GPO PRICE $ ____________

CFSTI PRICE(S) $ ____________

Hard copy (HC) 3.00

Microfiche (MF) .65

ff 653 July 65

FACILITY FORM 602

N68-37582	
(ACCESSION NUMBER)	(THRU)
53	1
(PAGES)	(CODE)
CR-86108	06
(NASA CR OR TMX OR AD NUMBER)	(CATEGORY)

SDC

TM(L)-3052/001/00

AN ON-LINE SYSTEM FOR UTILIZING HAND-PRINTED INPUT: A PROGRESS REPORT

18 December 1967

TM-(L)-3052/001/00

AUTHOR DELIVERED

TECHNICAL MEMORANDUM

(TM Series)

AN ON-LINE SYSTEM FOR UTILIZING HAND-PRINTED INPUT: A PROGRESS REPORT

by

M. I. Bernstein

December 18, 1967

SYSTEM
DEVELOPMENT
CORPORATION
2500 COLORADO AVE.
SANTA MONICA
CALIFORNIA
90406

The work reported herein was supported by contract F1962867C0004, Information Processing Techniques, for the Advanced Research Projects Agency, and contract NAS 12-526, On-Line Character Recognition, for the National Aeronautics and Space Administration.

ABSTRACT

This document is a report of the progress made through September 1967 on developing a program to recognize hand-printed characters on-line in real-time. The program operates under the SDC Time-Sharing System on the Q-32 computer. This document contains details on all of the important facets of the program, primarily in the form of flow charts. A glossary of terms and notation used is included.

FOREWORD

Distribution of this report is provided in the interest of information exchange and should not be construed as endorsement by NASA of the material presented. Responsibility for the contents resides with the organization that prepared it.

The work reported herein was monitored by:

Mr. David Kipping
Technical Monitor
NAS 12-526
Electronics Research Center
575 Technology Square
Cambridge, Massachusetts 02139

Requests for copies of this report should be referred to:

NASA Scientific and Technical Information Facility
P. O. Box 33
College Park, Maryland 20740

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (continued)

1. INTRODUCTION

The feasibility of our approach to the character recognition problem was demonstrated during 1966 on a stand-alone, Philco-2000 (210) computer system. Concurrent with the beginning of support from NASA, the task of developing a usable character recognition technique within the constraints of the SDC Q-32 Time-Sharing System was begun. A brief review of the work performed during the past six months is given here.

In order to reach our goal, modifications were made to both the hardware and software of the Q-32. These modifications included interfacing a display controller to the Q-32 Input Memory and reserving an adequate amount of storage for the refresh buffer. The Time-Sharing System required modifications in two areas: (1) to permit writing into the protected Input Memory from the Q-32, and (2) to provide the real-time update of the display from the user's actions on the tablet. The entire set of modifications was only recently completed; however, a workable substitute has been available since March 1967. Prior to that time, various parts of the character recognition program were coded and checked out using raw data collected on magnetic tape from the Philco-2000 installation.

During the time that we worked from recorded inputs, several routines that are the heart of the process were written and checked out. Among these were the feature extraction, corner-detection and raw data smoothing programs, plus the basic programs for building and searching the character dictionary. Because of the limited supply of recorded data, all of the programs that dealt with the input data proved to be preliminary versions, and were subsequently modified when tested against "live" data. One program that has remained relatively stable throughout is one that computes the relationship between the strokes of multi-stroke characters.

Beginning in March, the analysis and dictionary routines were coupled to live inputs. Several unexpected problems arose. The first was noisy input from the tablet; we had no way of smoothing the data before it was filtered for redundancy. This (together with a difference between the Philco 2000 and Q-32 sampling rates) forced us to re-examine our corner-detection scheme. The discovery of a satisfactory post-filtering smoothing method overcame most of the problem.

Subsequently, several capabilities were added: (1) the ability to save and recall dictionaries; (2) a program that would permit testing under conditions that approach those of the real world, incorporating some minor editing ability (namely, erasure and replacement of any character displayed); (3) two routines to manipulate the dictionaries themselves. One of these dictionary-manipulation routines can be used to remove all definitions for a given output character or all definitions that have a usage level below a threshold chosen by the user. The other routine is used to "optimize" the dictionary by creating--where possible--additional cross-linkages between multi-stroke definitions of the same output character and by removing redundant strokes. This process not only saves space and search time, but improves the recognition level by increasing the number of definitions (or combinations of strokes) available.

Tasks remaining to be done include testing alternative feature-extraction methods, further investigation into corner-detection, and the gathering and interpretation of performance statistics for the various versions of the character recognition program used by an appropriate set of users.

Most of this report is devoted to a description of the current version of the entire program. Basically, this description consists of a set of flowcharts and accompanying annotation and descriptions. Many of the algorithms employed are "empirical," i.e., generated by trying successive approximations on recorded data until some optimum had been reached or some local minimum or maximum passed through.

Although recognition rates of 100 percent for large alphabets are probably not attainable within the bounds of our current approach, the present recognition program (with appropriate "tuning") will perform very well on smaller or simpler character sets.

2. ANALYSIS

No formal testing has yet been performed to determine recognition rates for a given set of individuals. Several problems remain to be solved before this can be done: (1) Corner-detection is still under study. Methods that work for individuals who write slowly do not work well for those who write rapidly, and vice versa. We are still looking for a technique that is relatively foolproof. (2) The feature extraction requires some changes and some tuning. The prime problem involves strokes that generate a single loop feature. Our discrimination among this set is inadequate. In a set of 60 or so characters, this is unlikely to be of any consequence, since the ambiguities occur between such things as α and 6.

The other principal objectives of this study have been or will be met. The ones for which we held little hope are already accomplished, such as the dictionary-manipulation program for purging and optimizing a dictionary. The only task remaining to be done is that of determining if single-stroke characters can be optimized in a way similar to that in which we optimized the dictionary for multi-stroke characters.

3. NOTATION

Because there exists little, if any, standard notation in either the field of programming or character recognition, we have invented notation where we felt it lent both brevity and clarity to the presented material, and have stayed as close as possible to "accepted" representation in all other cases.

xy or p

Represent an ordered pair of coordinates; usually provided by the input. p is used when it will not cause ambiguity. When operations or functions are applied to p, they are applied fully to both x and y. Both representations may be subscripted, either to denote a particular order in the sequence of coordinates that make up a stroke, or to identify a unique pair.

dxy or Δxy or exy

Represent the signed difference between two pairs of coordinates, x_1-x_2, y_1-y_2. When tests or operations are performed on individual members of the pair, they are separated in the form Δx and Δy.

h

Represents the heading--as computed by the function DIRQ (Δxy)--between two points. Though the values are shown in the table as signed integers, those numbers are actually treated as signed two's complement fractions for computational purposes.

Δh

The signed difference between two headings, h_1 and h_2. The sign indicates the direction of rotation (minus for counter-clockwise and plus for clockwise), and the magnitude represents the amount of directional change. Note that since the computation is done on signed two's complement fractions, the largest change that can occur is 180°, indicated by a -1 (-16 in the DIRQ table of values, t_{32}).

S

Stands for stroke. In our program, a stroke is made up of the ordered set of coordinates, p, input to the program between the sequence of pen switch on and pen switch off.

Min and Max

Used in their usual mathematical meaning. When either is applied to a stroke, S, the x and y coordinates are treated independently. Therefore, min (P_i) means min (x_i) and min (y_i); max (P_i) has a similar meaning.

R(Z)

The minimum rectangle surrounding the set of points, Z, which may be a stroke, S, or a subset of the stroke. Usually that subset that constitutes a feature R(Z) is obtained by computing Min(Z) and Max(Z) and is a pair of ordered pairs (x_{min},y_{min}), (x_{max},y_{max}).

$\bar{R}(Z)$

The center of the minimum rectangle R(Z). It is an ordered pair $\bar{x},\bar{y}$. It is computed as $1/2(x_{min}+x_{min})$, $1/2(y_{min}+y_{max})$.

Size (A)

An ordered pair of differences, xy, specifying the size of an entity A. It is usually computed from R(A). The notation used in the flow charts is Size (R(S)). It is computed as $\Delta x=x_{max}-x_{min}$, $\Delta y=y_{max}-y_{min}$. Note that both Δx and Δy are ≥ 0.

Ch

Stands for character. $Ch^{\emptyset}$ is used to specify an output character, and may only be a legitimate member of the output character set (the set from which the user defines his input). Ch^{I} is used to specify an input character, and is a collection of from one to n strokes, $(S_0, S_1 \ldots S_n)$.

σ

Represents output from the ANALYZER routine other than a $Ch^{\emptyset}$. Five σ values have been used. They are:

σ_o = a stroke with too many features or (in the SAMPLE routine) a Ch^{I} with too many strokes.

σ_1 = a stroke or character not found in the dictionary; thus, an undefined character.

σ_2 = a stroke or character in the dictionary that ends at an intermediate node, has no $Ch^{\emptyset}$ attached, and is, therefore, an intermediate, undefined character. For example, if the first sample provided to SAMPLE were a four-stroke M and the dictionary were empty, the first three strokes entered would have a σ_2 appended to them; the fourth stroke would have the $Ch^{\emptyset}$ "M" appended as its definition.

σ_3 = end of strokes or a vacuous stroke.

σ_4 = an invalid stroke, defined as one for which $\Delta x+\Delta y \geq 30$ for any adjacent pair, p_i, p_{l+1}.

A(S) Represents the ANALYZER output for S. A(S) may have 0 as an intermediate value, but its final value is always either $Ch^{\emptyset}$ or σ.

4. PROGRAM DESCRIPTION

4.1 DATA FLOW

Ignoring those portions of the implemented program that are used for testing and debugging, we shall describe the sequence of events that take place during the execution of the program. The flow of data through the system, including both input data and control signals, will be described.

The over-all functioning of the system involves three hardware units, plus the software of the time-sharing system. The three hardware units are: (1) the PDP-1 computer, which interfaces and buffers all interactive input and output; (2) the Input Memory, a 16 K core storage module of 48-bit words that is directly addressable by both the PDP-1 and Q-32 computers; and (3) the Q-32 computer, a large (65K 48-bit word), fast (2.5-μsec cycle time), general-purpose digital computer on which TSS runs.

The Input Memory can be directly read by an object program running under the Time-Sharing System in the Q-32, but only the TSS supervisor may write in Input Memory. Because there are no user programs running in the PDP-1, there are no restrictions on its use of Input Memory.

The Graphic Tablet Display Console, around which this program is constructed, interfaces with the above hardware as follows: the RAND Tablet (Grafacon 1010A) used for input is connected to the PDP-1 through a hardware interface. Each time an input is ready at that interface, the PDP-1 is interrupted. These interrupts occur at two different rates: every 30 ms when the tablet's pen switch is off, and every 4 ms when it is on. This pen switch on rate is variable between 1 ms and 16 ms. The 1-ms rate cannot be handled by the PDP-1; samples greater than 4 ms apart would not provide adequate data.

The CRT display for the console is directly connected to Input Memory through another hardware interface. A block of 1024 words of Input Memory is reserved in a fixed place for the refresh buffer of the display. The interface reads from this buffer and "paints" the contents on the display (continuously) at a rate of approximately 32 frames per second. The display refresh may be turned on and off manually. When on, the contents of the 1024-word buffer are shown on the display without regard to the condition of other parts of the system.

To start the character recognition program, it is loaded for execution under TSS via a teletype console in the same way as any other user's program. The program then queries the user as to his intent. For purposes of discussion, assume he wishes to use the tablet for input. He indicates this intent. The program* (running in the Q-32 under TSS) then requests that the tablet (in reality, the GRID program of the PDP-1, which is a system routine, not a user program) be attached to this user as a private device. This insures that all inputs from the tablet are directed only to the user's program. The GRID program at this point is most likely to have SW1 set to IGN (see Figure 2), though it may have been left set to TB, but that will not affect program execution. The program then generates a display image in Q-32 core consisting of three pushbuttons, labeled "DRAW", "SAMPLE", and "TEST." Only the latter two concern us here. This image block also contains the appropriate control words to set the GRID program for use--that is, to set SW1 to TB and assign the first relative location in the buffer where GRID may store its inputs. This block of Q-32 core (which we shall refer to as DB) is transferred to the Input Memory display buffer (which we shall call IMB) by a call to the TSS Supervisor. When TSS returns control to the user's program, it then requests input from GRID and goes into "wait" status through another supervisor call. This means that the object program in the Q-32 will not be run again until the GRID program informs the Q-32 that input has been completed and it is giving up control.

With GRID in control, the user now positions his reflected pen position spot over the "SAMPLE" Button and presses hard enough to turn the pen switch on; he then releases the pressure and the switch goes off. GRID has placed the input points in IMB, so that they are now a part of the display image. The Q-32 only allows space for one input point by GRID. When it finds that the buffer is full, it calls TSS to take the user out of "wait" status; GRID then essentially goes to wait status itself. (If there were more space in IMB, control would have been returned after a time delay on pen up of 1/4 sec.)

*This routine is not part of the recognition program itself, and thus is not documented here.

The user is informed that control has returned to the Q-32 by the lack of the moving spot reflecting his current pen position. Note that it is possible to enable GRID to accept inputs without putting the Q-32 program into "wait" status, but we have not tried this, even accidentally. If this were done, the two programs (GRID and the user's program in the Q-32) could get out of synchronization and both end up waiting for each other, thus effectively killing the program.

The Time-Sharing System next takes the user's Q-32 program out of "wait" status. The program reads the pen input from Input Memory, decodes the coordinates in terms of button position, and either rejects or accepts them. If the input is rejected, the program simply goes back to the beginning, rewrites Input Memory, calls GRID and waits. If the inputs are accepted, the program named in the button pushed (in this case, SAMPLE) is called and control passed to it.

SAMPLE (see Figure 4) generates its initial frame in DB. Having divided up the buffer block appropriately, it sets the time delay (T_D) to 2 seconds, turns the smoothing flag (SMFLG) on, calls the supervisor to write DB into IMB, calls GRID, and goes into "wait"status.

The user next sees a new display (see Figure 3). He may then push a button to change the keyboard displayed, or draw a character in the writing area. GRID reacts the same way as above. Assume that the user has drawn a character, say a "|". The user should then wait for some response from the Q-32 program to show on the display before he proceeds. If he inadvertently starts to draw or push a button before GRID has given up control, he will add information to the input that he has not intended and may--as a result--unknowingly add a meaningless character definition to the dictionary. When control is returned to SAMPLE by the system, it checks the validity of the input as a function of both the setting of CHSW, and the position of the initial and final coordinates of the first input stroke. (A stroke is defined as all of the input occurring between pen switch on and pen switch off).

Finding a valid stroke, SAMPLE calls ANALYZER. The stroke is analyzed and the resultant feature string searched for in the dictionary. Note that when SAMPLE is in control, ANALYZER processes all strokes before returning, on the assumption that all of the input belongs to one character. Since the user's dictionary is empty, the input will not be found, SAMPLE adds the special character "∿" to the display image, relocates the INKORG so that the inputs will not be erased, sets the CHSW to U, writes DB in the IMB, calls GRID and waits. The user must now "push" one of several buttons. If he draws another character or pushes the wrong button, his input will be erased and the program will wait for another input.

After completing the dictionary, the user makes an input at coordinate 0,0. This, in essence, causes SAMPLE to return control to the program that called it--the control program of three pushbuttons. If the user were to make an input at 0,0 in this control program, he would return control to the teletype.

The difference between SAMPLE and TEST is that there are no invalid inputs to TEST. Multi-stroke inputs are not treated as a single character.

The ability to erase or replace an output character from the display is not crucial to the objectives of the TEST. Its main function is to test the user-constructed dictionary. Replacement and erasure were incorporated to provide a more realistic usage environment, and to create a better demonstration vehicle. Generating an output character of the same size and at the same position as the input character aids in creating the proper atmosphere. These capabilities were added to TEST in order to solve some of the known problems that will occur when the character recognizer is used as a tool for an actual problem solution.

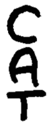
All of the information concerning output character position, size, and location in the output buffer is placed in DLIST, which is a simple linked list, external to the actual buffer. DLIST could have been implemented in several ways. The linked list was a design choice.

If the user then pushes "TEST", control will be turned over to the TEST program (see Figure 7). TEST sets up DB in two portions: (1) a simple linked list to minimize wasted space in displaying arbitrary characters in 5 x 7 dot matrix form, and (2) a block reserved for "ink".

