

AD-770 241

SYNTAX DIRECTED ON-LINE RECOGNITION
OF CURSIVE WRITING

Yung Taek Kim, et al

Utah University

Prepared for:

Advanced Research Projects Agency

July 1968

DISTRIBUTED BY:

NTIS

National Technical Information Service
U. S. DEPARTMENT OF COMMERCE
5285 Port Royal Road, Springfield Va. 22151

DISCLAIMER NOTICE

THIS DOCUMENT IS THE BEST
QUALITY AVAILABLE.

COPY FURNISHED CONTAINED
A SIGNIFICANT NUMBER OF
PAGES WHICH DO NOT
REPRODUCE LEGIBLY.

Technical Report 4-8

Yung Taek Kim
David C. Evans

AD 770241

SYNTAX DIRECTED ON-LINE RECOGNITION OF CURSIVE WRITING

July 1968

COMPUTER SCIENCE *Div.*
Information Processing Systems
University of Utah
Salt Lake City, Utah

DDC
RECEIVED
NOV 1 1973
B

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
U S Department of Commerce
Springfield VA 22151

Advanced Research Projects Agency • Department of Defense • ARPA order 829

Program code number 6D30

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

TABLE OF CONTENTS

CHAPTER	PAGE
LIST OF TABLES	iii iii
LIST OF FIGURES	iv iv
ABSTRACT	vi vi
I. INTRODUCTION	1
1.1 Motivations	1
1.2 Scope and Depth	2
II. DATA FILTERING PROCESS	6
2.1 System Requirements	6
2.2 Noise Filtering and Curve Smoothing	7
III. FIGURE EXTRACTION	12
3.1 Writing Cutting and Stroke Breaking Algorithm	12
3.2 Stroke Naming and Relative Position Coding Rule	14
3.3 Character Decompositions	17
IV. SYNTAX FOR CURSIVE WRITING RECOGNITION	37
4.1 A Hierarchical Organization by the Stroke Characteristics	37
4.2 Syntax Specifications	39
4.3 Semantic Interpretations	40
V. IMPLEMENTATION OF THE SYNTAX ANALYZER	48
5.1 System Analyzer	48
5.2 Character Compositions	55
5.3 Local Discriminating Routines	70

CHAPTER	PAGE
VI. SELF-CORRECTING ALGORITHMS	78
6.1 Self-Corrections by Iteration	78
6.2 Stability of the Self-Correcting Algorithms	81
VII. CONCLUSION	84
7.1 Accuracy of Recognition	84
7.2 Further Work	85
VIII. BIBLIOGRAPHY	86
IX. APPENDIX	87
9.1 Listing of Program	87

LIST OF TABLES

TABLE	PAGE
I. Syntax specifications for handwriting recognition with self-correcting loops	41
II. Stroke combinations for circled characters	57
III. Stroke combinations for first-fixed characters	58
IV. Stroke combinations for relative characters	61

LIST OF FIGURES

FIGURE	PAGE
1. Display of handwritten word and its recognition	5
2. X-value correction for low initial value	10
3. X-value correction for high initial value	10
4. Y-value correction for low initial value	11
5. Y-value correction for high initial value	11
6. Cutting algorithm for writing "na"	13
7. Breaking algorithm for writing "a"	13
8. Codings and boundaries of major diagonal directions	16
9. Example of stroke naming algorithm	16
10. Example of normal sized stroke	18
11. Comparison between normal sized stroke and above positioned stroke	18
12. Comparison between normal sized stroke and down positioned stroke	19
13. Comparison between normal sized stroke and full sized stroke	19
14. Character structures by the stroke feature	21
15. Character structures by the stroke name	31
16. Stairing characteristics for character 'z'	49
17. Stairing characteristics for character 's'	49
18. System analyzer diagram	51
19. Characteristics for the circled character	52

FIGURE	PAGE
20. Characteristics for o-correction	52
21. Characteristics for b-correction	54
22. Characteristics for osculated character	54
23. Tree organization for the relative stroke 30-1	62
24. Tree organization for the relative stroke 302-1	64
25. Tree organization for the relative stroke 32-1	66
26. Tree organization for the relative stroke 3202-1	68
27. Characteristic evaluation for CCHEK	72
28. Characteristic evaluation for CVSE	72
29. Characteristic evaluation for GVSQ	73
30. Characteristic evaluation for HKVSL	73
31. Characteristic evaluation for HVSK	75
32. Characteristic evaluation for PCHEK	75
33. Characteristic evaluation for RCHEK	76
34. Characteristic evaluation for SCHEK	76
35. Characteristic evaluation for VCHEK	77
36. Block diagram for error correcting system	82

ABSTRACT

A syntax organization for recognition of handwritten connected-word is studied in this work.

Each writing is cut into strokes at the middle point of every down cave of the writing, and the strokes are named using their directional characteristics and relative size among the strokes.

A syntax is organized using the hierarchy of the stroke characteristics and self-iteration for the error corrections.

The strokes are classified by the hierarchy and processed to combine the strokes into characters by the hierarchical characteristics.

The lowest level of hierarchy collects those strokes which can not be combined into characters by their solid stroke characteristics and organizes a two dimensional family relation for relative combination of the strokes into characters.

The local classifying routines are called for those stroke relations which require the evaluation of the relative characteristics between the strokes for the optimal decision.

CHAPTER I

INTRODUCTION

1.1 Motivations

Two dimensional input devices of the computing machines attract many new studies in the man-machine communication field.

Although a two dimensional computer input is very attractive, a limited pattern recognition capability surely restricts its practical value.

For this reason, those works directed toward high dimensional interactive computer studies in software as well as in hardware have been discouraged.

Various experiments have been performed in recognition of hand-written, single characters and symbols on two-dimensional computer input devices during the past decade (2,3). Experiments were also conducted in the recognition of connected words of handwriting, using dictionary-driven word matching programs for limited combinations of characters (2,5).

In contrast, the object of the experiment reported here is to recognize arbitrary, handwritten connected words as well as single characters. In order to achieve this result, a different approach was required.

For the recognition of cursive writings without dictionaries or large tables, the writing was cut at the middle point of each down cave of the writing, and a syntax was organized for these strokes using a hierarchical system which is determined by the characteristics of all characters.

A semantics is implemented for each level of hierarchy to evaluate the characteristics for more efficient discrimination at each level of the hierarchy.

This organization produced a strong flexibility for many efficient alternatives and large room for decisive factors requiring only simple logic evaluation. In one use of this flexibility, self-checking and self-correcting algorithms were implemented for the specified types of errors. The criteria for stability of such iterative processes were determined experimentally.

In operation, the hand-writing from a Sylvania Tablet was displayed on the screen of the graphic system as the characters were written. After recognition, printed symbols were also displayed on the screen. The time between completion of the writing of the word of six characters and display of the printed word was less than two seconds.

The program was written in Algol 60 of UNIVAC 1108 and had been run in real time and on-line under the UNIVAC 1108 EXEC II system of the University of Utah (10).

1.2 Scope and Depth

Because the range of the handwriting pattern is broad, and the manner of various writers is different, many syntax entries must be examined, and many difficult organizational problems must be solved.

The accuracy of recognition and efficiency of the program is a function of such factors as the number of syntax entries and other complexities by improper writing.

This experiment is concerned with recognition of cursive writing of the twenty-six lower case letters based on Palmer's method of penmanship (7). These letters are coded along with other popular characters as written by normal writers for broader applicability.

Well-written Palmer's penmanship requires only a short program for recognition without a mistake. For such writing, the cutting algorithm would work completely. The coding of the direction of the strokes and the relative height of the writing would not cause any difficulty for such good writings. The syntax would be organized in a simple pattern and the semantics would be implemented with a relatively small number of entries, which reduces the number of discriminating algorithms automatically.

For a relatively uncaredful writing, errors are made in cutting the writing into strokes. Coding errors would also occur because of the difficulty in recognizing the direction of the stroke and the relative positions of the strokes.

The most difficult part in organizing a program for such imperfect writing is the syntax organization and semantic implementation. For such writing, the syntax must have some types of and some number of correcting algorithms, which require changes in hierarchy of the system and cause each iteration to contain complicated feedback loops.

In this experiment, relatively broad correcting algorithms were implemented and quite diverse writings are recognized by this algorithm.

Figure 1 shows the display of the handwritten word and its recognized symbols on the display screen of the graphics system at the University of Utah.



Figure 1. Display of handwritten word and its recognition

CHAPTER II

DATA FILTERING PROCESS

2.1 System Requirements

The graphics system at the University of Utah for cursive writing recognition consists of a UNIVAC 1108 as the central computer and a PDP-8 as the terminal machine with an Information Display Inc. system to generate the picture and a Sylvania Tablet as the two-dimensional data input device.

The UNIVAC 1108 has 36 bit words and 3 control bits for each input channel and output channel. The PDP-8 has 12 bit words for each input and output channel, and skip bit and interrupt bit for its input-output system. The two machines are coupled together through a linkage logic which modifies the word size of each machine and controls the order of transmission of the words.

The analog outputs of the Sylvania Tablet are connected to analog to digital converter to sample them into digital outputs and are transmitted to the PDP-8 accumulator. The display system and typewriter are also connected into the PDP-8 accumulator for communication during the processing. Since many machines are connected together, many alternatives in the system programming as well as some limitations for efficiency have to be considered for this graphic system.

The speed of machine and efficiency of the system program must be considered with higher priority in designing the system especially for the interactive design.

The resolution of the tablet was another important factor in designing experiments because it determines the minimum size of the writing which can tolerate the distortion of the sampling process.

The resolution of 1% per inch is required for normal writing, and higher resolution reduces the filtering part of the program such as noise filtering, density control, and curve smoothing processes.

A swapping mode is available from the graphic system at the University of Utah, written for the UNIVAC 1108 EXEC II system. Under this mode, the user takes only the running time from the central computer, which provides a good flexibility during the experiment.

The program was compiled and coded into relocatable language at a secondary memory of the central computer awaiting the next swapping interruption. While waiting for the 1108 to run the program, the terminal machine, the PDP-8, is running to pick up data from the tablet and display the writing on the CRT. As soon as the central computer is interrupted to swap with new data which is provided by the terminal machine, the swapping mode is executed to run the program with the new data.

2.2 Noise Filtering and Curve Smoothing

The locations of the stylus on the tablet are sampled by the analog to digital converter and transferred to the PDP-8 accumulator. The density of the sampled data can be controlled by the system program and by the filtering program simply checking the distance between the two consecutive samples.

The rate of the sampling process and the speed of system program for the collection of the sampled data are very important factors for filtering purposes. The speed of writing is always slow enough for the converter to sample the writing at any point. The system program has to be fast enough to look at every sampled point. These factors are the main limitations to the density of handwriting sampling.

Finite resolution of the tablet, as well as noises due to hardware failure require the checking and correcting algorithms which vary depending on the purpose of the experiment. A four-point noise checking and correcting rule was implemented for the cursive writing recognition experiment.

This algorithm operates as follows:

In Figure 2

```

if X(1) less than X(2) and X(2) greater than X(3) and X(3)
less than X(4)
    then CX(2) = X(1);

```

In Figure 3

```

if X(1) greater than X(2) and X(2) less than X(3) and X(3)
greater than X(4)
    then CX(2) = X(1);

```

In Figure 4

```

if Y(1) less than Y(2) and Y(2) greater than Y(3) and Y(3)
less than Y(4)
    then CY(2) = Y(1);

```

In Figure 5

if $Y(1)$ greater than $Y(2)$ and $Y(2)$ less than $Y(3)$ and $Y(3)$
greater than $Y(4)$

then $CY(2) = Y(1)$;

Other algorithms can be compared to the four-point algorithm regarding the reliability of the system. For instance, the three-point correcting algorithm is quite strong for checking and correcting purposes, but there would be a high probability of destroying information by changing the values at the sharp edges or sharp curves.

A curve smoothing rule could be implemented using some interpolating algorithms. This idea is generally the same as the three point rule except using the median value instead of current value. Especially for the iterating cases, there would be a big chance of destroying the original information producing wrong codings.

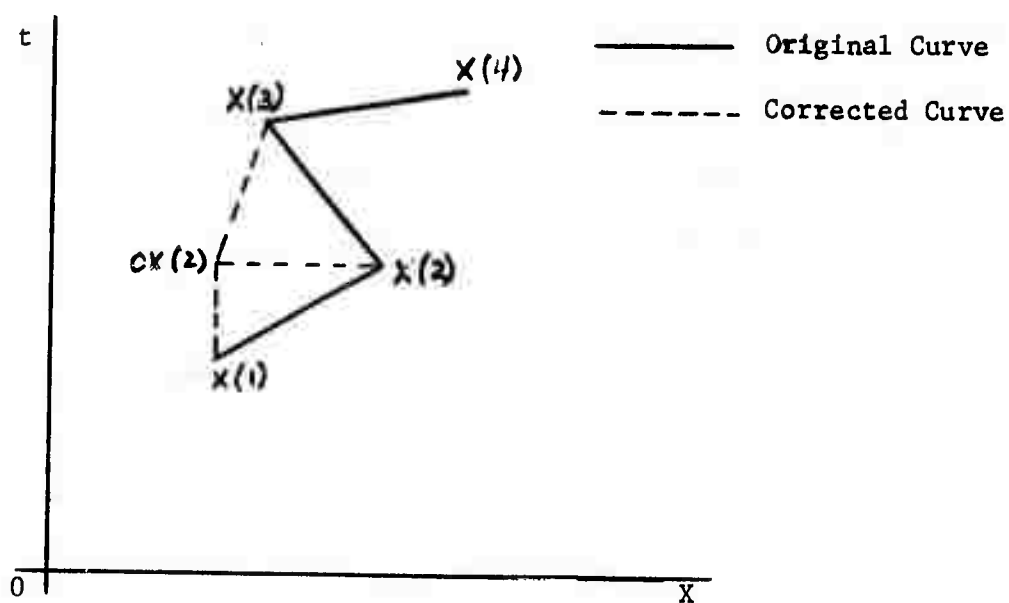


Figure 2. X-value correction for low initial value

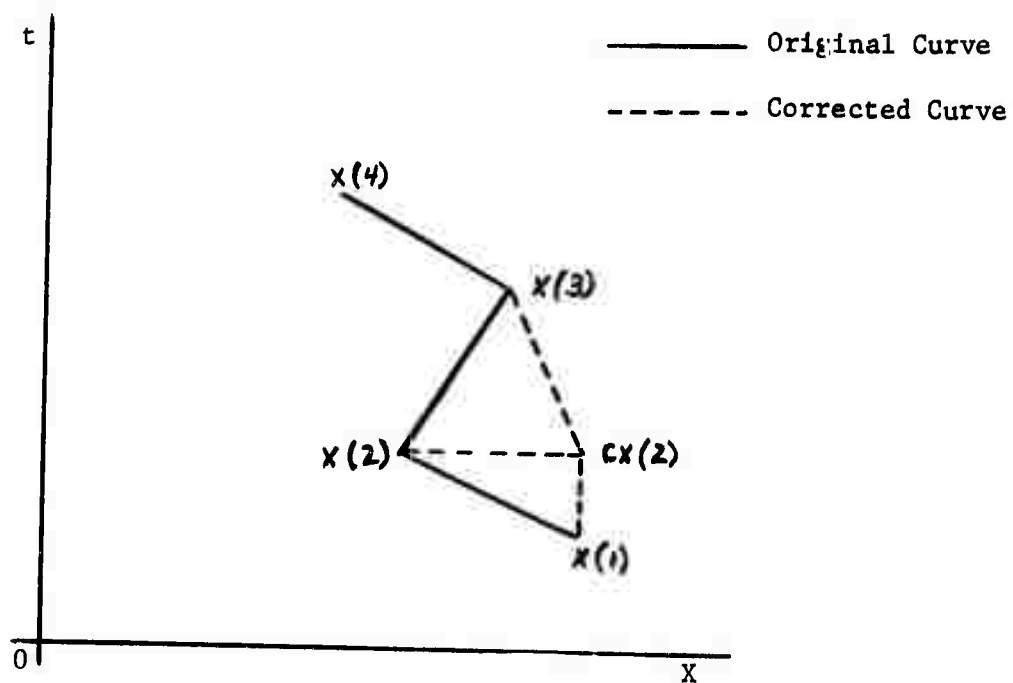


Figure 3. X-value correction for high initial value

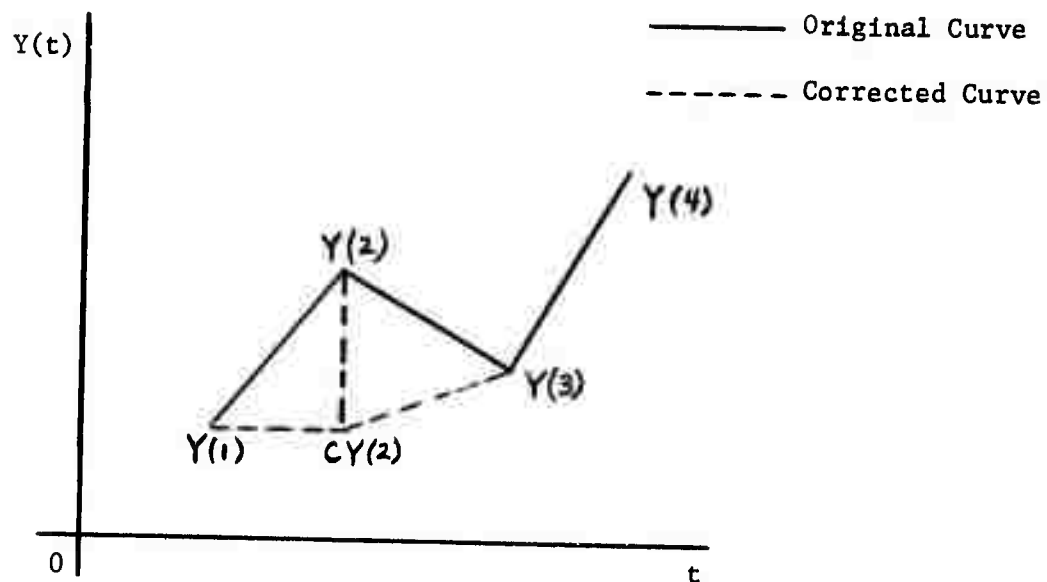


Figure 4. Y-correction for low initial value

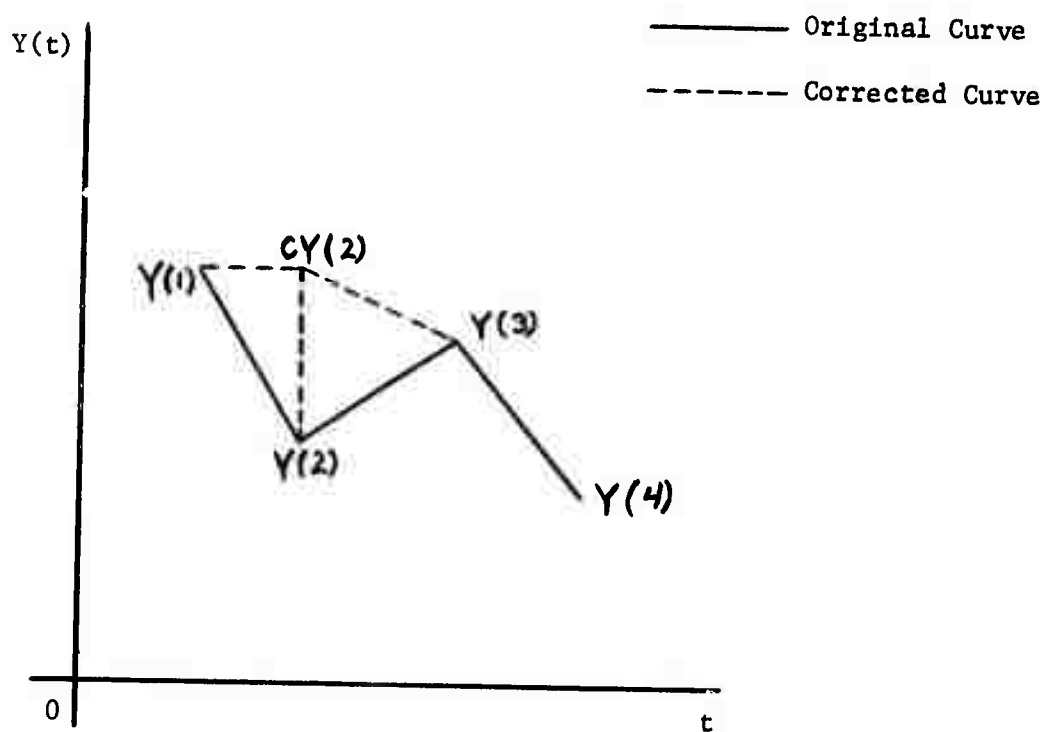


Figure 5. Y-correction for high initial value

CHAPTER III

FIGURE EXTRACTION

3.1 Writing Cutting and Stroke Breaking Algorithm

The normal writing of a word consists of several segments of piecewise continuous writing. In this experiment each discontinuity is marked at the discontinuous point by the third dimensional variable in the data structure. Each segment of writing is then cut into strokes at only the lowest point of every down cave of the writing, and each piece of stroke is indexed for further processing.

This low point cutting algorithm carries more information for later evaluation than any other algorithm because all high point and middle point information is completely available as in the original data.

The low parts of writing are always less important for the recognition than any other part of the writing, because nearly every written character ends by tailing down the last part of the last stroke of the character.

Each stroke is broken into branches at the points where the first difference of the X or Y variable changes the sign. Each branch is indexed and a two dimensional marker is used to point to the boundaries between branches.

This marking algorithm carries all data of the original system without losing or destroying any part of the feature, and this method

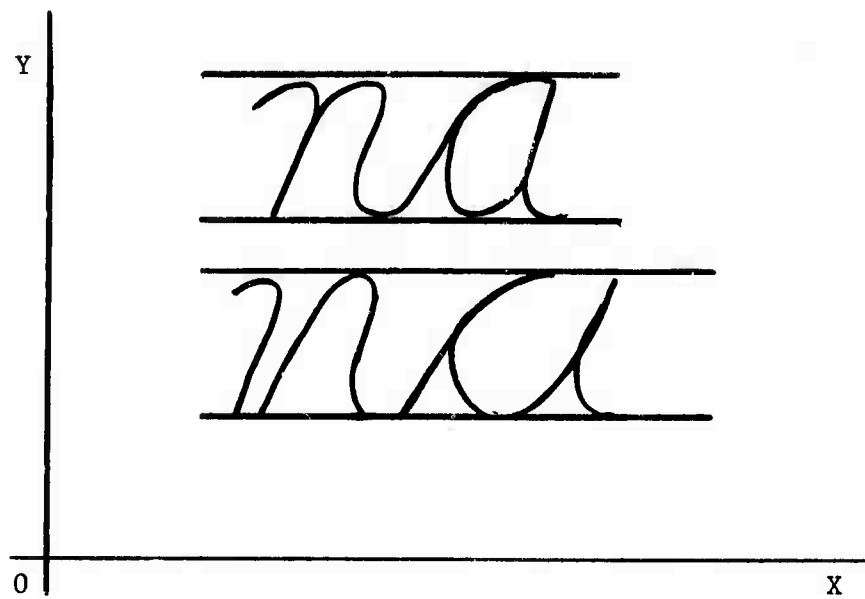


Figure 6. Cutting algorithm for writing "na"

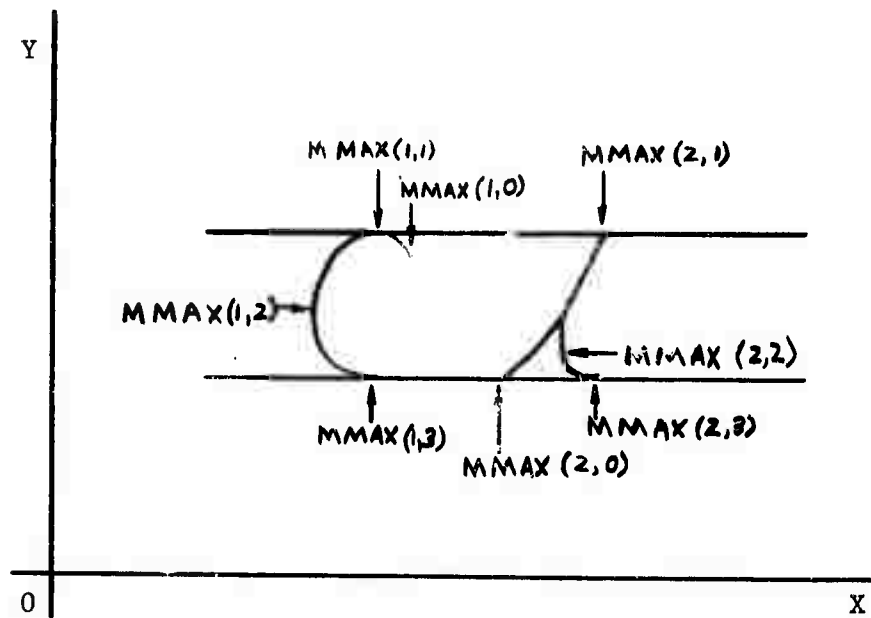


Figure 7. Breaking algorithm for writing "a"

enables the programmer to operate a large amount of data even with small capacity of memory.

The cutting and breaking algorithms are shown in Figure 6 and in Figure 7.

3.2 Stroke Naming and the Relative Position Coding Rule

Each stroke was broken into several branches to code the direction of the branches for naming of the stroke.

As soon as the stroke changes its major direction, a new directional code is assigned, which is added to the previous code multiplied by ten; this is repeated until the coding of the entire stroke is completed.

Four diagonal major directions are used in this work, and they are defined below:

- if X and Y both decrease then Code = 0;
- if X decreases and Y increases then Code = 1;
- if X increases and Y decreases then Code = 2;
- if X and Y both increase then Code = 3;

The horizontal stroke and short stroke were specially coded using 7 in this work.

In this procedure the stroke coding has less than six or seven digits because normal hand writing stroke does not have more than five or six directional changes.

The diagonal direction rather than rectangular direction was employed in this work because it has many advantages compared to other

directional coding algorithms. The checking and correcting algorithm for the directional code and boundary marker would have great advantage because of the characteristics of the diagonal direction.

The definition of major diagonal directions and the boundary of the four directions are shown in Figure 8 and the examples of the named strokes are shown in Figure 9 with their names.

Another important feature in hand writing is the information of relative stroke sizes and the relative positions of the top parts and bottom parts of the neighboring strokes. This idea may be implemented in many different manners depending on the writing. For careful writing, the absolute normal size might be implemented for the program efficiency, but for poor writing the relative normal size might be used to check the normal size for each character.

The relative size and positions were classified as following:

1. Normal sized stroke;

Every character has this size of stroke as the first part or last part except the characters as 'f', 'l', 'j', 't'.

The normal size and position of stroke is shown in Figure 10.

2. Above positioned stroke;

Some strokes are significantly large because of the upward extension compared to normal sized strokes. For example, the second stroke of the character 'd' is larger in upward direction compared to a normal sized stroke.

This example is shown in Figure 11.

