in the table frames are the names of objects (stroke, radical or character objects). The slots themselves store the names of the corresponding

frames, that is, these slots are pointers. Each data frame represents an element of an object. It stores the relevant information such as the name of the corresponding object ("object_name"/"radical_name" slot) and its boundary positions (startx, starty, endx and endy slots). The elements of an object are linked together by the "next" slot field. In the case of the stroke database, the startx, starty, endx and endy slots store the actual starting and ending positions of the corresponding segment.

4.2.2. The Input Decoder Module

This module is responsible for the following functions:

(1) It performs the by-pass operation on input data when the character_mode is ENGLISH. All data are sent to this module for the data synchronization purpose.

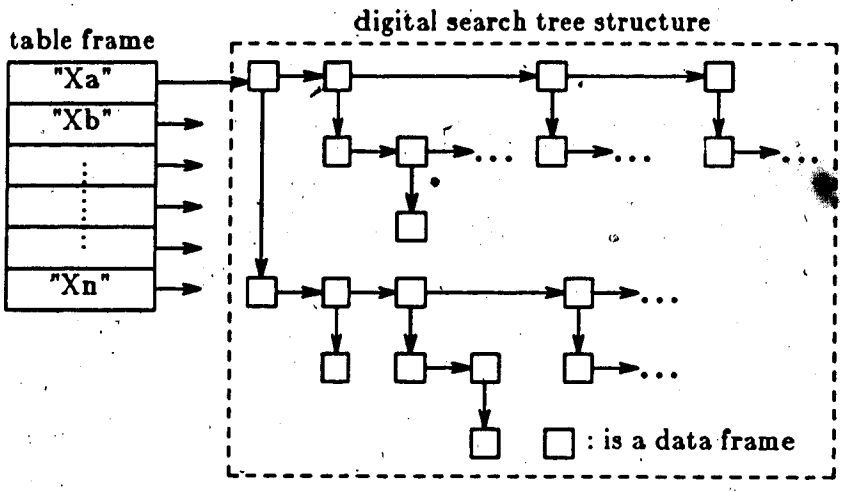
(2) It converts every six successive digits (TCCM code) into the corresponding Chinese-code when the character_mode is CHINESE and input_method is TCCM. All other characters are by-passed. A table look-up method is employed in the conversion process. This table is implemented using the FDB and is created by the pattern editor during the font editing process.

(3) It performs hand-written Chinese character recognition when the character_mode is CHINESE and input_method is RECOGNITION. The character recognition scheme described in the last chapter (Section 3.4) is implemented.

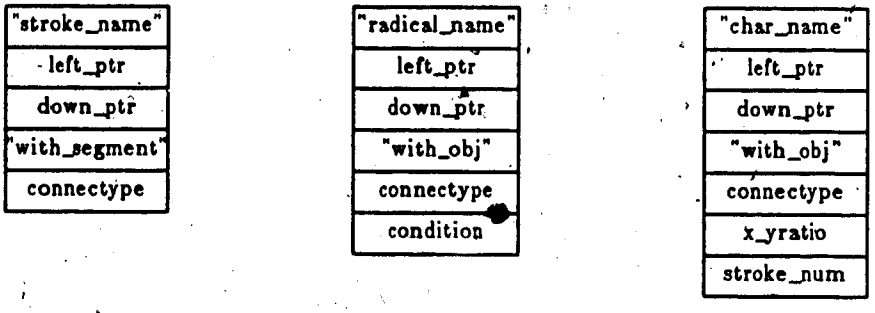
Standard I/O channels are used for input and output data. A special Chinese-code ($81C0C0C0_{16}$) is designated to be an error code. Whenever a failure occurs, either in the conversion process or in the recognition process, this special code is sent (as an error indicator) instead of the internal code of the input character.

4.2.2.1. Data Structure of the Recognition Dictionary

The recognition dictionary is also implemented using the FDB. The basic organization of the dictionary is divided into three separate databases, the stroke, radical and character databases. The data structures of these databases are all based on the structure of a digit search tree. Figure 4.3 shows the basic organization of these databases.



(a): Basic structure



(i): stroke (ii): radical (iii): character

(b): data frame structure

Figure 4.3 Organization of the recognition dictionary

Similar to the pattern dictionary, the names of the slots in the table frames are the names of objects (stroke, radical or character objects). The slots themselves store the names of the corresponding frames. These tables provide a direct access to all

terminate data frames. A data frame of each of the databases contains the relevant information shown in Figure 4.3(b). The connectype slot stores the connection type of the current pattern with the next object. The name of the next object is stored in the "with_segment"/"with_obj" slot. The character composition, besides matching the next object ("with_obj") and the connectype, also matches the ratio between the length in X and Y boundaries of the next radical stored in the x_yratio slot. When a data frame is a terminate frame, the "stroke_name", "radical_name" and "char_name" slots of the stroke, radical and character databases, store the name of the corresponding stroke, radical and character respectively. The condition slot in a data frame of the radical database stores the ambiguity information of the corresponding radical. It is used to indicate whether the ambiguity resolution is needed or not.