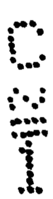
Assume that the user draws

This takes five strokes. When GRID returns control to TEST, the inputs are moved from IMB to DB, and ANALYZER is called. ANALYZER returns control to TEST if one of several events has occurred: (1) all of the strokes in DB have been processed; (2) the matching feature string in the dictionary has no successor stroke appended to it; (3) the next stroke processed does not match any of the successor strokes or it cannot match a first stroke; (4) the input has too many features and is thus considered a "scrub" or erasure, or it has a noisy point and is considered invalid. One of the results of this kind of interactive control is that a non-recognized stroke can change the "meaning" of the remaining strokes from that intended. Each time ANALYZER returns control to TEST, TEST adds or deletes characters in DB; only when all strokes are processed does the result appear on the user's display, replacing his hand-drawn input. If all goes well, the user will see appear at once in the display, although each character was placed in DB as it was recognized and returned by ANALYZER.

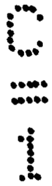
To illustrate what can happen if a stroke is not found, assume that the user draws the following:



on the tablet in the order,C,∩,-,-,|, and the ANALYZER could not find the "∩" of the "A" in the dictionary. The user would see either



or



depending on where the center of the "∩" is with respect to the two strokes that are not taken as an "=". The first of these two illustrations shows the most likely case.

4.2 RESPONSE TIME

We have been able to provide instantaneous response in the one critical area required--namely, feedback from pen inputs to the display so that "ink" flows from the "pen" in real time.

Response time for the user upon completion of input is a function of many variables. Naturally, the more interactive users there are on the system, the longer the delays. In particular, the more people using tape and disc, the longer the delays. Also, the more input there is from the tablet, the longer are the potential delays. The effect of this factor is hard to measure because of the other variables, but if the processing takes more than one quantum of interactive user time (currently about 600 ms), the user is swapped out and is at the bottom of the interactive queue for one cycle.

In general, when there are less than 20 other users on the system, response is acceptable (though not instantaneous). When there are more than 25 users on the system, response is slow enough to be annoying; when there are more than 30 users on, response is intolerable.

If the SDC Time-Sharing System had provisions for priorities among the interactive users, some of these annoyances could be lessened, but it doesn't and so from time to time we will be annoyed.

5. DESCRIPTION OF PROGRAM SEGMENTS

5.1 GRID SUBROUTINE

GRID is a program that runs on the PDP-1 and provides the interface between the RAND Tablet and the Q-32 Time-Sharing System.

All communication between GRID and the calling program takes place via the Input Memory Buffer (IMB). The calling program supplies the variables (SMFLG, TD and INKLOC) in the control words and GRID places the tablet input data (ink) in the buffer beginning at the location specified in INKLOC. The format of these inputs is shown in Figure 1.

A flow chart of the GRID subroutine is shown in Figure 2. It is not essential that this routine start with SWI at IGN, since the time-out on TD guarantees that SWI cannot remain indefinitely in the TB position if any inputs have been made, regardless of the state of the Q-32 program. Note that the PDP-1 is interrupted at two different intervals: at 4-ms intervals when psw (pen switch) is on, and at 30-ms intervals when psw is off.

SW2 is set when psw is first detected. This saves both time and space, because of the way the PDP-1 reads Input Memory. The smoothing algorithm (enabled when SW2 is set to "on") merely replaces the oldest xy with the one just read, then computes the average xy as output. The index j is operated as a ring counter, modulo 8.

The value of 3 used in the filter was arrived at empirically for drawing very small characters (using no smoothing). Small characters could be drawn with smoothing using a filter value of 2. N_s is the number of strokes input. Until recently, N_s has not been implemented properly, thus it is used only by time-out testing in GRID and not by the Q-32 routines. Another function that could have been included, had space been available (time was no problem), is the computation of the minimum rectangle surrounding each stroke, R(S). This would have saved some time in the Q-32 programs, at the cost of buffer space in the INK area.

5.2 SAMPLE SUBROUTINE

The SAMPLE subroutine is used to build and test dictionaries in a one-character-at-a-time mode. This program allows only one action at a time--drawing a character (multi-stroke characters are allowed) or pushing a button.

	Byte							
Location	0	1	2	3	4	5	6	7
0					SMFLG	N_S		
1								
2								
INKLOC	X		Y		M	P_C		
1023	TB Location				T_D	INKLOC		

Typical Display Word Inserted by INK

Figure 1. Storage Map for Display Buffer (DB) and Input Memory Buffer (IMB)

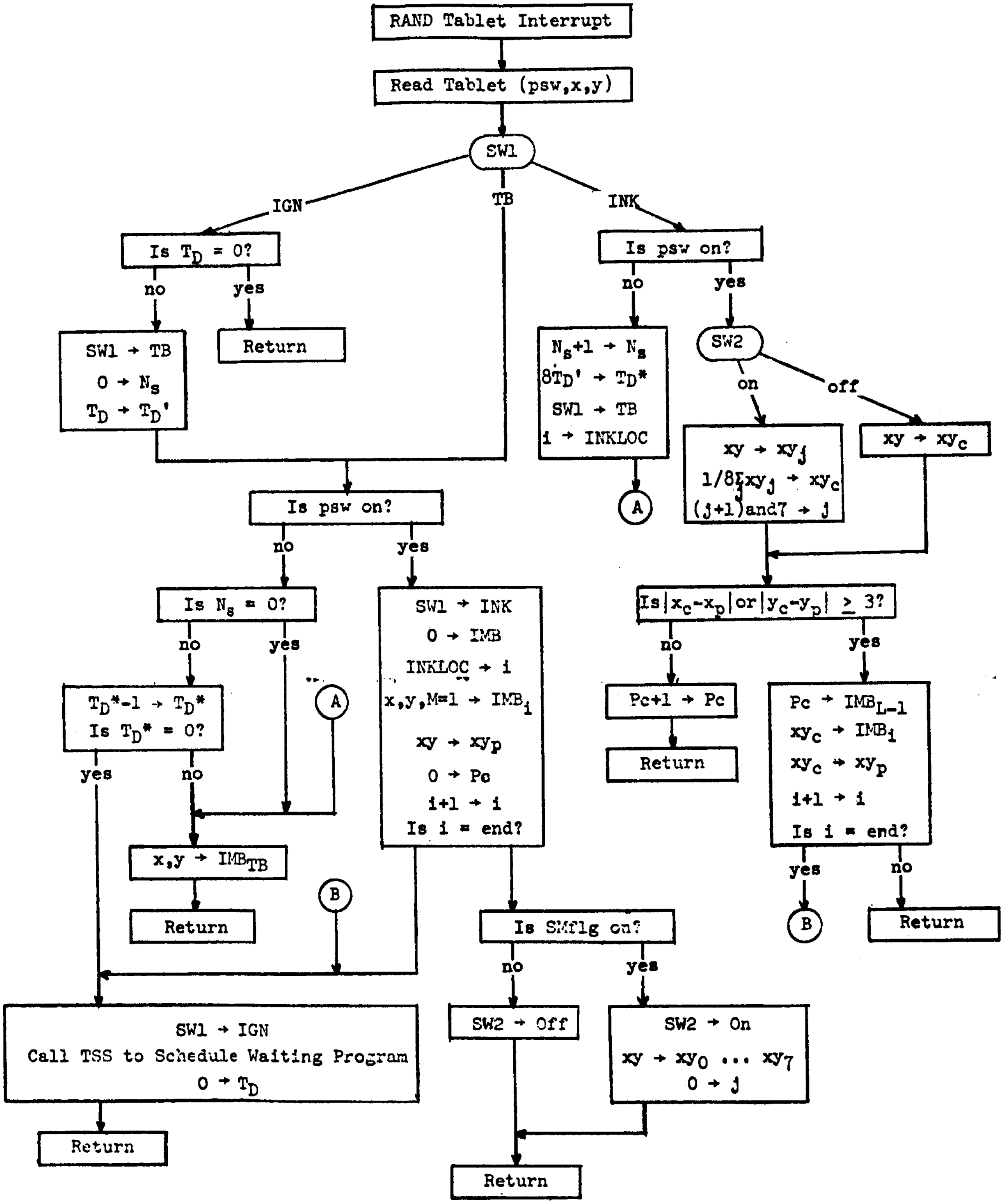


Figure 2. GRID Subroutine Flow Chart

Figure 3 shows a layout of the display the user sees when this program is called. The user can then select one of five keyboards. The five keyboards contain (1) digits, brackets, and relationals; (2) upper-case Roman letters; (3) lower-case Roman letters; (4) punctuation and special marks; (5) Greek letters. The maximum number of characters per keyboard is 26.

A flow chart of the SAMPLE subroutine is shown in Figure 4. Those parts of the flow chart concerned with the detailed control of the Display Buffer indicate when various strokes are left and when they are erased. The buffer is allocated as shown in Figure 5.

Communication between SAMPLE, ANALYZER, and DEFINE is through the inputs in the DB and a set of standard global communication registers plus, of course, the computer's accumulator.

When an input is found in the dictionary as a defined character, the output character replaces the input at the same place on the display at approximately the same size. If the input is not defined, it is permitted to remain and one of two special characters, "ʔ" or "~", is output at the ILØC (a position on the display surface) (see Figure 1).

The CALL GRID AND WAIT is a Time-Sharing System dispatcher call. The call could be given without a wait, but there is nothing that needs to be done in the interval; also synchronization between the two programs would be more difficult.

Although not shown in Figure 4, there is a small master program to which SAMPLE and TEST (see below) exit when the input stroke is found to be on the 0,0 coordinate of the tablet.

5.3 TEST SUBROUTINE

TEST allows the user to test a dictionary in a multi-character, interactive mode. In addition, it provides two editing features: replacement of an existing character with an input, and erasure of one or many characters with one scrub. A flow chart of the subroutine is shown in Figure 7.

The mode flag is set to "test" so that ANALYZER does not assume that all of the input strokes constitute one character.

The Display Buffer for TEST is organized as shown in Figure 6.

The Display List (DLIST) is a linked list of output characters and other information required for physically locating the individual characters on the display surface.