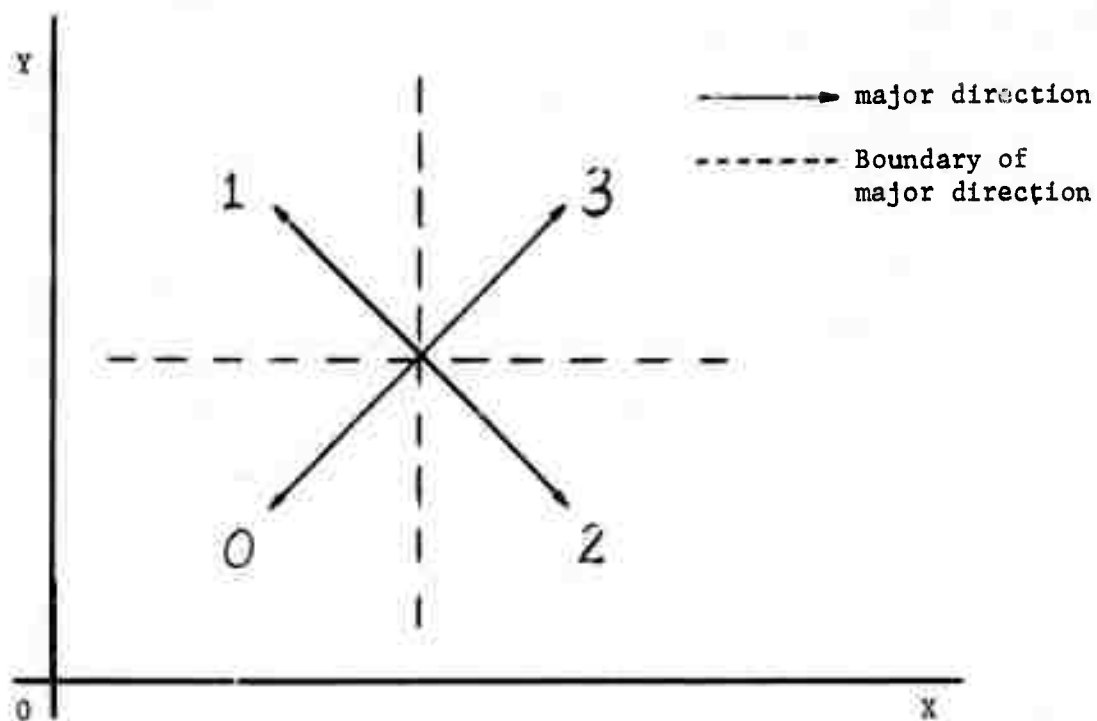


Figure 8. Codings and boundary of the major diagonal directions

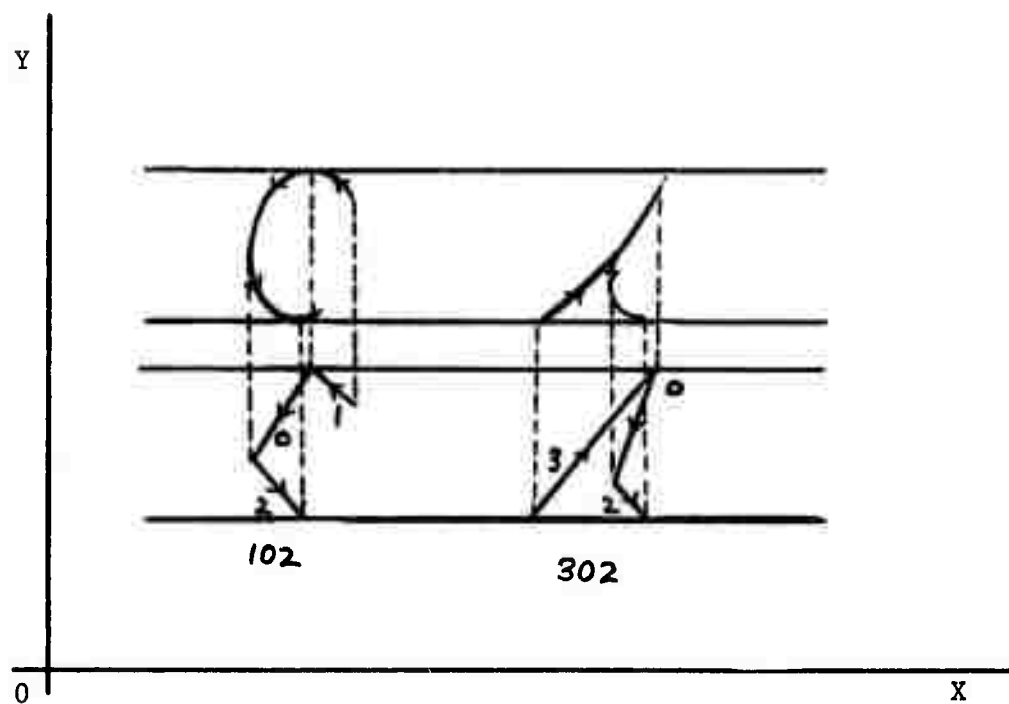


Figure 9. Example of stroke naming algorithm

3. Down positioned stroke;

Some strokes are longer in the downward direction compared to the normal sized stroke.

An example using character 'g' is shown in Figure 12.

4. Full sized stroke;

The character 'f' is the only character having a full sized stroke out of lower case alphabets, which extends above and below a stroke of normal size and position.

The example is shown in Figure 13.

In the program, the positions were named as following:

```
if normal sized then    pocode = 1
if above positioned then    pocode = 2
if down positioned then    pocode = 3
if full sized then    pocode = 4
else then    pocode = 0;
```

3.3 Character Decomposition

The writing cutting algorithm and the stroke breaking algorithm are implemented for each different pattern of each lower case letter of the alphabet. The patterns of each character are picked up from Palmer's note (7) and from the commonly used writings.

The range of patterns which were implemented in this coding determines the capability and the reliability of the system.

The features of decomposed characters are shown in Figure 14. Nearly all writing patterns of lower case alphabets generated by the normal writers who did not have any writing training for the standard writing are coded.

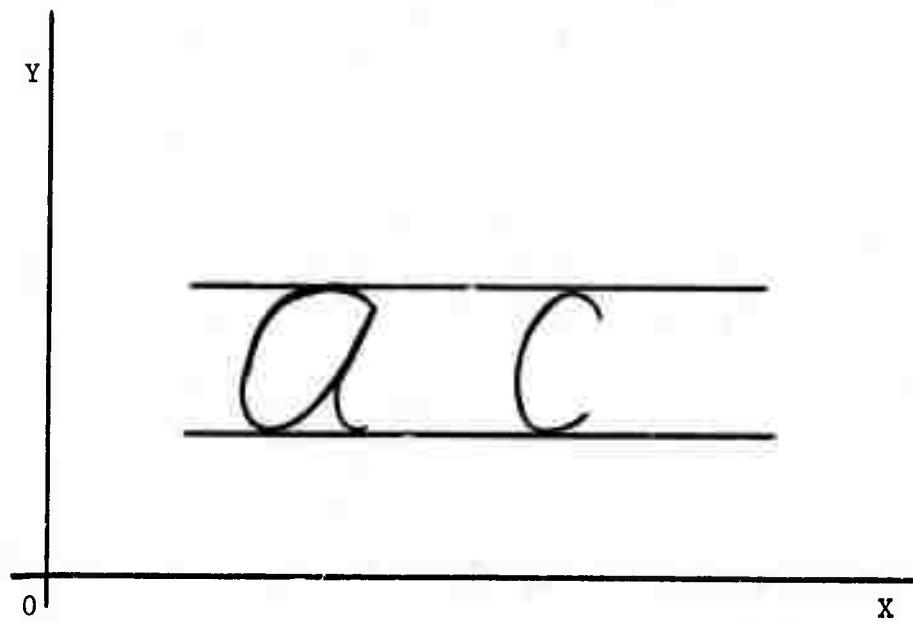


Figure 10. Examples of normal sized stroke

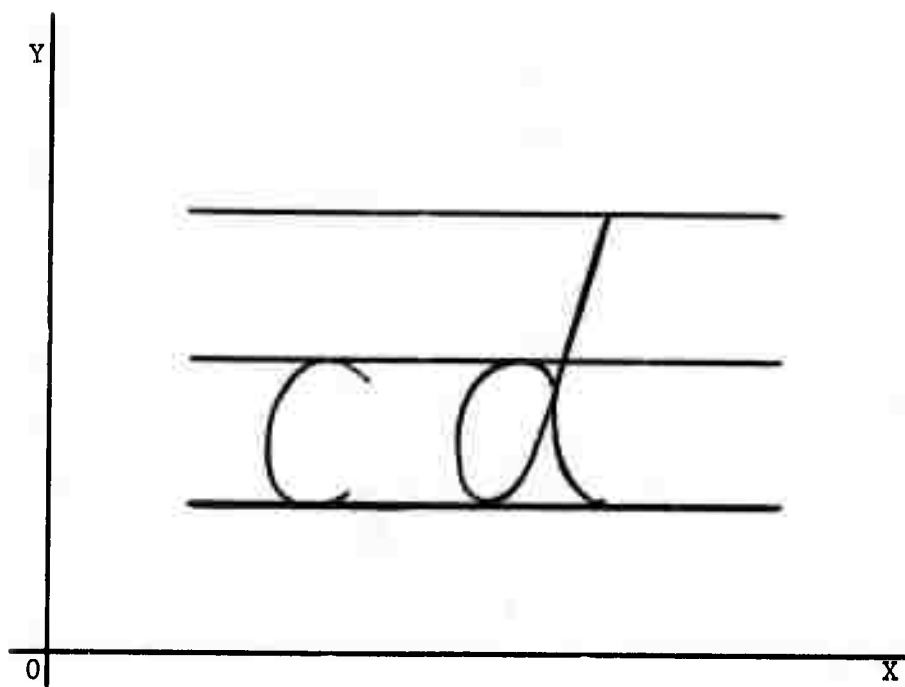


Figure 11. Comparison between normal sized stroke and above positioned stroke.

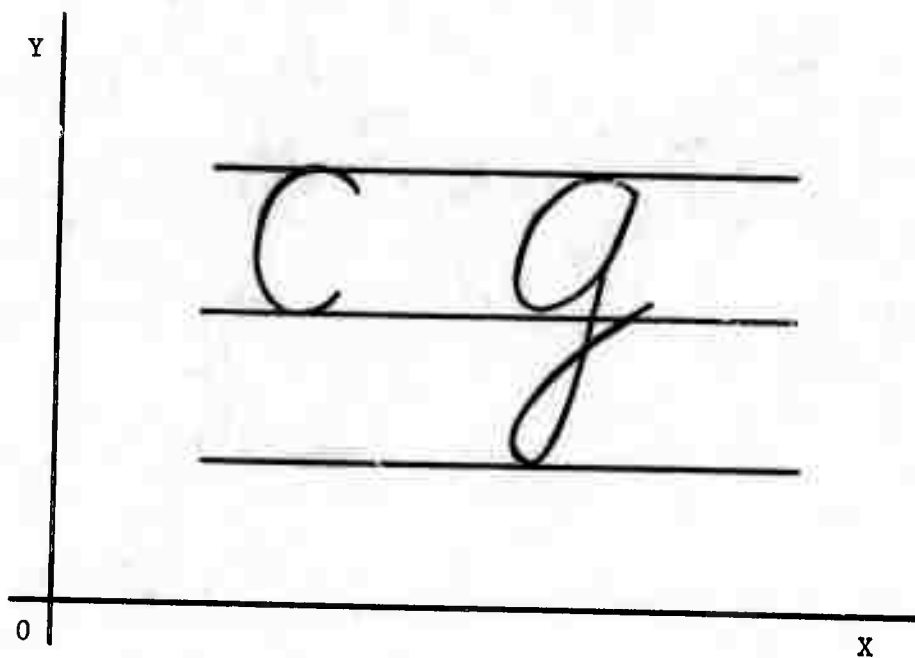


Figure 12. Comparison between normal sized stroke and down positioned stroke

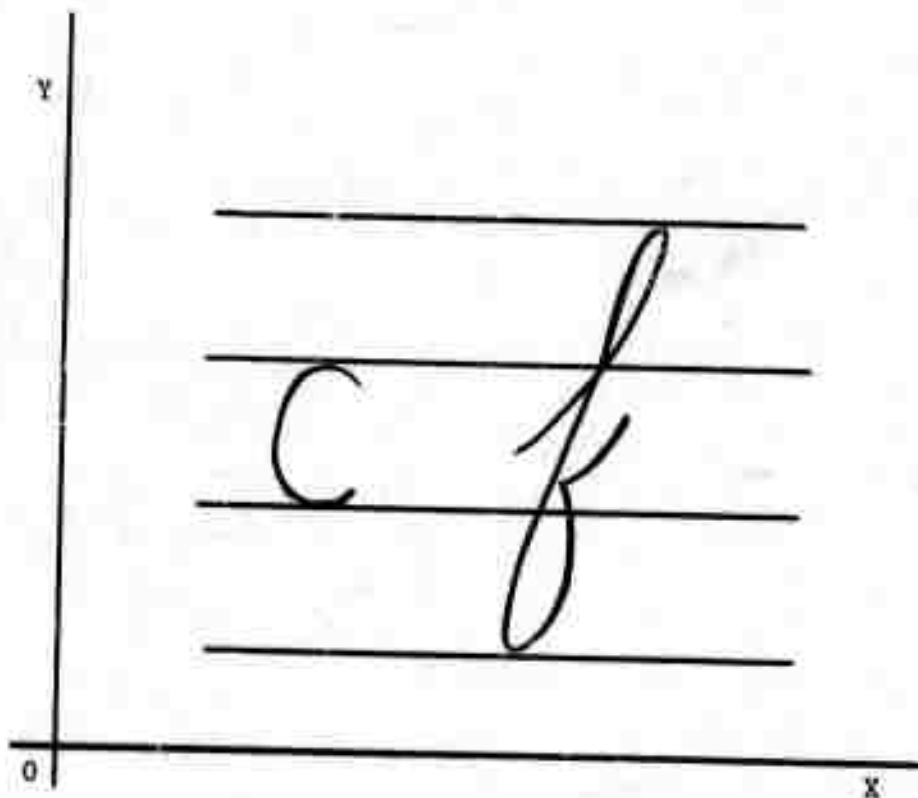


Figure 13. Comparison between normal sized stroke and full sized stroke

The first stroke in the figure is the first part of the character and the second stroke or the last stroke in the figure is the next part of the character for the character composition.

In Figure 15, the possible stroke combinations for each character composition which was listed in Figure 14 by the stroke feature is shown using the stroke name.

There are several different types of character composition in stroke combinations. Some characters have only a single column of strokes and others have two or three columns of strokes. The characters which have a single stroke column will be recognized by the characteristics of the stroke in that column, and the characters which have two columns of strokes would be recognized by identifying the stroke combination between any stroke in the first column and any stroke in the second column. The three column characters would have every possible stroke combination using any one stroke from each column and the characters would be recognized by identifying one of the characteristics in the combinations.

Each stroke is classified by two coding names, the first name is the directional coding and the second name is positional code. The strokes which do not have the positional code are classified by only the directional coding and the positional code is not considered for the particular stroke combination.

The stroke which is negligible in character structure is listed as 'neglected' and the stroke which is negligible in stroke combination and is significant for evaluation of stroke characteristics is listed as 'optional' in the stroke column of Figure 15.

a a a
a-1 a-2 a-3

< / 5 7 | *Λ Λ Λ*
First Stroke (a-1) Second Stroke (a-1)

Λ Λ Λ | *Λ Λ Λ*
First Stroke (a-2) Second Stroke (a-2)

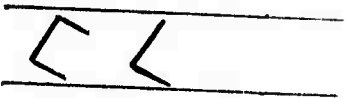
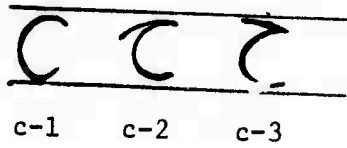
2 7 | *Λ Λ Λ*
First Stroke (a-3) Second Stroke (a-3)

b b
b-1 b-2

Λ Λ Λ Λ | *Λ Λ*
First Stroke (b-1) Second Stroke (b-1)

Λ Λ Λ Λ | *Λ /*
Second Stroke (b-2) Second Stroke (b-2)

Figure 14. Character structures by the stroke feature



Strokes (c-1)



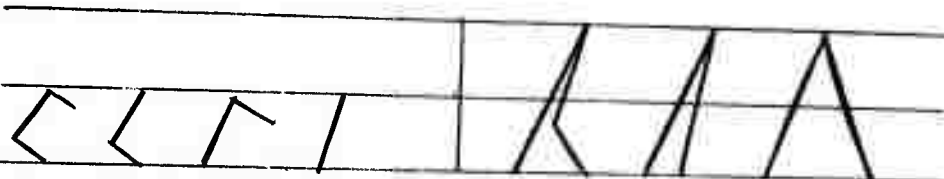
Strokes (c-2)



Strokes (c-3)



d-1 d-2 d-3



First Stroke (d-1)

Second Stroke (d-1)



First Stroke (d-2)

Second Stroke (d-2)

Figure 14. Continued, the second

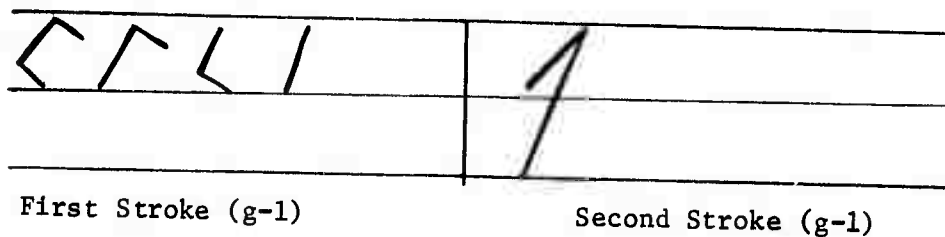
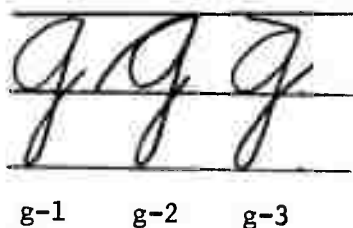
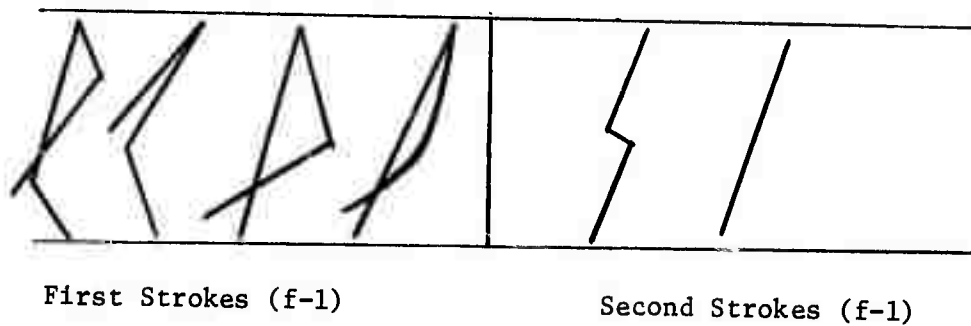
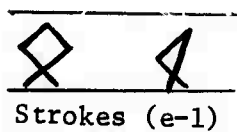
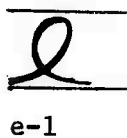
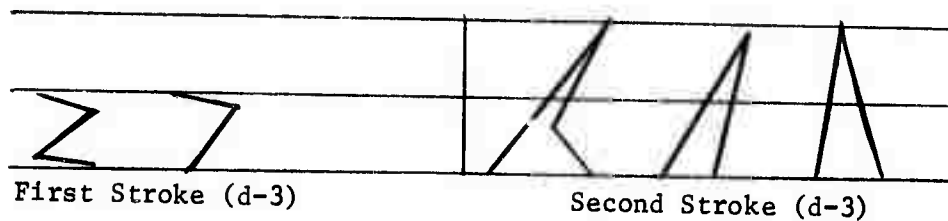


Figure 14. Continued, the third

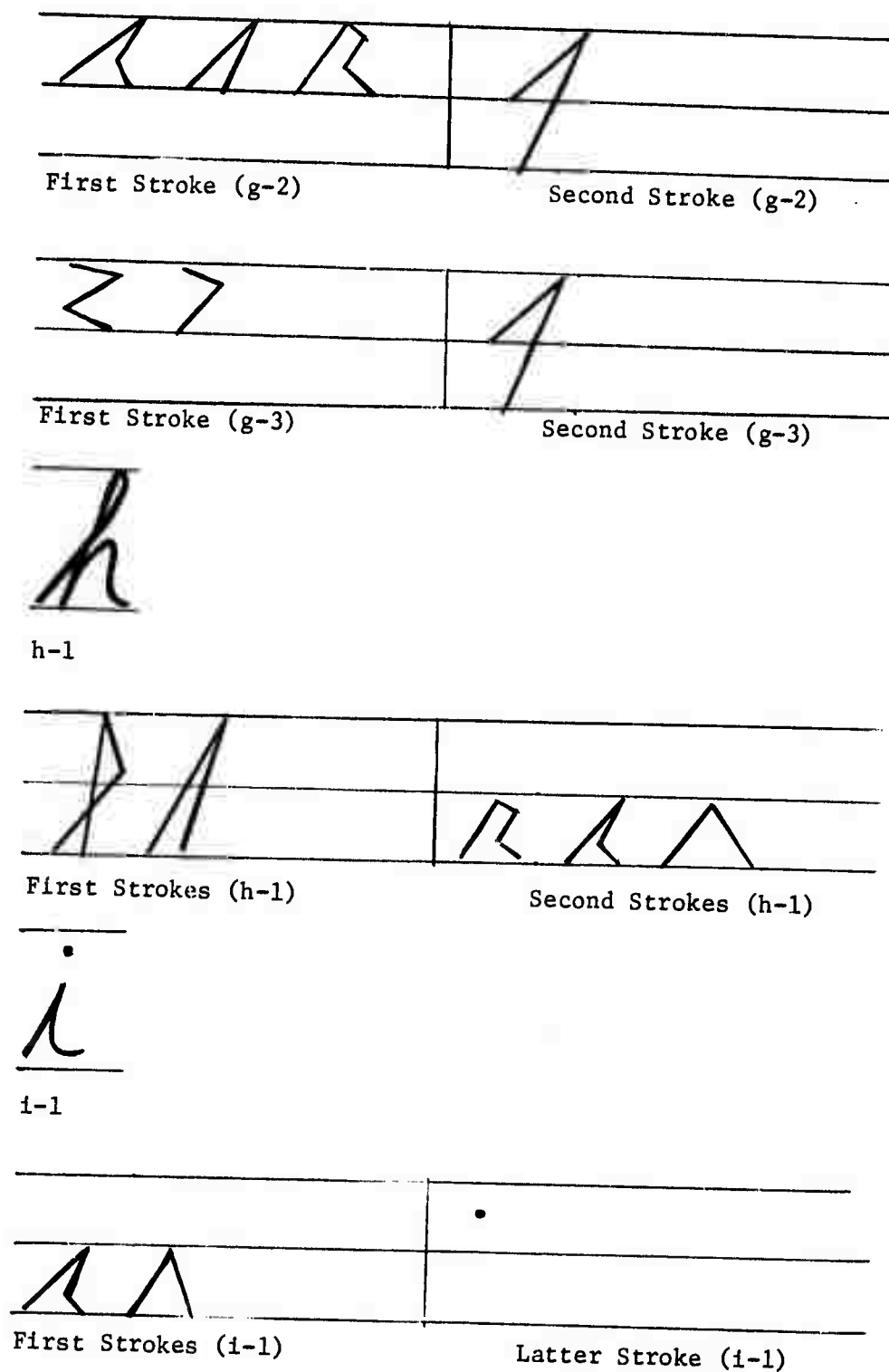
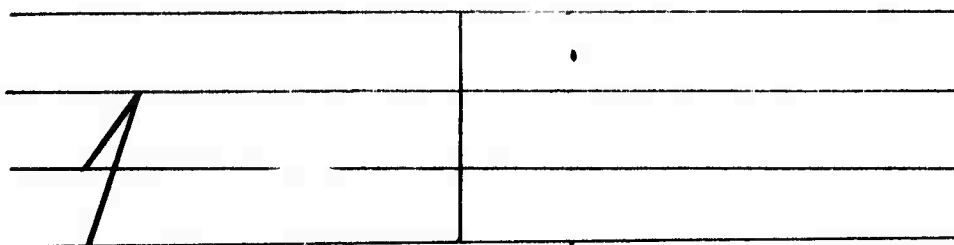


Figure 14. Continued, the fourth

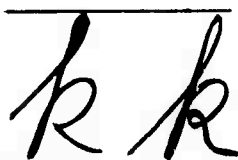


j-1



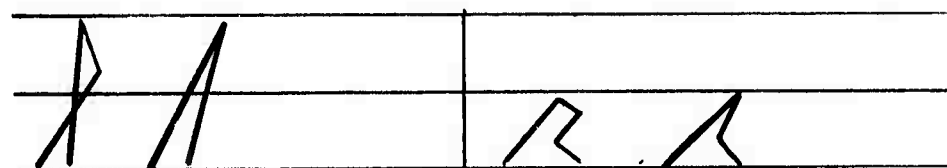
First Stroke (j-1)

Latter Stroke (j-1)



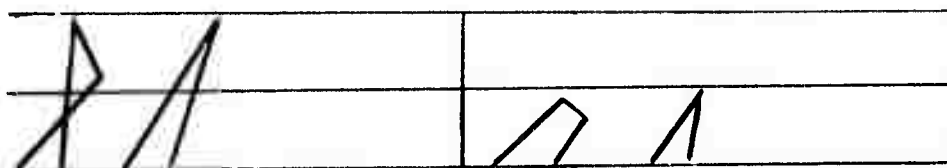
k-1

k-2



First Strokes (k-1)

Second Strokes (k-1)

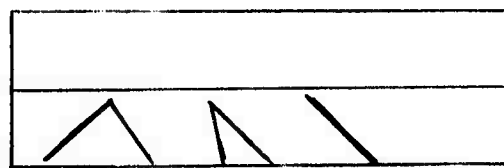


First strokes (k-2)

Middle Strokes (k-2)



l-1



Last Strokes (k-2)

Figure 14. Continued, the fifth



Strokes (1-1)



m-1

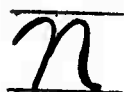


First Stroke (m-1)

Middle Stroke (m-1)



Last Stroke (m-1)



n-1



First Stroke (n-1)

Second Stroke (n-1)



o-1

o-2

o-3



First Stroke (o-1)

Second Stroke (o-1)

Figure 14. Continued, the sixth

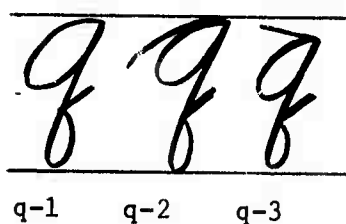
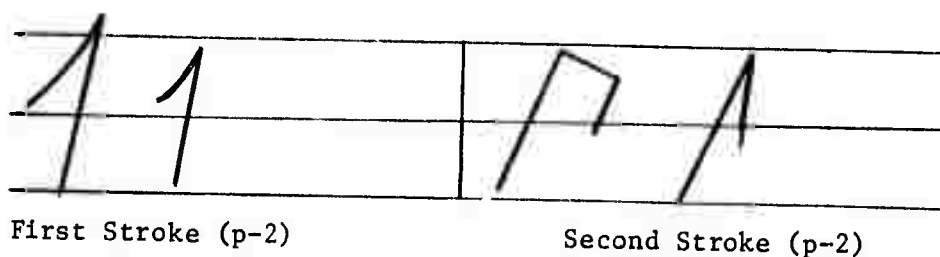
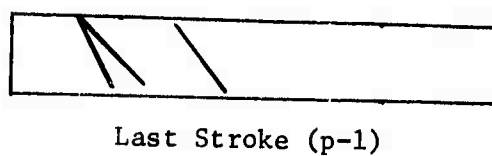
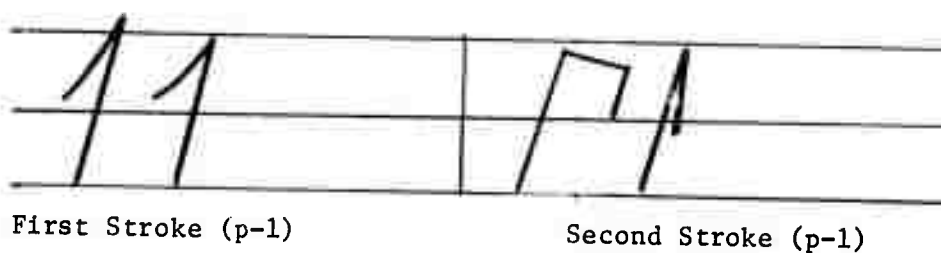
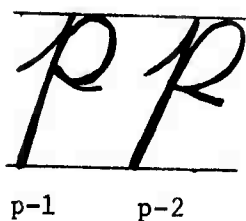
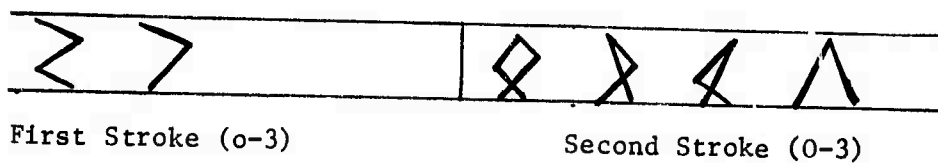
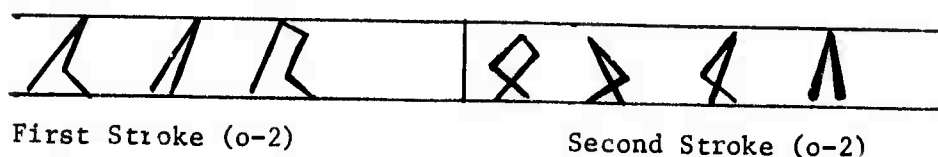
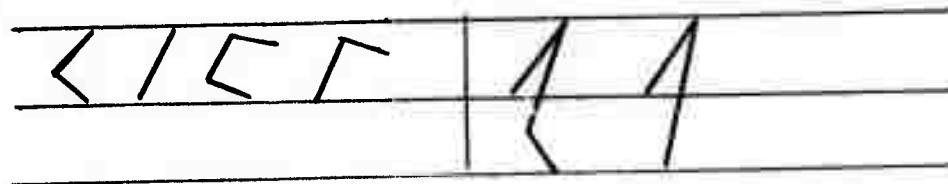
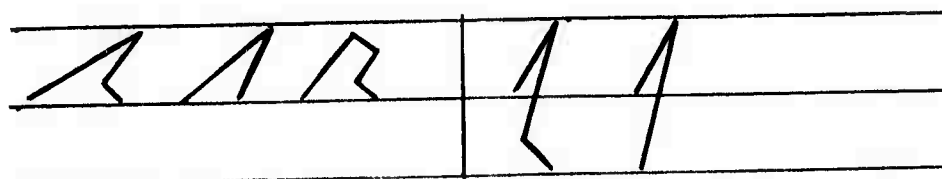


Figure 14. Continued, the seventh.