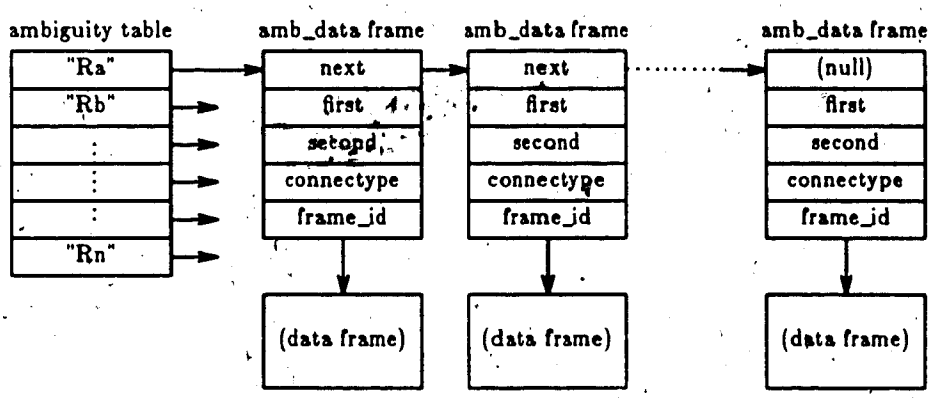


Figure 4.4 Data structure of the ambiguity resolution data

Figure 4.4 shows the data structure of the ambiguity resolution data, it is stored in the radical database. The ambiguity table frame contains all the entrances of the ambiguity radical lists. Each of the amb_data frames stores the resolution information. The connectype slot stores the relation of the pair of strokes that provides a positive identification of the radicals. The sequence numbers of these strokes are stored in the first and second slots. The frame_id slot points to the data frame that contains the corresponding radical information.

4.2.3. The Interface Module

The interface module is implemented based on the basic structure of WINDLIB. The screen is divided into windows, Figure 4.5 shows the layout of the screen. An event handler is associated with each of the windows. These handlers deal with user interactions.

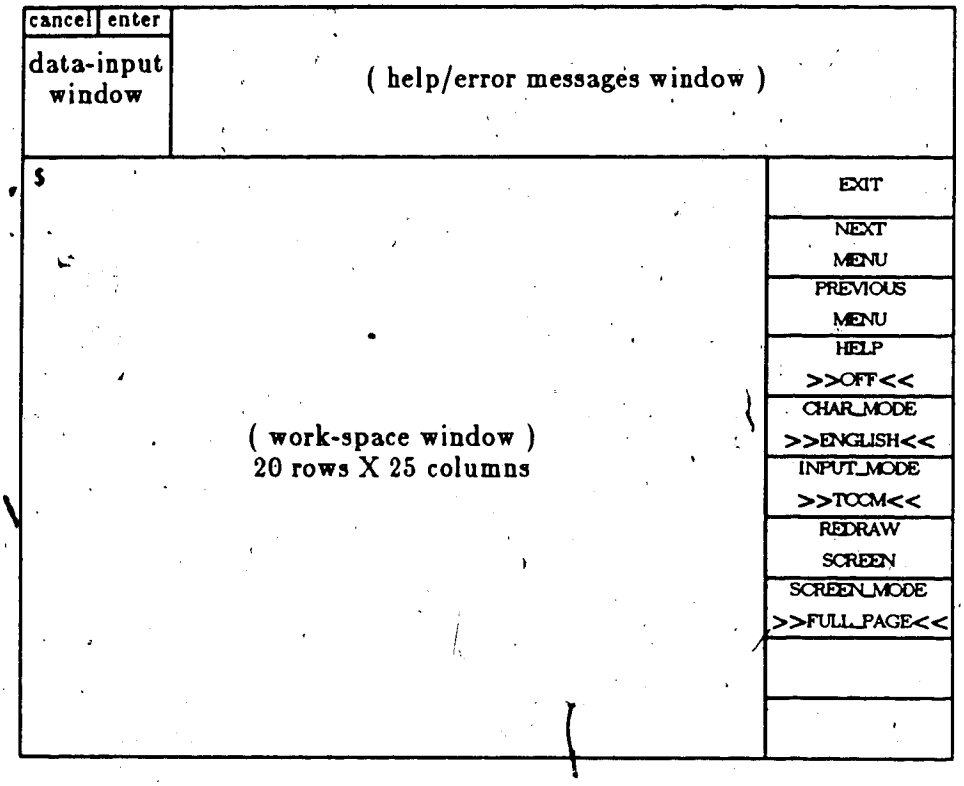


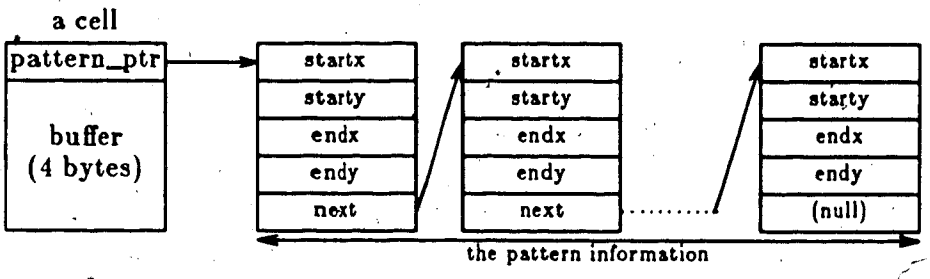
Figure 4.5 The screen layout of the LCT

All the menu items, including the menu items defined by an application program, are linked together and displayed in the menu window. Whenever a menu item is selected, an appropriate action (function) will be executed. The NEXT MENU and PREVIOUS MENU commands are provided for displaying the next and previous set of menu items respectively. These two menu items and the EXIT menu item (the command to quit the LCT program) are displayed permanently. Two special commands, the cancel and the enter commands, are associated with the data-input window. The

cancel command is for deleting the current editing (writing) of a Chinese character and the enter command is used to indicate the completion of writing a Chinese character.

The data-input window is for inputing data. Its handler performs the inputing (writing) of Chinese characters, TCCM codes and conventional characters. All characters typed within other windows are passed on to this handler as though they were typed within the data-input window. Inputing data, writing of a character or inputing a TCCM code, is echoed and displayed in the data-input window to provide an immediate feedback.

