TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
FAA-RD-75-163.I		

4. Title and Subtitle	5. Report Date
SOFTWARE FOR AN EXPERIMENTAL AIR-GROUND DATA LINK Volume I: Functional Description and Flow Charts	October 1975
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
C.J. Goodrow and E. Rachlis	DOT-TSC-FAA-75-21.I
9. Performing Organization Name and Address	10. Work Unit No.
Input Output Computer Services, Incorporated* 689 Concord Avenue Cambridge MA 02138	FA536/R6130
	11. Contract or Grant No.
	DOT-TSC-887-1
12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
U.S. Department of Transportation Federal Aviation Administration Systems Research and Development Service Washington DC 20591	Final Report August 1974 - July 1975
	14. Sponsoring Agency Code

15. Supplementary Notes

*Under contract to: U.S. Department of Transportation
Transportation Systems Center
Kendall Square
Cambridge MA 02142

16. Abstract

This report documents the complete software system developed for the Experimental Data Link System which was implemented for flight test during the Air-Ground Data Link Development Program (FAA-TSC Project Number FA-13).

The software development is presented in three volumes as follows:

Volume I: - - Functional Description and Flowcharts
Volume II: - - System Operation Manual
Volume III: - - Program Listings.

The material contained in Volume I describes the design and implementation of the system software. It is intended to be used as a complementary document to Volumes II and III.

17. Key Words	18. Distribution Statement
VHF Air-Ground Data Link ATC Data Link	DOCUMENT IS AVAILABLE TO THE PUBLIC THROUGH THE NATIONAL TECHNICAL INFORMATION SERVICE, SPRINGFIELD, VIRGINIA 22161

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	278	

Form DOT F 1700.7 (8-69)

PREFACE

The effort reported in this contract represents a portion of the Link Characteristics Phase of the Air-Ground Data Link Development Program (FAA-TSC Project Number FA-13).

The work performed included the design, programming, installation, check-out, and maintenance of a dual-minicomputer controlled, air-ground data link, communication system, flight tested at the National Aviation Facilities Experimental Center (N.A.F.E.C.). In conjunction with the airborne station and ground station operational systems, a set of two data reduction and analysis programs were also designed, programmed, implemented and utilized at N.A.F.E.C. The post-experiment analysis programs are intended as an adjunct to system performance evaluation.

Table of Contents

I. SYSTEM OBJECTIVE/FUNCTION

The experimental data-link system provides a test bed facility for enabling a ground station/air station communication link to be utilized in determining the link management requirements for the data-link in an air traffic control environment.

The entire software scheme incorporates an on-line communication system and off-line, post-experiment data-reduction system to facilitate data-link performance evaluation.

The on-line software is designed such that the ground station controls the communication cycle and the air station responds in a slave-like manner at a data rate of either 2400 or 4800 bits per second.

The ground station operates as a single-channel, simplex (half-duplex) communication controller. Based upon parametric inputs received during the dialogue initialization phase, the ground station transmits a properly formatted message, and awaits a response. The response (or lack of response) is evaluated with respect to technical acknowledgement (ACK/NAK), block check sequence (BCS) error(s), message format validity, and identifier validity. The transmitted messages are stored on a cassette tape and read into a core memory pad for on-line access. The message selection sequence is a function of the initial dialogue responses.

In order to evaluate data-link performance characteristics, the on-going performance occurrances are maintained in memory in a statistics buffer, and a cassette record is maintained

for each transaction (message round-trip). The ground and air stations record each transaction (and its peculiarities) as they occur. The statistics, which are maintained in memory at both stations, are scrolled onto the cassette when a test sequence is completed. These statistics reflect such system performance criteria as the number of message transactions, number of non-responses, number of BCS errors, number of non-acknowledgments, number of bits sent (for echo-type messages), number of bits in error (for echo-type messages), number of characters sent, number of acknowledgments. These criteria are classified as to echo and non-echo and further classified as to message text length.

II. SYSTEM OVERVIEW/DESCRIPTION

The ground station data-link software system operates on three levels. Functionally, there is an initialization level which operates in a communications-inactive, dialogue (computer requests/operator responds) environment. There is a communications environment in which an ongoing communications scheme is maintained. There is also a third (transparent) level, input-output processing and interrupt recognition/handling, which is maintained transparently to the current function (initialization or communication). The air station data-link software system hasn't any dialogue-type initialization; it does, however, have an ongoing communications environment and the input-output and interrupt recognition/handling processing.

The ground station configuration consists of the following equipment:

a) A Texas Instruments TI-960A minicomputer with 16,384 sixteen bit words of memory and interval-timer (real-time clock).

b) Two Texas Instruments Silent-700 ASR-733 electronic data terminals consisting of a keyboard, printer, and twin cassette units, and computer interface.

c) A McDonnell-Douglas, MDL-510, 2400/4800 bit-per-second minimum shift keying modulator-demodulator (modem), and computer interface.

d) V.H.F. receive and transmit antennae (type CA-1781, swastika).

e) A V.H.F. transmitter (ITT, GRI-21).

f) A V.H.F. receiver (ITT, GRR-23)

(Note: The receiver, transmitter, and antenna are not connected with the minicomputer, except as incidental through the modem).

The air station configuration consists of the following equipment:

a) Same as ground station.

b) Same as ground station.

c) Same as ground station.

d) An aircraft Blade antenna (type 37R-2U, VHF/UHF).

e) A three-channel analog-to-digital converter, with computer interface.

f) A V.H.F. transceiver (in accordance with ARINC Characteristic 566A).

The ground station software is comprised of fifteen separate software modules; all of which are linked together such that the system appears as one load module. The modules are functionally inter- and intra-dependent. The selection of module division is one of software design and programming consideration and not necessarily functional consideration. Full advantage was taken of the Texas Instruments assembler, linker, and loader in the construction of the data-link system.

In broad generalities, the following statements may be made about the fifteen modules of the ground station:

Module	Function
DATALP-DATALF-DATALD	Dialog initialization
GSUPR	Communication supervisor

Module	Function
GSUPRA	Communications supervisor
GU	Comm. supervisor utilities
DLFS-DLDS	Comm. supervisor data-base
SIH	Input-output/Interrupt handling
SIHA	"
SIHD-DSB	"
SIHK	"
SIHO	"
INS	"
SDRP	"
SDRP	"
SDRF-SDRD	"
SIHI	"

The air station software is comprised of fourteen separate software modules. The following table is designed to be similar to the table shown for the ground station software.

Module	Function
ASUPR	See GSUPR
ASUPRA	See GSUPRA
AU	" GU
ABF-ABD	" DLFS-DLDS
SIH	See ground station table
SIHA	"
SIHD-DSB	"
SIHK	"
SIHB	"

Module	Function
SIHO	See ground station table
INS	"
SDRP	"
SDRF-SDRD	"
SIHI	"

The data reduction and analysis program is comprised of fourteen separate software modules. In broad generalities the modules have the following functions:

Module	Function
MAIN	
UTIL-MTR-GREC-AREC	
PRINRE	
FIN	
SIH	See ground station table
SIHA	"
SIHD-DSB	"
SIHK	"
SIHB	"
SIHO	"
INS	"
SDRP	"
SDRF-SDRF	"
SIHI	"

It is not the purpose of this document to describe the engineering design and operation of the equipment utilized, nor to describe the utility software utilized for data-link system preparation. It will be stated that the editor used to create and alter source files is TSE-960. The assembler used is either SAL-960 or SALM. The linker used is LRL-960 and the loader used is Texas Instruments bootstrap, relocating loader. The editor, linker, and assemblers are supplied by Texas Instruments and run in the environment of T.I.'s programming support monitor (PSM) or process automation monitor (PAM). It should be noted that the object which results from assembling under SAL-960 is <u>indistinguishable</u> from the object which results from assembling under SALM.

III. DIALOGUE INITIALIZATION

This section describes the function and operation of the ground station operator/computer dialogue and ground station initialization.

After the ground station software is loaded, the system automatically begins a sequence of questions. As each question is asked, the current value is given. The operator must respond to the question.

Some keys are for special functions.

a)	CARRIAGE RETURN (CR)	-	Depress 'CR' key to terminate response to a question.
b)	RUBOUT	-	Use the 'RUBOUT' key to delete one character to the left each time RUBOUT is depressed.
c)	CONTROL/G	-	Force the program to go back to the beginning of the dialogue.
d)	CONTROL/U	-	This allows the user to force the same question to be asked again.
e)	CONTROL/E	-	This key is to terminate entry of the free form dialogue.

The program does the following:

1. Set base registers (Reg. 4, 5, 6, 7).
2. Call 'INITIO' routine to initiate I/O interrupt.
3. Print the self-announcement message.
4. 1) Print the 'HOUR: MINUTE: SECOND = ', followed by the current time-of-day.

   2) Set flag 'FTIME' to show time given.
5. Wait for action.

   1) To terminate this question (i.e. no change), simply depress 'CR' key.

   2) To change the value as seen, enter the desired value with 'CR'.

6. Check all input characters.

   1) If it is a 'CR', go 'CRLOOP' routine.

   2) If it is a 'RUBOUT', go 'RUBLOO' routine.

   3) a. If it is not a special character, check for 'RUBOUT' flag (FRUB) first. If it is set, or no more character for 'RUBOUT', print the back slash or ' \ ' then show the character. If it is not set, or no 'RUBOUT' at all, show the character directly, then go to step (b).

      b. Store the character, clear the 'RUBOUT' flag, set 'INPUT' flag (FIN) then keep accepting characters until a 'CR' or special key is depressed.

CARRIAGE RETURN (CRLOOP)

This routine checks to see what step should be executed next and branches to appropriate place.

1. Check the 'RUBOUT' flag for 'RUBOUT' processing. If it is set, print the back slash '\' and reset the 'RUBOUT' flag. Then go to 2.
   If it is not set, go to 2.

2. Check the 'FREE FORM' flag (FPS);
   If this CR is in free form dialogue, store CR-LF for recording, then reset the counter to continue to accept the input.
   If it is not set, go to 3.

3. Check 'INPUT' flag (FIN) for any changes. If it is not set, go to 'STEP'. If it is set, something is to be changed; go to 4.

4. Check 'TIME' flag (FTIME) to see if the changes are for the time-of-day.
   If it is not set, go to 5.
   If it is set, time is being changed (TIMECH).
   Routine 'TIMECH' does the following:
   1. Sets the position counter properly.
   2. Checks the input string for:
      a) 8 characters exactly.
         If not, goes to 'ERRMEG' routine.
      b) Number represented (CHKNUM).
   3. Checks to see 'HOUR' in range (00-23), 'MINUTE' in range (00-59), 'SECOND' in range (00-59).
   4. Checks to see that two colons are in the right places.
   5. Stores new time in ASCII with space (HHH).
   6. Resets 'INPUT' flag.
   7. Goes to 'STEP' routine.

5. Check 'BIT RATE' flag to see if the changes are in the transmission rate.
   If it is not set, go to 6.
   If it is set, go 'BAUDCH' routine.
   This routine does the following:
   1) Sets the position counter right.
   2) Checks input characters:
      a) 4 characters exactly, or to to 'ERRMEG' routine.
      b) Number represented (CHKNUM).
   3) Changes to binary (MASK12)
   4) Gets number of thousands, hundreds, tens, ones (KITHTO).
   5) Converts to hexadecimal expression (TOHEXA).
   6) If the number is 4800_{10}, set flag BAUD12 = 1.
      If the number is 2400_{10}, set flag BAUD12 = 0.
      Otherwise, goes to 'ERRMEG' routine.
   7) Stores new bit rate in ASCII (for recording).
      Clears 'INPUT' flag (III).

6. Check the appropriate flag to see if the changes are to 'KNR' (the number of non-responses).
   If it is not set, go to 7.
   If it is set, go to 'NREPCH' routine.
   This routine does the following:
   1) a) Arranges input characters.
      b) Converts to hexadecimal expression (TOHEXA).
      c) Checks acceptable limits (1 To 10_{10}).
   2) Stores new number of non-responses, resets 'INPUT' flag (III).
   3) Goes to 'STEP'.

7. Check the appropriate flag to see if the changes are to 'KNAKS' (the number of NAKs allowed).
   If it is not set, go to 8.
   If it is set, go to 'NAKCH' routine.
   This routine does the following:
   1) a) Arranges input characters.
      b) Converts to hexadecimal expression (TOHEXA).
      c) Checks limits ($1-10_{10}$).
   2. a) Resets 'INPUT' flag.
      b) Stores new number of NAKs.
   3. Goes to 'STEP'.

8. Check the appropriate flag to see if the changes are to 'KBCSF' (the number of BCS errors allowed).
   If it is not set, go to 9.
   If it is set, go to 'BCSCH' routine.
   This routine does the following:
   1) a) Arranges input characters.
      b) Converts to hexadecimal expression (TOHEXA).
      c) Checks limits ($1-10_{10}$).
   2) a) Resets 'INPUT' flag.
      b) Stores new number of BCS errors.
   3) Goes to 'STEP'.

9) Check the appropriate flag to see if the changes are to 'KNPK' (the number of allowable prekeys).
   If it is not set, go to 10.
   If it is set, go to 'NPKCH' routine.
   This routine does the following:
   1) Adjusts the position counter properly.
   2) Checks input characters.
      a) 3 input characters.
         If more, goes to 'ERRMEG' routine.
      b) Number represented (CHKNUM).
   3) Changes to binary (MASK12).
   4) Finds number of thousands, hundreds, tens, ones from input characters (KITHTO).
   5) Converts to hexadecimal expression (TOHEXA).
   6) Checks limits (1 to 150_{10}).
   7) Stores new number of prekeys (III).
      Resets 'INPUT' flag.
   8) Goes to 'STEP'.

10. Check the appropriate flag to see if the changes to the number of message repeats (KMSGR).
If it is not set, go to 11.
If it is set, go to 'MSGRCH' routine.
This routine does the following:
1) Sets the position counter properly.
2) Checks input characters.
   a) 5 input characters.
      If more, goes to 'ERRMEG'.
   b) Number represented (CHKNUM).
   c) Change to binary.
   d) If there are 5 input characters exactly, check the 1st character.
      If it is 1, check the other 4 characters to see whether they are all zero.
      If they are all zeroes, new number is 2710_{16}, save it and clear input flag, then go to next step.
      If one of them is not zero, go to 'ERRMEG'.
      If the first character is 0, ignore this character and treat as 4 input characters.
3) Finds number of thousands, hundreds, tens, ones from input characters (KITHTO).
4) Converts to hexadecimal expression (TOHEXA).
5) Checks limits (1 to 10000_{10}).
6) Stores new number of message repeats (III).
7) Goes to 'STEP'.

11. Check the appropriate flag to see the changes are to the scenario messages desired.
If it is not set, go back to the beginning.
If it is set, go to routine 'SCENCH',
This routine does the following:
1) Sets the position counter properly.
2) Checks input characters:
   a) 2 characters
      If more, 'ERRMEG' is entered.
   b) Number represented (CHKNUM).
   c) Changes to binary (MASK12).
3) Converts to hexadecimal expression (TOHEXA).
4) Checks limits (1 To 16_{10}).
5) Checks first scenario flag.
   If it is set, stores this new value for the first scenario, temporarily.
   Resets first scenario flag and goes to 'STEP'.
   If it is not set, this new value is for number of the last scenario.
   Compares it with number of first scenario.
   If it is less, number goes to 'ERRMEG'.
   If not, stores new value of first and last scenario.
   Clears the last scenario flag, then goes to 'STEP'.

'CHANGE' SUBROUTINE

This subroutine is used to arrange the input characters for changing the number of non-responses, the number of NAKs, or the number BCS errors.

It does as following:

1) Sets the position counter properly.

2) Checks input characters

   a. 2 input characters, exactly.
      If more, ERROR

   b. Number represented (CHKNUM).

3) Finds the thousands, hundreds, tens, ones from these modified input characters (KITHTO).

4) Converts to hexadecimal expression (TOHEXA).

5) Checks limits.

6) Returns to main program.

'MSGPRT' SUBROUTINE

This subroutine is used to print the message, character by character, using 'PRINTC' subroutine.

It does as following:

1) Gets a word (two character).

   If the word is a -1, the end of message is implied, therefore, to return to main program.

   a. Right justify the first character.

   b. Mask to get the left character.

   c. Print the left character (PRINTC).

2) Gets the word again.

   a. Masks out the left character to get the right character.

   b. Prints the right character (PRINTC).

3) Increase pointer by one.

4) Go to 1.

SUBROUTINE 'CHKNUM'

This subroutine is used to check all input characters and to see if all acceptable numbers are in range.

It does the following:

1) Gets the character.
2) Tests to see of it is in range:
   a. X'30' to X'39' (0-9)
   b. X'41' to X'46' (A-F)

   If it is in range, save the data, otherwise go to 'ERRMEG'.
3) Goes to 1 until all characters are checked.
4) Returns to main program after all characters are tested.

'BUFFST' SUBROUTINE

This subroutine is used to save the fixed data for recording purposes, The format is five locations per parameter, and a space to separate the data (except first 8 locations for storing time-of-day).

It does the following:

1) Sets up the pointer.
2) Stores ASCII '30' first (zeroes).
3) Loads and stores thousands.
4) Loads and stores hundreds.
5) Loads and stores tens.
6) Loads and stores ones.
7) Loads and stores space.
8) Initializes values for next step.
9) Returns to main program.

- 'PRINTQ' SUBROUTINE

This subroutine is used to print a character using the 'PRINTC' routine, and then momentarily delaying.

It does the following:

1) Sets return address.
2) Sets delay factor.
3) Prints the character by using 'PRINTC'.
4) Returns to main program.

'MASK12' SUBROUTINE

This subroutine is used to change ASCII character to binary; it is used right after the 'CHKNUM' subroutine.

It does the following:

1) Gets the character.
2) Compares it with X'40' (one more than 9).
   If it is greater than X'40', it must be from the set A-F.
   Subtracts X'37' from that ASCII character. Otherwise,
   subtracts X'30' from it.
3) Saves the binary character.
4) Returns to main program after looping.

'ELOOP' ROUTINE

This routine is used to terminate the free form data, when the CONTROL/E is accepted. This special character is for free form data only.

1) Checks flag 'FPS' when the control/E is recognized.
   If it is not set, ERROR.
   If it is set, shows CR-LF and saves the position counter.
2) Resets 'FPS' flag.
3) Goes to 'STEP'.

'ULOOP' ROUTINE

This routine is used to erase the question and to re-ask the same question (for the convenience of the operator).

It handles the following:

1) Set the position counter properly.

2) If the table pointer is -1, re-start by showing time-of-day again.

   Otherwise, decrease the table pointer by one.

3) If'FPS' is set, adjust the position counter first.

   Otherwise go to 4.

4) Initializes some values for calculation.

5) Resets two flags:

   a. 'RUBOUT' flag (FRUB)

   b. 'INPUT' flag (FIN);,

6) Goes to 'STEP'.

'GLOOP' ROUTINE

This routine is used to handle the special character 'CONTROL/G'.

It does the following:

1) Initializes some values for calculation.
2) Restarts the dialogue from the beginning (time-of-day).

'STEP' ROUTINE

This routine is used to set up the proper table pointer and branch to the appropriate place.

It does the following:

1) Increases the table pointer by one.
2) Picks up the branch address via table look up and transfers control.

'ERRMEG' ROUTINE

This routine is used to show a question mark '?' when an ERROR is made.

It does the following:

1) Prints the question mark (?).
2) Re-asks the same question.

'MSGNAK' ROUTINE

This routine is used to show the number of NAKS.

It handles the following:

1) Prints the question.

2) Shows the value of KNAKS (the previous response).

3) Saves the value of KNAKS for recording.

4) Clears flag 'FNR'.

5) Sets flag 'FNAK' to indicate program position.

6) Waits for action (key-in).

'RUBLOO' ROUTINE

This routine is used to handle the character which has been erased, when the 'RUBOUT' key is depressed.

It does the following:

1) Checks the 'RUBOUT' flag.
   If it is not set, it means this is the first 'RUBOUT'.
   A back-slash is issued and flag 'FRUB' is set.
2) Checks the counter to see if any input characters remain:
   If none, clear the 'RUBOUT' flag, and go to 'ERRMEG' routine.
   If there is, decrement the position counter by one.
3) Shows the deleted character.
4) Waits for action (key-in).

'MSGNRE' ROUTINE

This routine is used to show the number of non-responses.

It handles the following:

1) Prints the question.
2) Shows the current value of KNR (the previous response).
3) Saves the value for recording purposes.
4) Resets 'BAUD' flag.
5) Sets flag 'FNR' to show KNR given.
6) Waits for action (key-in).

'GAME' ROUTINE

This routine is used to make sure all fixed data and free-form data exists.

It does the following:

1) Records the entry data on cassette.
2) Assures all data is written.
3) Switches unit 2 to playback.
4) Rewinds both cassettes to load point.
5) Sets flag 'SCEN' for playback.
6) Reads the data acquisition cassette into memory.
7) Clears flag 'SCEN' .
8) Switches unit 1 to playback.
9) Prints the data which was read.
10) Goes to 'STEP'.

'READS' ROUTINE

This routine is used to handle SCENARIO messages.

There are sixteen messages on the cassette.

This routine does the following:

1) Initializes the message number and reserved locations.

2) Sets flag 'SCEN'.

3) Reads one SCENARIO message from the cassette.

4) Finds the total number of characters in this message to be stored.

5) Finds the total number of locations needed.

6) Packs the data and stores it in 'CSCEN'.

7) Increases the message number.

8) Fills the pointer table.

9) Goes to (2) (Indexed loop of 16 scenario messages).

10) After looping:

    a. Checks BAUD12: if = 1, set Reg. 1 = 1
                      if = 0, set Reg. 1 = 0

    b. Initializes ground station by calling subroutine IGS.

    c. Goes to 'GSUPR'.

'ATOM' SUBROUTINE

This subroutine outputs values with leading zeroes suppression.

'KITHTO' SUBROUTINE

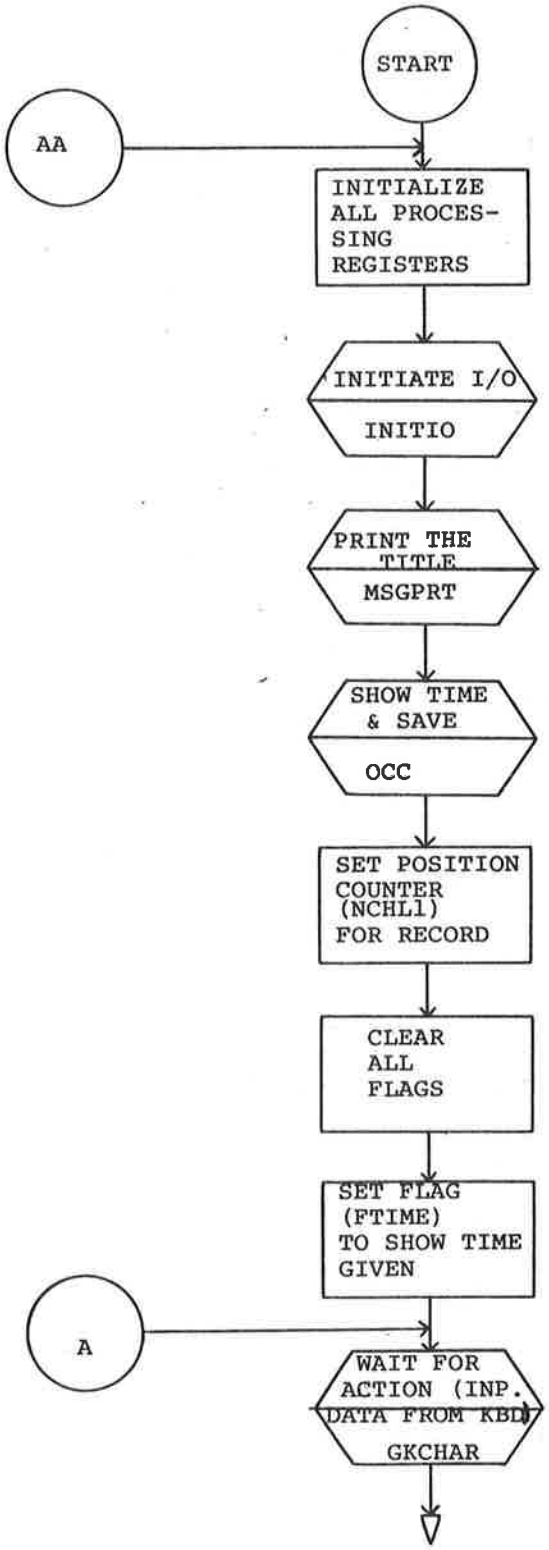
This subroutine justifies a keyed-in value by adjusting for the number of keyboard strokes. This allows the operator to suppress leading zeroes.

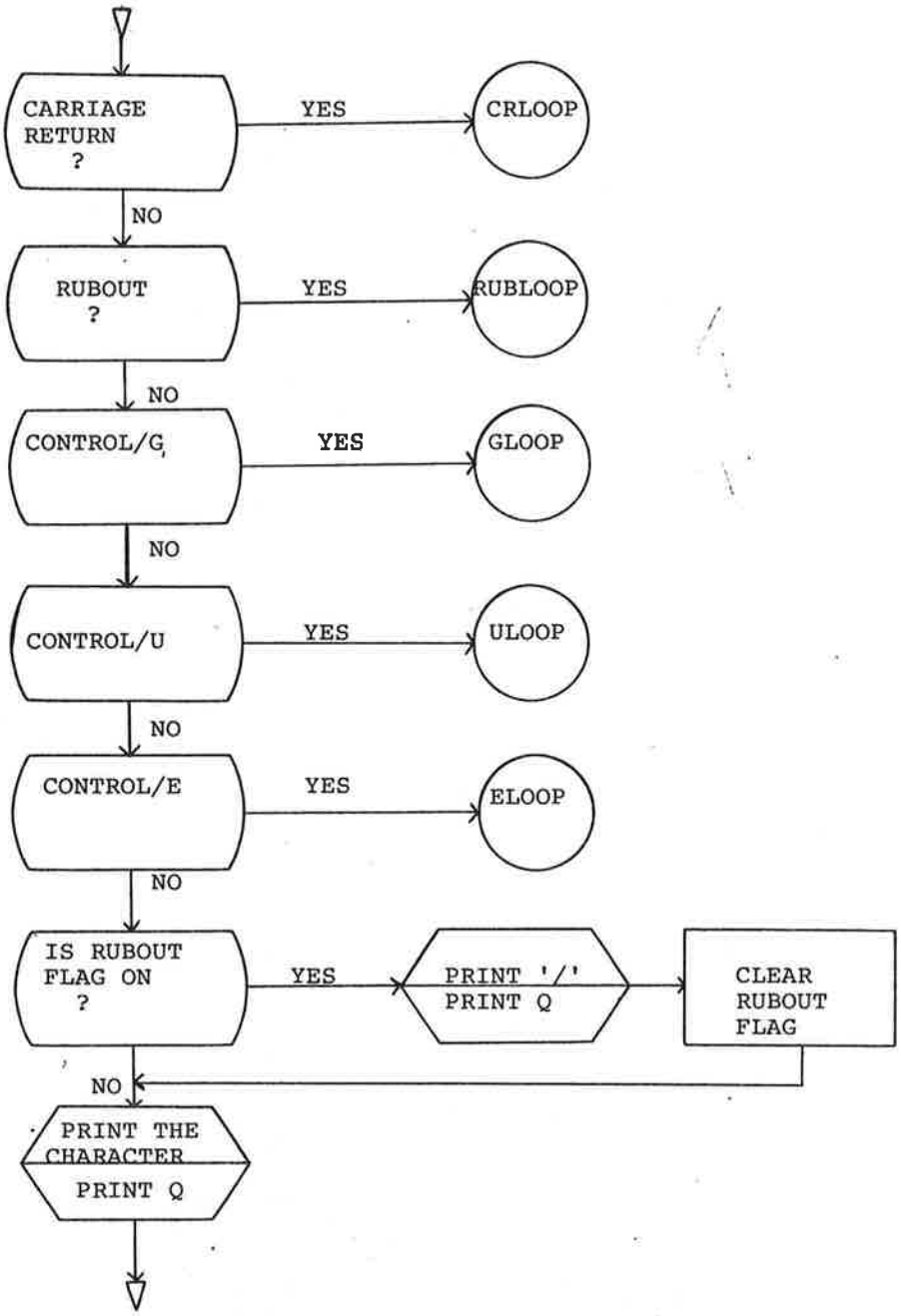
'TOHEXA' SUBROUTINE

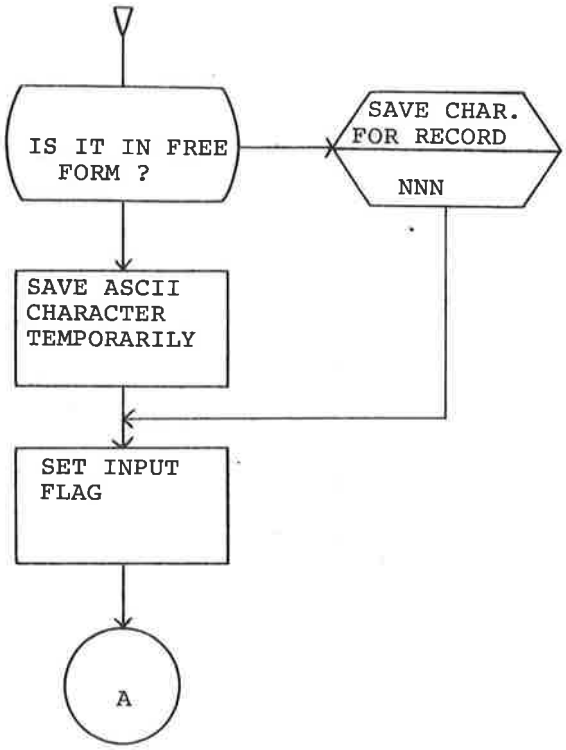
This subroutine is used to covert a given decimal value (represented in ASCII) to hexadecimal (binary). The process is done by repetitive addition of thousands, hundreds, tens, and ones by position.

