

SYSREC:
A COMPUTER PROGRAM TO RECORD
TRANSMISSION NETWORK STRUCTURES

by

BARRY HUGH GRAY
B.A., University of Victoria, 1968

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
Master of Science
in the Department
of
Computer Science

We accept this thesis as conforming to the
required standard

THE UNIVERSITY OF BRITISH COLUMBIA

April, 1971

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study.

I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the Head of my Department or by his representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Computer Science.

The University of British Columbia
Vancouver 8, Canada

Date May 28/71

ABSTRACT

The design of large-scale industrial or experimental facilities oftentimes involves the routing of complex networks of power supply lines, control lines or signals, heating or cooling ducts, product transmission pipes etc. In such cases it is essential to be able to maintain an up-to-date, easily accessible, global record of each cable, duct or signal path, the function each performs and its relation to like elements within the system. It is the purpose of this thesis to describe such a facility as implemented by a conversational computer program - SYSREC.

The facility discussed allows the user to build network configurations, perform minimum path routing of signals etc., delete connectors or signals and save the record of a network for further modification at some later date. In addition, it is possible to generate output summarizing, in detail, the status of a given network. The program is designed to perform the above operations interactively by means of commands inputted by the user.

The thesis is composed of two logically separate sections; a User's Manual describing the capabilities and usage of the program and a Logic Manual describing the internal logic of the program. The purpose of such a separation is to provide self contained references for two distinct levels of user: the general user and the user who wishes to extend or modify the existing implementation. Since both the User's Manual and the Logic Manual contain their own tables of contents, appendices etc., the thesis may be split into two physically distinct parts should the need arise.

ACKNOWLEDGEMENT

The author wishes to thank his advisors Mr. W. Dettwiler and Dr. R. Johnson for their support, guidance and encouragement during the preparation of this thesis. In addition, the author is thankful to fellow-student Greg Shannan for his many helpful hints regarding assembler language coding. Financial support from Dr. D. Seeley, the Department of Computer Science and TRIUMF is gratefully acknowledged. The typing of the final manuscript was done by Olwen Sutton to whom the author also extends his thanks.

SYSREC USER'S MANUAL

Barry H. Gray

Department of Computer Science

University of British Columbia

April, 1971

TABLE OF CONTENTS

	Page
Preface	I.1
Section 1 General Introduction	I.3
1.1 Purpose	I.3
1.2 Functions Performed	I.4
1.3 Limitations	I.5
1.4 General Remarks	I.5
Section 2 SYSREC Command Language	I.7
2.1 Concepts	I.7
2.2 Input Formats	I.7
2.3 Command Descriptions	I.9
2.3.1 Notation and Terminology	I.9
2.3.2 Commands	I.10
Section 3 Output	I.34
3.1 Output from SYSREC Commands	I.34
3.1.1 LIST output	I.34
3.1.2 SUMMARY output	I.34
3.1.3 TRACE output	I.37
3.2 Output from error messages	I.40
Section 4 Program Execution	I.41
4.1 General Startup/Termination	I.41
4.2 Interrupt Procedure	I.41
4.3 Startup requirements under MTS	I.42

Bibliography		I.44
Appendix A	Program Generated Messages	I.A.1
Appendix B	Minimum Path Algorithm and its Operation	I.B.1
Appendix C	SYSREC Command Prototypes	I.C.1

SYSREC USER'S MANUALPreface

This manual is a procedural guide to the use of the program SYSREC. It is directed towards users familiar with both transmission network designs and the use of time-shared computing facilities. The manual contains the following information:

- 1) program capabilities
- 2) program restrictions
- 3) available SYSREC Commands
- 4) initiation/termination procedures
- 5) input/output particulars
- 6) error message explanations

The manual is composed of four sections, the first three of which deal with the overall functional capabilities of SYSREC without reference (for the most part) to any particular operating environment.

These three sections are:

Section 1 - General Introduction

Section 2 - SYSREC Command Language

Section 3 - Obtainable Output

The fourth section discusses the procedures required to initiate execution of the program and, in particular, how to initiate execution under the control of the Michigan Terminal System (MTS).

The above four sections are complemented by three appendices:

Appendix A describes the meaning of all program-generated messages;

Appendix B describes the minimum path algorithm used by SYSREC in its

routing of pathways; Appendix C summarizes the prototypes of all

available SYSREC commands.

This manual does not describe usage of the MTS command language; thus persons wishing to use SYSREC with UBC-MTS and who are not familiar with MTS will be forced to learn the MTS command language.

Section 1: GENERAL INTRODUCTION

1.1 Purpose

The design of large-scale industrial or experimental facilities oftentimes involves the routine of complex networks* of power supply lines, control lines or signals, heating or cooling ducts, product transmission pipes, etc. In such cases it is essential to be able to maintain an up-to-date, easily accessible, global record of each cable, duct or signal path, the function each performs and its relation to like elements within the system. The purpose of SYSREC*† is to provide such a facility by means of a conversational computer program. The SYSREC program was designed following the requirements of the TRIUMF project, which in this light is concerned with recording status information of a network of electrical cables and control signals.† The program is, however, not limited to use in connection with electrical networks, and obvious generalizations should suffice to visualize the use of the program with other types of transmission networks.

SYSREC is intended for use:

- 1) in the design stages of a project as a 'scratchpad' permitting formulation and modifications of initial network designs
- 2) in the expansion stages of a project as a means of detailing the currently existing configuration and possible expansion points

* In what follows the terms network, configuration and system will be used interchangeably to denote any global organized structure of nodes and connecting pathways - in particular, structures in an electrical sense, i.e. terminal boards, connecting cables, signals.

*† SYStem RECOrd

† For this reason alone the manual is written in terms of electrical cabling networks.

- 3) as a bridge between design and implementation personnel by outputting detailed information about required interconnections and signal routes
- 4) in the maintenance of a project by keeping an up-to-date global record of a given network.

1.2 Functions Performed

SYSREC allows the users to perform the following functions:

- 1) define terminal boards by name and size
- 2) define cables by name and size and place same as connectors between terminal boards
- 3) define signals by name and route these signals automatically between terminal boards with automatic insertion of required jumper lines
- 4) remove signals from a cable line or disconnect a cable from terminal boards
- 5) extend existing signals to other parts of the network
- 6) list all current terminal boards, cables and signals
- 7) trace a signal throughout the network, examine a cable line or terminal board pin
- 8) summarize the current status of individual or all signals, cables or terminal boards within the network
- 9) save the current record for use by a subsequent run

The automatic signal routing algorithm ensures minimal length paths between specified terminal boards. When an operation contradictory to the physical constraints of the current network is attempted, or whenever the SYSREC input syntax has been violated, the program

generates an error message.

1.3 Limitations

SYSREC does not optimize network flows nor does it optimize physical routing of cables and signals except on minimum length criteria. Network configurations structured or recorded by SYSREC must conform to the following requirements:

- 1) a cable can be attached to exactly two terminal boards
- 2) cable lines are attached to consecutive terminal board pins
- 3) the dimension of the signal may not be split over two parallel cables: i.e. if a signal is conducted by a cable at any point, the complete dimension of the signal must run in that cable
- 4) a dimensioned signal occupies consecutive lines within a cable
- 5) jumpers between sets of pins on terminal boards are attached consecutively within each set (e.g. pins 10, 11, 12 may be jumpered to pins 34, 35, 36)
- 6) a single terminal board pin may have at most one cable line and two jumper lines attached to it
- 7) the maximum number of lines/cable or pins/terminal board is 32,767

1.4 General Remarks

The SYSREC program operates under the Michigan Terminal System at the University of British Columbia. SYSREC source language is 360/Assembler Language, which was chosen for the following reasons:

- 1) high execution speed - being a conversational program, SYSREC should operate with as little delay as possible

- 2) low operating cost - minimum core storage (currently 20,000 bytes excluding expandable storage vectors), maximum efficiency and execution speeds result from the use of Assembler Language

The program operates in both conversational and batch modes, the former normally being used. Batch mode is a means of generating quantities of output data.

Section 2: SYSREC COMMAND LANGUAGE

2.1 Concepts

SYSREC operates in response to a sequence of commands given by the user, which are entered either from a terminal or via a batch stream. When given a command, SYSREC interprets it, performs the requested operation, and generates an output statement indicating successful completion of the requested operation or the encounter of some error condition. Commands are accepted until a QUIT command, which terminates execution of the program, is encountered. Commands are phrased in the SYSREC Command Language, which follows a rather rigidly defined syntactical format. Each command must be in a form defined by the prototype for that particular command. If not, an error condition will arise, processing of the command will cease, and the user will have to re-enter the command in corrected form.

2.2 Input Formats

Commands submitted to SYSREC are presented on a line-by-line basis, the length of the line being a function of the input device being used. A command may be continued from one input line to the next by inserting a minus sign (-) as the last character of the line to be continued. A command may be given on not more than three separate, consecutive input lines thus allowing a maximum of 396 characters per command (a single input line may thus contain not more than 132 characters - currently more than the possible input line length for any remote terminal). The following considerations also govern the command language:

- 1) The only acceptable input characters are

A-Z, 0-9, (,), =, blank character, *, ".

The use of any other characters will generate an invalid character error condition. The minus sign is unique, and is not a valid character except for continuation purposes.

- 2) Any names given to terminal boards, cables, or signals must not exceed eight characters in length, must begin with a letter (A-Z), and must be composed of alphanumeric characters.
- 3) Any numbers given to specify pin numbers or line numbers must be within the range 0-32767 inclusive.
- 4) A blank character must be used as a delimiter between entities in a command statement.
- 5) Keywords for a command may be given in any order, but must contain an equals sign (=) delimiting the keyword, or its abbreviated form, from the value given.
- 6) The characters comprising an input line need not start in column one of the line.
- 7) The character slash (/) is an automatic prefix to any input to, or output from, SYSREC. The slash character appearing by itself at the beginning of a line indicates that SYSREC is ready to accept input.
- 8) The appearance of the word DONE following the submission of a command indicates that processing requested by that command has been successfully completed, regardless of the appearance of other messages (see Section 3.2 on error messages).

2.3 Command Descriptions

2.3.1 Notation and Terminology

The first two letters of each command identify that command uniquely; thus only those letters *need* be given. Similarly, only the first two letters of *most* keywords are needed. Some keywords have identical first two letters; these are distinguished by their first three letters. Acceptable minimum abbreviations for commands and key-words are underlined in the prototype statements. The following conventions are used in the prototype statements:

- 1) Items in lower case represent symbolic quantities which are to be supplied by the user.
- 2) Items in upper case are part of the command and must be given as indicated.
- 3) [] denotes enclosed material is optional.
- 4) { } denotes possible choices, only one of which is allowed.

The following commands are available to the SYSREC user:

2.3.2 Commands

Name: ALTER

Purpose: to increase the number of pins on a terminal board

Prototype: ALTER tbname(totalpins)

tbname is the name of the terminal board on which a new number of pins is requested.

totalpins is the new number of total pins requested for the terminal board. It must be greater than or equal to the total number of pins currently on the terminal board.

Use: To increase the number of pins on a terminal board in the event that all current pins are in use. A maximum of nine individual alterations per terminal board is permitted.

Effect: The total number of pins on tbname is set to totalpins.

Examples: ALTER TBN(100)

AL BOARD10(575)

Name: CONNECT

Purpose: to route a signal along a specified cable.

Prototype: CONNECT cname[(startlineno.)] sname(sigdim)

[CODE=code value] [DESC=description]

cname is the name of the cable along which the signal is to run.

startlineno. is a cable line number specifying the first of sigdim consecutive lines along which signal is to run. Default is startlineno.=first of sigdim consecutive lines not yet carrying a signal.

sname is the name assigned to the signal being routed.

sigdim represents the 'dimension' of the signal, the number of lines within one cable necessary to carry the signal.

codevalue is a two-digit number indicating that this signal may only run along those cables with code value equal to codevalue. Default is CODE=00.

description is any phase descriptive of the signal being routed. It may not contain embedded quotation marks or minus signs.

Use: To route signals explicitly from terminal board to terminal board. It is an alternative to the PUT command which routes signals automatically.

Effect: The named signal is routed along the specified cable.
If a sufficient number consecutive lines is not available within the cable, the signal is not routed. Should the cable code and the requested code value not match, the signal is not routed.

Examples: CONNECT CABL(10) SIGNAL1(4)
C1 SIG100(50) CODE=99 DESC='CONTROL SIGNAL'

Name: CREATE

Purpose: to create a terminal board

Prototype: CREATE tbname(no.pins) [WEIGHT=weightvalue]
[DESC='description']

tbname is the name assigned to the terminal board being created

no.pins is the number of pins on the terminal board.

The pins are numbered sequentially from 1.

weightvalue represents a weighted value to be assigned to the terminal board. Weight values are considered by the minimum path algorithm. Default is WEIGHT=0.

description is any phrase descriptive of the terminal board. It may not contain embedded quotation marks, or minus signs.

Use: To create terminal boards and to set appropriate initial values.

Effect: Terminal board tbname is created with the specified initial characteristics.

Examples: CREATE TBN(100)

CR BOARD(120) WEIGHT=1000 DESC='THIS IS THE MAIN PANEL'

Name: DESCRIP

Purpose: to attach a descriptive phrase to a cable, signal or terminal board

Prototype: $\underline{\text{DESCRIP}} \left\{ \begin{array}{l} \underline{\text{CABLE}}=\text{cname} \\ \underline{\text{SIGNAL}}=\text{sname} \\ \underline{\text{TB}}=\text{tbname} \end{array} \right\} \underline{\text{DESC}}=\text{'description'}$

cname is the name of a cable

sname is the name of a signal

tbname is the name of a terminal board

description is the appropriate descriptive phrase. It may not contain embedded quotation marks or minus signs.

Use: Either to describe a network entity not having a description field specified when first created, or to modify an already existing description field.

Effect: The description phrase given becomes the current description for the specified entity.

Examples: DESCRIP CABLE=C1 DESC='THIS CABLE WAS FORMERLY A COAX CABLE'.

Name: DISCONN

Purpose: to disconnect a signal or cable

Prototype: DISCONN $\left\{ \begin{array}{l} \text{SIGNAL=sname} \\ \text{CABLE=cname} \end{array} \right\}$

sname is the name of the signal to be disconnected.

cname is the name of the cable to be disconnected.

Use: This command deletes signals or cables from the current configuration. If SIGNAL=sname is specified, an announcement is made indicating that sname is to be disconnected and confirmation of intent is requested.

The only affirmative response is OK.

If CABLE=cname is specified, all signals currently running through cname (if any) are listed followed by an announcement that cname is to be disconnected and confirmation of intent is requested. The affirmative reply is OK.

Effect: If SIGNAL=sname is specified, sname is removed from the current configuration. All terminal board pins and cable lines formerly occupied by the signal are considered available to carry new signals. Any jumper lines inserted at the time sname was routed are removed.

If CABLE=cname is specified, cname is removed from the current configuration. All terminal board pins to which cname was formerly attached are considered available for connections to new cables. In addition, each signal currently running through the cable is disconnected in the same manner as described for the SIGNAL=sname option.

Examples: DISCONN SIGNAL=S101

DI CABLE=CABLE100

Name: EXTEND

Purpose: to extend the range of an existing signal

Prototype: EXTEND sname BETWEEN tbyname1[(startpinno.)]
AND tbyname2[(startpinno.)]
 [DIRECT={ON }] [SL=sublabel]
 {OFF }

sname is the name of an existing signal, the range of which is to be extended.

tbyname1 is the name of a terminal board through which the signal currently runs.

tbyname2 is the name of the terminal board to which the signal is to be extended

startpinno. is a terminal board pin number specifying the first pin of the terminal board through which the extended signal is to run. Default is startpinno.=first of sigdim consecutive pins not yet carrying a signal where sigdim is the dimension of the signal being EXTENDED.

sublabel is a number to be used as a descriptive quantity for this particular extension of the signal. Default is SL=00

Use: To extend a signal previously routed in accordance with the PUT or CONNECT commands. The dimension and code attributes of the extension are the same as those specified when the signal was initially routed. The DIRECT=keyword indicates whether or not the extension is

to be made according to the minimum path criterion. DIRECT=ON requires that there be at least one cable with sufficient free consecutive lines and code type connecting the terminal boards. The signal is extended along this cable, regardless of possible shorter routes. DIRECT=OFF specifies that the extension is to be made on a minimum path length basis. The default for this option is DIRECT=OFF.

This command with the DIRECT=ON option may be used to route a defined signal explicitly from terminal board to terminal board.

Effect: The named signal is extended as requested.

For the DIRECT=OFF option, the effect of insufficient terminal board pins, insufficient cable lines, or dissimilar code values is the same as in the case of the PUT command. For the DIRECT=ON option these situations result in the signal not being extended. In the case of successful completion under either option, jumper lines are inserted on tbname1 from those pins through which the signal currently passes to an appropriate set of pins on tbname1 not occupied by a signal.

Examples: EXTEND SIG1 BETWEEN TB1 AND TB2
EX SG1000 B TBD(10) A TBE(100) DIR=ON SL=1

Name: LIST

Purpose: to obtain a list of all signal names, cable names, or
terminal board names

Prototype: LIST { SIGNALS }
 { CABLES }
 { TBS }

Use: To obtain a listing of all currently defined signals,
cables or terminal boards.

Examples: LIST TBS
LIST SIGNALS

Name: MTS

Purpose: to return to MTS command mode

Prototype: MTS

Use: To cause input processing to return to the control of MTS. Any MTS command other than RUN, LOAD, UNLOAD or SIGNOFF may then be issued. A return to processing by SYSREC may be made by issuing the MTS command \$RESTART.

Example: MTS

Name: PUT

Purpose: To place a minimum length signal between two terminal boards.

Prototype: PUT sname(sigdim) BETWEEN tbyname1[(startpinno.)]
AND tbyname2[(startpinno.)] [CODE=codevalue] [DESC='description']

sname is the name assigned to the signal to be placed between the two terminal boards

sigdim is a number representing the dimension of the signal (ie the number of lines needed to carry the signal within one cable)

tbyname1, tbyname2 are the names of the terminal boards between which the signal runs.

startpinno. is a terminal board pin number specifying the first of sigdim consecutive pins to which the signal is to run. Default is startpinno.=first of sigdim consecutive pins not yet carrying a signal.

codevalue is a two-digit number indicating that this signal may run along only those cables having code values equal to codevalue. Default is CODE=00.

description is any phrase descriptive of the signal. It may not contain embedded blanks, or minus signs.

Use: To create and route a signal between (certain pins of) two terminal boards by the shortest path.

Effect: The named signal is routed between the given terminal boards according to a minimum path criterion on the basis of the *current* network configuration. Jumpers between pins on the same terminal board are added automatically as required. Should there be insufficient consecutive free pins on a terminal board, or insufficient consecutive free lines within a cable, that terminal board or cable is discarded from consideration as a possible conductor of the signal. Further, should a cable code and the requested code not match, that cable is also discarded from consideration. (See Appendix B for details of the minimum path algorithm and its operations.)

Examples: PUT S1(4) BETWEEN TB1 and TB2
PUT SIG1(100) B T1(10) and TBB2(50) CODE=25
DESC='DESCRIPTION OF SIG1'

Name: QUIT

Purpose: to terminate execution of SYSREC

Prototype: QUIT

Use: To terminate program execution and to return to MTS command mode. Prior to terminating execution, QUIT initiates the procedure which allows the user to save, in a file of his choice, the current network configuration. When QUIT is issued, the user is prompted to enter an MTS file name into which the current network will be written. The name given may indicate either a permanent or scratch file.

If the file name given does not already refer to an existing file, SYSREC will create a file of appropriate type (permanent or scratch) and of sufficient size to accommodate the data record being written.

In the case of a permanent file, if the file size required to hold the record is larger than the maximum file space remaining to the user, he will be prompted to re-enter a file name denoting a scratch file. If the file name does refer to an existing file, the user will be asked to confirm an intention to destroy the file.

If the user response is - OK - the file is destroyed and another of the same name is created in its place.

If the file name given is an illegal name, the user is prompted to enter a corrected form.

Effect: When a save file of correct size has been established, SYSREC saves the current configuration in this file. The program is then terminated.

Example: QUIT

Note 1: In the event that the configuration is saved in a scratch file because the user does not have available sufficient permanent file space, the user should, once SYSREC has terminated, either destroy sufficient of his existing files to provide permanent file space into which the scratch file may be copied, or copy the scratch file onto tape. If the user creates any files, *by himself*, which are to be used as save files, such files *must be of sequential type without line numbers*. Otherwise SYSREC will be unable to re-read the saved record.

Note 2: Users are strongly recommended to keep backup records of all 'saved' files so that, in the event of system failure, only the current run will be lost.

Name: ROUTE

Purpose: to determine if there is a signal route between two terminal boards

Prototype: ROUTE tbnamel[(startpinno.)] AND tbnam2[(startpinno.)]
DIMEN=sigdim [CODE=codevalue]

tbnamel, tbnam2 are the names of the terminal boards between which the route is to be determined

startpinno. is a terminal board pin number for the associated terminal board specifying the first of sigdim consecutive pins to which the signal is to run.

Default is startpinno.=first of sigdim consecutive pins not yet carrying a signal.

sigdim is a number representing the dimension of the signal to be routed (ie the number of lines required within one cable to carry the signal)

codevalue is a two-digit number indicating that this signal route may run along only these cables having code values equal to codevalue. Default is CODE=00.

Use: This command is used to determine whether or not there is a signal route of specified dimension and type between two terminal boards. Unlike the PUT command, no signal is actually installed.

Effect: An attempt is made to determine a route between the given terminal boards using the minimum path algorithm. Insufficient free terminal board pins or cable lines within a cable result in that terminal board or cable being discarded from consideration as a possible element of the signal route. Should a cable code and the given code value not match, that cable is also discarded from consideration. (See Appendix B concerning the minimum path algorithm and its operation.)

Examples: ROUTE TB1 AND TB2 DIM=10 CODE=04
RO BOARD1(10) A BOARD2(40) DIM=6 CODE=90

Name: RUN

Purpose: to create a cable and place it between two terminal boards

Prototype: RUN cname(no.lines) BETWEEN tbyname1[(startpinno.)]
AND tbyname2[(startpinno.)] LENGTH=cablelength
[CODE=codevalue] [DESC='description']

cname is the name assigned to the cable being connected to the specified terminal boards

no.lines is the number of lines assigned to the cable.

tbyname1, tbyname2 are the names of the terminal boards between which the cable is placed

startpinno. is a terminal board pin number specifying the first of no.lines consecutive pins to which to which the lines of the cable are to be consecutively attached. Default is startpinno.=first of no.lines consecutive pins not yet attached to a cable.

cablelength is a number representing the length of the cable running between the terminal boards. Units are arbitrary.

codevalue is a two-digit number setting the coded value for cname. This value is used to ensure similarity of code types when routing signals. Default is CODE=00.

description is any descriptive phrase about the cable. It may not contain embedded blanks, or minus signs.