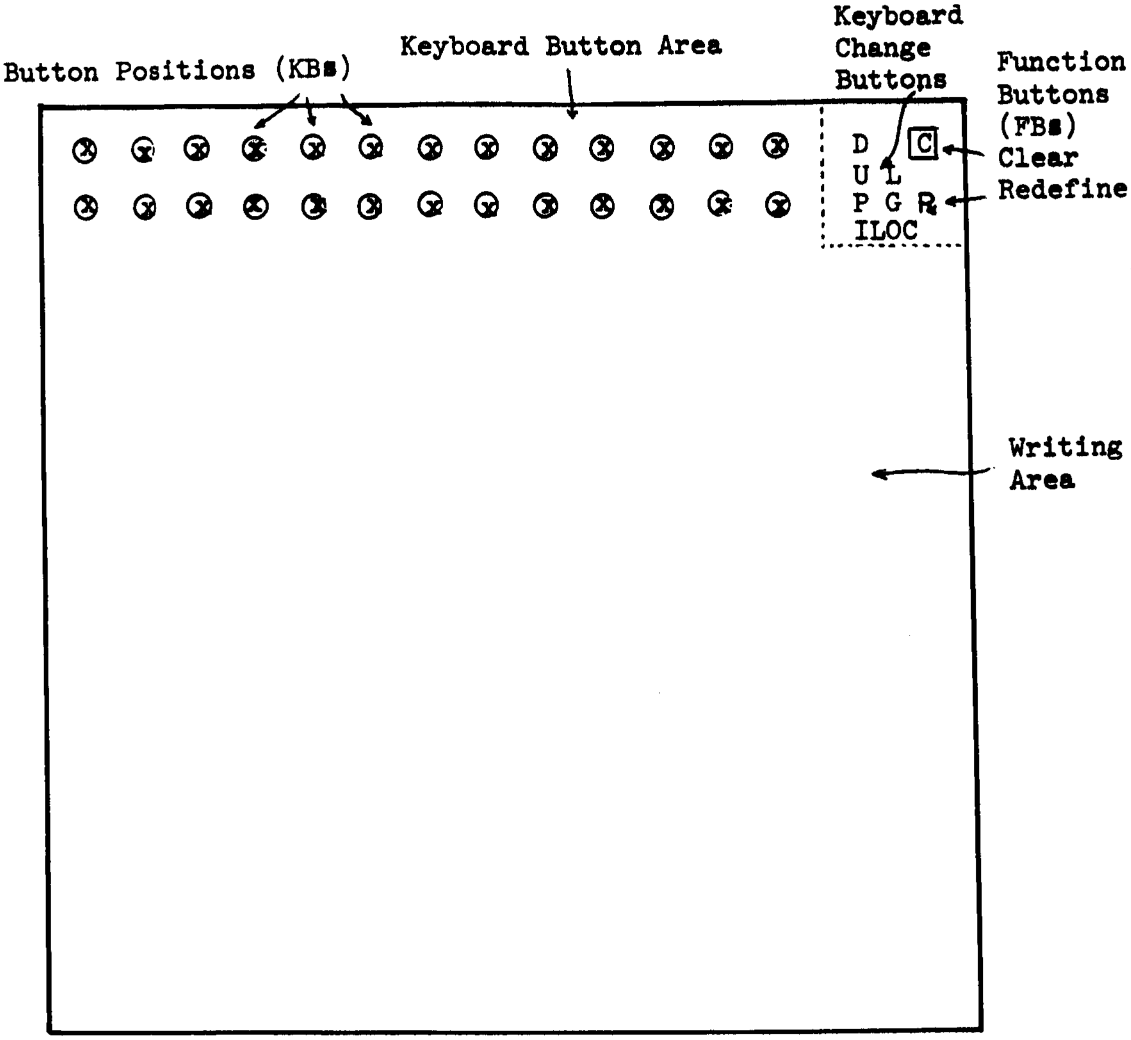


Figure 3. Display for SAMPLE Program

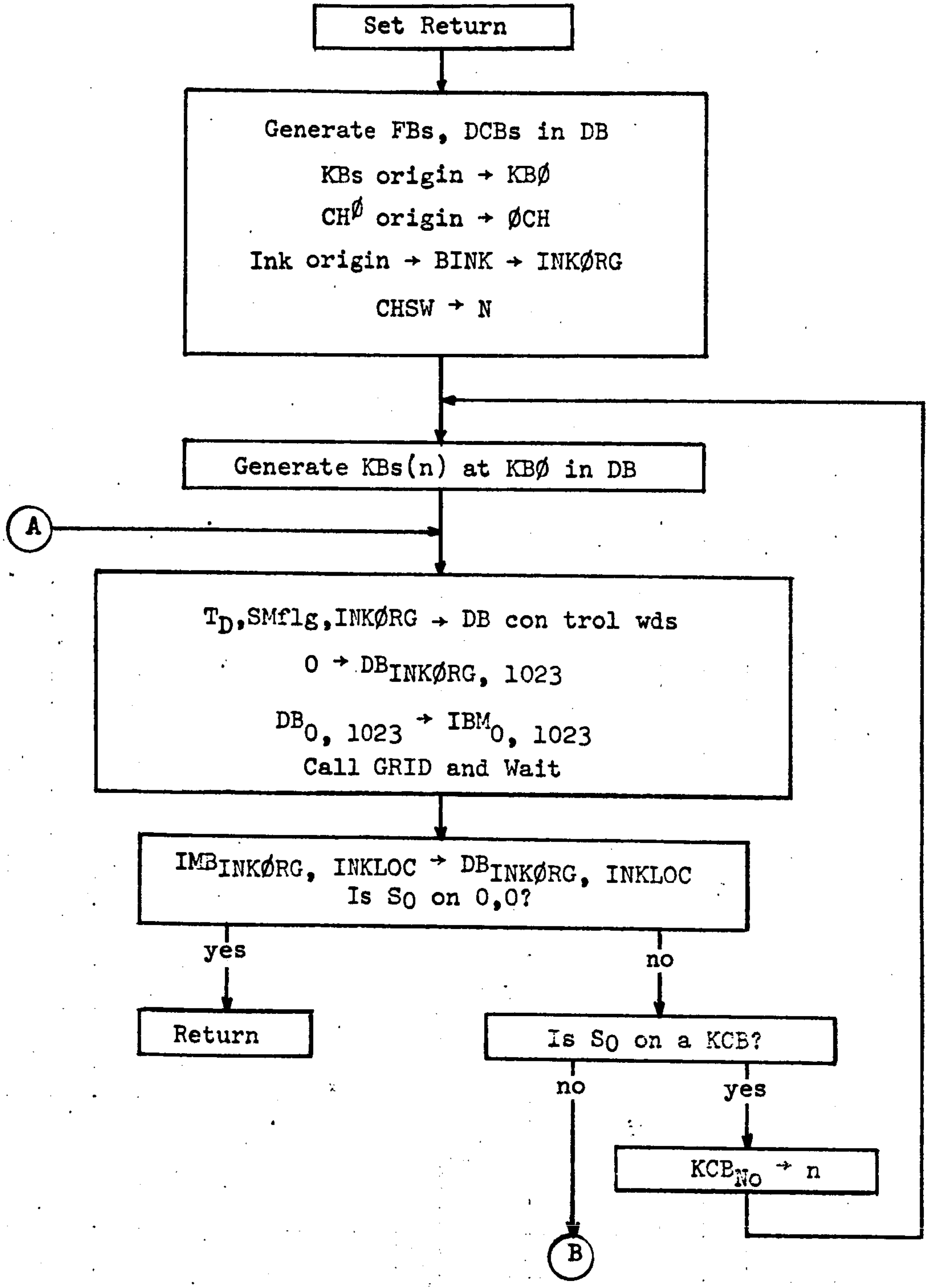


Figure 4a. SAMPLE Subroutine Flow Chart

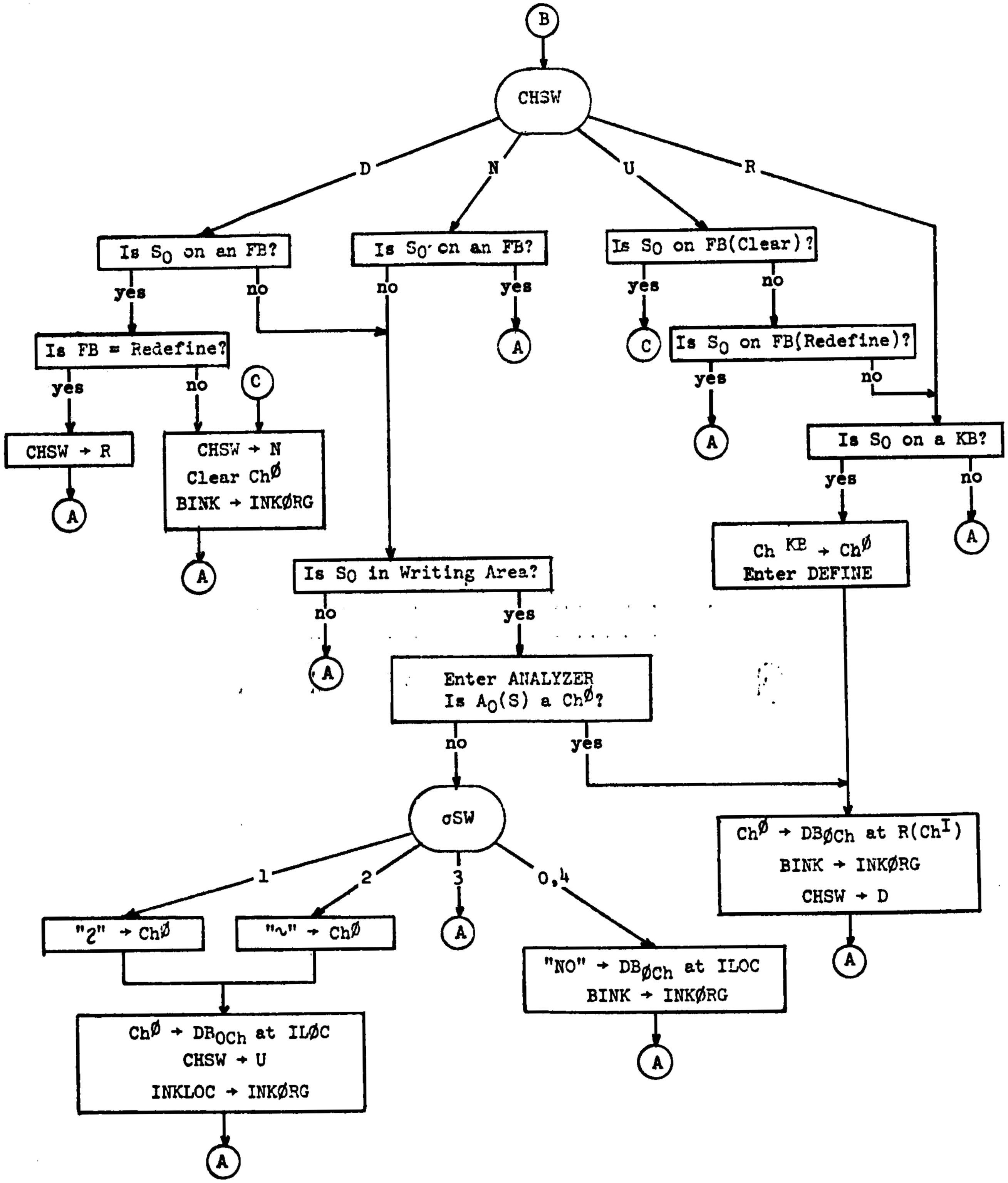


Figure 4b. SAMPLE Subroutine Flow Chart (continued)

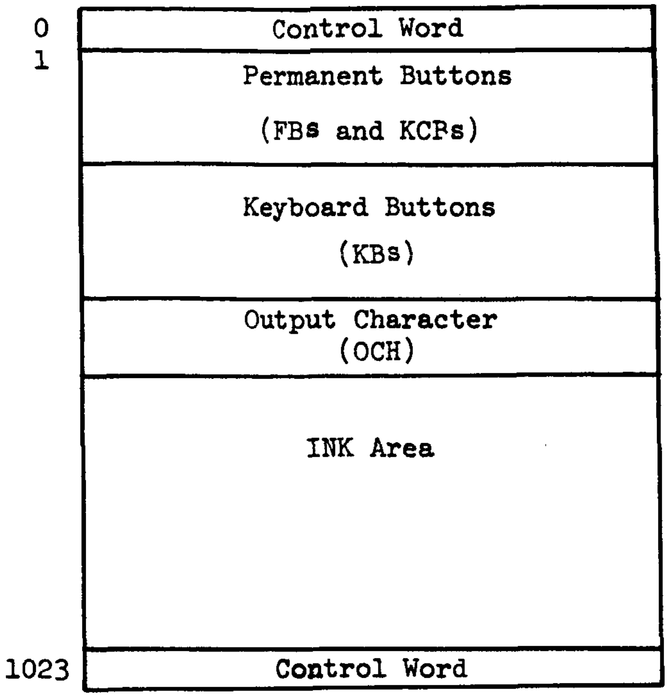


Figure 5. Display Buffer Allocation for SAMPLE

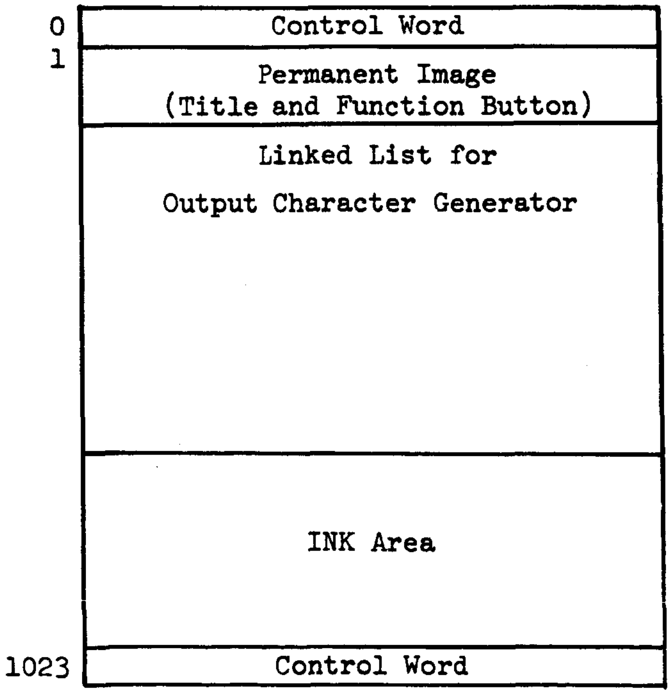


Figure 6. Display Buffer Allocation for TEST

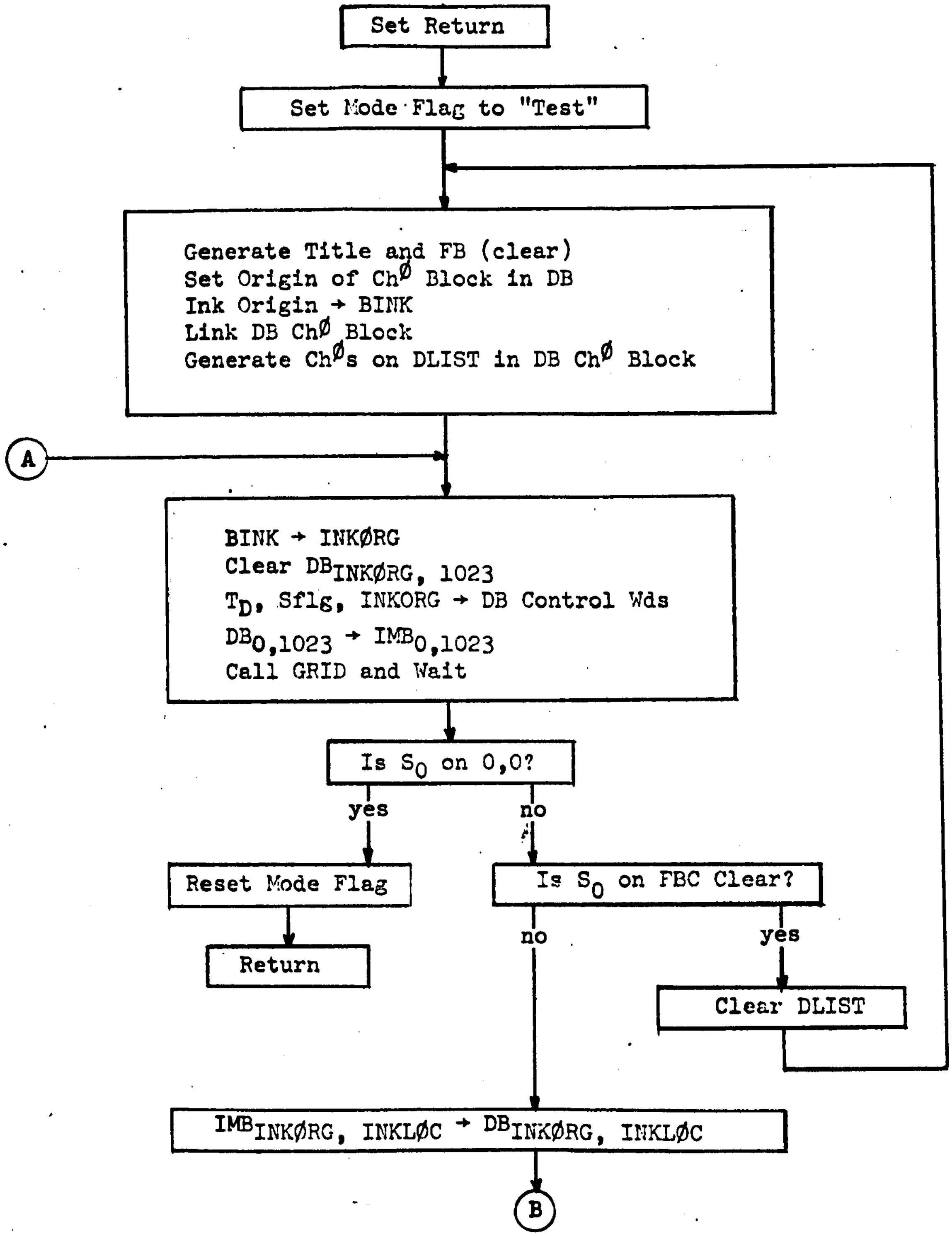

Set Return
Set Mode Flag to "Test"
Generate Title and FB (clear)
Set Origin of ChØ Block in DB
Ink Origin → BINK
Link DB ChØ Block
Generate ChØs on DLIST in DB ChØ Block
A
BINK → INKØRG
Clear $DB_{INKØRG, 1023}$
T_D, Sflg, INKORG → DB Control Wds
$DB_{0,1023}$ → $IMB_{0,1023}$
Call GRID and Wait
Is S_0 on 0,0?
yes
no
Reset Mode Flag
Return
Is S_0 on FBC Clear?
no
yes
Clear DLIST
$IMB_{INKØRG, INKLØC}$ → $DB_{INKØRG, INKLØC}$
B

Figure 7a. TEST Subroutine Flow Chart

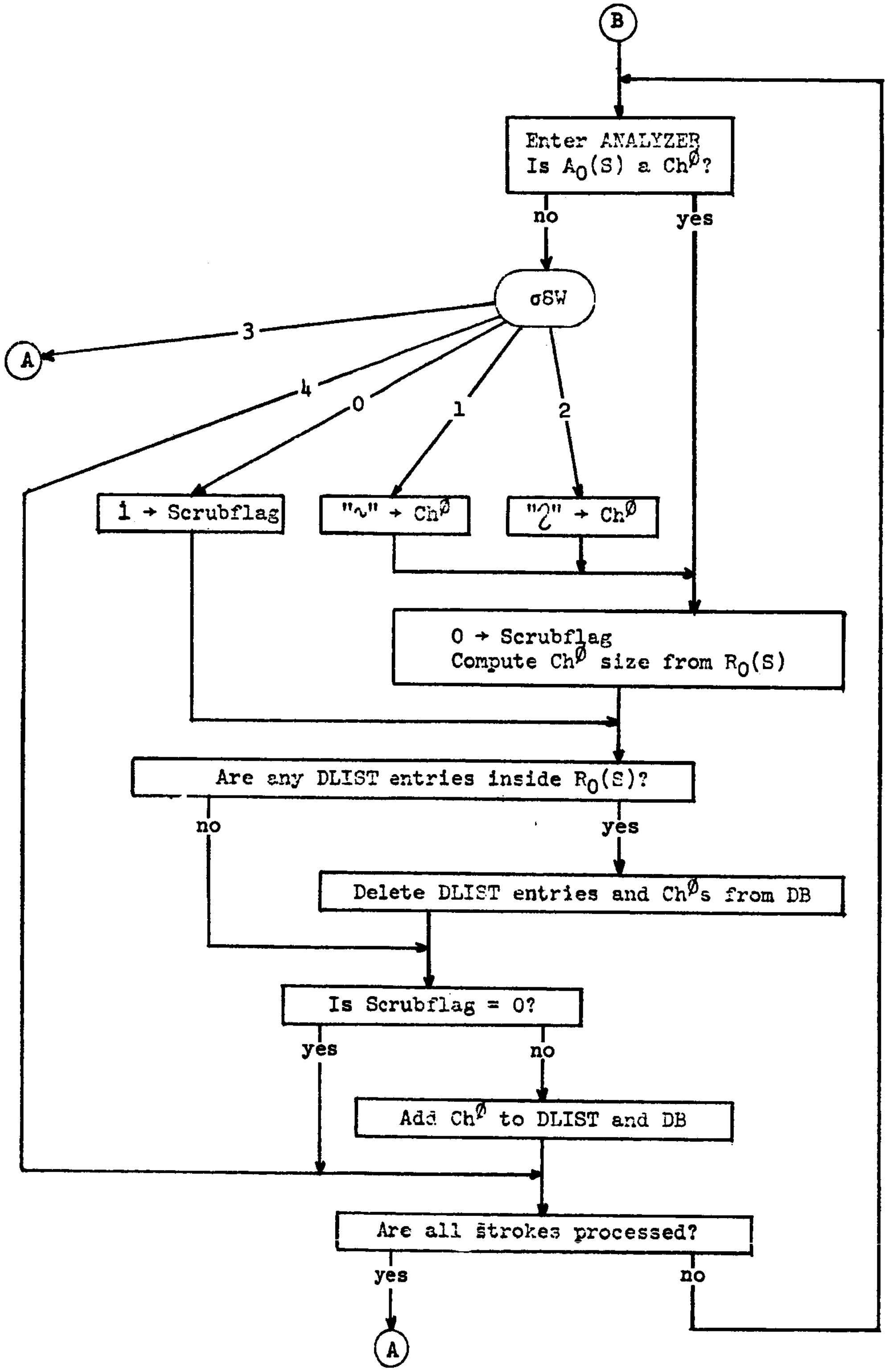

B
Enter ANALYZER
Is $A_0(S)$ a $Ch^{\emptyset}$?
no
yes
σSW
3
A
4
0
1
2
1 → Scrubflag
"∿" → $Ch^{\emptyset}$
"ʔ" → $Ch^{\emptyset}$
0 → Scrubflag
Compute $Ch^{\emptyset}$ size from $R_0(S)$
Are any DLIST entries inside $R_0(S)$?
no
yes
Delete DLIST entries and $Ch^{\emptyset}$s from DB
Is Scrubflag = 0?
yes
no
Add $Ch^{\emptyset}$ to DLIST and DB
Are all strokes processed?
yes
no
A

Figure 7b. TEST Subroutine Flow Chart (continued)

Each element contains the following:

1. Location of the lower left-hand corner of the output character.
2. Size ($R(Ch^I)$). (Size ($R(Ch^Ø)$) may be preferable.
3. A pointer into the Display Buffer to the origin of the points forming the output character.
4. The output character code.
5. A link to the next item (0 indicates the end of the Display List).