The work-space window is the logical screen. It is sub-divided into twenty (rows) by twenty-five (columns) cells. Each cell holds (displays) a Chinese character or two ASCII characters. These cells are implemented using a two dimensional array of records. The structure of a cell is shown in Figure 4.6. To decrease the re-display time, each of the Chinese character patterns is stored in a simple linked list. The pattern-ptr of a cell record points to the head of the corresponding pattern.



pattern_ptr: points to null when the cell holds ASCII character(s)

Figure 4.6 A cell structure of the logical screen

The help/error messages window is used to display the help and error messages. "Help" commands on all menu items are provided. A help message contains a brief description of the corresponding menu item. When the menu item is selected and the "HELP" is turned ON, the corresponding help message will be displayed in the help/error messages window.

Basically, the current version of the interface module is responsible for the following activities.

(1) Controls all commands/data flow between the various modules.

(2) Converts user interactions into input data of other modules.

(3) Executes commands sent from an application program. (A set of functions, for communication between an application program and the LCT, is written and is presented in Appendix A4.)

(4) Data display (screen) management.

(5) Provides various kinds of feedbacks and a help facility.

4.3. Implementation of the Pattern Editor

The pattern editor is also implemented based on the basic structure of WINDLIB. The editor provides an interactive environment allowing users to create the display patterns which include stroke, radical and character patterns. The second function of this editor is to generate the recognition schemes of those patterns. The editing environment is best illustrated by the screen layout of the editor itself.

An important consideration in the implementation of this editor is the use of the screen that is the layout of the various windows. Figure 4.7 shows the screen layout of the editor. An event handler is associated with each of the windows. These handlers deal with the user interactions such as selecting commands, and execute appropriate functions like retrieving a pattern from the pattern dictionary or updating (storing a new pattern) the pattern and the recognition dictionaries.

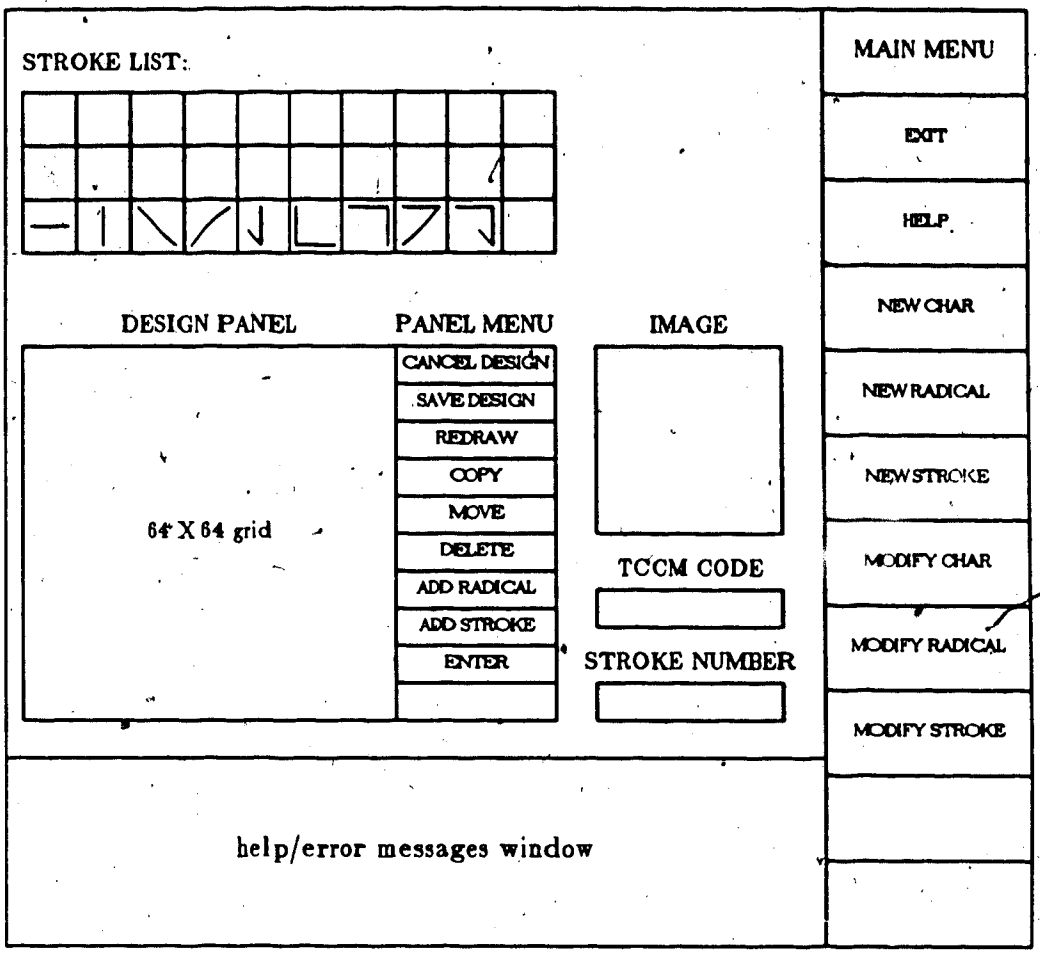


Figure 4.7 The screen layout of the editor

To release users from thinking too deeply about how to use this editor, all commands are displayed as menu items and the stationary menus are used as the menu type. The commands are grouped into "MAIN MENU" and "PANEL MENU" as shown in the Figure 4.7. All selected menu items are highlighted. The STROKE LIST window displays all the currently available stroke types. The DESIGN PANEL is a work area for designing the display (stroke, radical and character) patterns. A grid is provided in this window. The purpose is to ease the process of positioning segments. Below the DESIGN PANEL is a text window, the help/error messages window. This window is used to display all error and help messages. The TCCM CODE and STROKE

NUMBER windows are used for inputing the corresponding TCCM code and the stroke number when designing a character pattern. The IMAGE window provides a view of the current pattern that is in the design process.