Use: To create and place a named cable between certain pins of two terminal boards.

Effect: The cable named is placed as follows between the two terminal boards; starting with startpinno. (defaulted or otherwise) for each terminal board the lines of the cable beginning with line 1 are attached consecutively to each terminal board until all cable lines are attached. If there are insufficient available pins on either terminal board no connection of any line or pin is made and reference to cname is not generated. Cablelength is taken to be the absolute length between the two terminal boards for purposes of minimum path routing. Any cable(s) subsequently placed between tbnamel and tbnam2 will have their lengths defaulted to cablelength.

Examples: RUN CABLE1(10) BETWEEN TB1 AND TB2 LENGTH=10
RUN C100(100) B TBN1(10) A TBN2(50) LE=100 CODE=01
RUN C1(4) B BOARD1 AND BOARD2(10) LE=31 CODE=99
DESC='A CABLE DESCRIPTION'

Name: SET

Purpose: to alter the maximum possible number of terminal boards or cables.

Prototype: SET $\left\{ \begin{array}{l} \text{TB}=\text{totaltbs} \\ \text{CA}=\text{totalcabs} \end{array} \right\}$

totaltbs is a number representing the maximum number of terminal boards desired

totalcabs is a number representing the maximum number of cables desired.

Use: Initially, SYSREC allows the definition of a maximum of 100 terminal boards, and 100 cables (this includes cables which are defined and then later deleted via DISCONN)
The SET command is a means of altering these initial maximum element restrictions. (The new maximum number(s) must be greater than the number of terminal boards or cables currently defined.)

Effect: The maximum number of terminal boards or cables permitted is altered as specified.

Note 1: Due to internal processing considerations issuing the SET command results in program termination once a maximum element count has been altered. (See QUIT command re termination events). If the user wishes to continue after issuing SET he must restart the program in the continuation mode.

Examples: SET TB=200
SET CA=1000

Name: SUMMARY

Purpose: to obtain summarized output describing the current configuration

Prototype:
$$\text{SUMMARY} \left\{ \begin{array}{l} \text{SIGNALS} = \{ \text{sname} \} \\ \text{CABLES} = \text{cname} \\ \text{TB} = \text{tbname} \quad [\text{PRINT} = \{ \text{LONG} \}] \\ \text{ALL} = [\text{PRINT} = \{ \text{LONG} \}] \end{array} \right\}$$

sname is the name of a signal

cname is the name of a cable

tbname is the name of a terminal board

* indicates that all the currently defined elements within a class (ie terminal boards, cables, signals) are to be summarized.

Use: To obtain summaries of individual signals, all signals, individual cables, all cables, individual terminal boards, all terminal boards or all signals, cables and terminal boards. PRINT=LONG indicates that terminal board summaries are to include the current status of each individual pin. PRINT=SHORT suppresses this individual pin synopsis. The default is PRINT=SHORT.

Specifying ALL=generates summaries of all terminal boards, followed by summaries of all cables, followed by summaries of all signals. Formats and explanations of output generated by this command are given in Section 3.

Effect: Obvious

Examples: SUMMARY SI=SIG1

SUMMARY TB=* PRINT=LONG

SU ALL= PR=L

Name: TRACE

Purpose: to trace a signal or to summarize the status of an individual cable line or an individual terminal board pin.

Prototype: $\text{TRACE} \left\{ \begin{array}{l} \text{SIGNAL}=\text{sname} \\ \text{CABLE}=\text{cname}(\text{lineno.}) \\ \text{TB}=\text{tbname}(\text{pinno.}) \end{array} \right\}$

sname is the name of a signal.

cname is the name of a cable.

lineno. is the number of a line in cname which is to be summarized

tbname is the name of a terminal board

pinno. is the number of a pin on tbname which is to be summarized.

Use: To obtain a detailed trace of a signal route or summaries of individual cable lines or terminal board pins. Formats and explanations of output generated by this command are given in Section 3.

Effect: Obvious

Examples: TRACE SI=SIGAL2

TR CA=CAB1(100)

Name: WEIGHT

Purpose: to alter the weight value assigned to a terminal board.

Prototype: WEIGHT tbnam(weightvalue)

tbnam is the name of a terminal board whose weight value is to be changed.

weightvalue is a number representing the new weight value assigned to the terminal board.

Use: To define an initial weight value for a terminal board or to redefine a current weight value. Weight values are considered by the minimum path algorithm. (See Appendix B concerning minimum path algorithm and its operation).

Effect: weightvalue becomes the weight value for the terminal board replacing the previous weight value.

Examples: WEIGHT TBN1(100)

WE TB2(0)

Section 3: OUTPUT

3.1 Output from SYSREC Commands

Output is generated in response to the commands LIST, SUMMARY and TRACE. This section describes, by means of examples, the format of output produced by each command and the meaning of such output.

3.1.1 LIST output

The LIST command is used to obtain a listing of the names of all currently defined signals, cables or terminal boards. Thus issuing the command

LIST SIGNALS

might produce the output

LIST OF SIGNALS FOLLOWS

SIGNAL

SIG2

CONTSIG

S10112

ABCSIG

Similar output will be generated by LIST CABLES or LIST TBS. If there are no elements within the class specified, an indication to that effect will be made.

3.1.2 SUMMARY Output

The summary command is used to obtain summarized status information about signals, cables, or terminal boards. Summaries may be generated for individual elements within the classes of signals, cables, or terminal boards or for all elements within a class (see 2.3 SUMMARY Command). What is described here is the

summarized output for individual items within a class. Total class summaries are composed of the summaries of all items within that class (Void classes are indicated as such).

For a signal summary, the command

```
SUMMARY SIG=SIGNA10
```

might produce the output

```
SUMMARY:SIGNA1=SIGNA10
```

```
THIS SIGNAL IS A TEMPERATURE MONITOR
```

```
DIM=50 LENGTH=1414
```

```
DATE=AUG21/70
```

The first line of output serves to identify that which is being summarized, in this case a signal labelled SIGNA10. Following this is the descriptive phrase associated with this signal (if any).

Further indications are that the dimension or width of the signal is 50 lines; the total route length for the signal is 1414 units; the date on which the signal was last modified (or initially defined if it has not yet been EXTENDED) was August 21, 1970.

For a cable summary, the command

```
SUMMARY CA=CAB
```

might produce

```
SUMMARY: CABLE=CAB
```

```
THIS IS A POWER SUPPLY CABLE
```

```
NO. LINES=2000 NO. LINES FREE=500
```

```
LENGTH=375 CODE=01
```

```
CONNECTS TB=TB100 PINS=1-2000 AND TB=TB101
```

```
PINS=2001-4000
```

The first line affirms the intent of the summary followed by the descriptive phrase attached to this cable (if any). Further indications are that the cable has a total of 2000 lines, 500 of which are currently unoccupied by a signal; the length of the cable is 375 units; the code value of the cable is 01; the cable runs between pins 1 to 2000 of terminal board TB100 and pins 2001 to 4000 of terminal board TB101.

A terminal board summary, requested by

SUMMARY TB=TCBN1 PR=L

might produce

SUMMARY: TB=TCBN1

LOCATION P1 FLOOR 2

NO. PINS=6 NO. PINS FREE=2

WEIGHT=100

PIN NO.	ATTACHED CABLE: LINE NO.	SIG CARRIED	JUMPERS TO	DATE
1	CABL1: 1	SIG1	3	Aug21/70
2	CABL1: 2			
3	CAB100: 1	SIG1	1	Aug21/70
4	C1001: 1	SGNALA		Aug16/70
5	*FREE*			
6	*FREE*			

Again the first two lines of output confirm the intent of the summary and note the descriptive phrase attached to this terminal board (if any). Following this, it is stated that the terminal board has a total of six pins, two of which are unattached to a cable (or cables) and the current weight value of this terminal board is 100. If PRINT=SHORT had been specified in the command, or if the PRINT keyword had been

omitted, the summary would terminate at this point. For this example however, PRINT=LONG has been specified and the remainder of the summary is a result of such a specification. For the above example, the further indications are that

- 1) pin 1 is attached to line 1 of a cable, CABL1, part (in this case all) of a signal labelled SIG1 runs through this pin, and cable line; the signal was most recently modified (or defined if not yet EXTENDED) on the date given; there is a jumper line from pin 1 to pin 3.
- 2) pin 2 is attached to line 2 of the cable, CABL1; no signal flows through this pin and there are no jumpers on the pin.
- 3) pin 3 is attached to line 1 of a cable CAB100; this pin is jumpered to pin 1 and thus carries the same signal (or part signal) as does pin 1.
- 4) pin 4 is attached to line 1 of a cable C1001; a signal labelled SGNALA runs through this pin and the pin has no jumpers.
- 5) pins 5 and 6 are as yet not connected to a cable(s) and are thus available to accommodate such connections.

3.1.3 TRACE Output

The TRACE command is used to trace a signal route or to obtain information about the status of an individual cable line or terminal board pin.

A signal trace issued as

TRACE SI=SIGNALL

might produce

TRACE: SIGNAL=SIGNALL DIM=2 DATE=AUG 21/70

TBA : 1 TO TBB : 1 (C1:1) SL=0

TBB : 1 TO TBB : 7 TO TBC : 3 (C3:1) SL=1

TBC : 3 TO TBC : 7 TO TBD : 1 (C4:1) SL=2

(In the following explanation, pin numbers refer to the first of, in this case, DIM=2 pins through which the signal runs. The same applies to cable line numbers). The first line of output identifies the signal being traced, its dimension, and the date on which it was last modified. The following lines indicate that the signal runs from pin 1 on a terminal board, TBA, to pin 1 on a terminal board TBB via line 1 in a cable labelled C1; between pins 1 and 7 on TBB (via a jumper) and then from pin 7 on TBB to pin 3 on TBC via line 1 of cable C3; between pins 3 and 7 on TBC (via a jumper) and then from pin 7 on TBC to pin 1 on TBD via line 1 of cable C4. The SL=X is a reference label that may be attached to extensions of a signal for identification purposes. The following figure illustrates the signal route for the preceeding example.

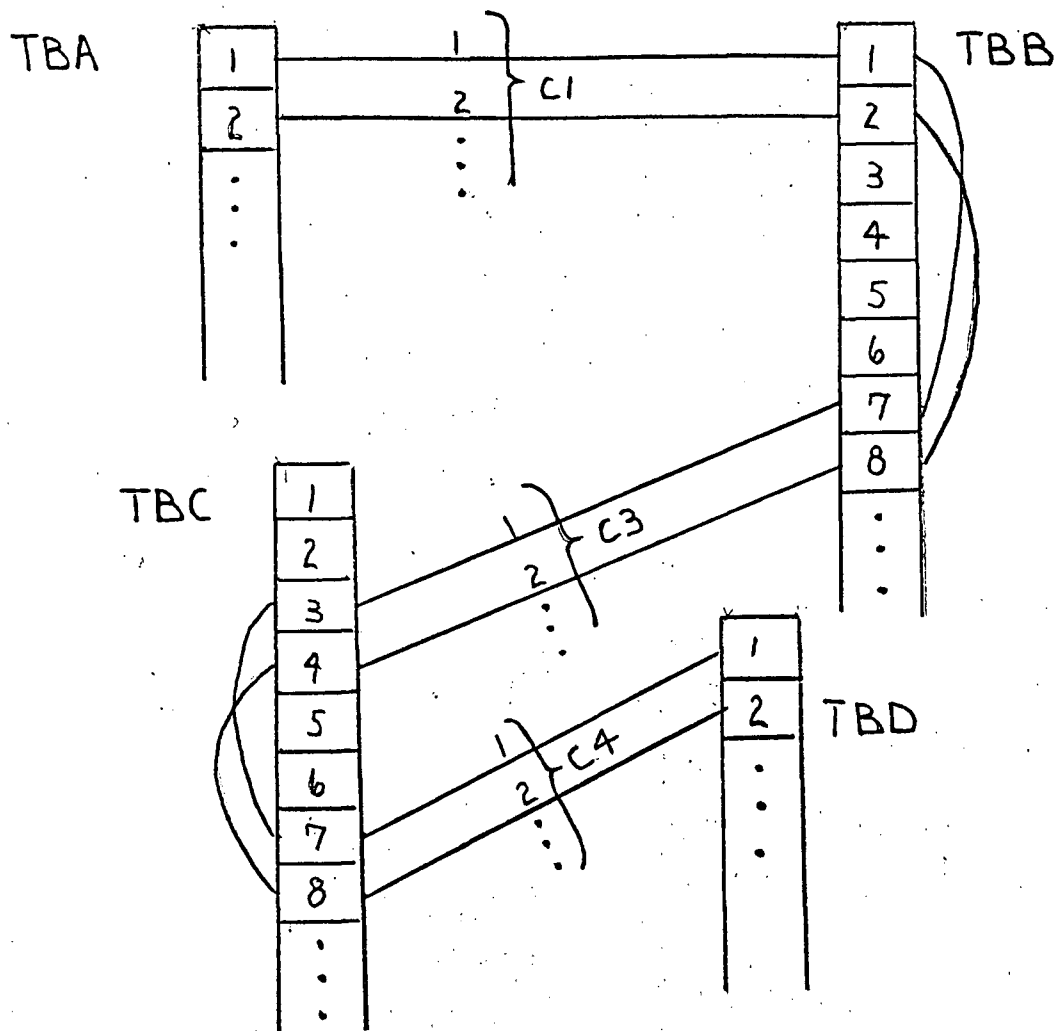


Figure 1

The first and last terminal boards given in the output list (in this case TBA and TBD) may not in all cases, refer to the end points of the signal route.

The status of pin 1 on cable CAB1 may be obtained by using

TRACE CAB1(1)

perhaps generating the output

TRACE: CABLE=CAB1 LINE=1

CONNECTS TB=TB1 PIN=10 AND TB=TBOARD4 PIN=4

SIGNAL CARRIED=SIGNAL1 SL=41

The first output line affirms the intent of the trace while succeeding lines indicate that the cable line connects pin 10 on terminal board TB1 with pin 4 on terminal board TBOARD4 and that the name of the signal carried by this cable line and the sub label relating to this portion of the signal route are SIGNAL1 and 41 respectively. If the cable line carries no signal, indication to that effect is made.

For the status of a terminal board pin,

TRACE TB=TBN1(10)

might produce:

```
TRACE:TB=TBN1      PIN=10
CONNECTED  CABLE=CABL100  LINE=10
SIGNAL CARRIED=S1  SL=0
JUMPED TO PIN(S)  20, 30
```

The first line affirms the intent of the trace while the second line indicates that this pin is connected to line 10 of a cable labelled CABL100. The third line indicates the name of a signal running through this pin and the signal's sublabel at this point. Finally, any pins jumpered to on this terminal board are listed. If there is currently no cable attached to this pin an indication is so made and the trace is terminated. If there is a cable line attached to the pin but no signal through the pin or no jumpers from the pin indications are so made.

3.2 Output from Error Messages

In addition to data generated by the submission of explicit commands, SYSREC also outputs error messages indicating the detection of invalid operations. Such messages and their meanings are described in Appendix A.

Section 4: PROGRAM EXECUTION

This chapter describes general startup and termination procedures for SYSREC. In addition, specific details are provided of how to initiate execution of SYSREC operating under MTS.

4.1 General Startup/Termination

SYSREC may be started in one of two modes; initial mode or continuation mode. Initial mode means that a new network record is about to be created. Continuation mode indicates that a network record previously created and saved by SYSREC is about to be modified in some fashion. The starting mode for SYSREC is determined by means of a parameter field passed to the program by the user when he initiates execution. If the parameter field is the character string INITIAL, execution is started in initial mode. If the parameter field is the string CONTINUE, or if no parameter field is given, execution begins in continuation mode. If the startup mode is continuation, then the user must supply reference to a file or data set that contains a network configuration previously created and saved by SYSREC.

SYSREC execution is normally terminated by the QUIT command. This command saves the current configuration in a file specified by the user. When the record is completely saved, the program terminates.

4.2 Interrupt Procedure

SYSREC is designed to accept attention interrupts from a remote terminal. In the event that such an interrupt is given while operations relating to a submitted command are in progress, such operations are immediately terminated regardless of whether or not processing requested by the command is complete. The message

SYSREC ATTENTION INTERRUPT

is issued, signifying the interrupt, and SYSREC is readied to accept a subsequent command.

Since processing terminated by an attention interrupt is *NOT* restarted once the interrupt has been accepted, care must be taken not to interrupt the processing associated with certain commands. Commands which must not be interrupted are those which involve the completion of internal data tables. To interrupt such commands could result in incomplete data structures thus causing the program to fail at a later stage. It is recommended that only the LIST, SUMMARY and TRACE commands be interrupted. In fact, the interrupt facility was implemented solely as a means of prematurely terminating output lists. Any other commands which the user might deem it necessary to interrupt have associated with them counter-commands (or the same command itself) enabling a reversal of action (eg CONNECT, DISCONN). The exceptions to the latter statement are the QUIT and SET commands whose effects are irreversible except for restarting the program. In no circumstances should the QUIT command be interrupted.

4.3 Startup Requirements under MTS

This section deals with the initiation of SYSREC execution under MTS. Readers of this material are assumed to be familiar with the MTS command languages and MTS file structures.

Having successfully signed onto the system, SYSREC may be started in the initial mode by issuing

```
$RUN SYSREC PAR=INITIAL
```

or in the continuation mode by issuing

```
$RUN SYSREC 5=infile [PAR=CONTINUE]
```

infile is the name of a file containing a previously created SYSREC record (ie a file name previously specified in response to the QUIT (or SET) command).

Should the PAR= specification be incorrect, or, in the case of continuation mode, the reference to logical I/O unit 5 be missing or incorrect, a message will be issued and the user will have to repeat the startup procedure. The prefix character slash (/) will appear when SYSREC is ready to accept commands.

BIBLIOGRAPHY

1. Dijkstra, E. W., "A Note on Two Problems in Connexion with Graphs", Numerische Mathematik 1, 269-271 (1959).
2. Dreyfus, S. E., "An Appraisal of Some Shortest-Path Algorithms", Operations Research 17, 395-412 (1969).
3. IBM System/360 Operating System: Assembler Language, Form C28-6514-5.
4. IBM System/360 Principles of Operation, Form A22-6821-7.
5. Nicholson, T. A. J., "Finding the Shortest Route Between Two Points in a Network", Computer Journal 9, 275-280 (1966).
6. Walsh, D., A Guide for Software Documentation, Inter-ACT, New York, N.Y. (1969).
7. Whiting, P. D. and J. A. Hillier, "A Method for Finding the Shortest Route through a Road Network", Operational Research Quarterly 11, 37-40 (1960).

APPENDIX A

Program Generated Messages

(Messages are listed in alphabetical order)

<u>Message</u>	<u>Explanation</u>
BLANK LINE-LINE IGNORED	A blank line was entered; it has been ignored
CABLE ALREADY EXISTS	The cable name has been previously assigned to another cable
CABLE CODES NOT SIMILAR	The requested signal code differs from a cable code type
CABLE DOES NOT EXIST	A given cable name cannot be found; the cable has not yet been RUN
CABLE LENGTH MUST BE > ZERO	The cable length given in a RUN command is zero; it must be greater than zero
CABLE LINES EXCEED FREE TB PINS	A terminal board does not possess sufficient consecutive free pins to permit its connection to the specified cable
CABLE NAME LIST OVERFLOW	The maximum allowable number of cables has already been run; use SET command to alter the list size
CANNOT DESTROY THIS FILE	SYSREC cannot destroy the file as requested; enter a new file name
CODE VALUE EXCEEDS 99	Code values must be in the range 00-99
COMMAND CANCELLED	A reply other than OK has been given in response to a program query
CONFLICTING KWS GIVEN	Contradictory keywords are given in the same command
DESCRIPTION OMITTED	The DESC keyword has been omitted from the DESCRIP command or a description of zero length has been given
ERROR IN GIVEN FILE NAME	The given file name contains an invalid character etc.; enter a new name

GIVEN PIN DOESNT LEAD TO TB1	DIRECT=ON has been specified in the EXTEND command; the specified pin on the 'to' board does not lead directly to the 'from' board
GIVEN PIN DOESNT LEAD TO TB2	DIRECT=ON has been specified in the EXTEND command; the pin given on the 'from' board doesn't lead directly to the 'to' board
GIVEN PINS DO NOT CONNECT	DIRECT=ON has been specified in the EXTEND command; the specified pin numbers do not connect the terminal boards directly
INCOMPLETE COMMAND	The command given is incomplete: one or more expected parameters is missing
INPUT FILE-NAME NOT FOUND	SYSREC is being started in continuation mode but the file given as containing an existing configuration cannot be found
INPUT FILE-TYPE NOT SEQUENTIAL	The file given as containing an existing record is not of sequential type; SYSREC reads only sequential files
INSUFF FREE LINES(CONNECT)	The cable specified in a CONNECT command does not possess sufficient free lines to accomodate this signal width
INSUFF FREE LINES(EXTEND)	DIRECT=ON has been specified in the EXTEND command; there are insufficient free lines to accomodate the requested extension
INSUFFICIENT FREE LINES	More cable lines than are currently free have been requested
INSUFFICIENT LINES	A line number has been specified; this is resulting in the attempted use of a line which is not defined
INSUFFICIENT PINS	A terminal board pin number has been specified; this results in the requested use of a non existant pin
INVALID (ENCOUNTERED	A left parenthesis has been unexpectedly encountered; an unwanted parameter has been given

INVALID ALPHA SYMBOL	An invalid alphanumeric symbol has been detected; the symbol probably begins with other than A-Z
INVALID CABLE CONNECTION	Cables cannot be RUN from and to the same terminal board
INVALID CHARACTER ENCOUNTERED	A character other than those accepted by SYSREC has been encountered
INVALID CHARACTER IN NUMBER	A non-numeric character has been detected as part of a number
INVALID COMMAND	Given command is illegal for SYSREC
INVALID KW FOR THIS COMMAND	An invalid keyword has been given with a particular command
INVALID NUMERIC SYMBOL	An invalid number has been given; the number does not begin with 0-9
INVALID PAR= SPECIFICATION	During the startup procedure an invalid argument has been given in the PAR field of the MTS 'RUN' command
INVALID PARAMETER	A given parameter is invalid for the command(LIST) or the argument of a keyword is invalid(EXTEND,SUMMARY)
INVALID PHRASE DELIMITER	Either the delimiter AND(or A) or the delimiter BETWEEN(or B) is not given as required
LENGTH NOT SPECIFIED	The LENGTH keyword has been omitted from the RUN command; it is mandatory
MISSING (OR)	A left or right parenthesis has been omitted; a required parameter has been forgotten
MISSING = SIGN IN KW PARAMETER	The equals sign(=) is missing from a keyword parameter
MISSING BLANK OR EOL CHAR	A blank has not been found where expected or the last character of a command is not as expected
MISSING QUOTE MARK	One or both quote marks required to delimit a description field is missing

MORE THAN NINE EXTENSIONS	A maximum of nine alterations is allowed for any single terminal board
MORE THAN 3 I/P LINES	The maximum number of three input lines has been exceeded
NO CABLE CONNECTED TO THIS PIN	A pin number has been given(or implied) which is not connected to a cable line
NO CABLES CONNECTED TO THIS TB	A terminal board has been named to which no cables are attached
NUMBER EXCEEDS 32767	A number has been specified which exceeds the allowable maximum(32767)
PIN NO. EXCEEDS TOTAL PINS	A pin number has been given which is greater than the highest numbered pin currently on the board
PINS CONNECTED TO TWO CABLES	A pin number has been specified which would cause a signal's width to be split over two cables
PINS LESS THAN CURRENT PINS	In the ALTER command the number of pins to be set is less than the current number of pins on that board.
REQD LINE/PIN ALREADY ALLOCATED	A pin or line currently carrying a signal has been requested to carry another signal
REQUESTED MAXIMUM TOO SMALL	The number of terminal cards or cables being defined by the SET command is less than the present number of such elements
REQUESTED ROUTE IMPOSSIBLE	A signal cannot be successfully routed as requested
ROUTING SUCCESSFUL	The ROUTE command has determined that the signal route requested is feasible
SIGNAL ALREADY EXISTS	The signal name has been previously assigned to another signal
SIGNAL DOES NOT EXIST	The given signal name cannot be found; it has not yet been PUT
SIGNAL NOT ROUTED THRU HERE	The signal named does not pass through the first terminal board name; an extension cannot be started here

SYMBOL EXCEEDS 8 CHARACTERS	An alphabetic symbol exceeds eight characters
TB NAME LIST OVERFLOW	The maximum allowable number of terminal boards has already been CREATED; use SET command to alter the list size
TBS NOT DIRECTLY CONNECTED	DIRECT=ON has been specified in the EXTEND command but the terminal boards are not directly connected
TERMINAL BOARD ALREADY EXISTS	The terminal board name has been previously assigned to another terminal board
TERMINAL BOARD DOES NOT EXIST	A given terminal board name cannot be found; it has yet to be CREATED
TOO MANY JUMPERS REQUIRED	A request has been made which will necessitate assigning more than two jumper lines to a single pin
ZERO DIMENSION SPECIFIED	The dimension(width) of a signal has been given as zero; it must be greater than zero
ZERO LINES SPECIFIED	The number of cable lines specified in a RUN command is zero; it must be greater than zero
ZERO PINS SPECIFIED	The number of terminal board pins specified in a CREATE or ALTER command is zero; the number must be greater than zero

APPENDIX B

The Minimum Path Algorithm and its Operation

SYSREC uses a minimum path algorithm (MPA) to effect the routing of signals other than by explicit means. This appendix discusses the general nature of this algorithm and the way in which it is implemented by SYSREC.