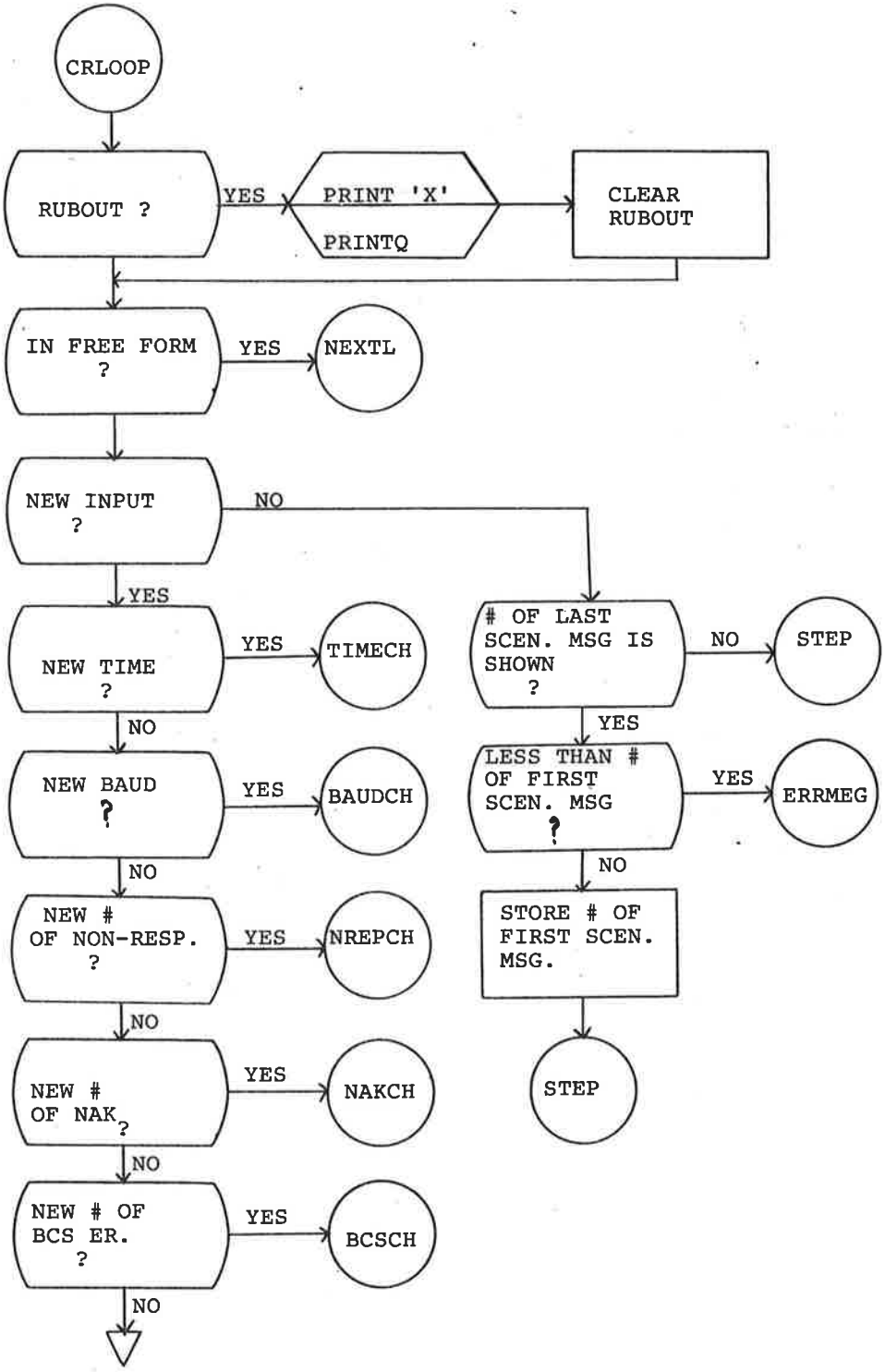
'THTO' SUBROUTINE

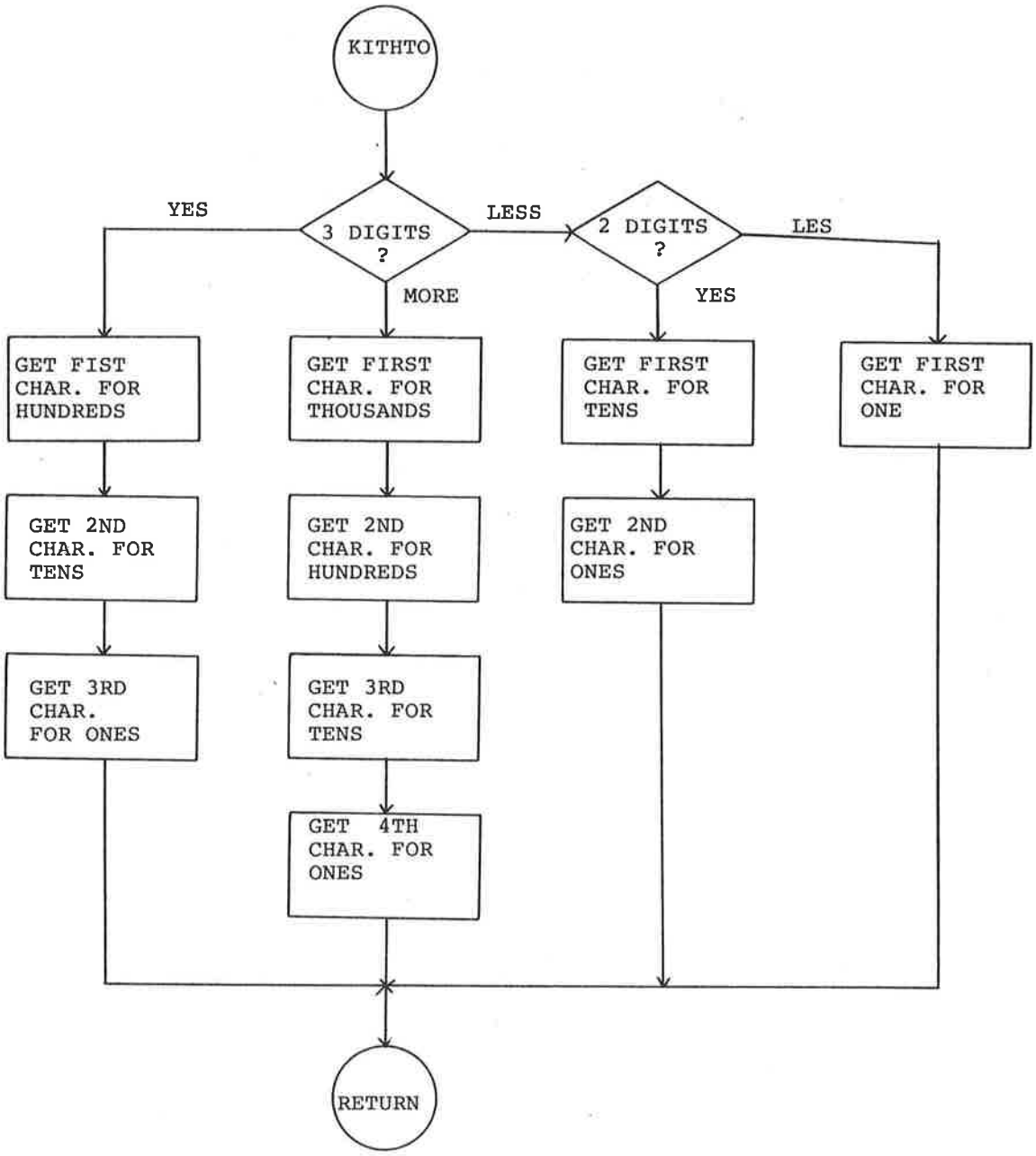
This subroutine converts a binary value to decimal representation by repetitive subtraction of thousands, hundreds, tens, and ones by proper position.

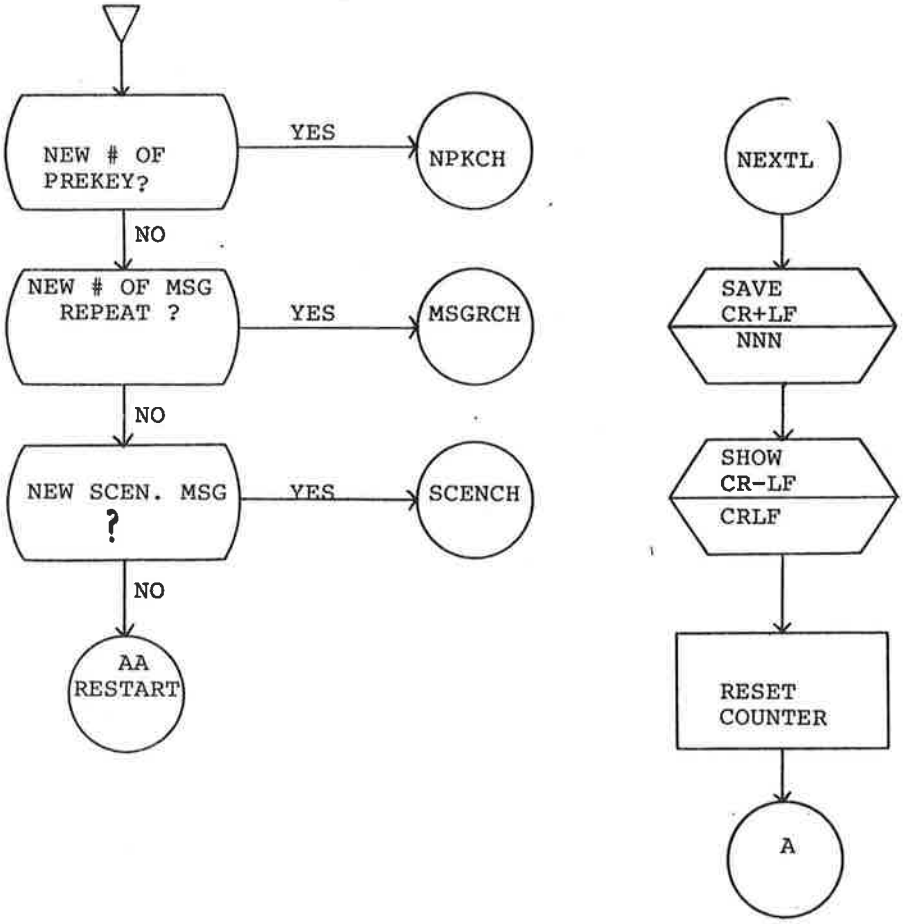

START
AA
INITIALIZE ALL PROCES-SING REGISTERS
INITIATE I/O
INITIO
PRINT THE TITLE
MSGPRT
SHOW TIME & SAVE
OCC
SET POSITION COUNTER (NCHL1) FOR RECORD
CLEAR ALL FLAGS
SET FLAG (FTIME) TO SHOW TIME GIVEN
A
WAIT FOR ACTION (INP. DATA FROM KBD)
GKCHAR

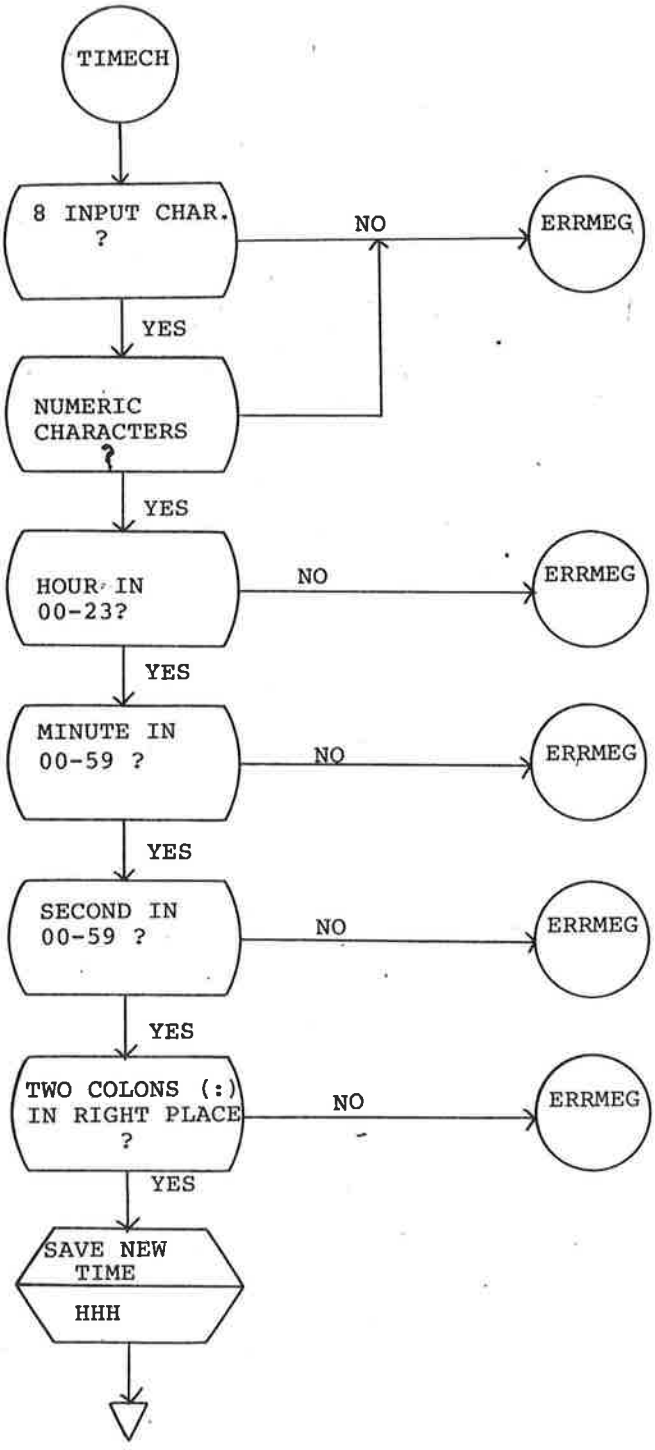

CARRIAGE RETURN ?
YES
CRLOOP
NO
RUBOUT ?
YES
RUBLOOP
NO
CONTROL/G
YES
GLOOP
NO
CONTROL/U
YES
ULOOP
NO
CONTROL/E
YES
ELOOP
NO
IS RUBOUT FLAG ON ?
YES
PRINT '/'
PRINT Q
CLEAR RUBOUT FLAG
NO
PRINT THE CHARACTER
PRINT Q

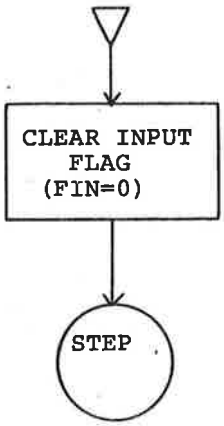

IS IT IN FREE FORM ?
SAVE CHAR. FOR RECORD
NNN
SAVE ASCII CHARACTER TEMPORARILY
SET INPUT FLAG
A

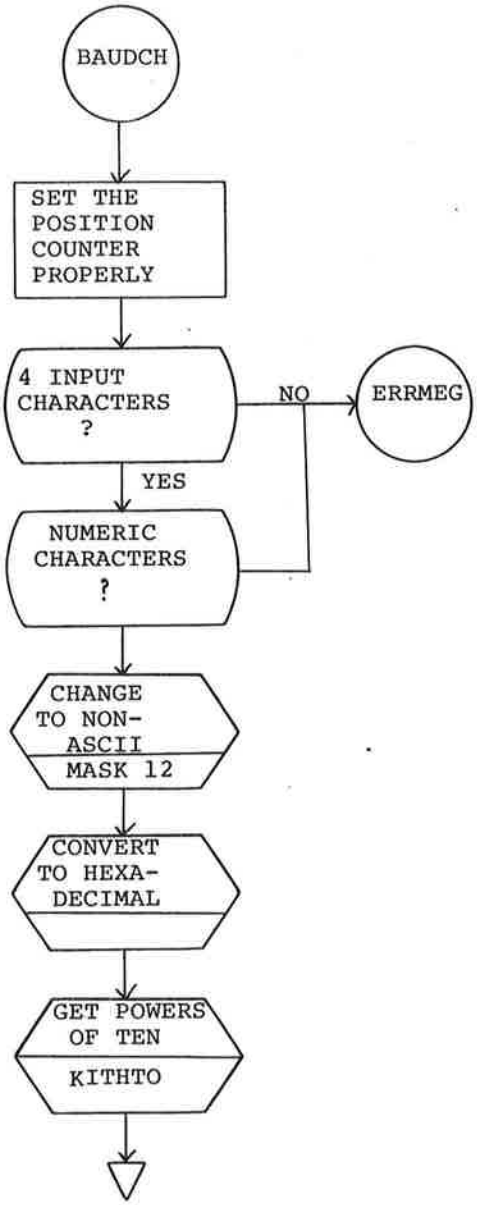

CRLOOP
RUBOUT ?
YES
PRINT 'X'
PRINTQ
CLEAR RUBOUT
IN FREE FORM ?
YES
NEXTL
NEW INPUT ?
NO
YES
NEW TIME ?
YES
TIMECH
NO
NEW BAUD ?
YES
BAUDCH
NO
NEW # OF NON-RESP. ?
YES
NREPCH
NO
NEW # OF NAK ?
YES
NAKCH
NO
NEW # OF BCS ER. ?
YES
BCSCH
NO
OF LAST SCEN. MSG IS SHOWN ?
NO
STEP
YES
LESS THAN # OF FIRST SCEN. MSG ?
YES
ERRMEG
NO
STORE # OF FIRST SCEN. MSG.
STEP

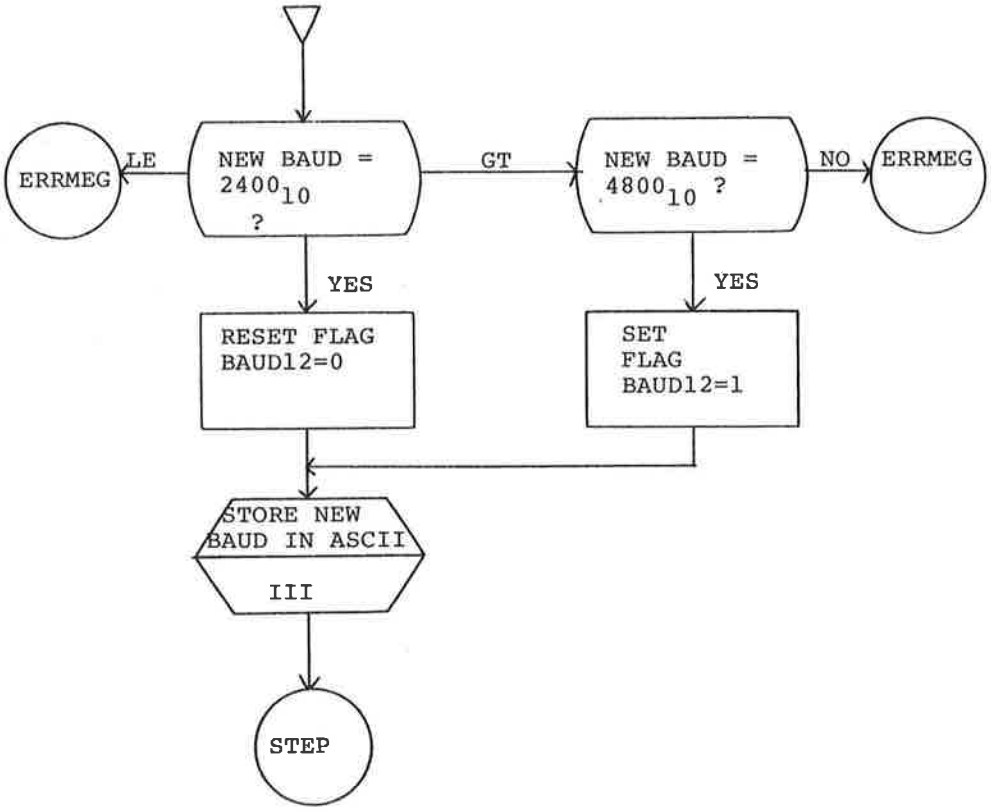

KITHTO
3 DIGITS ?
YES
LESS
MORE
2 DIGITS ?
YES
LES
GET FIST CHAR. FOR HUNDREDS
GET 2ND CHAR. FOR TENS
GET 3RD CHAR. FOR ONES
GET FIRST CHAR. FOR THOUSANDS
GET 2ND CHAR. FOR HUNDREDS
GET 3RD CHAR. FOR TENS
GET 4TH CHAR. FOR ONES
GET FIRST CHAR. FOR TENS
GET 2ND CHAR. FOR ONES
GET FIRST CHAR. FOR ONE
RETURN

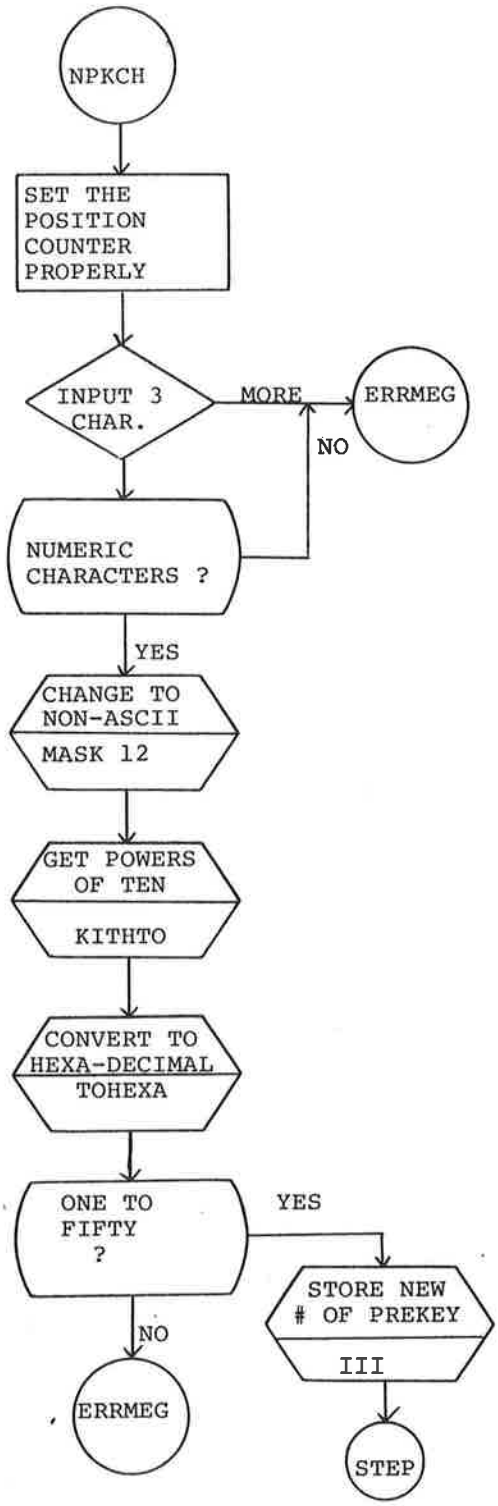

NEW # OF PREKEY?
YES
NPKCH
NO
NEW # OF MSG REPEAT ?
YES
MSGRCH
NO
NEW SCEN. MSG ?
YES
SCENCH
NO
AA RESTART
NEXTL
SAVE CR+LF
NNN
SHOW CR-LF
CRLF
RESET COUNTER
A

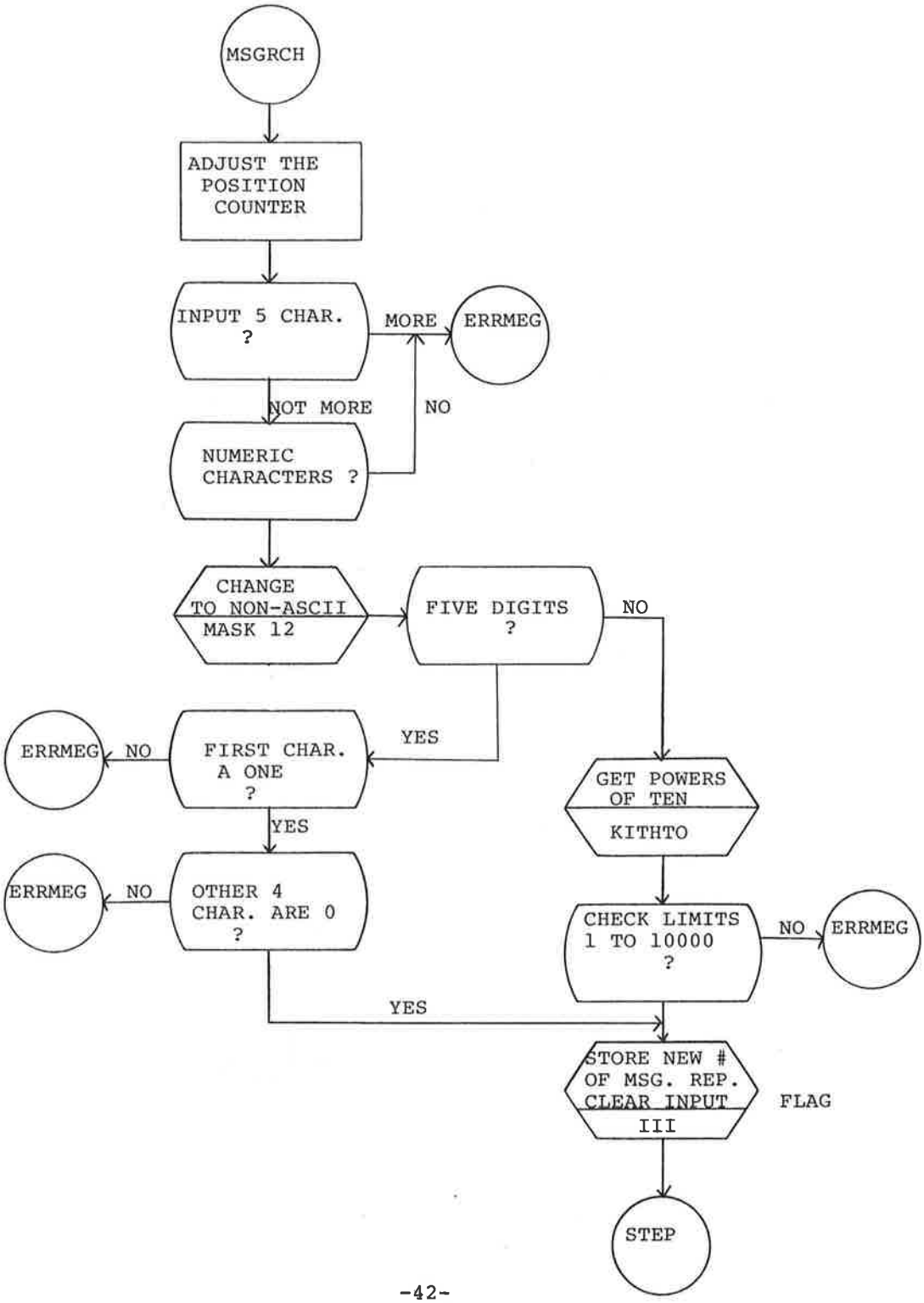

TIMECH
8 INPUT CHAR. ?
NO
ERRMEG
YES
NUMERIC CHARACTERS ?
YES
HOUR IN 00-23?
NO
ERRMEG
YES
MINUTE IN 00-59 ?
NO
ERRMEG
YES
SECOND IN 00-59 ?
NO
ERRMEG
YES
TWO COLONS (:) IN RIGHT PLACE ?
NO
ERRMEG
YES
SAVE NEW TIME
HHH

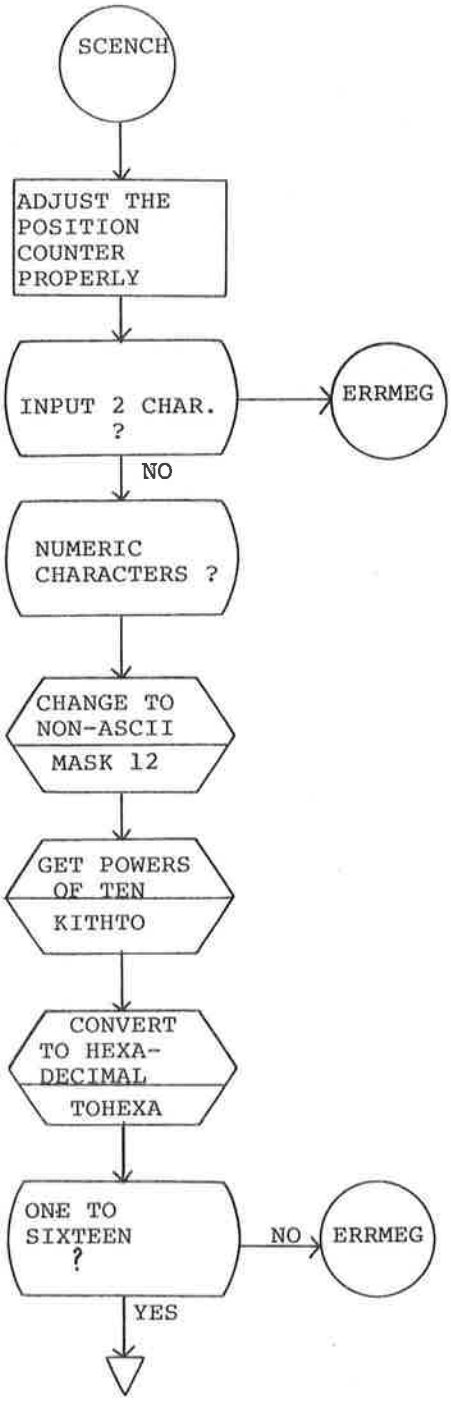

CLEAR INPUT
FLAG
(FIN=0)
STEP

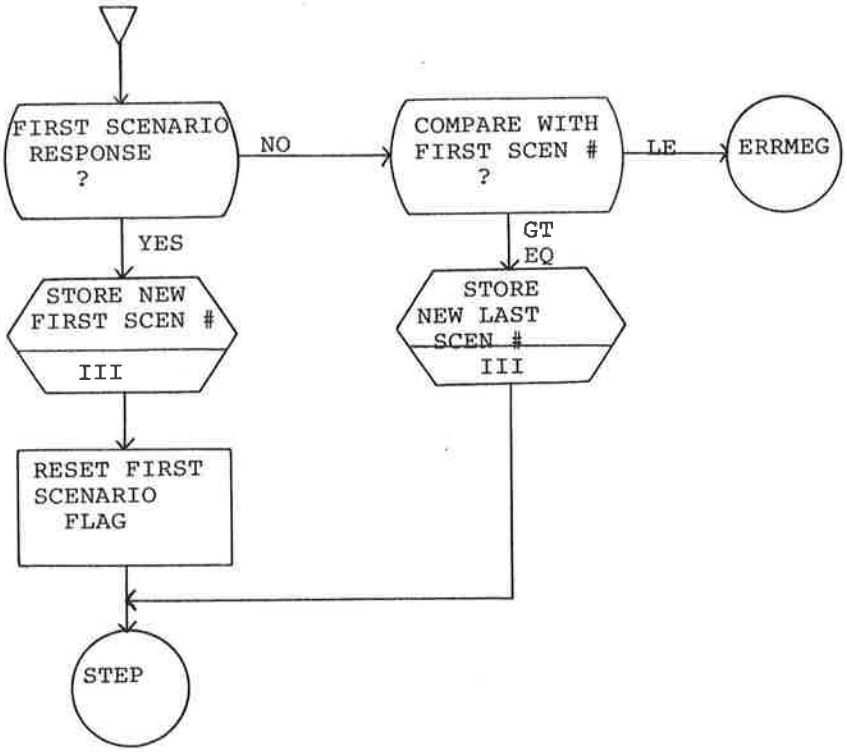

BAUDCH
SET THE POSITION COUNTER PROPERLY
4 INPUT CHARACTERS ?
NO
ERRMEG
YES
NUMERIC CHARACTERS ?
CHANGE TO NON-ASCII
MASK 12
CONVERT TO HEXA-DECIMAL
GET POWERS OF TEN
KITHTO

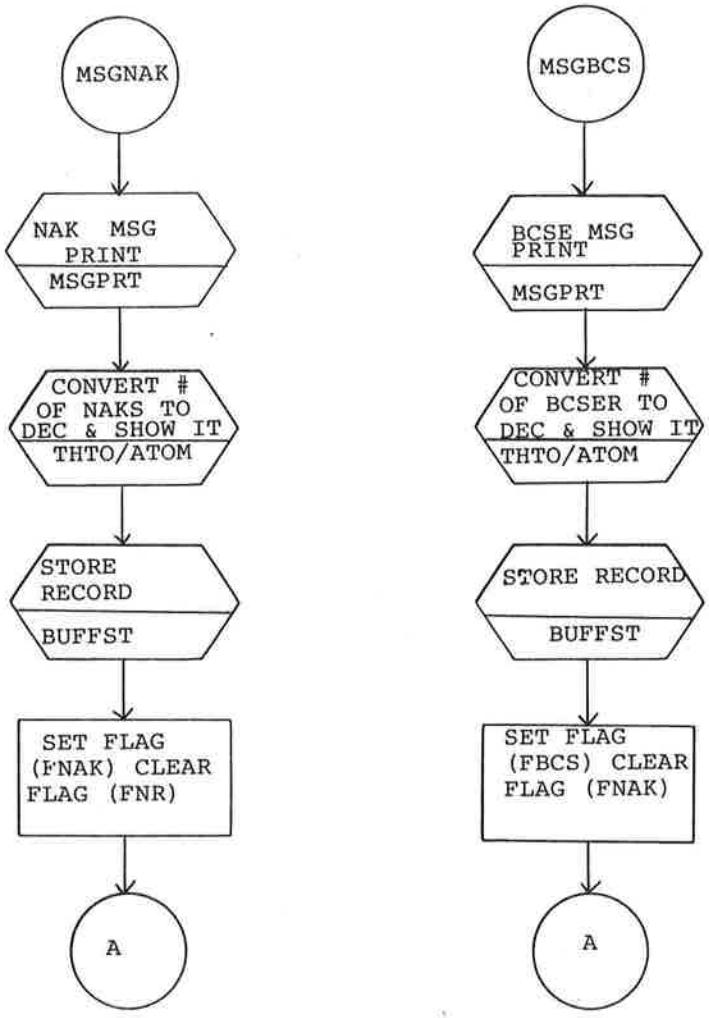

ERRMEG
LE
NEW BAUD =
2400 10
?
GT
NEW BAUD =
4800 10 ?
NO
ERRMEG
YES
YES
RESET FLAG
BAUD12=0
SET
FLAG
BAUD12=1
STORE NEW
BAUD IN ASCII
III
STEP