Basically, the design process is simple. The user first selects a global command from the MAIN MENU and then starts the design process. The design process is a repetitive process, that is, continuously selecting and positioning an existing stroke or radical pattern as a sub-pattern of the new pattern. These sub-patterns are temporarily linked together using a simple link list data structure. The ADD RADICAL and ADD STROKE commands are used to add a radical pattern and a stroke pattern into the current (new) pattern respectively. The SAVE command activates the save process. The CANCEL DESIGN command is for canceling the design process. A REDRAW function is also provided; when selected (at any moment), the DESIGN PANEL and the IMAGE windows will be redisplayed.

A stroke pattern is selected from the STROKE LIST and a radical is selected by writing the desired radical onto the DESIGN PANEL window. The ENTER command activates the radical retrieval process. The process first involves retrieving the internal code of the input radical. This is done by the recognition process. The corresponding pattern is then retrieved from the pattern dictionary using its internal code. When the SAVE DESIGN command is selected, the current design pattern will be saved only after the negative result from the test for a duplicate pattern is obtained. The test is actually the recognition process. This display pattern is then used as the reference pattern for generating the corresponding recognition scheme. The pattern and recognition dictionaries are updated accordingly.

Three editing commands are provided to help ease the design process. They are the COPY, MOVE and DELETE commands. These commands can be used for all the composition sub-patterns of the current pattern. In general, the editor detects and

reports all possible errors, errors such as saving an incomplete pattern, designing duplicated patterns, and others. It also allows users to select whether or not help messages are needed. When the HELP command is turned off (default is on) all help messages will be omitted.

Chapter 5

Conclusion

5.1. Conclusions

The initial intent of this research was to study "on-line Chinese character recognition" as a general input method for novice users. However, because of lack of Chinese processing support facilities under UNIX (the target OS for the intended implementation), we had to consider other issues such as the internal coding and display method of Chinese characters. The approach that has been taken was to produce a system that allows for growth and for experiments on various system components. Hence, the invested efforts could be re-utilized. The goal was to establish a research tool for the study of Chinese information processing using UNIX systems.

Currently, the implementation has reached the point where application programs can be built. The implemented system generally met the requirements stated in Section 3.1 and the design criteria listed in Section 3.5.

Based on the fundamental requirements for the CCSS system discussed in Chapter 3 (3.1), the following issues have been investigated (designed and implemented).

(1) The internal coding of Chinese characters

An internal coding (Chinese-code) has been proposed that supports file access methods including random file access. Editing requirements such as searching, sorting and others are also supported. Both Chinese and ASCII codes can be freely mixed within a file.

(2) A display model

Based on the hierarchical model of Chinese characters, a display pattern dictionary is designed and implemented.

(3) Hand-written Chinese character recognition method

A recognition method based on a syntactic approach is developed. This method recognizes restricted hand-written Chinese characters. Stroke order and number variations in inputing Chinese characters are allowed.

Finally, two major components, the logical Chinese terminal (LCT) and the pattern editor, of CCSS system are designed and implemented. The LCT is responsible for I/O handling tasks. It supports the I/O of both Chinese and conventional characters. Two Chinese input methods are currently implemented; the Three Corner Coding Method (TCCM) and an on-line character recognition method aimed at expert and novice users respectively. A set of communication functions between LCT and application programs is written. The purpose is to ease the development of applications involving Chinese character processing.

The pattern editor provides an interactive environment for designing display patterns of Chinese characters. The editor also automatically generates the recognition schemes of the designed patterns. The display patterns and recognition schemes gathered and/or generated by the editor are stored in the pattern and recognition dictionaries respectively. To eliminate the need for memorizing the internal codes of radicals, the hand-written character recognition method is employed in the radical pattern retrieving process.

A Frame Based Database (FDB) is used in the implementation of the dictionaries. Therefore, the performance of the system in accessing the stored information depends solely on the efficiency of data access facilities used in FDB. Unfortunately, the suitability of using FDB as the underlying database management system (DBMS) was not considered, rather FDB was used to ease the implementation. Hence the performance of the system is not known at the present moment.

5.2. Further Work

The design and implementation of the current system has not been completed. There are a number of enhancements that must be added. In the rest of this section we will present some suggestions, possible directions for improvements and necessary further work.

As mentioned in the conclusion, the performance of our system depends heavily on the efficiency of data access facilities used in FDB. These facilities may or may not be optimized for our data structures. A possible research direction is to investigate/study appropriate storage structures for the dictionaries. Based on the results of the studies, we could then choose or implement an optimum DBMS for our system.

In the current system, two Chinese input methods are implemented. As mentioned in Chapter two, many input methods are proposed and no single method has yet received universal acceptance. Therefore, it would be more appropriate to extend the system to include other input methods. By doing so, the system not only becomes more serviceable but also provides a practical environment for evaluating various input methods.

Currently, we are ready to build application programs. The initial programs that needed to be designed and implemented are (i) Chinese text editing and (ii) Chinese text formatting facilities. These facilities present some interesting design/implementation difficulties owing to different needs in Chinese documents. For example, the current writing styles of Chinese text include both the traditional and the English styles of writing. Traditional Chinese text is written/presented from top to bottom and then from right to left! Preparing a document in both styles may cause formatting problems that are not found in normal English document preparation. "Chinese document preparation" certainly will be an interesting research topic. Furth-

ermore, an extension of adding a Chinese programming language such as "Chinese BASIC" or "Chinese COBOL [Chu79b]" would enhance the system and in itself be challenging.

There are many applications that indeed are needed. The author hopes that the work reported in this thesis will stimulate researchers to put further efforts into the development of Chinese information processing using UNIX systems as development tools, especially at University of Alberta.

References

[Ara83] H. Arakawa, On-Line Recognition Of Handwritten Characters-Alphanumerics, Hiragana, Katakana, Kanji, *Pattern Recognition Vol. 16*, (1983), 9-16.