The limited ink space in the Display Buffer (actually the IM Buffer) allows input of 6 to 8 small characters, 3 or 4 medium-sized characters, 1 or 2 large characters, and 1 or less very large characters. (This limitation or extra-large characters is a problem.)

When an output is passed to TEST from ANALYZER, the display list is searched to see if any characters are to be deleted. The test is performed by computing the center of the character on the DLIST and comparing to see if it lies within the minimum rectangle surrounding the input, $R(Ch^I)$. All characters for which this is true are deleted from the DLIST and DB.

Although the program currently outputs one of two special characters on the display ("ʔ " or "~") when an unknown character occurs, it may actually be preferable to do nothing, that is, to ignore the input. For testing and debugging purposes, though, these special characters have been of value.

5.4 ANALYZER SUBROUTINE

ANALYZER is--in reality--two programs: one is coupled to SAMPLE and the other to TEST. In fact, each part could have been included as part of those routines. The two parts are clearly indicated in Figures 8 and 9.

For the part associated with SAMPLE, the stroke flag is set to "no more" and the mode flag is not set to "TEST." The assumption is that all of the strokes that have been input belong to a single character. The program processes each stroke until the end of inputs (σ_3) occurs, or until an invalid stroke is detected. An invalid stroke is defined (in STROKE) as one in which the sum of the absolute values of the differences between two adjacent points (xy_i, xy_{i+1}) is greater than or equal to 30. The program builds a table of outputs for the input strokes consisting of the feature string for the stroke (F_m and E_m), the geometric relation (G_m), the dictionary definition and link if any ($A_m(S)$ and D_m), the minimum rectangle surrounding the stroke (R_m), plus the composite rectangle surrounding all of the strokes in R_O and the number of strokes in the character (N_s).

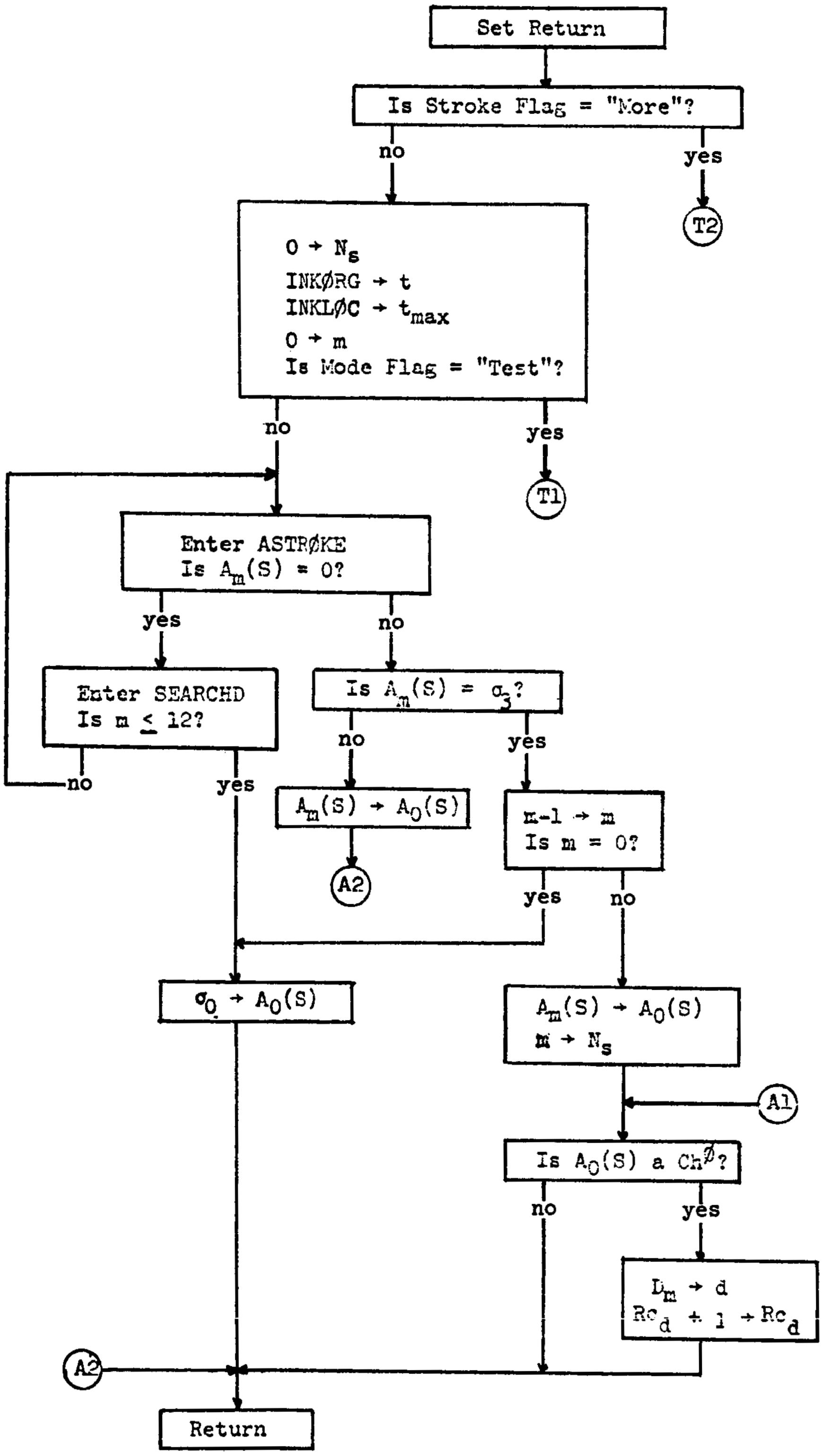


Figure 8. ANALYZER Subroutine (SAMPLE Mode) Flow Chart

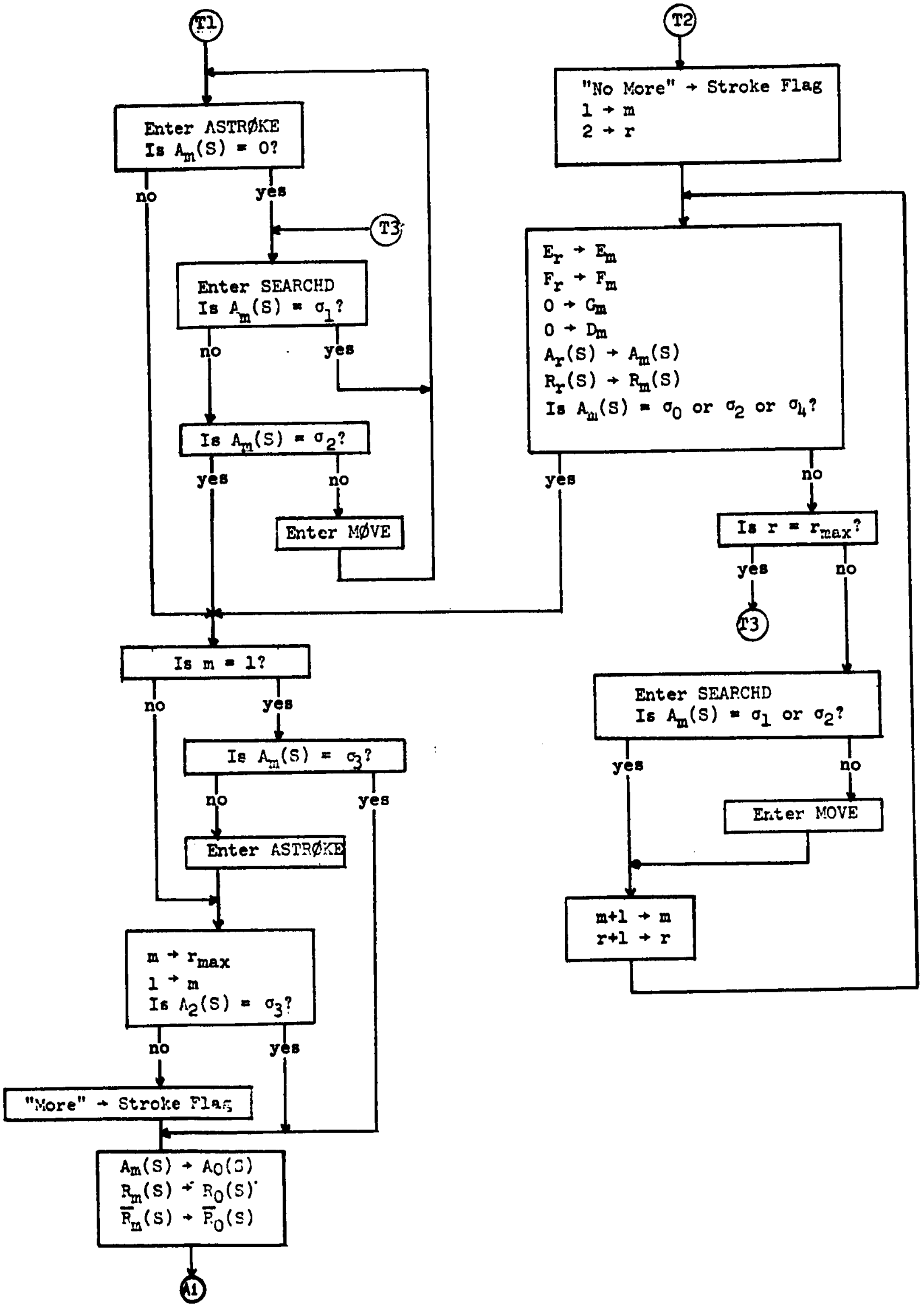


Figure 9. ANALYZER Subroutine (TEST Mode) Flow Chart

The control for the TEST portion of ANALYZER is much more complex. It can only determine when an output character is ready for output from the result of the dictionary search. Thus, it must be able to back up to the last legitimate character that it found, in some cases. It returns control to TEST each time it has an output ready, which may be a legitimate character (a $Ch^{\emptyset}$), and intermediate but undefined node in the dictionary (σ_1, output as "?"), an undefined input (σ_2, output as "~") or a scrub stroke (σ_0, too many features), interpreted by TEST as an erasure, or the end of input (σ_3). An invalid stroke (σ_4) is ignored. All of the pertinent data for TEST is placed in the first entry of all indexed lists, $A_o(s)$, $R_o(s)$, and $\bar{R}_o(s)$.

5.5 ASTRØKE AND STROKE SUBROUTINES

ASTRØKE is merely a mechanism by which STRØKE can be changed, modified, or replaced without changing ANALYZER (see Figure 10.) It clears the current items in the lists indexed by m (having first incremented m) and decides whether or not to call the feature-extraction routine, in this case, SEG2. If the system had various feature extractors for different alphabets (and thus different dictionaries), the dictionary would be queried here as to what feature extractor to call.

STRØKE does three things (see Figures 11a and 11b). First, it smooths the data. It has become apparent only since we have had the PDP-1 smoothing routine working that post-filter smoothing is of value, particularly in making corner detection better and in giving a smoother path so that a simpler (and faster) feature extractor can be used. The smoothing algorithm is a simple one: $p_{\text{smoothed}} = \dfrac{C_1 + C_2P_2}{C_1 + C_2}$, where p is the point xy and C (denoted as P_c in the GRID program) is the count associated with it.

Second, STRØKE re-filters the points so that none are closer than two raster units, and computes the heading (h) (one of thirty-two) from point to point and the heading differences (Δh) for use by the corner detector and feature extractor. It is here that invalid strokes are discovered.

Third, STRØKE computes the minimum rectangle surrounding the stroke R(s), the center of that rectangle $\bar{R}(s)$, and the Size (S) and average point count ($\bar{C}$) for use by the corner detector. We have recently decided to modify the $\bar{C}$ computation by eliminating high-count (slow-path) points from the average, in order to get a more consistent $\bar{C}$.