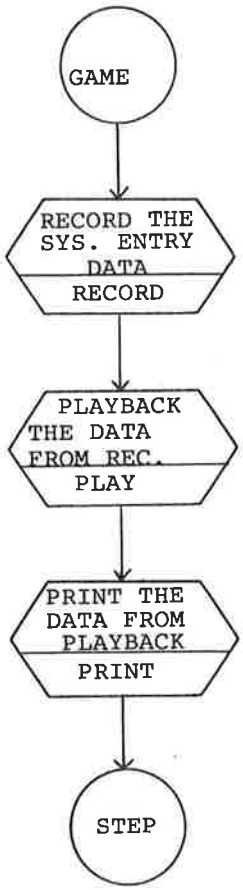

NPKCH
SET THE POSITION COUNTER PROPERLY
INPUT 3 CHAR.
MORE
ERRMEG
NO
NUMERIC CHARACTERS ?
YES
CHANGE TO NON-ASCII
MASK 12
GET POWERS OF TEN
KITHTO
CONVERT TO HEXA-DECIMAL
TOHEXA
ONE TO FIFTY ?
YES
NO
ERRMEG
STORE NEW # OF PREKEY
III
STEP

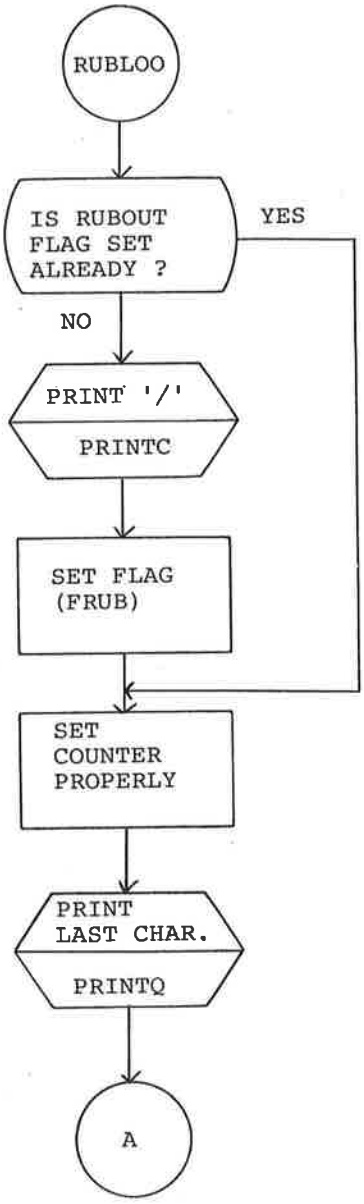

MSGRCH
ADJUST THE POSITION COUNTER
INPUT 5 CHAR. ?
MORE
ERRMEG
NOT MORE
NO
NUMERIC CHARACTERS ?
CHANGE TO NON-ASCII MASK 12
FIVE DIGITS ?
NO
YES
ERRMEG
NO
FIRST CHAR. A ONE ?
YES
GET POWERS OF TEN
KITHTO
ERRMEG
NO
OTHER 4 CHAR. ARE 0 ?
CHECK LIMITS 1 TO 10000 ?
NO
ERRMEG
YES
STORE NEW # OF MSG. REP.
CLEAR INPUT
III
FLAG
STEP

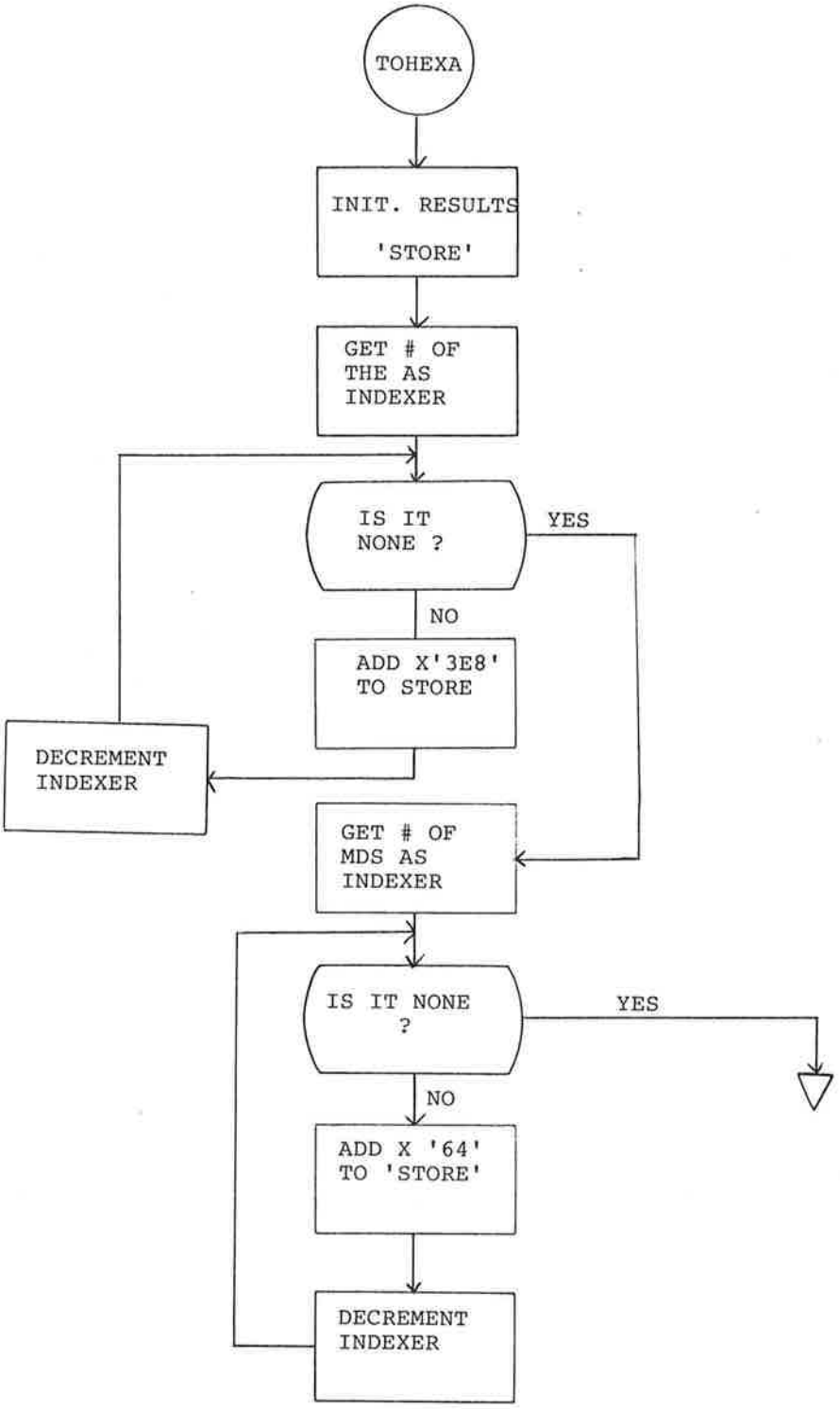

SCENCH
ADJUST THE POSITION COUNTER PROPERLY
INPUT 2 CHAR. ?
ERRMEG
NO
NUMERIC CHARACTERS ?
CHANGE TO NON-ASCII
MASK 12
GET POWERS OF TEN
KITHTO
CONVERT TO HEXA-DECIMAL
TOHEXA
ONE TO SIXTEEN ?
NO
ERRMEG
YES

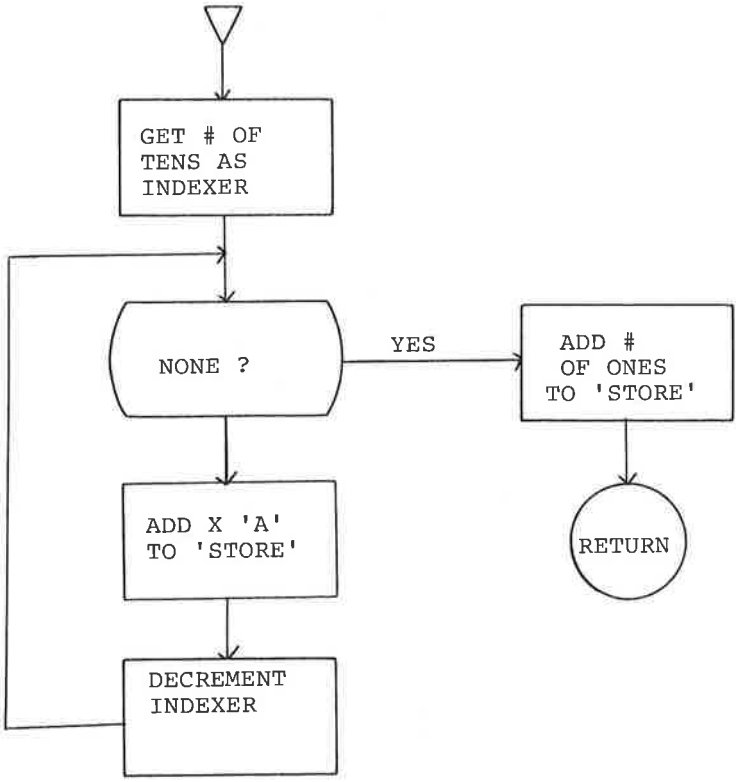

FIRST SCENARIO RESPONSE ?
NO
COMPARE WITH FIRST SCEN # ?
LE
ERRMEG
YES
GT
EQ
STORE NEW FIRST SCEN #
III
STORE NEW LAST SCEN #
III
RESET FIRST SCENARIO FLAG
STEP

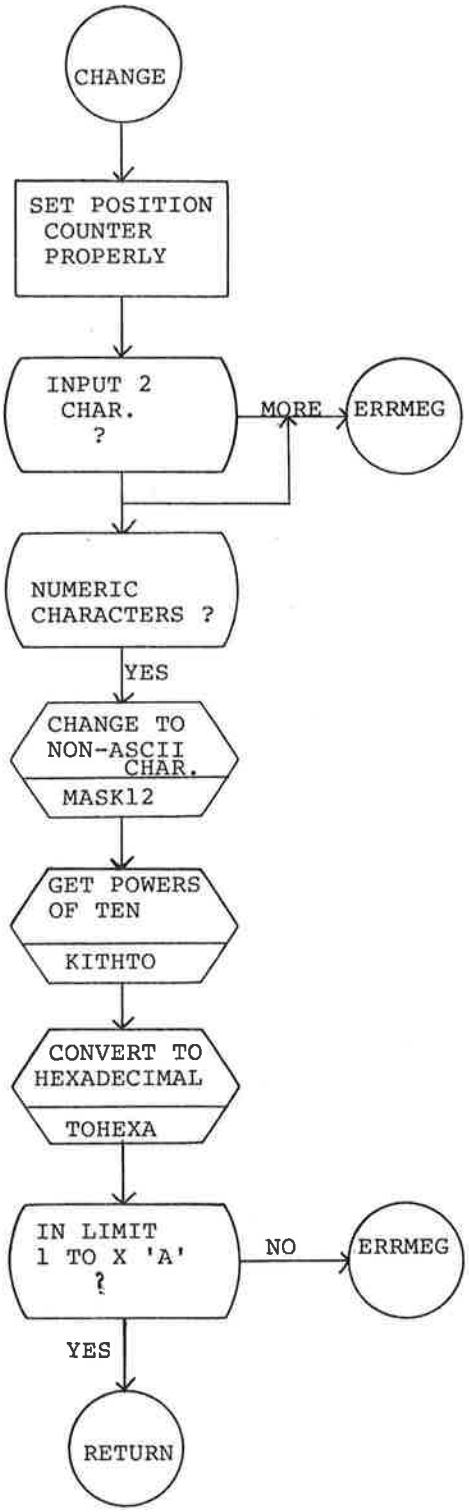

MSGNAK
NAK MSG PRINT
MSGPRT
CONVERT # OF NAKS TO DEC & SHOW IT
THTO/ATOM
STORE RECORD
BUFFST
SET FLAG (FNAK) CLEAR FLAG (FNR)
A
MSGBCS
BCSE MSG PRINT
MSGPRT
CONVERT # OF BCSER TO DEC & SHOW IT
THTO/ATOM
STORE RECORD
BUFFST
SET FLAG (FBCS) CLEAR FLAG (FNAK)
A

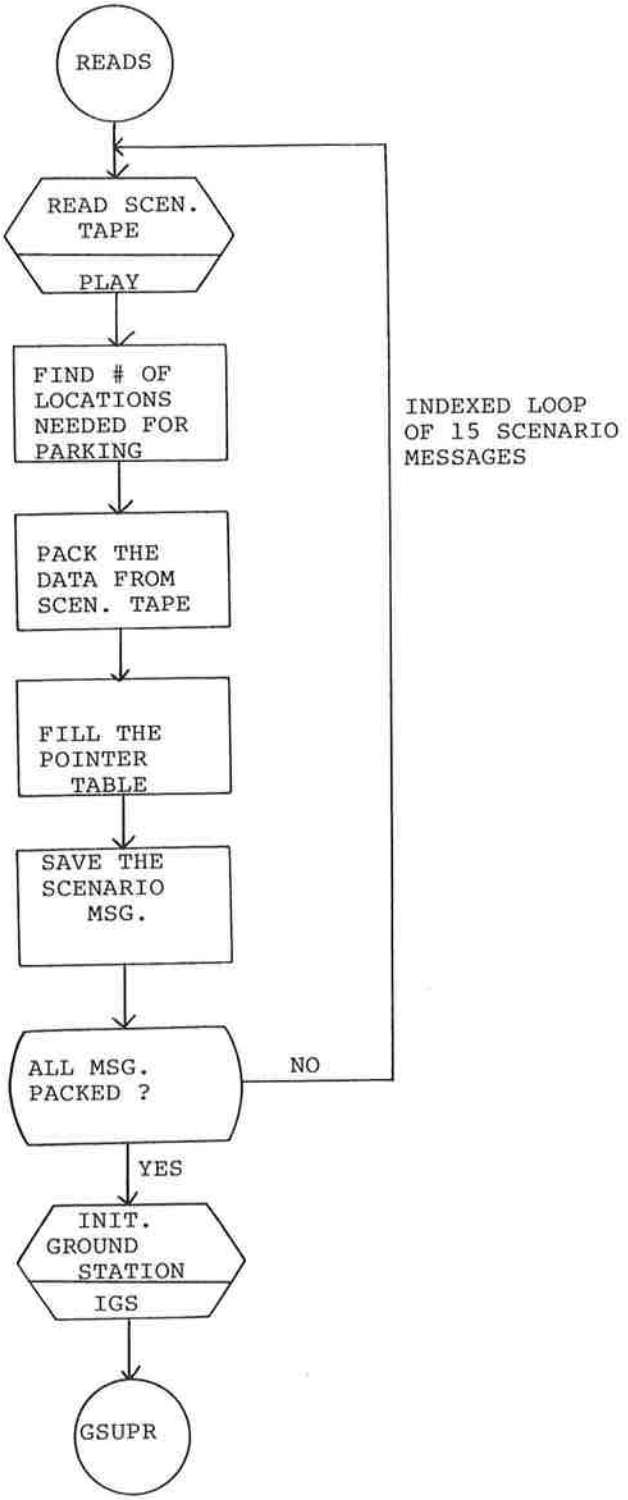

GAME
RECORD THE SYS. ENTRY DATA
RECORD
PLAYBACK THE DATA FROM REC.
PLAY
PRINT THE DATA FROM PLAYBACK
PRINT
STEP

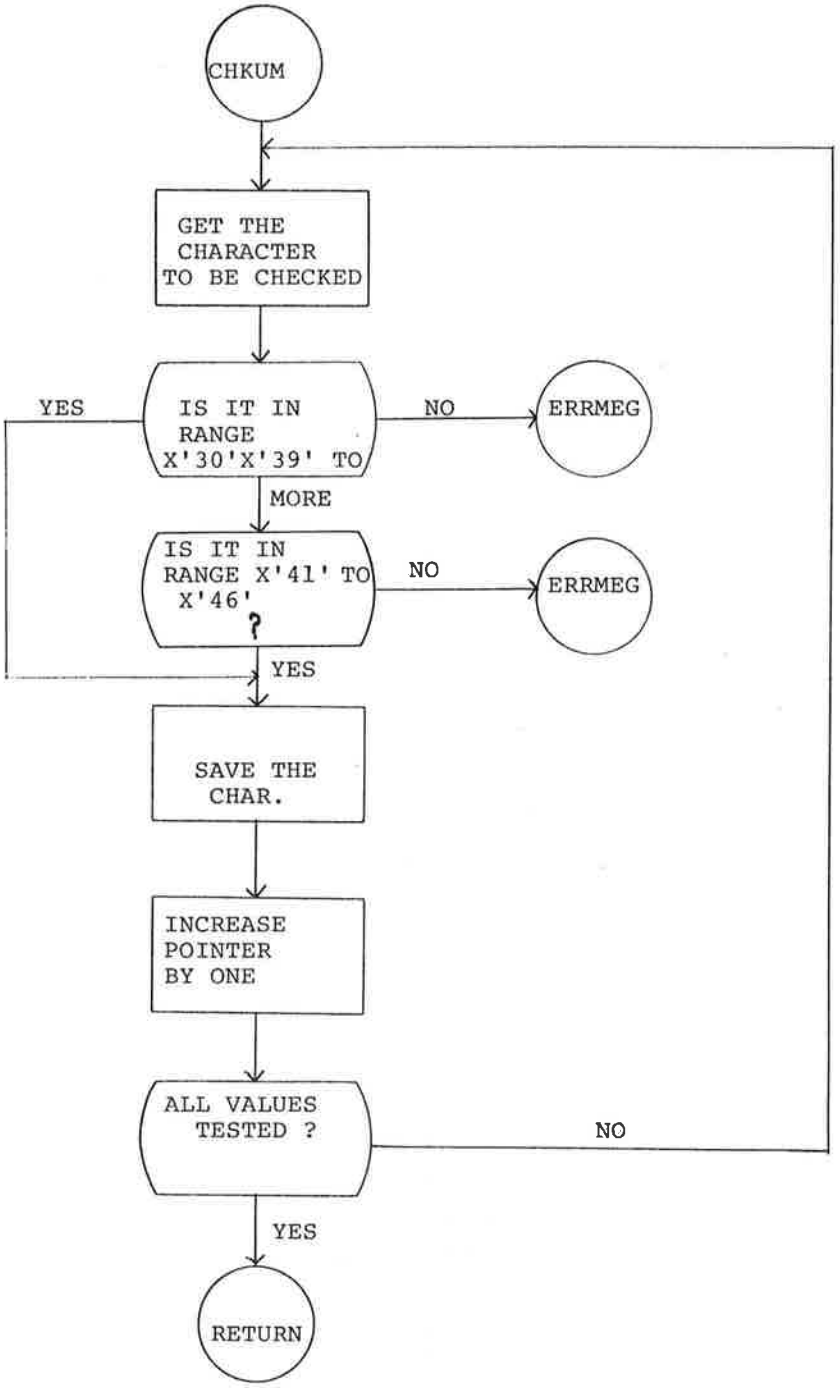

RUBLOO
IS RUBOUT FLAG SET ALREADY ?
YES
NO
PRINT '/'
PRINTC
SET FLAG (FRUB)
SET COUNTER PROPERLY
PRINT LAST CHAR.
PRINTQ
A

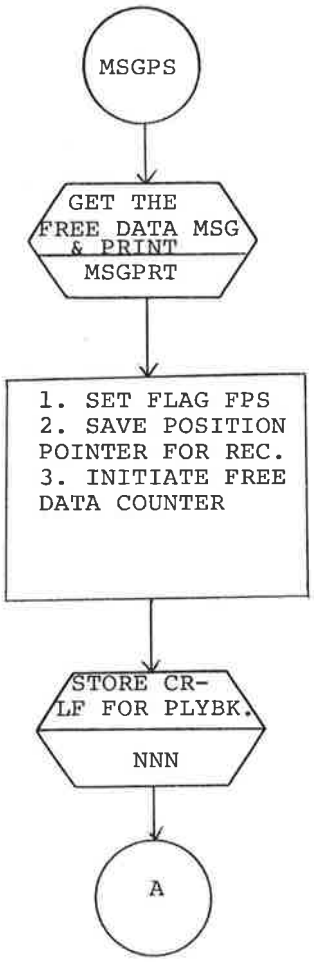

TOHEXA
INIT. RESULTS
'STORE'
GET # OF
THE AS
INDEXER
IS IT
NONE ?
YES
NO
ADD X'3E8'
TO STORE
DECREMENT
INDEXER
GET # OF
MDS AS
INDEXER
IS IT NONE
?
YES
NO
ADD X '64'
TO 'STORE'
DECREMENT
INDEXER

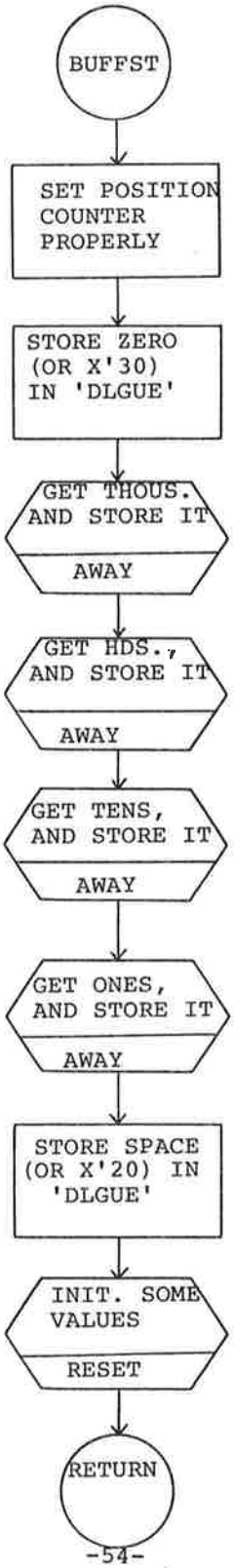

GET # OF
TENS AS
INDEXER
NONE ?
YES
ADD #
OF ONES
TO 'STORE'
ADD X 'A'
TO 'STORE'
RETURN
DECREMENT
INDEXER

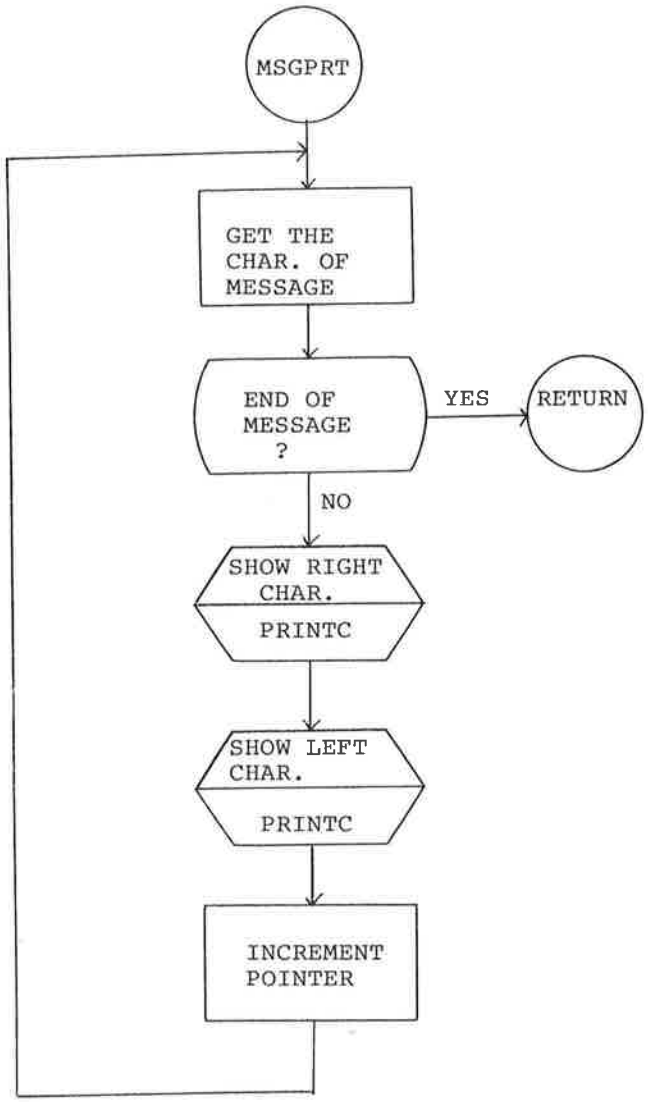

CHANGE
SET POSITION COUNTER PROPERLY
INPUT 2 CHAR. ?
MORE
ERRMEG
NUMERIC CHARACTERS ?
YES
CHANGE TO NON-ASCII CHAR.
MASK12
GET POWERS OF TEN
KITHTO
CONVERT TO HEXADECIMAL
TOHEXA
IN LIMIT 1 TO X 'A' ?
NO
ERRMEG
YES
RETURN

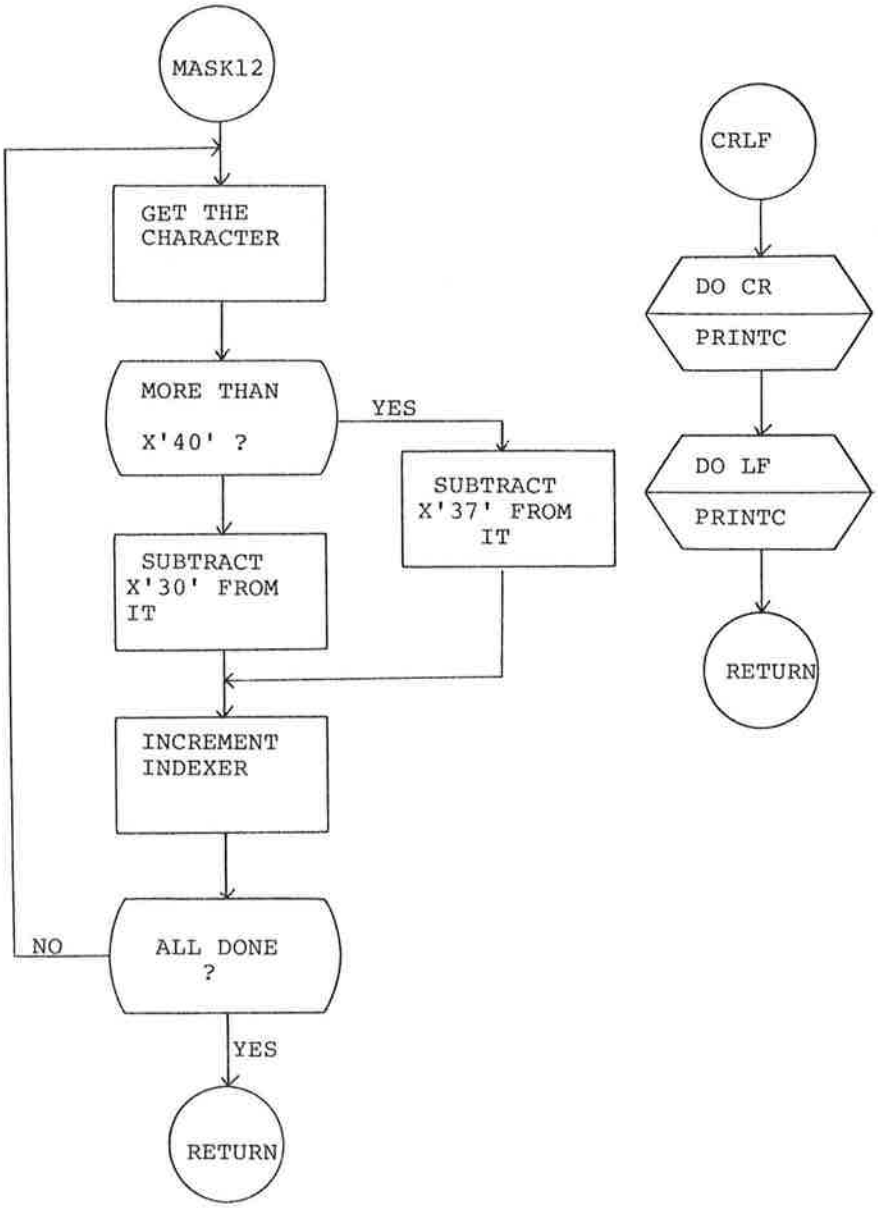

READS
READ SCEN. TAPE
PLAY
FIND # OF LOCATIONS NEEDED FOR PARKING
PACK THE DATA FROM SCEN. TAPE
FILL THE POINTER TABLE
SAVE THE SCENARIO MSG.
ALL MSG. PACKED ?
NO
YES
INIT. GROUND STATION
IGS
GSUPR
INDEXED LOOP OF 15 SCENARIO MESSAGES

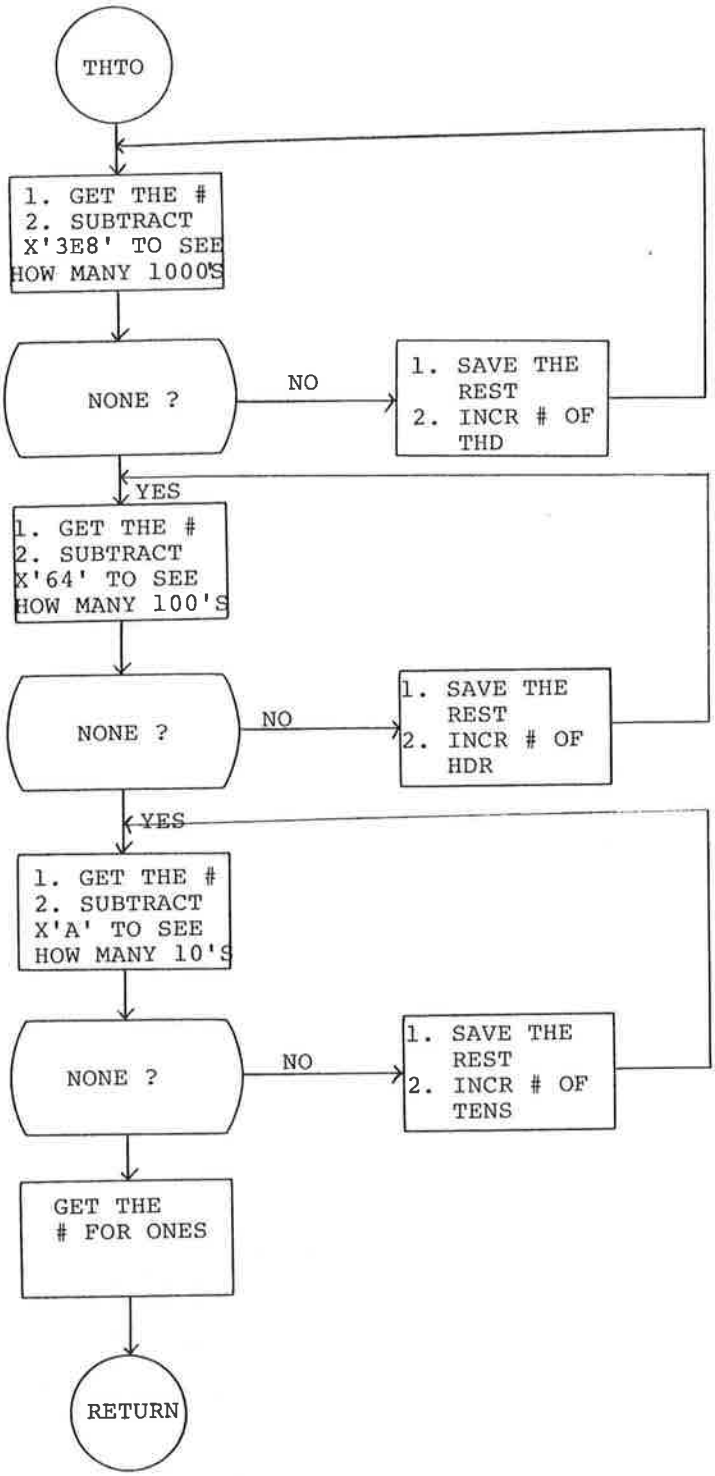

CHKUM
GET THE CHARACTER TO BE CHECKED
IS IT IN RANGE X'30'X'39' TO
YES
NO
ERRMEG
MORE
IS IT IN RANGE X'41' TO X'46' ?
NO
ERRMEG
YES
SAVE THE CHAR.
INCREASE POINTER BY ONE
ALL VALUES TESTED ?
NO
YES
RETURN