B.1 The Algorithm

The algorithm adopted is due to Dijkstra [1] and Whiting and Hillier [7] and is implemented in the form suggested by Dreyfus [2]. This particular algorithm was chosen over others considered because:

- (1) for the given problem, this algorithm is more efficient than any others encountered (see Dreyfus).
- (2) the algorithm is easily programmed.

To illustrate the algorithm, consider the following network of nodes and paths. Circled numbers on the paths indicate the path lengths.

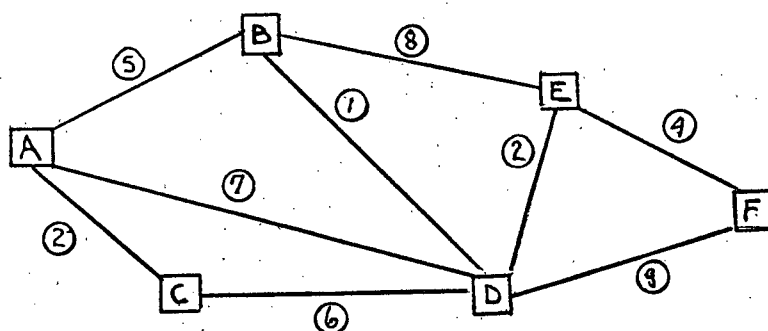


Figure 2

The object is to find the path of shortest route from node A to node F.

The algorithm operates basically by assigning labels to nodes and then using the value of these labels to determine the correct path.

Two types of labels are distinguished, permanent labels and temporary labels. For the above example, the first step is to assign labels to each node such that node A has a permanent label of 0 and all other nodes have temporary labels ∞ . Thus to start we have

<u>Node</u>	<u>Type</u>	<u>Value</u>
A	P	0
B	T	∞
C	T	∞
D	T	∞
E	T	∞
F	T	∞

Next, all paths emanating from node A are considered. For each such path the path length from node A to whichever node the path runs is compared to the current temporary label of that node (∞ in each case). Since the value of each path length from node A (A-B=5, A-C=2, A-D=7) is less than the current temporary label at each of the nodes to which A is connected, each of these temporary labels is replaced by its corresponding lesser value. Thus

<u>Node</u>	<u>Type</u>	<u>Value</u>
A	P	0
B	T	5
C	T	2
D	T	7
E	T	∞
F	T	∞

Having considered all paths from A, the smallest of the current set of temporary labels is defined to be a permanent label giving:

<u>Node</u>	<u>Type</u>	<u>Value</u>
A	P	0
B	T	5
C	P	2
D	T	7
E	T	∞
F	T	∞

Node C now replaces node A in the above procedure and the algorithm cycles again giving: (after having set the next permanent label)

<u>Node</u>	<u>Type</u>	<u>Value</u>
A	P	0
B	P	5
C	P	2
D	T	7
E	T	∞
F	T	∞

(Note that the temporary label at node D remains 7 since it is less than the path A-C-D of length 8.) Node B now becomes the current node. The iterations continue until finally node F has been permanently labelled, at which point, the minimum length path from node A to node F can be determined. The remaining sequence of steps leading to the solution is as follows:

<u>Node</u>	<u>Type</u>	<u>Value</u>	<u>Node</u>	<u>Type</u>	<u>Value</u>	<u>Node</u>	<u>Type</u>	<u>Value</u>
A	P	0	A	P	0	A	P	0
B	P	5	B	P	5	B	P	5
C	P	2	C	P	2	C	P	2
D	P	6	D	P	6	D	P	6
E	T	13	E	P	8	E	P	8
F	T	∞	F	T	15	F	P	12

By noting which node a permanently labelled node was labelled from, one can, starting from node F, determine the shortest route as A-B-D-E-F.

The general procedure for this algorithm is:

(1) permanently label the initial node 0 and temporarily label all other nodes ∞ (any large number).

(2) for each path from the initial node compare the sum of the permanent label at the initial node and the path length with the temporary label at the node to which the path leads (if the path leads to a permanently labelled node ignore that path). If this sum is less than the current temporary label at that node, replace this temporary label by the above sum but still leaving it a temporary label.

(3) find the smallest of all the temporary labels, label it permanent and consider its associated node to be the initial node. Go to step 2.

(4) if the terminal node becomes permanently labelled - stop - a solution has been found.

(5) if at any point the smallest remaining temporary value equals ∞ and the terminal node has not yet been permanently labelled, no solution exists (the terminal node is in fact isolated).

B.2 Implementation of the MPA by SYSREC

SYSREC implements the preceding algorithm by using as the path length between any two connected nodes (terminal boards):

(1) the length of the cable(s) connecting the two terminal boards (Note that there is at most 1 cable length between any two terminal boards) plus,

(2) the weight factor associated with the non-initial (in terms of the preceding section) terminal board. (Cable length is determined from the LENGTH keyword of the RUN command while terminal board weight factors are determined from the WEIGHT keyword of the CREATE command or from the WEIGHT command.)

An example may clarify this: consider the following terminal boards and their weight factors;

TBA weight factor = 10

TBB " " = 20

TBC " " = 30

and these cables and their lengths

C1 length = 4

C2 " = 6

C3 " = 8

all of which form the following configuration

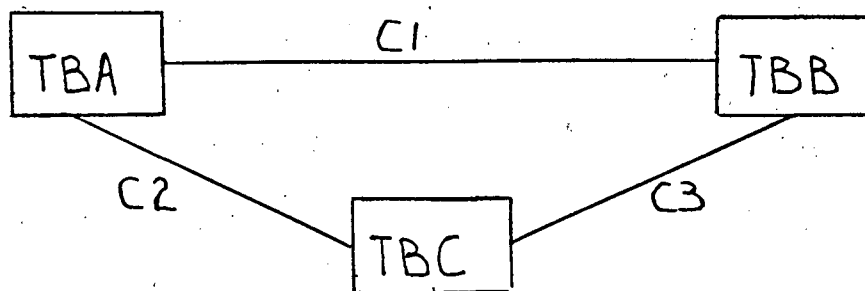


Figure 3

Then the direct path length from TBA to TBB is defined as: length of C1 + weight factor of TBB = $4 + 20 = 24$. The path length from TBA to TBB via TBC is: length of C2 + weight factor of TBC + length of C3 + weight factor of TBB = $6 + 30 + 8 + 20 = 64$. Therefore, by means of the WEIGHT command, the user may directly govern which terminal boards (and associated cables) are most likely to be a part of a particular minimum length signal route.

For the routing of a particular signal, the conditions necessary for a cable (and therefore the terminal board(s) to which it is connected) to be considered a part of the tree which defines the set of *possible* paths for the signal are: (1) the cable must be of code type corresponding to the given signal code type, (2) the cable must have sufficient free consecutive lines to carry the complete width of the signal and (3) it must be possible to supply sufficient jumper lines to this cable should the need arise.

For the implemented version of the MPA, a cable which contradicts either (or both) of conditions (1) or (2) above will be dropped from

consideration as a possible path element and an indicative message will be issued to that effect. Thus, for example, if the PUT command has been issued and messages subsequently appear indicating conflicting code types or insufficient cable lines the inference should be that the MPA has excluded these cables as possible elements of the final route. It may well be the case that such messages are issued along with the DONE indications. This denotes that the signal was successfully routed but that certain cables were deleted as possible path elements during the routing procedure. If (3) of the required conditions is violated, an indicative message will be issued and the attempt at routing the signal is terminated. The user must then determine at which point more than two jumper lines are required and either route the signal explicitly around this point or increase the weighting factor of the associated terminal board, thus causing the MPA to distinguish a different path at this point.

One further problem occurs in the MPA implementation and it relates to the specification of pins on the end point terminal boards. Signals are routed *from* the terminal board first mentioned in a command *to* the terminal board mentioned second. That is, the algorithm takes as the initial node the "from" terminal board and proceeds via iteration until the "to" terminal board is permanently labelled. If pins are specified for the "from" terminal board they must be sufficient in number to carry the signal width and they must be connected to a cable of the correct code type. Otherwise, the signal is deemed impossible to route as requested and indicative messages are generated. The same applies to the "to" terminal board

if pins are specified for it. In routing a signal between specified pins on two terminal boards it may be the case that the route determined is in fact not the minimal length route between the terminal boards but rather the minimal length route *between the specified pins* on the terminal boards. That is, since pins have been specified through which the signal is to run, the route chosen may be greater in length than if such pins had not been specified. However, whether pins have been specified or not, the successful routing of a signal by the MPA guarantees that the path so determined is the minimal length path satisfying the given restrictions.

APPENDIX C

SYSREC Command Prototypes

1. ALTER t_bname(totalpins)
2. CONNECT c_name[(startlineno.)] s_name(sigdim)
 [CODE=codevalue] [DESC='description']
3. CREATE t_bname(no.pins) [WEIGHT=weightvalue]
 [DESC='description']
4. DESCRIP { CABLE=c_name DESC='description'
 SIGNAL=s_name
 TB=t_bname }
5. DISCONN { SIGNAL=s_name
 CABLE=c_name }
6. EXTEND s_name BETWEEN t_bname1[(startpinno.)] AND
 t_bname2[(startpinno.)] [DIRECT= { ON }
 OFF]
 [SL=sublabel]
7. LIST { SIGNALS
 CABLES
 TBS }
8. MTS
9. PUT s_name(sigdim) BETWEEN t_bname1[(startpinno.)]
 AND t_bname2[(startpinno.)] [CODE=codevalue]
 [DESC='description']
10. QUIT
11. ROUTE t_bname1[(startpinno.)] AND t_bname2[(startpinno.)]
 DIMEN=sigdim [CODE=codevalue]
12. RUN c_name(no.lines) BETWEEN t_bname1[(startpinno.)]
 AND t_bname2[(startpinno.)] LENGTH=cablelength
 [CODE=codevalue] [DESC='description']

13. SET { TB=totaltbs
 CA=totalcables }
14. SUMMARY { SIGNALS= sname
 *
 CABLES= cname
 *
 TB= tbname [PRINT={ LONG \ }]
 * { SHORT }
 ALL= [PRINT={ LONG \ }]
 { SHORT } }
15. TRACE { SIGNAL=sname
 CABLE=cname(lineno.)
 TB=tbname(pinno.) }
16. WEIGHT tbname(weightvalue)

SYSREC INTERNAL LOGIC MANUAL

Barry H. Gray
Department of Computer Science
University of British Columbia

April, 1971

TABLE OF CONTENTS

	Page
Preface	II.1
Section 1 Program Logic - overview	II.2
1.1 Introduction	II.2
1.2 CABLE csect	II.2
1.3 WORKER csect	II.5
Section 2 Program Logic - detailed	II.8
2.1 Section organization	II.8
2.2 CABLE csect - detail	II.8
2.2.1 Routines	II.8
2.2.2 Tables	II.10
2.3 The Data Base	II.13
2.4 WORKER csect - detail	II.17
2.4.1 Routines	II.17
2.4.2 Subroutines	II.20
Bibliography	II.23
Appendix A Flowcharts	II.A.1
Appendix B Reference to SYSREC source listing	II.B.1

SYSREC INTERNAL LOGIC MANUALPreface

This manual describes the internal operation and functional capabilities of the SYSREC program. It is intended for use by persons wishing to extend or modify the existing program implementation.

Prerequisites to the reading of this manual include:

- (1) knowledge of material presented in the SYSREC USER'S MANUAL.
- (2) basic knowledge of the Michigan Terminal System (MTS).
- (3) knowledge of 360/Assembler Language.

The manual is organized into two sections: Section 1 discusses SYSREC from a global point of view; Section 2 discusses the main routines and data structures employed in SYSREC. In addition appendices A and B provide respectively, flowcharts of the major routines and reference to a documented source listing of the program.

Section 1: PROGRAM LOGIC - OVERVIEW

1.1 Introduction

From a global viewpoint, the SYSREC program is divided into two logical phases: a command processing phase and a work phase.

Summarily the command processing phase accepts SYSREC commands, checks for correct syntax, sets up parameter lists and then, barring detection of errors to this point, passes control to the work phase. The detection of an error in the command processing phase usually requires the reissuing of a command in corrected form. The work phase executes the task defined by the issued command and in so doing uses the parameters passed from the command processing phase. Successful completion of the work phase causes control to pass back to the command processing phase thus enabling the program to accept the next input command.

Error detection in the work phase has results dependent upon the error encountered but in general control passes, via an error exit, back to the command processing phase thus prematurely terminating the work phase. The above two phases are realized in the program by two csects labelled CABLE and WORKER, representing the command processing and work phases respectively.

1.2 CABLE csect

The CABLE csect is that segment of SYSREC which processes SYSREC commands. This csect contains routines which:

- 1) read in a logical input line.
- 2) scan an input line for invalid symbols.
- 3) collect alphanumeric and numeric symbols which have been delimited by preassigned characters.

- 4) check the validity of a given command name.
- 5) process keywords and phrases associated with a given command.
- 6) check given commands for syntax errors.
- 7) set up parameter lists if necessary.
- 8) generate error messages.
- 9) pass control to the WORKER csect.

The overall flow of this csect is shown in Figure 1.

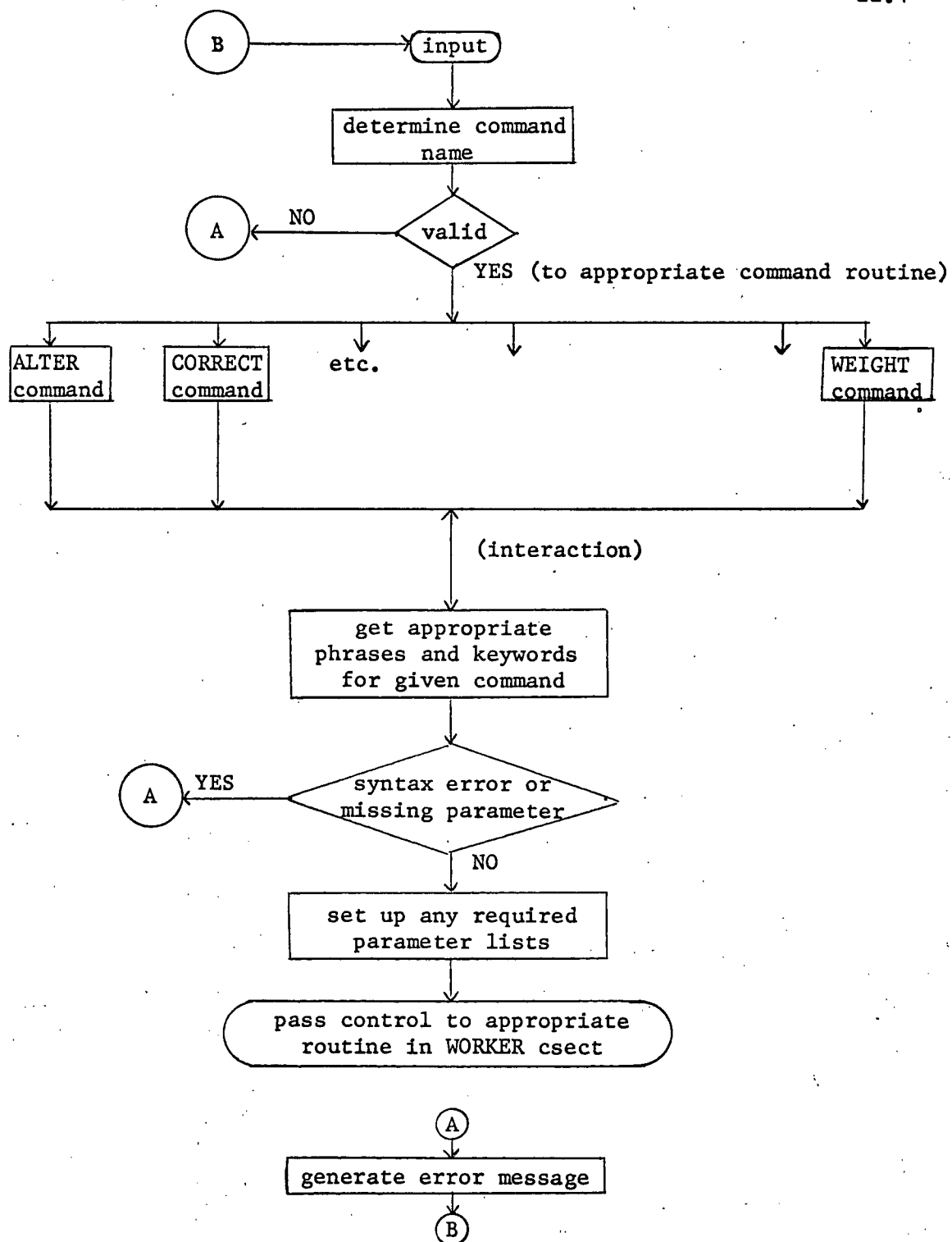


Figure 1

Thus, once the initial input procedure is complete and it has been determined that the command name given does in fact correspond to a SYSREC command, control passes to a routine corresponding to the command given. (In the program listing, these routines are each labelled in the form XXCP where XX denotes the first two letters of the corresponding command name. Thus ALTER — ALCP, DISCONN — DICP). These routines then call upon those specific subroutines which are required to process the parameters and keywords associated with the given command.

1.3 WORKER csect

The WORKER csect is that phase of the SYSREC program in which the task determined by an input command is executed. Control is passed to this csect from the CABLE csect. The WORKER csect contains routines which:

- 1) perform the functions defined by each SYSREC command.
- 2) perform minimum path routing.
- 3) read in existing systems and write out the current system.
- 4) allocate core to the SYSREC data structure.
- 5) for a given command, check the current data structure to ensure the task requested does not violate any limitations.
- 6) scan and update the data structure.
- 7) generate error messages.
- 8) pass control back to the CABLE csect.