[Autom] Automated, ALS208 Chinese Computer Typewriter, Automated Systems(HK) Ltd., Brochure, (date unknown).

[Chan] S. W. Chan, *Tsang-chi Chinese Input Method*, Computer Technology Publisher, Macau, (date unknown). In Chinese.

[ChH79] C. F. Chong and S. W. Ho, An On-Line System For Handwritten Chinese Character Input, *Proceedings of the 1979 International Conference Of Chinese Language Computer Society*, 1979, 87-100.

[Chu79a] Yaohan Chu, Directions for the Chinese Language Computer Society, *Proceedings of the 1979 International Conference Of Chinese Language Computer Society*, 1979, 115-119.

[Chu79b] Yaohan Chu, Chinese Micro-COBOL: A Data Processing Language For Microprocessor Systems, *Proceedings of the 1979 International Conference Of Chinese Language Computer Society*, 1979, 189-212.

[FoD83] J. D. Foley and A. Van Dam, *Fundamentals of Interactive Computer Graphics*, Addison Wesley, Reading, MA, 1983.

[FuL79] K. S. Fu and S. Y. Lu, Applicability of Pattern Handling Methods To Chinese Language Processing, *Proceedings of the 1979 International Conference Of Chinese Language Computer Society*, 1979, 131-172.

[Gre82] Mark Green, FDB: A Frame Based Database System, University Of Alberta, User Manual, 1982.

[GrB84] Mark Green and Nancy Bridgeman, WINDLIB Programmer's Manual, University Of Alberta, User Manual, 1984.

[HaY80] S. Hanaki and T. Yamazaki, On-Line Recognition Of Handprinted Kanji Characters, *Pattern Recognition Vol. 12*, (1980), 421-429.

[HCD82] Ergong He, Francis Chin, Wayne A. Davis and Changsong Sun, The Design of an Input Method and Software System for Chinese Language Minicomputer, *Proceedings of the 1982 International Conference Of Chinese Language Computer Society*, 1982, 252-263.

[HCH79] L. R. Hu, Y. W. Chang, K. T. Huang and C. A. Hoffman, The Three Corner Coding Method, *Proceedings of the 1979 International Conference Of Chinese Language Computer Society*, 1979, 44-61.

[HuS83] Eing M. Huang and Ching Y. Suen, Computational Analysis Of The Structural Compositions Of A Frequently Used Chinese Character Set, *Proceedings of the 1983 International Conference On Text Processing With A Large Character Set*, 1983, 292-297.

[Hua85] kai-tung Huang, The Input and Output of Chinese and Japanese Characters, *Computer Vol. 18 Number 1*, (January 1985), 18-26.

[IBM] IBM, IBM Multistation 5550, Distributor: Dodwell Computer Division(H.K.), Brochure, (date unknown).

[IPX] IPX, IPX 5486 Automatic Send/Receive(ASR) Telecommunications Terminal, Ideógraphix Inc., Sunnyvale, CA 94086, Brochure, (date unknown).

[IYM81] Kakashi Ikeda, Takashi Yamamura, Yasumasa Mitamura, Shiokazu Fujiwara, Yoshiharu Tominaga and Takeshi Kiyono, On-Line Recognition Of Handwritten Characters Utilizing Positional And Stroke Vector Sequences, *Pattern Recognition Vol. 13*, (1981), 191-206.

[KeP84] B. W. Kernighan and R. Pike, *The UNIX Programming Environment*, Prentice Hall, Englewood Cliffs, NJ, 1984.

[Kia82] T. Y. Kiang, The Construction Of Chinese Characters In Terms Of Certain Basic Strokes And Its Transformation To Different Fonts, *Proceedings of the 1982 International Conference Of Chinese Language Computer Society*, 1982, 12-17.

[LeC80] C. C. Lee and C. S. Chou, Structure Analysis And Coding Of Chinese Characters, *Proceedings of International Computer Conference: Hong Kong 1980*, 1980, Session 2-1.

[MaS80] J. Mathias and K. J. Schmucker, Ideograph Writing Patterns And Computer Input, *Proceedings of International Computer Conference: Hong Kong 1980*, 1980, Session 2-4.

[NNN83] H. Nagahashi, M. Nakatsuyama and N. Nishizuka, Pattern Generation Of Chinese Characters In Terms Of Graphics, *Proceedings of the 1983 International Conference On Text Processing With A Large Character Set*, 1983, 281-286.

[NMA83] Masaki Nakagawa, Toshihiko Manabe, Katsuo Aoki, Yuji Ikeda and Nobumasa Takahashi, On-Line Handwritten Character Recognition As A Japanese Method, *Proceedings of the 1983 International Conference On Text Processing With A Large Character Set*, 1983, 191-196.

[Sue79] Ching Y. Suen, *Computational Analysis of Mandarin*, Basel, Boston, Stuttgart: Birkauser, 1979. Interdisciplinary systems research 72.

[Sue80] K. S. Suen , *Fundamental Knowledge of Chinese Characters*, Publisher of He Bei People, China , 1980. In Chinese.

[Sue83] Ching Y. Suen, Computer Recognition Of Kanji Characters, *Proceedings of the 1983 International Conference On Text Processing With A Large Character Set*, 1983, 429-435.

[TNM83] T. Takahashi, S. Naito and I. Masuda, Handprinted Chinese Character Discrimination Using Stroke Extraction Method With Referral To Peripheral Structural Information, *Proceedings of the 1983 International Conference On Text Processing With A Large Character Set*, 1983, 163-168.

[Tie82] H. C. Tien, A Pinyin-Based Computer-Chinese Language System, *Proceedings of the 1982 International Conference Of Chinese Language Computer Society*, 1982, 178-190.

[Tie85] H. C. Tien, The Pinxxiee Chinese Word Processor, *Computer Vol. 18 Number 1*, (January 1985), 65-66.