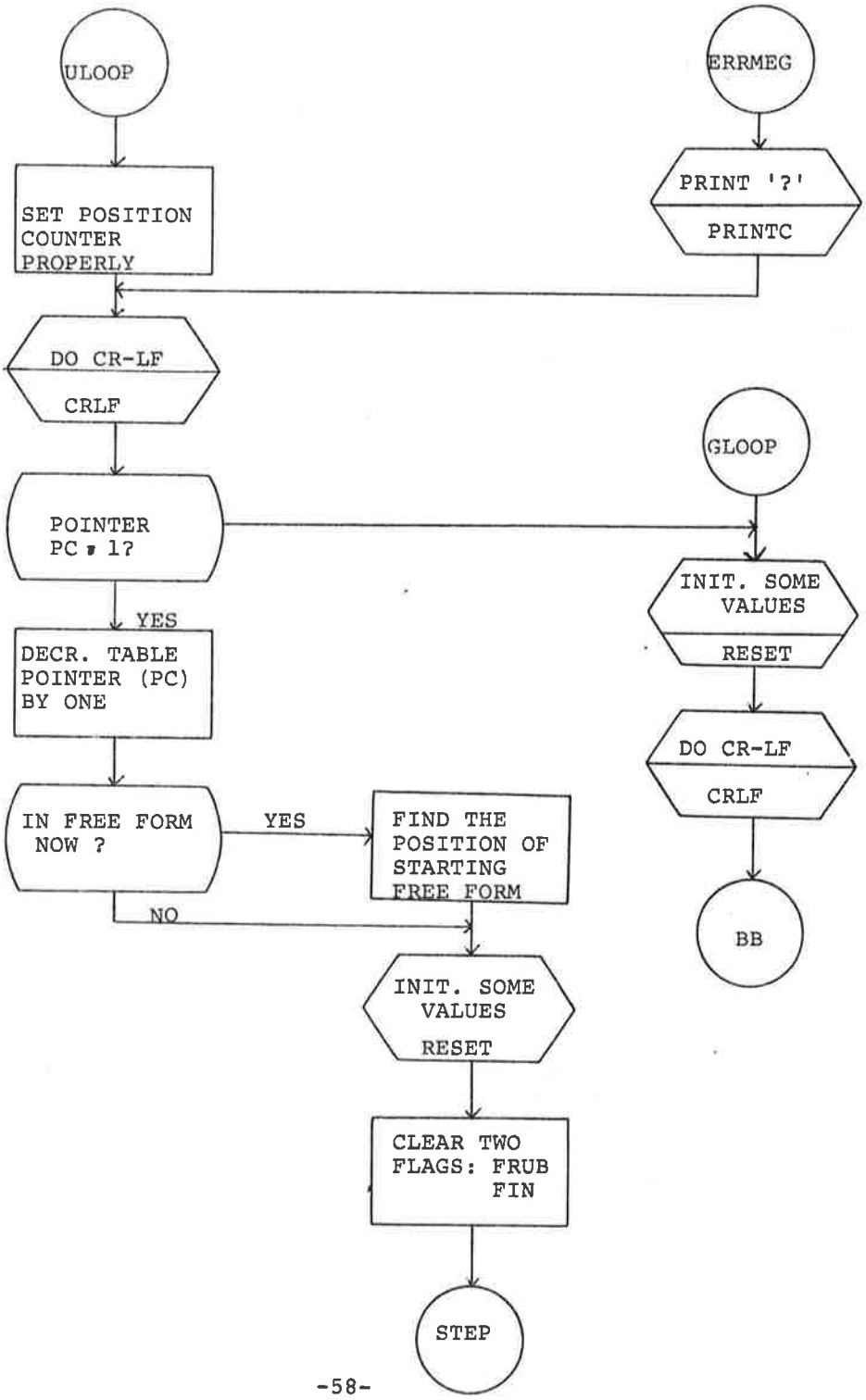

MSGPS
GET THE FREE DATA MSG & PRINT
MSGPRT
1. SET FLAG FPS
2. SAVE POSITION POINTER FOR REC.
3. INITIATE FREE DATA COUNTER
STORE CR-LF FOR PLYBK.
NNN
A

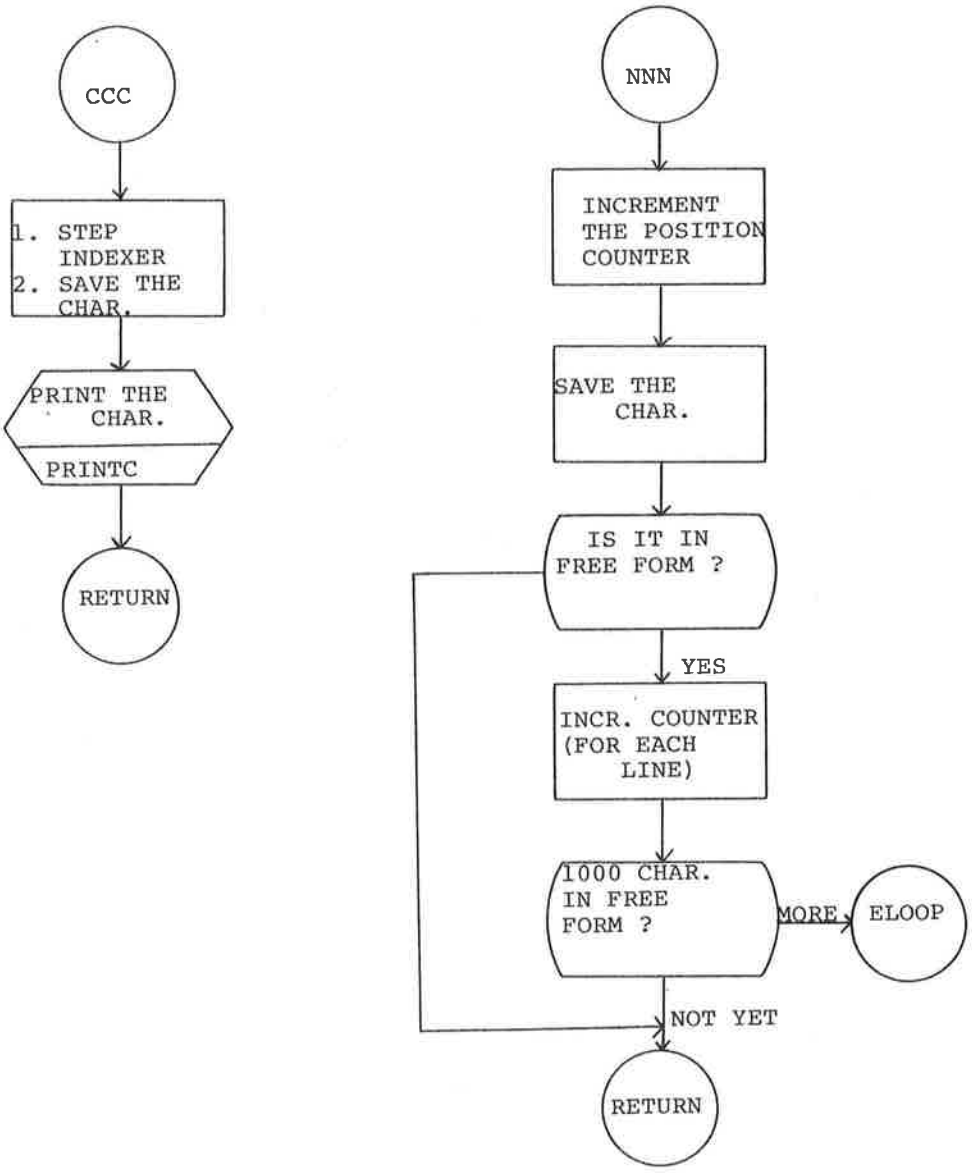

BUFFST
SET POSITION COUNTER PROPERLY
STORE ZERO (OR X'30) IN 'DLGUE'
GET THOUS. AND STORE IT
AWAY
GET HDS., AND STORE IT
AWAY
GET TENS, AND STORE IT
AWAY
GET ONES, AND STORE IT
AWAY
STORE SPACE (OR X'20) IN 'DLGUE'
INIT. SOME VALUES
RESET
RETURN

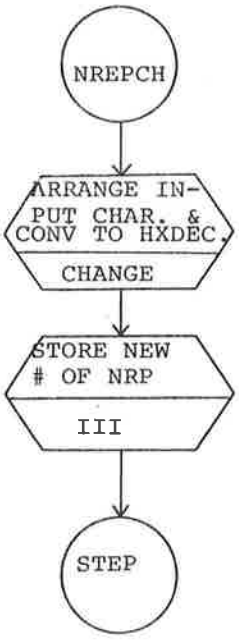

MSGPRT
GET THE CHAR. OF MESSAGE
END OF MESSAGE ?
YES
RETURN
NO
SHOW RIGHT CHAR.
PRINTC
SHOW LEFT CHAR.
PRINTC
INCREMENT POINTER


MASK12
GET THE CHARACTER
MORE THAN X'40' ?
YES
SUBTRACT X'37' FROM IT
SUBTRACT X'30' FROM IT
INCREMENT INDEXER
NO
ALL DONE ?
YES
RETURN
CRLF
DO CR
PRINTC
DO LF
PRINTC
RETURN


THTO
1. GET THE #
2. SUBTRACT
X'3E8' TO SEE
HOW MANY 1000'S
NONE ?
NO
1. SAVE THE
REST
2. INCR # OF
THD
YES
1. GET THE #
2. SUBTRACT
X'64' TO SEE
HOW MANY 100'S
NONE ?
NO
1. SAVE THE
REST
2. INCR # OF
HDR
YES
1. GET THE #
2. SUBTRACT
X'A' TO SEE
HOW MANY 10'S
NONE ?
NO
1. SAVE THE
REST
2. INCR # OF
TENS
GET THE
FOR ONES
RETURN

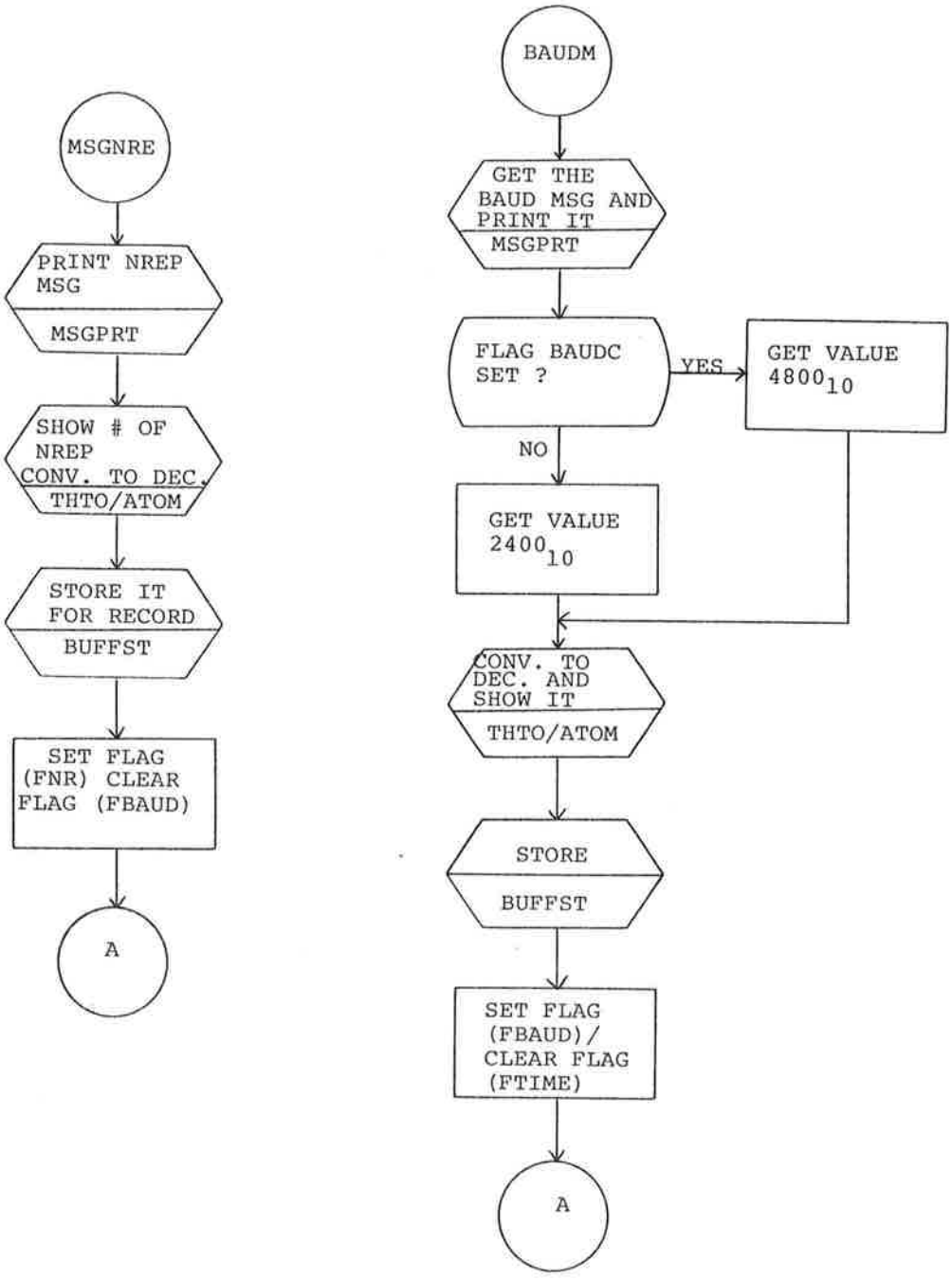

ULOOP
SET POSITION COUNTER PROPERLY
DO CR-LF
CRLF
POINTER PC = 1?
YES
DECR. TABLE POINTER (PC) BY ONE
IN FREE FORM NOW ?
YES
FIND THE POSITION OF STARTING FREE FORM
NO
INIT. SOME VALUES
RESET
CLEAR TWO FLAGS: FRUB FIN
STEP
ERRMEG
PRINT '?'
PRINTC
GLOOP
INIT. SOME VALUES
RESET
DO CR-LF
CRLF
BB

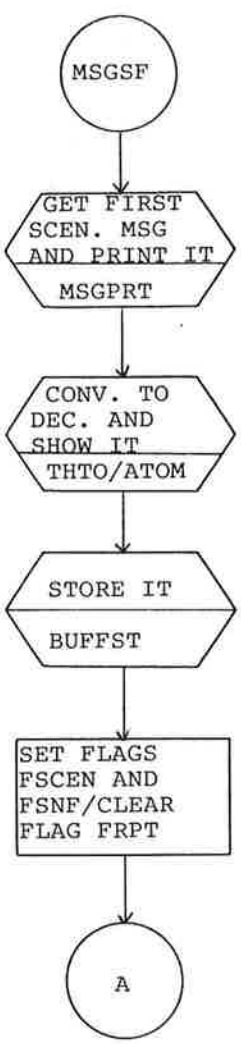

CCC
1. STEP INDEXER
2. SAVE THE CHAR.
PRINT THE CHAR.
PRINTC
RETURN
NNN
INCREMENT THE POSITION COUNTER
SAVE THE CHAR.
IS IT IN FREE FORM ?
YES
INCR. COUNTER (FOR EACH LINE)
1000 CHAR. IN FREE FORM ?
MORE
ELOOP
NOT YET
RETURN

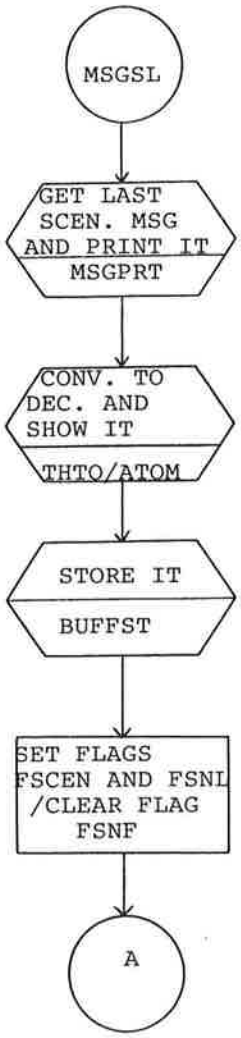

NREPCH
ARRANGE IN-
PUT CHAR. &
CONV TO HXDEC.
CHANGE
STORE NEW
OF NRP
III
STEP

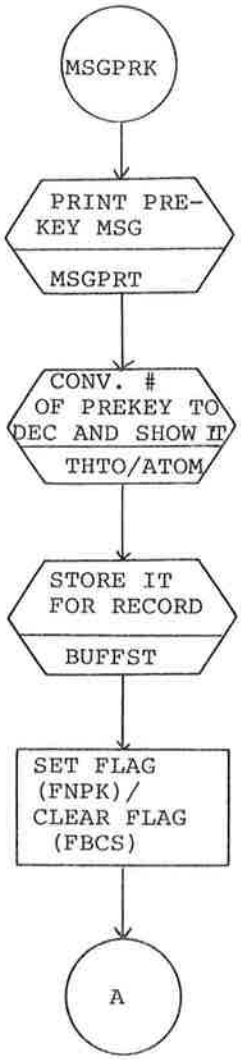

NAKCH
ARRG INPUT
CHAR. & CONV.
TO HEXDEC.
CHANGE
STORE NEW
OF NAK
III
STEP


BCSCH
ARRANGE IN-
PUT CHAR. &
CONV. TO HXDEC.
CHANGE
STORE NEW
OF BCS
III
STEP


MSGNRE
PRINT NREP MSG
MSGPRT
SHOW # OF NREP CONV. TO DEC.
THTO/ATOM
STORE IT FOR RECORD
BUFFST
SET FLAG (FNR) CLEAR FLAG (FBAUD)
A
BAUDM
GET THE BAUD MSG AND PRINT IT
MSGPRT
FLAG BAUDC SET ?
YES
GET VALUE 4800_{10}
NO
GET VALUE 2400_{10}
CONV. TO DEC. AND SHOW IT
THTO/ATOM
STORE
BUFFST
SET FLAG (FBAUD)/ CLEAR FLAG (FTIME)
A


MSGSF
GET FIRST SCEN. MSG AND PRINT IT
MSGPRT
CONV. TO DEC. AND SHOW IT
THTO/ATOM
STORE IT
BUFFST
SET FLAGS FSCEN AND FSNF/CLEAR FLAG FRPT
A


MSGSL
GET LAST SCEN. MSG AND PRINT IT
MSGPRT
CONV. TO DEC. AND SHOW IT
THTO/ATOM
STORE IT
BUFFST
SET FLAGS FSCEN AND FSNL /CLEAR FLAG FSNF
A


MSGPRK
PRINT PRE-KEY MSG
MSGPRT
CONV. # OF PREKEY TO DEC AND SHOW IT
THTO/ATOM
STORE IT FOR RECORD
BUFFST
SET FLAG (FNPK)/ CLEAR FLAG (FBCS)
A


MSGRPT
PRINT MSG REPEAT
MSGPRT
CONV # OF MSG REPEAT TO DEC AND SHOW IT
THTO/ATOM
STORE IT FOR RECORD
BUFFST
SET FLAG (FRPT)/ CLEAR FLAG (FNPK)
A

IV. GROUND STATION COMMUNICATIONS SUPERVISOR

This section describes the function and operation of the supervisory logic of the ground station.

A. GSUPR

The main controlling sequence of the communication handler resides is this module. This module makes the decisions relative to message order, message acceptability, and it also continually adjusts the running statistical values which reflect system performance.

When the ground station dialogue initialization (DATALP-DATALF-DATALD) module completes its task, it calls an initialization subroutine for GSUPR, and then releases control to GSUPR, where a "header" line is printed on the status terminal.

The first task of GSUPR is to establish communication with the airborne station. The keyboard is enabled for operator input (to allow operator inserted messages and/or the "quit" command). A system entry poll is transmitted continually until an acceptable response is detected. Each transmission of the system entry poll has an incremented message identifier (with the alpha character being an ASCII "blank"). As soon as an acceptable response is detected, a completion record is written on the data collection cassette and a "." is printed on the status (secondary/auxiliary) terminal. No other recording, nor status printing is done; meaning that there are no records nor indicators of unsuccessful system entry polls.

The next message sent to the aircraft is a time synchronization poll. This poll contains, in its text field, the current value of the ground station's time-of-day clock, and the value of two ground station parameters: KNPK and KNAKS. The time poll is transmitted until it is successfully acknowledged. Each subsequent transmission of this poll has an incremented message identifier and incremented alpha character (blank, then A through Z), until one of ENAKS, ENR, or EBCSF is exhausted. If one of the counters becomes exhausted, the message identifier is incremented, and the alpha character is resequenced with a "blank".

Once the system entry poll and time poll are successfully completed, the message communication discipline begins. The discipline requires that message transmissions occur in a fixed order, beginning with the scenario message number identified by SFIRST and continuing, in increments of one, up to, and including, the scenario message number identified by SLAST. Each message is sent repeatedly, until the repeat counter is exhausted, before the next one is sent. The repeat condition is specified by EMSGR. Two types of occurrances cause the stepping of EMSGR. It is tallied whenever a successful poll occurs. It is also tallied whenever one of the error parameters (EBCSF, ENAKS, and ENR) becomes exhausted. Each transmission contains an incremented message identifier. In addition, each stepping of EMSGR causes the poll to contain a "blank" alpha character. Each subsequent re-transmission of the same poll, due to some error, has

unconditional branch is executed to NS, to advance to the next scenario message.

If the response checks non-valid, flag BCSE is interrogated. If set, one of five conditions has occurred. There has occurred either a "pure" BCS error, or the seen message was too long, or too short, or had an improper aircraft identifier, or there is a bit compare error (echo type only). Any one of the last four conditions can happen in conjunction with any of the other five conditions. Any of the five conditions causes an error record to be filed and an indicator to be printed on the status terminal. The priority of check is in the following descending order:

Type	Indicator	Condition
W	W	Aircraft I.D.
L	L	Too Long
S	S	Too Short
I	I	Bit Compare
B	B	"Pure" BCS

The indicator is printed in upper case if an 'ACK' is seen, and in lower case if a 'NAK' is seen. All of these conditions cause error records to be recorded, but only one is filed per poll, based upon the shown high-to-low priority. Either ECHOBC or NEBC is tallied and GSUPR is prompted to send a 'NAK'. If EBCSF is not now caused to be exhausted, the message is re-transmitted with an incremented identifier and stepped alpha. If EBCSF is now exhausted, the same type of transaction record is filed, and an unconditional branch

to NS is executed, to advance to the next scenario message.

If the response is not valid, and if BCSE is not raised, then the received message contained a 'NAK'. An 'N' type error record is filed and either ECHONK or NENK is tallied, an 'N' is printed on the status terminal. If ENAKS is not now caused to be exhausted, the message is re-transmitted with an incremented message identifier and a stepped alpha. If ENAKS is now caused to be exhausted, an 'N' type completion record is filed and NS is entered unconditionally, to advance to the next scenario message.

At NS, the keyboard buffer is released, if appropriate (if this is a response to a keyboard message). If either the timing poll of general poll flag is raised, GSUPR is recycled. If a scenario message is to be next, subroutine SMSTEP (GU) is called to step EMSGR and to move in the next scenario message (if necessary), and GSUPR is recycled.

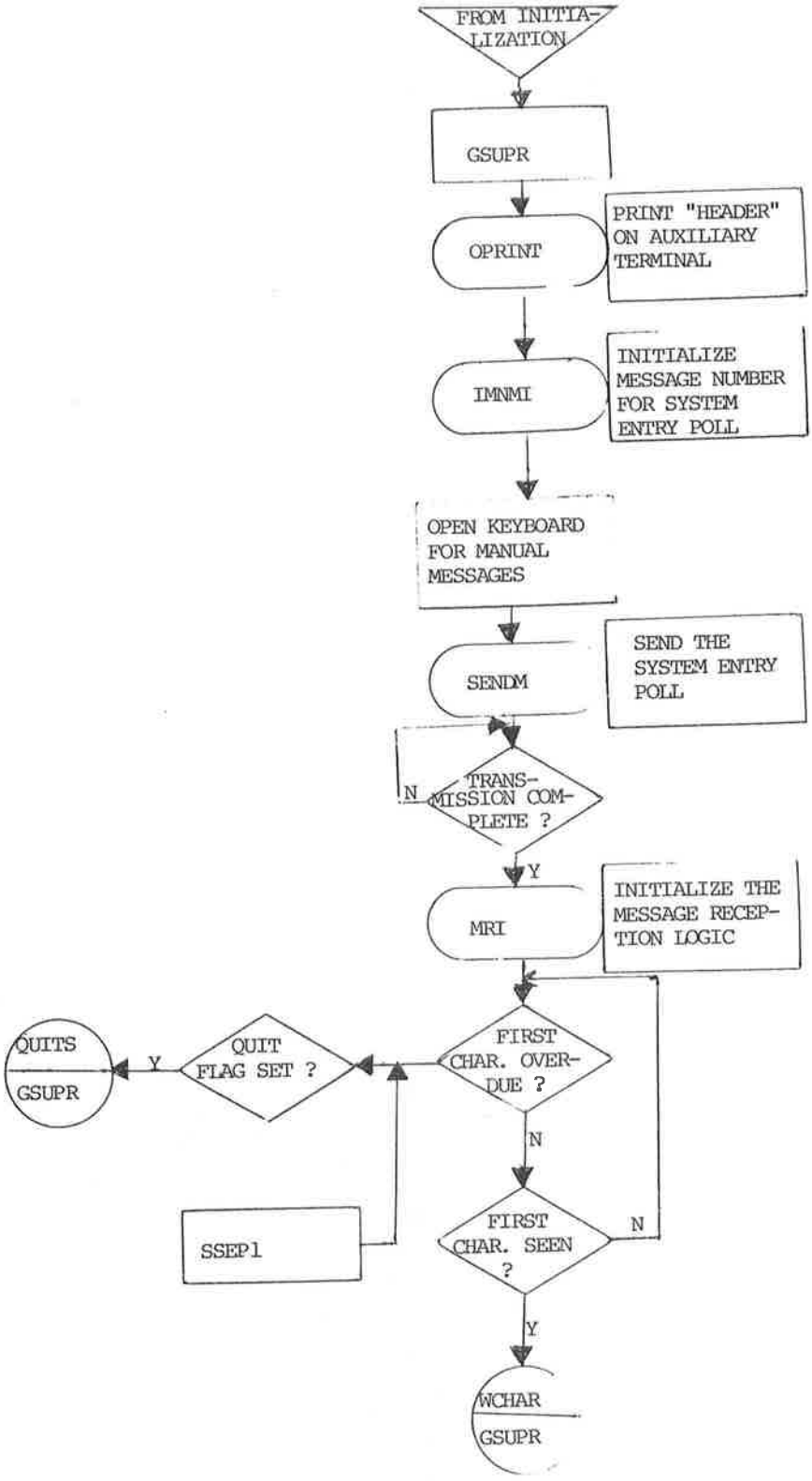
Whenever the quit flag is raised, an unconditional branch is executed to QUITS where a simulated keyboard message is sent to the aircraft. This message tells the operator of the airborne station to abort the airborne system operation. The flag to accept operator is disabled. An unconditional branch to QUIT (GSUPRA) is executed after acknowledgement of the message ("Q" on the status terminal) or non-acknowledgement ("U" on status terminal) fifty times.

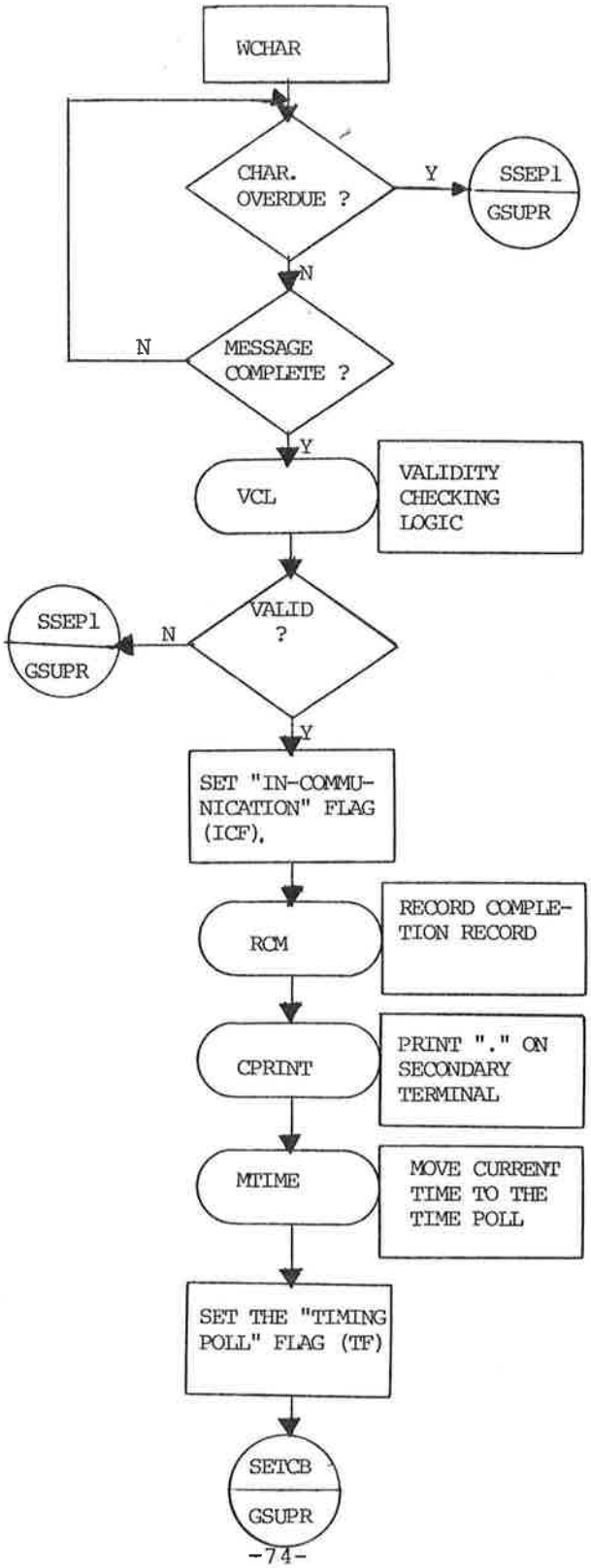
The validity checking logic (VCL) assumes no bit errors will occur. If any of the length or aircraft identifier errors occur, the flag BCSE is set. Flag BCSE is set to one by the message reception logic (MSGREC in module GSUPRA) whenever a received message contains a BCS error. If BCSE is set to one, then the message is not interrogated for a possible printer output. If BCSE is not set to one, the message is interrogated for the characteristics requiring printer output and is printed, if required. The printing occurs on the secondary (status) terminal, and is preceded by the time-of-day.