The WORKER csect is similar to the CABLE csect in organization in that it contains separate routines corresponding to each of the allowable SYSREC commands. Control is passed to the appropriate one

II.6

of these routines from the CABLE csect after the latter has determined the command name and established any required parameter lists. Each of these separate routines in the WORKER csect then calls upon subroutines necessary to effect the given command. Thus the overall flow is:

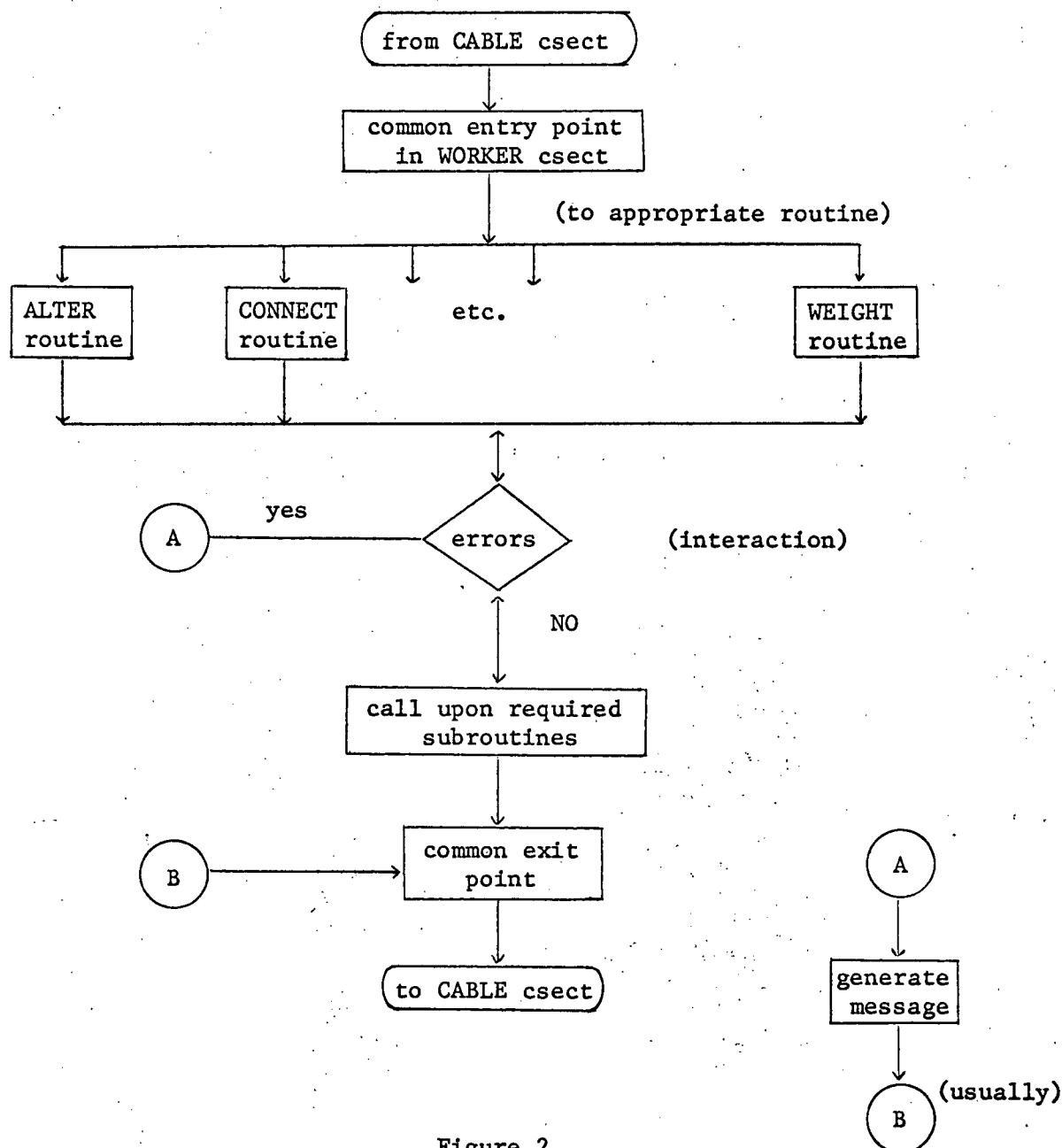


Figure 2

(In the program listing, the routines corresponding to commands are each labelled in the form XXR where XX denotes the first two letters of the command thus ALTER — ALR, CONNECT — COR).

Section 2: PROGRAM LOGIC - DETAILED

2.1 Section Organization

This section provides reference to the operation of the main routines/subroutines of the SYSREC program. In addition, the data base structure is discussed from a functional standpoint.

Section 2.2 of this chapter details those routines and major data elements found in the CABLE csect. Section 2.3 discusses the SYSREC data base and Section 2.4 describes the major routines of the WORKER csect and how they relate to the data base. It is intended that routine headings in the SYSREC source listing be read concurrently with the following sections. In particular, many of the lesser subroutines of the WORKER csect are referenced only by the source listing.

2.2 CABLE csect - detail

2.2.1 Routines

A) CPROC

Since both csects use the same base registers, the values in these bases must be reset upon each entry into a csect. Program termination is determined at this point from the value of 'ENDSW'. Command names are distinguished here.

B) LINE, GETLINE

These routines together read a logical input line which is composed of not more than three physical input lines.

C) GETNEXT, GETSYM

These are low level routines which are used to return respectively a single character, and a single symbol to higher level routines.

D) GETNAM GETTBS, GETKWS, GETNUM

These routines are used to process various phrases, parameters, and keywords associated with a given command name. They rely heavily on the GETNEXT and GETSYM routines. GETKWS temporarily passes control to routines which process the values associated with a given keyword. These routines, GETNAM, GETTBS etc. are generally called by a routine from (E).

E) CRCP, ALCP, WECF, RUCF, PUCF, EXCF, ROCP, COCP, DICP, TRCP, SUCF, DECF, LICP, MTCP, SECF, QUCP.

These routines correspond one for one with permissible SYSREC commands. The first two letters of each routine are the same as the first two letters of the command to which it corresponds. When a valid SYSREC command has been distinguished control passes to the corresponding routine (ie one of the above). This routine will then call lower level routines (see section D above) to collect the parameters associated with the specific command (if any). When all parameters associated with a given command have been determined, the routine passes control, indirectly, to an associated routine in the WORKER csect. An exception to this last statement is the MTCP routine which has no counterpart in the WORKER csect.

F) GETWGT, GETSL, GETCODE, GETDIM, GETLEN, GETDIR, GETDESC, GETSN, GETCN, GETTBN, ALL, PRPSIZE, TBSIZ, CVSIZ

These routines are temporarily passed control by GETKWS. They process the values corresponding to keyword parameters. When a particular keyword value has been processed, control passes back to GETKWS.

G) MESSAGE

This routine prints all messages generated by this csect. It is called via any routine in this csect which has detected the need for a message and returns control to that routine when the message has been printed. There is a similar routine in the WORKER csect. Messages are referred to by number when they are to be generated. The message number and the calling sequence to this routine are components of the NOTE macro definition.

2.2.2 Tables

A) COMTAB

This is a table of valid SYSREC commands. Since only the first two letters of a command are required to distinguish it from all other commands, only these two letters are needed in this table. The routine to which control should pass given a match between an inputted command and one of the valid commands follows the command name definition.

B) CHARTAB

This is a table of offsets corresponding to those EBCDIC symbols allowed by SYSREC. This table is used by the GETSYM routine.

C) CRETAB, RUNTAB, PUTTAB, EXTAB, ROTAB, CONTAB, DISTAB, TRTAB, SUMTAB, DESTAB, SETTAB

These jump tables define those keywords which are valid for a particular command. The first two letters of each table name are the same as the first two letters of the command for which the table defines allowable keywords. These tables are used by the GETKWS routine.

D) MSGTAB

This table contains all messages generated by SYSREC. The messages are in order according to their associated message number. Note that each of these messages is exactly 32 bytes long (possibly padded with blanks). This corresponds to the action of the MESSAGE routine.

E) PARLIST

This is the list area used to pass parameters extracted from the input stream to the WORKER csect.

Flow of CABLE csect

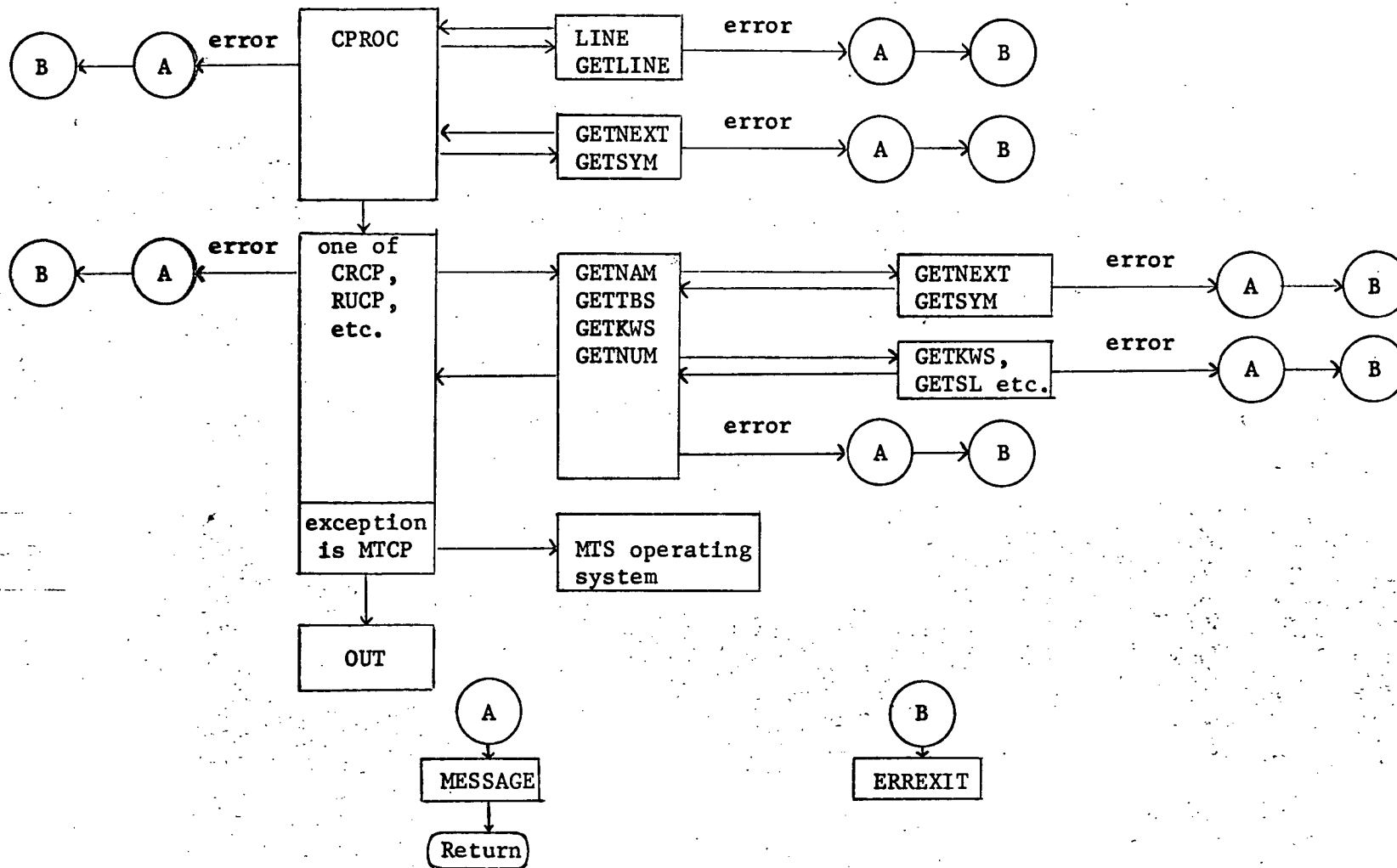


Figure 3

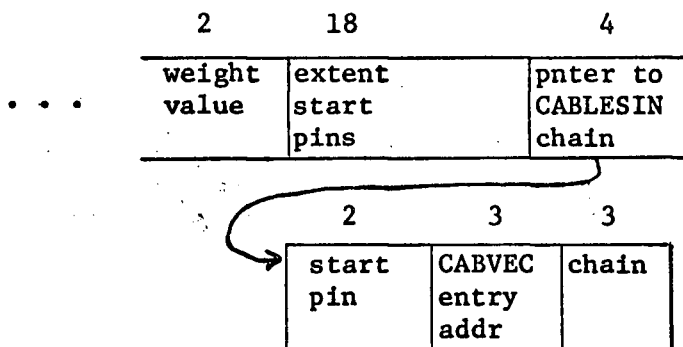
2.3 The Data Base

The major data items of the SYSREC program are the TBVEC (terminal board vector) and its associated JVECS (jumper vectors), the DVEC (distance vector), the CABVEC (cable vector) and its associated LVECS (line vectors) and the SIGVEC (signal vector) whose elements are pointed to by the hash table. The format and function of each of these items is discussed in this section.

1) TBVEC

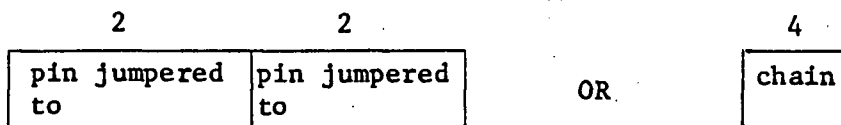
The TBVEC is composed of entries referring to created terminal boards. The format of such an entry is¹

8	1	3	4	4	2	2
NAME	no. of extns	JVEC addr	DVEC entry addr	DESC addr	no. of pins	no. of free pins



A JVEC is created for each entry in the TBVEC. The number of elements in the JVEC is equal to the number of pins associated with the terminal board plus the number of times the size of the terminal board has been altered. The JVEC format is

¹ Numbers above element fields refer to the number of bytes associated with that field.

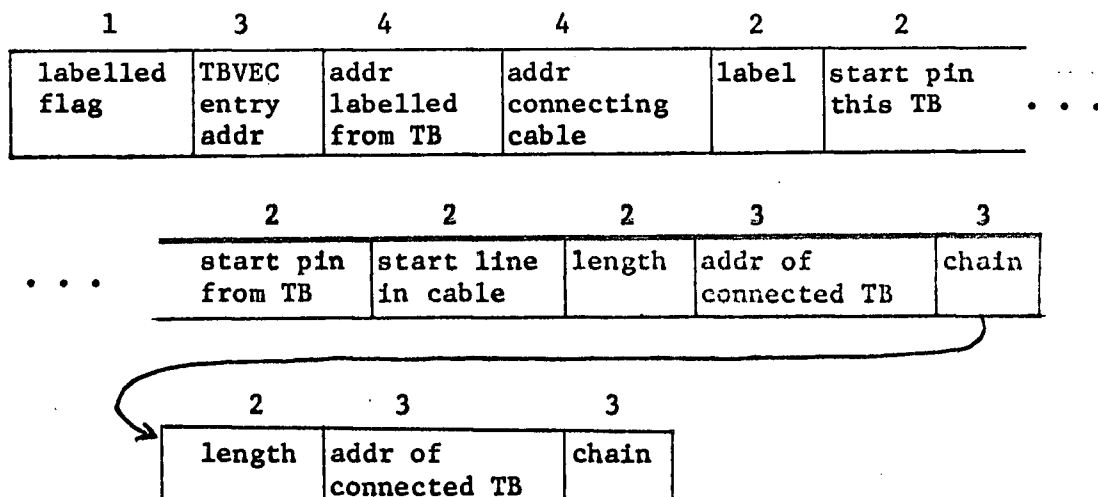


Thus a pin may be jumpered to at most two other pins. The maximum number of alterations (via ALTER command) permitted for a given terminal board is nine. This corresponds to the 'extent start pins' field of a TBVEC entry which allows nine half word pin values to be stored. These values represent the highest numbered pin added by each alteration. The number of alterations is stored in the 'no. of extns' field.

The entries in the 'CABLESIN' chain for a TBVEC entry record the addresses of cables which are connected to this terminal board and the lowest numbered pin to which the cable is connected. The chain is ordered by ascending pin number.

2) DVEC

The DVEC is composed of elements which relate terminal board interconnections. There is one element in the DVEC for each element in the TBVEC. Additionally, the DVEC is used to temporarily record a minimum path signal route. The format of a DVEC entry is



The chain of terminal board interconnections records the addresses of the terminal boards to which the terminal board corresponding to this DVEC element is connected and the length to that terminal board. The chain is unordered. The remainder of the fields in each DVEC element are used to pass signal route information between the minimum path routing subroutine (ROOT) and the routine which stores this information in the data base (PUR).

3) CABVEC

The CABVEC is composed of entries corresponding to cables in the configuration. This format of a CABVEC element is

8	4	4	1	3	4	2	2	
NAME	'A' TB addr	'B' TB addr	code value	LVEC addr	desc addr	cable length	no. of lines	...
	2	1	1					
...	no. of free lines	removed flag	reserved for expansion					

Corresponding to each element in the CABVEC, there exists a LVEC. The number of elements in each LVEC equals the number of lines in the associated cable. The LVEC records which signal, if any is carried on each line of the cable and the terminal board pin numbers to which each line is connected. The LVEC format is

1	3	2	2	2	2
status flag	SIGVEC entry addr	'A' TB pin	'B' TB pin	signal sub-label	reserved for expansion

The 'A' and 'B' TB addresses in a CABVEC element point to the names of the terminal boards which this cable connects. These

addresses also represent the terminal boards to which pins on the 'A' and 'B' pin fields of each LVEC belong. An entry is considered deleted from the CABVEC if its 'removed flag' field is appropriately set.

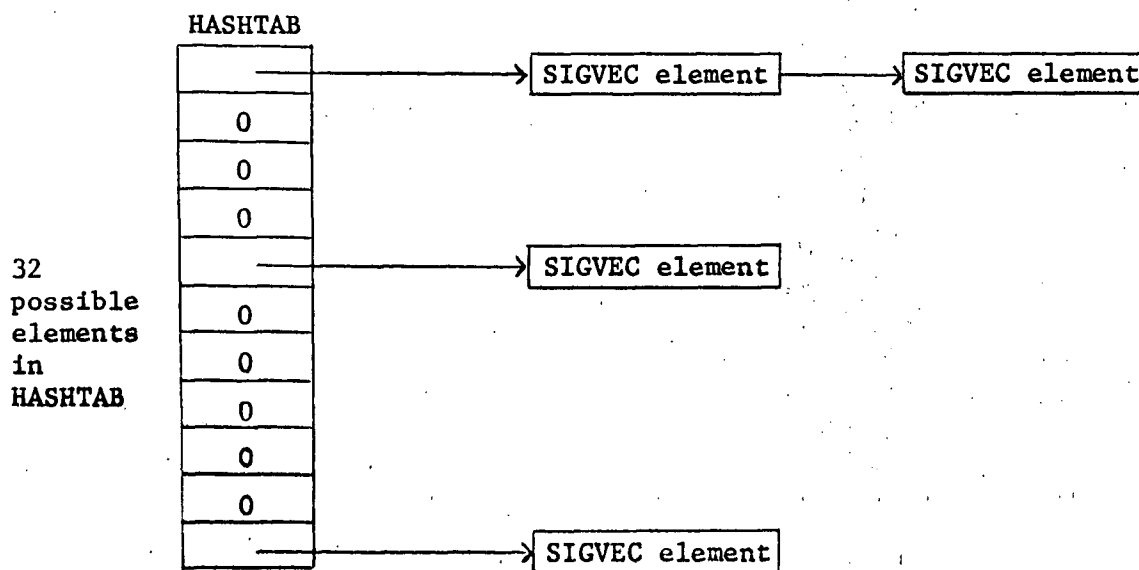
4) SIGVEC

The SIGVEC is composed of at most 32 separate chains of elements. Each element in a chain contains information about a currently existing signal. The format of such an element is

8	4	4	4	2	2	2
NAME	addr of start cable	desc addr	chain	dimension	length	date

The location of a SIGVEC element is determined by hashing the corresponding signal name and then using the hash table (HASHTAB) to determine to which chain the element belongs. Entries are added to the tail of this particular chain as required.

This may be illustrated by:



Thus all the signal names in a given chain hash to the same value. Each SIGVEC element contains a pointer to a CABVEC element which is considered to be the cable in which the signal originated.

2.4 WORKER csect - detail

2.4.1 Routines¹

A) CRR *

This routine creates an entry for a terminal board defined in the TBVEC. A corresponding DVEC entry is also made and the JVEC for this terminal board is allocated and initialized.

B) RUR *

This routine runs a cable between two existing terminal boards. The subroutine PFRE is called to determine which pins the cable may be attached to on each terminal board.

C) ROR *

This routine determines the feasibility of a signal route.

D) PUR *

This routine puts a signal between two existing terminal boards. PUR calls the subroutine ROOT to perform signal routing on a minimum path length basis. The ROOT subroutine, if successful, returns the DVEC set up such that PUR may use the information stored in it to trace out the assigned route. The DVEC is set up so that starting from TB2 (the "to" terminal board), each terminal board points to the terminal board that precedes it on this route. In addition, the DVEC contains information as to which terminal board

¹ an * denotes a routine for which a flowchart can be found in Appendix A.

pins, cables and cable lines are part of the route.

In storing the signal route in the SYSREC data base, PUR first ensures the route can be stored, and then if in fact it can, stores the route. This action avoids having to remove part of a stored signal should it be found that there is insufficient jumper space at some point to continue storing this route.

E) EXR *

This routine extends a signal from one terminal board to another. Depending upon the setting of the DIRECT keyword, this routine calls ROOT and then a subsection of PUR to complete the extension, or it attempts the extension on its own.

F) ALR *

This routine is used to alter the number of pins assigned to a terminal board. In increasing the number of pins on a board, this routine extends the JVEC chain for that board.

G) COR *

This routine puts a signal along a specified cable. The cable is checked to ensure it has sufficient free lines to accommodate the signal width.

H) DIR *

This routine is used to disconnect cables or signals. If a cable is to be disconnected, all signals running in that cable are also disconnected. To disconnect a signal this routine calls the TRAC subroutine with an appropriate flag setting. Entries in the SIGVEC are removed by DIR. To disconnect a cable once all associated signals have been disconnected, this routine sets a flag in the CABVEC entry for that

cable and then resets all lines between this cable and connected terminal boards.

I) TRR *

This routine traces three types of elements: signals, a terminal board pin or cable lines. Signals are traced by the TRAC subroutine while pin and line tracings are determined by TRR itself.

J) LIR *

This routine lists the names of all entities within a specified class (ie signals, terminal boards, cables).

K) SUR *

This routine summarizes individual signals, terminal boards, cables, or all the elements within one of the preceding classes, or all the elements within all three classes.

L) WER

This routine alters the weight value associated with a terminal board.

M) DER

This routine appends a descriptive phrase to a signal, terminal board or cable. The given phrase replaces any existing phrase.

N) SER

This routine is used to set a new maximum number of elements in either the TBVEC or CABVEC. Since this routine simply alters the count of maximum allowable elements in either vector (ie change in TBV#JCB or CV#JCB) but does not allocate additional memory space to a vector, continuing to use SYSREC after issuing a SET command could

result in overwriting of data. Hence, to avoid this, SER passes control to the QUR routine when it is finished altering the element count and QUR proceeds to terminate execution. When this configuration is specified as input for a PAR=CONTINUE run, the RESIN routine will then allocate space to the vector based on the maximum element count.

O) QUR *

This routine terminates execution. It is entered as a result of either the QUIT or SET command being issued. QUR (via FNAME) prompts the user to define a file name and then attempts to create a sequential file with the given name. The size of the file created is a function of the cumulative number of pages SYSREC has obtained since execution on this configuration began (GSCOUNT). Error procedures exist for the case that a file with the specified name cannot be created.

When a file has been successfully created, QUR temporarily passes control to the SAVE routine which writes the current configuration into that file.

2.4.2 Subroutines

These are the major subroutines of the WORKER csect which are called directly or indirectly by those routines discussed in Section 2.4.1.

A) SAVE *

This subroutine saves the current configuration in a file, the name of which is determined by QUR. The elements of the data base are saved in the following order: CABVEC, TBVEC, DVEC, SIGVEC.

Each element of these vectors is written as a single logical record after all address fields in that element have been altered to relocatable format. Address fields pointing to chains or other external elements (ie descriptions) contain a count of the number of elements or bytes in the chain. All chains and external elements (including JVEC and LVEC) are saved immediately following the CABVEC, TBVEC, DVEC or SIGVEC entry to which they correspond.

B) RES

This subroutine restores a previously created configuration from a specified input file. Elements are restored in exactly the same order as they were saved by the SAVE subroutine.

C) PFRE *

This subroutine is called by RUR to determine which pins on a terminal board a cable of specified dimension may be connected to. The routine accepts user requested starting pins or if none are given, it attempts to find a suitable subset of pins.