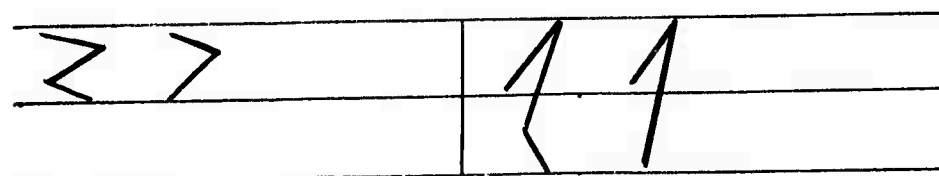
First Stroke (q-1)

Second Stroke (q-1)



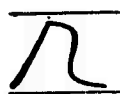
First Stroke (q-2)

Second Stroke (q-2)

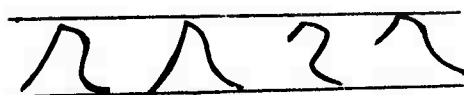


First Stroke (q-3)

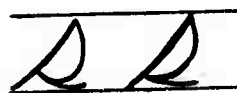
Second Stroke (q-3)



r-1



Strokes (r-1)



s-1 s-2



First Stroke (s-1)

Last Stroke (s-1)

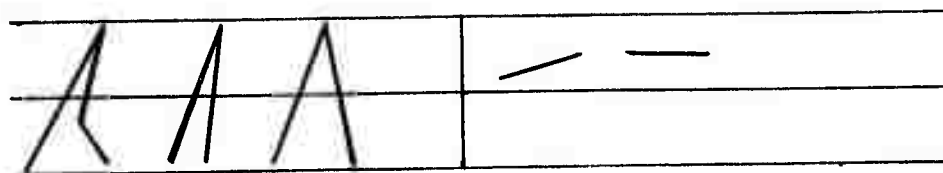


Strokes (s-2)

Figure 14. Continued, the eighth



t-1



First Stroke (t-1)

Second Stroke (t-1)

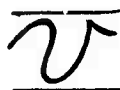


u-1



First Strokes (u-1)

Second Strokes (u-1)



v-1



First Stroke (v-1)

Second Stroke (v-1)



w-1



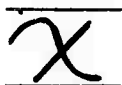
First Stroke (w-1)

Middle Stroke (w-1)

Figure 14. Continued, the ninth



Last Stroke (w-1)



x-1

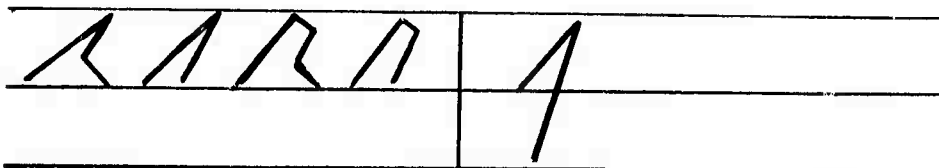


First Stroke (x-1)

Second Stroke (x-1)

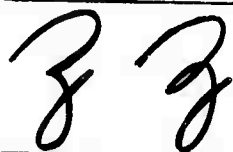


y-1



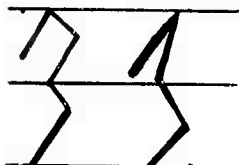
First Stroke (y-1)

Second Stroke (y-1)

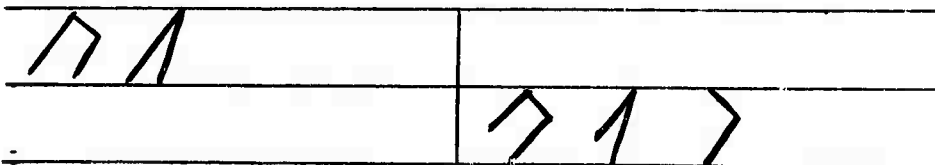


z-1

z-2



Strokes (z-1)



First Stroke (z-2)

Second Stroke (z-2)

Figure 14. Continued, the tenth

Character Patterns	First Stroke	Second Stroke	Third Stroke
<i>a</i>	02-1 01-1 102-1 10-1	302-1 30-1 32-1	Neglected
<i>ā</i>	302-1 30-1 3202-1	302-1 30-1 32-1	Neglected
<i>ä</i>	202-1 20-1	302-1 30-1 32-1	Neglected
<i>b</i>	3102-2 310-2 302-2 30-2	312-1 32-1	Neglected
<i>β</i>	3102-2 310-2 302-2 30-2	312-1 32-1	Neglected
<i>c</i>	102-1 02-1	Neglected	
<i>τ</i>	302-1 30-1 3202-1	Neglected	

Figure 15. Character structures by the stroke name

Character Patterns	First Stroke	Second Stroke	Third Stroke
Ʒ	202-1 20-1	Neglected	
d	102-1 02-1 10-1 0-1	302-2 30-2 32-2	Neglected
d	302-1 30-1 3102-1	302-2 30-2 32-2	Neglected
d	202-1 20-1	302-2 30-2 32-2	Neglected
e	3102-1 302-1	Neglected	
f	3102-4 302-4 310-4 30-4	Optional	
g	102-1 10-1 02-1 0-1	30-3	Optional

Figure 15. Continued, the second

Character Pattern	First Stroke	Second Stroke	Third Stroke
	302-1 30-1 3202-1	30-3	Optional
	202-1 20-1	30-3	Optional
	310-2 30-2	3202-1 302-1 32-1	Neglected
	302-1 32-1	Neglected	7
	30-3	Optional	7
	310-2 30-2	3102-1 302-1	
	310-2 30-2	320-1 ¹ 30-1 ¹	32-1 ¹ 12-1 ¹ 2-1 ¹
	3102-2 302-2 30-2	Neglected	
	30-1 20-1 320-1	320-1 30-1	3202-1 302-1 320-1 30-1 32-1

Figure 15. Continued, the third

¹The positional coding is not required in stroke combination

Character Pattern	First Stroke	Second Stroke	Third Stroke
<i>n</i>	30-1 20-1 320-1	3202-1 302-1 320-1 30-1 32	Neglected
<i>o</i>	02-1 0-1 102-1 10-1	3102-1 312-1 302-1 32-1	Neglected
<i>o</i>	302-1 30-1 3202-1	3102-1 312-1 302-1 32-1	Neglected
<i>o</i>	202-1 20-1	3102-1 312-1 302-1 32-1	Neglected
<i>p</i>	30-4 30-3	320-1 ¹ 30-1 ¹	12-1 ¹ 2-1 ¹
<i>p</i>	30-4 30-3	320-1 30-1	Neglected
<i>p</i>	30-4 30-3	3202-1 302-1	Neglected

Figure 15. Continued, the fourth

¹ The positional coding is not required in stroke combination

Character Pattern	First Stroke	Second Stroke	Third Stroke
	02-1 0-1 102-1 10-1	302-3 30-3	Optional
	302-1 30-1 3202-1	302-3 30-3	Optional
	202-1 20-1	302-3 30-3	Optional
	3202-1 32-1	Neglected	
	30-1 ¹ 320-1 ¹	12-1 ¹ 2-1 ¹	Neglected
	30-1 320-1	Neglected	
	302-1 3202-1	Neglected	
	302-2 30-2 32-2	Neglected	3-1 7
	302-1 30-1 32-1	302-1 30-1 32-1	Neglected

Figure 15. Continued, the fifth

¹The positional coding is not required in stroke combination

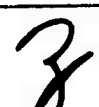
Character Pattern	First Stroke	Second Stroke	Third Stroke
	320-1 30-1 20-1 3202-1 302-1	302-1 32-1	Neglected
	302-1 30-1 02-1 0-1	302-1 30-1 32-1	302-1 32-1
	32-1 302-1	Neglected	0-1 ¹ 7
	302-1 30-1 3202-1 320-1	30-3	Optional
	32020-3 ¹ 3020-3 ¹	Optional	
	320-1 30-1	320-3 ¹ 30-3 ¹ 20-3 ¹	Optional

Figure 15. Continued, the sixth

¹The positional coding is not required for the stroke combination

CHAPTER IV

SYNTAX FOR CURSIVE WRITING RECOGNITION

4.1 A Hierarchical Organization by Stroke Characteristics in the Character Structure

The characteristics of character structure which were described in the previous chapter are used in this section for the construction of a syntax for cursive writing.

The group of strokes for each level is selected by the stroke characteristics and stroke functions. The selection is made such that a processing at a given level is independent of any lower level process that may exist.

The highest level is processed first using its own strong characteristics with highest reliability for recognition of the writing. Then the next higher level is processed in the same manner using its own characteristics with a high reliability for the recognition and without further reference to any higher levels.

The other low levels would follow the above routine for their individual processing steps until all the strokes were processed.

The Baye's theorem (8) can analyze this organization of writing recognition in terms of conditional probability. As long as the higher level keeps better reliability, this syntactic organization would promise optimal reliability for the entire experiment.

Each characteristic of the stroke is classified and the system hierarchy is listed in the following:

I. Unique characters:

Some stroke can occur in a particular character and this character can be recognized immediately. Such strokes are independent of any other stroke in the system. Characters containing these strokes are recognized at the highest level of priority. An example is the letter 'z'.

II. Stairway characters:

Some characters have a stroke which has another stroke positioned just underneath of the first stroke to build a stairway.

Examples: 'z', 's', 'p', 'k'

III. Intersecting characters:

Some stroke intersects another stroke to build a character.

Examples: 't', 'x'

IV. Pointed characters:

A short stroke points certain positions regard to some strokes to specify some characters.

Examples: 'i', 'j'

V. Circled characters:

Some characters begin with a circle and the first stroke of the circle is required to be the normal sized stroke.

Examples: 'a', 'd', 'g', 'o', 'q'

VI. First-fixed character

Some strokes are employed to build a character only as the first stroke of the character. The size of this group can be varied by changing the order of the hierarchy.

Examples: 'h', 'l'

VII. Osculated characters:

Checking the osculating property between the neighboring strokes, it is possible to realize whether they are members of the same character.

Examples: 'm', 'n'

VII. Last-fixed character:

Some strokes are employed to build a character only as the last stroke of the character. The size of this group will vary depending on the priority of this level

Examples: 'v', 'w'

IX. Relative Characters:

Other strokes which do not belong to the above groups are called relative strokes. They will be processed as relative strokes by the information which they carry for further decision. The algorithm will be discussed in later chapters using more complicated routines.

4.2 Syntax Specifications

The organization of syntax would vary depending on the system specifications such as the range of writing pattern and carefulness of writers, even for the same text. This specification would be very complicated for the writings of uncaredful and untrained users, because a correcting system has to be implemented in addition to the basic system.

A self correcting algorithm was used in an early paper (6) for character recognition using a reference generator for the feedback loop.

A different implementation for the correcting loop is used for this cursive writing recognition using syntax-directed logic.

In Table I the syntax specifications are shown using a list procedure language in a formalism similar to Bakus normal form (1).

The first classification, second classification, third classification, fourth classification, fifth classification, sixth classification would form a complete routine for writing recognition in case of the users who write well.

A correcting loop which includes the third and fourth classification is inserted between the second classification and the fifth classification of the above system to correct errors in third classification and fourth classification of the system.

A new system was necessary, because the priority of operation is changed by the new considerations. The complexity of an error correcting system is a function of the error priority which is determined by the types of error and the levels of error considered.

In this work the characters 'b' and 'o' are checked and corrected for some users who fail to give enough down cave of the second stroke of the character.

A correcting loop is shown in the syntax specifications of Table I, and the iteration idea is also shown in the entries of the specifications. The semantics is explained in the next section.

4.3 Semantic Interpretations

The figure extracting procedures are applied to the structure of the sampled data system to name each stroke for the next recognizing algorithm.

TABLE I

Syntax specifications for handwriting recognition
with self correcting loop

<characters>	:=<first classification><second classification><correcting loop><fifth classification><sixth classification>
<correcting loop>	:=<o-correction><b-correction>
<o-correction>	:=<third classification><first classification><second classification>
<b-correction>	:=<fourth classification><first classification><second classification><third classification>
<first classification>	:=<unique character><stairway character>
<second classification>	:=<intersecting character><pointed character>
<third classification>	:=<circled character>
<fourth classification>	:=<first-fixed character>
<fifth classification>	:=<osculated character><last-fixed character>
<sixth classification>	:=<relative character>
<unique character>	:= 'z'
<stairway character>	:= 'z'/'s'/'k'/'p'
<intersecting character>	:= 'x'/'t'
<pointed character>	:= 'i'/'j'
<circled character>	:= 'a'/'d'/'g'/'o'/'q'
<first-fixed character>	:= 'b'/'c'/'l'/'f'/'h'/'k'/'r'/'w'
<osculated character>	:= 'm'/'n'
<last-fixed character>	:= 'r'/'w'
<relative character>	:= 'c'/'e'/'p'/'v'/'s'/'u'/'v'/'w'/'y'

After each stroke is named by the early procedures, the routines of classification are called to check the characteristics of each stroke and classify the strokes by their individual characteristics.

Each level of classification is studied and explained in the following, displaying the structure and characteristics of the level.
<first classification>:

The string of the strokes is checked and classified for the first time during this routine, and the following characters are recognized:

<unique character>:

The following strokes are coded as the unique characters. The symbol $SD(I)$ is defined as the I th stroke of the string and $CHA(I)$ is defined as the I th character of the string.

if $SD(I) = 32020^1$ or 3020^1 or 2020^1
then $CHA(I) = 'z'$

<stairway character>:

The stairing property is checked between the neighboring strokes for certain strokes, and the following stroke sequences are recognized as the stairing characters.

if $SD(I) = 320^1$ or 30^1
and $SD(I+1) = 320$ or 30 or 20
then $CHA(I) = 'z'$
and $SD(I+1) = 32$ or 12 or 2
then $CHA(I) = 's'$

¹These strokes do not have positional code and they are used regardless of the positional value.

and $SD(I-1) = 30$ or 310

and $SD(I+1) = 32$ or 12 or 2

then $CHA(I) = 'p'$ or $'k'$

Each sequential combination might require further classifying routines depending on the number of members in that class.

<second classification>

After the first classification, the second classification follows to check the second level characteristics. The following characters belong to this level.

<intersecting character>:

The intersecting property is checked in this step and the strokes were coded as following:

if $SD(I) = 0$ or 7

and <other strokes which are intersected by above strokes>

then $CHA(I) = 'x'$ or $'t'$

<pointed character>:

The pointing property is checked during this step and the following strokes are coded as the pointed characters:

if $SL(I) = 7$

and <other strokes which are pointed by above stroke>

then $CHA(I) = 'i'$ or $'j'$

<o-correction>

The characteristics of character 'o' are checked and other errors will be corrected which were generated by the character 'o'.

<third classification>:

The circled characteristics are checked for this level characters.

<circled character>:

The following strokes were coded to the characters of this level.

if SD(I) = 0-1 or 10-1 or 02-1 or 102-1

then check character 'Q' and recognize it

and correct the related informations

and SD(I+1) = 30-2 or 32-2 or 302-2

then CHA(I) = 'd'

and SD(I+1) = 302-3

then CHA(I) = 'q'

and SD(I+1) = 30-3

then CHA(I) = 'g' or 'q'

and SD(I+1) = 30-1 or 32-1 or 302-1

then CHA(I) = 'a'

if SD(I) = 302-1 or 3102-1 or 3202-1 or 30-1 or 202-1 or
20-1

then check the character 'Q' and recognize it

and correct other related informations

and SD(I+1) = 30-2, 32-2, 302-2

then CHA(I) = 'd'

and SD(I+1) = 302-3

then CHA(I) = 'q'

and SD(I+1) = 30-3

then CHA(I) = 'g' or 'q'

and $SD(I+1) = 30-1$ or $32-1$ or $302-1$

then $CHA(I) = 'a'$

Some classes might require further discriminations depending on the complexity of the class.

<first classification>, <second classification> in this level is same as defined at early step.

<b-correction>:

The characteristics of a character '*b*' are checked and the distorted information by the character '*b*' is corrected during this procedure.

<fourth classification>:

The strokes which are the first stroke of each character are checked to be classified as this level of character.

<first-fixed character>:

The following strokes are coded as the first-fixed characters

if $SD(I) = 3102-2$ or $302-2$

then check the characteristics of the character '*b*' and recognize it. Correct all related informations

if $SD(I) = 310-2$ or $30-2$

and $SD(I+1) = 32-1$

then $CHA(I) = 'h'$ or $'b'$

correct informations related to '*b*'

and $SD(I) = 3202-1$ or $302-1$

then $CHA(I) = 'h'$ or $'k'$ or $'b'$

correct informations related to '*b*'

and $SD(I+1) = 30-1$

then $CHA(I) = 'h' \text{ or } 'b'$

correct informations related to 'b'

else check character 'b' and recognize it

correct informations related to 'b'

and <first classification>, <second classification>, <third classification> are same as defined at early steps.

<fifth classification>:

The osculating characters and the last-fixed characters are checked during this step.

<osculating characters>:

The following strokes are coded to be the characters of this level

if $SD(I) = 32-1 \text{ or } 320-1 \text{ or } 30-1 \text{ or } 20-1$

and $SD(I+1) = 302-1 \text{ or } 3202-1$

then $CHA(I) = 'n'$

and $SD(I+1) = 32-1 \text{ or } 320-1$

then $CHA(I) = 'n'$

and $SD(I+2) = 30-1 \text{ or } 32-1 \text{ or } 302-1 \text{ or } 3202-1 \text{ or } 320-1$

then $CHA(I) = 'm'$

<last-fixed character>:

The following strokes are coded as the last-fixed characters

if $SD(I) = 312-1$

and $SD(I-1) = 320-1 \text{ or } 302-1 \text{ or } 30-1 \text{ or } 202-1 \text{ or } 20-1$

then $CHA(I) = 'v'$

<sixth classification>:

This step collects all other strokes which are not processed previously until this step.

<relative character>:

The following strokes are coded as the relative character. The detailed coding procedures for this particular character are analyzed in the next chapter.

```

if SD(I) = 32-1
    then CHA(I) = 'r' or 'u' or 'v' or 'w'
if SD(I) = 320-1
    then CHA(I) = 's' or 'r' or 'v' or 'y'
if SD(I) = 3202-1
    then CHA(I) = 'r' or 's' or 'v' or 'y'
if SD(I) = 30-1
    then CHA(I) = 's' or 'u' or 'v' or 'w' or 'y'
if SD(I) = 30-3
    then CHA(I) = 'p'
if SD(I) = 302-1
    then CHA(I) = 'c' or 'e' or 'r' or 's' or 'u' or 'v' or 'w'

```

The entire program is listed in Appendix 9.1.

CHAPTER V

IMPLEMENTATION OF THE SYNTAX ANALYZER

5.1 System Analyzer

A string of strokes is ready to be processed by the recognizing algorithms after a writing is completely coded.

The first classification checks the entire string of strokes as a beginning part of the recognizing algorithm. The unique characters are encoded from the stroke characteristics and the stairway characters are coded from the stairing characteristics.

The stairing properties are defined for a couple of different cases as following:

In Figure 16

if $(Y(3)-Y(2))/(Y(1)-Y(2))$ less than 0.7 and
 $(Y(3)-Y(2))/(Y(3)-Y(4))$ less than 0.7 and
 $(X(3)-X(1))$ less than $(Y(1)-Y(2))$ then
 stairing is true.

In Figure 17

if $X(3)$ less than $X(1)$ and
 $X(4)$ greater than $Y(3)$ then
 stairing is true.

In the second classification level, the intersecting characters and the pointing characters are coded by their characteristics.

The intersecting characteristic is a geometric intersection between any two strokes, and the pointing characteristic is a simple

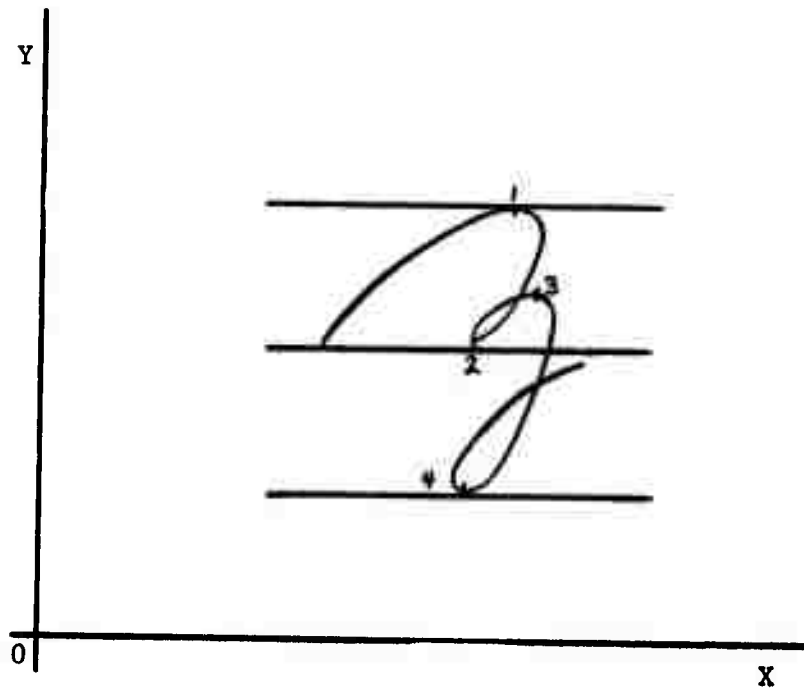


Figure 16. Stairing characteristics for character 'z'

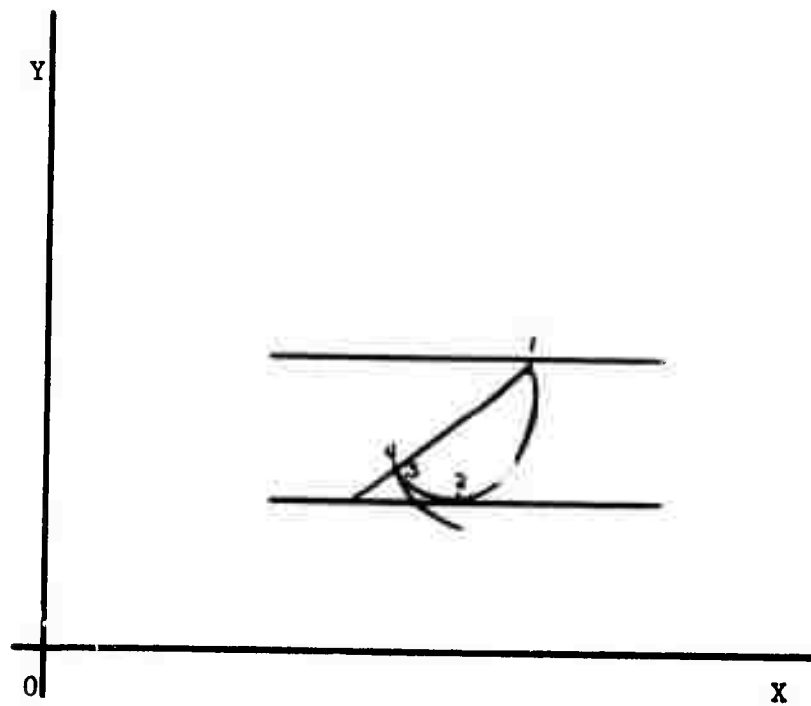


Figure 17. Stairing characteristics for character 's'

checking for the distance between the point and a head of the strokes. Such an algorithm can be checked by evaluating a simple arithmetic feature for the characters.

The complete system is shown in Figure 18 and the correcting algorithm will be discussed at the next step.

During the third classification the circled characteristics will be checked before any correcting procedure is applied.

The circled characteristic is defined as follows:

In Figure 19

if position (2) - position (1) is less than $NY/4$
then circled is true.

After the circled property is checked, the character o-correction algorithm is called and the other errors are checked and corrected.

In Figure 20

if OX is greater than $NY/5$ or NY/OY is less than 2.0
then o-correction is true.

As soon as the o-correction is checked, the neighboring stroke will be redefined and the number of the iteration will be incremented by one for each correction.

This additional iteration will recall the classifications which were previously made. Since the error is corrected and the neighboring information is corrected, it is very necessary to scan the system again and reprocess the entire string.

The b-correction routine is processed at the next classification step. The characteristics of the character 'b' are checked and then

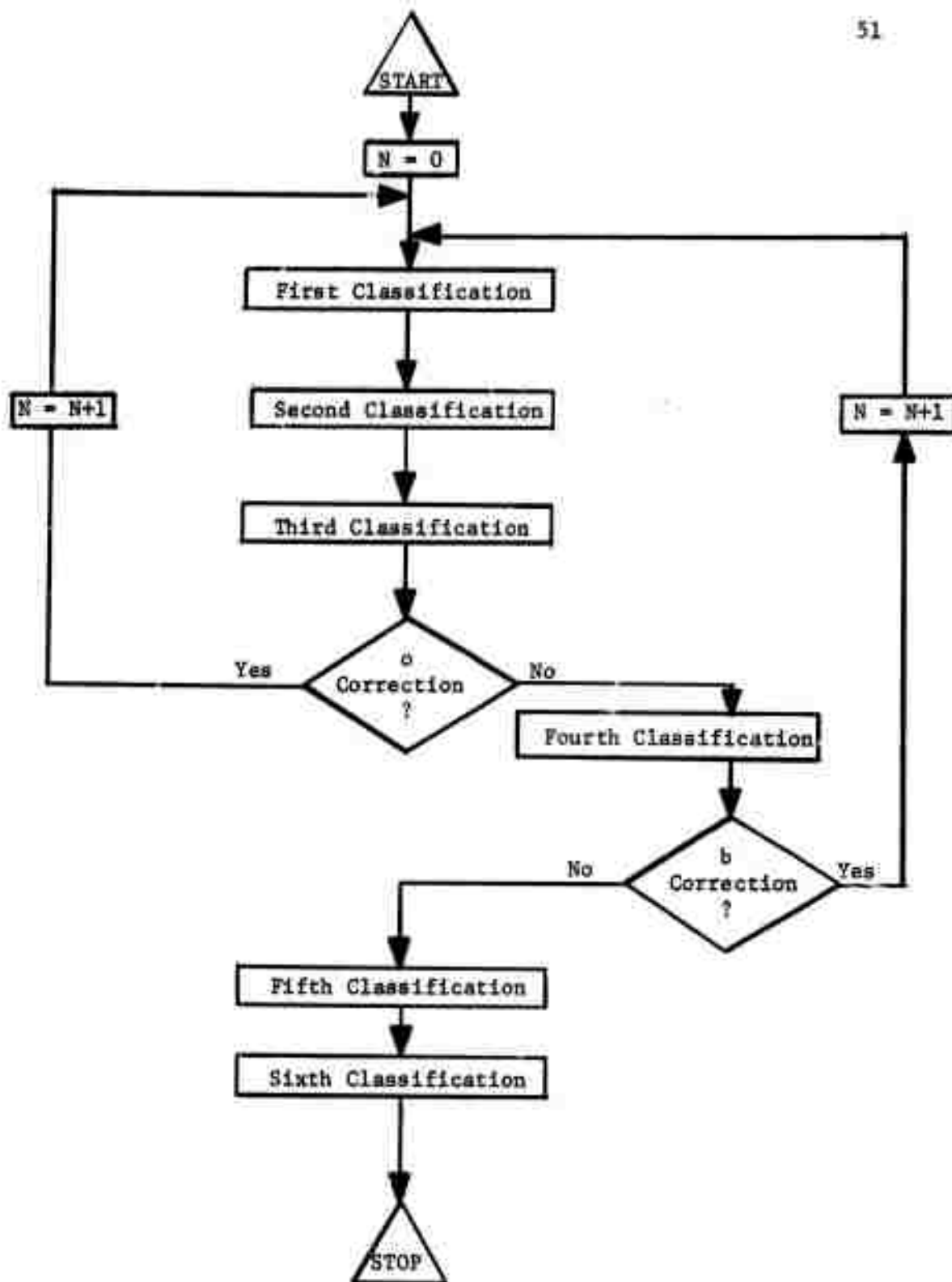


Figure 18. System analyzer diagram

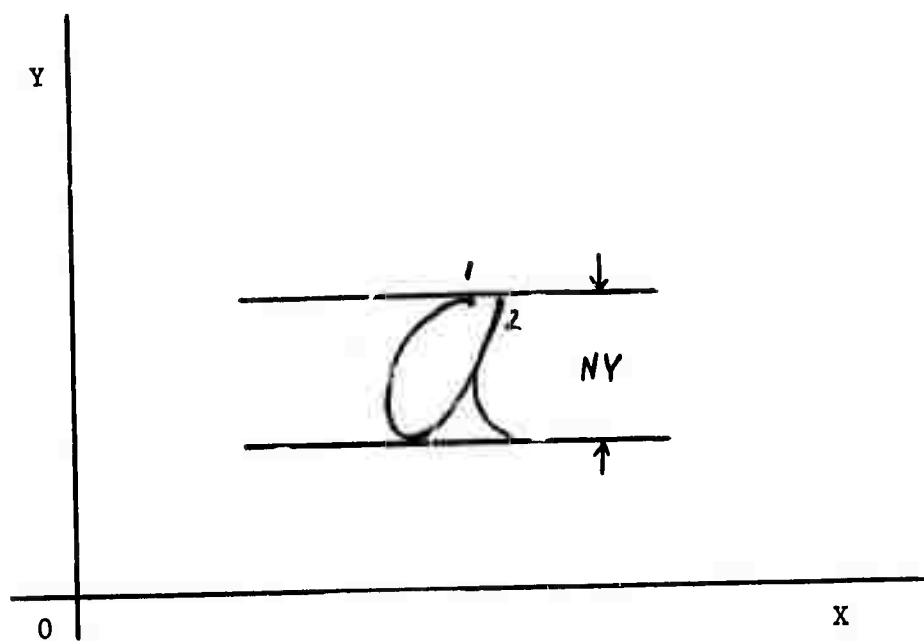


Figure 19. Characteristics for the circled character

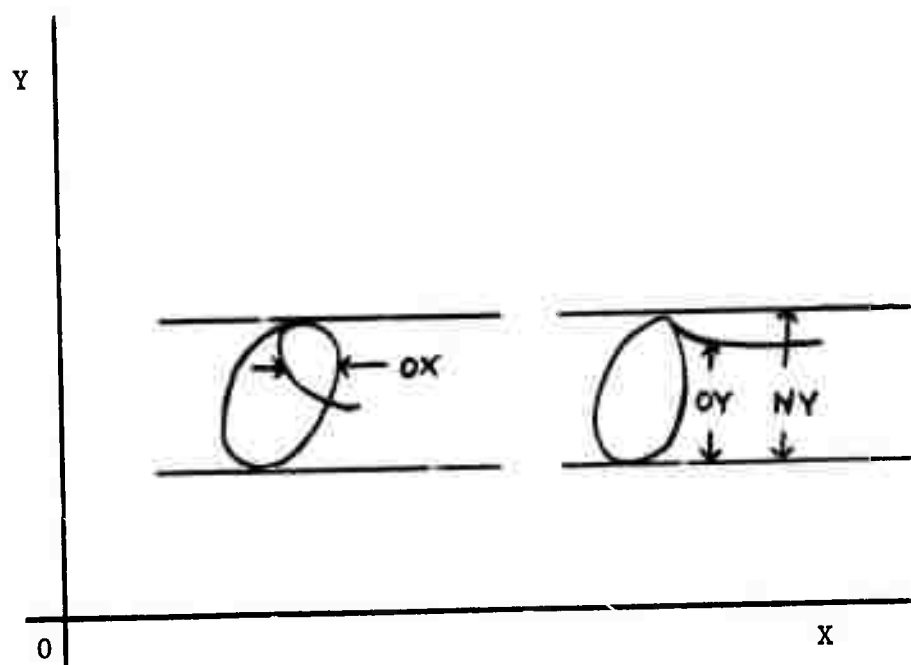


Figure 20. Characteristics for o-correction.

other correcting routines are called for the correction of the distorted information which was generated by the error in the character 'b'.