[WaC83] T. Wakahara and K. Christian, *The Unix Operating System*, John Wiley & Sons, New York, NY, 1983.

[WaU83] T. Wakahara and M. Umeda, Stroke-Number And Stroke-Order Free On-Line Character Recognition By Selective Stroke Linkage Method, *Proceedings of the 1983 International Conference On Text Processing With A Large Character Set*, 1983, 157-162.

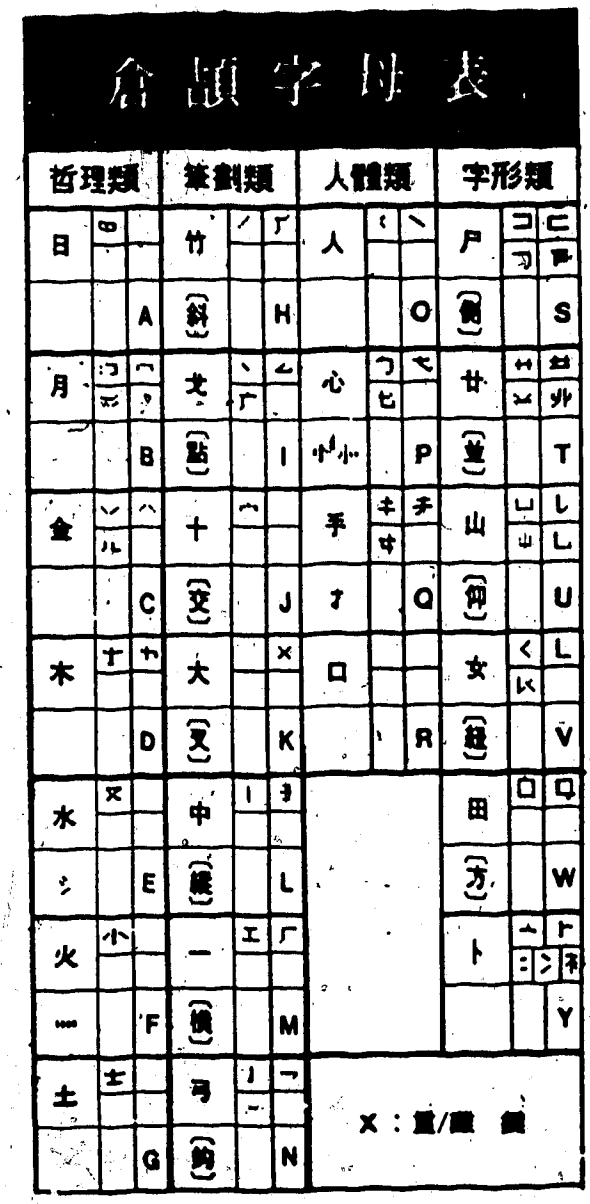
[YPA83] H. K. Yim, C. J. Poh and S. P. Ang, An experimental Chinese and English Word Processing System, Concordia University, Project Report, 1983.

[Yip85] L. C. Yip, A Discussion On Simplified Chinese characters, *People's Daily (Overseas Edition)*, 26 Oct., 1985.

Appendix A1

Tsang-chi alphabet

This table is extracted from the book titled "Tsang-chi Chinese Input Method" by Chan [Chan].