D) ROOT *, PINCHK1 *, PINCHK5 *

These routines are the essence of the minimum path routing algorithm. ROOT executes the algorithm while calling PINCHK1 and PINCHK5 to establish the set of all possible routes for given signal. PINCHK1 determines if it is possible to start or terminate the signal at given terminal boards and start pins. PINCHK5 returns to ROOT cables emanating from a terminal board. The returned cables are such that they are of sufficient free lines to accommodate the signal. The ROOT routine is the implementation of the algorithm described in the SYSREC USER'S MANUAL.

E) TBCHN *

This subroutine is called by RUR to record a cable emanating from a terminal board. The routine makes an ordered entry into the CABLESIN chain of a TBVEC element. Entries in the chain are ordered by ascending connected start pin number (see discussion of TBVEC in Section 2.3).

F) TRAC *

This subroutine extracts from the data base a prerecorded signal route. This route is either written out or disconnected depending upon whether TRR or DIR is the calling routine.

The remaining subroutines employed by the WORKER csect may be referred to via the program source listing (see Appendix B).

BIBLIOGRAPHY


1. Dijkstra, E. W., "A Note on Two Problems in Connexion with Graphs", Numerische Mathematik 1, 269-271 (1959).
2. Dreyfus, S. E., "An Appraisal of Some Shortest-Path Algorithms", Operations Research 17, 395-412 (1969).
3. IBM System/360 Operating System: Assembler Language, Form C28-6514-5.
4. IBM System/360 Principles of Operation, Form A22-6821-7.
5. Nicholson, T. A. J., "Finding the Shortest Route Between Two Points in a Network", Computer Journal 9, 275-280 (1966).
6. Walsh, D., A Guide for Software Documentation, Inter-ACT, New York, N.Y. (1969).
7. Whiting, P. D. and J. A. Hillier, "A Method for Finding the Shortest Route through a Road Network", Operational Research Quarterly 11, 37-40 (1960).

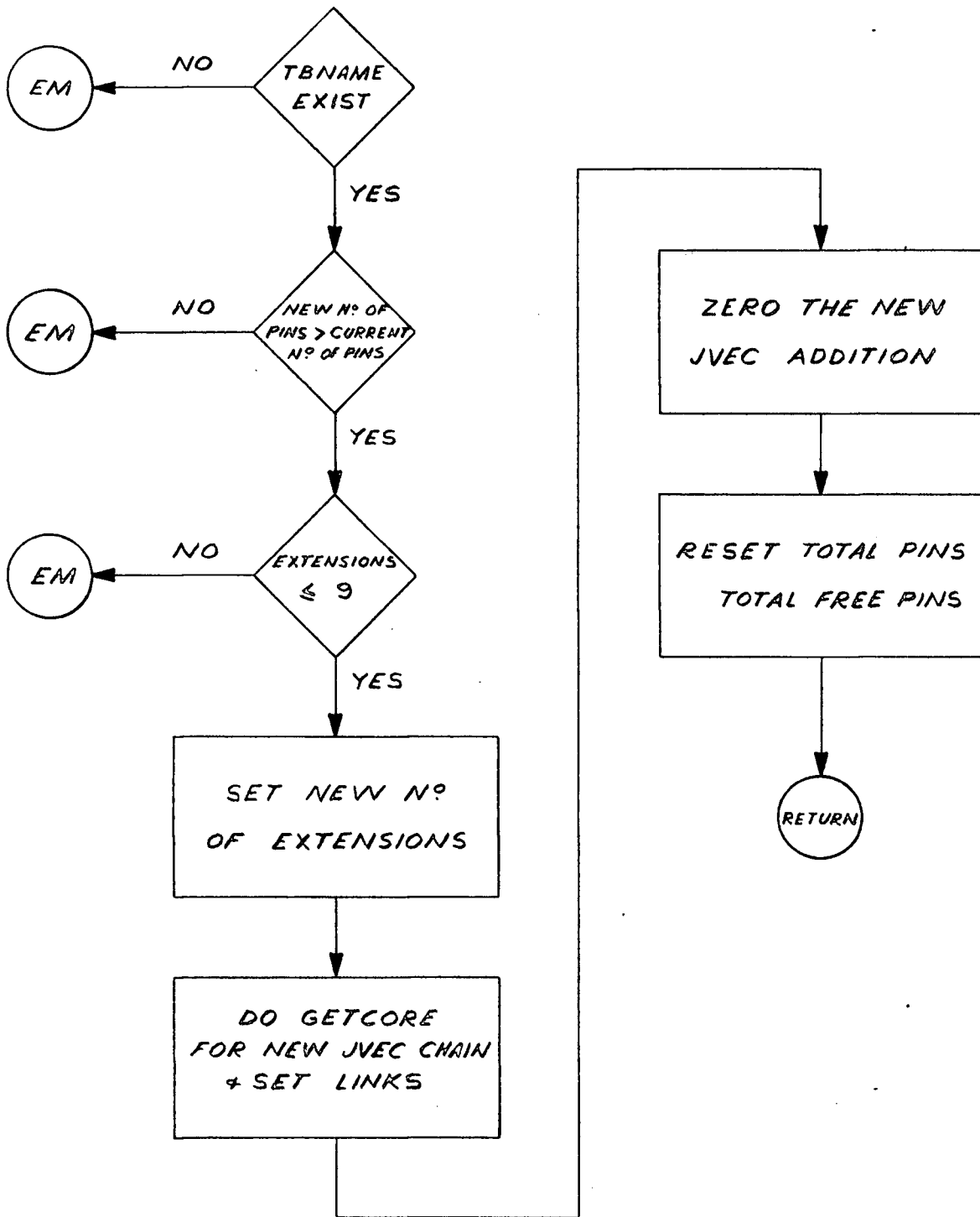
APPENDIX A

Flowcharts

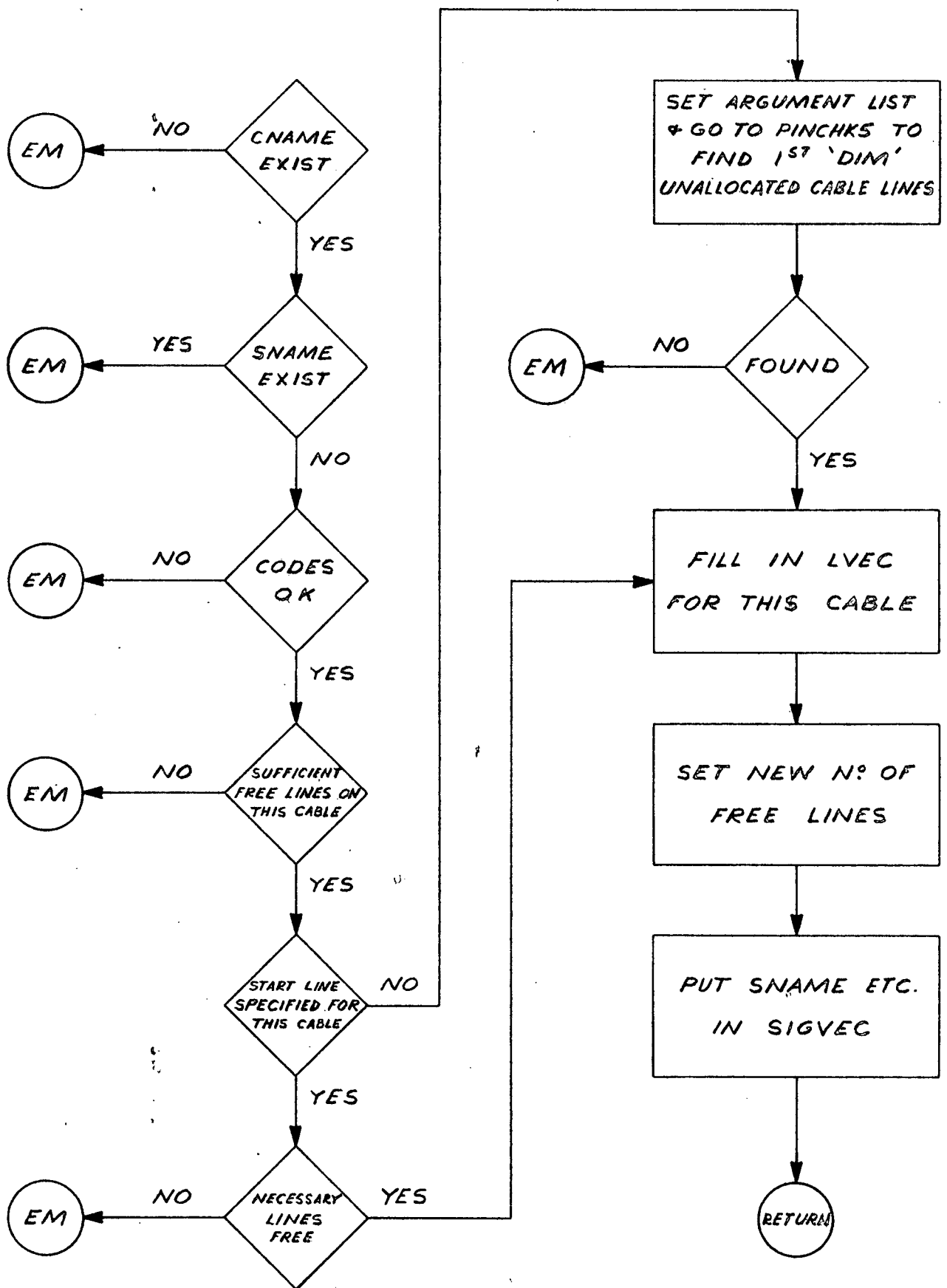
This appendix contains flowcharts of the major routines of the SYSREC program. Flowcharts for the following routines appear in the order indicated.

- | | |
|---------|-------------|
| 1. CRR | 11. SUR |
| 2. RUR | 12. QUR |
| 3. ROR | 13. SAVE |
| 4. PUR | 14. PFRE |
| 5. EXR | 15. PINCHK1 |
| 6. ALR | 16. PINCHK5 |
| 7. COR | 17. ROOT |
| 8. DIR | 18. TBCHN |
| 9. TRR | 19. TRAC |
| 10. LIR | |

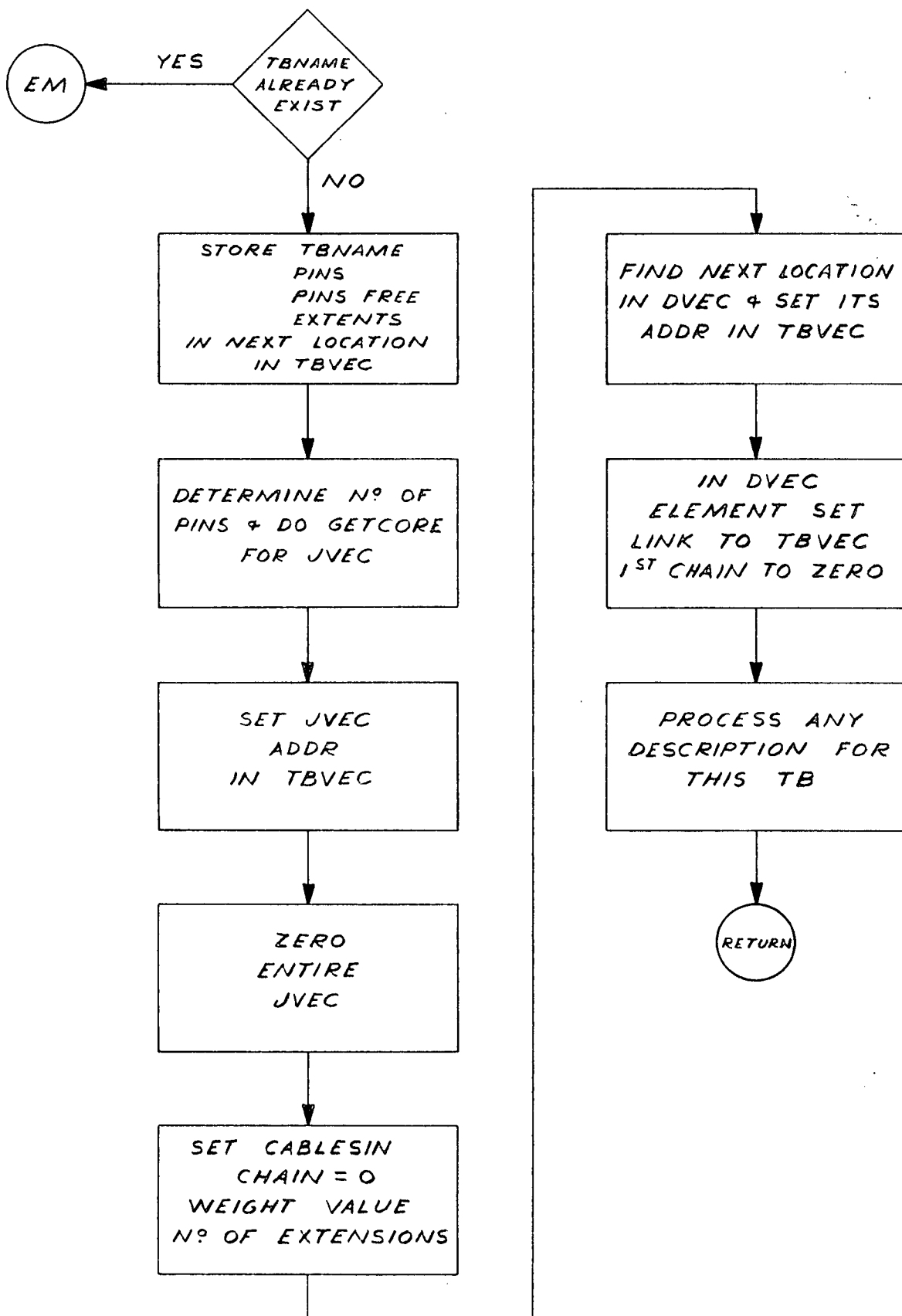
Note: The symbol  denotes a message generating and/or error exit routine.