The characteristics of the character 'b' are defined as following:

In Figure 21

if (position (2) - position (1)) is less than
 (position (1) - position (3))/4
 then b-correction is true.

As soon as the b-correction is suggested by the error checking routine, the neighboring stroke would be checked and corrected and the number of the iteration is incremented by one. During this iteration, the higher classification level will be called to recheck the entire string including the newly corrected strokes.

At the fifth step of classification, the osculated characters and the last-fixed characters are checked and processed.

The osculating property is defined as following:

In Figure 22

Measure ND at each point of the line from point 1 to 2
 count the number of times for the following cases:
 case 1: ND less than NY/5
 case 2: ND greater than NY/5

if case 1 is more than case 2 then osculation is true.

The character 'm' requires another osculating operation for the next stroke to identify the property of character 'm' from character 'n'.

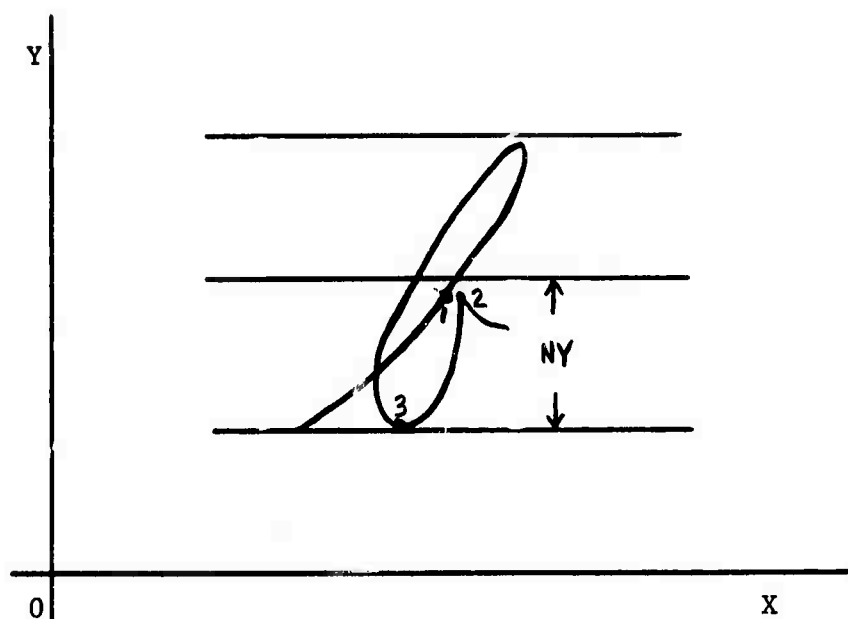


Figure 21. Characteristics for b-correction

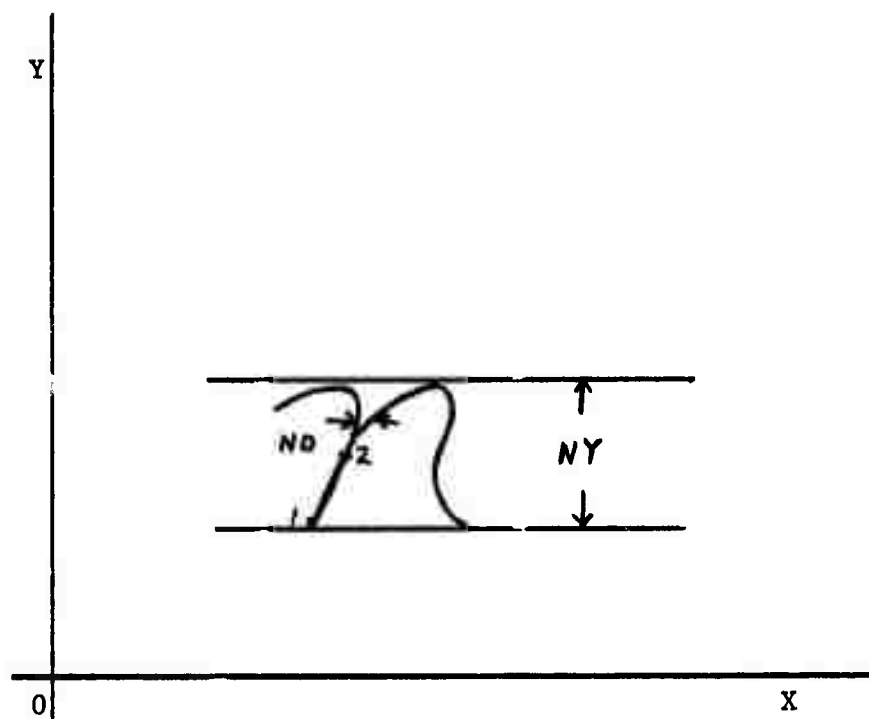


Figure 22. Characteristics for osculated character

The last-fixed characters are coded from the stroke characteristics and stroke functions for the particular strokes.

The remaining strokes which were not processed by the previous classifications are collected to be processed by the local analyzer to use the idea of the relatively discriminating classification.

5.2 Character Compositions

Most characters are recognized by the combination of the matched strokes during the classifying steps. The syntax organization is specified for the optimal combination of strokes by their characteristics in this work.

The combinations in higher level strokes are more solidly defined and the recognition is more reliable than that of lower level groups.

The unique characters, the intersecting characters and the pointed characters would have special combinations for the character composition because their classifications are specified quite uniquely by their characteristics as pointed out in the section on semantic explanation.

For the stairing characters, the characters 'h', 'k', are discriminated from the characteristics of character 's' using the local routine HKVSL and the character 'h' or 'k' is identified by the local routine HVSK for the final decision of recognition.

The local discriminating routines are studied in the next section of this chapter in more detail.

The possible number of stroke combinations for the low level group of strokes is increased and the discriminating algorithms are more complicated for each class of combination in this lower level.

The circled characters have the fixed stroke combination for the character composition, and the stroke combinations for this classification are defined in Table II. The row strokes are the first strokes of the characters defined in the boxes and the column strokes are the last strokes of the characters defined in the boxes.

In Table II, most boxes are well defined with a single character, and some other boxes have two characters. For the boxes having characters 'g' and 'q' together in that box a discriminating routine GVSQ is called to evaluate the comparative characteristics between the characters and for the boxes having characters 'a' and 'g' together in that box, the character o is picked out by the o-correcting routine before getting any other process for this level.

The local routine GVSQ is presented in next section for further discussions.

For the first-fixed characters, the character combinations are shown in Table III. This table is similar to Table II except that it has a parent node. This parent node will build a tree using other strokes in the same column as the children nodes.

From the Table III, the column strokes 02-1, 102-1, 310-4, 3102-4, 30-4, 302-4 have only one character as the parent node without any children node. The parent strokes would be coded as the characters in the boxes.

TABLE II
Stroke Combinations for Circled Character

	0-1	02-1	10-1	102-1	20-1	202-1	30-1	302-1	3102-1	3202-1
30-1	a	a	a	a	a	a	a	a	a	a
30-2	d	d	d	d	d	d	d	d	d	d
30-3	g q	g q	g q	g q	g q	g q	g q	g q	g q	g q
302-1	a o	a o	a o	a o	a o	a o	a o	a o	a o	a o
302-2	d	d	d	d	d	d	d	d	d	d
302-3	q	q	q	q	q	q	q	q	q	q
310-1	o	o	o	o	o	o	o	o	o	o
3102-1	o	o	o	o	o	c	o	o	o	o
312-1	o	o	o	o	o	o	o	o	o	o
32-1	a o	a o	a o	a o	a o	a o	a o	a o	a o	a o
32-2	d	d	d	d	d	d	d	d	d	d

TABLE III
Stroke Combinations for First-Fixed Characters

	02-1	102-1	20-1	202-1	310-2	310-4	3102-2	3102-4	30-2	30-4	302-2	302-4
parent node	c	c			ℓb^2	f	ℓb^2	f	ℓb^2	f	ℓb^2	f
30-1			w^1	w^1	h				h			
302-1			$v w^1$	$v w^1$	h k				h k			
32-1			$v w^1$	$v w^1$	h				h			
3202-1					h k				h k			

¹Character w is not completed to realize it and requires another step to recognize completely.

²Character b requires the correcting algorithm.

The columns 20-1, 202-1 have only children nodes without having any parent node like the columns in Table II. The local routine VCHECK will be called to discriminate the characters 'v' and 'w' which were in the same box.

The columns 3102-2, 302-2 have two characters in the parent node without having any children node. The b-correcting routine would be called to check and recognize character b and correct the neighboring informations which might have been distorted by the character b.

For the column strokes 310-2, 30-2, a tree is built and analyzed as in the following:

if $SD(I) = 310-2$ or $30-2$

and $SD(I)$ is the last stroke of segment of string

then $CHA(I) = 'l'$

and $SD(I+1) = 30-1$ or $32-1$

check '*hk*' calling HKVSL

if true then $CHA(I) = 'h'$

if false then check and correct calling BCOREC

and $SD(I+1) = 3202-1$ or $302-1$

check '*hk*' calling HKVSL

if true then compare '*h*' and '*k*' calling HVSK

if false then check and correct calling BCOREC

else check and correct calling BCOREC

The osculated characters and last-fixed characters do not have many classes to build any tree analyzers and the stroke combinations which were explained in the semantic explanation section can be implemented without any other considerations for the characters of the classes.

The relative characters have more complicated relations among the strokes for the character composition and the stroke combinations for the relative characters are defined in Table IV.

This table is similar to Table III except that the row strokes are the earlier stroke in the tree organization, and the column strokes are the next stroke to the row strokes. Each stroke can be any side of the character depending on the neighboring conditions.

From Table IV, the stroke 30-1 has a parent node which can be recognized as the character 's' or can be first side of any character having the children node as the next side of the character depending on the neighboring conditions.

The following list shows the summarized program for the priority in stroke combination and evaluation of the characteristics of each stroke for the tree of the stroke 30-1 as shown in Figure 23.

if SD(I) = 30-1

and SD(I+1) = 30-1

check 's' calling SCHEK

if true then CHA(I) = 's'

if false then check 'w' at children nodes calling VCHEK

if true then CHA(I) = 'w'

if false then CHA(I) = 'u'

and SD(I+1) = 30-3

check 'y' calling SCHEK

if true then CHA(I) = 's'

if false then CHA(I) = 'y'

and SD(I+1) = 302-1 or 32-1

check 's' calling SCHEK

TABLE IV

Stroke Combinations for Relative Character

	30-1	30-3	302-1	3102-1	32-1	320-1	3202-1
	s		c e r s	c e	r	s r	r s
30-1	u w	p	u w		u w		
30-3	y		y			y	y
302-1	u v w	p	u v q		u v w		v
3102-1							
32-1	u v w		u v w		u v w		v
320-1		p					
3202-1		p					

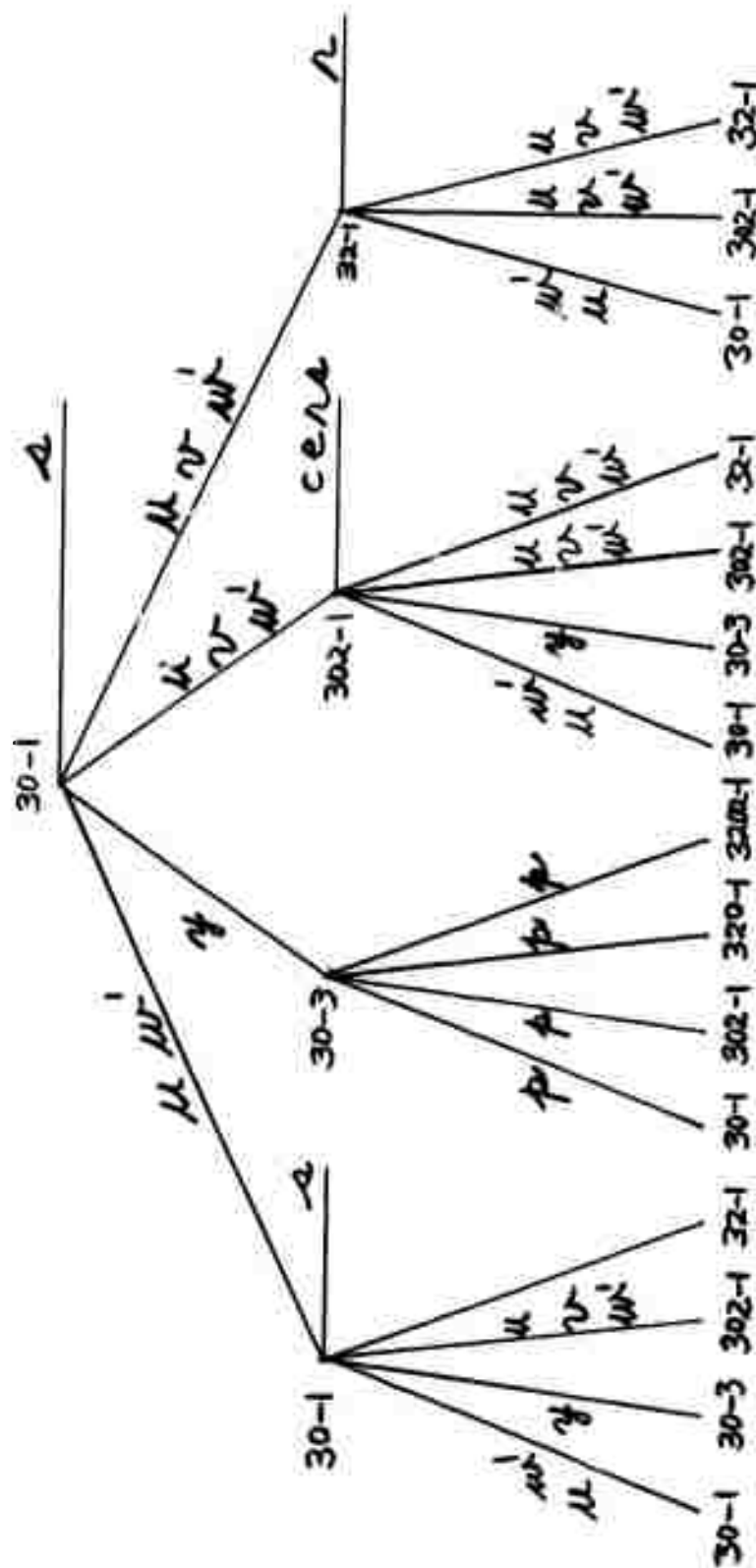


Figure 23. Tree organization for the relative stroke 30-1

```

if true then CHA(I) = 's'
if false then check 'v' calling VCHEK
    if true then CHA(I) = 'v'
    if false then check 'v' at children nodes calling VCHEK
        if true then CHA(I) = 'w'
        if false then CHA(I) = 'u'

```

The stroke 302-1 has a parent node which could be coded into several different characters depending on the characteristics of the parent node and four children nodes which could be coded as second stroke of the character with the parent stroke as the first part of the character depending on the decision for the parent node during the beginning execution of the tree shown in Figure 24.

The following is the summarized program for the stroke 302-1;
if SD(I) = 302-1

and SD(I+1) = 30-1

check 's' calling SCHEK

if true then CHA(I) = 's'

if false then check 'r' calling CCHEK

if true then compare 'r' to 'e' calling CVSE

if false then check 'r' calling RCHEK

if true then CHA(I) = 'r'

if false then check 'v' at children node

if true then CHA(I) = 'w'

if false then CHA(I) = 'u'

and SD(I+1) = 30-3

check 's' calling SCHEK

if true then CHA(I) = 's'

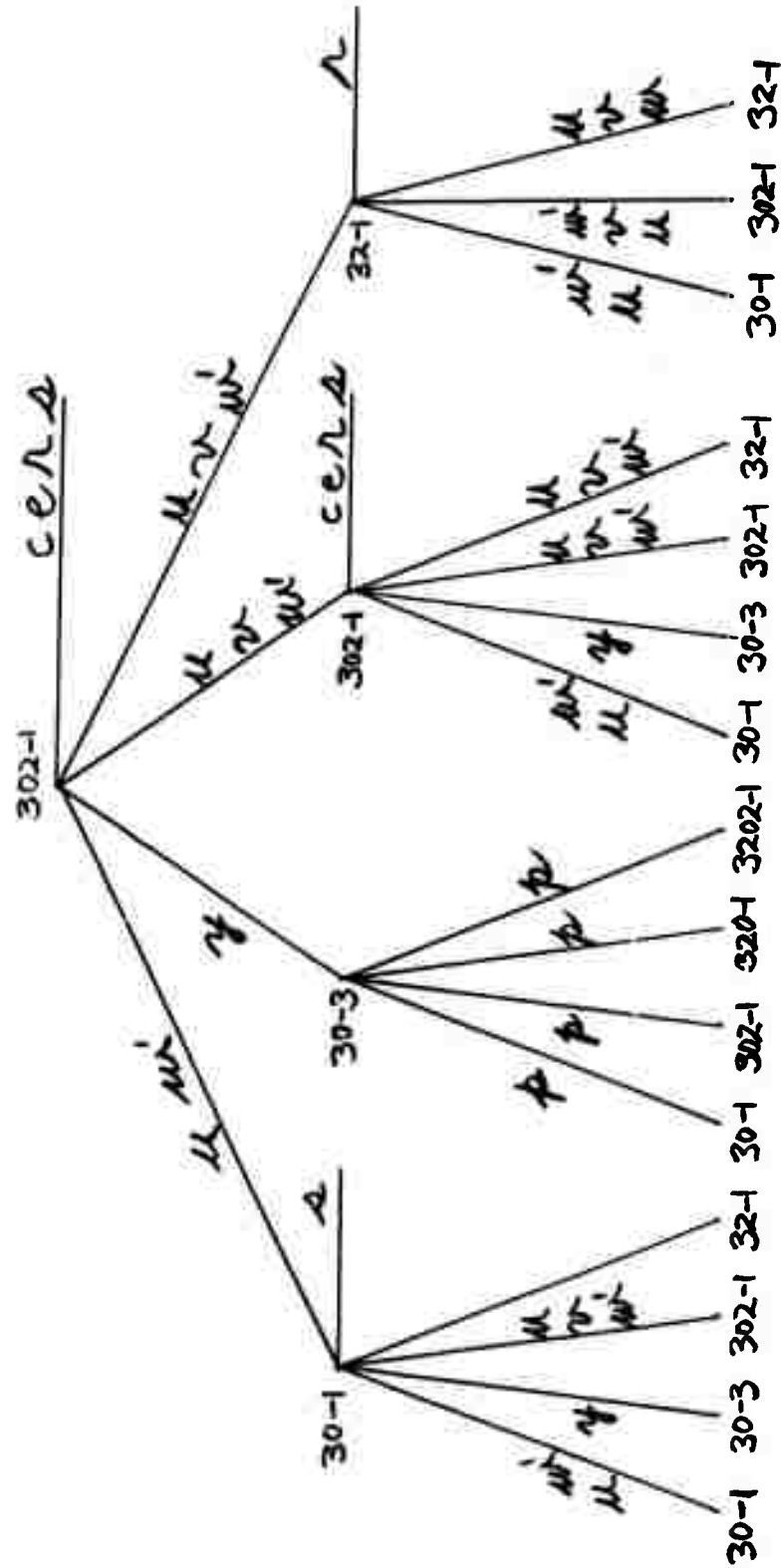


Figure 24. Tree organization for the relative stroke 302-1

```

if false then check 'ㄨ' calling CCHEK
  if true then compare 'ㄨ' to 'ㄝ' calling CVSE
  if false then check 'ㄆ' at children node calling PCHEK
    if true then CHA(I) = 'r'
    if false then CHA(I) = 'y'
and SD(I+1) = 302-1 or 32-1
  check 'ㄟ' calling SCHEK
  if true then CHA(I) = 's'
  if false then check 'ㄴ' calling VCHEK
    if true then CHA(I) = 'v'
    if false then check 'ㄹ' calling CCHEK
      if true then compare 'ㄹ' to 'ㄝ' calling CVSE
      if false then check 'ㄺ' calling RCHEK
        if true then CHA(I) = 'r'
        if false then check 'ㄽ' at children nodes
          if true then CHA(I) = 'w'
          if false then CHA(I) = 'u'

```

The stroke 32-1 has a parent node which could be coded as the character 'ㄹ' or as the first stroke of a character having the children nodes as the second strokes of the characters as shown in Figure 25.

The tree would be analyzed by processing the parent node first and then the children node will be processed along with the parent node. The following is the summarized program for the tree of the relative stroke 32-1.

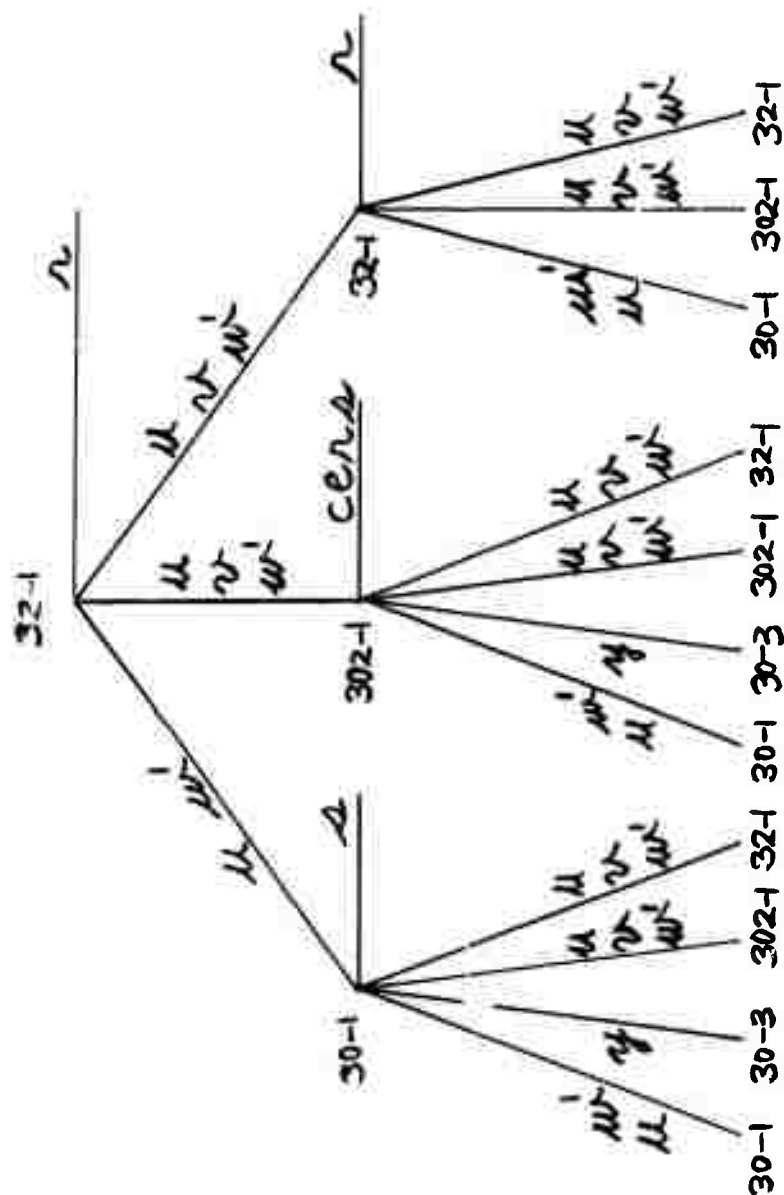


Figure 25. Tree organization for the relative stroke 32-1

```

if SD(I) = 32-1
  and SD(I+1) = 30-1
    check 'ㄣ' calling RCHEK
    if true then CHA(I) = 'r'
    if false then check 'ㄤ' at children node
      if true then CHA(I) = 'w'
      if false then CHA(I) = 'u'
  and SD(I+1) = 302-1 or 32-1
    check 'ㄥ' calling VCHEK
    if true then CHA(I) = 'v'
    if false then check 'ㄣ' calling RCHEK
      if true then CHA(I) = 'r'
      if false then check 'ㄤ' at children nodes
        if true then CHA(I) = 'w'
        if false then CHA(I) = 'u'

```

The stroke 3202-1 has a parent node which could be coded into several different characters depending on the characteristics of the node and three children nodes which could be coded as the second stroke of the character having the parent node as the first stroke of the character.

The tree is shown in Figure 26 and the analyzer is listed as following:

```

if SD(I) = 3202-1
  and SD(I+1) = 30-3
    check 'ㄣ' calling SCHEK
    if true then CHA(I) = 's'

```

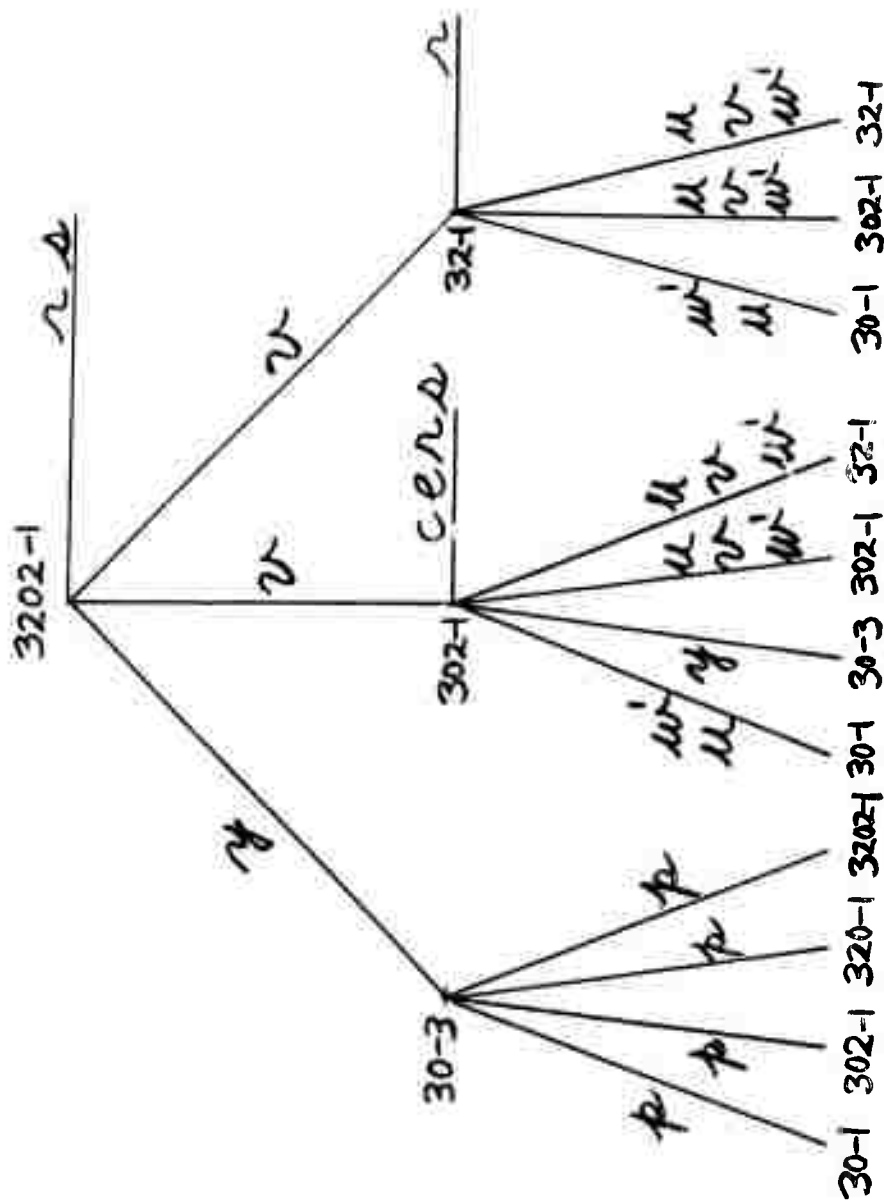


Figure 26. Tree organization for the relative stroke 3202-1

```

if false then check 'p' at children nodes calling PCHEK
  if true then CHA(I) = 'r'
  if false then CHA(I) = 'y'

```

```

and SD(I+1) = 302-1 or 32-1

```

```

check 's' calling SCHEK

```

```

if true then CHA(I) = 's'

```

```

if false then check 'v' calling VCHEK

```

```

  if true then CHA(I) = 'v'

```

```

  if false then CHA(I) = 'r'

```

The stroke 320-1 has a parent node and two children nodes as shown in Table IV. The following is the list for the summarized program for the tree of stroke 320-1.

```

if SD(I) = 320-1

```

```

  and SD(I+1) = 30-3

```

```

    check 's' calling SCHEK

```

```

    if true then CHA(I) = 's'

```

```

    if false then check 'p' at children node calling PCHEK

```

```

      if true then CHA(I) = 'r'

```

```

      if false then CHA(I) = 'y'

```

```

  and SD(I+1) = 32-1

```

```

    check 'v' calling VCHEK

```

```

    if true then CHA(I) = 'v'

```

```

    if false then check 's' calling SCHEK

```

```

      if true then CHA(I) = 's'

```

```

      if false then CHA(I) = 'r'

```

The parent stroke 30-3 does not have the parent node and the direct combination is suggested to combine the strokes for character composition.