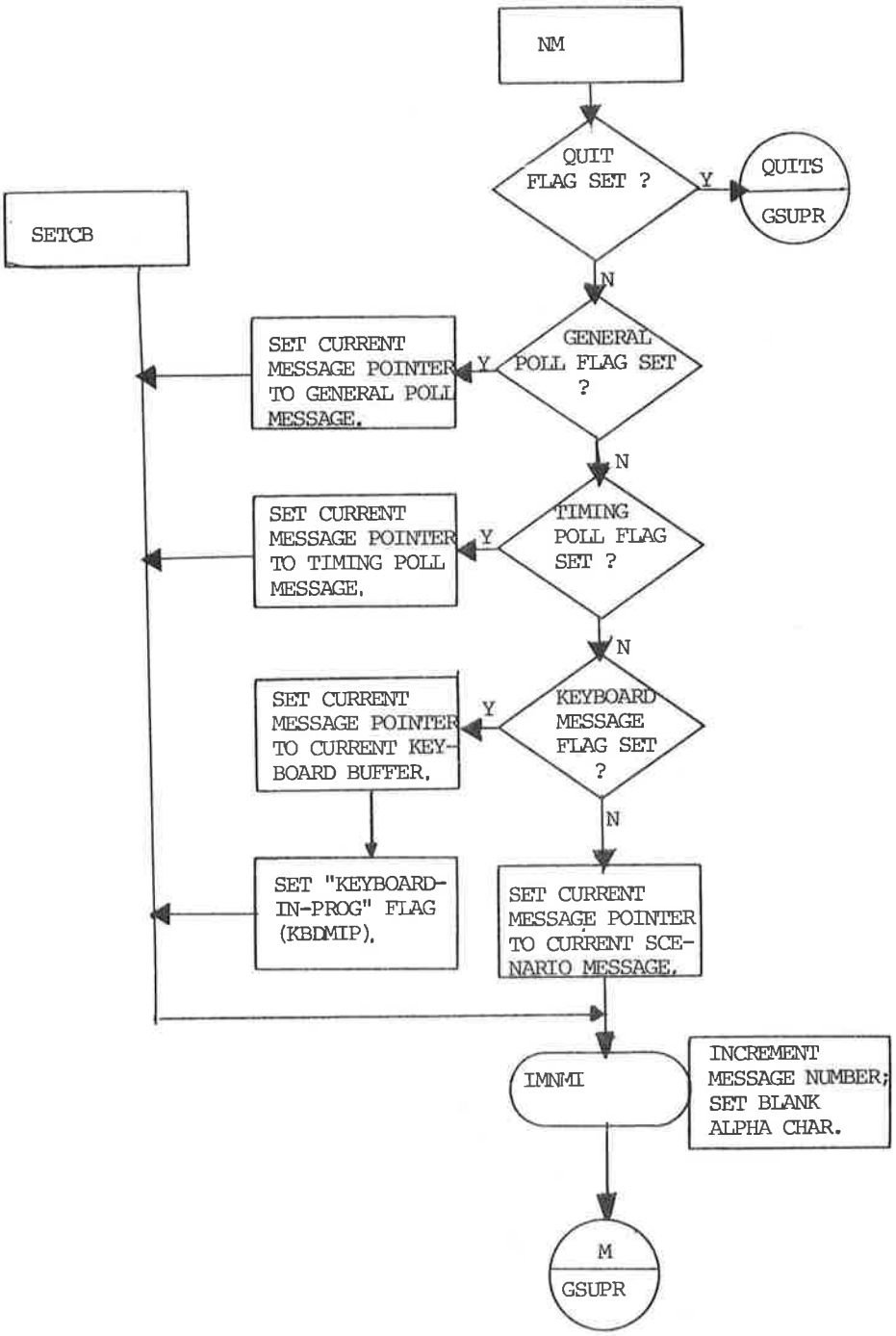
If the message contains transparent text (mode H) all characters are bit-by-bit compared with the originating text. Bit errors, if any, are tallied in a double precision counter labeled ECHOBE and flag BCOME is set to one.

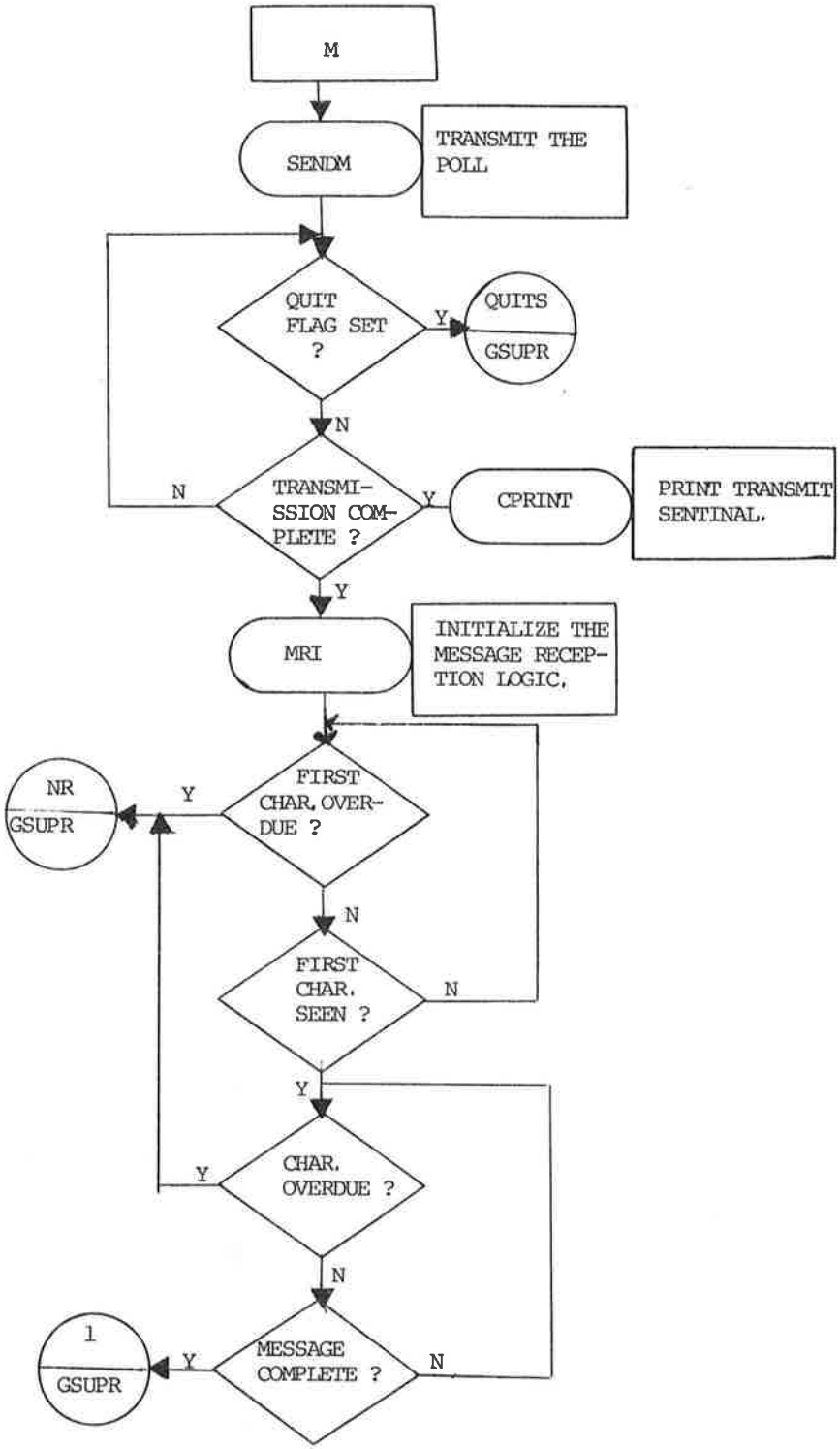
If the message doesn't contain an "ACK", then the originating message is reloaded with the current time, if it was a time re-synchronization poll. Flag NAKE is set to one. If an "ACK" is detected, the general poll flag (GPF) is turned off.

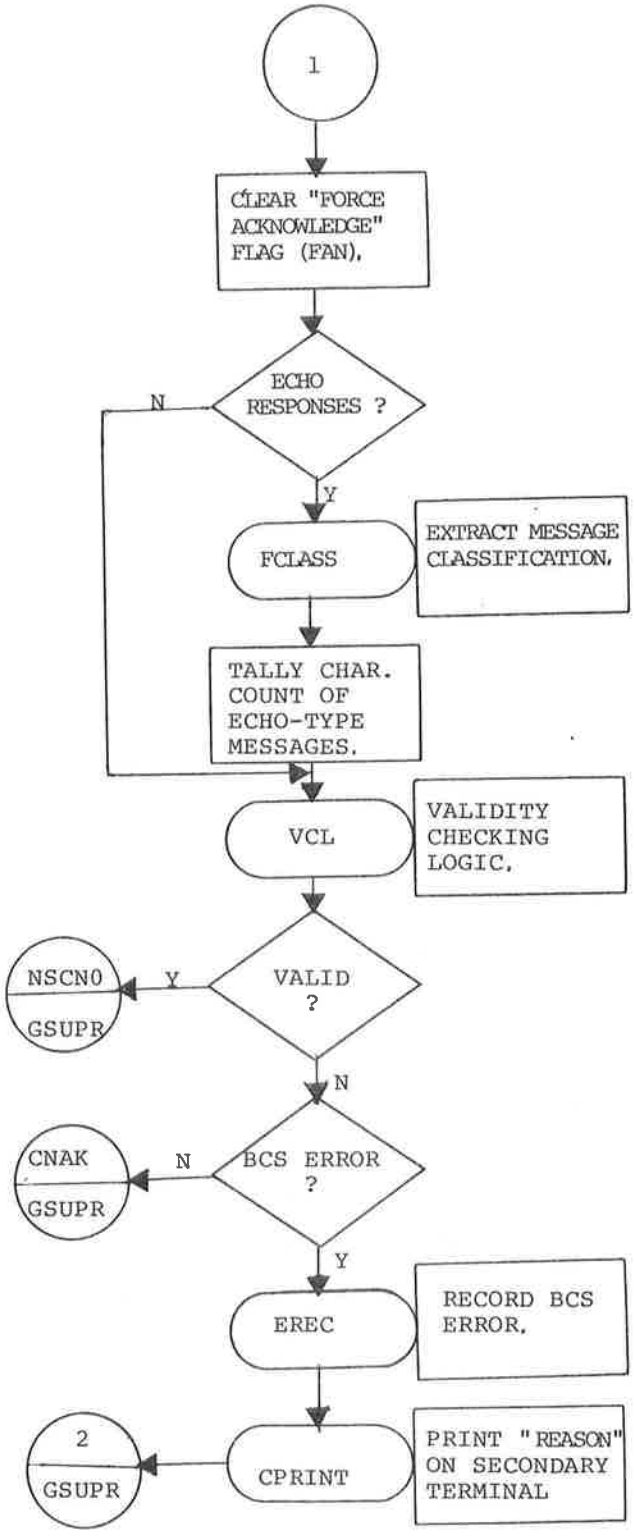
If any bit compare errors occurred, flag BCSE is set to one. If flag BCSE is set to one the normal subroutine return is executed; else, the instruction following the normal return is executed.

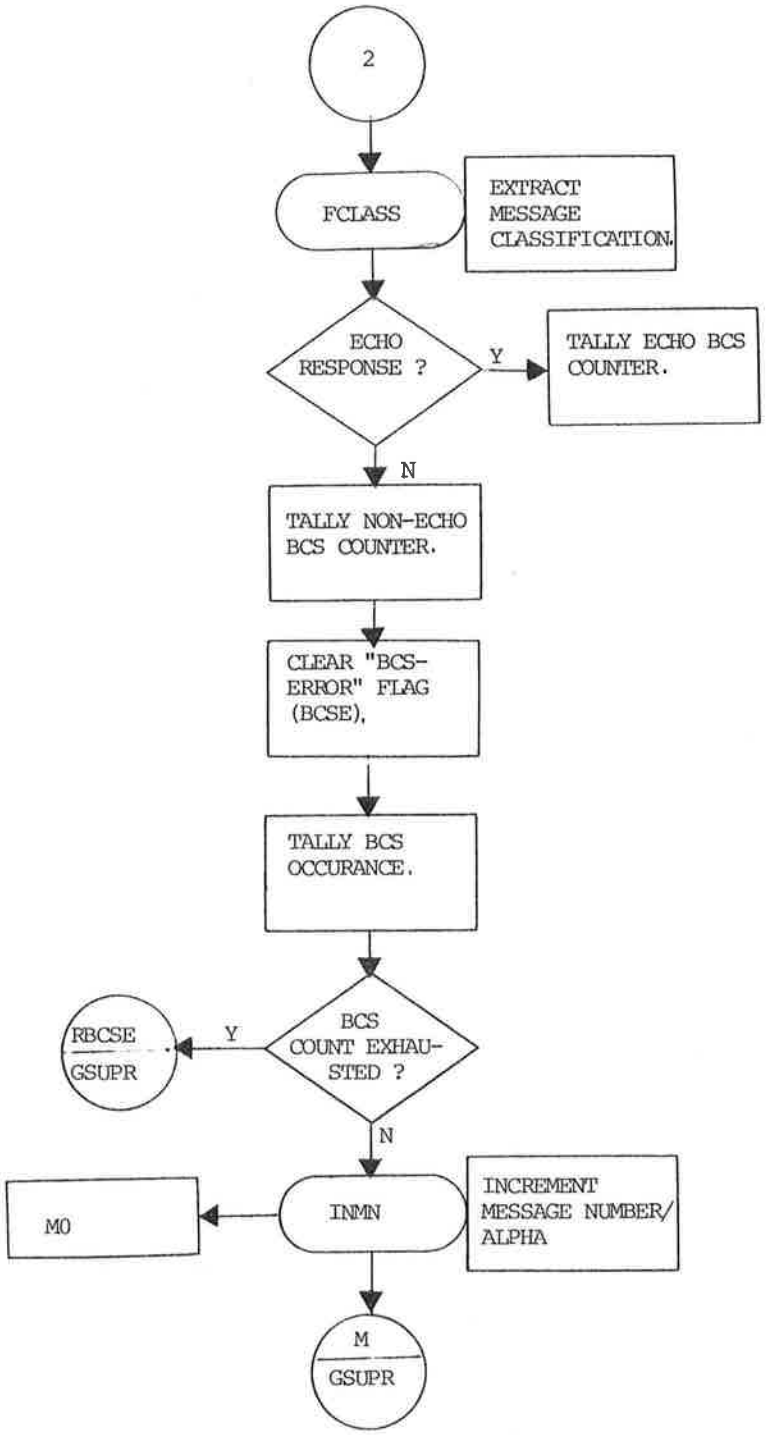

FROM INITIA-
LIZATION
GSUPR
OPRINT
PRINT "HEADER"
ON AUXILIARY
TERMINAL
IMNMI
INITIALIZE
MESSAGE NUMBER
FOR SYSTEM
ENTRY POLL
OPEN KEYBOARD
FOR MANUAL
MESSAGES
SENDM
SEND THE
SYSTEM ENTRY
POLL
N
TRANS-
MISSION COM-
PLETE ?
Y
MRI
INITIALIZE THE
MESSAGE RECEP-
TION LOGIC
QUITS
GSUPR
Y
QUIT
FLAG SET ?
FIRST
CHAR. OVER-
DUE ?
N
SSEP1
FIRST
CHAR. SEEN
?
N
Y
WCHAR
GSUPR

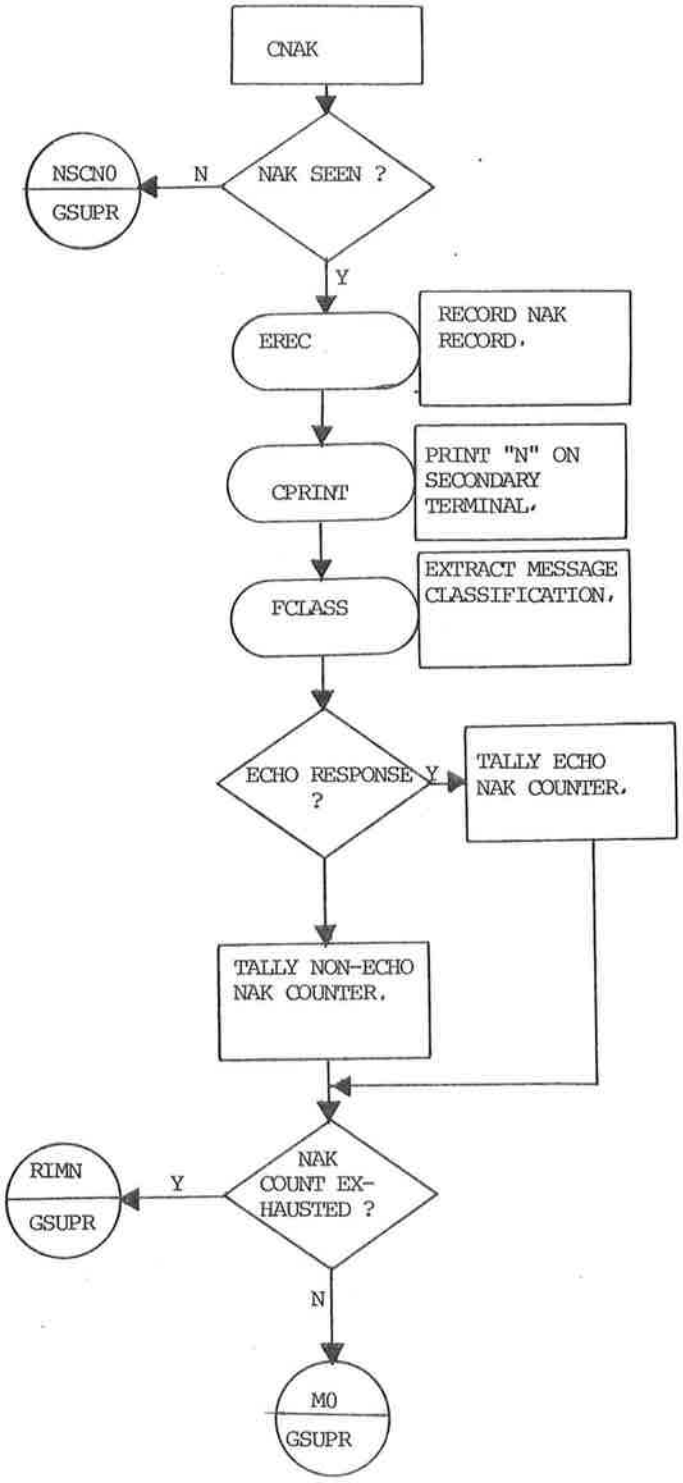

WCHAR
CHAR. OVERDUE ?
Y
SSEP1
GSUPR
N
MESSAGE COMPLETE ?
N
Y
VCL
VALIDITY CHECKING LOGIC
VALID ?
N
SSEP1
GSUPR
Y
SET "IN-COMMU-NICATION" FLAG (ICF).
RCM
RECORD COMPLE-TION RECORD
CPRINT
PRINT "." ON SECONDARY TERMINAL
MTIME
MOVE CURRENT TIME TO THE TIME POLL
SET THE "TIMING POLL" FLAG (TF)
SETCB
GSUPR

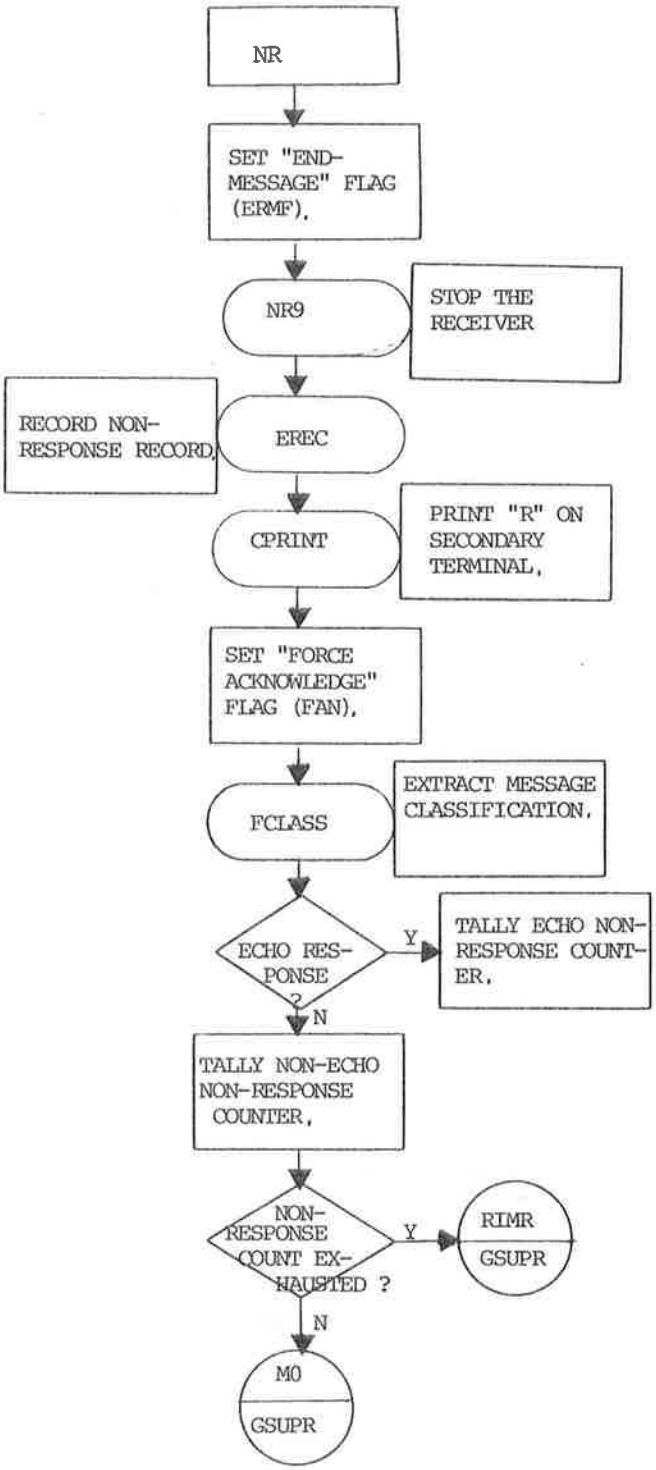

NM
QUIT FLAG SET ?
Y
QUITS
GSUPR
N
SETCB
GENERAL POLL FLAG SET ?
Y
SET CURRENT MESSAGE POINTER TO GENERAL POLL MESSAGE.
N
TIMING POLL FLAG SET ?
Y
SET CURRENT MESSAGE POINTER TO TIMING POLL MESSAGE.
N
KEYBOARD MESSAGE FLAG SET ?
Y
SET CURRENT MESSAGE POINTER TO CURRENT KEY-BOARD BUFFER.
N
SET CURRENT MESSAGE POINTER TO CURRENT SCE-NARIO MESSAGE.
SET "KEYBOARD-IN-PROG" FLAG (KBDMIP).
IMNMI
INCREMENT MESSAGE NUMBER; SET BLANK ALPHA CHAR.
M
GSUPR

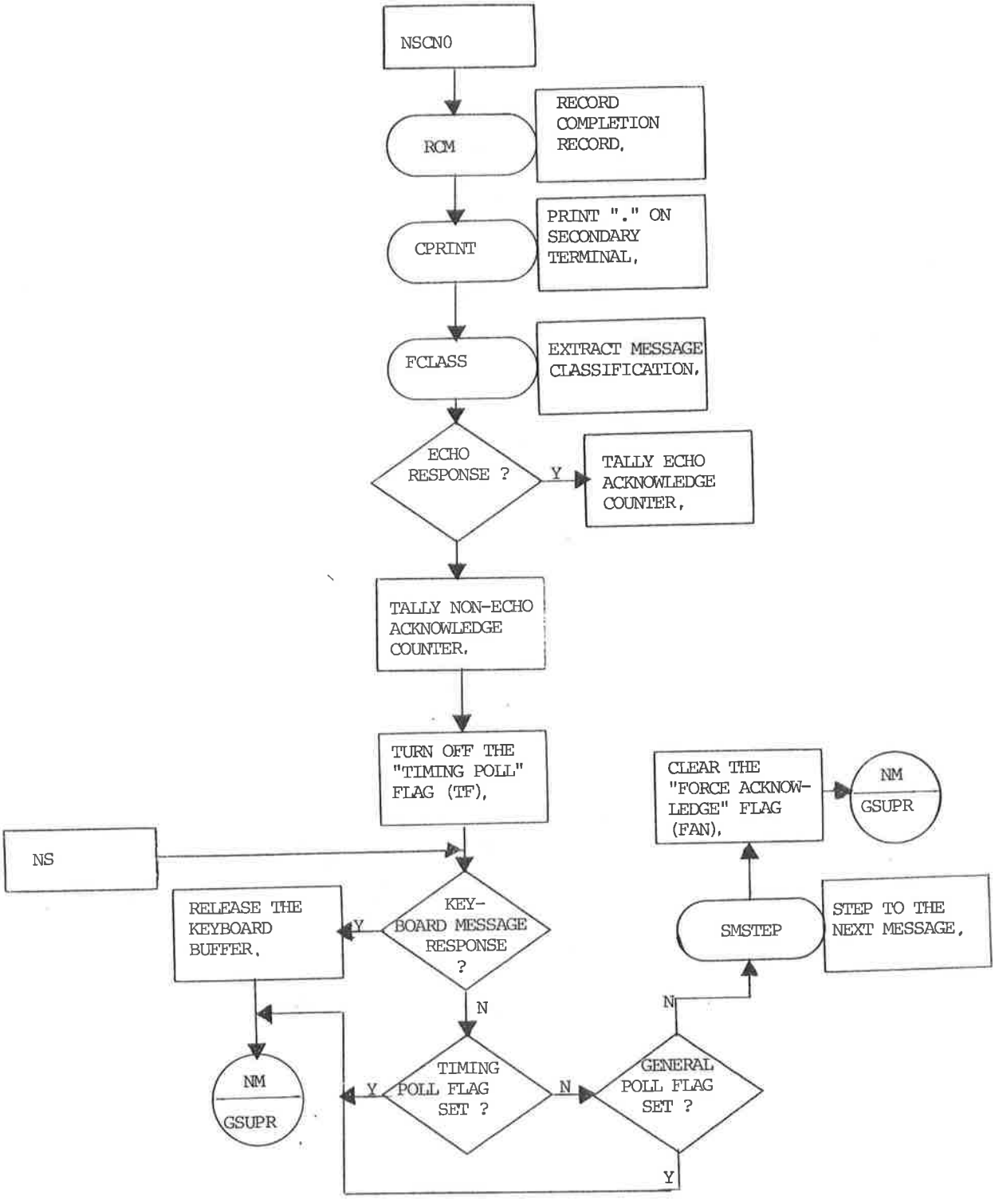

M
SENDM
TRANSMIT THE POLL
QUIT FLAG SET ?
Y
QUITS
GSUPR
N
TRANSMI-SSION COM-PLETE ?
N
Y
CPRINT
PRINT TRANSMIT SENTINAL.
Y
MRI
INITIALIZE THE MESSAGE RECEP-TION LOGIC.
NR
GSUPR
Y
FIRST CHAR. OVER-DUE ?
N
FIRST CHAR. SEEN ?
N
Y
Y
CHAR. OVERDUE ?
N
1
GSUPR
Y
MESSAGE COMPLETE ?
N

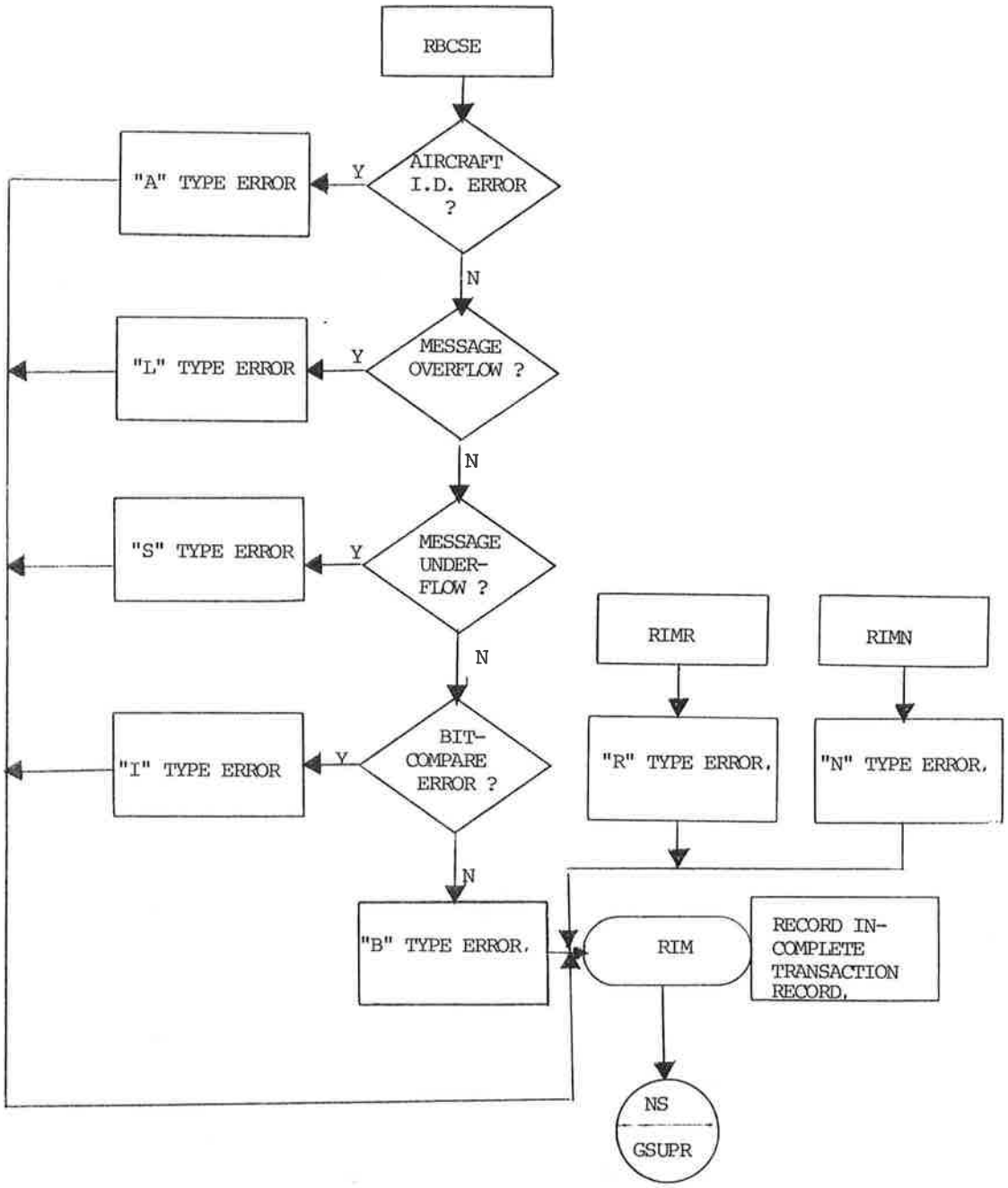

1
CLEAR "FORCE ACKNOWLEDGE" FLAG (FAN).
ECHO RESPONSES ?
N
Y
FCLASS
EXTRACT MESSAGE CLASSIFICATION.
TALLY CHAR. COUNT OF ECHO-TYPE MESSAGES.
VCL
VALIDITY CHECKING LOGIC.
VALID ?
Y
NSCN0
GSUPR
N
BCS ERROR ?
N
CNAK
GSUPR
Y
EREC
RECORD BCS ERROR.
CPRINT
PRINT "REASON" ON SECONDARY TERMINAL
2
GSUPR