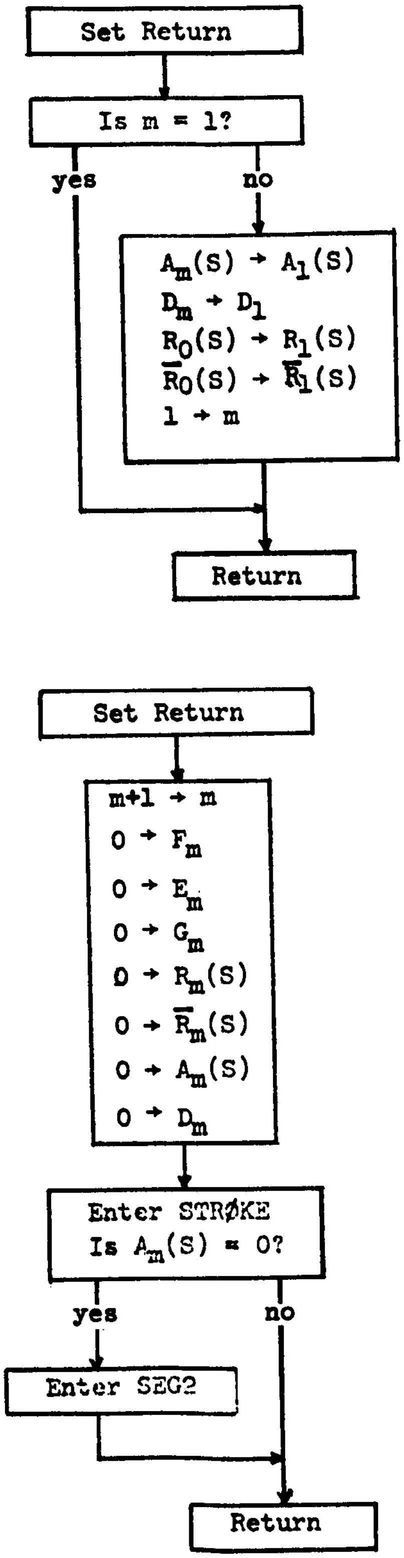


Figure 10. ASTROKE Subroutine Flow Chart

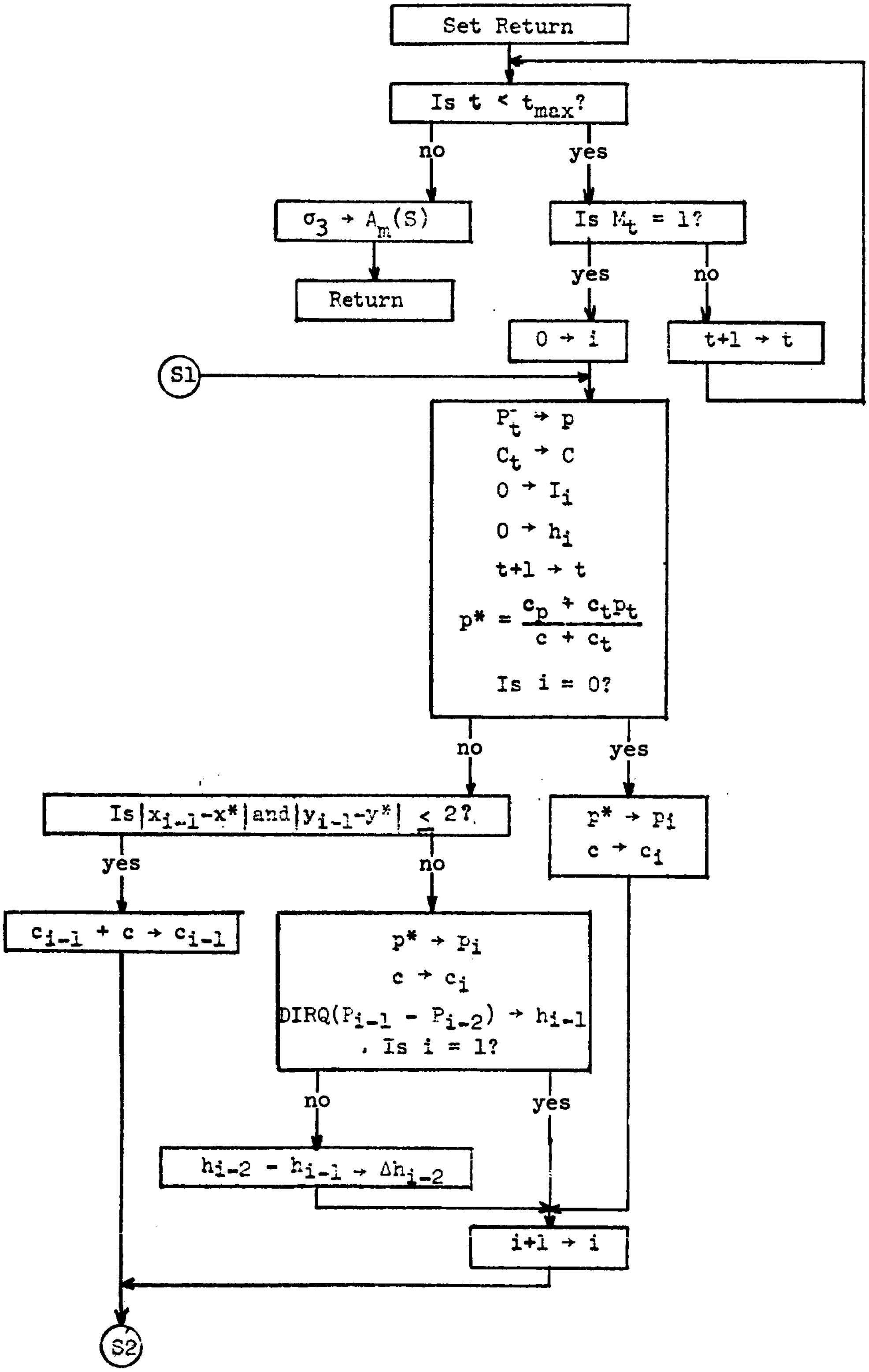


Figure 11a. STROKE Subroutine Flow Chart

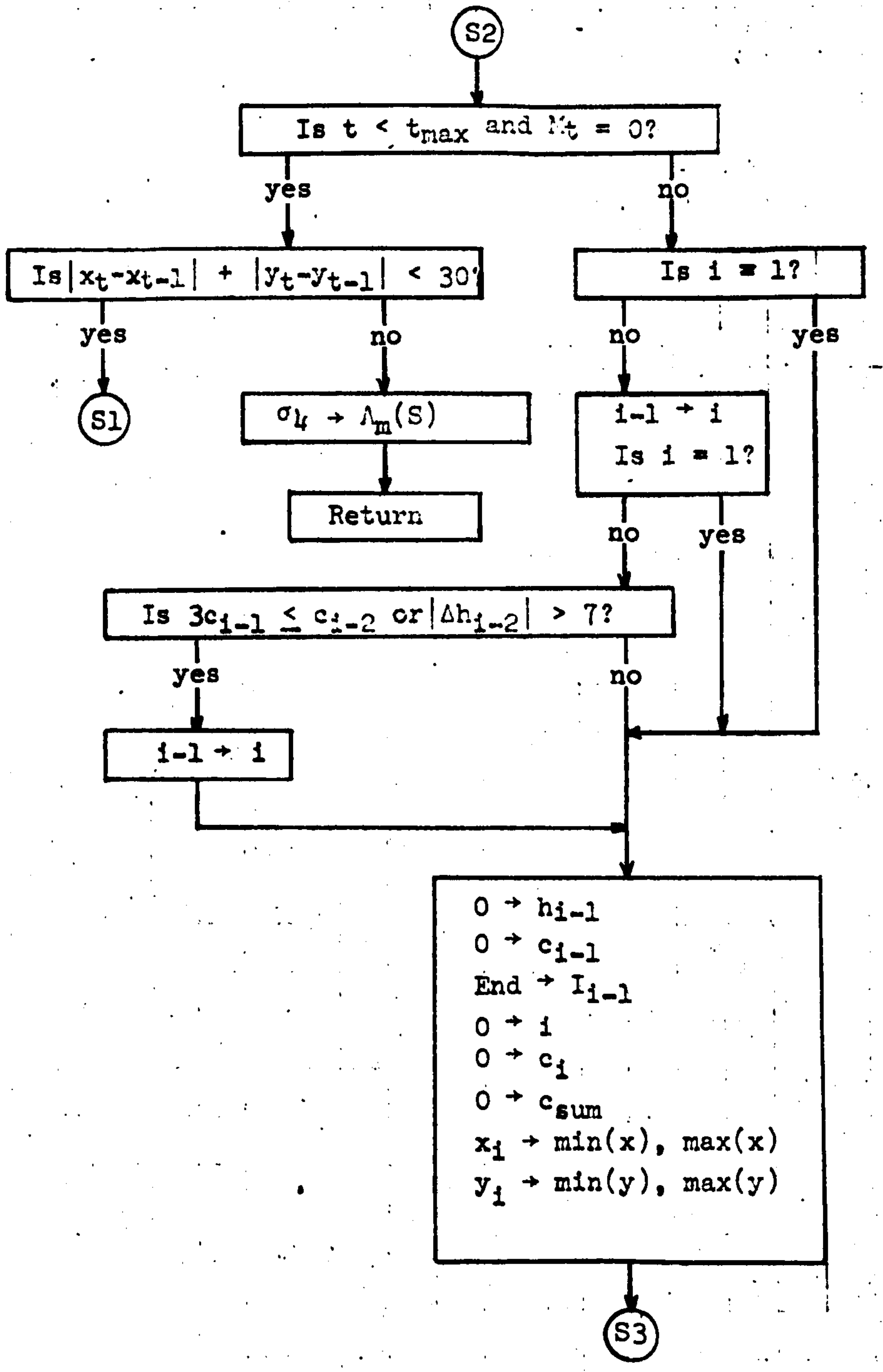

S2
Is $t < t_{max}$ and $M_t = 0$?
yes
no
Is $|x_t - x_{t-1}| + |y_t - y_{t-1}| < 30$?
Is $i = 1$?
yes
no
no
yes
S1
$\sigma_4 \rightarrow \Lambda_m(S)$
$i-1 \rightarrow i$
Is $i = 1$?
Return
no
yes
Is $3c_{i-1} \leq c_{i-2}$ or $|\Delta h_{i-2}| > 7$?
yes
no
$i-1 \rightarrow i$
$0 \rightarrow h_{i-1}$
$0 \rightarrow c_{i-1}$
End $\rightarrow I_{i-1}$
$0 \rightarrow i$
$0 \rightarrow c_i$
$0 \rightarrow c_{sum}$
$x_i \rightarrow \min(x), \max(x)$
$y_i \rightarrow \min(y), \max(y)$
S3

Figure 11b. STROKE Subroutine Flow Chart (continued)

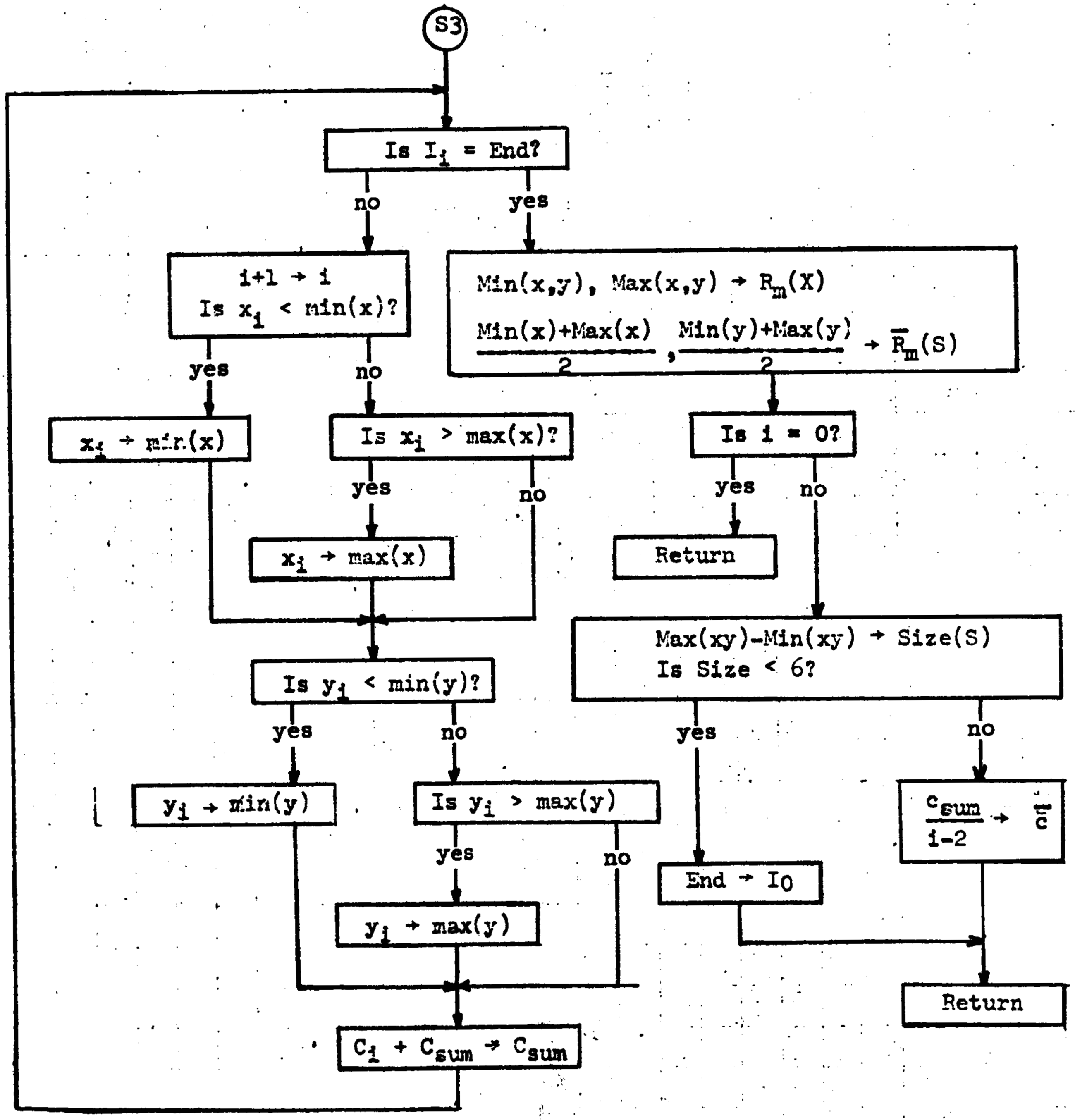


Figure 11c. STROKE Subroutine Flow Chart (continued)

Before exiting, Size (S) is tested to see that it is large enough to be something other than a spot 6x6 raster units or less, which we take to be a period.

The output from STRØKE is the table P_i, C_i, Δh_i, I_i, and the quantities Rm(s), $\overline{R}_m$(s), $Size_m$(S), and $\overline{C}_m$(S).

5.6 SEG2 SUBROUTINE

SEG2 is the heart of the program; it performs the feature extraction (see Figures 12a, 12b and 12c). This routine has been changed most often, and will continue to be improved. Among currently proposed changes is the elimination of the "loop" feature ($| \Sigma | > 24$), because of the problem of discrimination between some characters formed from a single loop. The test of whether j=0, dcode≠0, and fcode = 3 or 5 is an attempt to resolve this by adding (as an additional feature, if this test is true) the heading from the beginning of the stroke to a point ¼ of the way along the path.

The most complex part of the routine is the location and identification of an inflection point. This could be done in other ways, but less efficiently.

This routine computes the minimum rectangle surrounding each feature and the collected rectangle of the predecessor features; it then computes the center-to-center direction for those that are not coincident. No features in a single stroke can be "far" from the collected rectangle of its predecessors. The value of 48 for determining a maximum sum--and thus the limit of a loop--is an arbitrary one, as are the values used for determining the bounds of the other features (line, curve and cup).

The DIRF subroutine, which is called by SEG2, computes the value of dcode. (See Figure 12b.)

The total feature, made up of fcode, dcode and ccode, is appended to the right end of the feature string, but this is not of significance except where another procedure would create an ambiguity or other difficulty.

5.7 SEARCHD SUBROUTINE

SEARCHD is the program that searches the definition dictionary (DICT) for each stroke as it is presented by ANALYZER (see Figure 14). It returns control to the calling program after replacing A_m(S) with the contents of the definition, and replacing D_m with the definition location in the dictionary (if an exact match is found). Otherwise, it returns with a location of 0 in D_m and σ_2 in A_m(S).

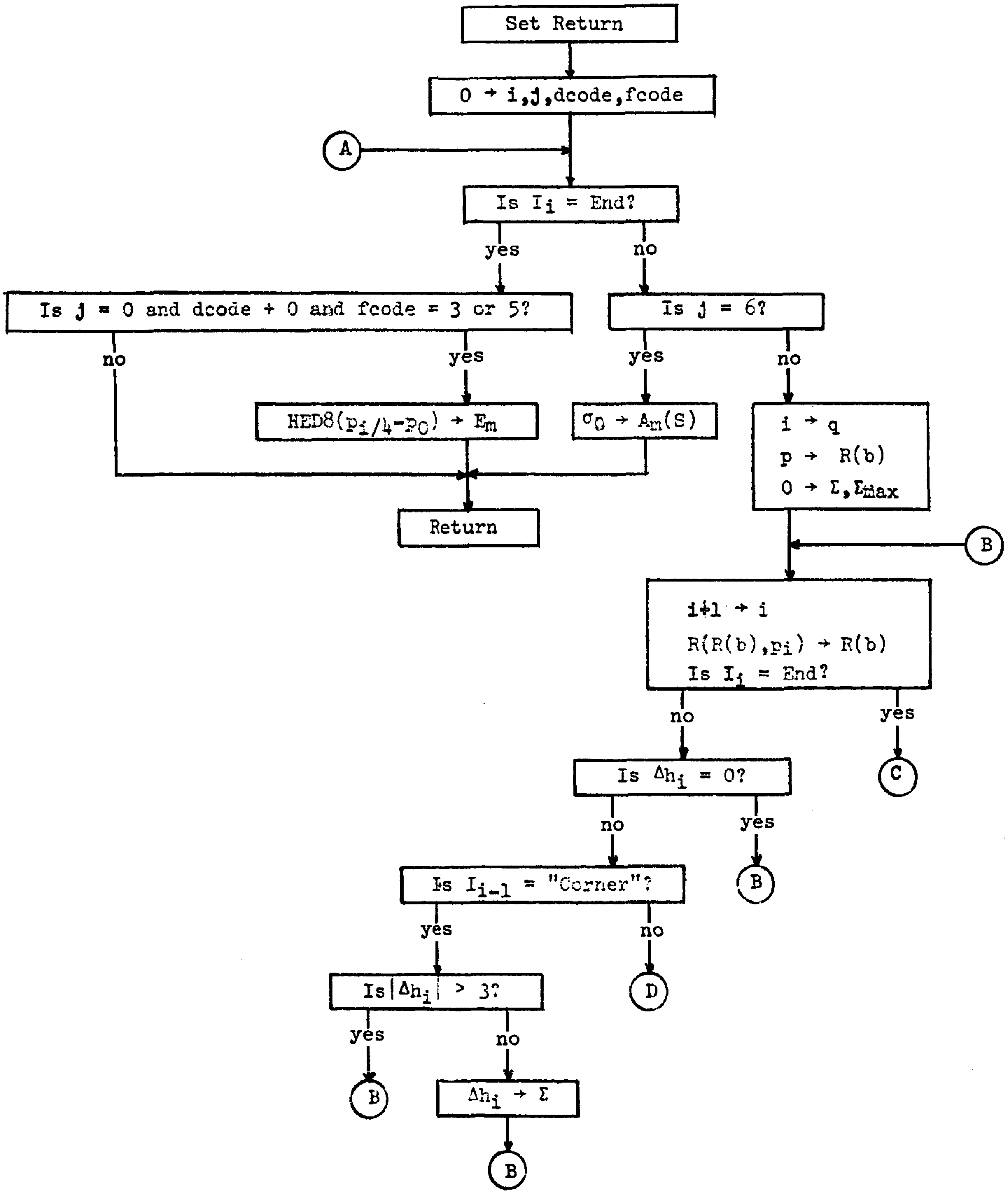

Set Return
$0 \rightarrow i, j, dcode, fcode$
A
Is I_i = End?
yes
no
Is $j = 0$ and dcode ≠ 0 and fcode = 3 or 5?
Is $j = 6$?
no
yes
yes
no
$HED8(p_i/4-p_0) \rightarrow E_m$
$\sigma_0 \rightarrow A_m(S)$
$i \rightarrow q$
$p \rightarrow R(b)$
$0 \rightarrow \Sigma, \Sigma_{max}$
Return
B
$i+1 \rightarrow i$
$R(R(b), p_i) \rightarrow R(b)$
Is I_i = End?
no
yes
Is $\Delta h_i = 0$?
C
no
yes
Is I_{i-1} = "Corner"?
B
yes
no
Is $|\Delta h_i| > 3$?
D
yes
no
B
$\Delta h_i \rightarrow \Sigma$
B