The parent stroke 3102-1 does not have any children node and the only routine CVSE is requested to identify the difference of the characteristics between the characters.

Those local routines which were used in this section will be presented in the next section.

5.3 Local Discriminating Routines

The hierarchical organization of the major characteristics of the strokes classified the strokes of the writing into several levels of priority for the character compositions. The stroke group of each level is classified into further discriminated groups by the order in character combination.

To get the final decision for the recognition of the characters from the stroke combinations which already are classified by the characteristics of the strokes and orders in combination of strokes, some further discriminating routines must be applied for complete recognition of the groups which still have more than one member.

These routines do not have complicated functions to complete the recognition. Each routine has only independent operations from other routines, and the operations are simple and fairly short because only specified parts of the characteristics are checked by the routine.

Each routine might take a simple logical evaluation to compute the relative characteristics comparing the characteristics of the members in that class.

The evaluations of the characteristics in each routine are listed briefly as following.

1. CCHEK

In Figure 27

if XD greater than NY/5 then check = true
else check = false

2. CVSE

In Figure 28

check the curve 1-3 compare to dashed line 1-3
if the curve is lower than the dashed line
and the maximum YD greater than NY/8 then
CHA(I) = 'e' else CHA(I) = 'c'

3. GVSQ

In Figure 29

check the locations of the curve 1-2 regard to curve 2-3
if curve 2-3 is left to curve 1-2 then CHA(I) = 'g' else
CHA(I) = 'q'

4. HKVSL

In Figure 30

check the XD between the point 1 and point 2
count two counters when

① XD is greater than NY/5

② XD is less than NY/5

if counter ② is less than counter ① then check = true
else check = false

The evaluations of the characteristics in each routine are listed briefly as following.

1. CCHEK

In Figure 27

if XD greater than NY/5 then check = true

else check = false

2. CVSE

In Figure 28

check the curve 1-3 compare to dashed line 1-3

if the curve is lower than the dashed line

and the maximum YD greater than NY/8 then

CHA(I) = 'e' else CHA(I) = 'c'

3. GVSQ

In Figure 29

check the locations of the curve 1-2 regard to curve 2-3

if curve 2-3 is left to curve 1-2 then CHA(I) = 'g' else

CHA(I) = 'q'

4. HKVSL

In Figure 30

check the XD between the point 1 and point 2

count two counters when

① XD is greater than NY/5

② XD is less than NY/5

if counter ② is less than counter ① then check = true

else check = false

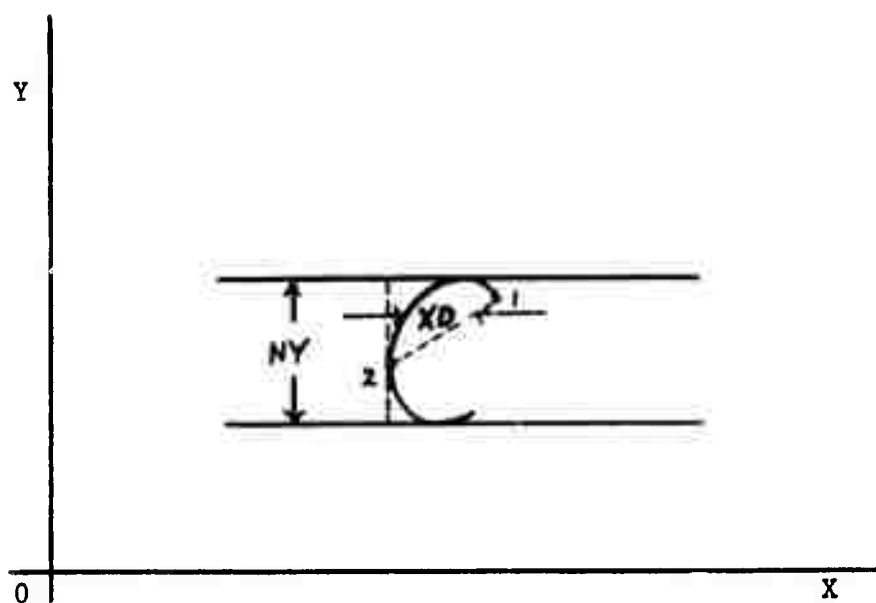


Figure 27. Characteristic evaluation for CCHEK

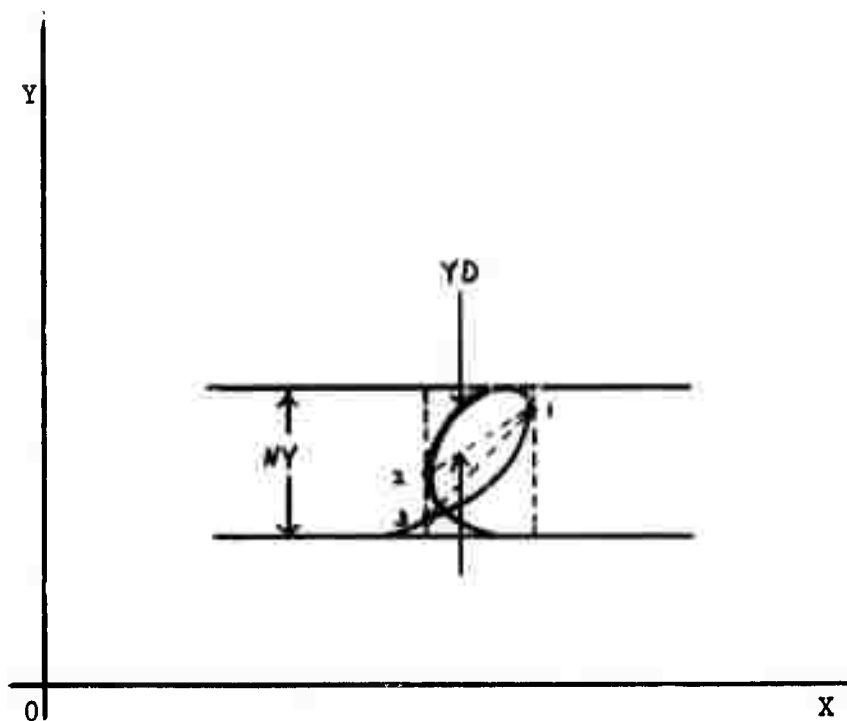


Figure 28. Characteristic evaluation for CVSE

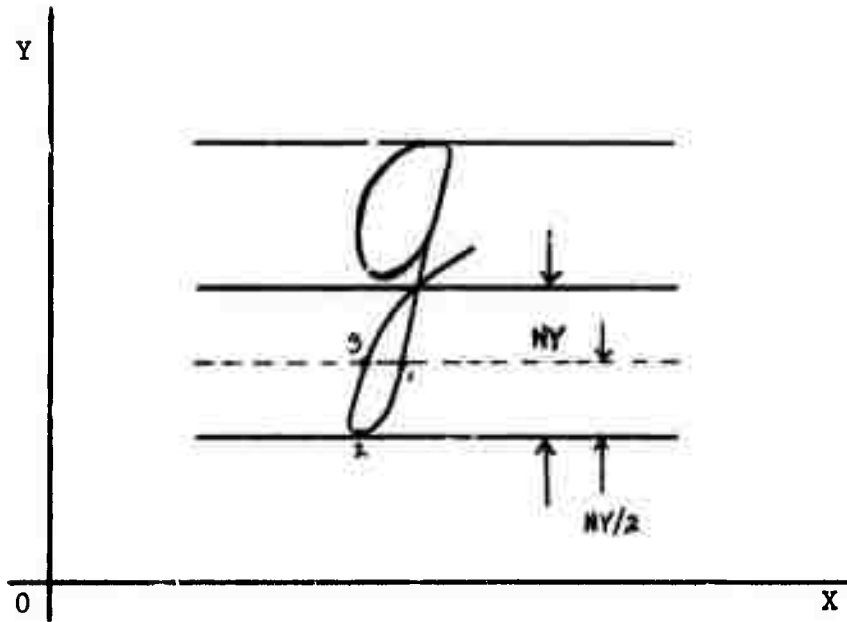


Figure 29. Characteristic evaluation for GVSQ

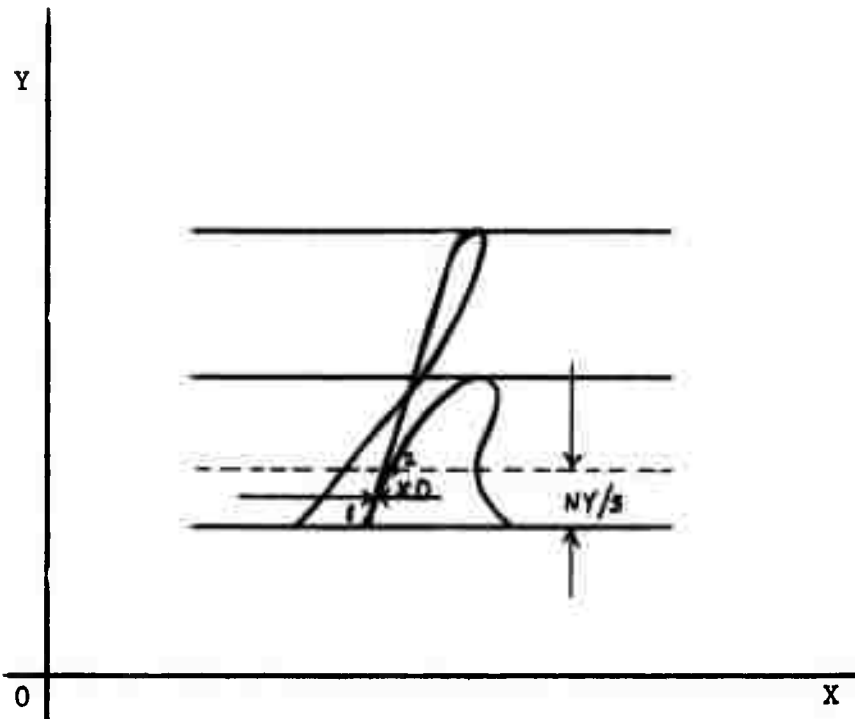


Figure 30. Characteristic evaluation for HKVSL

5. HVSK

In Figure 31

if $(X(1)-X(2))/(X(4)-X(3))$ greater than 1.5 and
the tangent of point 4 toward point 1 is less
than 1.0 then $CHA(I) = 'k'$ else $CHA(I) = 'h'$

6. PCHEK

In Figure 32

if the curve 3-4 is right side to dashed line 3-4
and the curve 2-3 is left to the dashed line 2-3
and $X(4)-X(1)$ is less than $NYD/6$ then check = true
else check = false

7. RCHEK

In Figure 33

if XD greater than $NYD/5$ then check = true
else check = false

8. SCHEK

In Figure 34

if the slope of dashed line 1-2 is less than 10 and $X(1)$ is
greater than $X(2)$ then check = true else check = false

9. VCHEK

In Figure 35

if $X(1)-X(3)$ greater than $NY/4$ or
 $Y(2)$ greater than $Y(3)-NY/2$ then
check = true else check = false

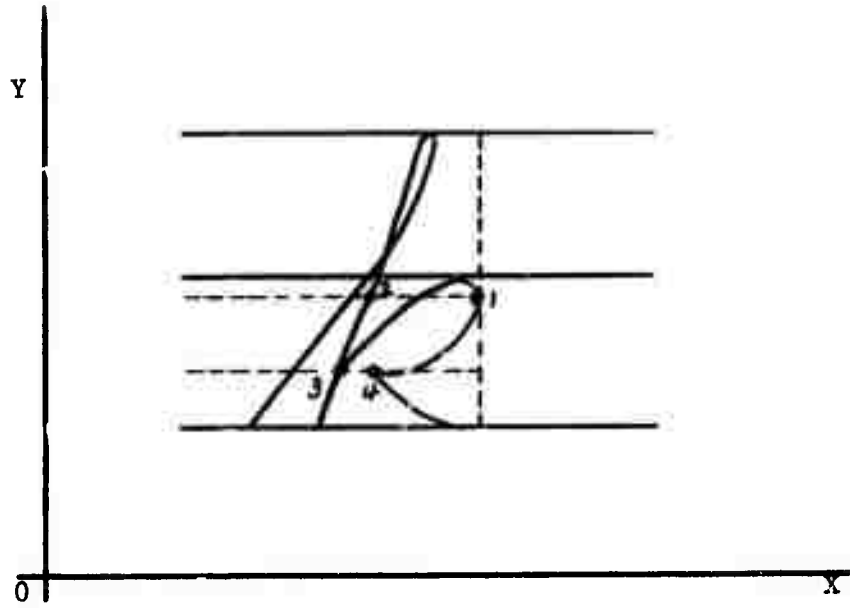


Figure 31. Characteristics evaluation for HVSK

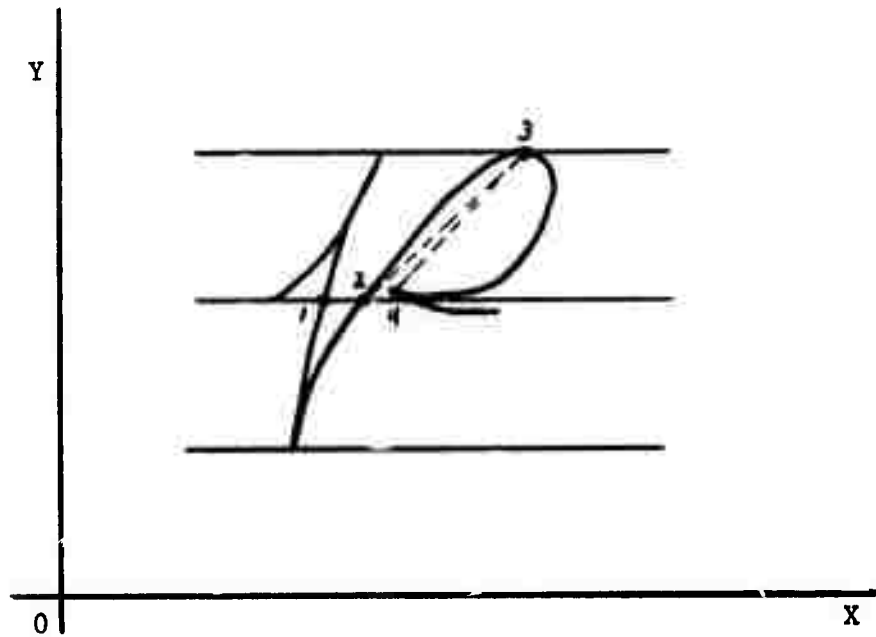


Figure 32. Characteristics evaluation for PCHEK

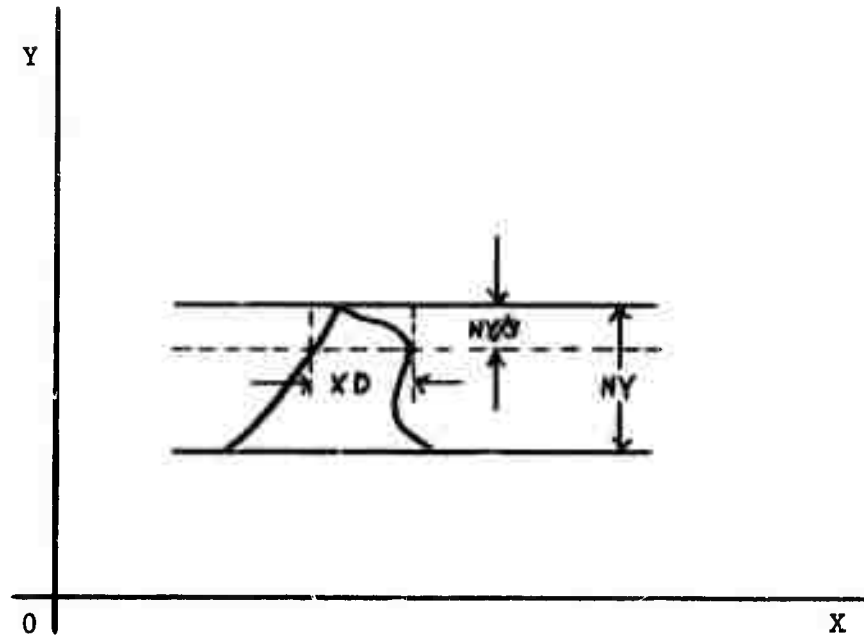


Figure 33. Characteristic evaluation for RCHEK

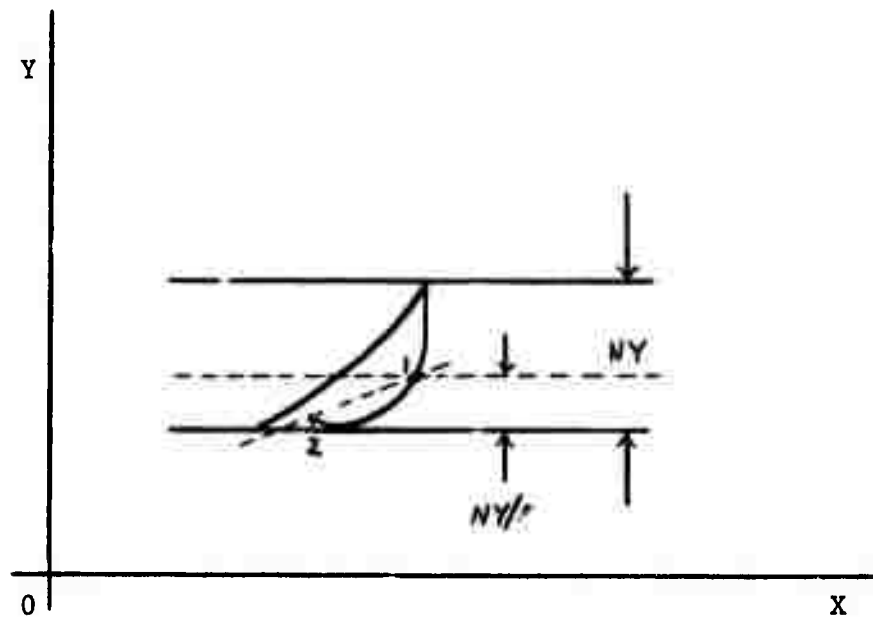


Figure 34. Characteristic evaluation for SCHEK

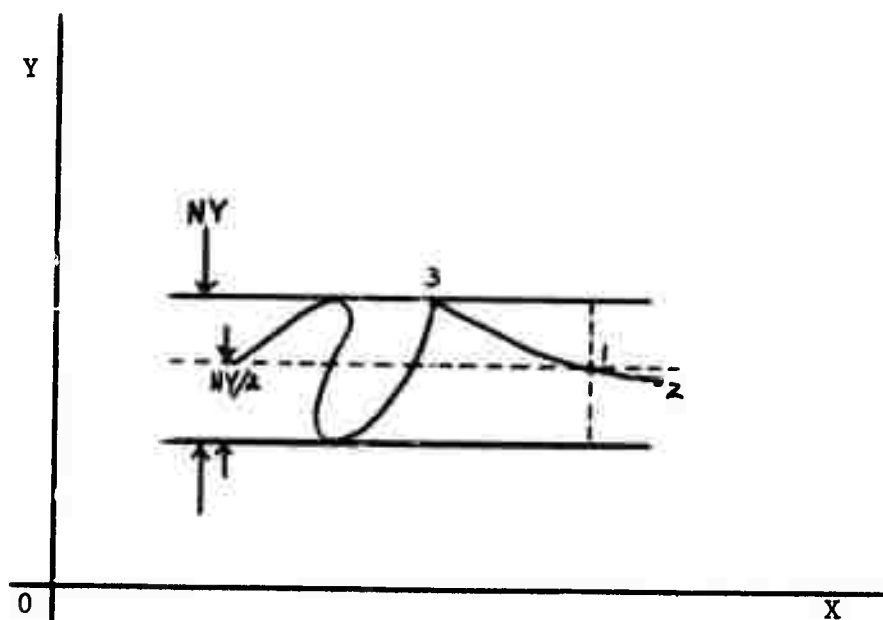


Figure 35. Characteristics evaluation for VCHEK

CHAPTER VI

SELF-CORRECTION BY ITERATION

6.1 Self-Correction by Iteration

The syntax organization for this system has dealt with the error corrections of cutting algorithms and the error corrections in stroke combination for character compositions.

The major types of error in this experiment are classified as the following cases; the relative position coding error, the writing cutting error and the decision error in relative discriminating routines.

The regular writing would match the bottom of each character to the base line of the writing except the strokes of position code 3 and position code 4.

The iterative algorithm for this position coding routine cover the wider range of writings which do not match the base line of the writing.

The iterative coding routine for positional code is designed as following:

Pick the lowest point out of a group of the highest points from each stroke, and define it as the upper base line. Pick the highest point out of a group of the lowest points from each stroke and define it as the lower base line. Define the allowable margin as half of the difference between these two base lines.

Check the highest point of each stroke to determine whether the point belongs to the upper region which is bounded by the upper base

line as the bottom side line and the upper base line position plus allowable range margin as the upper side line of the region. Evaluate the average of the highest point using the highest points of the strokes which belong to the upper region. Define this average as the new upper base line.

Check the lowest point of each stroke to determine whether the point belongs to the lower region which is bounded by the lower base line as the top side line and the lower base line position minus the allowable range margin as the lower side line of the region. Evaluate the average of the lowest point using the lowest points of the strokes which belong to the lower region. Define this average as the new lower base line.

This routine is repeated a number of times for the iteration and then the position coding is determined through the rules explained in earlier chapters.

The writing cutting algorithm works well just as it is defined in earlier chapters, except for the characters 'b', 'o', 'v', 'w', because of the uncareful writings.

The corrections for writing cutting algorithms were implemented in syntax organization for the characters 'b' and 'o' and the hierarchy was reorganized from the un-correcting syntax and the analyzer was adjusted for the reorganization of the syntax.

The error from this cutting algorithm generates an improper combination of the strokes for the character compositions. The feedback loop is designed to cover this early level of hierarchy for the iteration of this algorithm.

The correcting routines for the characters 'o' and 'b' are quite similar except the local organization of the priority for the comparative discriminating routines.

The combination of the strokes is checked by the characteristics of the character and the correction of the distorted information is followed for the neighboring strokes.

The major distorted information of the neighboring stroke is the stroke coding error. The stroke can have wrong stroke parts which belong to the previous character.

The character correcting routine will regenerate the lost stroke after checking the character characteristics, and other information which was lost because of the error in writing cutting algorithm for previous stroke will be restored by the correcting routine.

The decision error in relative discriminating routines comes from the threshold value of the characteristic evaluation.

Since each routine is for the comparative evaluation of the characteristics of the members of the classified group, there are always many relative characteristics to give solid information for the decision.

The threshold value can be fixed by making some learning experiment, and since the routine is relatively referenced only for the member of the class which the routine is related, the threshold value can be locally relative without considering the entire system for the optimizations.

6.2 Stability of the Self-Correcting Algorithm

Feedback theory has been studied in many engineering aspects to control the stable output of the analog systems.

The analog computer is another system which applies the feedback theory to simulate a mathematical model.

A digital system can be substituted for the analog system or part of the analog system for control purpose or computing purpose. Digital Differential Analyzer can be a typical example for the digital feedback system.

The stability of the system has been discussed in many areas for the analog system and sampled data system for the hardware organization.

The feedback theory for the stability of the software iterating system has been implemented in this experiment during the organization of the syntax and the criteria of the stability for iterative software system is discussed in this section.

In Figure 36, the block diagram for the recognition scheme was shown and the feedback from error correcting block to error checking block is designed for this experiment.

After the procedures of figure extraction, the program will start to recognize the stroke combinations as the character composition by checking the errors in character recognition, and it would correct the rest of the information if any error is found by the syntax analyzer.

The corrected information requires another iteration to check any possible error and take new stroke combination for the recognition.

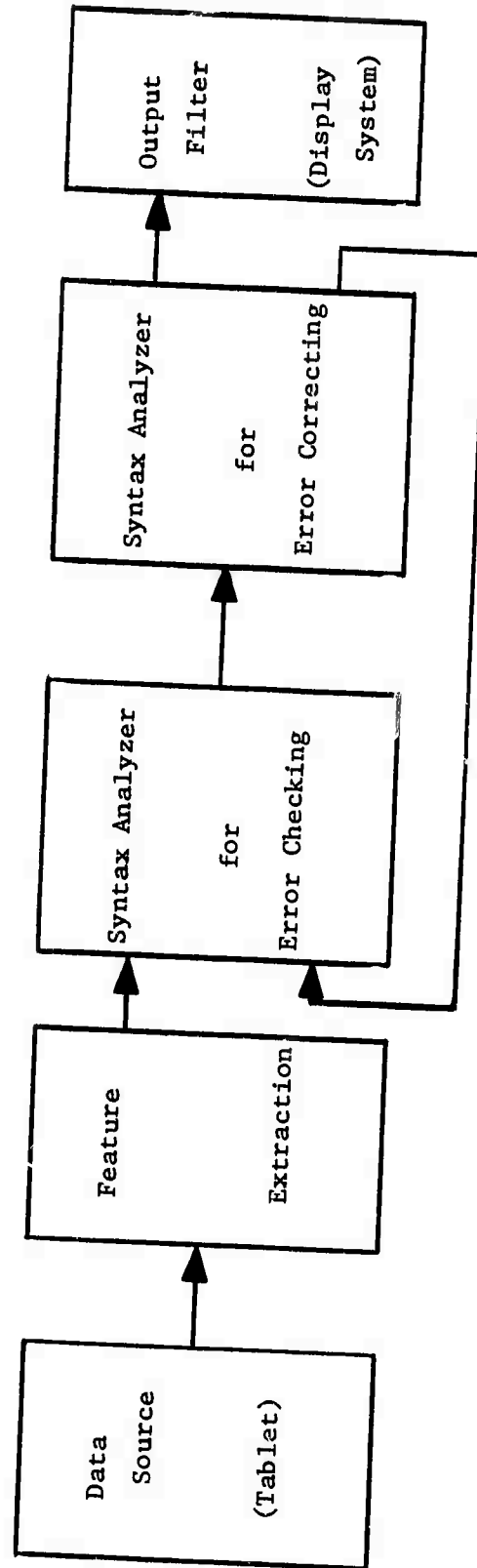


Figure 36. Block diagram for error correcting system

The number of iterations is decided by the error correcting algorithm.

To design a general control system, the stability of the hardware system may be checked by the Routh Criteria (9).

For the self-corrective system by iteration, the stability will be designed by the syntax organization, and the priority of the characteristics has to be studied for each stroke, and the level of error has to be qualified by the types of errors. The hierarchical organization for such work is always complicated, but a good solution gives always a reliable recognition.

CHAPTER VII

CONCLUSIONS

7.1 Accuracy of Recognition

The syntax-directed algorithm for handwriting recognition was constructed using self-correcting loops to check for writing errors and to correct the errors made by normal writers. A simple experiment without feedback loops for handwriting recognition was constructed during the early period of laboratory work and the recognition was highly reliable for writers who had short training using the standard styles.

For more reliable recognition for uncareful writers, the feedback loops were implemented to eliminate the major failures in recognition by the first system. The recognition rate for the system with feedback loops was quite reliable and the types of error which were the major errors in the early system were eliminated, and the variations in allowable writings for normal users were much broader.

To improve the recognition for all types of writing, a number of feedback loops should be implemented. Such increased capability to recognize all types of writing may be of limited value because of increased cost of operation.