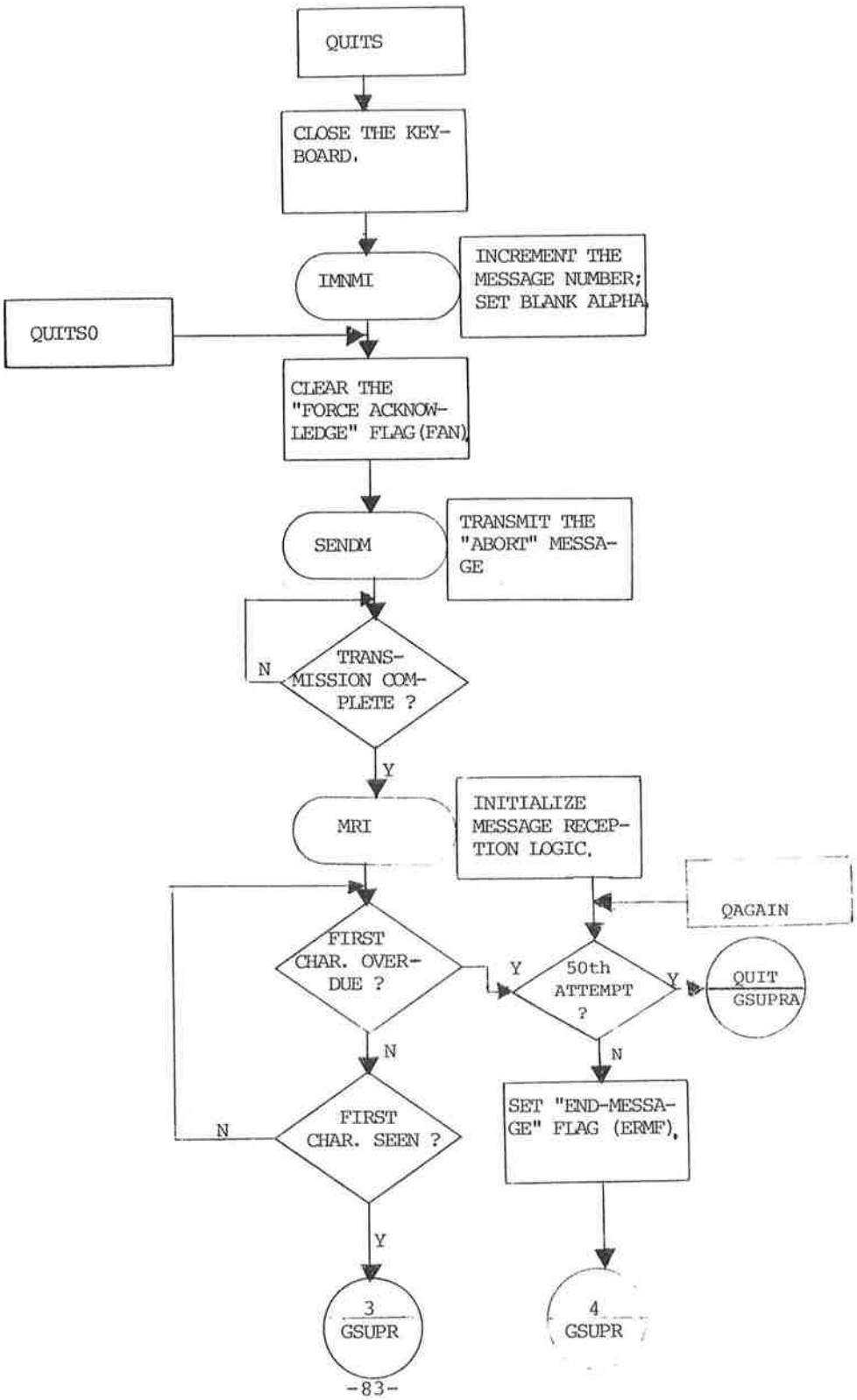

2
FCLASS
EXTRACT MESSAGE CLASSIFICATION.
ECHO RESPONSE ?
Y
TALLY ECHO BCS COUNTER.
N
TALLY NON-ECHO BCS COUNTER.
CLEAR "BCS-ERROR" FLAG (BCSE).
TALLY BCS OCCURANCE.
BCS COUNT EXHAU-STED ?
Y
RBCSE
GSUPR
N
INMN
INCREMENT MESSAGE NUMBER/ ALPHA
M0
M
GSUPR

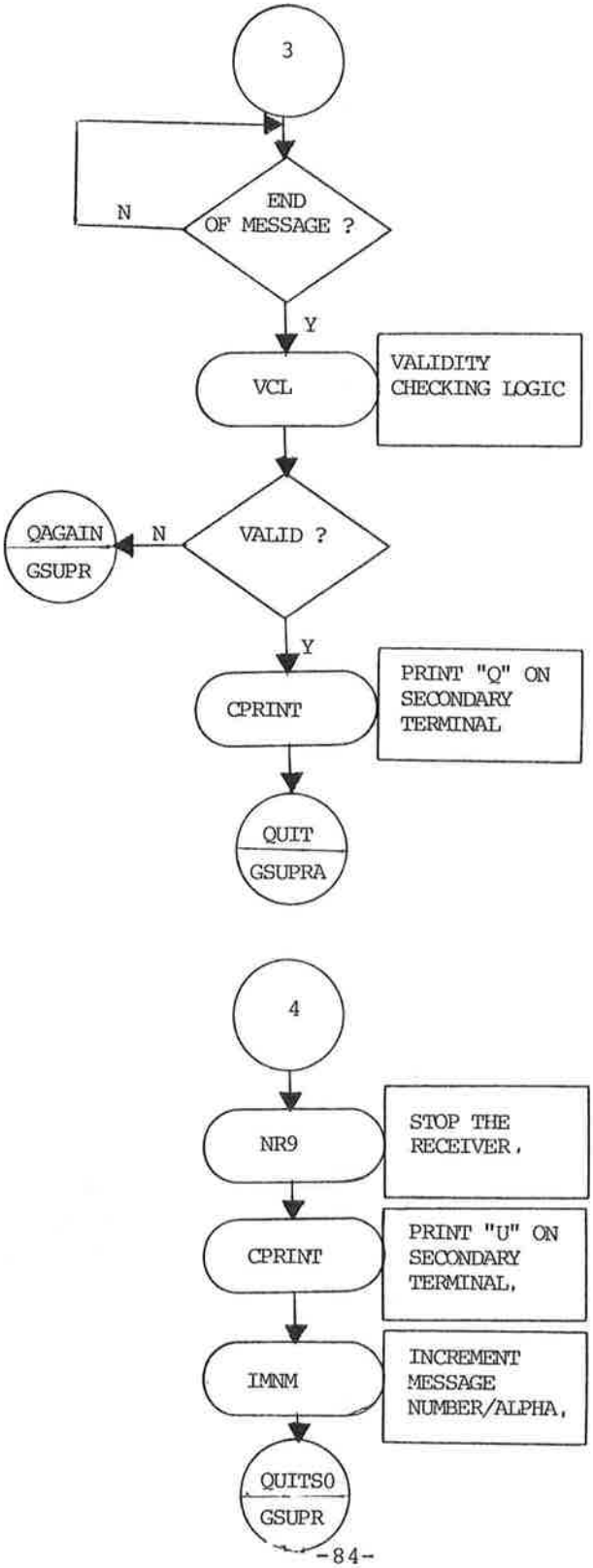

CNAK
NAK SEEN ?
N
NSCN0
GSUPR
Y
EREC
RECORD NAK RECORD.
CPRINT
PRINT "N" ON SECONDARY TERMINAL.
FCLASS
EXTRACT MESSAGE CLASSIFICATION.
ECHO RESPONSE ?
Y
TALLY ECHO NAK COUNTER.
TALLY NON-ECHO NAK COUNTER.
NAK COUNT EX-HAUSTED ?
Y
RIMN
GSUPR
N
M0
GSUPR

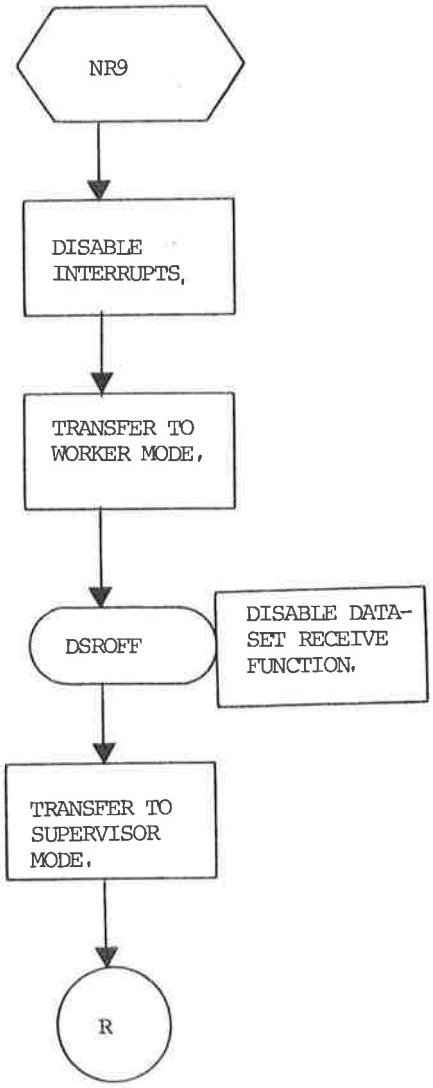

NR
SET "END-MESSAGE" FLAG (ERMF).
NR9
STOP THE RECEIVER
RECORD NON-RESPONSE RECORD.
EREC
CPRINT
PRINT "R" ON SECONDARY TERMINAL.
SET "FORCE ACKNOWLEDGE" FLAG (FAN).
FCLASS
EXTRACT MESSAGE CLASSIFICATION.
ECHO RES-PONSE ?
Y
TALLY ECHO NON-RESPONSE COUNT-ER.
N
TALLY NON-ECHO NON-RESPONSE COUNTER.
NON-RESPONSE COUNT EX-HAUSTED ?
Y
RIMR
GSUPR
N
M0
GSUPR

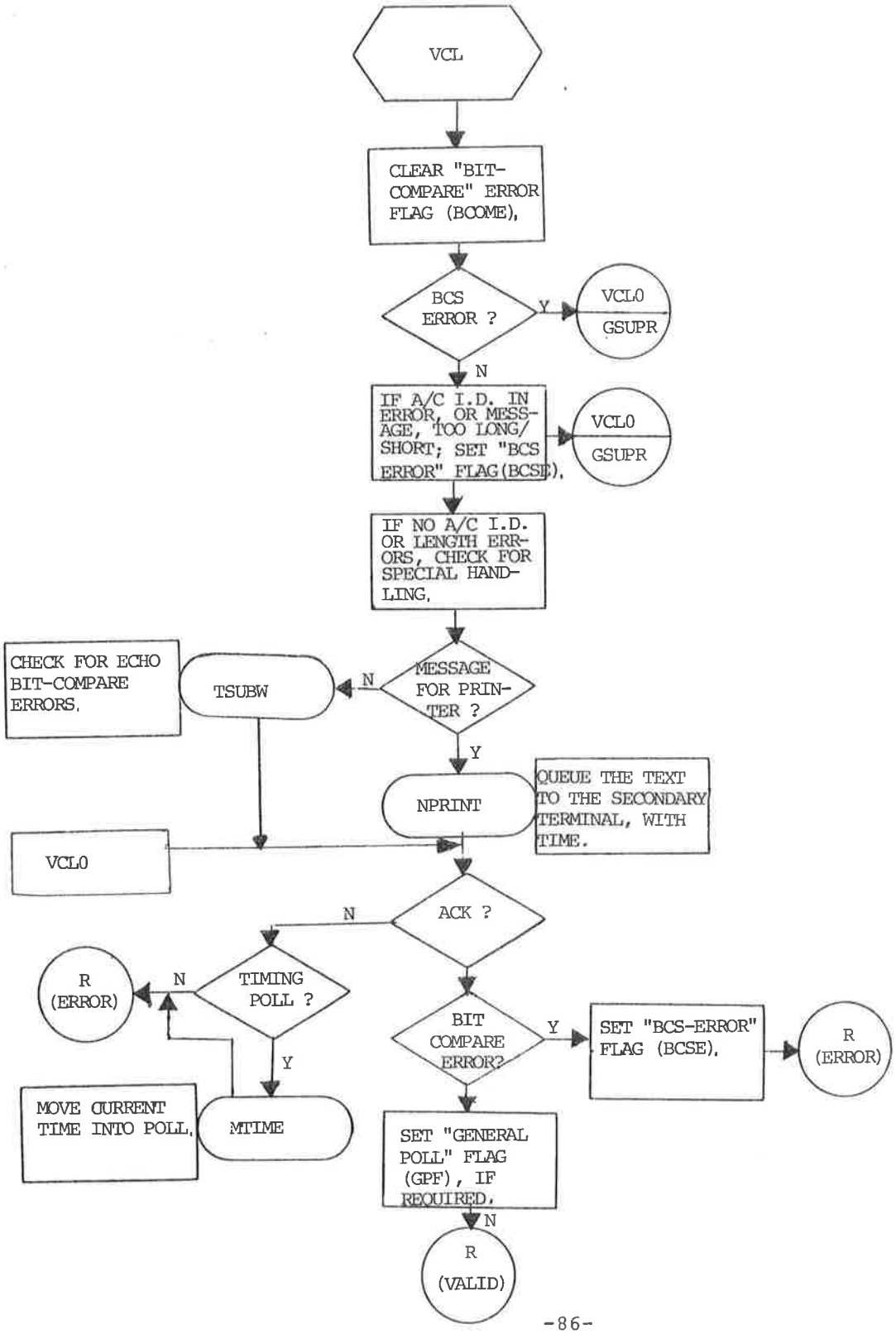

NSCN0
RCM
RECORD COMPLETION RECORD.
CPRINT
PRINT "." ON SECONDARY TERMINAL.
FCLASS
EXTRACT MESSAGE CLASSIFICATION.
ECHO RESPONSE ?
Y
TALLY ECHO ACKNOWLEDGE COUNTER.
TALLY NON-ECHO ACKNOWLEDGE COUNTER.
TURN OFF THE "TIMING POLL" FLAG (TF).
NS
KEY-BOARD MESSAGE RESPONSE ?
Y
RELEASE THE KEYBOARD BUFFER.
N
TIMING POLL FLAG SET ?
Y
NM / GSUPR
N
GENERAL POLL FLAG SET ?
N
Y
SMSTEP
STEP TO THE NEXT MESSAGE.
CLEAR THE "FORCE ACKNOW-LEDGE" FLAG (FAN).
NM / GSUPR


RBCSE
AIRCRAFT I.D. ERROR ?
Y
"A" TYPE ERROR
N
MESSAGE OVERFLOW ?
Y
"L" TYPE ERROR
N
MESSAGE UNDER-FLOW ?
Y
"S" TYPE ERROR
N
BIT-COMPARE ERROR ?
Y
"I" TYPE ERROR
N
"B" TYPE ERROR.
RIMR
"R" TYPE ERROR.
RIMN
"N" TYPE ERROR.
RIM
RECORD IN-COMPLETE TRANSACTION RECORD.
NS
GSUPR


QUITS
CLOSE THE KEY-BOARD.
IMNMI
INCREMENT THE MESSAGE NUMBER; SET BLANK ALPHA.
QUITSO
CLEAR THE "FORCE ACKNOW-LEDGE" FLAG (FAN).
SENDM
TRANSMIT THE "ABORT" MESSA-GE
N
TRANS-MISSION COM-PLETE ?
Y
MRI
INITIALIZE MESSAGE RECEP-TION LOGIC.
QAGAIN
FIRST CHAR. OVER-DUE ?
Y
50th ATTEMPT ?
Y
QUIT GSUPRA
N
N
FIRST CHAR. SEEN ?
SET "END-MESSA-GE" FLAG (ERMF).
N
Y
3 GSUPR
4 GSUPR


3
END
OF MESSAGE ?
N
Y
VCL
VALIDITY
CHECKING LOGIC
QAGAIN
GSUPR
N
VALID ?
Y
CPRINT
PRINT "Q" ON
SECONDARY
TERMINAL
QUIT
GSUPRA
4
NR9
STOP THE
RECEIVER.
CPRINT
PRINT "U" ON
SECONDARY
TERMINAL,
IMNM
INCREMENT
MESSAGE
NUMBER/ALPHA,
QUITS0
GSUPR


NR9
DISABLE INTERRUPTS.
TRANSFER TO WORKER MODE.
DSROFF
DISABLE DATA-SET RECEIVE FUNCTION.
TRANSFER TO SUPERVISOR MODE.
R


VCL
CLEAR "BIT-COMPARE" ERROR FLAG (BCOME),
BCS ERROR ?
Y
VCL0
GSUPR
N
IF A/C I.D. IN ERROR, OR MESS-AGE, TOO LONG/ SHORT; SET "BCS ERROR" FLAG(BCSE),
VCL0
GSUPR
IF NO A/C I.D. OR LENGTH ERR-ORS, CHECK FOR SPECIAL HAND-LING,
CHECK FOR ECHO BIT-COMPARE ERRORS,
TSUBW
N
MESSAGE FOR PRIN-TER ?
Y
NPRINT
QUEUE THE TEXT TO THE SECONDARY TERMINAL, WITH TIME.
VCL0
N
ACK ?
R (ERROR)
N
TIMING POLL ?
Y
MOVE CURRENT TIME INTO POLL,
MTIME
BIT COMPARE ERROR?
Y
SET "BCS-ERROR" FLAG (BCSE),
R (ERROR)
SET "GENERAL POLL" FLAG (GPF), IF REQUIRED,
N
R (VALID)

B. GSUPRA

This module is comprised solely of subroutines or functions which serve the ground supervisor (GSUPR).

Subroutine IGS serves as the initializing link between the dialogue function and the on-line communication function. It initializes variables of both operational and statistical nature. It is a one-shot subroutine, used only once per experiment. Its function is to insure that statistical information is not biased by previous experiments, and to set operational parameters, values, and flags to their proper initial state.

Subroutine SENDM is used as the pass-through for sending a message. Its purpose is to accept a buffer that is ready for transmission except for the two BCS characters, and the technical acknowledgement character. Based upon the condition of flag FAN, the technical acknowledgement character is forced to either "ACK" or "NAK". Location LSTCC is set with the number of characters in the message text. The entire buffer is rippled through the BCS calculation, and the resultant BCS characters are jammed on the end of the buffer. The transmission flag MD is set to one and the output subroutine DSOUT is called. As soon as control is returned from DSOUT, control is released from SENDM.

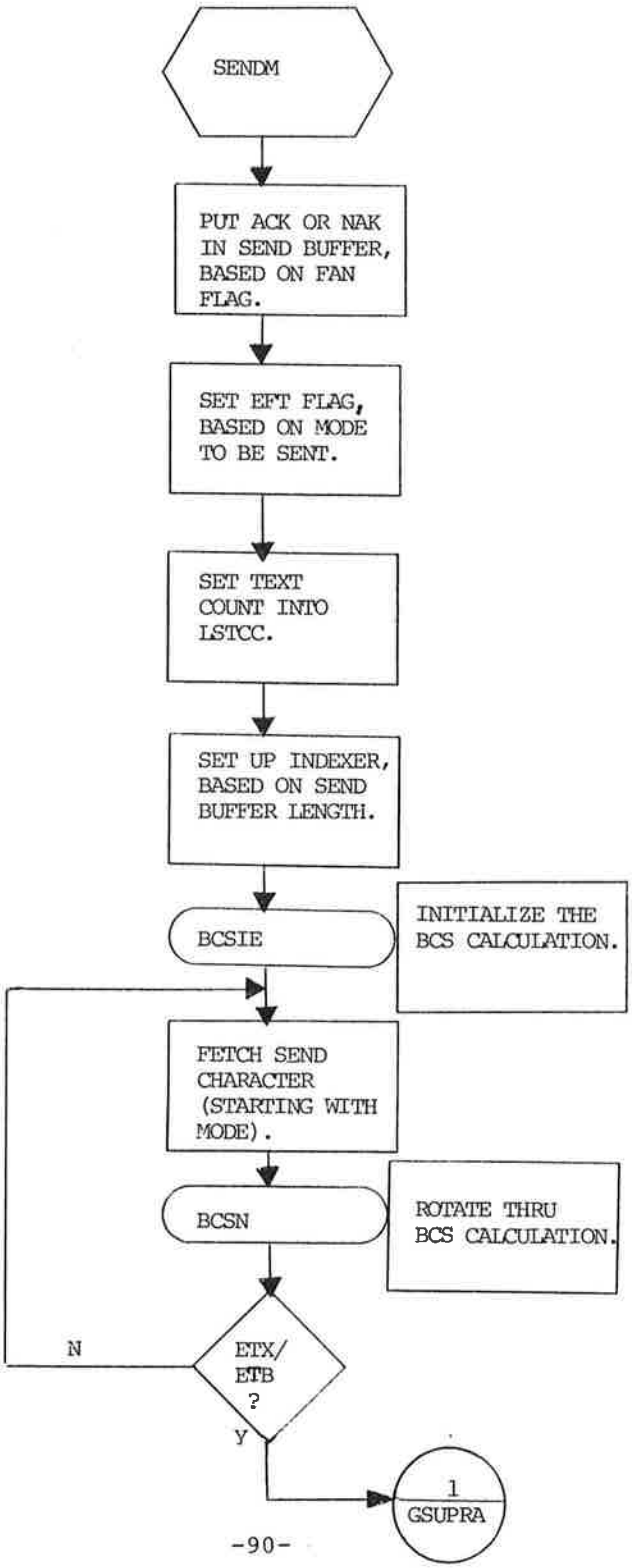
Subroutine MRI is used to initialize the logic for the message repection subroutine MSGREC. The reception area is cleared to

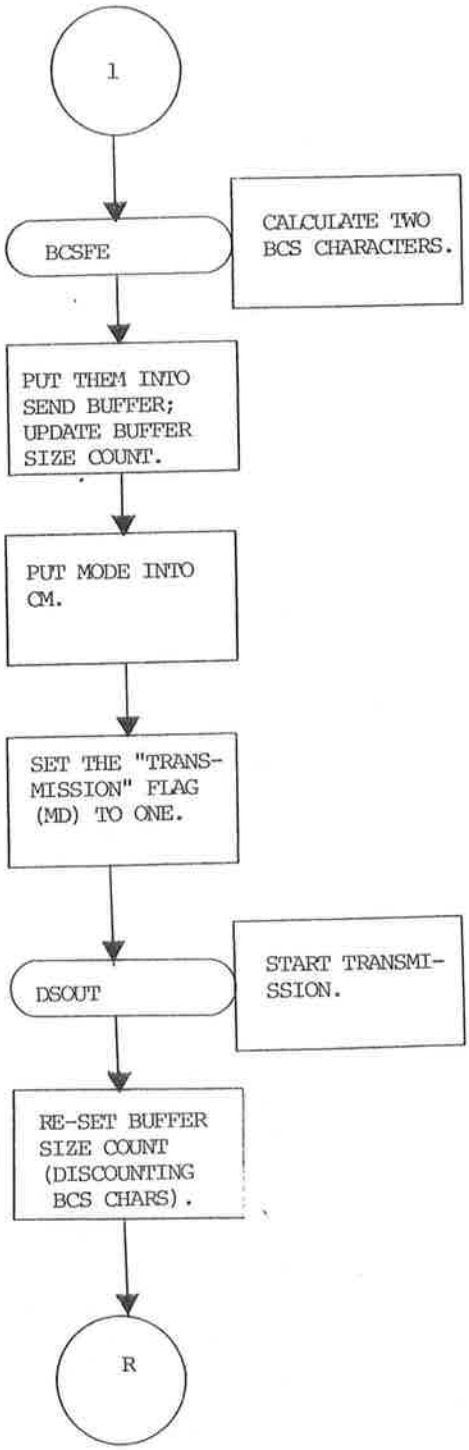
all zeroes, all of the logic flags and switches are reset (set to zero), the turn-around timer TT is initialized and the turn-around clock is started.

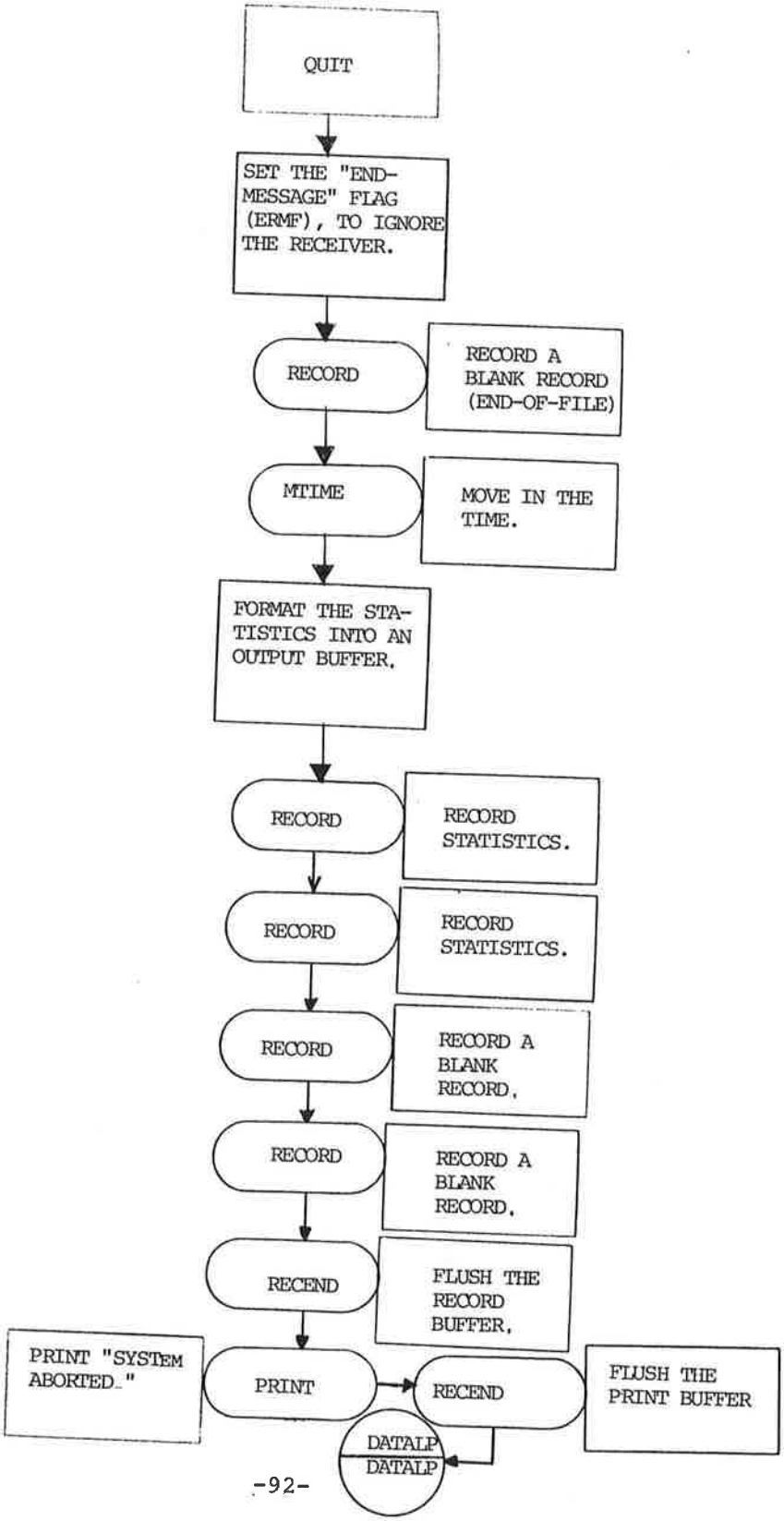
Subroutine MSGREC is called from the modem interrupt handler. This is the logic which is used to compile a message in core memory, as it is received. When the interrupt handler recognizes a modem interrupt, MSGREC is called at which time it will simply release control if a message is not expected. This type of logic is sufficient for the reason that the system is designed and implemented as a half-duplex communications handler. If modem interrupts are acceptable, the characters are accepted as forming a message if, and only if, each of the first six characters is exactly what it should be. This is a selective filtering technique which was constructed to improve communications accuracy. Beyond the first six characters, all characters are accepted and passed through the BCS calculation until a "natural" termination is detected. For transparent mode messages, the text terminates when a counter (initialized by the character following the STX character) expires. For normal mode messages the message terminates with either an end-of-text (ETX) or an end-of-text-block (ETB) character.

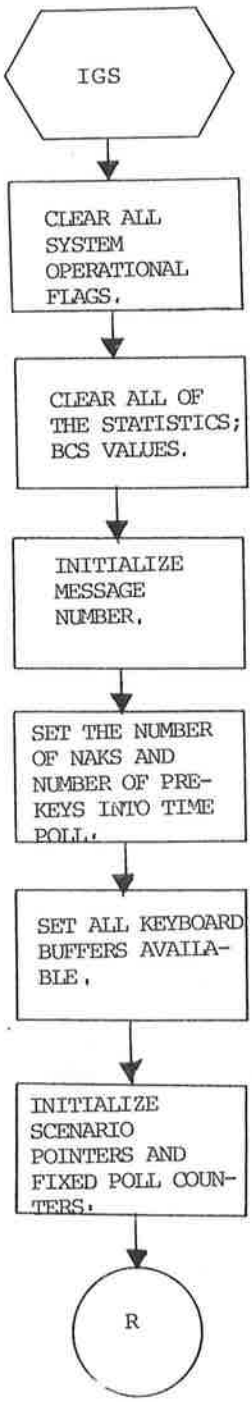
After the message is completely compiled by the message reception logic (MSGREC), it is checked for all combinations of BCS error, improper aircraft identifier, and incorrect message length (long or short), and the correct respective flags are set. All of this

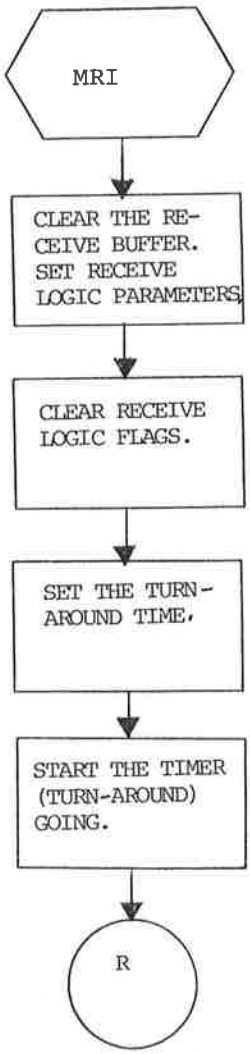
checking is prompted by the interrupt accompanying the final BCS character. The final step is to disable the modem receiver and to set lockout flags for honoring the interrupts from the receiver.