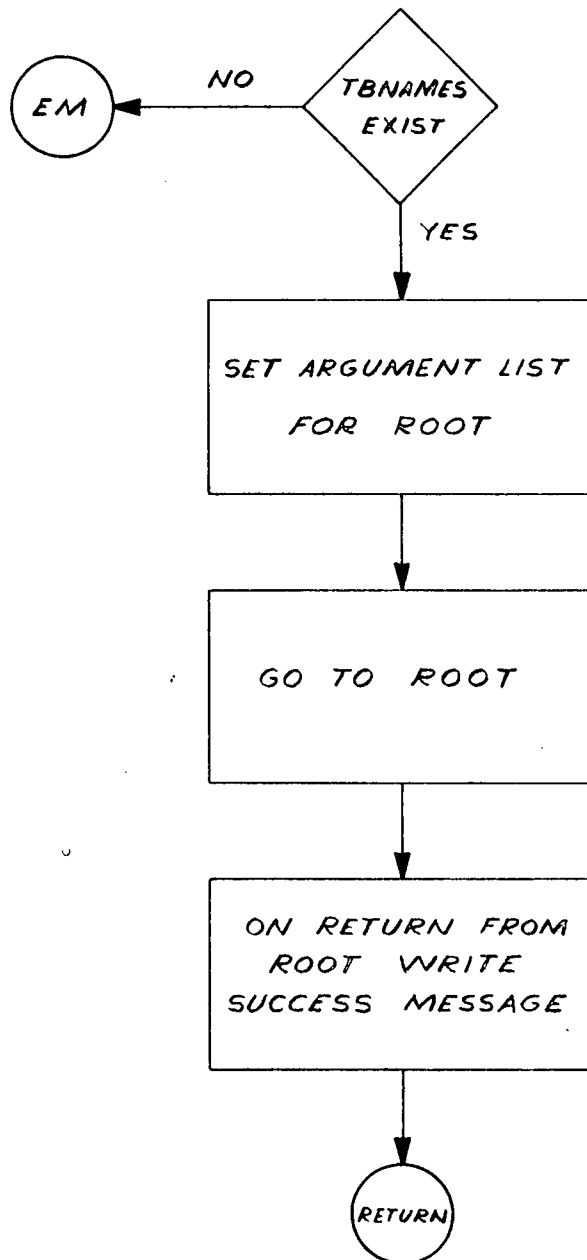
ALR FLOWCHART



COR FLOWCHART

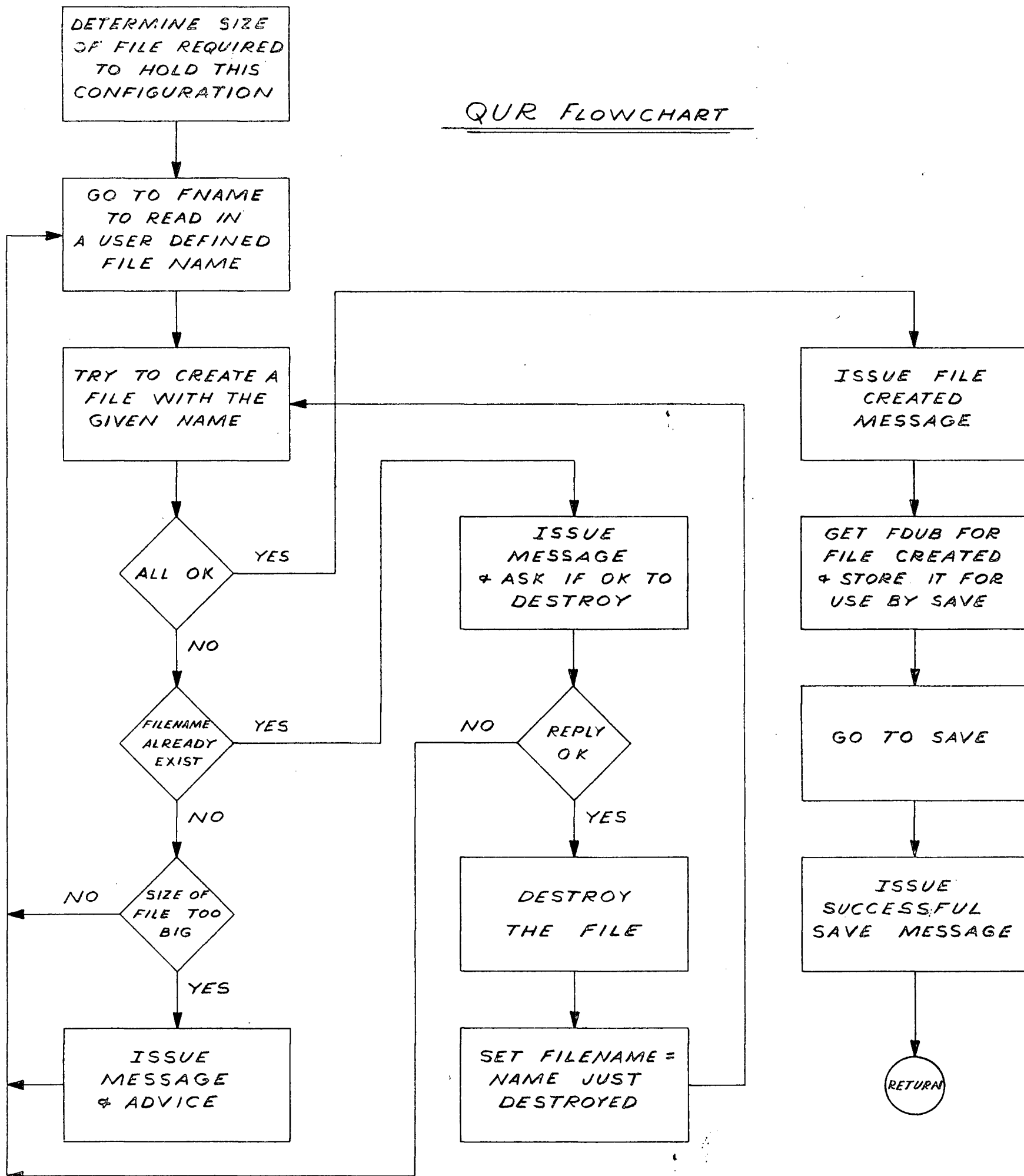


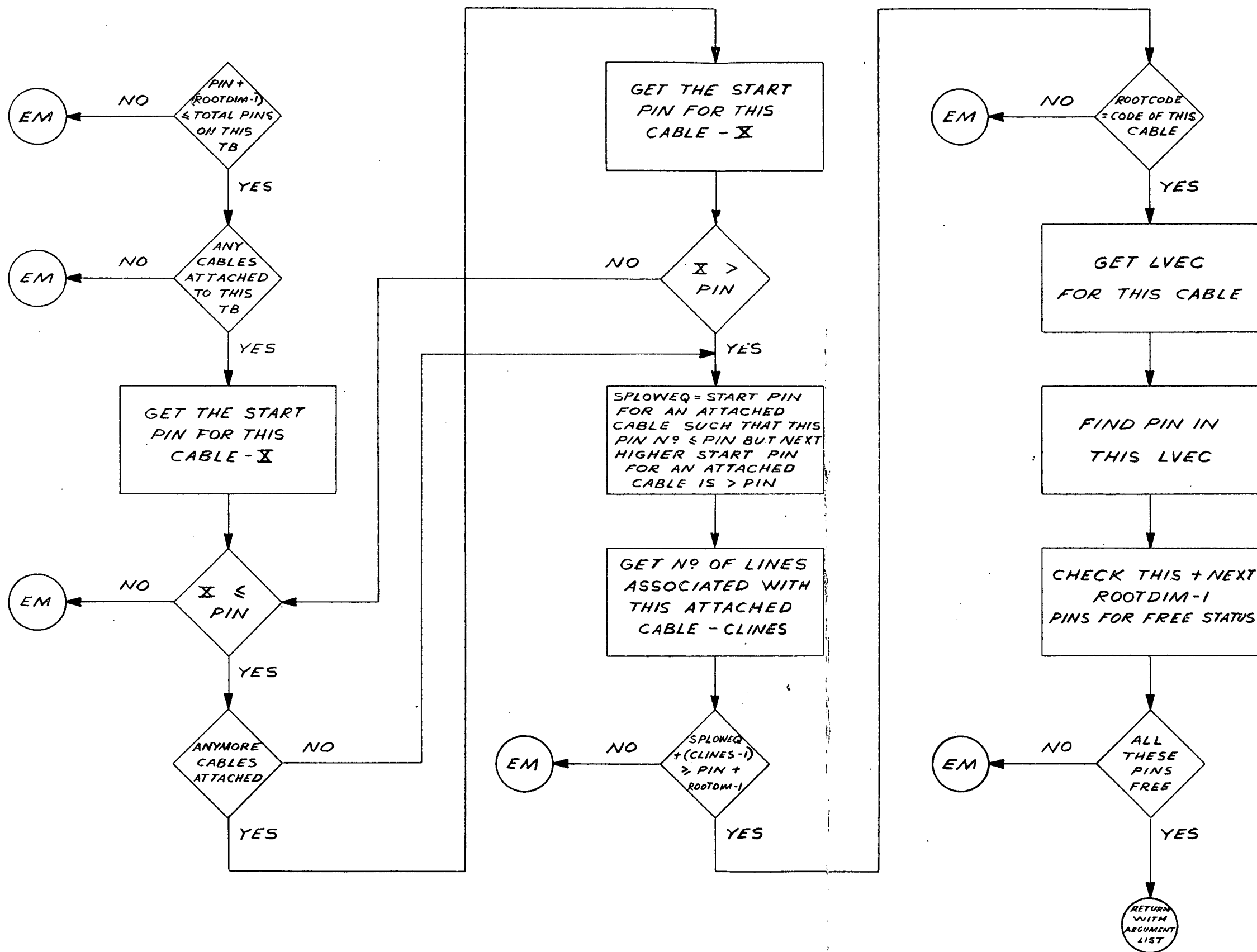
CRR FLOWCHART



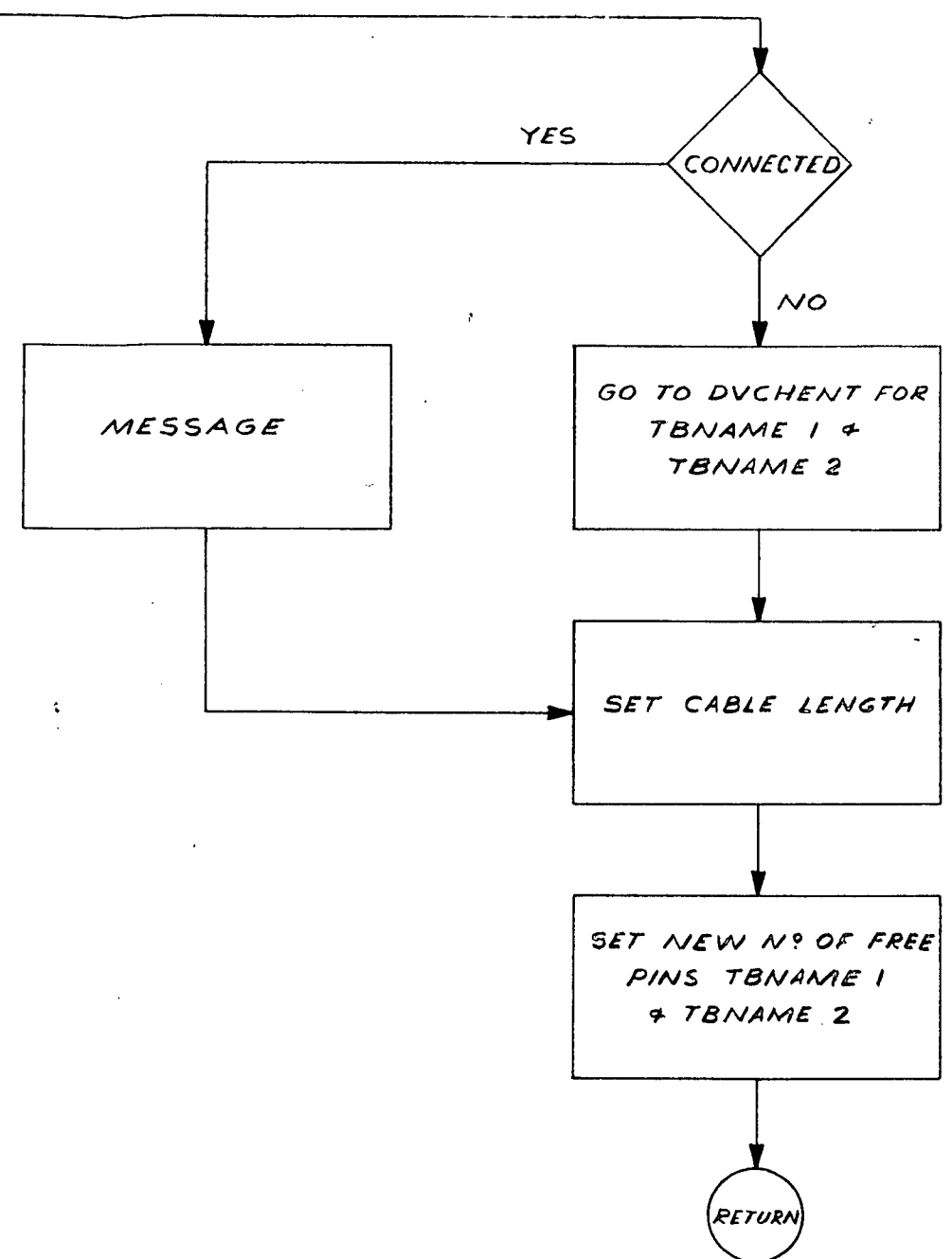
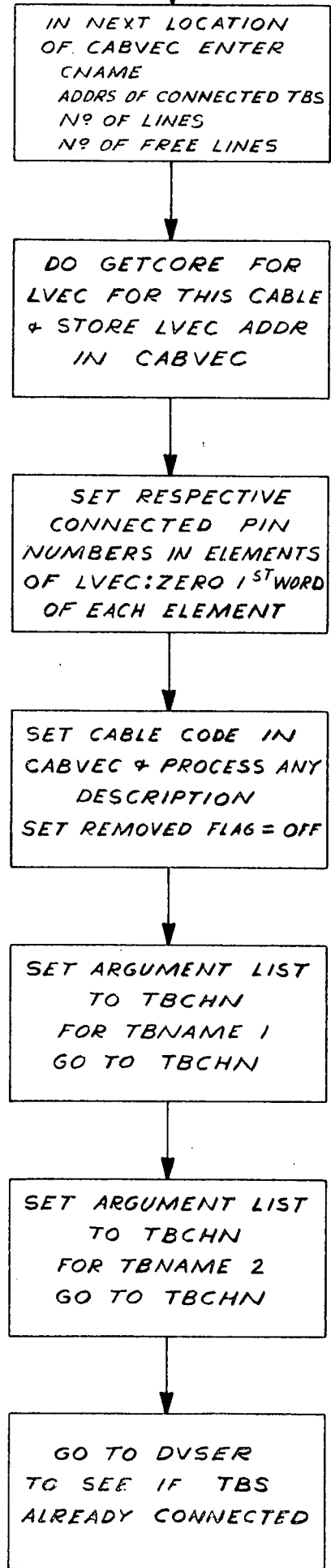
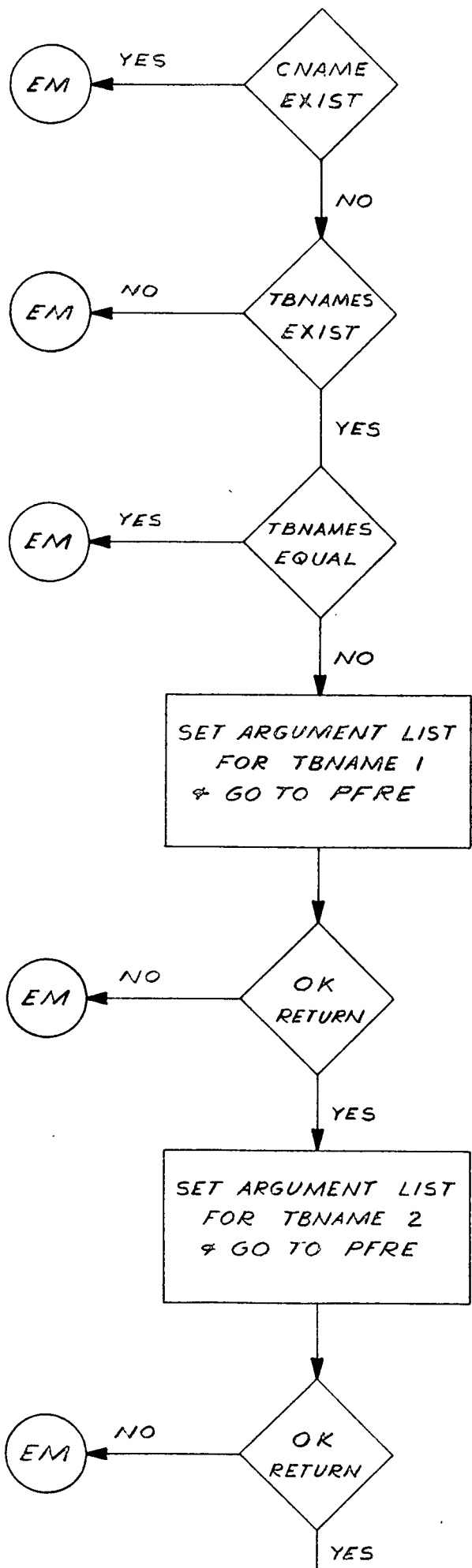
ROR FLOWCHART