The feedback loops were implemented in such a way as to widen the range of users and also keep the efficiency of the program high. The present degree of feedback iteration is quite reliable for the interactive communication experiment for recognition of cursive writing.

7.2 Further Work

For further efficiency and broader applications, the probable future work in organizing this system can be discussed as follows.

The data tablet is required to have increased resolution for writing recognition. The present 3% per inch resolution limits the size of characters and the larger characters are inconvenient to write neatly for the writers. A higher resolution tablet is important for reliable and accurate recording of inputs.

Since the system program for the sampling procedure in the PDP-8 is linked with other system programs in the graphics system, the speed of the data sampling is forced to be slower than in a single purpose graphic system. The speed of sampling procedure is another important factor for the higher resolution system, which usually varies depending on the size of program instruction.

A hierarchical organization with feedback loops for the positional code may be suggested for the relative normal size coding algorithm to correct in the positional code which usually arise from unknown noises. It is suggested to design the feedback loops independent of the loops of the recognizing procedures to avoid the complexity of the system.

Also more loops can be added to the recognizing routines without interfering with the existing loops to check the minor errors.

Simple learning procedures may be used for some particular entries of the syntactic organization which do not receive the reliable threshold value for the decision in local discriminating routines.

This syntactic organization can employ such conventional ideas in pattern recognition (4) in any part and as any application.

BIBLIOGRAPHY

1. Backus, J.W., "Revised Report on the Algorithm Language Algol 60," Assn. Computing Machinery Communications, Vol. 6, No. 1, pp. 1-17, Jan. 1963.
2. Eden, M., "Handwriting and Pattern Recognition," IRE Trans. on Information Theory, Vol. IT-8, pp. 160-166, 1962.
3. Groner, G.F., "Real-Time Recognition of Hand Printed Text," Am. Federation of Information Processing Soc. (AFIPS) - Fall Joint Computer Conference Proc., pp. 591-601, 1966.
4. Highleyman, W.H., "Linear Decision Functions, with Application to Pattern Recognition," IRE Proc., Vol. 50, pp. 1501-1514, 1962.
5. Mermelstein, P., and M. Eyden, "A System for Automatic Recognition of Handwritten Words," Am. Federation of Information Processing Soc. (AFIPS) - Fall Joint Computer Conference Proc., Vol. 26, pp. 333-342, 1964.
6. Nagy, G., and G.L. Shelton, Jr., "Self-Corrective Character Recognition System," IEEE Trans. on Information Theory, Vol. IT-12, No. 2, pp. 215-222, 1966.
7. Noble's Better Handwriting for Everyone, Noble and Noble Publishers Inc., New York, N.Y., 1962.
8. Papoulis, A., Probability, Random Variables and Stochastic Processes, McGraw-Hill Book Company Inc., New York, N.Y., 1965.
9. Truxal, J.C., Automatic Feedback Control System Synthesis, McGraw-Hill Book Company Inc., New York, N.Y., 1955.
10. UNIVAC 1108 Multi-Processing System Exec II Programming Reference Manual, UP-4058, UNIVAC Division of Sperry Rand Corp., 1966.

APPENDIX

9.1 Listing of Program

```

QA RUN PATTERN,496606,2,90   Y.  T.  KIM   SYNTAX
@ ASG A=$DEVON
@ ASG G=$$SB2$
@ XQT CUR
  IN A
BEGIN
EXTERNAL FORTRAN PROCEDURE BLANK,CHAR,CHRINT,GETCHR,GETTAB,
  IDI,IDLE,INTEN,LINE,LNTYPE,RELOAD,SETBUF,SETMAX,SETMIN,
  SETSIZ,SETSWP,SNOFLE,SWAP,TABABL,TABINT,TABTOL$
INTEGER ARRAY DFILE(1:3000),X(0:3000),Y(0:3000),Z(1:3000)$
INTEGER A,B,DUMMY,J,K,K1,K2,KMAX,L,LMAX,N,SW$
STRING CHA(30),CX(1),PCHA(2),SCHA(8)$
LOCAL LABEL CHR,ENIT,ID,MAIN,OT,XY$
  RELOAD$
  IDI(DFILE,DUMMY)$
  INTEN(2)$
  SETMAX(4)$
  SETMIN$
  TABABL$
  SETSWP(4,4000)$
  TABINT(1,XY)$
  CHRINT(1,CHR)$
  TABTOL(3)$
  SETSIZ(1)$
OT: J=0$ SETBUF(5,DUMMY+3)$
  SNOFLE(DFILE,DUMMY)$ IDI(DFILE,DUMMY)$
ID: IDLE$ SWAP$
  GO TO ID$
XY: GETTAB(A,R,SW)$
  IF SW NEQ 0 THEN BEGIN J=J+1$ X(J)=X(J-1)$
    Y(J)=Y(J-1)$ Z(J)=1$ ENDS
  IF SW EQL 1 THEN LNTYPE(0) ELSE LNTYPE(3)$
  IF SW EQL -1 THEN GO TO MAIN$
LINE(DFILE,A,R,DUMMY)$
  J=J+1$ X(J)=A$ Y(J)=B$ Z(J)=2$
  GO TO XY$
CHR: GETCHR(CX(1))$
  IF CX(1) EQL 'X' THEN GO TO ENITS
  IDI(DFILE,DUMMY)$
  GO TO OT$

```

```

MAIN: LMAX=J$
IF J LSS 50 THEN GO TO OTS
COMMENT DATA FILTER ***** $
FOR J=(3,1,LMAX-2) DO
IF Z(J-1) EGL 2 AND Z(J) EGL 2 AND Z(J+1) EGL 2 THEN BEGIN
  IF X(J) LSS X(J-1) AND X(J) LSS X(J+1)
    AND X(J+2) LSS X(J+1)
  OR X(J) GTR X(J-1) AND X(J) GTR X(J+1)
    AND X(J+2) GTR X(J+1)
  THEN X(J)=X(J-1)$
  IF Y(J) LSS Y(J-1) AND Y(J) LSS Y(J+1)
    AND Y(J+2) LSS Y(J+1)
  OR Y(J) GTR Y(J-1) AND Y(J) GTR Y(J+1)
    AND Y(J+2) GTR Y(J+1)
  THEN Y(J)=Y(J-1)$
END$
N=LMAX$ L=1$ LMAX=1$
FOR J=(2,1,N) DO BEGIN
  IF X(J) EGL X(L) AND Y(J) EGL Y(L) AND Z(J) EGL Z(L)
    OR Z(J) EGL 1 AND Z(L) EGL 1 THEN GO TO LL$
  L=L+1$ LMAX=LMAX+1$
  X(L)=X(J)$ Y(L)=Y(J)$ Z(L)=Z(J)$
LL: END$
KMAX=LMAX$
IF Z(KMAX) EGL 2 THEN BEGIN X(KMAX+1)=X(KMAX)$
  Y(KMAX+1)=Y(KMAX)$ Z(KMAX+1)=1$ KMAX=KMAX+1$ END$
COMMENT BREAK WRITING ***** $
BEGIN
INTEGER ARRAY CHEK(0:20),DX(0:2000),DY(0:2000),JMAX(0:20),
  LMAX(1:20),LXD(1:20,1:7),LXI(1:20,1:7),LYD(1:20,1:7),
  MMAX(1:20,0:7),OMAX(1:20),PD(1:20),PMAX(1:20,0:7),
  SCOD(1:20,1:7),SD(1:20),YMAX(1:20),YMIN(1:20)$
REAL ARRAY SSCOD(1:20)$
INTEGER DEC,DXDEC,DXINC,DYDEC,DYINC,I,IMAX,INC,J1,J2,LX,LY,
  M,MX,N1,NEDE,NEIN,PODE,POIN,YB,YDEC,YDEIN,YINC,YINDE,YT$
BOOLEAN ARRAY XCOD(1:20,1:7),YCOD(1:20,1:7)$
LOCAL LABEL LL3$
FOR K=(2,1,KMAX) DO BEGIN
  DX(K)=X(K)-X(K-1)$ DY(K)=Y(K)-Y(K-1)$ END$
  I=1$ IMAX=1$ JMAX(0)=0$ N1=4$
FOR K=(N1+4,1,KMAX-1) DO BEGIN
  IF Z(K) NEQ 2 THEN BEGIN
  INTEGER YBIG,YSMA,K3,K4$
  JMAX(I)=K$ MMAX(I,1)=K$
  IF K EGL KMAX THEN GO TO LL3$
  I=I+1$ IMAX=IMAX+1$
  YBIG=0$ YSMA=0$
FOR K3=(K+1,1,KMAX) DO BEGIN
  IF Y(K3) GTR Y(K3-1) THEN YBIG=YBIG+1$
  IF Y(K3) LSS Y(K3-1) THEN YSMA=YSMA+1$

```

```

IF YB16 GEQ 8 OR YSMA GEQ 8 THEN GO TO LL3
IF Z(K1) NEQ 2 THEN DLOTH K4=K5$ K3=KMAX+1$ ENDS$ ENDS
IF YB16 LSS 8 AND YSMA LSS 8 THEN
    BEGIN SL(I)=7$ LMAX(I)=1$ K=K4-1$ ENDS
GO TO LL3$ ENDS
IF DY(K-2) LEQ 0 AND DY(K-1) LEQ 0 AND DY(K) GTR 0 THEN BEGIN
    YINC=0$ YDEC=0$ YINDE=0$ YIIN=0$
    FOR K2=(K-1,-1,JMAX(I-1)+2) DO BEGIN
        IF DY(K2) LSS 0 THEN YDEC=YDEC+1$
        IF DY(K2) GTR 0 THEN YDEIN=YDEIN+1$
    IF YDEC GEQ 2 OR YDEIN GEQ 2 OR Z(K2) NEQ 2 THEN
        K2=JMAX(I-1)$ ENDS
    FOR K1=(K,1,KMAX-1) DO BEGIN
        IF DY(K1) LSS 0 THEN YINDE=YINDE+1$
        IF DY(K1) GTR 0 THEN YINC=YINC+1$
        IF YINC GEQ 2 OR YINDE GEQ 2 OR Z(K1) NEQ 2 OR
            Z(K1+1) NEQ 2 OR Z(K1+2) NEQ 2 THEN K1=KMAX$ ENDS
    IF YDEC GEQ 2 AND YINC GEQ 2 THEN BEGIN I=I+1$ IMAX=IMAX+1$
        JMAX(I)=K$
        IF JMAX(I-1)-JMAX(I-2) LSS 4 AND SD(I-1) NEQ 7 THEN BEGIN
            I=I-1$ IMAX=IMAX-1$ ENDS$ ENDS$ ENDS
LL3: JMAX(I)=K$ MMAX(I,1)=K$
        ENDS
    IF JMAX(IMAX)-JMAX(IMAX-1) LSS 6 AND SD(IMAX) NEQ 7 THEN
        IMAX=IMAX-1$
    FOR I=(1,1,IMAX) DO BEGIN CHA(I)='+'$ CHEK(I)=2$
        PD(I)=0$ ENDS
    COMMENT BREAK CURVE *****$
        MMAX(1,0)=0$
    FOR I=(1,1,IMAX) DO BEGIN
        FOR L=(1,1,7) DO BEGIN
            LXI(I,L)=0$ LXD(I,L)=0$ LYD(I,L)=0$ ENDS
            IF I GTR 1 THEN MMAX(I,0)=MMAX(I-1,LMAX(I-1))$
            IF SD(I) EQL 7 THEN GO TO LL5$
            L=1$ LMAX(I)=1$ MMAX(I,1)=JMAX(I-1)+7$
        FOR J=(JMAX(I-1)+7,1,JMAX(I)) DO BEGIN
            IF DX(J-2) LEQ 0 AND DX(J-1) LEQ 0 AND DX(J) GTR 0 THEN BEGIN
                NEDE=0$ NEIN=0$ PODE=0$ POIN=0$
            FOR J1=(J,1,JMAX(I)) DO BEGIN
                IF DX(J1) LSS 0 THEN PODE=PODE+1$
                IF DX(J1) GTR 0 THEN POIN=POIN+1$
                IF POIN GEQ 2 OR PODE GEQ 2 THEN J1=JMAX(I)+1$ ENDS
            FOR J2=(J-1,-1,MMAX(I,L-1)+2) DO BEGIN
                IF DX(J2) LSS 0 THEN NEDE=NEDE+1$
                IF DX(J2) GTR 0 THEN NEIN=NEIN+1$
                IF NEDE GEQ 2 OR NEIN GEQ 2 THEN J2=MMAX(I,L-1)$ ENDS
            IF NEDE GEQ 2 AND POIN GEQ 2 THEN LXI(I,L+1)=1$ ENDS
        IF DX(J-2) GEQ 0 AND DX(J-1) GEQ 0 AND DX(J) LSS 0 THEN BEGIN
            NEDE=0$ NEIN=0$ PODE=0$ POIN=0$
        FOR J1=(J,1,JMAX(I)) DO BEGIN

```

```

      IF DX(J1) LSS 0 THEN PODE=PODE+1$
      IF DX(J1) GTR 0 THEN POIN=POIN+1$
      IF POIN GEQ 2 OR PODE GEQ 2 THEN J1=JMAX(I)+1$ ENDS
    FOR J2=(J-1,-1,MMAX(I,L-1)+2) DO BEGIN
      IF DX(J2) LSS 0 THEN NEDE=NEDE+1$
      IF DX(J2) GTR 0 THEN NEIN=NEIN+1$
      IF NEIN GEQ 2 OR NEDE GEQ 2 THEN J2=MMAX(I,L-1)$ ENDS
    IF NEIN GEQ 2 AND PODE GEQ 2 THEN LXD(I,L+1)=1$ ENDS
  IF DY(J-2) GEQ 0 AND DY(J-1) GEQ 0 AND DY(J) LSS 0 THEN BEGIN
    NEDE=0$ NEIN=0$ PODE=0$ POIN=0$
    FOR J1=(J,1,JMAX(I)) DO BEGIN
      IF DY(J1) LSS 0 THEN PODE=PODE+1$
      IF DY(J1) GTR 0 THEN POIN=POIN+1$
      IF POIN GEQ 2 OR PODE GEQ 2 THEN J1=JMAX(I)+1$ ENDS
    FOR J2=(J-1,-1,MMAX(I,L-1)+2) DO BEGIN
      IF DY(J2) LSS 0 THEN NEDE=NEDE+1$
      IF DY(J2) GTR 0 THEN NEIN=NEIN+1$
      IF NEIN GEQ 2 OR NEDE GEQ 2 THEN J2=MMAX(I,L-1)$ ENDS
    IF NEIN GEQ 2 AND PODE GEQ 2 THEN LYD(I,L+1)=1$ ENDS
  IF LXI(I,L+1) EQL 1 OR LXD(I,L+1) EQL 1 OR LYD(I,L+1) EQL 1
    THEN BEGIN L=L+1$ LMAX(I)=LMAX(I)+1$ MMAX(I,L)=J$ END
    ELSE MMAX(I,L)=J$ ENDS
LL5: ENDS
COMMENT NAME CURVE ***** $
FOR I=(1,1,IMAX) DO BEGIN IF SD(1) EQL 7 THEN GO TO LL4$
FOR L=(1,1,LMAX(I)) DO BEGIN
  DXINC=0$ DXDEC=0$ DYINC=0$ DYDEC=0$
  FOR K=(MMAX(I,L-1)+2,1,MMAX(I,L)) DO BEGIN
    IF DX(K) GTR 0 THEN DXINC=DXINC+1$
    IF DX(K) LSS 0 THEN DXDEC=DXDEC+1$
    IF K GTR MMAX(I,L-1)+25 THEN K=MMAX(I,L)+1$ ENDS
  FOR K=(MMAX(I,L-1)+2,1,MMAX(I,L)) DO BEGIN
    IF DY(K) GTR 0 THEN DYINC=DYINC+1$
    IF DY(K) LSS 0 THEN DYDEC=DYDEC+1$
    IF K GTR MMAX(I,L-1)+25 THEN K=MMAX(I,L)+1$ ENDS
  IF DXINC GTR DXDEC THEN XCOD(I,L)=TRUE ELSE
    XCOD(I,L)=FALSE$
  IF DYINC GTR DYDEC THEN YCOD(I,L)=TRUE ELSE
    YCOD(I,L)=FALSE$
  IF XCOD(I,L) AND YCOD(I,L) THEN SCOD(I,L)=3 ELSE
  IF XCOD(I,L) AND NOT YCOD(I,L) THEN SCOD(I,L)=2 ELSE
  IF NOT XCOD(I,L) AND YCOD(I,L) THEN SCOD(I,L)=1 ELSE
    SCOD(I,L)=6$ ENDS
FOR K=(1,1,3) DO BEGIN
  M=LMAX(I)-1$
  FOR L=(1,1,M) DO
    IF SCOD(I,L) EQL SCOD(I,L+1)
    OR SCOD(I,L) EQL 1 AND SCOD(I,L+1) EQL 3 THEN BEGIN
      FOR J=(L,1,M) DO BEGIN
        MMAX(I,J)=MMAX(I,J+1)$

```

```

        SCOD(I,J)=SCOD(I,J+1)$ ENDS$
        LMAX(I)=LMAX(I)-1$ M=LMAX(I)-1$ ENDS$
    ENDS$
    FOR L=(1,1,LMAX(I)) DO BEGIN
        IF SCOD(I,L) EQL 6 THEN SCHA(L)='6' ELSE
        IF SCOD(I,L) EQL 1 THEN SCHA(L)='1' ELSE
        IF SCOD(I,L) EQL 2 THEN SCHA(L)='2' ELSE
        IF SCOD(I,L) EQL 3 THEN SCHA(L)='3' ELSE
        IF SCOD(I,L) EQL 7 THEN SCHA(L)='7'$
        SSCOD(I)=SSCOD(I)+SCOD(I,L)*10*(-L)$ ENDS$
        SD(I)=SSCOD(I)*10*LMAX(I)$
    LL4: FOR J=(LMAX(I)+1,1,8) DO SCHA(J)=' '$
    SCHA(8)='Δ'$
    LNTYPE(0)$
        K1=50$ K2=500-I*30$
    LINE(DFILE,K1,K2,DUMMY)$
    CHAR(DFILE,SCHA,DUMMY)$
    ENDS$
    FOR I=(1,1,JMAX) DO BEGIN
        OMAX(I)=LMAX(I)$
        FOR L=(0,1,LMAX(I)) DO PMAX(I,L)=MMAX(I,L)$ ENDS$
        COMMENT BEGIN OF PROCS *****$
    BEGIN
        INTEGER ARRAY XB(1:20),YAV(1:20),XAV(1:20),FYMI(1:20)$
        INTEGER BACK,BMIN,DIST,HD,IC,IX,IR,JJ,JX,KY,LX,M1,M2,MJ,NYD,
            SLD,SLI,SMAX,TIN,TIX,TMAX,TMIN,XL,XRS$
        BOOLEAN CLOS,CYES,FEED,HKYE,OSCL,PYES,RYES,SYES,VYES$
        PROCEDURE SORMAX(T,I,JMAX,TMAX,MJ)$
            INTEGER ARRAY T,JMAX,TMAX$
            INTEGER I,MJ$
            BEGIN
                INTEGER J,TBIG$
                TBIG=0$
                FOR J=(JMAX(I-1)+3,1,JMAX(I)) DO
                    IF T(J) GTR TBIG THEN BEGIN TBIG=T(J)$ MJ=J$ ENDS$
                TMAX(I)=TBIG$
            ENDS$
        PROCEDURE SORMIN(T,I,JMAX,TMIN)$
            INTEGER ARRAY T,JMAX,TMIN$
            INTEGER I$
            BEGIN
                INTEGER J,TSMA$
                TSMA=1000$
                FOR J=(JMAX(I-1)+3,1,JMAX(I)) DO
                    IF T(J) LSS TSMA THEN TSMA=T(J)$
                TMIN(I)=TSMA$
            ENDS$
        PROCEDURE AVSO(X,Y,LMAX,MMAX,CHA,I,CHEK,NYD)$
            INTEGER ARRAY X,Y,LMAX,MMAX,CHEK$
            INTEGER I,NYD$

```

```

STRING CHA$
BEGIN
    INTEGER XL,XD,YB,KY,N,M1,M2,CONT,MT,MR$
    XB=0$ XD=0$ YB=0$ KY=0$ CONT=0$
    FOR M=(MMAX(I+1,0)+2,1,MMAX(I+1,LMAX(I+1)-1)) DO BEGIN
        IF Y(M) GTR YB THEN BEGIN YB=Y(M)$ MT=M$ ENDS
        IF X(M) GTR XB THEN BEGIN XD=X(M)$ MR=M$ ENDS ENDS
    FOR M=(MR,1,MMAX(I+1,LMAX(I+1))-5) DO
        IF X(M+1) LSS X(M) AND Y(M+1) GTR Y(M) THEN
            CONT=CONT+1$
    FOR M2=(MT,1,MMAX(I+1,LMAX(I+1))-5) DO
        FOR M1=(MT,-1,MMAX(I+1,0)+5) DO
            IF Y(M2) GEQ Y(M1) AND Y(M2) LEQ Y(M1+1) THEN
                BEGIN
                    IF X(M1)-X(M2) GTR XD THEN XD=X(M1)-X(M2)$
                    M1=MMAX(I+1,0)$ ENDS
                FOR M=(MT,1,MMAX(I+1,LMAX(I+1))) DO
                    IF Y(M) LSS Y(MMAX(I+1,0)+1)+NYD/2 THEN BEGIN
                        IF X(M) GTR XB+NYD/8 THEN KY=1$
                        M=MMAX(I+1,LMAX(I+1))+1$ ENDS
                    IF XD GTR NYD/5 OR KY EQL 1 OR CONT GEQ 4 THEN
                        CHA(I)='0' ELSE CHA(I)='A'$
                        CHA(I+1)='- '$$ CHEK(I)=1$ CHEK(I+1)=1$
                ENDS
PROCEDURE BCOREC(X,Y,LMAX,OMAX,MMAX,PMAX,CHA,I,CHEK,FEED,SD,
NYD)$
    INTEGER ARRAY X,Y,LMAX,OMAX,MMAX,PMAX,CHEK,SD$
    INTEGER I,NYD$
    BOOLEAN FEED$
    STRING CHA$
    BEGIN
        INTEGER M1,M2,MB,ME,HD,DIST,L,LE,N,TY,YB$
        DIST=1000$ BD=1000$ YB=0$
        IF LMAX(I+1) GTR 1 THEN MB=2 ELSE MB=1$
        FOR M2=(MMAX(I+1,0)+1,1,MMAX(I+1,MB)) DO
            FOR M1=(MMAX(I,0)+1,1,MMAX(I,1)) DO BEGIN
                DIST=SQRT((X(M1)-X(M2))**2+(Y(M1)-Y(M2))**2)$
                IF DIST LSS BD THEN BEGIN BD=DIST$ ME=M2$ ENDS ENDS
            IF BD LSS NYD/4 THEN BEGIN
                FOR L=(1,1,LMAX(I+1)) DO
                    IF ME GEQ MMAX(I+1,L-1) AND ME LEQ MMAX(I+1,L)
                        THEN BEGIN LE=L-1$
                            M1=10**((LMAX(I+1)-LE)$
                            SD(I+1)=SD(I+1)-(SD(I+1)//M1)*M1$
                            L=LMAX(I+1)+1$ ENDS
                FOR L=(LE,1,LMAX(I+1)) DO
                    MMAX(I+1,L-LE)=MMAX(I+1,L)$
                    LMAX(I+1)=LMAX(I+1)-LE$
                FOR L=(1,1,OMAX(I+1)) DO
                    IF ME GEQ PMAX(I+1,L-1) AND ME LEQ PMAX(I+1,L)

```

```

        THEN BEGIN LE=L-1$ L=OMAX(I+1)+1$ ENDS$
        IF PMAX(I+1,LE+1)-ME LSS 5 THEN LE=LE+1$
        FOR L=(LE,1,OMAX(I+1)) DO
            PMAX(I+1,L-LE)=PMAX(I+1,L)$
        OMAX(I+1)=OMAX(I+1)-LE$
        IF CHEK(I) EQL 2 THEN BEGIN CHA(I)='B'$
            CHEK(I)=1$ ENDS$
        IF OMAX(I+1) GTR 1 THEN FEED=TRUE ELSE FEED=FALSE$
            END ELSE BEGIN CHA(I)='L'$ CHEK(I)=1$ ENDS$
        IF OMAX(I+1) LEQ 1 AND RD LSS NYD/4 THEN BEGIN
            M1=ME+1$ M2=MMAX(I+1,LMAX(I+1))-1$
            FOR M=(M1,1,M2) DO BEGIN
                TY=Y(M1)+(Y(M2)-Y(M1))*(X(M)-X(M1))/(X(M2)-X(M1))$
                IF Y(M)-TY GTR YB THEN YB=Y(M)-TY$ ENDS$
            IF YB/NYD GTR 0.3 THEN CHA(I+1)='R'$
            CHA(I+1)='- '$ CHEK(I+1)=1$ ENDS$
        ENDS$
    PROCEDURE CCHEK(X,Y,LMAX,MMAX,I,CYES,NYD)$
        INTEGER ARRAY X,Y,LMAX,MMAX$
        INTEGER I,NYD$
        BOOLEAN CYES$
        BEGIN
            INTEGER M,ML,MR, XB,XD, XSMA,XTEM$
            XB=0$ XSMA=1000$ XD=0$
            FOR M=(MMAX(I,0)+1,1,MMAX(I,2)) DO
                IF X(M) GTR XB THEN BEGIN XB=X(M)$ MR=M$ ENDS$
            FOR M=(MMAX(I,1)+1,1,MMAX(I,LMAX(I))) DO
                IF X(M) LSS XSMA THEN BEGIN XSMA=X(M)$ ML=M$ ENDS$
            FOR M=(MR+1,1,ML-1) DO BEGIN
                XTEM=X(MR)+(X(ML)-X(MR))*(Y(M)-Y(MR))/(Y(ML)-Y(MR))$
                IF XTEM-X(M) GTR XD THEN XD=XTEM-X(M)$ ENDS$
            IF XD GTR NYD/5 THEN CYES=TRUE ELSE CYES=FALSE$
        ENDS$
    PROCEDURE CLOSE(X,Y,OMAX,PMAX,I,XR,NYD,CLOS)$
        INTEGER ARRAY X,Y,OMAX,PMAX$
        INTEGER I,NYD,XR$
        BOOLEAN CLOS$
        BEGIN
            INTEGER M,MB,ME,XB,DS,DIST$
            DS=1000$ XB=0$
            IF OMAX(I+1) GTR 1 THEN BEGIN
                FOR M=(PMAX(I,0)+1,1,PMAX(I,XR)) DO
                    IF X(M) GTR XB THEN BEGIN XB=X(M)$ MB=M$ ENDS$
                FOR M=(PMAX(I+1,1)+1,1,PMAX(I+1,OMAX(I+1))) DO BEGIN
                    DIST=SQRT((X(MB)-X(M))**2+(Y(MB)-Y(M))**2)$
                    IF DIST LSS DS THEN BEGIN DS=DIST$ ME=M$ ENDS$ ENDS$
                    IF X(ME)-X(MB) LSS NYD/5 THEN CLOS=TRUE ELSE
                        CLOS=FALSE$ END ELSE CLOS=FALSE$
            ENDS$
    PROCEDURE CVSE(X,Y,LMAX,MMAX,CHA,I,CHEK,NYD)$

```

```

INTEGER ARRAY X,Y,LMAX,MMAX,CHEK$
INTEGER I,NYD$
STRING CHA$
BEGIN
  INTEGER M,M1,M2,ML,MR,MB,YD,YT,XS,XB,CYE,EYE$
  XS=2000$ XB=0$ YD=0$ CYE=0$ EYE=0$
  FOR M=(MMAX(I,0)+1,1,MMAX(I,2)) DO
    IF X(M) GTR XB THEN BEGIN XB=X(M)$ MR=M$ ENDS
  FOR M=(MMAX(I,1)+1,1,MMAX(I,LMAX(I))) DO
    IF X(M) LSS XS THEN BEGIN XS=X(M)$ ML=M$ ENDS
  FOR M=(MMAX(I,0)+1,1,MMAX(I,2)) DO
    IF X(M) GTR X(ML) THEN BEGIN MB=M$ M=MMAX(I,2)+1$ ENDS
  FOR M=(MB+1,1,MR-1) DO BEGIN
    YT=Y(MB)+(Y(MR)-Y(MB))*(X(M)-X(MB))/(X(MR)-X(MB))$
    IF Y(M)-YT LSS 0 THEN EYE=EYE+1 ELSE CYE=CYE+1$ ENDS
  FOR M1=(MR,1,ML) DO FOR M2=(MR-1,-1,MB) DO
    IF X(M1) GEQ X(M2) AND X(M1) LEQ X(M2+1) THEN BEGIN
      IF Y(M1)-Y(M2) GTR YD THEN YD=Y(M1)-Y(M2)$
      M2=MB-1$ ENDS
  IF YD GTR NYD/8 AND EYE GTR CYE THEN CHA(I)='E' ELSE
    CHA(I)='C'$ CHEK(I)=1$
END$
PROCEDURE FIYMIN(T,I,JMAX,TMIN,YB)$
  INTEGER ARRAY T,JMAX,TMIN$
  INTEGER I,YB$
  BEGIN
    INTEGER J,TSMA,SLY$
    TSMA=1000$
    FOR J=(JMAX(I-1)+2,1,JMAX(I)) DO BEGIN
      SLY=T(J)-T(J-1)$
      IF SLY GEQ 0 AND T(J) LEQ YB THEN GO TO LL7$
      IF T(J) LSS TSMA THEN TSMA=T(J)$
    LL7: ENDS
    TMIN(I)=TSMA$
  ENDS
PROCEDURE GVSQ(X,Y,LMAX,MMAX,CHA,I,CHEK,NYD)$
  INTEGER ARRAY X,Y,LMAX,MMAX,CHEK$
  INTEGER I,NYD$
  STRING CHA$
  BEGIN
    INTEGER M,M1,XG,XQ,YS$
    XG=0$ XQ=0$ YS=2000$
    FOR M=(MMAX(I+1,1),1,MMAX(I+1,LMAX(I+1))) DO
      IF Y(M) LSS YS THEN YS=Y(M)$
    FOR M=(MMAX(I+2,0)+1,1,MMAX(I+2,1)) DO BEGIN
      FOR M1=(MMAX(I+1,LMAX(I+1)),-1,MMAX(I+1,
        LMAX(I+1)-1)+5) DO
        IF Y(M) GEQ Y(M1+1) AND Y(M) LEQ Y(M1) THEN BEGIN
          IF X(M) GTR X(M1) THEN XQ=XQ+1 ELSE XG=XG+1$
          M1=MMAX(I+1,LMAX(I+1)-1)$ ENDS