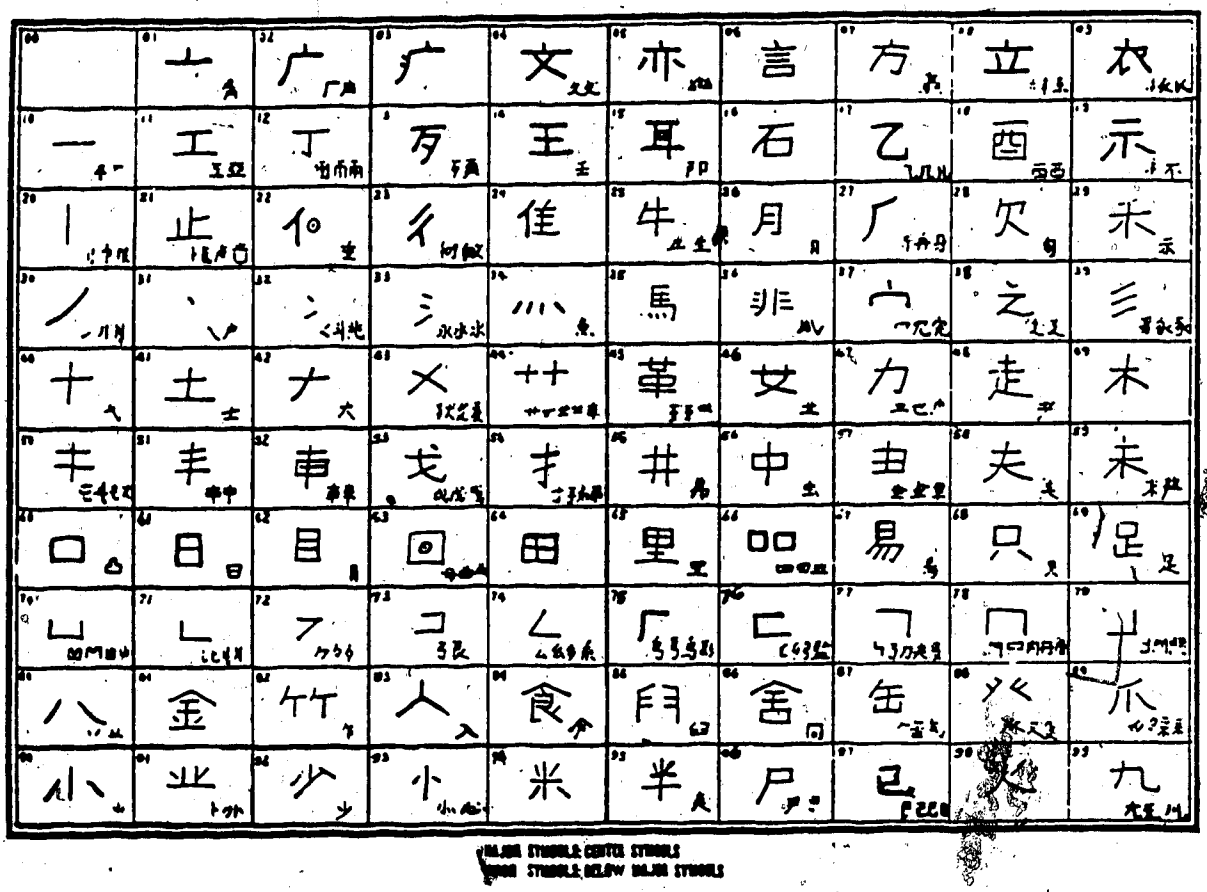


Appendix A2

The TCCM Fundamental Symbol Chart

This chart is extracted from the paper titled "The Three Corner Coding Method" by Hu et al [HCH79].

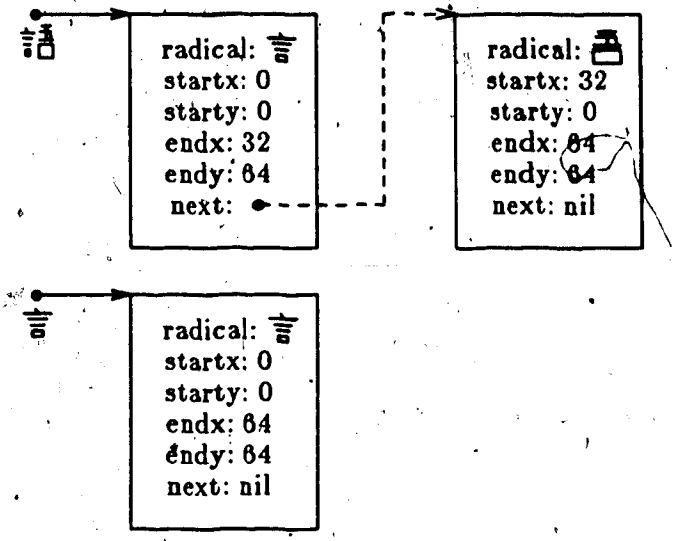
FUNDAMENTAL SYMBOL CHART



Appendix A3

Information Stored in Pattern Dictionary (An Example)

The following figures show an example of the information stored in the pattern dictionary for the Chinese word *language*, displayed in the Figure 1.2. FDB is used to store the information.