QUR FLOWCHART

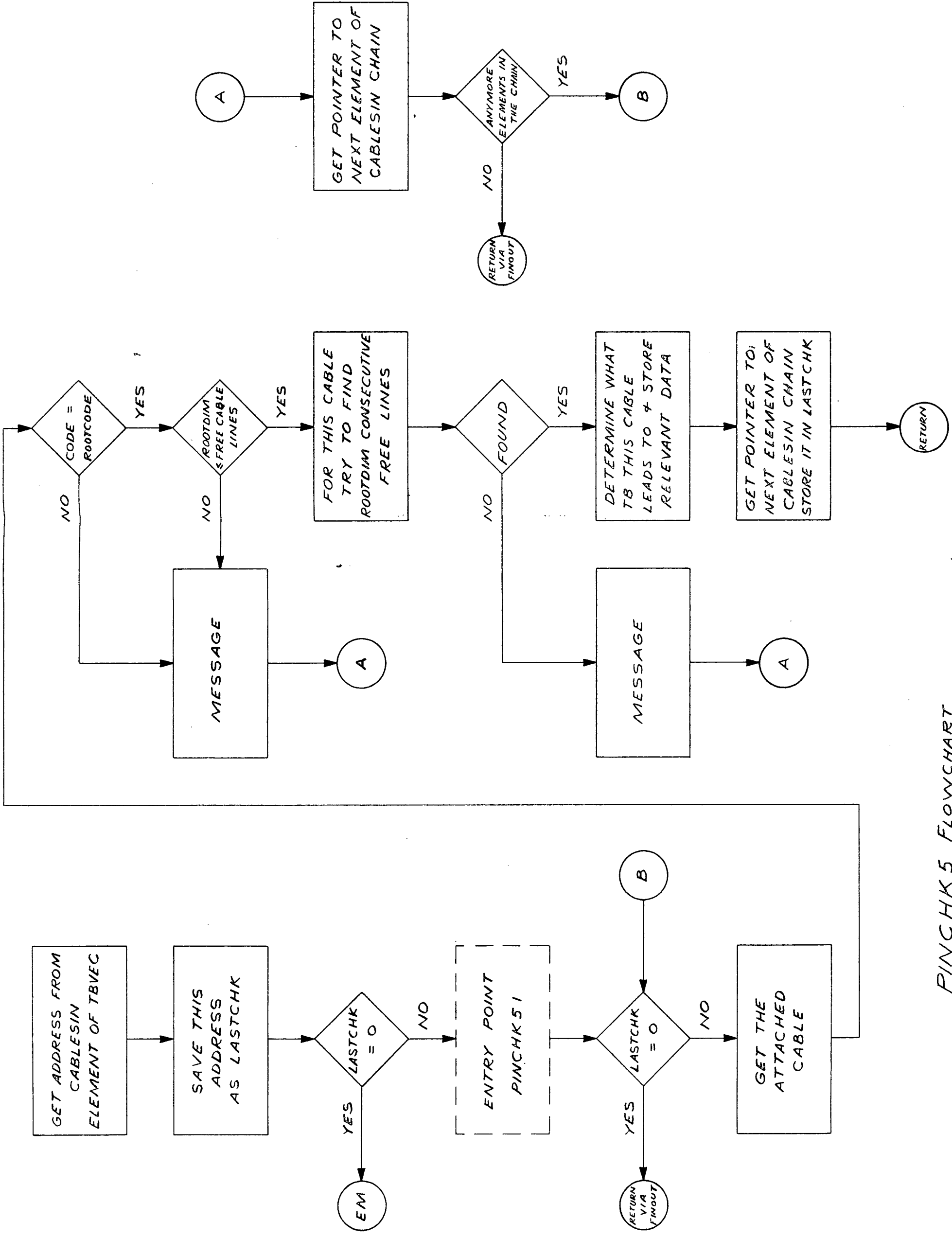




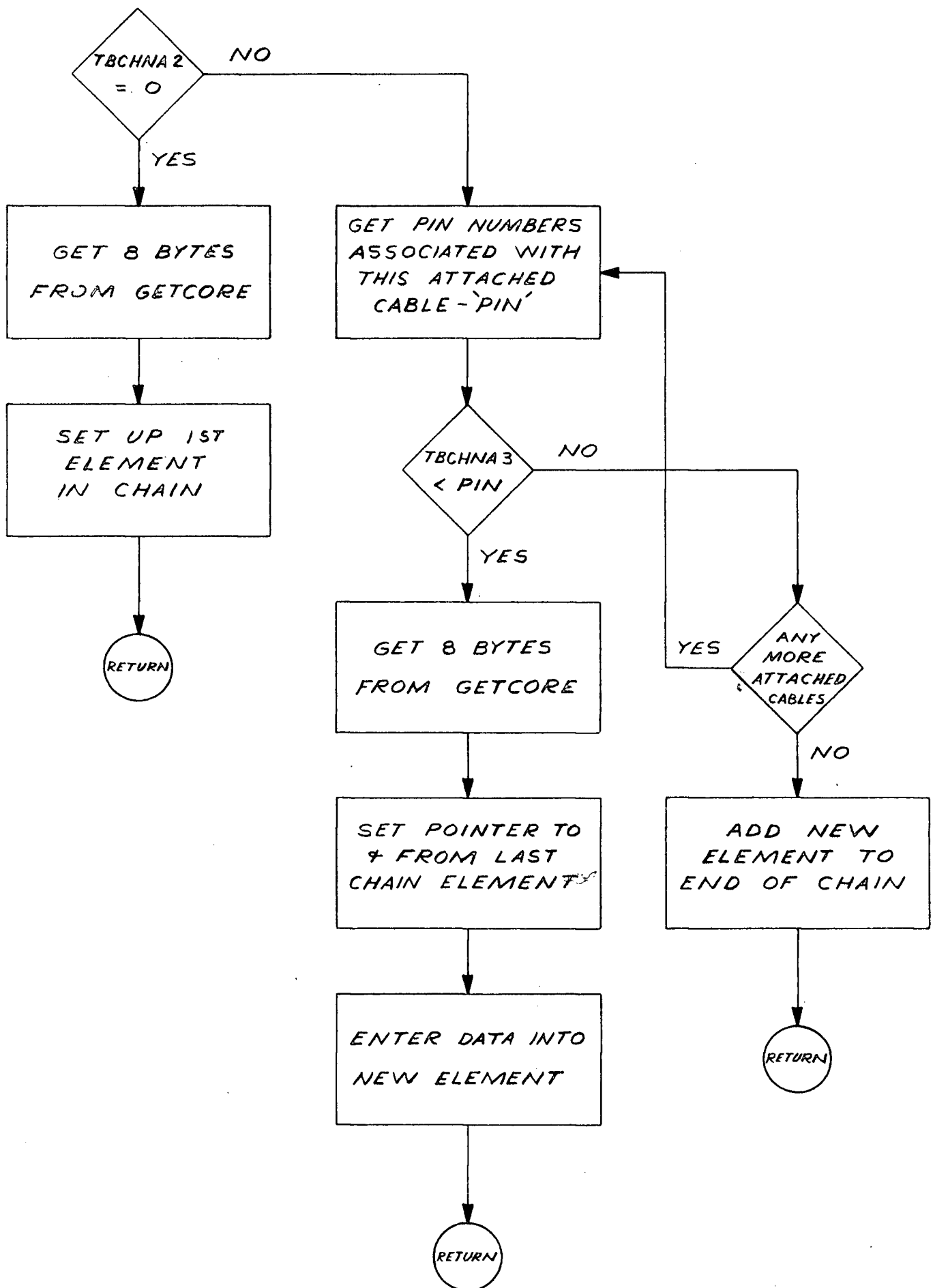
PINCHK1 FLOWCHART



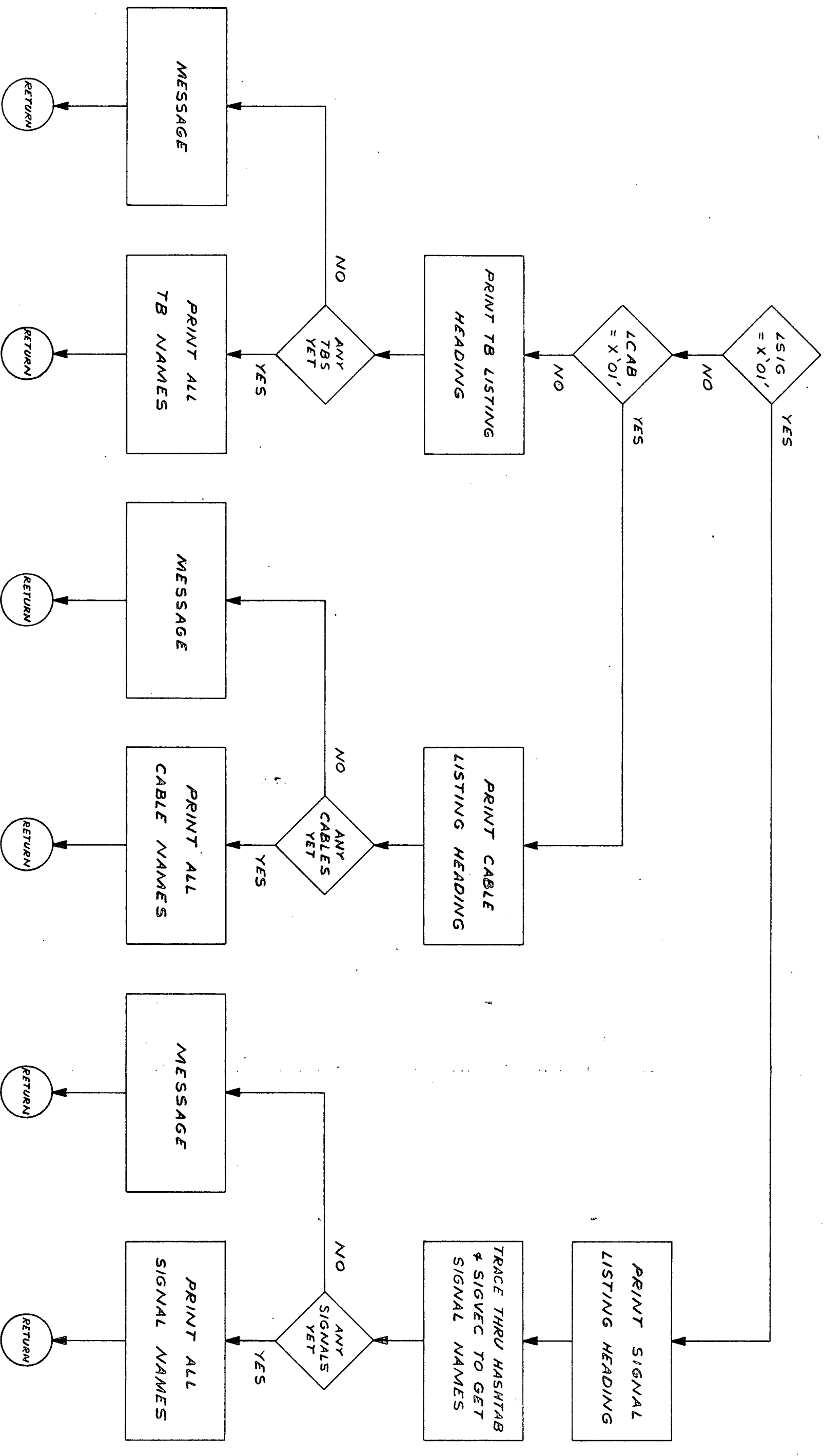
RUR FLOWCHART



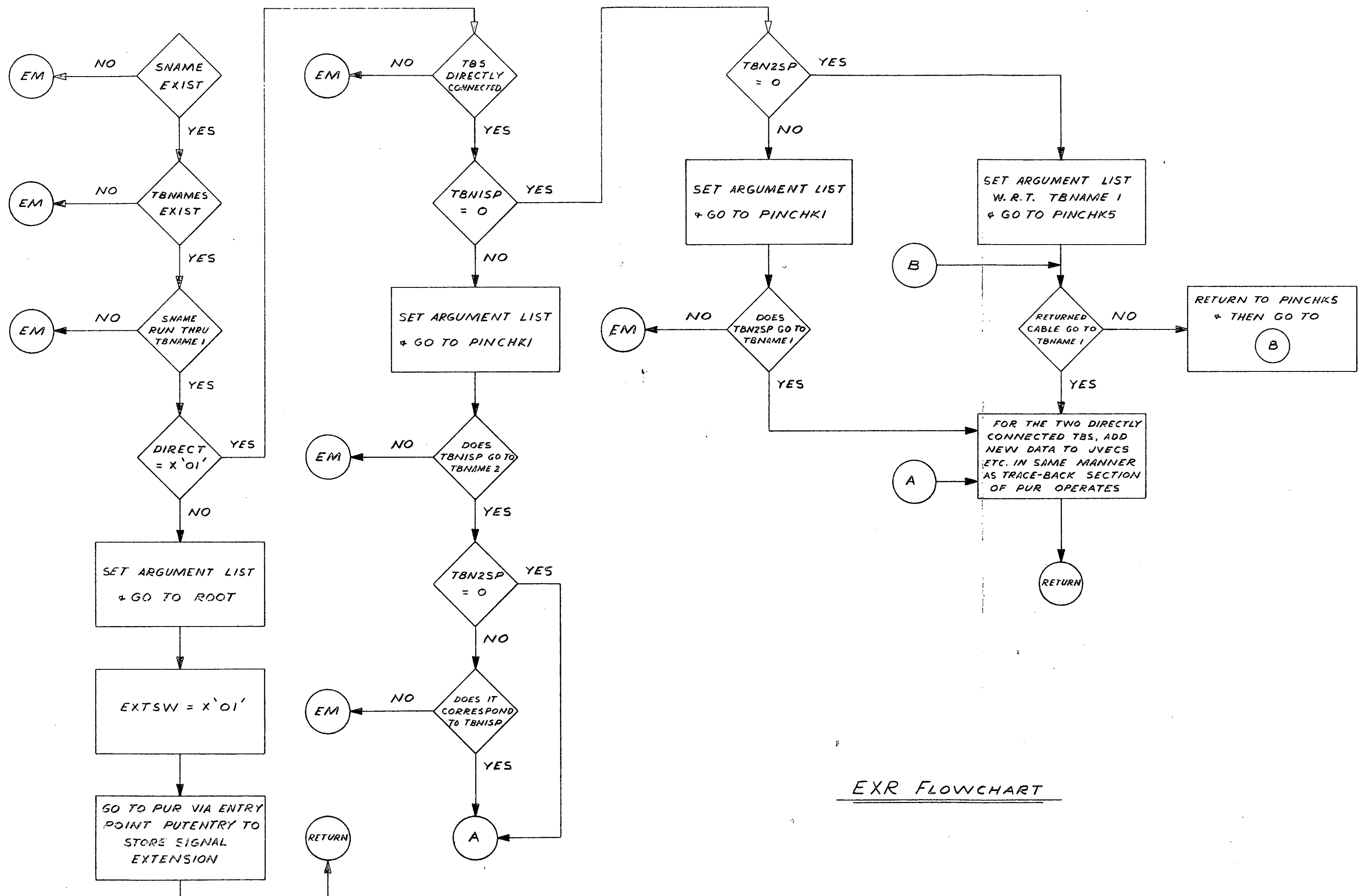
PINCHK5 FLOWCHART



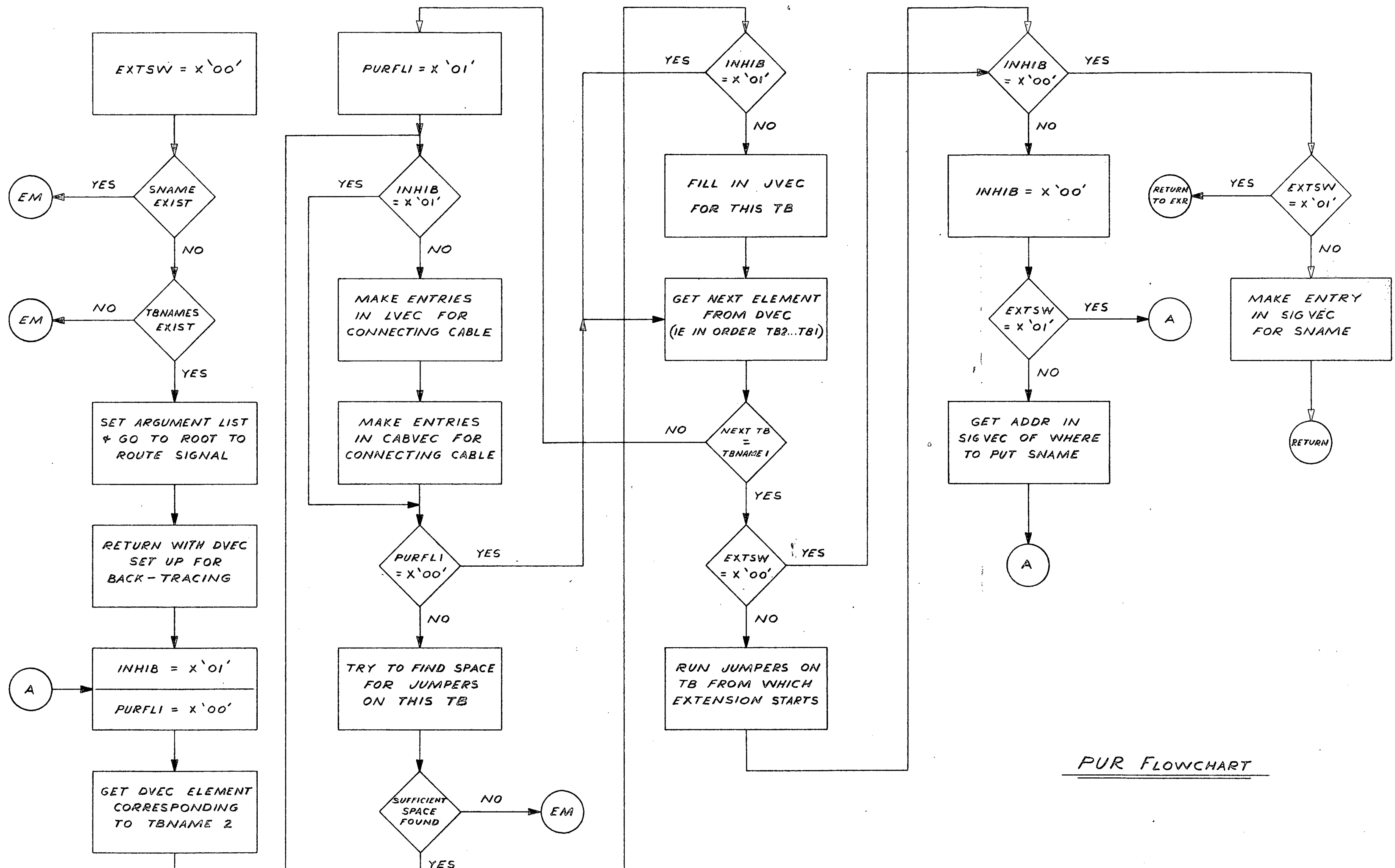
TBCHN FLOWCHART

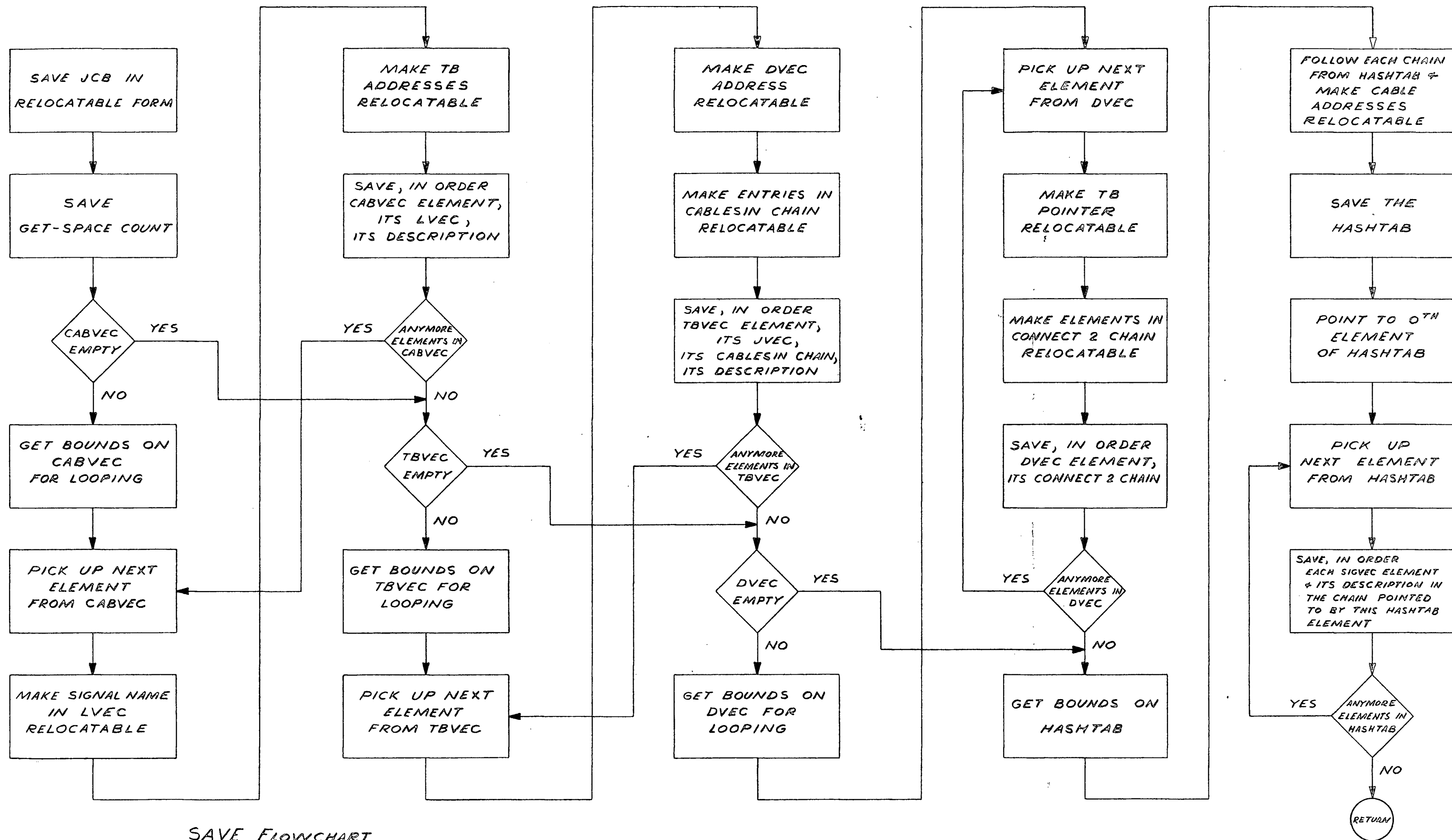


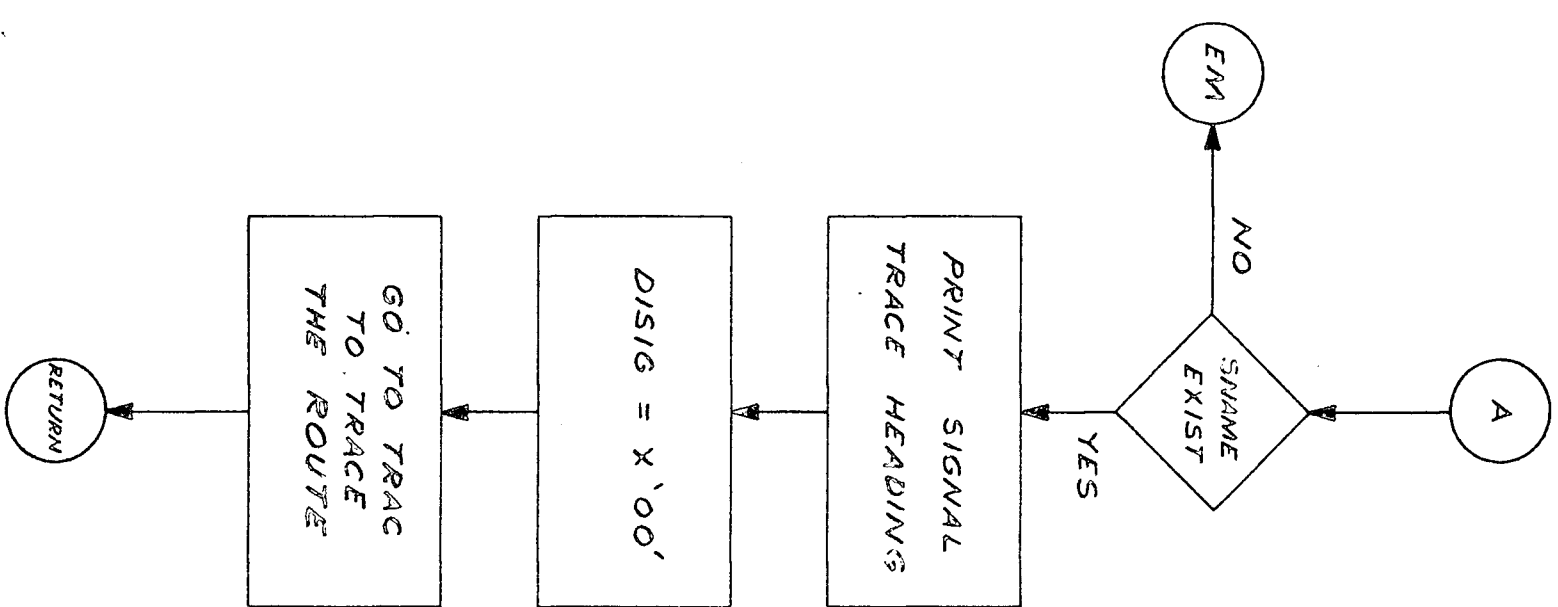
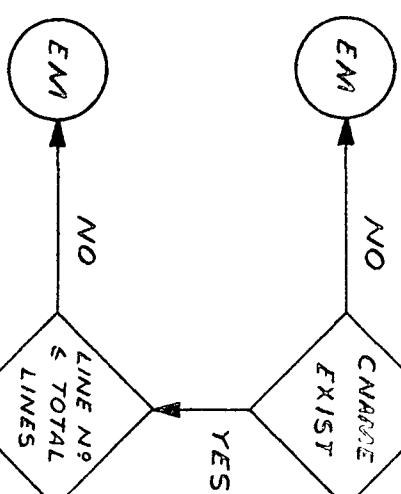
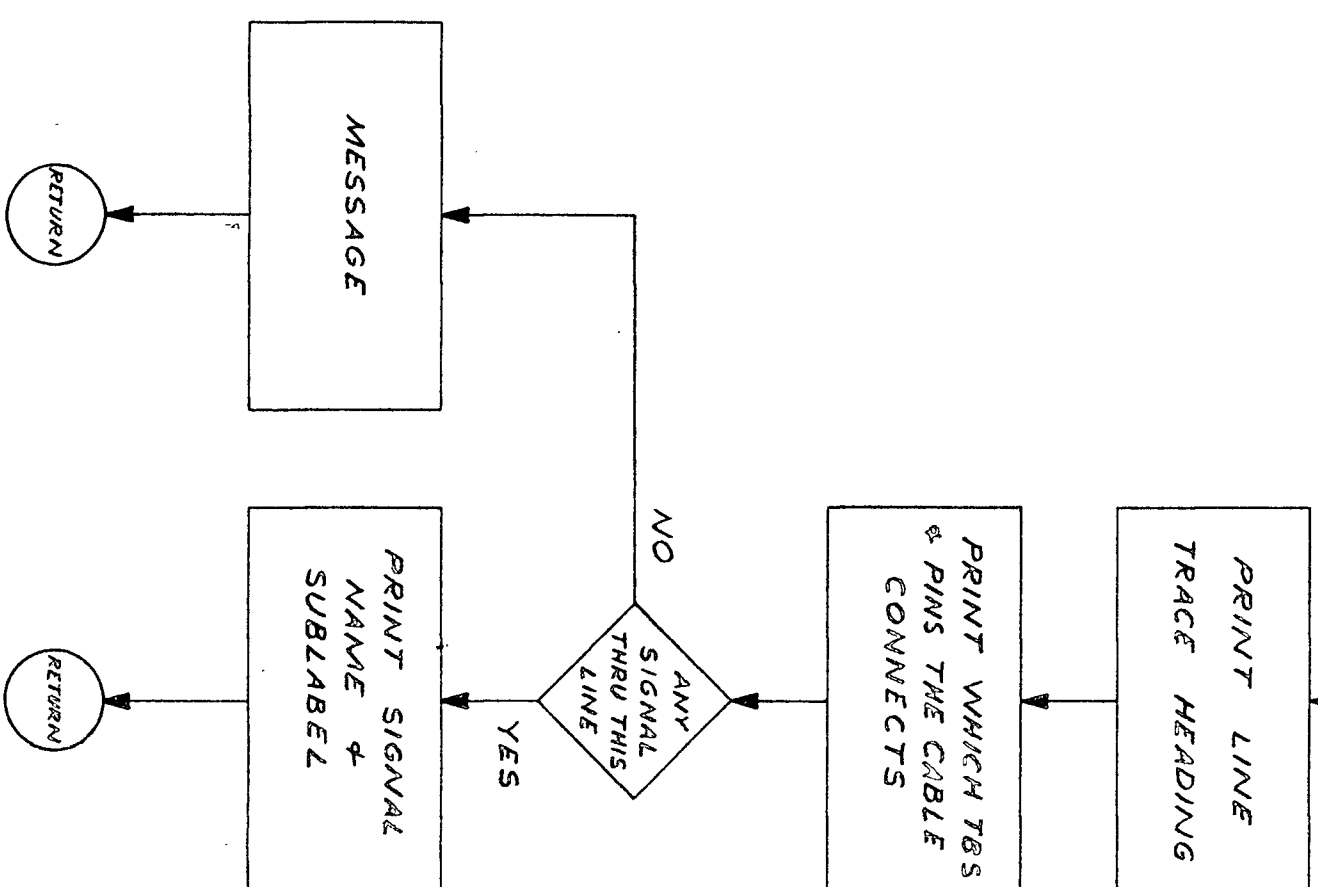
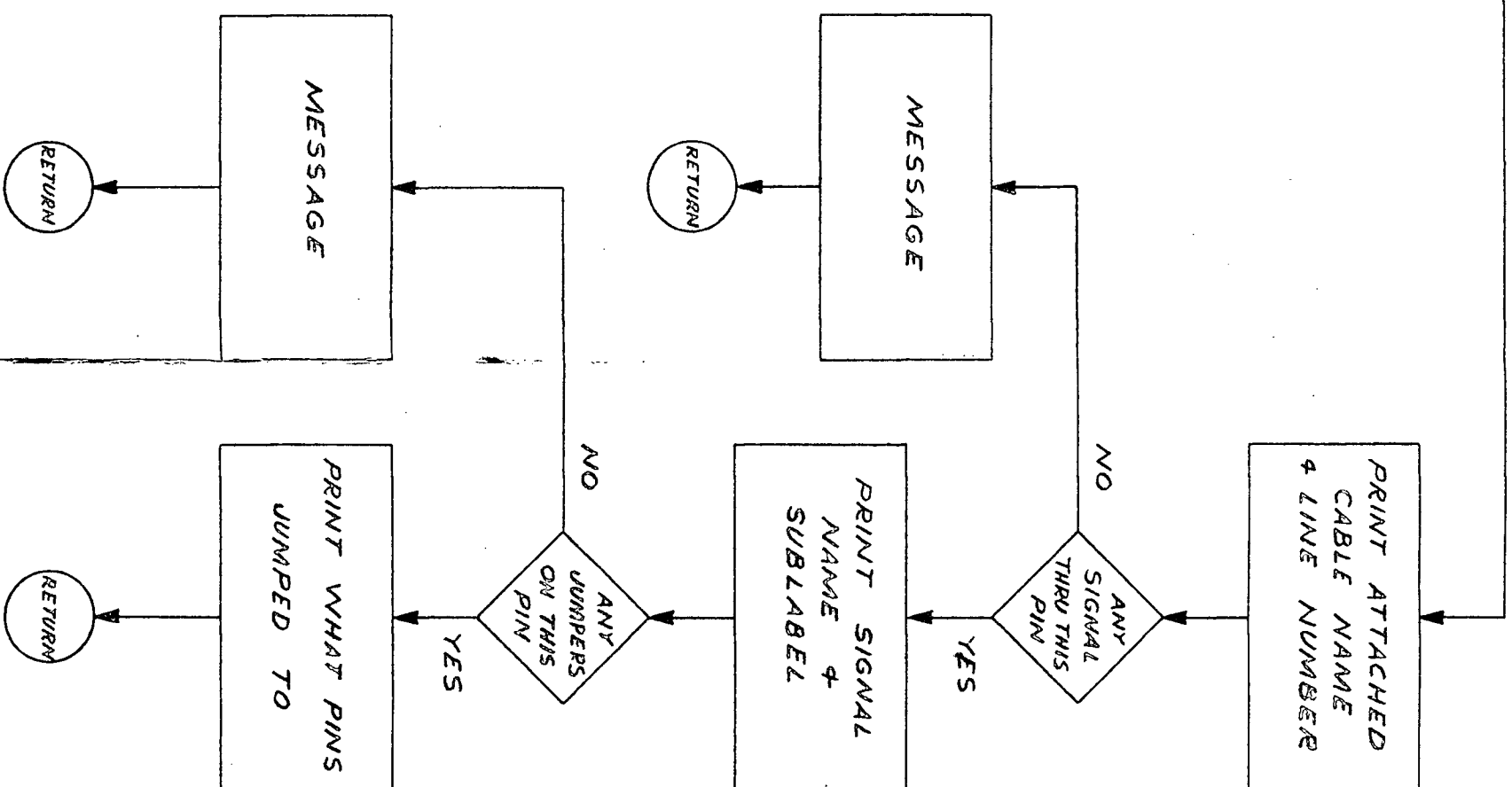
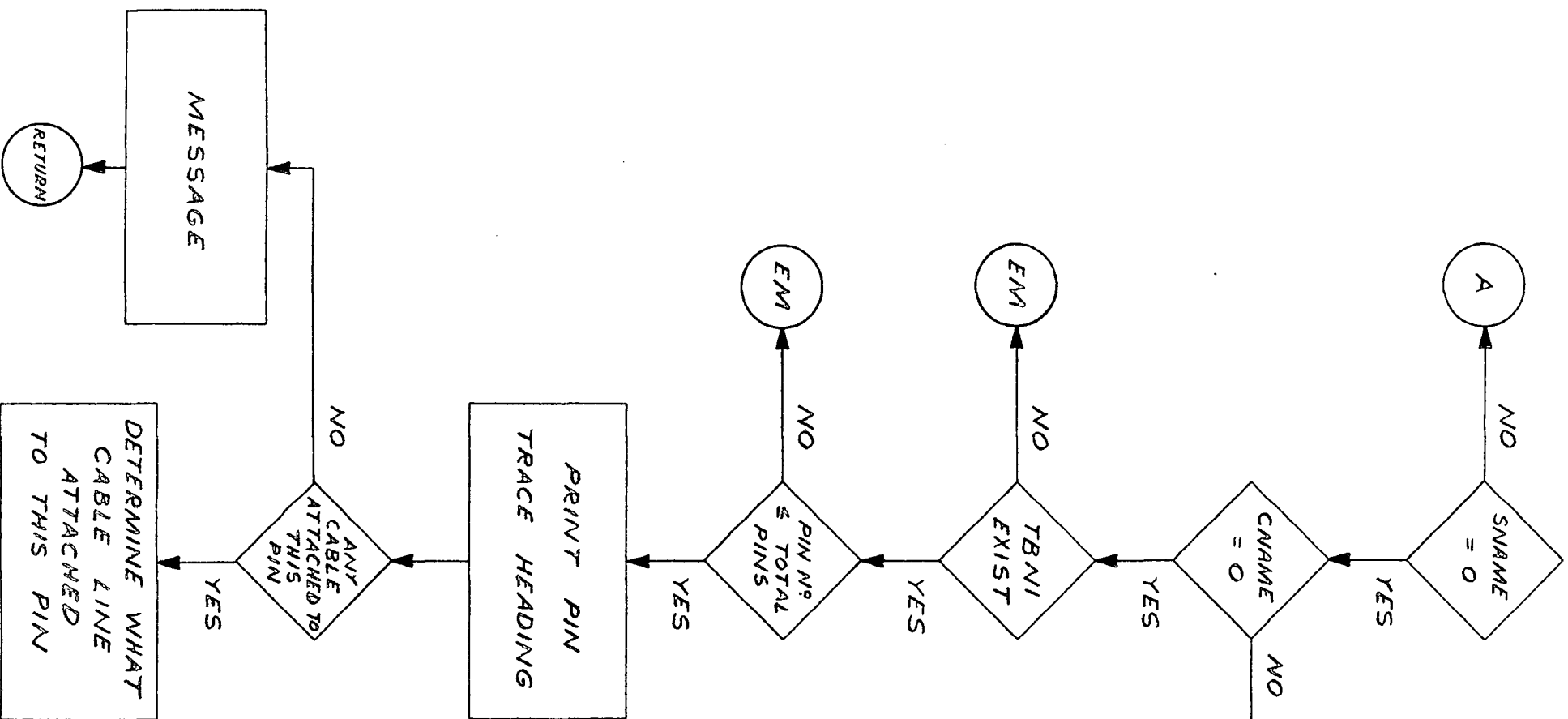
LIR FLOWCHART