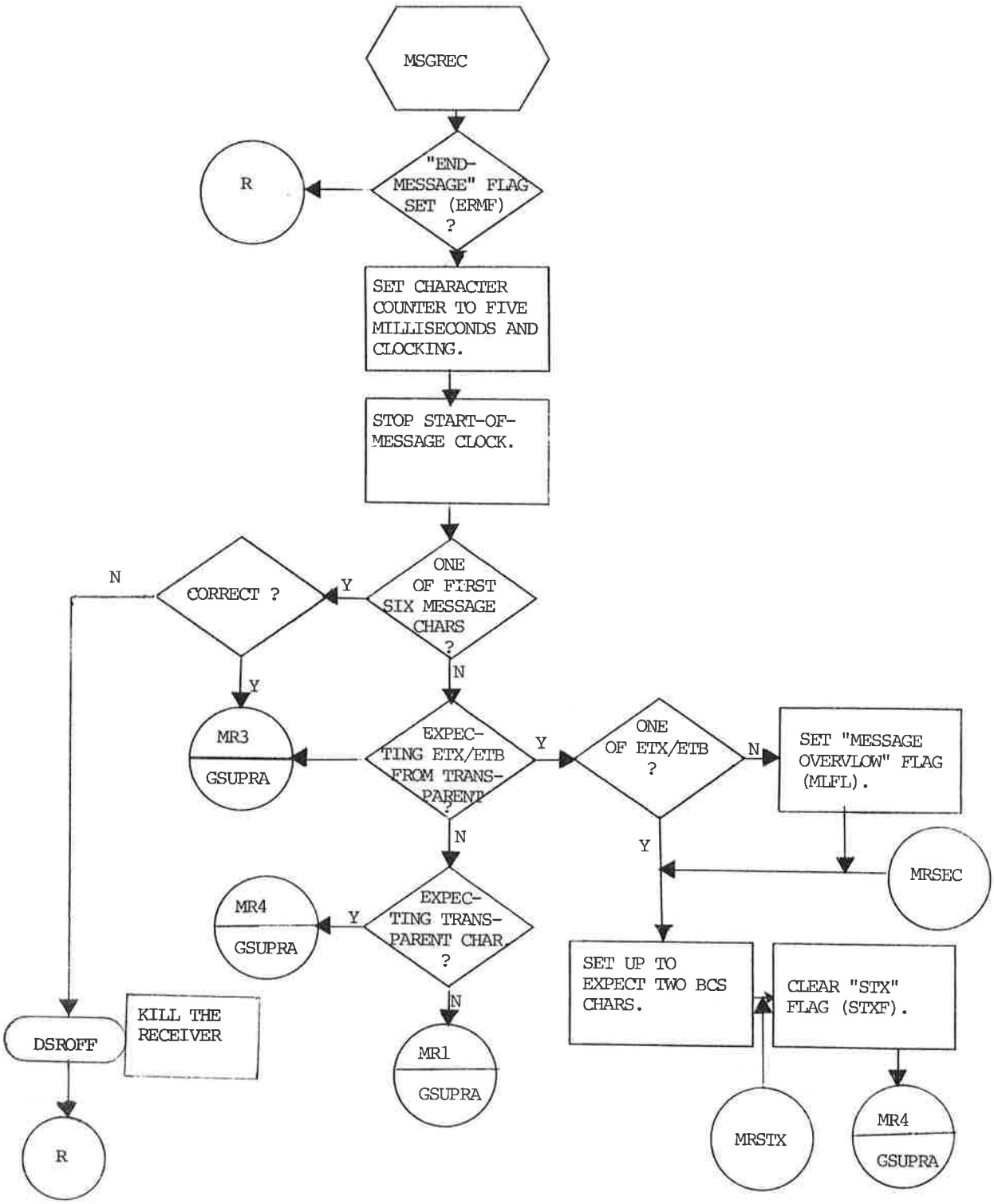

SENDM
PUT ACK OR NAK IN SEND BUFFER, BASED ON FAN FLAG.
SET EFT FLAG, BASED ON MODE TO BE SENT.
SET TEXT COUNT INTO LSTCC.
SET UP INDEXER, BASED ON SEND BUFFER LENGTH.
BCSIE
INITIALIZE THE BCS CALCULATION.
FETCH SEND CHARACTER (STARTING WITH MODE).
BCSN
ROTATE THRU BCS CALCULATION.
ETX/ ETB ?
N
Y
1
GSUPRA

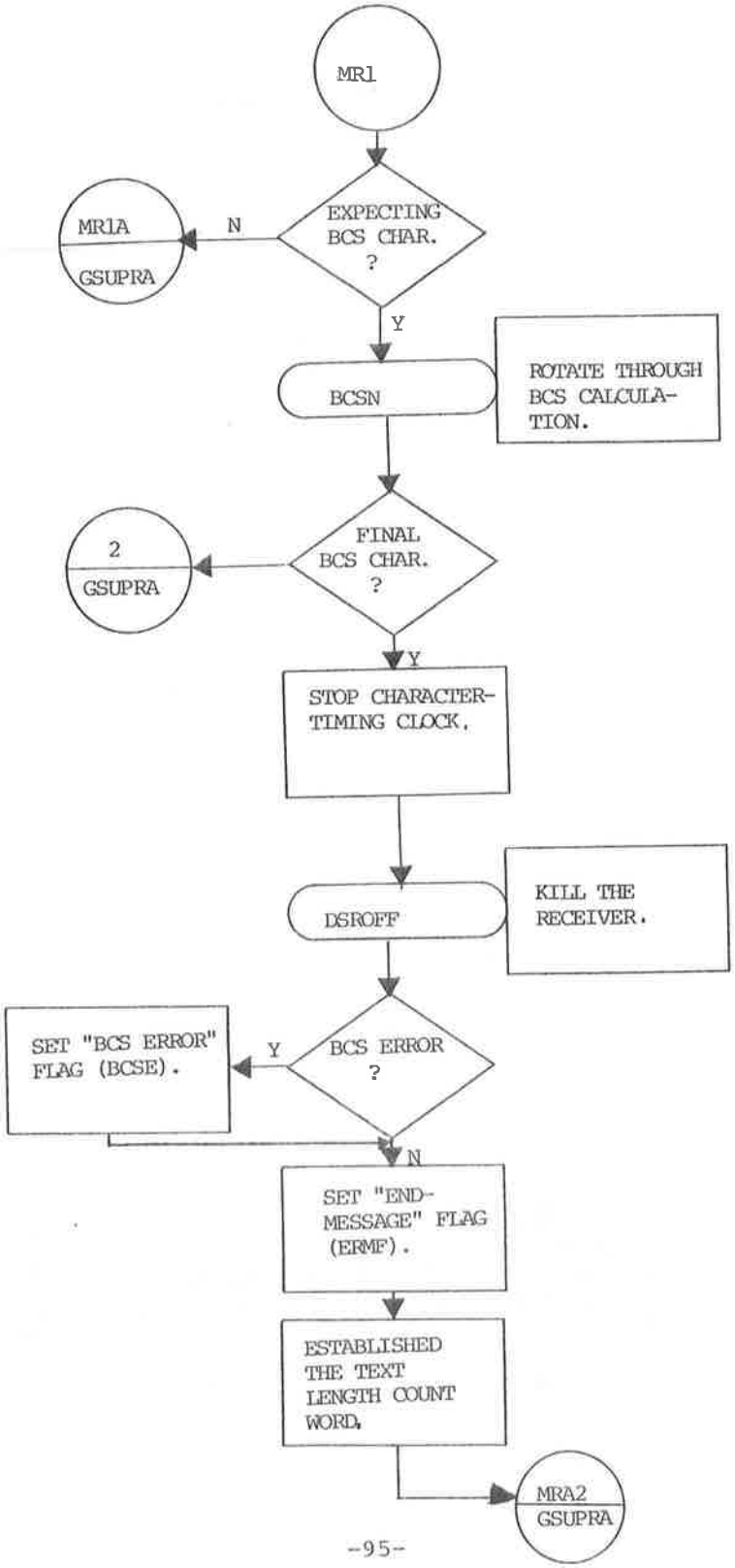

1
BCSFE
CALCULATE TWO BCS CHARACTERS.
PUT THEM INTO SEND BUFFER; UPDATE BUFFER SIZE COUNT.
PUT MODE INTO CM.
SET THE "TRANS-MISSION" FLAG (MD) TO ONE.
DSOUT
START TRANSMI-SSION.
RE-SET BUFFER SIZE COUNT (DISCOUNTING BCS CHARS).
R

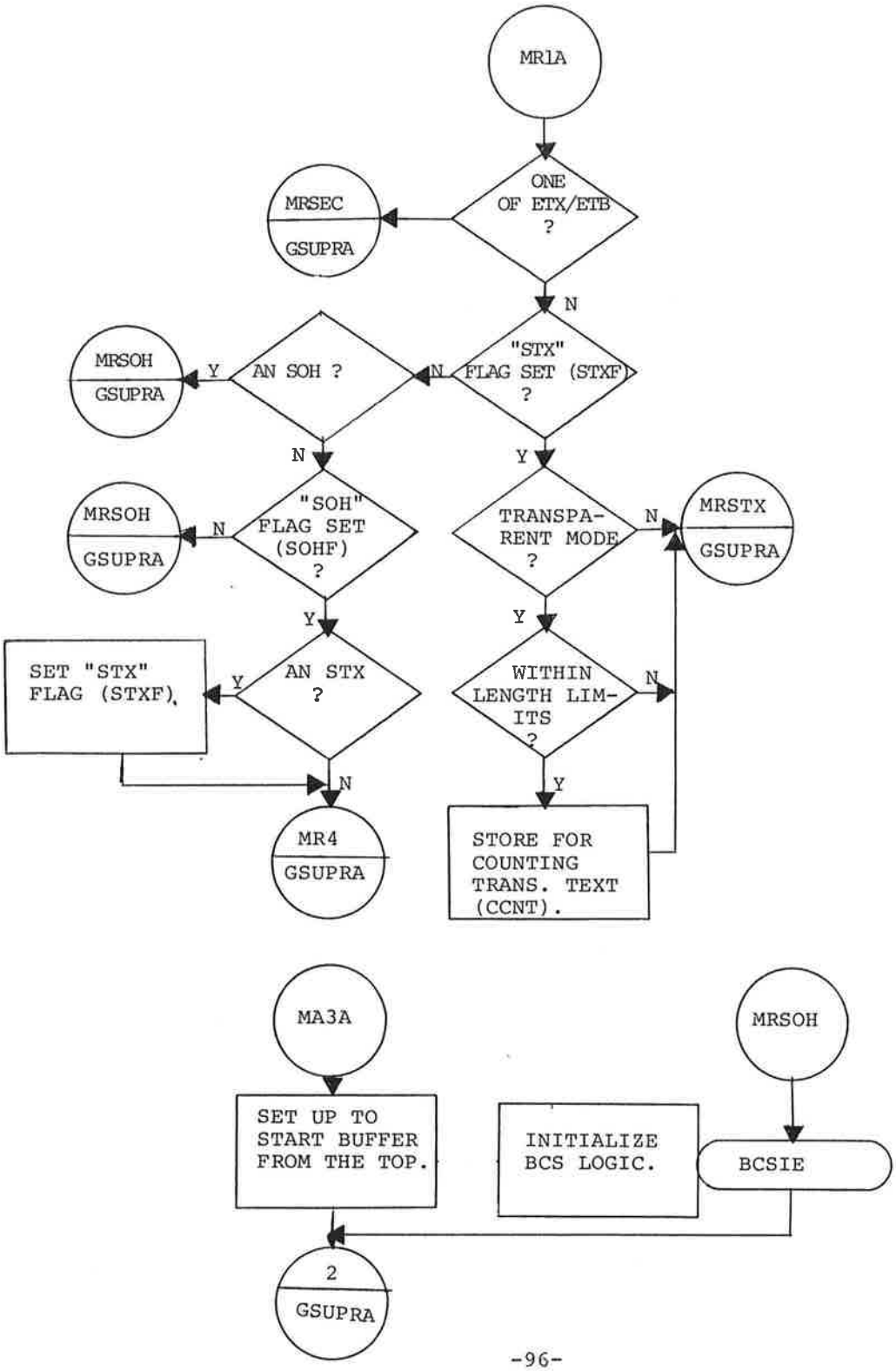

QUIT
SET THE "END-MESSAGE" FLAG (ERMF), TO IGNORE THE RECEIVER.
RECORD
RECORD A BLANK RECORD (END-OF-FILE)
MTIME
MOVE IN THE TIME.
FORMAT THE STATISTICS INTO AN OUTPUT BUFFER.
RECORD
RECORD STATISTICS.
RECORD
RECORD STATISTICS.
RECORD
RECORD A BLANK RECORD.
RECORD
RECORD A BLANK RECORD.
RECEND
FLUSH THE RECORD BUFFER.
PRINT "SYSTEM ABORTED."
PRINT
RECEND
FLUSH THE PRINT BUFFER
DATALP
DATALP

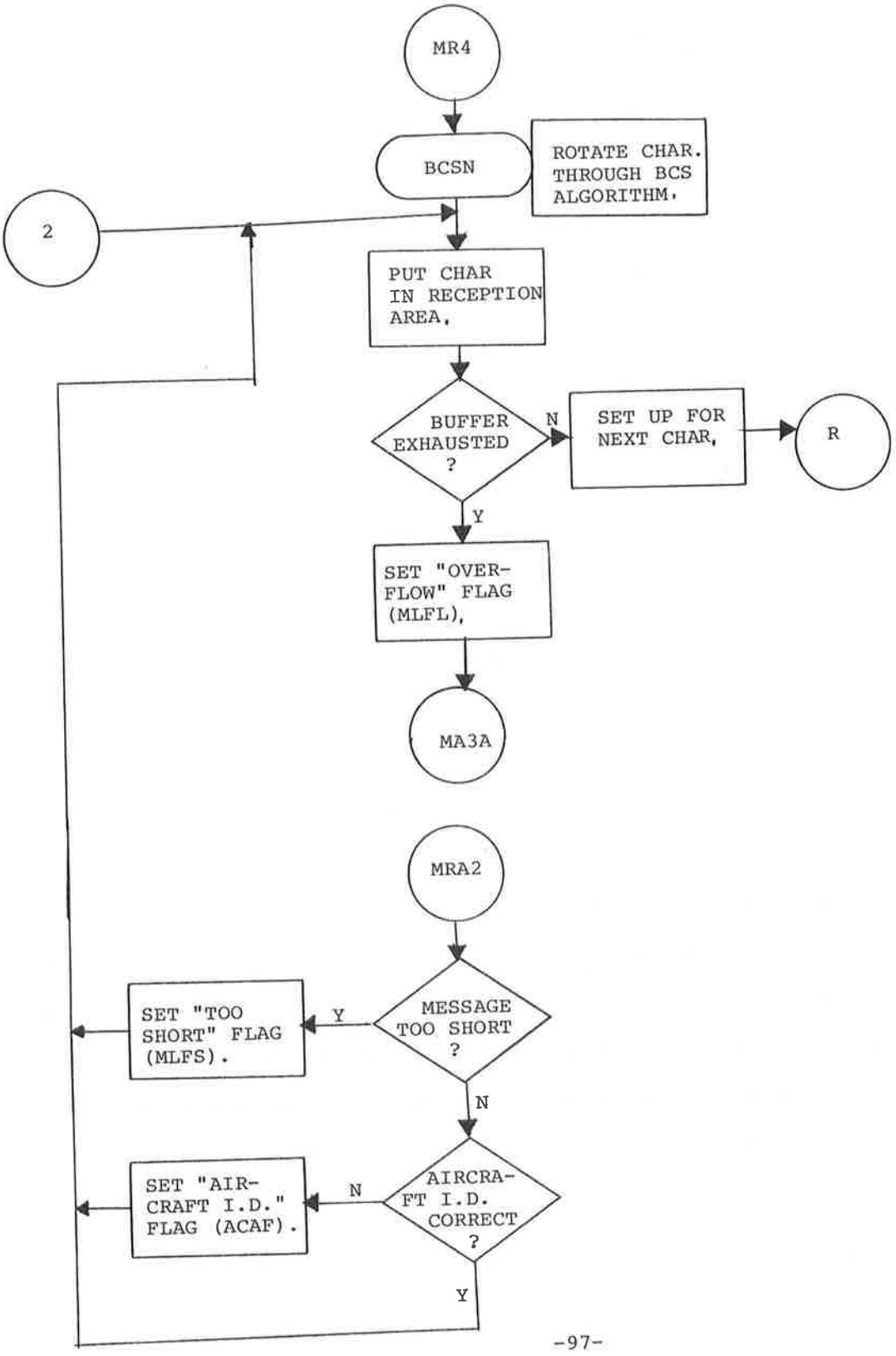

IGS
CLEAR ALL SYSTEM OPERATIONAL FLAGS.
CLEAR ALL OF THE STATISTICS; BCS VALUES.
INITIALIZE MESSAGE NUMBER.
SET THE NUMBER OF NAKS AND NUMBER OF PRE-KEYS INTO TIME POLL.
SET ALL KEYBOARD BUFFERS AVAILA-BLE.
INITIALIZE SCENARIO POINTERS AND FIXED POLL COUN-TERS.
R


MRI
CLEAR THE RE-CEIVE BUFFER. SET RECEIVE LOGIC PARAMETERS.
CLEAR RECEIVE LOGIC FLAGS.
SET THE TURN-AROUND TIME.
START THE TIMER (TURN-AROUND) GOING.
R


MSGREC
"END-MESSAGE" FLAG SET (ERMF) ?
R
SET CHARACTER COUNTER TO FIVE MILLISECONDS AND CLOCKING.
STOP START-OF-MESSAGE CLOCK.
ONE OF FIRST SIX MESSAGE CHARS ?
Y
N
CORRECT ?
N
Y
MR3
GSUPRA
EXPEC-TING ETX/ETB FROM TRANS-PARENT ?
Y
N
ONE OF ETX/ETB ?
N
SET "MESSAGE OVERVLOW" FLAG (MLFL).
Y
MRSEC
MR4
GSUPRA
EXPEC-TING TRANS-PARENT CHAR. ?
Y
N
SET UP TO EXPECT TWO BCS CHARS.
CLEAR "STX" FLAG (STXF).
MR1
GSUPRA
MRSTX
MR4
GSUPRA
DSROFF
KILL THE RECEIVER
R


MR1
EXPECTING BCS CHAR. ?
N
MR1A
GSUPRA
Y
BCSN
ROTATE THROUGH BCS CALCULA-TION.
FINAL BCS CHAR. ?
2
GSUPRA
Y
STOP CHARACTER-TIMING CLOCK.
DSROFF
KILL THE RECEIVER.
BCS ERROR ?
Y
SET "BCS ERROR" FLAG (BCSE).
N
SET "END-MESSAGE" FLAG (ERMF).
ESTABLISHED THE TEXT LENGTH COUNT WORD.
MRA2
GSUPRA


MR1A
ONE OF ETX/ETB ?
MRSEC GSUPRA
N
"STX" FLAG SET (STXF) ?
AN SOH ?
MRSOH GSUPRA
Y
N
N
Y
"SOH" FLAG SET (SOHF) ?
MRSOH GSUPRA
N
TRANSPA-RENT MODE ?
N
MRSTX GSUPRA
Y
Y
AN STX ?
SET "STX" FLAG (STXF).
Y
WITHIN LENGTH LIM-ITS ?
N
N
Y
MR4 GSUPRA
STORE FOR COUNTING TRANS. TEXT (CCNT).
MA3A
MRSOH
SET UP TO START BUFFER FROM THE TOP.
INITIALIZE BCS LOGIC.
BCSIE
2 GSUPRA


MR4
BCSN
ROTATE CHAR. THROUGH BCS ALGORITHM.
2
PUT CHAR IN RECEPTION AREA.
BUFFER EXHAUSTED ?
N
SET UP FOR NEXT CHAR.
R
Y
SET "OVER-FLOW" FLAG (MLFL).
MA3A
MRA2
MESSAGE TOO SHORT ?
Y
SET "TOO SHORT" FLAG (MLFS).
N
AIRCRA-FT I.D. CORRECT ?
N
SET "AIR-CRAFT I.D." FLAG (ACAF).
Y

C. GU

This module is comprised of utility-type subroutines designed to enable the ground supervisor to be designed as logically functional as possible. There are also some subroutines which are called on the interrupt level. The subroutines are described below in the order in which they occur in memory.

RTCIH

Each real-time clock (interval-timer) interrupt this subroutine is called from the system interrupt handler. The software locations which track the time of data are incremented by one millisecond. The clock is maintained as a twenty-four hour clock. There is no date maintained; hence, the clock simply cycles from 23:59:59.999 to midnight (00:00:00.000) with no indication of a new day. After the time of day has been accounted for, flag TC is interrogated. If it is a one, the system is timing character spacing and location CHART is tallied by minus one. If the clock has decremented to zero, the flag TCO is set to one, to indicate that the clock has overflowed. Next, flag TSOM is interrogated. If it is a one, the system is clocking the delay until a start-of-message is encountered. Therefore, location SOMTR is decremented. If it has overflowed, flag TSOMO is set to one to show that the clock has expired. After the two timers are checked and updated, RTCIH returns control to its caller.

BCSIE-BCSN-BCSFE

This subroutine has three entry points. BCSIE is called as an initialization procedure and BCSFE is called as finalization procedure. The purpose of calculating the two BCS characters are to lend confidence to the communications. The calculation of the block check sequence (BCS) characters is performed on both transmission and reception. The two BCS characters calculated during the transmission are rotated through the BCS algorithm during reception. If all calculation and transmission and reception are error-free, the two BCS characters which were sent will, when rotated through the BCS algorithm, result in two zero characters at the receiving station.

When an SOH (start-of-header) character is sent/received, entry BCSIE is called. The net effect is simply to clear the lower sixteen bits (word) of a thirty-two bit (double-word) value.

Every character following the SOH character causes an entry to BCSN, where the following takes place. The character is placed in the word representing the high order of the double-word value. The upper half of this upper word is cleared. The entire double-word is logically rotated one position to the right. If the upper word is now negative (i.e. the sign-bit is a one), the lower word is exclusive-or'ed with 8408_{16}. The double word rotation and resultant or'ing is performed eight times. After the eighth iteration the subroutine releases control to the caller.

As each character causes the algorithm iteration, the completion of the last character causes the system to request an entry to BCSFE where the routine causing the rotation and exclusive or'ing is performed sixteen times. At the completion of the sixteenth iteration, the lower half of the lower-word becomes the first BCS character and the upper half of the lower-word becomes the second BCS character. They become the two BCS characters which are appended onto every transmission and which should cause the receiving station to end-up with two zero BCS characters.

GETAB

This subroutine is called from the interrupt level. It is used to assign a buffer from a core pool of three buffers for use by the system to assemble a keyed-in message (by the system operator). The subroutine is not re-entrant; hence, if busy, it simply exits. If at least one of the three buffer areas is available, the pointer is passed to the caller. If none is available, a default return is performed.

GMOV

This subroutine is called to move a block of words from one section of memory to another section of memory. The two areas' pointers are pre-set-up and the count is also pre-set-up. The two areas must not overlap in the manner that the "receiving" area is within the number of move-words higher in memory (higher, as toward location zero-in a direction of smaller addresses).

MTIME

This subroutine uses GMOV to move the current time to any specified nine consecutive memory locations. During subroutine execution, the clock is instantaneously precluded from interrupting.

MMID

Using subroutine GMOV, this subroutine moves the current message identifier to any specified five consecutive memory locations.

IMNM-IMNMI

This is a double entry subroutine. Entry IMNMI is used to initialize a record area with the current time (using MTIME) and to force the alpha character of a message identifier to a "blank". The subroutine then continues at entry IMNM, where the alpha character of the message identifier is incremented in the manner that "blank" goes to A, A goes to B, B goes to C, . . ., Y goes to Z, Z goes to A, etc. The numeric value of the identifier is incremented bv one. The identifier is then moved (using MMID) to a message area which is specified as a calling parameter.

UM

This subroutine moves the scenario message address block specified by location CSB from the packed scenario area CSCEN to the unpacked scenario message area USCEN.

SMSTEP

This subroutine is called each time the ground station determines the need to advance to another message. Counter EMSGR is decremented to determine whether the repeat counter is exhausted. If it is not, the subroutine merely releases control to the caller. If it is, the next scenario message is requested and moved using subroutine UM, all of the message parametric controllers are re-stored to "full" condition, and the subroutine releases control.

RIM-RCM

This double entry subroutine records message transaction records. If the transaction to be recorded is not a "good" transaction, entry RIM is used, with the error code in previously set; else entry RCM is used. An output memory buffer area is compiled. The record code is stored, followed by the message identifier (using subroutine MMID). The current time is put behind the previously stored start time, using subroutine MTIME. The current "sent" text character count is encoded and stored. The current mode is stored. The current scenario message number is encoded and stored, unless the message originated external to the scenario (system entry, time poll, keyboard message, or general poll), in which case a "K" is stored.

The output area is recorded using subroutine RECORD of the I/O level. The message controlling parameters are re-stored to the "full" condition, and control is released to the caller.

EREC

This subroutine is called whenever the ground system supervisor determines that it is required to file an error record. The conditions causing an error record to be filed are enumerated in Appendix G and the record formats are displayed in Appendices A,E, and F. All non-responses to a poll result in the recording of the originating poll. All other recordings are of the received poll on non-echo type transactions, or of the text characters not correctly echoed.

EXCOM

Because of the cassettes being used to accumulate data, there are some characters which are incapable of being recorded (they cause the cassette controller to perform undesirable functions). This subroutine is entered with a character to be recorded. If the character is within the limits 10_{16} - C - 14_{16}, 51_{16} is added to the character. This effectively turns the excluded characters into the set of lower case A through lower case E. If the character is not in the given set, it is checked for the occurrance of a bit-set in the highest position (8th). Since the cassette cannot record eight bits of information, any character with track eight set to one is changed to $7E_{16}$, and is, therefore, recorded as the diacritical mark . If the character is not in either of the given sets, it is allowed to pass through EXCOM unchanged. Only characters which are supposed to be plain text (non-transparent) are passed through EXCOM.

MVCMP

This subroutine is entered with a count of the number of characters to be moved from one buffer to another buffer by passing them through EXCOM. Subroutines GET and PUT are also employed.

GET

This subroutine fetches a character pointed to by pointer GETP. It then advances GETP and increments GETC, which serves as a running total of "fetched" characters.

PUT

This subroutine stores a character in the location pointed to by PUTP. It then advances PUTP and increments PUTC, which serves as a running store counter. This is the complimentary subroutine to GET.

ECPUT

This subroutine is called for all characters which are to be encoded. Transparent and binary information are passed through ECPUT. The eight calling bits are divided into two four-bit bytes. Each byte is added to 30_{16} to produce a record-able character. If the calling character is a minus one (FFFF) the effect of this subroutine is to produce two $2E_{16}$ (period) characters. This allows successfully echoed characters to be shown as a double period and the non-successfully echoed characters to be recorded "as seen" (after encoding).

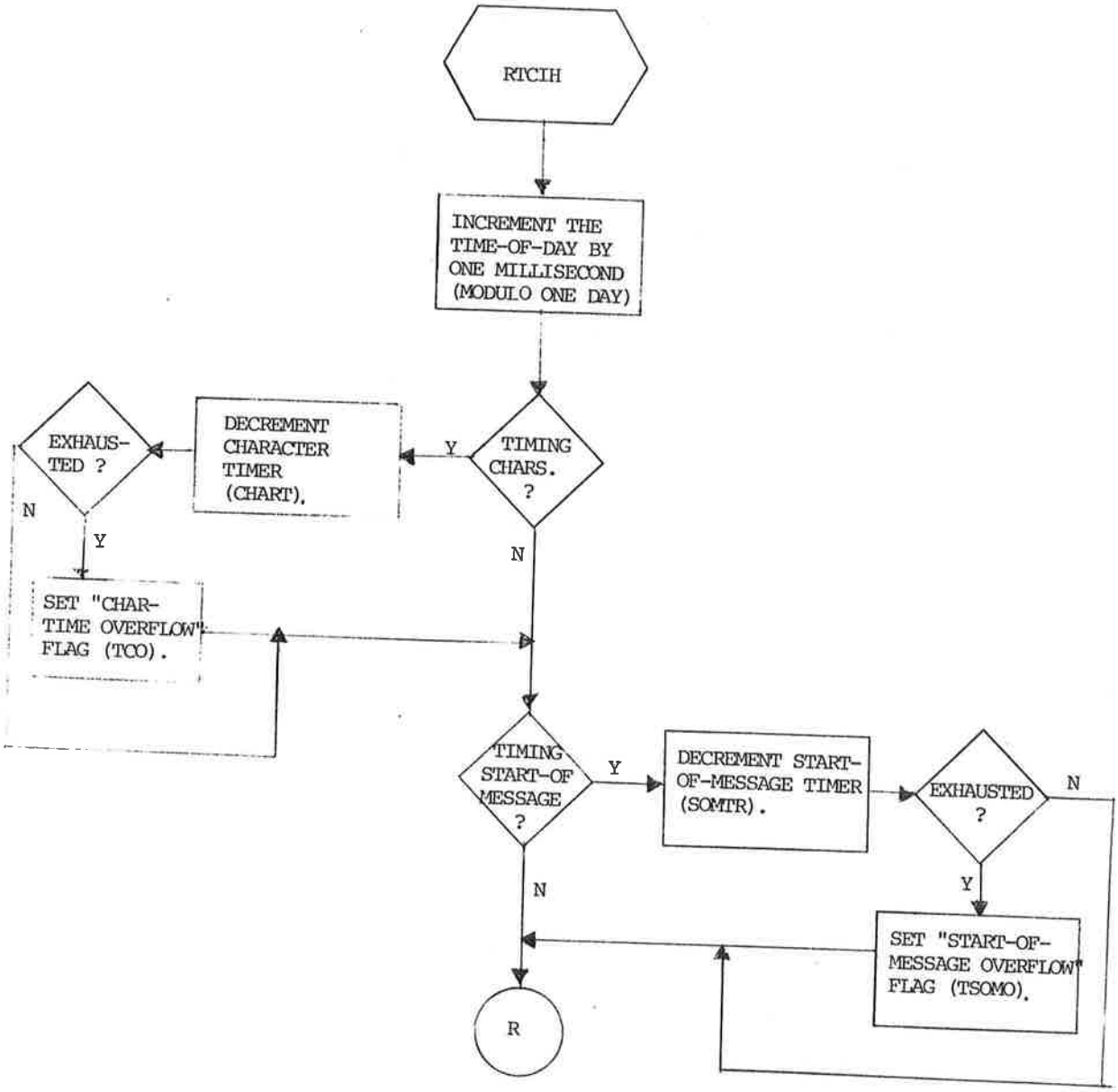
TSUBW

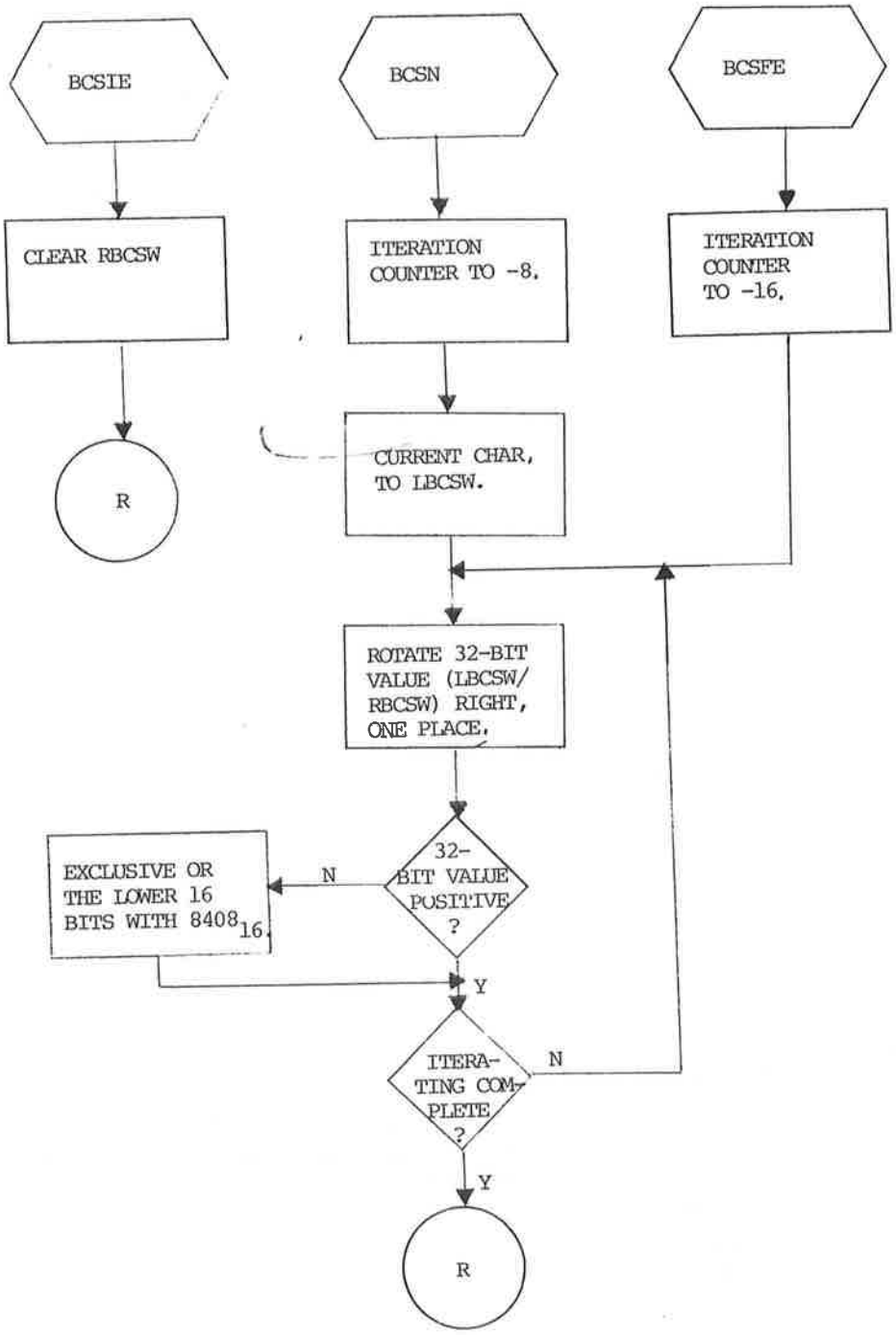
This subroutine is used to compare the text area of a received message with the text portion of the originating message (echo-mode messages). Subroutine FCLASS is called to establish the length classification of the originating message. Each character of both texts are sequentially exclusive or'ed with its corresponding character. A zero result implies a complete bit-by-bit agreement. A non-zero result causes the accumulation of the one bits in the or result. This tally is double precision added to the data base area ECHOBE, which stores the running count of bits in error. If there is no bit error in a character position, it is replaced with a negative one, which is used as a "no-error" sentinel by subroutine ECPUT.