Each BOX is a **frame** in FDB
The fields inside the BOXs are **slots**

Figure A3.1 Information Stored in Character database

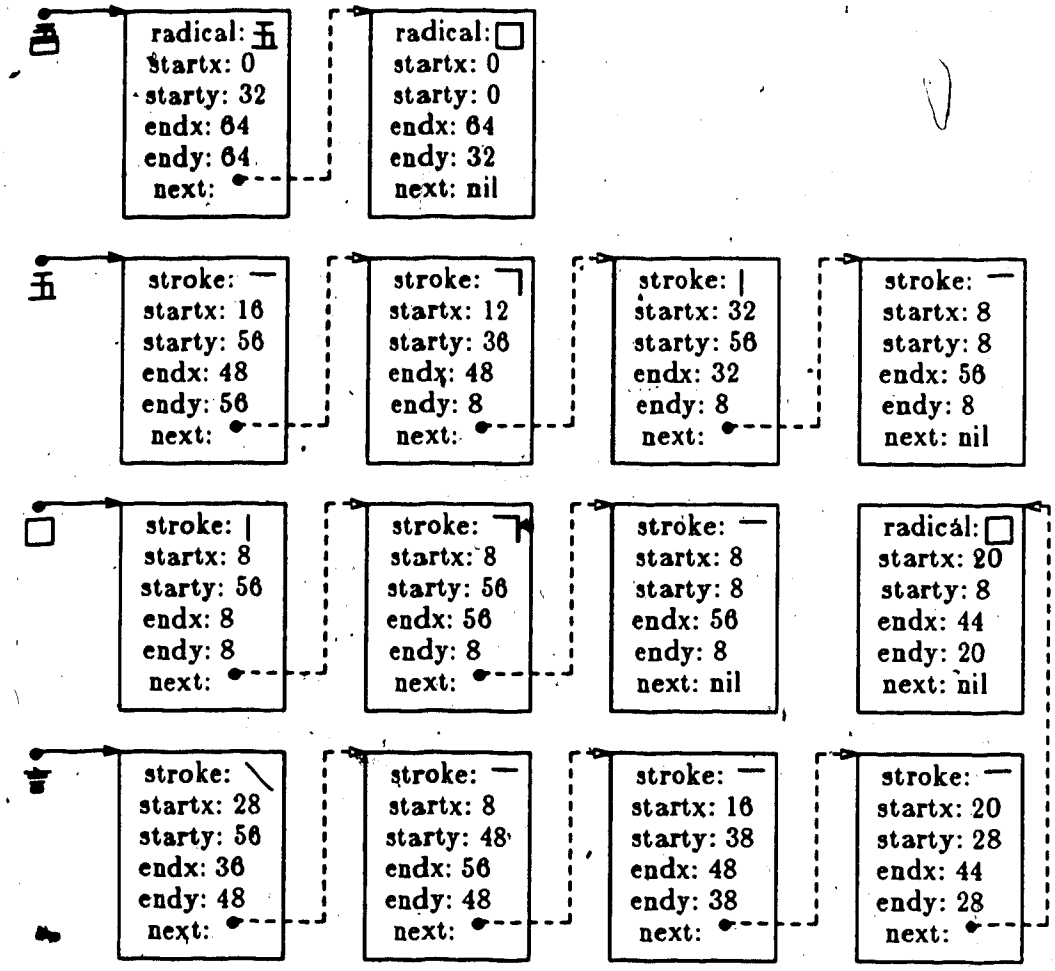


Figure A3.2 Information Stored in Radical Database

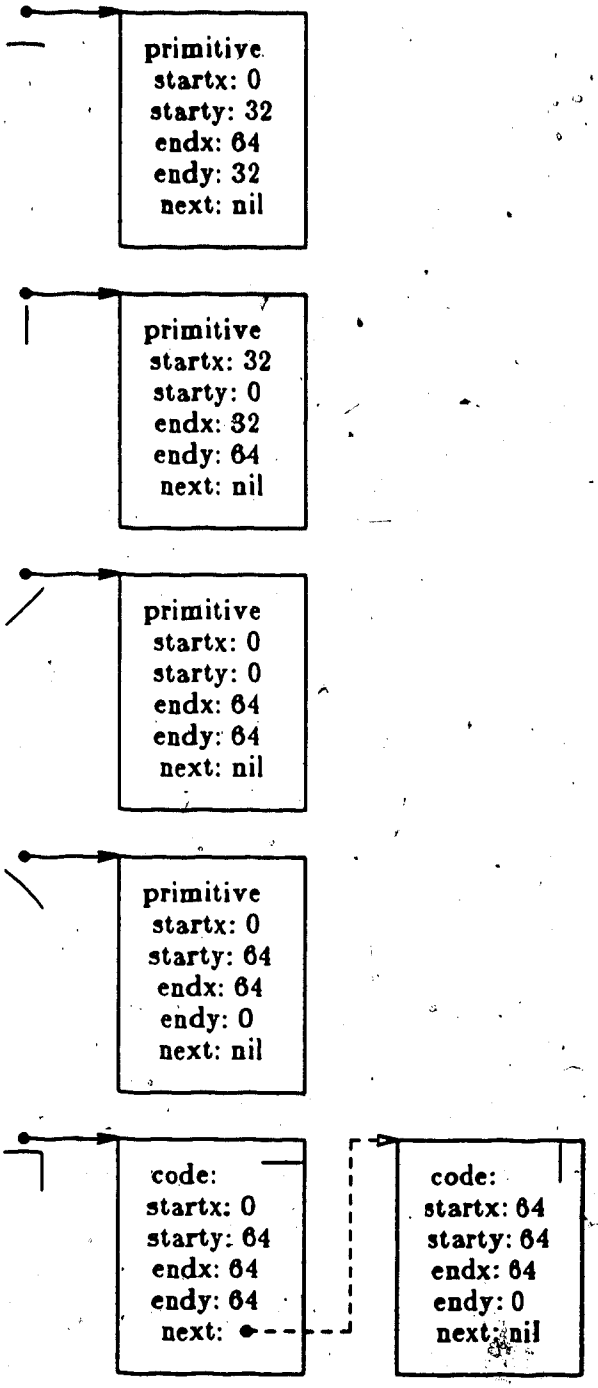


Figure A3.3 Information Stored in Stroke Database

Appendix A4

Communication between Application Programs and the LCT

A set of functions is written in the C programming language. The purpose is to ease the implementation of application programs, such as Chinese text visual editing and text formatting application programs, and others. These functions are presented in this section.

(1) int Define_Menu(menu_name, help_mess)
char *menu_name, *help_mess;
/* menu_name: a brief meaningful name of the corresponding menu item */
/* help_mess: the on-line help message (optional) of the menu item */

This function is used to define a menu item. A unique "menu_id", associated with the menu item, is returned. When a menu item is selected by the user, the interface module will generate a command together with the associated menu_id and send it to the application program.

(2) Remove_Menu(menu_id)
int menu_id;
/* menu_id: the identification number of the intended menu item. */

This function is used to remove/delete the associated menu item from the menu.

(3) Send_Mess(help_mess)
char *help_mess;
/* help_mess: help or error message. */

This function is used to display the corresponding message as a help/error message.

(4) Reset_Terminal()

This function is used to reset the LCT. All defined menu items will be removed and the terminal is reset to the default state.

(5) Clear_Terminal()

This function is used to clear the logical screen of the LCT.

(6) Echo_On()

This function is used to turn on the local echo mode.

(7) Echo_Off()

This function is used to turn off the local echo mode.

(8) Cursor_Allow()

This function is used to turn on the selectable cursor mode. When the mode is on, the interface module will generate a command, together with the corresponding cursor position as the argument to the application program, when the user uses the tracking cross to select a position (within the logical screen).

(9) Cursor_Disallow()

This function is used to turn off the selectable cursor mode.

(10) Screen_Disallow()

This function is used to disable the user from selecting the screen mode.

(11) Screen_Allow()

This function is used to enable the user to select the screen mode.

(12) Move_To(row, column)
int row, column;
/* row: the row value of the corresponding intended position. */
/* column: the column value of the corresponding intended position. */

This function is used to position the cursor to the intended position.

(13) Set_Menu_Color(menu_id, color)
int menu_id, color;
/* menu_id: the identification number of the intended menu item. */
/* color: the intended color for the corresponding menu item. */

This function is used to set the color of the menu (to highlight the selected menu item).

(14) EVENT Get_event()

This function is used for reading data. The data can be either ASCII characters, or Chinese characters or commands. The type of the data is determined by the "evtype" field. The data/arguments will be placed in the corresponding fields. The structure of EVENT is defined (in C programming language) as follow.

```
typedef struct event {
        int evtype;
        int posx;
        int posy;
        int menu_id;
        char ascii_char;
        char Chinese_char[4];
} *EVENT;
```