```

```

        IF Y(M) GTR YS+NYD/2 THEN M=MMAX(I+2,1)+1$
    ENDS$
    IF XG GTR XQ THEN CHA(I)='G' ELSE CHA(I)='Q'$
    CHEK(I)=1$ CHEK(I+1)=1$ CHA(I+1)='- '$
    ENDS$
PROCEDURE HKVSL(X,Y,LMAX,MMAX,I,HKYE,NYD)$
    INTEGER ARRAY X,Y,LMAX,MMAX$
    INTEGER I,NYD$
    BOOLEAN HKYE$
    BEGIN

        INTEGER M,M1,M2,YS,XK,XL$
        YS=2000$ XK=0$ XL=0$
        FOR M=(MMAX(I+1,0)+1,1,MMAX(I+1,1)) DO
            IF Y(M) LSS YS THEN YS=Y(M)$
            FOR M2=(MMAX(I+1,0)+1,1,MMAX(I+1,1)) DO BEGIN
                FOR M1=(MMAX(I,LMAX(I))-1,MMAX(I,LMAX(I)-1)+2) DO
                    IF Y(M2) GEQ Y(M1+1) AND Y(M2) LEQ Y(M1) THEN BEGIN
                        IF X(M2)-X(M1) LSS 5 THEN XK=XK+1 ELSE XL=XL+1$
                        M1=MMAX(I,LMAX(I)-1)$ ENDS$
                    IF Y(M2) GTR YS+NYD/3 THEN M2=MMAX(I+1,1)+1$ ENDS$
                IF XK GTR XL THEN HKYE=TRUE ELSE HKYE=FALSE$
            ENDS$
        ENDS$
PROCEDURE HVSK(X,Y,LMAX,MMAX,CHA,I,CHEK)$
    INTEGER ARRAY X,Y,LMAX,MMAX,CHEK$
    INTEGER I$
    STRING CHA$
    BEGIN
        REAL T$
        INTEGER M,M1,M2,M3,MX,XL,XU,XB$
        M1=MMAX(I,LMAX(I)-1)$ M2=MMAX(I,LMAX(I))$
        M3=MMAX(I+1,LMAX(I+1)-1)$ XB=0$
        FOR M=(M2+1,1,M3-1) DO IF X(M) GTR XB THEN BEGIN
            XB=X(M)$ MX=M$ ENDS$
        FOR M=(M1+1,1,M2) DO
            IF Y(MX) LEQ Y(M) AND Y(MX) GEQ Y(M+1) THEN BEGIN
                XU=X(M)$ M=M2+1$ ENDS$
        FOR M=(M1+1,1,M2) DO
            IF Y(M3) LEQ Y(M) AND Y(M3) GEQ Y(M+1) THEN BEGIN
                XL=X(M)$ M=M2+1$ ENDS$
        T=(Y(M3-3)-Y(M3))/(X(M3-3)-X(M3))$
        IF T LSS 1.0 AND (XB-XU)/(X(M3)-XL) GTR 1.5
            THEN CHA(I)='K' ELSE CHA(I)='H'$
        CHA(I+1)='- '$ CHEK(I)=1$ CHEK(I+1)=1$
    ENDS$
PROCEDURE OCOREC(X,Y,LMAX,OMAX,MMAX,PMAX,SD,I,NYD,FEED,XR)$
    INTEGER ARRAY X,Y,LMAX,OMAX,MMAX,PMAX,SD$
    INTEGER I,NYD,XR$
    BOOLEAN FEED$
    BEGIN
        INTEGER M1,M2,MC,ME,MT,L,LE,XB,XG,OD,DIST,YD,M,YB,TYS$

```

```

XG=0% OD=1000% DIST=1000% FEED=FALSE% YB=0%
FOR M1=(PMAX(I,0)+1,1,PMAX(I,XR)) DO
  IF X(M1) GTR XG THEN BEGIN XG=X(M1)% MT=M1% ENDS
FOR M2=(PMAX(I+1,1)+1,1,PMAX(I+1,OMAX(I+1))) DO BEGIN
  DIST=SQRT((X(M2)-X(MT))**2+(Y(M2)-Y(MT))**2)%
  IF DIST LSS OD THEN BEGIN OD=DIST% MC=M2% ENDS
ENDS
  ME=MMAX(I+1,LMAX(I+1))%
  XB=X(MC)+NYD/5%
FOR M2=(MC,1,MMAX(I+1,LMAX(I+1))) DO
  IF X(M2) LEQ XB AND X(M2+1) GEQ XB THEN BEGIN
    ME=M2% M2=MMAX(I+1,LMAX(I+1))+1% ENDS
  YD=Y(ME)-Y(MMAX(I+1,0)+1)%
  IF YD LEQ 0 THEN YD=2%
  IF (Y(MC)-Y(MMAX(I+1,0)+1))/YD LSS 2.0 THEN
    FEED=TRUE ELSE FEED=FALSE%
  IF FEED AND OD LSS NYD/4 THEN BEGIN
    FOR L=(1,1,LMAX(I+1)) DO
      IF ME GEQ MMAX(I+1,L-1) AND ME LEQ MMAX(I+1,L)
        THEN BEGIN LE=L-1%
          LE=L-1%
          M1=10**((LMAX(I+1)-LE)%
          SD(I+1)=SD(I+1)-(SD(I+1)//M1)*M1%
          L=LMAX(I+1)+1% ENDS
        FOR L=(LE,1,LMAX(I+1)) DO
          MMAX(I+1,L-LE)=MMAX(I+1,L)%
          LMAX(I+1)=LMAX(I+1)-LE%
        FOR L=(1,1,OMAX(I+1)) DO
          IF ME GEQ PMAX(I+1,L-1) AND ME LEQ PMAX(I+1,L)
            THEN BEGIN LE=L-1% L=OMAX(I+1)+1% ENDS
          IF PMAX(I+1,LE+1)-ME LSS 5 THEN LE=LE+1%
        FOR L=(LE,1,OMAX(I+1)) DO
          PMAX(I+1,L-LE)=PMAX(I+1,L)%
          OMAX(I+1)=OMAX(I+1)-1%
        ENDS
      IF OD LSS NYD/4 AND OMAX(I+1) LEQ 1 THEN BEGIN
        M1=ME+1% M2=MMAX(I+1,LMAX(I+1))-1%
        FOR M=(M1,1,M2) DO BEGIN
          TY=Y(M1)+(Y(M2)-Y(M1))*(X(M)-X(M1))/(X(M2)-X(M1))%
          IF Y(M)-TY GTR YB THEN YB=Y(M)-TY% ENDS
        IF YB/NYD GTR 0.3 THEN CHA(I+1)='R'%
        CHA(I+1)='- '% CHEK(I+1)=1% ENDS
      ENDS
    PROCEDURE OSCULA(X,Y,LMAX,MMAX,I,NYD,OSCL)%
      INTEGER ARRAY X,Y,LMAX,MMAX%
      INTEGER I,NYD%
      BOOLEAN OSCL%
      BEGIN
        INTEGER M,M1,XB,XS,YS%
        XB=0% XS=0% YS=2000%

```

```

FOR M=(MMAX(I+1,0)+1,1,MMAX(I+1,1)) DO
  IF Y(M) LSS YS THEN YS=Y(M)$
FOR M=(MMAX(I+1,0)+1,1,MMAX(I+1,1)) DO BEGIN
  FOR M1=(MMAX(I,LMAX(I)),-1,MMAX(I,LMAX(I)-1)+2) DO
    IF Y(M) GEQ Y(M1+1) AND Y(M) LEQ Y(M1) THEN BEGIN
      IF X(M)-X(M1) LSS 5 THEN XS=XS+1 ELSE XB=XB+1$
      M1=MMAX(I,LMAX(I)-1)$ ENDS$
    IF Y(M) GTR YS+NYD/3 THEN M=MMAX(I+1,1)+1$ ENDS$
  IF XS GTR XB THEN OSCL=TRUE ELSE OSCL=FALSE $
END$
PROCEDURE PCHEK(X,Y,LMAX,MMAX,I,PYES,NYD,YB)$
  INTEGER ARRAY X,Y,LMAX,MMAX$
  INTEGER I,YB,NYD$
  BOOLEAN PYES$
  BEGIN
    INTEGER M,MA,MB,ML,MT,YT,DXR,DXL,XD,XSMA,XNE,NOP$
    YT=0$ XD=0$ XNE=0$ NOP=0$ XSMA=2000$
    FOR M=(MMAX(I+1,0)+1,1,MMAX(I+1,2)) DO
      IF Y(M) GTR YT THEN BEGIN YT=Y(M)$ MT=M$ ENDS$
    FOR M=(MT,1,MMAX(I+1,LMAX(I+1))) DO
      IF X(M) LSS XSMA THEN BEGIN XSMA=X(M)$ ML=M$ ENDS$
    FOR M=(MMAX(I,1),1,MMAX(I,2)) DO
      IF Y(ML) LEQ Y(M) AND Y(ML) GEQ Y(M+1) THEN BEGIN
        MA=M$ M=MMAX(I,2)+1$ ENDS$
      FOR M=(MMAX(I+1,0),1,MMAX(I+1,1)) DO
        IF Y(ML) GEQ Y(M) AND Y(ML) LEQ Y(M+1) THEN BEGIN
          MB=M$ M=MMAX(I+1,1)+1$ ENDS$
        FOR M=(MT+1,1,ML-1) DO BEGIN
          DXL=X(MT)+(X(ML)-X(MT))*(Y(M)-Y(MT))/(Y(ML)-Y(MT))$
          IF X(M)-DXL LSS 0 THEN NOP=NOP+1$ ENDS$
        FOR M=(MB+2,1,MT-2) DO BEGIN
          DXR=X(MB)+(Y(M)-Y(MB))*(X(MT)-X(MB))/(Y(MT)-Y(MB))$
          IF DXR-X(M) GTR XD THEN XD=DXR-X(M)$
          IF DXR-X(M) LSS 0 THEN XNE=XNE+1$ ENDS$
        IF NOP LEQ 1 AND XNE LEQ 1 AND XD GTR NYD/10 AND
          X(ML)-X(MA) LSS NYD/6 THEN PYES=TRUE ELSE
          PYES=FALSE$
        ENDS$
      PROCEDURE RCHEK(X,Y,LMAX,MMAX,I,RYES,NYD)$
        INTEGER ARRAY X,Y,LMAX,MMAX$
        INTEGER I,NYD$
        BOOLEAN RYES$
        BEGIN
          INTEGER M,ML,MT,YB,XD$
          YB=0$
          FOR M=(MMAX(I,0)+1,1,MMAX(I,2)) DO
            IF Y(M) GTR YB THEN BEGIN YB=Y(M)$ MT=M$ ENDS$
          FOR M=(MT,-1,MMAX(I,0)+1) DO
            IF Y(M) LSS Y(MT)-NYD/4 THEN BEGIN ML=M$
              M=MMAX(I,0)$ ENDS$

```

```

FOR M=(MT,1,MMAX(I,LMAX(I))) DO
  IF Y(ML) LEQ Y(M) AND Y(ML) GEQ Y(M+1) THEN BEGIN
    XD=X(M)-X(ML)$
    M=MMAX(I,LMAX(I))+1$ ENDS$
  IF XD GTR NYD/5 THEN RYES=TRUE ELSE RYES=FALSE$
END$
PROCEDURE SCHEK(X,Y,MMAX,I,SYES,SD)$
  INTEGER ARRAY X,Y,MMAX,SD$
  INTEGER I$
  BOOLEAN SYES$
  BEGIN
    INTEGER M,N$
    REAL SLOP$
    IF SD(I) EQL 36 OR SD(I) EQL 362 THEN M=2 ELSE
      IF SD(I) EQL 326 OR SD(I) EQL 326? THEN M=3$
      N=3$
    LSC: N=N+1$
    IF X(MMAX(I,M)-N) EQL X(MMAX(I,M)) THEN GO TO LSC$
    SLOP=(Y(MMAX(I,M)-N)-Y(MMAX(I,M)))/
      (X(MMAX(I,M)-N)-X(MMAX(I,M)))$
    IF X(MMAX(I,M)-N) GTR X(MMAX(I,M)) AND SLOP LSS 1.0
      THEN SYES=TRUE ELSE SYES=FALSE$
  ENDS$
PROCEDURE VCHEK(X,Y,LMAX,MMAX,I,VYES,NYD)$
  INTEGER ARRAY X,Y,LMAX,MMAX$
  INTEGER I,NYD$
  BOOLEAN VYES$
  BEGIN
    INTEGER M,MB,MT,YB,MES$
    YB=0$
    FOR M=(MMAX(I+1,0)+1,1,MMAX(I+1,LMAX(I+1))) DO
      IF Y(M) GTR YB THEN BEGIN YB=Y(M)$ MT=M$ ENDS$
    FOR M=(MT,1,MMAX(I+1,LMAX(I+1))) DO
      IF Y(M) LSS Y(MT)-NYD/2 THEN BEGIN
        MB=M$ M=MMAX(I+1,LMAX(I+1))+1$ ENDS$
      IF X(MB)-X(MT) GTR NYD/4 OR
        Y(MMAX(I+1,LMAX(I+1))-1) GTR Y(MT)-NYD/2 THEN
        VYES=TRUE ELSE VYES=FALSE$
  ENDS$
COMMENT UNIQUE CODE ***** $
FOR I=(1,1,IMAX) DO
  IF SD(I) EQL 32626 OR SD(I) EQL 3626 OR SD(I) EQL 2626 THEN
    BEGIN CHA(I)='Z'$ CHEK(I)=1$ ENDS$
COMMENT STAIRING CODE ***** $
FOR I=(1,1,IMAX) DO BEGIN
  SORMAX(Y,I,JMAX,YMAX,MJ)$ SORMIN(Y,I,JMAX,YMIN)$
  XB(I)=X(MJ)$ YB=Y(MJ)$
  FIYMIN(Y,I,JMAX,FYMI,2000)$ ENDS$
FOR I=(1,1,IMAX) DO BEGIN
  IF SD(I) EQL 326 OR SD(I) EQL 36 OR SD(I) EQL 26 THEN BEGIN

```

```

IF SD(I+1) EQL 326 OR SD(I+1) EQL 36 OR SD(I+1) EQL 26
  THEN BEGIN
    YB=0$ XL=0$ XR=0$
    FOR M=(MMAX(I+1,0)+1,1,MMAX(I+1,1)) DO
      IF Y(M) GTR YB THEN BEGIN YB=Y(M)$ MJ=M$ END$
      M2=MMAX(I+1,LMAX(I+1))-1$
      FOR M=(MJ+1,1,M2-1) DO BEGIN
        DIST=X(MJ)+(Y(M)-Y(MJ))*(X(M2)-X(MJ))/(Y(M2)-Y(MJ))$
        IF X(M) LSS DIST THEN XL=XL+1$
        IF X(M) GTR DIST THEN XR=XR+1$ END$
      IF XR GTR XL THEN BEGIN
        IF (YMAX(I+1)-FYMI(I))/(YMAX(I)-FYMI(I)) LSS 0.7 AND
          (YMAX(I+1)-FYMI(I))/(YMAX(I+1)-FYMI(I+1)) LSS 0.7
          THEN BEGIN
            IF XB(I+1)-XB(I) LSS 10 THEN BEGIN
              CHA(I)='Z'$ CHA(I+1)='- '$ CHEK(I)=1$ CHEK(I+1)=1$
              END$
            YMAX(I+1)=YMAX(I)$ END$ END$ END$ END$
            KY=0$
          IF SD(I) EQL 326 OR SD(I) EQL 36 THEN BEGIN
            IF SD(I+1) EQL 32 OR SD(I+1) EQL 12 OR SD(I+1) EQL 2 THEN
              BEGIN
                FOR M=(MMAX(I,0)+1,1,MMAX(I,1)) DO
                  IF XB(I+1) GEQ X(M) AND XB(I+1) LEQ X(M+1) THEN BEGIN
                    MX=M$ M=MMAX(I,1)+1$ END$
                  IF XB(I)-XB(I+1) GTR 5 AND YMAX(I+1)-Y(MX) LSS 3 THEN
                    KY=1$
                IF KY EQL 1 THEN BEGIN
                  IF CHEK(I-1) NEQ 2 THEN BEGIN CHA(I)='S'$ CHA(I+1)='- '$
                    CHEK(I)=1$ CHEK(I+1)=1$ END$
                  IF I GEQ 2 THEN BEGIN
                    IF SD(I-1) EQL 36 OR SD(I-1) EQL 316 THEN BEGIN
                      NYD=YMAX(I)-YMIN(I+1)$
                      HKVSL(X,Y,LMAX,MMAX,I-1,HKYE,NYD)$
                      IF NOT HKYE THEN BEGIN CHA(I)='S'$ CHA(I+1)='- '$
                        CHEK(I)=1$ CHEK(I+1)=1$ END ELSE BEGIN
                          IF YMIN(I+1)-Y(JMAX(I-1)-1) LSS 0.5*NYD THEN BEGIN
                            CHA(I)='- '$ CHA(I+1)='- '$ CHEK(I-1)=1$ CHEK(I)=1$
                            CHA(I-1)='K'$ CHEK(I+1)=1$ END ELSE
                              IF (YMAX(I)-YMIN(I-1))/NYD GTR 1.5 THEN BEGIN
                                CHA(I)='- '$ CHA(I+1)='- '$ CHEK(I-1)=1$ CHEK(I)=1$
                                CHA(I-1)='P'$ CHEK(I+1)=1$ END$ END$ END$ END$
                                YMAX(I+1)=YMAX(I)$ YMIN(I)=YMIN(I+1)$
                                END$ END$ END$
                                COMMENT NAME POCODE *****$
                                BMIN=0$ SMAX=2000$
                                FOR I=(1,1,IMAX) DO BEGIN
                                  IF SD(I) EQL 7 OR SD(I) EQL 3 OR SD(I) EQL 13
                                    OR SD(I) EQL 12 OR SD(I) EQL 2 OR SD(I) EQL 23
                                    OR SD(I) EQL 6 OR SD(I) EQL 1 THEN GO TO LN1$

```

```

IF YMIN(1) GTR BMIN THEN BMIN=YMIN(1)$
IF YMAX(1) LSS SMAX THEN SMAX=YMAX(1)$
LN1: END$
FOR J=(1,1,3) DO BEGIN
  HD=(SMAX-BMIN)/2$
  TMAX=0$ TIX=0$ TMIN=0$ TIN=0$
  FOR I=(1,1,IMAX) DO BEGIN
    IF SD(I) EQL 7 OR SD(I) EQL 3 OR SD(I) EQL 13
    OR SD(I) EQL 12 OR SD(I) EQL 2 OR SD(I) EQL 23
    OR SD(I) EQL 6 OR SD(I) EQL 1 THEN GO TO LN2$
    IF YMAX(I)-SMAX LSS HD THEN BEGIN
      TMAX=TMAX+YMAX(I)$ TIX=TIX+1$ END$
    IF BMIN-YMIN(I) LSS HD THEN BEGIN
      TMIN=TMIN+YMIN(I)$ TIN=TIN+1$ END$
    LN2: END$
    IF TIX EQL 0 OR TIN EQL 0 THEN GO TO OT$
    YT=TMAX/TIX$ YB=TMIN/TIN$ NYD=YT-YB$ SMAX=YT$ BMIN=YB$
  END$
  HD=NYD/2$
  FOR I=(1,1,IMAX) DO IF CHEK(I) EQL 2 THEN BEGIN
    IF SD(I) EQL 3 OR SD(I) EQL 13 OR SD(I) EQL 23 THEN
      CHEK(I)=0$
      FIYMIN(Y,I,JMAX,FYMI,YB)$
      IF ABS(YT-YMAX(I)) LSS HD AND ABS(YB-FYMI(I)) LSS HD THEN
        PD(I)=1
      ELSE IF YMAX(I)-YT GTR HD AND ABS(YB-FYMI(I)) LSS HD THEN
        PD(I)=2
      ELSE IF ABS(YT-YMAX(I)) LSS HD AND YB-FYMI(I) GTR HD THEN
        PD(I)=3
      ELSE IF YMAX(I)-YT GTR HD AND YB-FYMI(I) GTR HD THEN
        PD(I)=4
      ELSE PD(I)=0$
      IF PD(I) EQL 0 THEN PCHA(1)='0' ELSE
      IF PD(I) EQL 1 THEN PCHA(1)='1' ELSE
      IF PD(I) EQL 2 THEN PCHA(1)='2' ELSE
      IF PD(I) EQL 3 THEN PCHA(1)='3' ELSE
      IF PD(I) EQL 4 THEN PCHA(1)='4'$
    PCHA(2)='Δ'$
    LNTYPE(0)$
    K1=900$ K2=500-I*30$
    LINE(DFILE,K1,K2,DUMMY)$
    CHAR(DFILE,PCHA,DUMMY)$
  END$
  FOR I=(1,1,IMAX-1) DO IF SD(I) EQL 32 THEN BEGIN
    IF SD(I+1) EQL 162 OR SD(I+1) EQL 16
    OR SD(I+1) EQL 62 OR SD(I+1) EQL 6 THEN BEGIN
      FOR M1=(JMAX(I),-1,JMAX(I-1)+1) DO BEGIN
        FOR M2=(JMAX(I)+1,1,JMAX(I+1)) DO
          IF X(M1) LEQ X(M2) AND X(M1) GEQ X(M2+1) THEN
            BEGIN

```

```

      IF Y(M1)-Y(M2) LEQ NYD/10 THEN K1=K1+1 ELSE
      K2=K2+1$ M2=JMAX(I+1)+1$ ENDS$
      IF X(JMAX(I))-X(M1) GTR NYD/4 THEN M1=JMAX(I-1)$
      ENDS$
      IF K1 GTR K2 THEN BEGIN CHA(I)='- '$ CHEK(I)=1$ ENDS$
      ENDS$ ENDS$
      COMMENT INTERSECT ***** $
      FOR I=(1,1,IMAX) DO
      IF SD(I) EQL 6 OR SD(I) EQL 7 THEN BEGIN
      FOR K=(1,1,IMAX) DO FOR L=(JMAX(K-1)+1,1,JMAX(K)) DO
      IF X(L) GEQ X(JMAX(I)-1) AND X(L) LEQ X(JMAX(I-1)+2)
      AND Y(L) GTR Y(JMAX(I)-1) THEN BEGIN
      FOR M1=(L,1,JMAX(K)) DO FOR M=(JMAX(I)-1,-1,JMAX(I-1)+2) DO
      IF X(M1) GEQ X(M) AND X(M1+1) LEQ X(M) OR
      X(M1) LEQ X(M) AND X(M1+1) GEQ X(M) THEN BEGIN
      IF Y(M1) LSS Y(M) THEN BEGIN
      CHA(K)='X'$ CHA(I)='- '$ CHEK(K)=1$ CHEK(I)=1$
      M1=JMAX(K)+1$ ENDS$ M=JMAX(I-1)$ ENDS$
      L=JMAX(K)+1$ K=IMAX+1$ ENDS$
      KY=0$
      FOR K=(1,1,IMAX) DO FOR L=(JMAX(K-1)+1,1,JMAX(K)) DO
      IF X(L) GEQ X(JMAX(I-1)+2) AND X(L) LEQ X(JMAX(I))
      AND Y(L) LSS Y(JMAX(I-1)+2) THEN BEGIN
      FOR M1=(L,1,JMAX(K+1)) DO FOR M=(JMAX(I-1)+2,1,JMAX(I)) DO
      IF X(M1) LEQ X(M) AND X(M1+1) GEQ X(M) OR
      X(M1) GEQ X(M) AND X(M1+1) LEQ X(M) THEN BEGIN
      IF Y(M1) GTR Y(M) THEN KY=1$
      IF KY EQL 1 AND Y(M1) LSS Y(M) THEN BEGIN
      CHA(K)='T'$ CHA(I)='- '$ CHEK(K)=1$ CHEK(I)=1$
      M1=JMAX(K+1)+1$ ENDS$ M=JMAX(I)+1$ ENDS$
      L=JMAX(K)+1$ K=IMAX+1$ ENDS$
      ENDS$
      COMMENT POINT ***** $
      FOR I=(1,1,IMAX) DO
      IF SD(I) EQL 7 THEN BEGIN
      DIST=1000$
      XAV(I)=(X(JMAX(I-1)+1)+X(JMAX(I)))/2$
      FOR IX=(1,1,IMAX) DO BEGIN
      IF IX EQL I THEN IX=IX+1$
      IF ABS(XAV(I)-XB(IX)) LEQ DIST THEN BEGIN
      DIST=ABS(XAV(I)-XB(IX))$ IC=IX$ ENDS$ ENDS$
      IF PD(IC) EQL 1 THEN CHA(IC)='I' ELSE
      IF PD(IC) EQL 3 THEN CHA(IC)='J'$
      CHEK(IC)=1$ CHEK(I)=1$ CHA(I)='- '$ ENDS$
      COMMENT FEEDBACK LOOP ***** $
      BACK=2$
      FOR IR=(1,1,BACK) DO BEGIN
      COMMENT FIRST CLOSE ***** $
      FOR I=(1,1,IMAX-1) DO
      IF CHEK(I) EQL 2 THEN BEGIN

```

```

IF SD(I) EQL 6 AND PD(I) EQL 1
OR SD(I) EQL 16 AND PD(I) EQL 1
OR SD(I) EQL 62 AND PD(I) EQL 1
OR SD(I) EQL 162 AND PD(I) EQL 1 THEN BEGIN
  CLOSE(X,Y,OMAX,PMAX,I,1,NYD,CLOS)$
  IF CLOS THEN BEGIN
    OCOREC(X,Y,LMAX,OMAX,MMAX,PMAX,SD,I,NYD,FEED,1)$
    IF FEED THEN BEGIN CHA(I)='0'$ CHEK(I)=1$
      BACK=BACK+1$ END$
    IF NOT FEED AND CHEK(I+1) EQL 2 THEN BEGIN
      IF SD(I+1) EQL 36 AND PD(I+1) EQL 2
      OR SD(I+1) EQL 32 AND PD(I+1) EQL 2
      OR SD(I+1) EQL 362 AND PD(I+1) EQL 2 THEN BEGIN
        CHA(I)='D'$ CHA(I+1)='- '$ CHEK(I)=1$ CHEK(I+1)=1$ END
      ELSE IF SD(I+1) EQL 362 AND PD(I+1) EQL 3 THEN BEGIN
        CHA(I)='Q'$ CHA(I+1)='- '$ CHEK(I)=1$ CHEK(I+1)=1$ END
      ELSE IF SD(I+1) EQL 36 AND PD(I+1) EQL 3 THEN BEGIN
        IF IMAX LEQ 2 THEN BEGIN CHA(I)='9'$ CHA(I+1)='- '$
          CHEK(I)=1$ CHEK(I+1)=1$ END ELSE
          GVSQ(X,Y,LMAX,MMAX,CHA,I,CHEK,NYD)$ END
      ELSE IF SD(I+1) EQL 36 AND PD(I+1) EQL 1 THEN BEGIN
        CHA(I)='A'$ CHA(I+1)='- '$ CHEK(I)=1$ CHEK(I+1)=1$ END
      ELSE IF SD(I+1) EQL 3162 AND PD(I+1) EQL 1
      OR SD(I+1) EQL 312 AND PD(I+1) EQL 1
      OR SD(I+1) EQL 316 AND PD(I+1) EQL 1 THEN BEGIN
        CHA(I)='C'$ CHA(I+1)='- '$ CHEK(I)=1$ CHEK(I+1)=1$ END
      ELSE IF SD(I+1) EQL 32 AND PD(I+1) EQL 1
      OR SD(I+1) EQL 362 AND PD(I) EQL 1 THEN BEGIN
        AVSO(X,Y,LMAX,MMAX,CHA,I,CHEK,NYD)$ END$ END$ END$ END$
      END$ END$
    COMMENT CLOSE ***** $
    FOR I=(1,1,IMAX-1) DO
      IF CHEK(I) EQL 2 THEN BEGIN
        IF SD(I) EQL 362 AND PD(I) EQL 1
        OR SD(I) EQL 3162 AND PD(I) EQL 1
        OR SD(I) EQL 3262 AND PD(I) EQL 1
        OR SD(I) EQL 36 AND PD(I) EQL 1
        OR SD(I) EQL 262 AND PD(I) EQL 1
        OR SD(I) EQL 26 AND PD(I) EQL 1 THEN BEGIN
          CLOSE(X,Y,OMAX,PMAX,I,2,NYD,CLOS)$
          IF CLOS THEN BEGIN
            OCOREC(X,Y,LMAX,OMAX,MMAX,PMAX,SD,I,NYD,FEED,2)$
            IF FEED THEN BEGIN CHA(I)='0'$ CHEK(I)=1$
              BACK=BACK+1$ END$
            IF NOT FEED AND CHEK(I+1) EQL 2 THEN BEGIN
              IF SD(I+1) EQL 36 AND PD(I+1) EQL 2
              OR SD(I+1) EQL 32 AND PD(I+1) EQL 2
              OR SD(I+1) EQL 362 AND PD(I+1) EQL 2 THEN BEGIN
                CHA(I)='D'$ CHA(I+1)='- '$ CHEK(I)=1$ CHEK(I+1)=1$ END
              ELSE IF SD(I+1) EQL 362 AND PD(I+1) EQL 3 THEN BEGIN