Figure 12a. SEG2 Subroutine Flow Chart

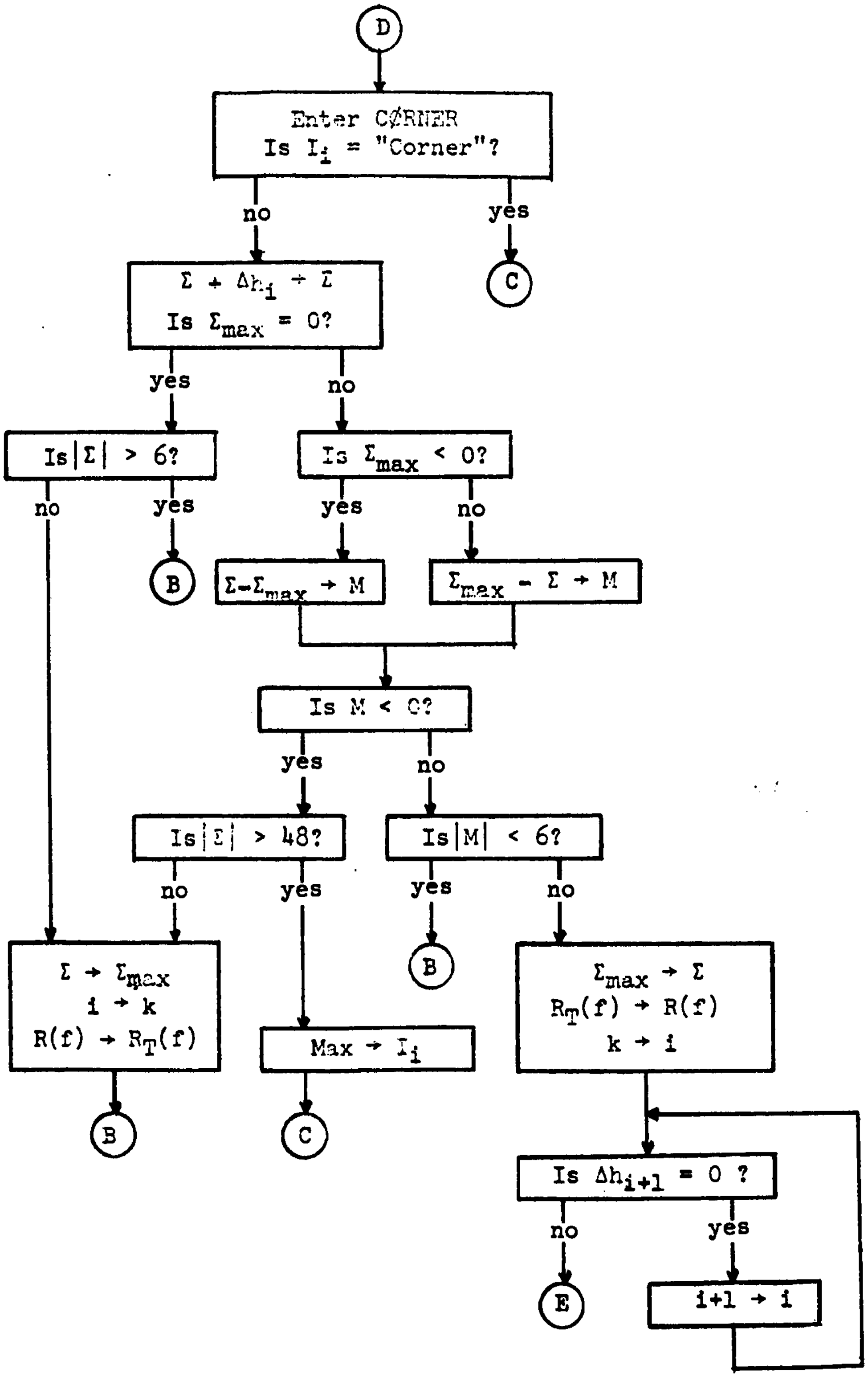


Figure 12b. SEG2 Subroutine Flow Chart (continued)

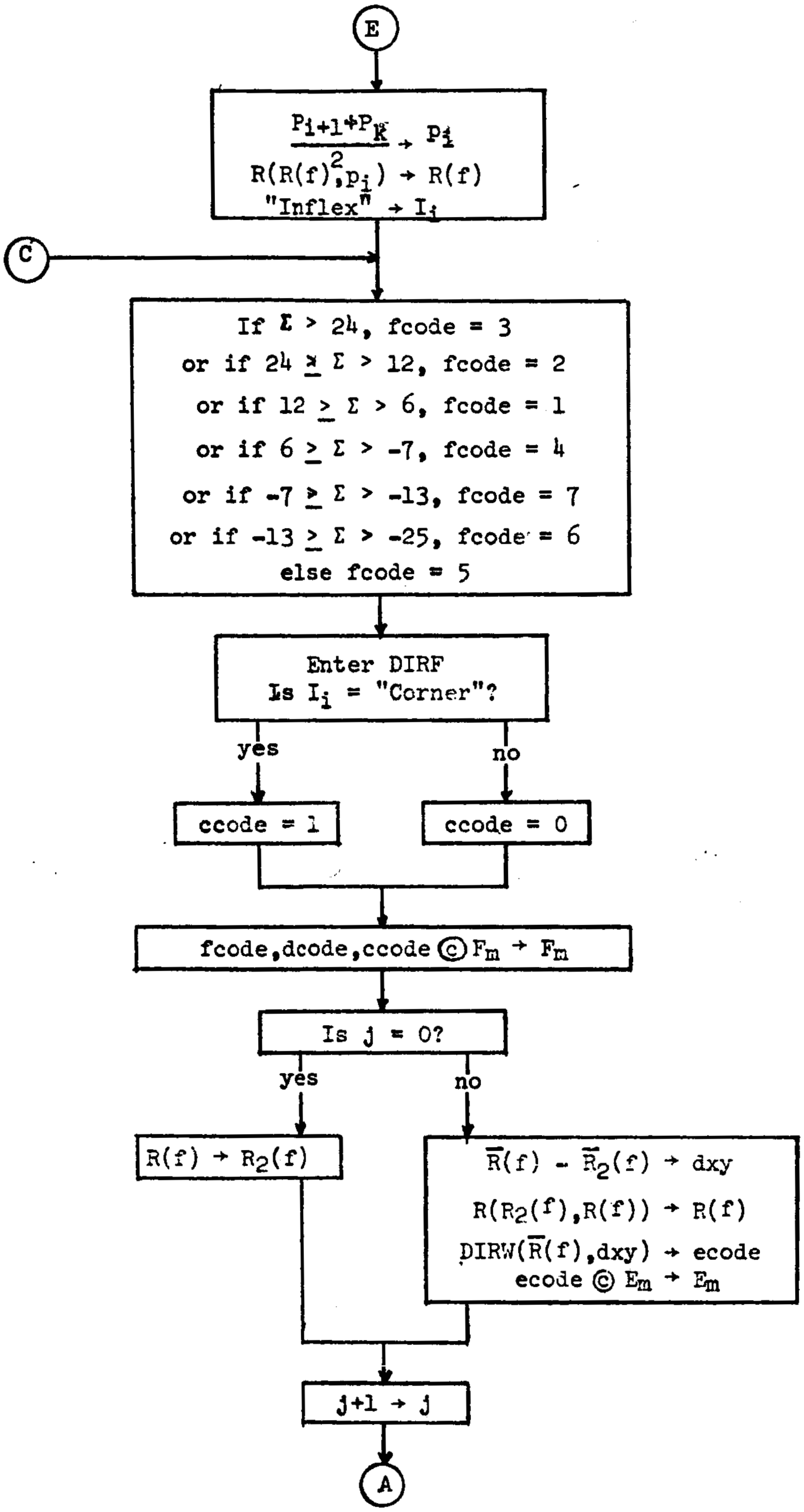

E
$\frac{p_{i+1}+p_K}{2} \rightarrow p_i$
$R(R(f),p_i) \rightarrow R(f)$
"Inflex" → I_i
C
If Σ > 24, fcode = 3
or if 24 ≥ Σ > 12, fcode = 2
or if 12 ≥ Σ > 6, fcode = 1
or if 6 ≥ Σ > -7, fcode = 4
or if -7 ≥ Σ > -13, fcode = 7
or if -13 ≥ Σ > -25, fcode = 6
else fcode = 5
Enter DIRF
Is I_i = "Corner"?
yes
no
ccode = 1
ccode = 0
fcode,dcode,ccode © F_m → F_m
Is j = 0?
yes
no
$R(f) \rightarrow R_2(f)$
$\overline{R}(f) - \overline{R}_2(f) \rightarrow dxy$
$R(R_2(f),R(f)) \rightarrow R(f)$
$DIRW(\overline{R}(f),dxy) \rightarrow ecode$
ecode © E_m → E_m
j+1 → j
A

Figure 12c. SEG2 Subroutine Flow Chart (continued)

Figure 13 shows the physical organization of the character dictionary. When the stroke to be matched is a successor stroke, its geometric relationship to its predecessors if first computed in GREL (see Figure 14). The successor link (SLINK) of the previous stroke (D_m-1) is placed in the index d. If it is a zero, there is no successor. NLINK points to alternative strokes at the same level if the current one under examination doesn't match. First strokes do not use NLINK, but rather successive locations in memory (for the sake of speed of search). The search returns with a σ_2 if it does not find a match for the stroke in the appropriate list, or the location of the entry (D_m) and the contents of the definition field, either σ_1 or a $Ch^{\emptyset}$.

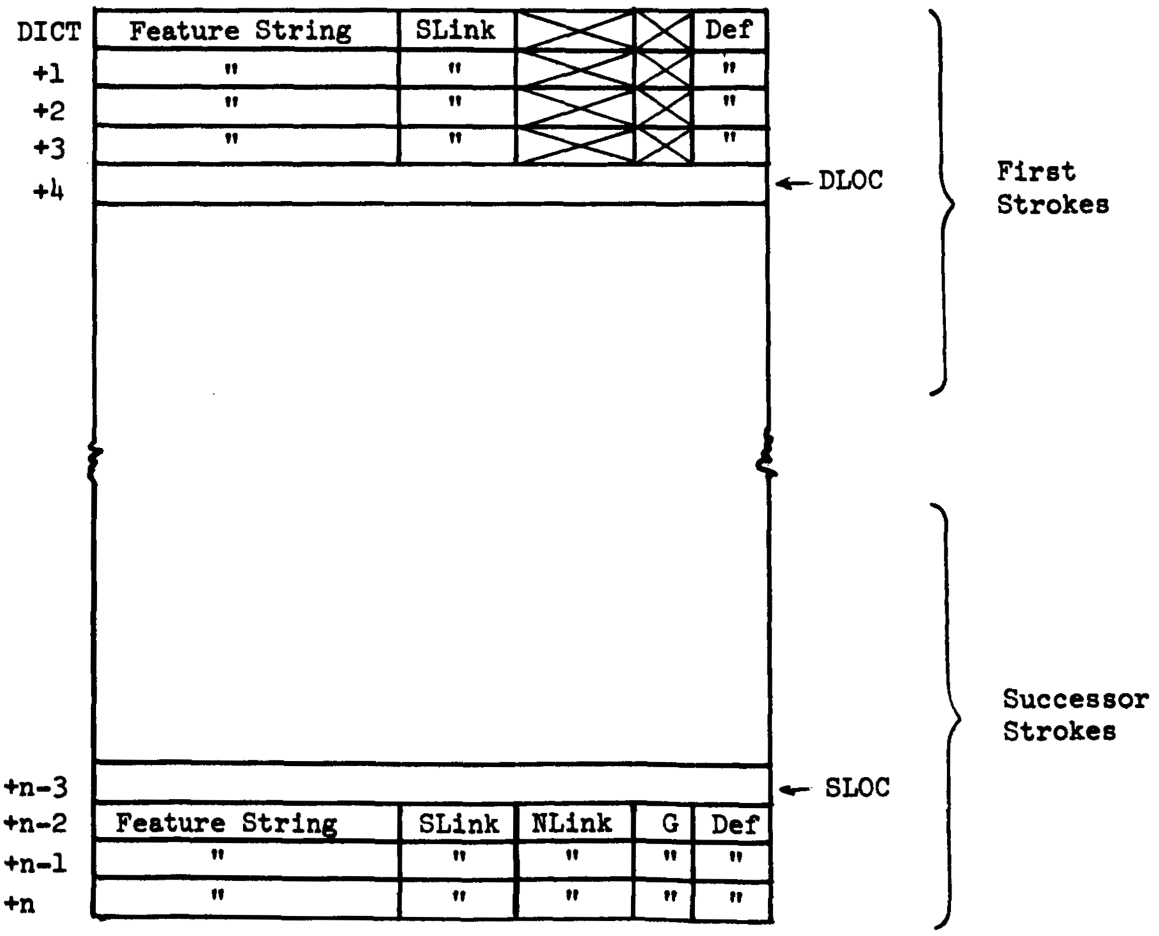


Figure 13. Character Dictionary Organization

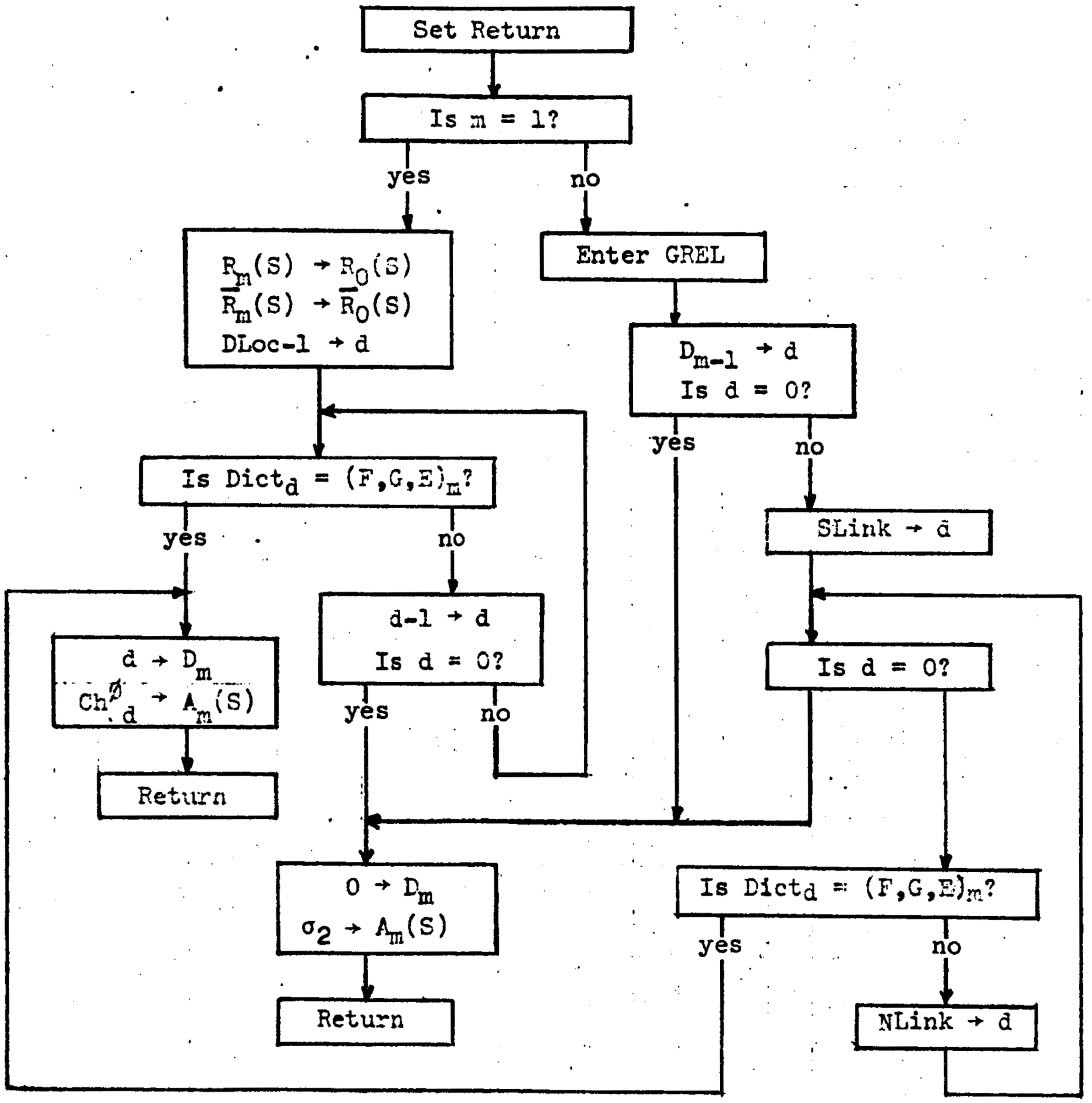

Set Return
Is $m = 1$?
yes
no
$R_m(S) \rightarrow R_0(S)$
$\overline{R}_m(S) \rightarrow \overline{R}_0(S)$
DLoc-1 → d
Enter GREL
$D_{m-1} \rightarrow d$
Is d = 0?
yes
no
Is $\mathrm{Dict}_d = (F,G,E)_m$?
yes
no
SLink → d
$d \rightarrow D_m$
$\mathrm{Ch}\text{Ø}_d \rightarrow A_m(S)$
d-1 → d
Is d = 0?
yes
no
Is d = 0?
Return
$0 \rightarrow D_m$
$\sigma_2 \rightarrow A_m(S)$
Is $\mathrm{Dict}_d = (F,G,E)_m$?
yes
no
Return
NLink → d