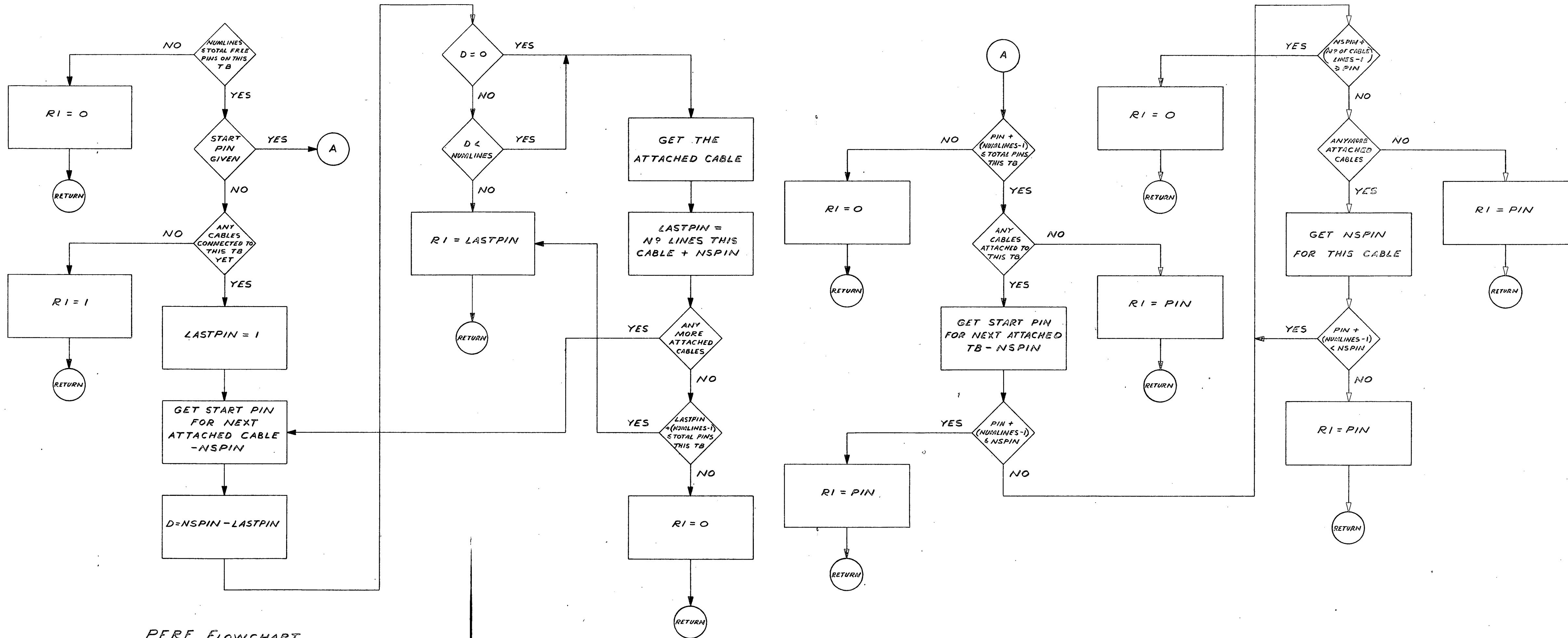
EXR FLOWCHART



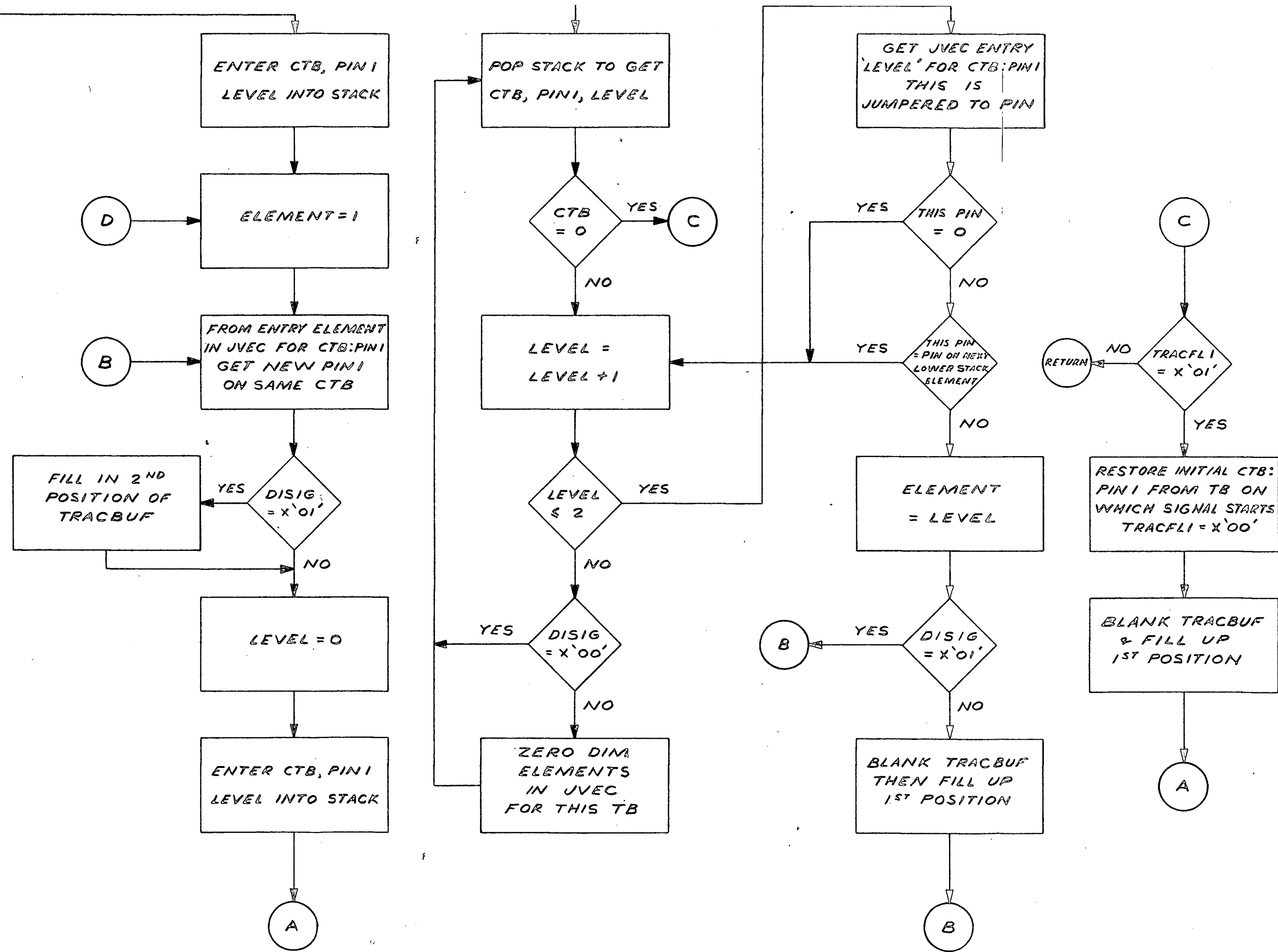
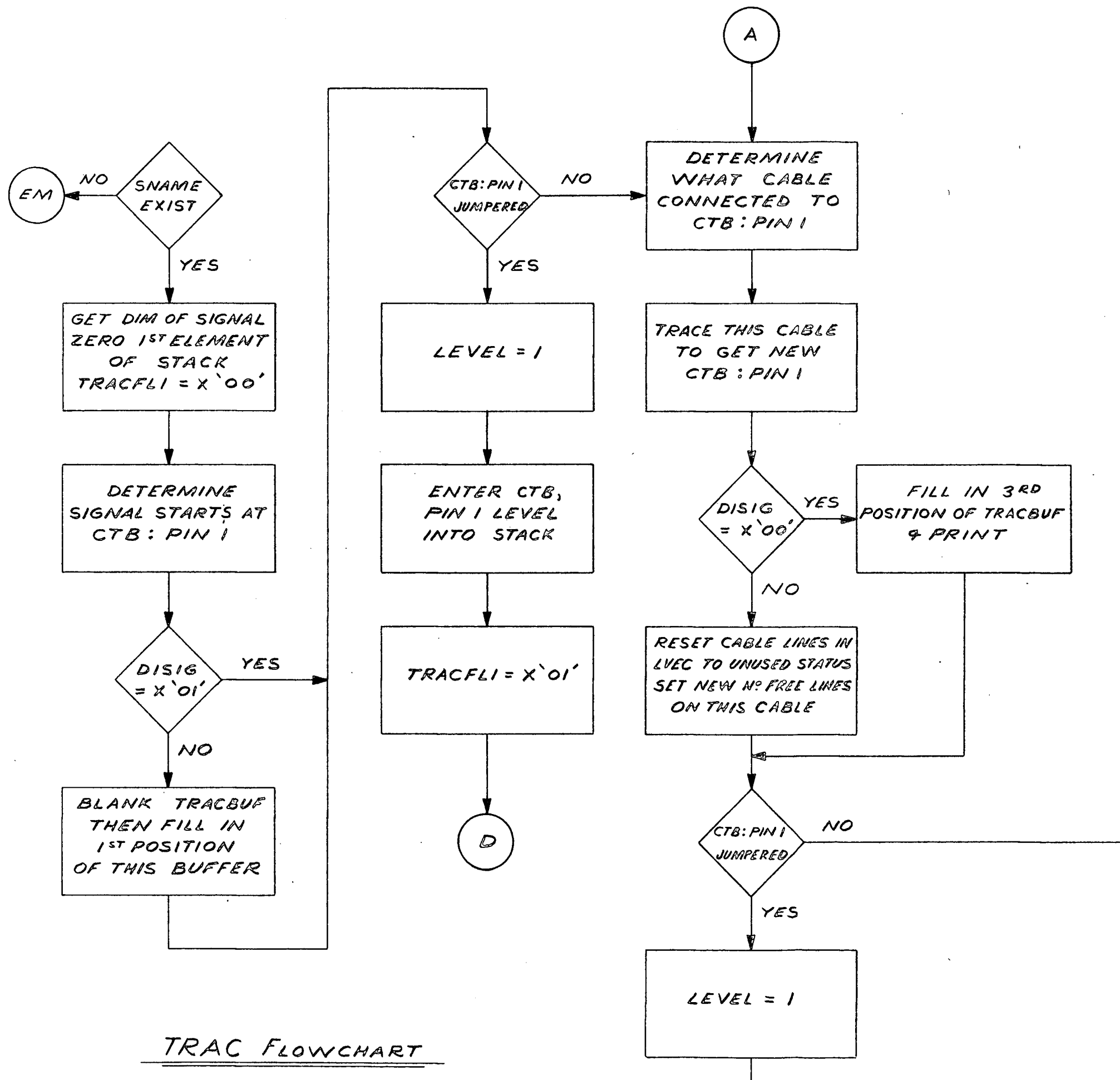


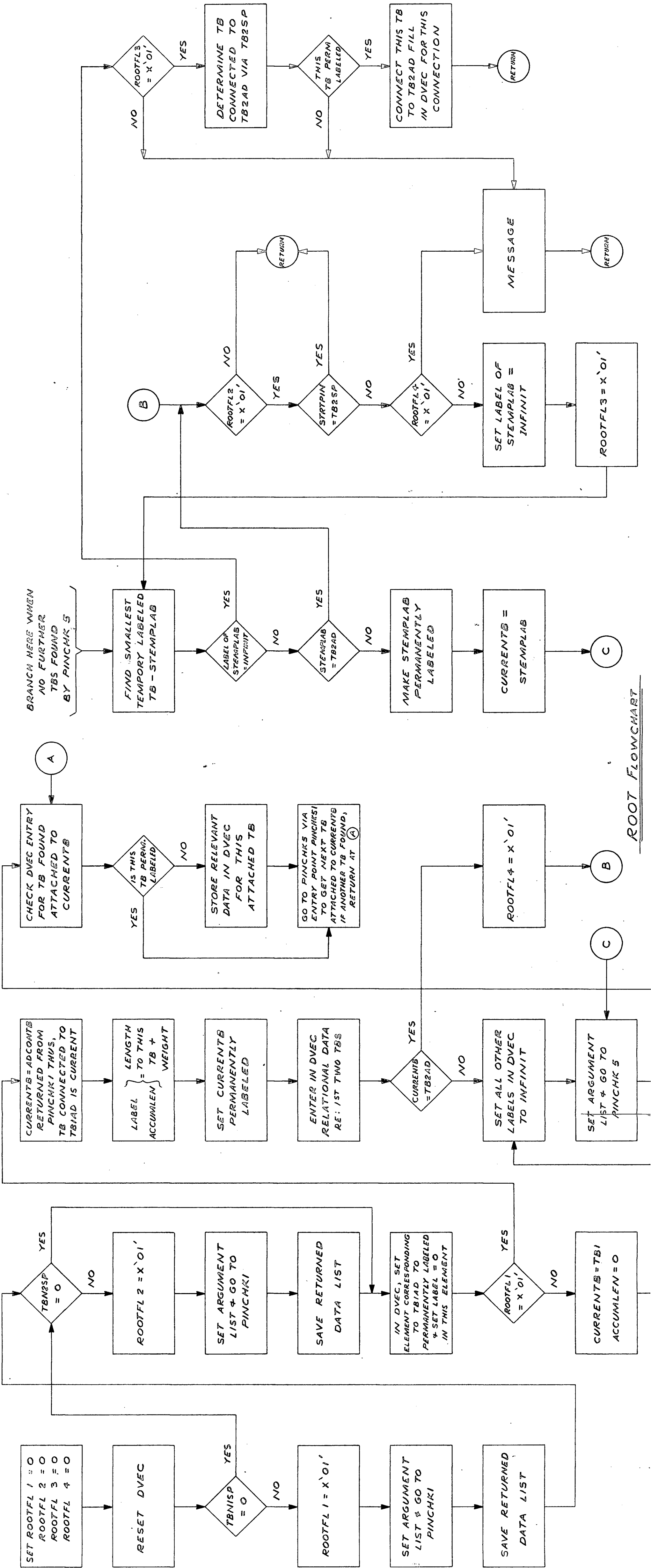


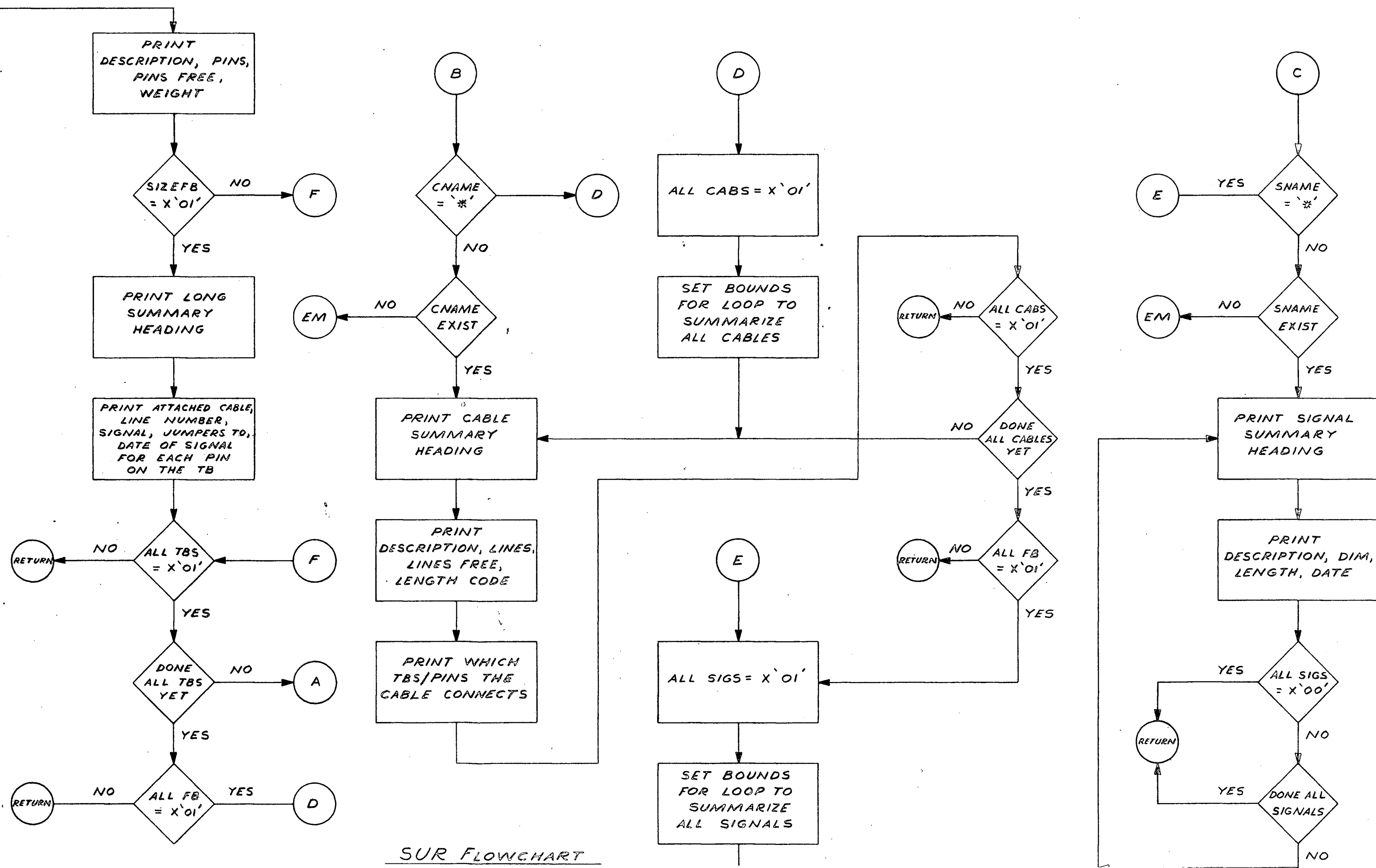
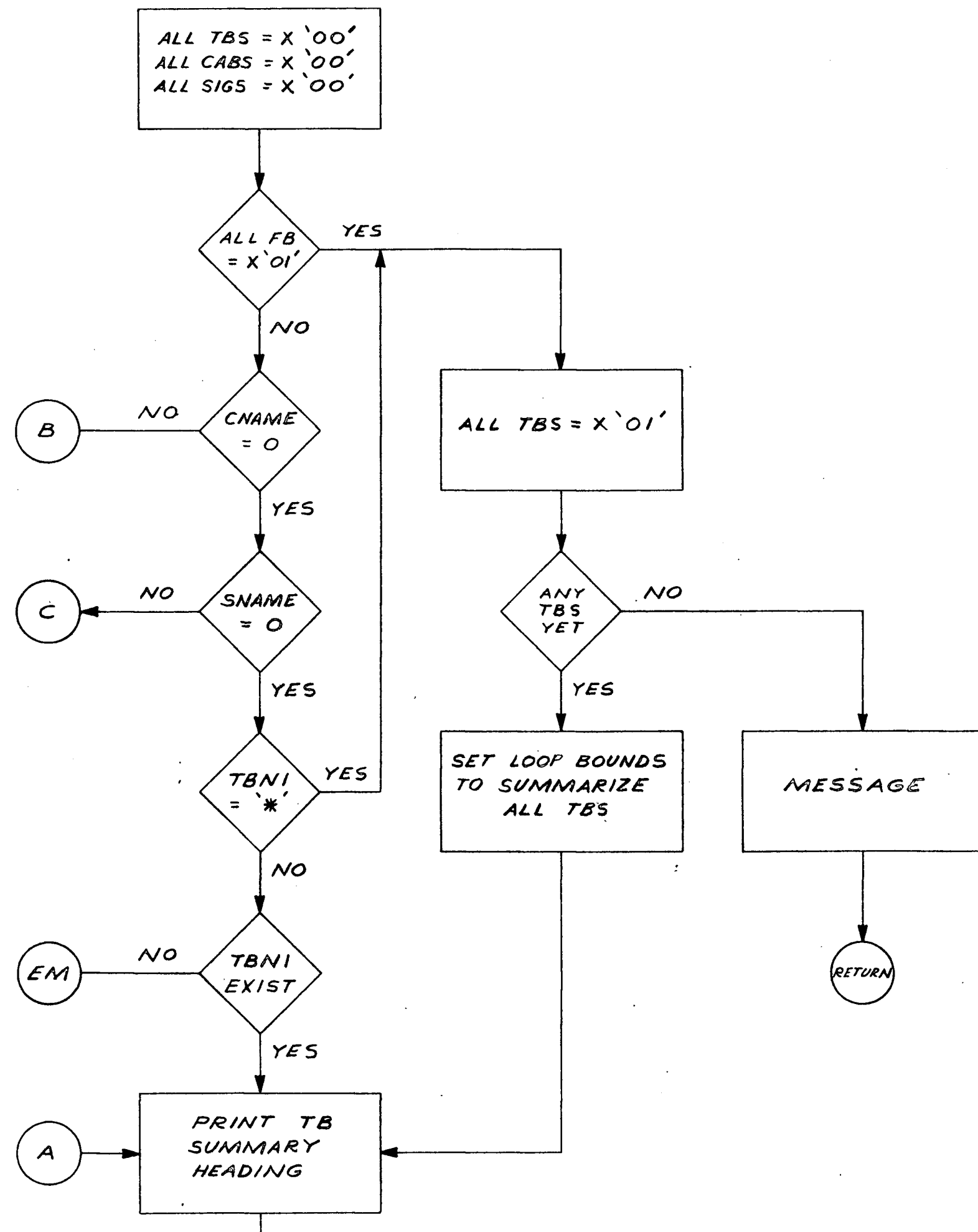
TRR FLOWCHART



PFRE FLOWCHART







APPENDIX B

SYSREC Source Listing

The SYSREC program generates approximately two hundred pages of assembler listing and thus, in the interest of practicality, is not included in this Logic Manual. Copies of this listing may be acquired from

Department of Computer Science
University of British Columbia
Vancouver 8, B. C.