```

```

      CHA(I)='Q'$ CHA(I+1)='- '$ CHEK(I)=1$ CHEK(I+1)=1$ END
    ELSE IF SD(I+1) EQL 36 AND PD(I+1) EQL 3 THEN BEGIN
      GVSQ(X,Y,LMAX,MMAX,CHA,I,CHEK,NYD)$ END
    ELSE IF SD(I+1) EQL 36 AND PD(I+1) EQL 1 THEN BEGIN
      CHA(I)='A'$ CHA(I+1)='- '$ CHEK(I)=1$ CHEK(I+1)=1$ END
    ELSE IF SD(I+1) EQL 3162 AND PD(I+1) EQL 1
    OR SD(I+1) EQL 312 AND PD(I+1) EQL 1
    OR SD(I+1) EQL 316 AND PD(I+1) EQL 1 THEN BEGIN
      CHA(I)='O'$ CHA(I+1)='- '$ CHEK(I)=1$ CHEK(I+1)=1$ END
    ELSE IF SD(I+1) EQL 32 AND PD(I+1) EQL 1
    OR SD(I+1) EQL 362 AND PD(I+1) EQL 1 THEN BEGIN
      AVSQ(X,Y,LMAX,MMAX,CHA,I,CHEK,NYD)$ ENDS ENDS
    ENDS ENDS ENDS
  COMMENT FIRST OTHERS ***** $
  FOR I=(1,1,IMAX) DO
    IF CHEK(I) EQL 2 THEN BEGIN
      IF SD(I) EQL 3162 AND PD(I) EQL 2
      OR SD(I) EQL 362 AND PD(I) EQL 2 THEN BEGIN
        IF I EQL IMAX THEN BEGIN CHA(I)='L'$ CHEK(I)=1$ END ELSE
          BCOREC(X,Y,LMAX,OMAX,MMAX,PMAX,CHA,I,CHEK,FEED,SD,NYD)$
        IF FEED THEN BACK=BACK+1$ END
      ELSE IF SD(I) EQL 316 AND PD(I) EQL 2
      OR SD(I) EQL 36 AND PD(I) EQL 2 THEN BEGIN
        IF SD(I+1) EQL 32 AND PD(I+1) EQL 1 THEN BEGIN
          HKVSL(X,Y,LMAX,MMAX,I,HKYE,NYD)$
          IF HKYE THEN BEGIN CHA(I)='H'$ CHA(I+1)='- '$
            CHEK(I)=1$ CHEK(I+1)=1$ ENDS
          IF NOT HKYE THEN
            BCOREC(X,Y,LMAX,OMAX,MMAX,PMAX,CHA,I,CHEK,FEED,
              SD,NYD)$ END
        ELSE IF SD(I+1) EQL 3262 AND PD(I+1) EQL 1
        OR SD(I+1) EQL 362 AND PD(I+1) EQL 1 THEN BEGIN
          HKVSL(X,Y,LMAX,MMAX,I,HKYE,NYD)$
          IF HKYE THEN HVSK(X,Y,LMAX,MMAX,CHA,I,CHEK)$
          IF NOT HKYE THEN
            BCOREC(X,Y,LMAX,OMAX,MMAX,PMAX,CHA,I,CHEK,FEED,
              SD,NYD)$ END
        ELSE IF SD(I+1) EQL 36 AND PD(I+1) EQL 1 THEN BEGIN
          HKVSL(X,Y,LMAX,MMAX,I,HKYE,NYD)$
          IF HKYE THEN BEGIN CHA(I)='H'$ CHA(I+1)='- '$
            CHEK(I)=1$ CHEK(I+1)=1$ ENDS
          IF NOT HKYE THEN
            BCOREC(X,Y,LMAX,OMAX,MMAX,PMAX,CHA,I,CHEK,FEED,
              SD,NYD)$ END
        ELSE BCOREC(X,Y,LMAX,OMAX,MMAX,PMAX,CHA,I,CHEK,FEED,
          SD,NYD)$
        IF FEED THEN BACK=BACK+1$ ENDS
      ENDS
    END OF FEEDBACK LOOP : ***** $
  FOR I=(1,1,IMAX) DO BEGIN

```

```

IF CHEK(I) EQL 2 AND CHEK(I+1) EQL 2 THEN BEGIN
IF SD(I) EQL 26 AND PD(I) EQL 1
OR SD(I) EQL 262 AND PD(I) EQL 1 THEN BEGIN
IF SD(I+1) EQL 32 AND PD(I+1) EQL 1
OR SD(I+1) EQL 362 AND PD(I+1) EQL 1 THEN BEGIN
VCHEK(X,Y,LMAX,MMAX,I,VYES,NYD)$
IF VYES THEN BEGIN CHA(I)='V'$ CHA(I+1)='- '$
CHEK(I)=1$ CHEK(I+1)=1$ END
ELSE IF CHEK(I+2) EQL 2 THEN BEGIN
VCHEK(X,Y,LMAX,MMAX,I+1,VYES,NYD)$
IF VYES THEN BEGIN CHA(I)='W'$ CHA(I+1)='- '$
CHEK(I)=1$ CHEK(I+1)=1$ CHEK(I+2)=1$ CHA(I+2)='- '$
END$ END$ END
ELSE IF SD(I+1) EQL 36 AND PD(I+1) EQL 1 AND
CHEK(I+2) EQL 2 THEN BEGIN
VCHEK(X,Y,LMAX,MMAX,I+1,VYES,NYD)$
IF VYES THEN BEGIN CHA(I)='W'$ CHA(I+1)='- '$
CHEK(I)=1$ CHEK(I+1)=1$ CHEK(I+2)=1$ CHA(I+2)='- '$
END$ END$ END$ END$
IF CHEK(I) EQL 2 THEN BEGIN
IF SD(I) EQL 62 AND PD(I) EQL 1
OR SD(I) EQL 162 AND PD(I) EQL 1 THEN BEGIN
CHA(I)='C'$ CHEK(I)=1$ END
ELSE IF SD(I) EQL 36 AND PD(I) EQL 4
OR SD(I) EQL 316 AND PD(I) EQL 4
OR SD(I) EQL 3162 AND PD(I) EQL 4
OR SD(I) EQL 362 AND PD(I) EQL 4 THEN BEGIN
CHA(I)='F'$ CHEK(I)=1$ END$
END$ END$
COMMENT LAST ***** $
FOR I=(2,1,IMAX) DO
IF CHEK(I) EQL 2 AND CHEK(I-1) EQL 2 THEN BEGIN
IF SD(I) EQL 312 AND PD(I) EQL 1 THEN BEGIN
IF SD(I-1) EQL 316 AND PD(I-1) EQL 2
OR SD(I-1) EQL 3162 AND PD(I-1) EQL 2
OR SD(I-1) EQL 36 AND PD(I-1) EQL 2
OR SD(I-1) EQL 362 AND PD(I-1) EQL 2 THEN BEGIN
CHA(I-1)='B'$ CHA(I)='- '$ CHEK(I-1)=1$ CHEK(I)=1$ END$
IF SD(I-1) EQL 326 AND PD(I-1) EQL 1
OR SD(I-1) EQL 3262 AND PD(I-1) EQL 1
OR SD(I-1) EQL 362 AND PD(I-1) EQL 1
OR SD(I-1) EQL 36 AND PD(I-1) EQL 1
OR SD(I-1) EQL 262 AND PD(I-1) EQL 1
OR SD(I-1) EQL 26 AND PD(I-1) EQL 1 THEN BEGIN
CHA(I)='V'$ CHA(I-1)='- '$ CHEK(I)=1$ CHEK(I-1)=1$ END$
END$ END$
COMMENT OSCULATE ***** $
FOR I=(1,1,IMAX) DO
IF CHEK(I) EQL 2 AND CHEK(I+1) EQL 2 THEN BEGIN
OSCUA(X,Y,LMAX,MMAX,I,NYD,OSCL)$

```

```

      IF OSCL THEN BEGIN
    IF SD(I) EQL 32 AND PD(I) EQL 1
    OR SD(I) EQL 326 AND PD(I) EQL 1
    OR SD(I) EQL 36 AND PD(I) EQL 1
    OR SD(I) EQL 26 AND PD(I) EQL 1 THEN BEGIN
      IF SD(I+1) EQL 362 AND PD(I+1) EQL 1
      OR SD(I+1) EQL 3262 AND PD(I+1) EQL 1 THEN BEGIN
        CHA(I)='N'$ CHA(I+1)='- '$ CHEK(I)=1$ CHEK(I+1)=1$ END
      ELSE IF SD(I+1) EQL 36 AND PD(I+1) EQL 1
      OR SD(I+1) EQL 32 AND PD(I+1) EQL 1
      OR SD(I+1) EQL 326 AND PD(I+1) EQL 1 THEN BEGIN
        OSCULA(X,Y,LMAX,MMAX,I+1,NYD,OSCL)$
        IF NOT OSCL THEN BEGIN CHA(I)='N'$
          CHA(I+1)='- '$ CHEK(I)=1$ CHEK(I+1)=1$ END$
        IF OSCL THEN BEGIN
          IF SD(I+2) EQL 36 AND PD(I+2) EQL 1
          OR SD(I+2) EQL 32 AND PD(I+2) EQL 1
          OR SD(I+2) EQL 362 AND PD(I+2) EQL 1
          OR SD(I+2) EQL 3262 AND PD(I+2) EQL 1
          OR SD(I+2) EQL 326 AND PD(I+2) EQL 1 THEN BEGIN
            CHA(I)='M'$ CHA(I+1)='- '$ CHA(I+2)='- '$
            CHEK(I)=1$ CHEK(I+1)=1$ CHEK(I+2)=1$ END$ END$
          END$ END$ END$ END$
        COMMENT RELATIVE RECOGNITION ***** $
        IF CHEK(1) EQL 2 AND IMAX EQL 1 THEN BEGIN
      IF SD(1) EQL 362 OR SD(1) EQL 3262 THEN BEGIN
        IF (YMAX(1)-Y(2))/(YMAX(1)-YMIN(1)) LSS 0.5 THEN BEGIN
          CHA(1)='2'$ CHEK(1)=1$ END$ END$ END
        ELSE IF CHEK(1) EQL 2 AND CHEK(2) EQL 2 THEN BEGIN
      IF SD(1) EQL 26 OR SD(1) EQL 326 OR SD(1) EQL 36 THEN BEGIN
        IF SD(2) EQL 3 OR SD(2) EQL 7 THEN BEGIN
          IF (YMAX(1)-Y(1))/(YMAX(1)-YMIN(1)) LSS 0.5 THEN BEGIN
            CHA(1)='2'$ CHA(2)='- '$ CHEK(1)=1$ CHEK(2)=1$ END$
          END$ END$ END$
        FOR I=(1,1,IMAX) DO BEGIN
          IF CHEK(I) EQL 2 THEN BEGIN
        IF SD(I) EQL 3162 AND PD(I) EQL 1 THEN BEGIN
          CVSE(X,Y,LMAX,MMAX,CHA,I,CHEK,NYD)$ END
        ELSE IF SD(I) EQL 32 AND PD(I) EQL 1 THEN BEGIN
          IF CHEK(I+1) NEQ 2 THEN BEGIN CHA(I)='R'$ CHEK(I)=1$ END
            ELSE BEGIN
          IF SD(I+1) EQL 32 AND PD(I+1) EQL 1
          OR SD(I+1) EQL 362 AND PD(I+1) EQL 1 THEN BEGIN
            VCHEK(X,Y,LMAX,MMAX,I,VYES,NYD)$
            IF VYES THEN BEGIN CHA(I)='V'$ CHA(I+1)='- '$
              CHEK(I)=1$ CHEK(I+1)=1$ END$
            IF NOT VYES THEN BEGIN
              RCHEK(X,Y,LMAX,MMAX,I,RYES,NYD)$
              IF RYES THEN BEGIN CHA(I)='R'$ CHEK(I)=1$ END ELSE
                BEGIN

```

```

IF CHEK(I+2) NEQ 2 THEN BEGIN CHA(I)='U'$
  CHEK(I)=1$ CHEK(I+1)=1$ CHA(I+1)='- '$ END$
IF CHEK(I+2) EQL 2 THEN BEGIN
  VCHEK(X,Y,LMAX,MMA,1+1,VYES,NYD)$
  IF VYES THEN BEGIN CHA(I)='W'$ CHA(I+1)='- '$
    CHEK(I)=1$ CHEK(I+1)=1$ CHEK(I+2)=1$
    CHA(I+2)='- '$ END$
  IF NOT VYES THEN BEGIN CHA(I)='U'$ CHA(I+1)='- '$
    CHEK(I)=1$ CHEK(I+1)=1$ END$ END$
END$ END$ END
ELSE IF SD(I+1) EQL 36 AND PD(I+1) EQL 1 THEN BEGIN
  RCHEK(X,Y,LMAX,MMA,I,RYES,NYD)$
  IF RYES THEN BEGIN CHA(I)='R'$ CHEK(I)=1$ END ELSE BEGIN
    IF CHEK(I+2) NEQ 2 THEN BEGIN CHA(I)='U'$
      CHEK(I)=1$ CHEK(I+1)=1$ CHA(I+1)='- '$ END$
    IF CHEK(I+2) EQL 2 THEN BEGIN
      VCHEK(X,Y,LMAX,MMA,I+1,VYES,NYD)$
      IF VYES THEN BEGIN CHA(I)='W'$ CHA(I+1)='- '$
        CHA(I+2)='- '$ CHEK(I)=1$ CHEK(I+1)=1$
        CHEK(I+2)=1$ END$
      IF NOT VYES THEN BEGIN CHA(I)='U'$ CHA(I+1)='- '$
        CHEK(I)=1$ CHEK(I+1)=1$ END$
      END$ END$ END ELSE
        BEGIN CHA(I)='R'$ CHEK(I)=1$ END$ END$ END
  ELSE IF SD(I) EQL 326 AND PD(I) EQL 1 THEN BEGIN
    IF CHEK(I+1) NEQ 2 THEN BEGIN SCHEK(X,Y,MMA,I,SYES,SD)$
      IF SYES THEN BEGIN CHA(I)='S'$ CHEK(I)=1$ END$
      IF NOT SYES THEN BEGIN CHA(I)='R'$ CHEK(I)=1$ END$ END
    ELSE BEGIN
      IF SD(I+1) EQL 32 AND PD(I+1) EQL 1 THEN BEGIN
        VCHEK(X,Y,LMAX,MMA,1,VYES,NYD)$
        IF NOT VYES THEN BEGIN SCHEK(X,Y,MMA,I,SYES,SD)$
          IF SYES THEN BEGIN CHA(I)='S'$ CHEK(I)=1$ END$
          IF NOT SYES THEN BEGIN CHA(I)='R'$
            CHEK(I)=1$ END$ END$
          IF VYES THEN BEGIN CHA(I)='V'$ CHA(I+1)='- '$
            CHEK(I)=1$ CHEK(I+1)=1$ END$ END ELSE
        IF SD(I+1) EQL 36 AND PD(I+1) EQL 3 THEN BEGIN
          SCHEK(X,Y,MMA,I,SYES,SD)$
          IF SYES THEN BEGIN CHA(I)='S'$ CHEK(I)=1$ END$
          IF NOT SYES THEN BEGIN
            PCHEK(X,Y,LMAX,MMA,I+1,PYES,NYD,YB)$
            IF NOT PYES THEN BEGIN CHA(I)='Y'$ CHA(I+1)='- '$
              CHEK(I)=1$ CHEK(I+1)=1$ END$
            IF PYES THEN BEGIN CHA(I)='R'$ CHEK(I)=1$ END$ END$
          END ELSE BEGIN SCHEK(X,Y,MMA,I,SYES,SD)$
            IF SYES THEN BEGIN CHA(I)='S'$ CHEK(I)=1$ END$
            IF NOT SYES THEN BEGIN CHA(I)='R'$ CHEK(I)=1$ END$ END$
          END$ END
        ELSE IF SD(I) EQL 3262 AND PD(I) EQL 1 THEN BEGIN

```

```

IF CHEK(I+1) NEQ 2 THEN BEGIN
  SCHEK(X,Y,MMAK,I,SYES,SD)$
  IF SYES THEN BEGIN CHA(I)='S'$ CHEK(I)=1$ ENDS
  IF NOT SYES THEN BEGIN CHA(I)='R'$ CHEK(I)=1$ ENDS END
  ELSE BEGIN
    IF SD(I+1) EQL 32 AND PD(I+1) EQL 1
    OR SD(I+1) EQL 362 AND PD(I+1) EQL 1 THEN BEGIN
      SCHEK(X,Y,MMAK,I,SYES,SD)$
      IF SYES THEN BEGIN CHA(I)='S'$ CHEK(I)=1$ END ELSE
      BEGIN VCHEK(X,Y,LMAK,MMAK,I,VYES,NYD)$
      IF VYES THEN BEGIN CHA(I)='V'$ CHA(I+1)='-'$
      CHEK(I)=1$ CHEK(I+1)=1$ ENDS
      IF NOT VYES THEN BEGIN CHA(I)='R'$ CHEK(I)=1$ ENDS
      ENDS END
    ELSE IF SD(I+1) EQL 36 AND PD(I+1) EQL 3 THEN BEGIN
      SCHEK(X,Y,MMAK,I,SYES,SD)$
      IF SYES THEN BEGIN CHA(I)='S'$ CHEK(I)=1$ END ELSE BEGIN
      PCHEK(X,Y,LMAK,MMAK,I+1,PYES,NYD,YB)$
      IF PYES THEN BEGIN CHA(I)='R'$ CHEK(I)=1$ ENDS
      IF NOT PYES THEN BEGIN CHA(I)='Y'$ CHA(I+1)='-'$
      CHEK(I)=1$ CHEK(I+1)=1$ ENDS ENDS END
    ELSE BEGIN SCHEK(X,Y,MMAK,I,SYES,SD)$
      IF SYES THEN BEGIN CHA(I)='S'$ CHEK(I)=1$ ENDS
      IF NOT SYES THEN BEGIN CHA(I)='R'$ CHEK(I)=1$ ENDS ENDS
      ENDS END
  ELSE IF SD(I) EQL 36 AND PD(I) EQL 1 THEN BEGIN
    IF CHEK(I+1) NEQ 2 THEN BEGIN
      CHA(I)='S'$ CHEK(I)=1$ END ELSE BEGIN
      IF SD(I+1) EQL 32 AND PD(I+1) EQL 1
      OR SD(I+1) EQL 362 AND PD(I+1) EQL 1 THEN BEGIN
        SCHEK(X,Y,MMAK,I,SYES,SD)$
        IF SYES THEN BEGIN CHA(I)='S'$ CHEK(I)=1$ END ELSE BEGIN
          VCHEK(X,Y,LMAK,MMAK,I,VYES,NYD)$
          IF VYES THEN BEGIN CHA(I)='V'$ CHA(I+1)='-'$
          CHEK(I)=1$ CHEK(I+1)=1$ END ELSE BEGIN
            IF CHEK(I+2) NEQ 2 THEN BEGIN CHA(I)='U'$
            CHEK(I)=1$ CHEK(I+1)=1$ CHA(I+1)='-'$ ENDS
          IF CHEK(I+2) EQL 2 THEN BEGIN
            VCHEK(X,Y,LMAK,MMAK,I+1,VYES,NYD)$
            IF VYES THEN BEGIN CHA(I)='W'$ CHA(I+1)='-'$
            CHEK(I)=1$ CHEK(I+1)=1$ CHEK(I+2)=1$
            CHA(I+2)='-'$ ENDS
            IF NOT VYES THEN BEGIN CHA(I)='U'$ CHA(I+1)='-'$
            CHEK(I)=1$ CHEK(I+1)=1$ ENDS ENDS ENDS
          ENDS END
        ELSE IF SD(I+1) EQL 36 AND PD(I+1) EQL 1 THEN BEGIN
          SCHEK(X,Y,MMAK,I,SYES,SD)$
          IF SYES THEN BEGIN CHA(I)='S'$ CHEK(I)=1$ END ELSE BEGIN
            IF CHEK(I+2) NEQ 2 THEN BEGIN CHA(I)='U'$
            CHEK(I)=1$ CHEK(I+1)=1$ CHA(I+1)='-'$ ENDS

```

```

IF CHEK(I+2) EQL 2 THEN BEGIN
  VCHEK(X,Y,LMAX,MMAX,I+1,VYES,NYD)$
  IF VYES THEN BEGIN CHA(I)='W'$ CHA(I+1)='- '$
    CHEK(I)=1$ CHEK(I+1)=1$ CHEK(I+2)=1$
    CHA(I+2)='- '$ END$
  IF NOT VYES THEN BEGIN CHA(I)='U'$ CHA(I+1)='- '$
    CHEK(I)=1$ CHEK(I+1)=1$ END$ END$ END$ END
ELSE IF SD(I+1) EQL 36 AND PD(I+1) EQL 3 THEN BEGIN
  SCHEK(X,Y,MMAX,I,SYES,SD)$
  IF SYES THEN BEGIN CHA(I)='S'$ CHEK(I)=1$ END$
  IF NOT SYES THEN BEGIN CHA(I)='Y'$ CHA(I+1)='- '$
    CHEK(I)=1$ CHEK(I+1)=1$ END$ END ELSE
  BEGIN CHA(I)='S'$ CHEK(I)=1$ END$ END$ END
ELSE IF SD(I) EQL 36 AND PD(I) EQL 3 THEN BEGIN
  IF SD(I+1) EQL 326 AND PD(I+1) EQL 1
  OR SD(I+1) EQL 3262 AND PD(I+1) EQL 1
  OR SD(I+1) EQL 36 AND PD(I+1) EQL 1
  OR SD(I+1) EQL 362 AND PD(I+1) EQL 1 THEN BEGIN
    CHA(I)='P'$ CHA(I+1)='- '$ CHEK(I)=1$ CHEK(I+1)=1$ END$
  END
ELSE IF SD(I) EQL 362 AND PD(I) EQL 1 THEN BEGIN
  IF CHEK(I+1) NEQ 2 THEN BEGIN
    SCHEK(X,Y,MMAX,I,SYES,SD)$
    IF SYES THEN BEGIN CHA(I)='S'$ CHEK(I)=1$ END$
    IF NOT SYES THEN BEGIN CCHEK(X,Y,LMAX,MMAX,I,CYES,NYD)$
      IF CYES THEN CVSE(X,Y,LMAX,MMAX,CHA,I,CHEK,NYD)$
      IF NOT CYES THEN BEGIN CHA(I)='R'$ CHEK(I)=1$ END$
      END$ END$
  IF CHEK(I+1) EQL 2 THEN BEGIN
    IF SD(I+1) EQL 32 AND PD(I+1) EQL 1
    OR SD(I+1) EQL 362 AND PD(I+1) EQL 1 THEN BEGIN
      SCHEK(X,Y,MMAX,I,SYES,SD)$
      IF SYES THEN BEGIN CHA(I)='S'$ CHEK(I)=1$ END ELSE BEGIN
        VCHEK(X,Y,LMAX,MMAX,I,VYES,NYD)$
        IF VYES THEN BEGIN CHA(I)='V'$ CHA(I+1)='- '$
          CHEK(I)=1$ CHEK(I+1)=1$ END ELSE BEGIN
          CCHEK(X,Y,LMAX,MMAX,I,CYES,NYD)$
          IF CYES THEN CVSE(X,Y,LMAX,MMAX,CHA,I,CHEK,NYD)
          ELSE BEGIN RCHEK(X,Y,LMAX,MMAX,I,RYES,NYD)$
            IF RYES THEN BEGIN CHA(I)='R'$ CHEK(I)=1$ END
            ELSE BEGIN
              IF CHEK(I+2) NEQ 2 THEN BEGIN CHA(I)='U'$ CHA(I+1)='- '$
                CHEK(I)=1$ CHEK(I+1)=1$ END$
              IF CHEK(I+2) EQL 2 THEN BEGIN
                VCHEK(X,Y,LMAX,MMAX,I+1,VYES,NYD)$
                IF VYES THEN BEGIN CHA(I)='W'$ CHA(I+1)='- '$
                  CHEK(I)=1$ CHEK(I+1)=1$ CHEK(I+2)=1$
                  CHA(I+2)='- '$ END$
                IF NOT VYES THEN BEGIN CHA(I)='U'$ CHA(I+1)='- '$
                  CHEK(I)=1$ CHEK(I+1)=1$ END$ END$ END$ END$

```

```

      END$ END$ END
ELSE IF SD(I+1) EQL 36 AND PD(I+1) EQL 1 THEN BEGIN
  SCHEK(X,Y,MMAX,I,SYES,SD)$
  IF SYES THEN BEGIN CHA(I)='S'$ CHEK(I)=1$ END ELSE BEGIN
    CCHEK(X,Y,LMAX,MMAX,I,CYES,NYD)$
    IF CYES THEN CVSE(X,Y,LMAX,MMAX,CHA,I,CHEK,NYD) ELSE
      BEGIN RCHEK(X,Y,LMAX,MMAX,I,RYES,NYD)$
        IF RYES THEN BEGIN CHA(I)='R'$ CHEK(I)=1$ END
      ELSE BEGIN
        IF CHEK(I+2) NEQ 2 THEN BEGIN CHA(I)='U'$ CHA(I+1)='-'$
          CHEK(I)=1$ CHEK(I+1)=1$ END$
        IF CHEK(I+2) EQL 2 THEN BEGIN
          VCHEK(X,Y,LMAX,MMAX,I+1,VYES,NYD)$
          IF VYES THEN BEGIN CHA(I)='W'$ CHA(I+1)='-'$
            CHEK(I)=1$ CHEK(I+1)=1$ CHEK(I+2)=1$
            CHA(I+2)='-'$ END$
          IF NOT VYES THEN BEGIN CHA(I)='U'$ CHA(I+1)='-'$
            CHEK(I)=1$ CHEK(I+1)=1$ END$ END$ END$
        END$ END$ END
      ELSE IF SD(I+1) EQL 36 AND PD(I+1) EQL 3 THEN BEGIN
        SCHEK(X,Y,MMAX,I,SYES,SD)$
        IF SYES THEN BEGIN CHA(I)='S'$ CHEK(I)=1$ END ELSE BEGIN
          CCHEK(X,Y,LMAX,MMAX,I,CYES,NYD)$
          IF CYES THEN CVSE(X,Y,LMAX,MMAX,CHA,I,CHEK,NYD)
        ELSE BEGIN
          IF CHEK(I+2) NEQ 2 THEN BEGIN CHA(I)='Y'$ CHA(I+1)='-'$
            CHEK(I)=1$ CHEK(I+1)=1$ END$
          IF CHEK(I+2) EQL 2 THEN BEGIN
            PCHEK(X,Y,LMAX,MMAX,I+1,PYES,NYD,YB)$
            IF PYES THEN BEGIN CHA(I)='R'$ CHEK(I)=1$ END$
            IF NOT PYES THEN BEGIN CHA(I)='Y'$ CHA(I+1)='-'$
              CHEK(I)=1$ CHEK(I+1)=1$ END$ END$
            END$ END$ END
          ELSE BEGIN SCHEK(X,Y,MMAX,I,SYES,SD)$
            IF SYES THEN BEGIN CHA(I)='S'$ CHEK(I)=1$ END$
            IF NOT SYES THEN BEGIN CCHEK(X,Y,LMAX,MMAX,I,CYES,NYD)$
              IF CYES THEN CVSE(X,Y,LMAX,MMAX,CHA,I,CHEK,NYD)$
              IF NOT CYES THEN BEGIN CHA(I)='R'$ CHEK(I)=1$ END$
            END$ END$ END$ END$
          END$ END$
        END$
      FOR I=(IMAX+1,1,30) DO CHA(I)=' '$
      CHA(30)='A'$
      LNTYPE(0)$
      K1=X(10)$ K2=YMAX(1)+100$
      LINE(DFILE,K1,K2,DUMMY)$
      CHAR(DFILE,CHA,DUMMY)$
      GO TO OT$
    END$
  ENIT: END$
  @ XQT PATTEN

```