Figure 14. SEARCHD Subroutine Flow Chart

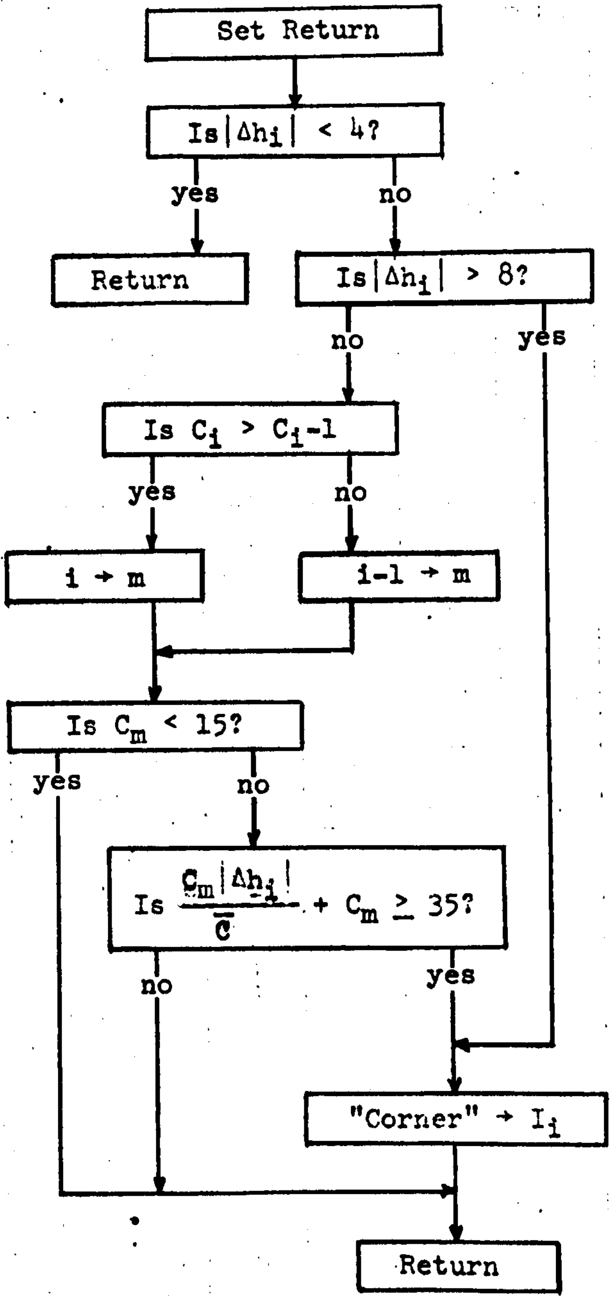


Figure 15. CORNER Subroutine Flow Chart

5.8 CORNER SUBROUTINE

The CORNER subroutine marks a corner if the local change of direction is greater than a right angle. A flow chart of this routine is given in Figure 15. Otherwise, the velocity is tested (using the best estimate we have, the counts). The test was arrived at empirically. In essence, it follows the rule: the sharper the turn, the faster the turn can be made (and be correctly detected).

This is but one version of CORNER, and works better than most we've tried. It will work with either post-filter smoothing alone or with both smoothing routines in operation. We are working on other versions of corner detection that we hope will prove even more effective.

5.9 GREL SUBROUTINE

GREL computes the relationship of two rectangles: the one surrounding the current stroke, and the one surrounding all of the predecessor strokes in this character. Figure 17 is a flow chart of this routine. GREL first tests for coincidence using DIRW. Overlap is tested by computing:

$$\Delta xy = \mathrm{Size}(R_o(S))-\mathrm{Size}(R_T(S))-\mathrm{Size}(R_m(S))$$

Figure 16 illustrates several examples of this computation. The value 16 has been assigned to proximate rectangles and 24 to those that are distant.

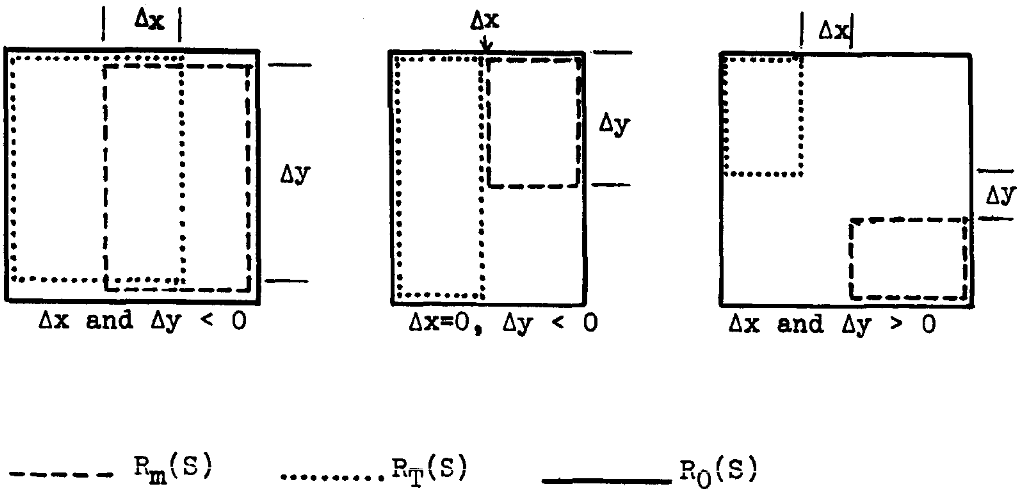


Figure 16. Examples of Overlap Computation

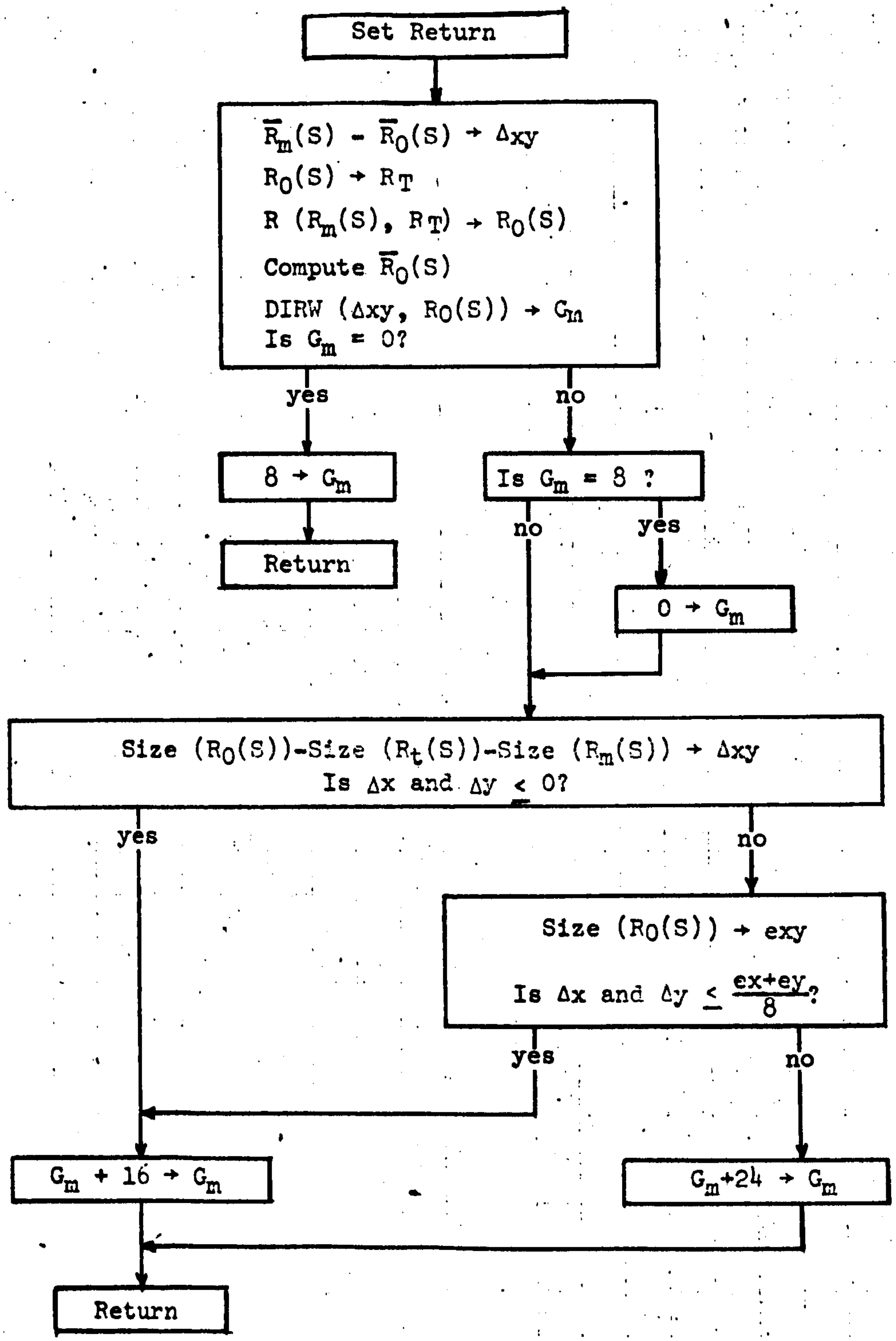

Set Return
$\overline{R}_m(S) - \overline{R}_0(S) \rightarrow \Delta xy$
$R_0(S) \rightarrow R_T$
$R\ (R_m(S),\ R_T) \rightarrow R_0(S)$
Compute $\overline{R}_0(S)$
DIRW $(\Delta xy,\ R_0(S)) \rightarrow G_m$
Is $G_m = 0$?
yes
no
$8 \rightarrow G_m$
Return
Is $G_m = 8$?
no
yes
$0 \rightarrow G_m$
Size $(R_0(S))$-Size $(R_t(S))$-Size $(R_m(S)) \rightarrow \Delta xy$
Is Δx and $\Delta y \leq 0$?
yes
no
Size $(R_0(S)) \rightarrow exy$
Is Δx and $\Delta y \leq \frac{ex+ey}{8}$?
yes
no
$G_m + 16 \rightarrow G_m$
$G_m + 24 \rightarrow G_m$
Return

Figure 17. GREL Subroutine Flow Chart

5.10 HED8, DIRQ, DIRW, AND DIRF SUBROUTINES

HED8 and DIRQ are simply a pair of table-driven direction assignment routines and should be self-explanatory (see Figures 18 and 19). DIRW is used by GREL to compute the value assigned to the geometric relationship between two rectangles (see Figure 20).

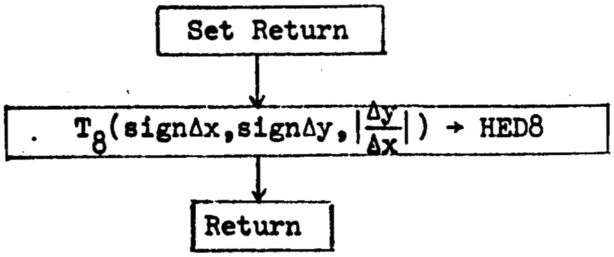
DIRF is used by the feature extractor to generate dcode, the relationship between the beginning and end points of each feature (see Figure 21). Special consideration is given to features whose aspect ratio is less than .25 or greater than 4, and to features that begin or end on an inflection point. DIRF determines if the feature began and ended in the same quadrant of the rectangle surrounding the feature. If it did not, one of eight values is assigned. On the other hand, if the end points are in the same quadrant but do not meet the closure test,

$$ex^2 + ey^2 \leq \frac{(\Delta x+\Delta y)^2}{8}$$

then HED8 is used to compute the assigned value. This method has proven far more consistent than using HED8 alone, because HED8 is responsive to small changes in aspect ratio, whereas DIRF is not.

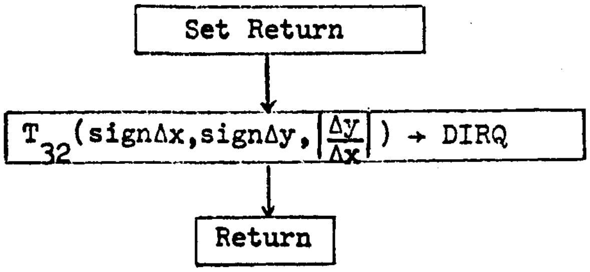
5.11 DEFINE SUBROUTINE

DEFINE adds definitions to the dictionary. The DEFINE routine starts with the last stroke of the input, inserting only those strokes that were not found in the dictionary by SEARCHD. If all of the required strokes for the definition cannot be added because of insufficient space, none are added to the dictionary, thus eliminating the problem of the dangling definition. (See Figure 22 for a flow chart of this subroutine.)


Set Return
$T_8(\text{sign}\Delta x, \text{sign}\Delta y, |\frac{\Delta y}{\Delta x}|) \rightarrow$ HED8
Return

T_8	$\Delta x>0$ $\Delta y>0$	$\Delta x\leq 0$ $\Delta y>0$	$\Delta x>0$ $\Delta y\leq 0$	$\Delta x\leq 0$ $\Delta y\leq 0$
$\infty\geq\left\|\frac{\Delta y}{\Delta x}\right\|\geq\sqrt{2}+1$	8	8	4	4
$\sqrt{2}+1>\left\|\frac{\Delta y}{\Delta x}\right\|\geq\sqrt{2}-1$	1	7	3	5
$\sqrt{2}-1>\left\|\frac{\Delta y}{\Delta x}\right\|\geq 0$	2	6	2	6

Figure 18. HED8 (Δxy) Subroutine Flow Chart and Table


Set Return
T_{32}(signΔx,signΔy,$\left|\frac{\Delta y}{\Delta x}\right|$) → DIRQ
Return

T_{32}	$\Delta x > 0$ $\Delta y > 0$	$\Delta x \leq 0$ $\Delta y > 0$	$\Delta x > 0$ $\Delta y \leq 0$	$\Delta x \leq 0$ $\Delta y \leq 0$
$\infty \geq \left\|\frac{\Delta y}{\Delta x}\right\| \geq 10.175$	0	0	-16	-16
$10.175 > \left\|\frac{\Delta y}{\Delta x}\right\| \geq 3.287$	1	-1	15	-15
$3.287 > \left\|\frac{\Delta y}{\Delta x}\right\| \geq 1.871$	2	-2	14	-14
$1.871 > \left\|\frac{\Delta y}{\Delta x}\right\| \geq 1.219$	3	-3	13	-13
$1.219 > \left\|\frac{\Delta y}{\Delta x}\right\| \geq 0.821$	4	-4	12	-12
$0.821 > \left\|\frac{\Delta y}{\Delta x}\right\| \geq 0.534$	5	-5	11	-11
$0.534 > \left\|\frac{\Delta y}{\Delta x}\right\| \geq 0.303$	6	-6	10	-10
$0.303 > \left\|\frac{\Delta y}{\Delta x}\right\| \geq 0.098$	7	-7	9	-9
$0.098 > \left\|\frac{\Delta y}{\Delta x}\right\| \geq 0.0$	8	-8	8	-8

Figure 19. DIRQ (Δxy) Subroutine Flow Chart and Table

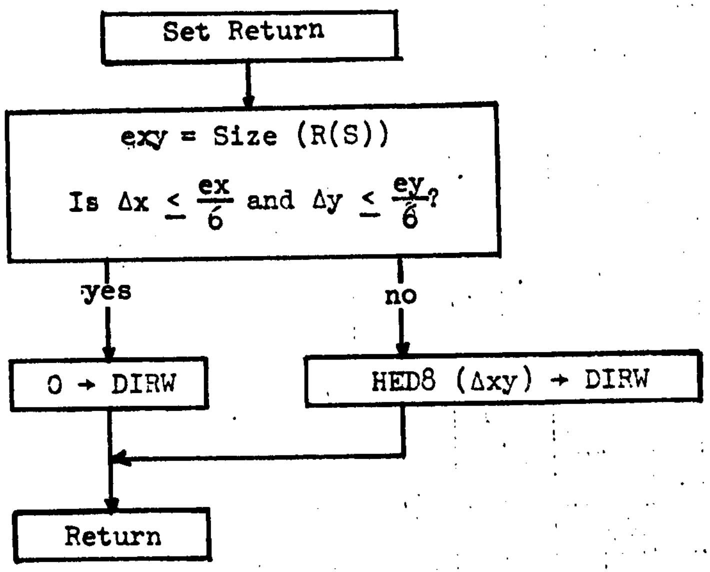
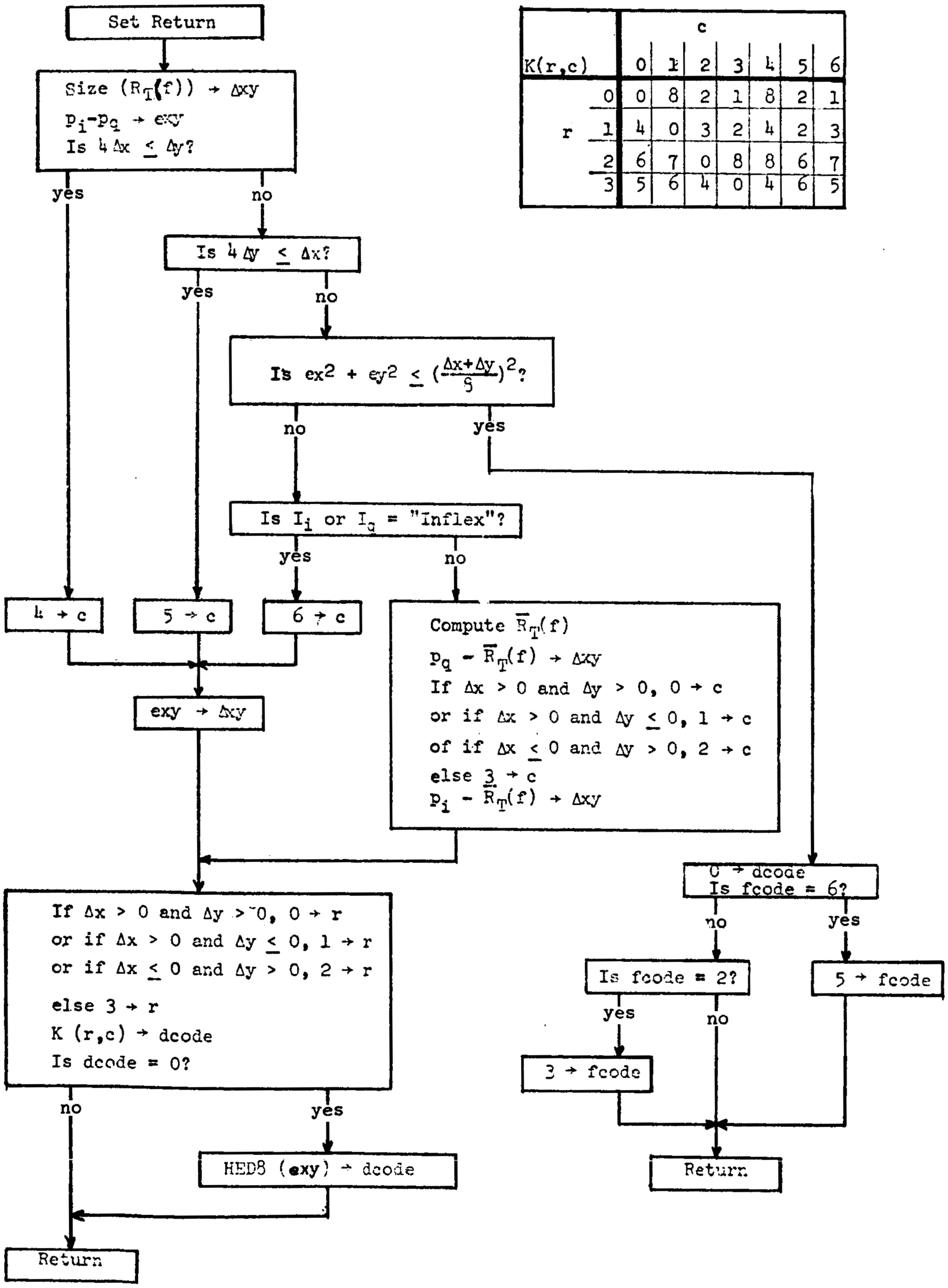


Figure 20. DIRW (Δxy, R(S)) Subroutine Flow Chart


Set Return
Size (R_T(f)) → Δxy
p_i-p_q → exy
Is 4 Δx ≤ Δy?
yes
no
Is 4 Δy ≤ Δx?
yes
no
Is ex2 + ey2 ≤ ((Δx+Δy)/8)2?
no
yes
Is I_i or I_q = "Inflex"?
yes
no
4 → c
5 → c
6 → c
Compute $\overline{R}_T$(f)
p_q - $\overline{R}_T$(f) → Δxy
If Δx > 0 and Δy > 0, 0 → c
or if Δx > 0 and Δy ≤ 0, 1 → c
of if Δx ≤ 0 and Δy > 0, 2 → c
else 3 → c
p_i - $\overline{R}_T$(f) → Δxy
exy → Δxy
If Δx > 0 and Δy > 0, 0 → r
or if Δx > 0 and Δy ≤ 0, 1 → r
or if Δx ≤ 0 and Δy > 0, 2 → r
else 3 → r
K (r,c) → dcode
Is dcode = 0?
no
yes
HED8 (exy) → dcode
Return
0 → dcode
Is fcode = 6?
no
yes
Is fcode = 2?
5 → fcode
yes
no
3 → fcode
Return

K(r,c)		c						
		0	1	2	3	4	5	6
r	0	0	8	2	1	8	2	1
	1	4	0	3	2	4	2	3
	2	6	7	0	8	8	6	7
	3	5	6	4	0	4	6	5

Figure 21. DIRF Subroutine Flow Chart and Table

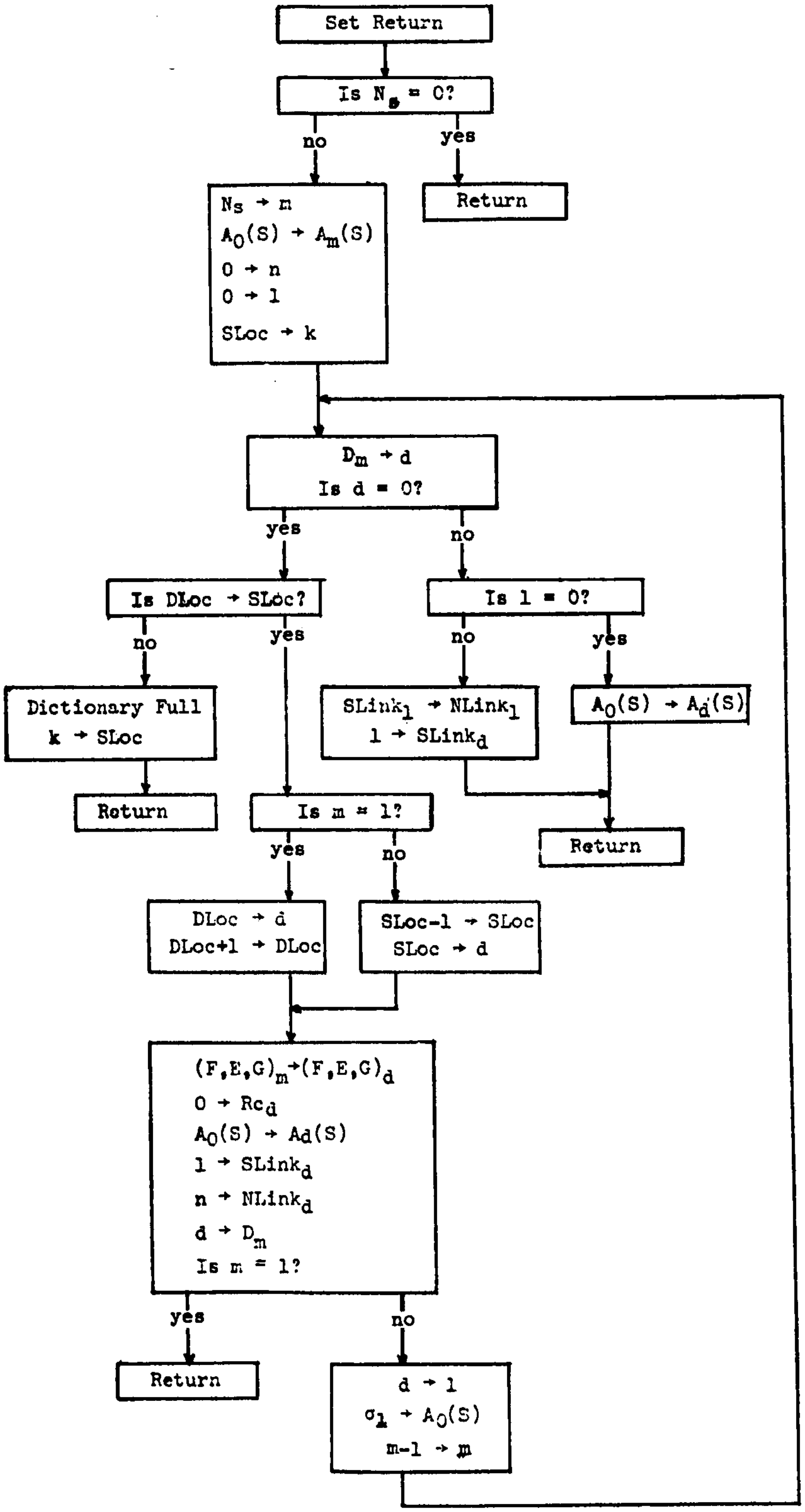


Figure 22. DEFINE Subroutine Flow Chart

APPENDIX A: GLOSSARY OF MNEMONICS AND ABBREVIATIONS

TERM	MEANING	REFERENCE
d, i, j, l, m, g, r, t	These letters designate indexes that are used in the usual programming sense of iteration counts. They are local (in the ALGOL sense) in that they are available to called subroutines but not to calling subroutines.	
psw	The mechanical switch in the tip of the RAND Tablet stylus. It has the values "on" and "off."	GRID
σSW	The program switch associated with returns from ANALYZER that are not $Ch^{\emptyset}$. It has the values "0", "1", "2", "3", and "4" that are the subscript values of the five meanings of σ.	SAMPLE, TEST
SW1	The program switch in GRID that is set to the appropriate function according to the value of psw and T_d, the time delay. It has the value IGN, meaning "ignore the tablet"; TB, meaning "post only the current position of the pen in IMB at TB"; and INK, meaning "filter and post the path of the pen on the tablet as long as psw is on and there is room in the buffer".	GRID
SW2	The program switch in GRID that enables or disables smoothing. It has the values of "on" and "off" according to SMFLG.	GRID

TERM	MEANING	REFERENCE
CHSW	The program switch in SAMPLE that controls program flow and the meaning of user actions at the tablet. CHSW may be set to "D" meaning "a defined character has been output or a new definition added to the dictionary"; N, meaning "there is no input" (this is the initial condition and the condition after a "clear"); "U" meaning "there is an undefined input pending; and "R", meaning "the function button REDEFINE has been pushed".	SAMPLE
BINK	The relative location where "ink" may begin in IMB.	SAMPLE and TEST
ccode	A subpart of a feature.	SEG2
C	The rejected point count provided by GRID and referred to as P_c in GRID.	STRØKE
dcode	A subpart of a feature.	SEG2
D	The dictionary pointer associated with a stroke, indexed by m.	ANALYZER
DB	The Display Buffer image resident with the object program in Q-32 core.	SAMPLE, TEST
DICT	The name of the definition dictionary.	DEFINE, SEARCHD
DLØC	The pointer used with DICT for the next available location to store a first stroke.	DEFINE, SEARCHD
DLIST	The linked list of output characters in TEST containing size and position information and the relative location of the actual character in IMB or DB.	TEST

TERM	MEANING	REFERENCE
ecode	The computed geometric relationship for E.	SEG2
E	The string of geometric relationships between a feature, F, and its predecessors in a stroke, indexed by m.	ASTRØKE, SEG2, DEFINE, SEARCHD
fcode	The subpart of a feature.	SEG2
F	The string of features for a stroke, indexed by m.	ASTRØKE, SEG2, DEFINE, SEARCHD
FB	The abbreviation for Function Button.	SAMPLE, TEST
G	The geometric relationship between a stroke and its predecessors in a multi-stroke character, indexed by m.	ASTRØKE, SEG2, DEFINE, SEARCHD
IMB	The display buffer in Input Memory from which the display is refreshed.	GRID, SAMPLE, TEST
INKORG	The relative location that "ink" is to start in IMB.	SAMPLE
INKLØC	The current relative location in IMB for "ink".	GRID
I	The indicator or flag generated from the inputs by STRØKE.	STRØKE. SEG2
ILØC	A position on the display surface where an output is placed for undefined or illegal inputs.	SAMPLE
KB	The actual keyboard pushbuttons associated with the output character subsets.	SAMPLE
KCB	The set of pushbuttons that allow the user to select one of the keyboards from among the available set.	SAMPLE
M	A temporary item in SEG2.	SEG2

TERM	MEANING	REFERENCE
NLINK	The link to the next stroke at the same level in the list of successor strokes in the dictionary.	DEFINE, SEARCHD
N_s	The number of strokes either input or in a character.	GRID, ANALYZER
Pc	The count of points rejected by the filter. Referred to as C in STRØKE.	GRID
Rc	The recognition count kept with each definition in the dictionary.	ANALYZER
SMFLG	The flag set in IMB to enable or disable smoothing in GRID.	GRID, SAMPLE, TEST
SLØC	The pointer for the next available location for a successor stroke in the dictionary.	DEFINE, SEARCHD
SLINK	The pointer to the first legitimate successor stroke for multi-stroke characters in a dictionary definition.	DEFINE, SEARCHD
T_d	The time delay set in IMB by the calling program to specify when GRID shall give up control after a psw "off" is detected. Time is specified in units of .25 sec.	GRID, SAMPEL, TEST
T_d'	An intermediate storage for T_d.	GRID
T_d^*	The computed value for use by GRID to effect the time delay test based upon a clock that increments in units other than .25 sec.	GRID

APPENDIX B: NEW TECHNOLOGY

It is difficult to say what, in particular, about this program lies within the realm of new technology. Rather than specific algorithms or solutions to particular problems, it is the general approach that is unique--the combination of existing methodology that is new.

At the actual working level of the program, two things in particular are different from earlier approaches to character recognition. One is the feature-extraction technique, including corner-detection; the other is the use of the dictionary to provide separation between adjacent characters, instead of some other measure.