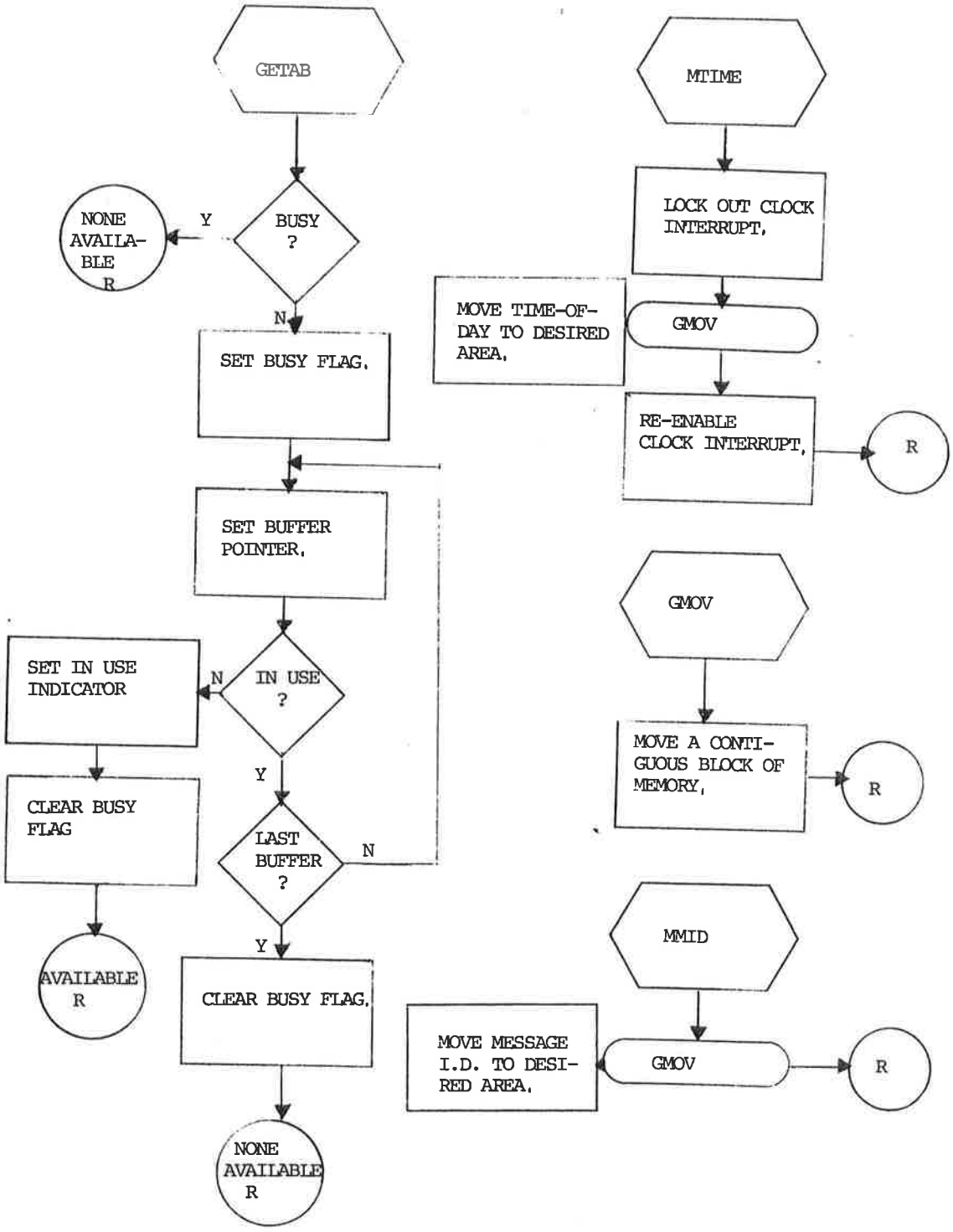
FCLASS

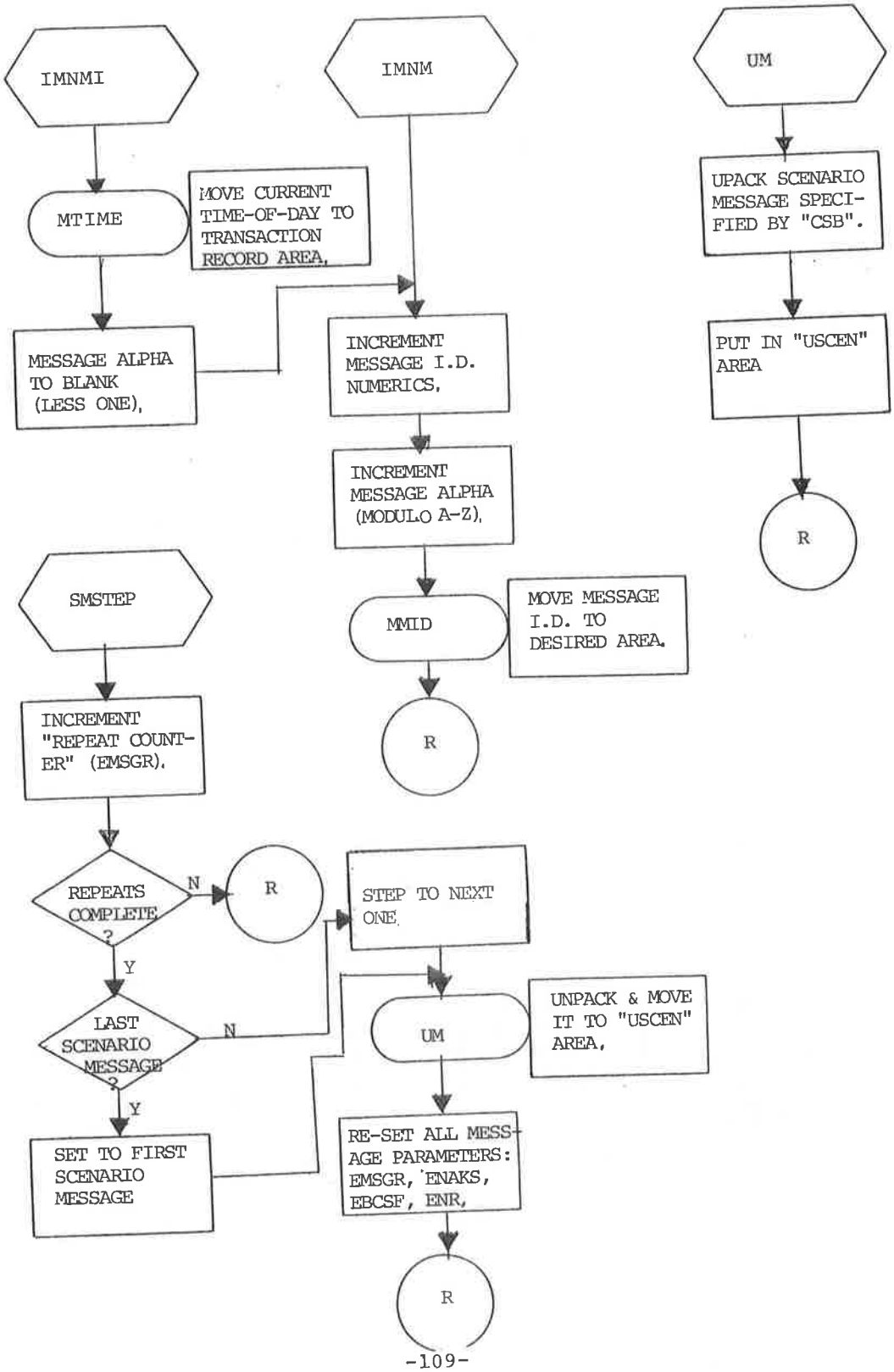
For statistical purposes, this subroutine determines the classification of a message by the following table:

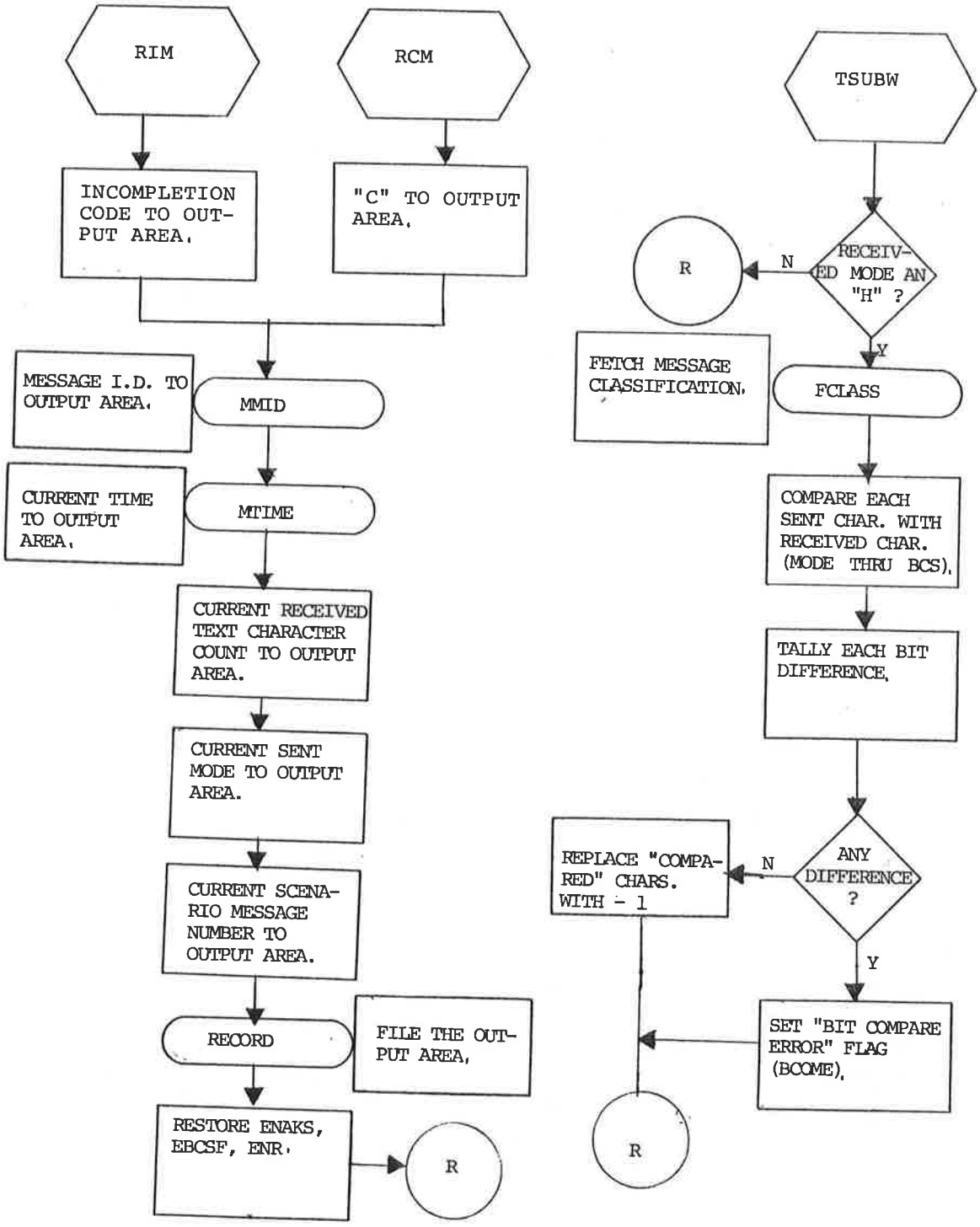
Class One:	0 or 1 text character
Class Two:	2 to 30 text characters, inclusive
Class Three:	31 to 120 text characters, inclusive
Class Four:	in excess of 120 text characters

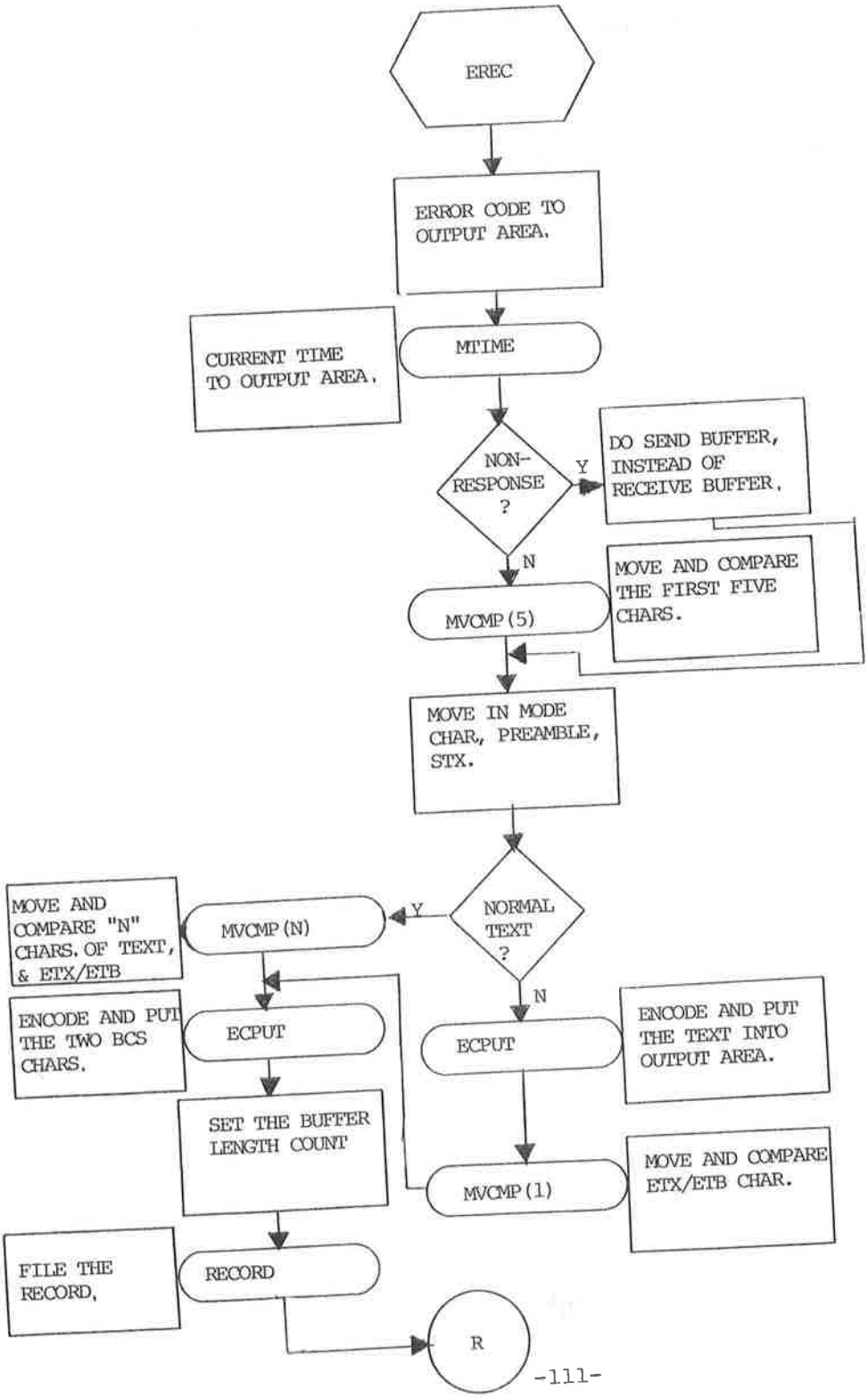

RTCIH
INCREMENT THE TIME-OF-DAY BY ONE MILLISECOND (MODULO ONE DAY)
TIMING CHARS. ?
Y
DECREMENT CHARACTER TIMER (CHART).
EXHAUS-TED ?
N
Y
SET "CHAR-TIME OVERFLOW" FLAG (TCO).
N
TIMING START-OF MESSAGE ?
Y
DECREMENT START-OF-MESSAGE TIMER (SOMTR).
EXHAUSTED ?
N
Y
SET "START-OF-MESSAGE OVERFLOW" FLAG (TSOMO).
N
R

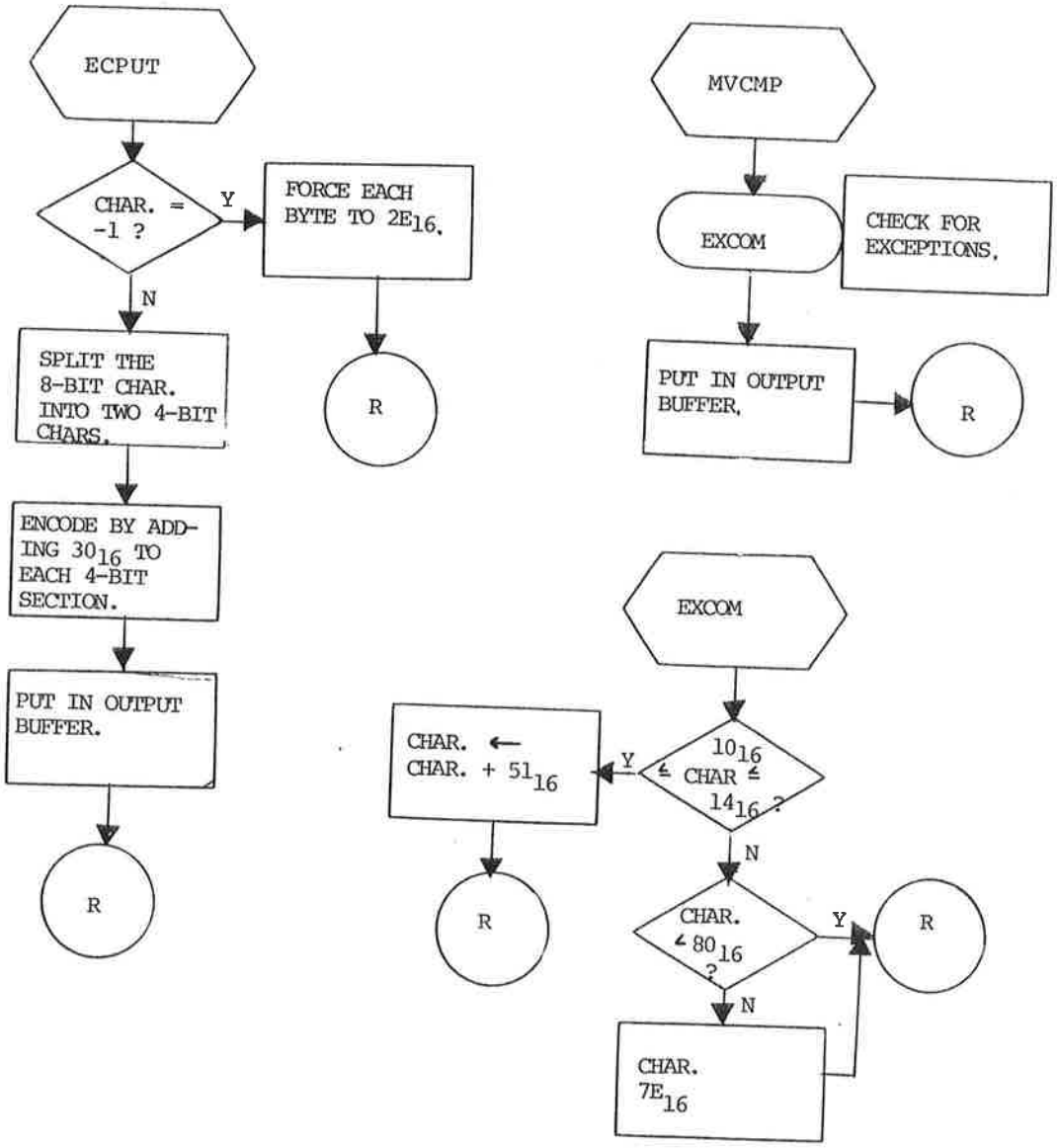

BCSIE
BCSN
BCSFE
CLEAR RBCSW
ITERATION COUNTER TO -8.
ITERATION COUNTER TO -16.
R
CURRENT CHAR, TO LBCSW.
ROTATE 32-BIT VALUE (LBCSW/ RBCSW) RIGHT, ONE PLACE.
32-BIT VALUE POSITIVE ?
N
EXCLUSIVE OR THE LOWER 16 BITS WITH 8408 16.
Y
ITERA-TING COM-PLETE ?
N
Y
R


GETAB
BUSY ?
Y
NONE AVAILA-BLE R
N
SET BUSY FLAG.
SET BUFFER POINTER.
IN USE ?
N
SET IN USE INDICATOR
Y
CLEAR BUSY FLAG
LAST BUFFER ?
N
Y
AVAILABLE R
CLEAR BUSY FLAG.
NONE AVAILABLE R
MTIME
LOCK OUT CLOCK INTERRUPT.
MOVE TIME-OF-DAY TO DESIRED AREA.
GMOV
RE-ENABLE CLOCK INTERRUPT.
R
GMOV
MOVE A CONTI-GUOUS BLOCK OF MEMORY.
R
MMID
MOVE MESSAGE I.D. TO DESI-RED AREA.
GMOV
R


IMNMI
MTIME
MOVE CURRENT TIME-OF-DAY TO TRANSACTION RECORD AREA.
MESSAGE ALPHA TO BLANK (LESS ONE).
IMNM
INCREMENT MESSAGE I.D. NUMERICS.
INCREMENT MESSAGE ALPHA (MODULO A-Z).
MMID
MOVE MESSAGE I.D. TO DESIRED AREA.
R
UM
UPACK SCENARIO MESSAGE SPECIFIED BY "CSB".
PUT IN "USCEN" AREA
R
SMSTEP
INCREMENT "REPEAT COUNTER" (EMSGR).
REPEATS COMPLETE ?
N
R
Y
STEP TO NEXT ONE.
LAST SCENARIO MESSAGE ?
N
Y
SET TO FIRST SCENARIO MESSAGE
UM
UNPACK & MOVE IT TO "USCEN" AREA.
RE-SET ALL MESSAGE PARAMETERS: EMSGR, ENAKS, EBCSF, ENR.
R


RIM
RCM
INCOMPLETION CODE TO OUTPUT AREA.
"C" TO OUTPUT AREA.
MESSAGE I.D. TO OUTPUT AREA.
MMID
CURRENT TIME TO OUTPUT AREA.
MTIME
CURRENT RECEIVED TEXT CHARACTER COUNT TO OUTPUT AREA.
CURRENT SENT MODE TO OUTPUT AREA.
CURRENT SCENARIO MESSAGE NUMBER TO OUTPUT AREA.
RECORD
FILE THE OUTPUT AREA.
RESTORE ENAKS, EBCSF, ENR.
R
TSUBW
RECEIVED MODE AN "H" ?
N
R
Y
FETCH MESSAGE CLASSIFICATION.
FCLASS
COMPARE EACH SENT CHAR. WITH RECEIVED CHAR. (MODE THRU BCS).
TALLY EACH BIT DIFFERENCE.
ANY DIFFERENCE ?
N
REPLACE "COMPARED" CHARS. WITH - 1
Y
SET "BIT COMPARE ERROR" FLAG (BCOME).
R


EREC
ERROR CODE TO OUTPUT AREA.
CURRENT TIME TO OUTPUT AREA.
MTIME
NON-RESPONSE ?
Y
DO SEND BUFFER, INSTEAD OF RECEIVE BUFFER.
N
MVCMP(5)
MOVE AND COMPARE THE FIRST FIVE CHARS.
MOVE IN MODE CHAR., PREAMBLE, STX.
NORMAL TEXT ?
Y
MVCMP(N)
MOVE AND COMPARE "N" CHARS. OF TEXT, & ETX/ETB
N
ECPUT
ENCODE AND PUT THE TWO BCS CHARS.
ECPUT
ENCODE AND PUT THE TEXT INTO OUTPUT AREA.
SET THE BUFFER LENGTH COUNT
MVCMP(1)
MOVE AND COMPARE ETX/ETB CHAR.
FILE THE RECORD.
RECORD
R


ECPUT
CHAR. = -1 ?
Y
FORCE EACH BYTE TO $2E_{16}$.
N
SPLIT THE 8-BIT CHAR. INTO TWO 4-BIT CHARS.
R
ENCODE BY ADDING 30_{16} TO EACH 4-BIT SECTION.
PUT IN OUTPUT BUFFER.
R
MVCMP
EXCOM
CHECK FOR EXCEPTIONS.
PUT IN OUTPUT BUFFER.
R
EXCOM
CHAR. ← CHAR. + 51_{16}
Y
$10_{16} \leq$ CHAR $\leq 14_{16}$?
R
N
CHAR. $< 80_{16}$?
Y
R
N
CHAR. $7E_{16}$

D. DLFS-DLDS

This module contains all of the flags, tables, variables, constants, and pointers utilized by the ground station.

The classes of locations are defined by: system constant (K), experiment constant (C), experiment variable (V), buffer (B), and pointer (P).

FLAG	STATE	IMPLICATION
ACAF	0	Aircraft I.D. in received message is correct.
	1	Aircraft I.D. in received message is not correct.
BAUDC	0	System is operating at 2400 baud.
	1	System is operating at 4800 baud.
BCOME	0	Echoed response has no bit compare error(s).
	1	Echoed response has at least one bit compare error.
BCSE	0	Received message has no BCS, length, or I.D. error.
	1	Received message has at least one of above errors.
EFT	0	Used as an echo/non-echo switch-non-echo.
	1	Used as an echo/non-echo switch-echo.
ERMF	0	End-of-message not seen (or, accept modem char.).
	1	End-of-message seen (or, don't accept modem char.).
ERS	0	Record receive buffer.
	1	Record send buffer.
FAN	0	Send an 'ACK'.
	1	Send a 'NAK'.
GETABF	0	'GETAB' subroutine not busy.
	1	'GETAB' subroutine busy.
GPF	0	General poll not required.
	1	Send a general poll.
ICF	0	System entry poll not acknowledged.
	1	System entry poll acknowledged.

FLAG	STATE	IMPLICATION
KBDM	0	No keyboard buffer needs attention.
	1	A keyboard buffer needs attention.
KBDMIP	0	No keyboard buffer is in transmission.
	1	A keyboard buffer is in transmission.
MD	0	Not transmitting.
	1	Transmitting.
MLFL	0	Received message not too long.
	1	Received message too long.
MLFS	0	Received message not too short.
	1	Received message too short.
NAKE	0	'ACK' seen.
	1	'ACK' not seen.
QF	0	Continue.
	1	Stop.
SOHF	0	This received char. is an 'SOH'.
	1	This received char. is not an 'SOH'.
STXF	0	An 'STX' not seen.
	1	An 'STX' seen.
TC	0	Do not clock characters.
	1	Clock characters.

FLAG	STATE	IMPLICATION
TCO	0	Character clock didn't overflow.
	1	Character clock did overflow.
TF	0	Timing poll not required.
	1	Timing poll required.
TSOM	0	Do not clock start-of-message.
	1	Clock start-of-message.
TSOMO	0	Start-of-message clock didn't overflow.
	1	Start-of-message clock did overflow.

LOCATION	CLASS	USE
BCSCNT	V	Iteration counter for BCS calculation.
BCSEEN	V	Calculated BCS of received message.
CB	P	Points to current transmission buffer.
CCNT	V	Temporary use by MSGREC.
CHART	V	Character timer clock.
CKBB	P	Contains the current keyboard transmission buffer.
CLASSC	V	Classification code.
CM	V	Last sent mode.
CRTCC	V	Current received text character count.
CSB	P	Current send buffer pointer.
CSCEN	B	First word of the 16 packed scenarios.
CTIME	B	Time of day portion of record buffer.
DLGUE	B	Dialogue character input buffer.
EBCSF	V	Number of BCS error--this message.
ECHOAK	V-B	Number of acknowledgements of echo messages.
ECHOBC	V-B	Number of BCS failures of echo messages.
ECHOBE	V-B	Number of bit errors.

LOCATION	CLASS	USE
ECHONK	V-B	Number of non-acknowledgements of echo messages.
ECHONR	V-B	Number of non-responses of echo messages.
ECHOTC	V-B	Total characters sent of echo messages.
ECNT	V	Temporary use by MSGREC.
EMSGR	V	Countdown of transmission repeats.
ENAKS	V	Number of non-acknowledgements--this message.
ENR	V	Number of non-responses-this message.
ERBUF	V	Recording buffer.
GP	B	General poll.
KBCSF	C	Keyed-in number of BCS errors.
KMSGR	C	Keyed-in number of message repeats.
KNAKS	C	Keyed-in number of non-acknowledgements.
KNPK	C	Keyed-in number of prekeys.
KNR	C	Keyed-in number of non-responses.
LBCSW	V	BCS calculation variable.
LSTCC	V	Last sent text character count.
MANB	K-P	Pointer table to MAN1, MAN2, MAN3.
MAN1	B	Keyboard buffer.

LOCATION	CLASS	USE
TODHRB	V-B	L.S. hours.
TODMNA	V-B	M.S. minutes.
TODNMB	V-B	L.S. minutes.
TODMSA	V-B	M.S. milliseconds.
TODMSB	V-B	Middle milliseconds.
TODMSC	V-B	L.S. milliseconds.
TODSCA	V-B	M.S. seconds.
TODSCB	V-B	L.S. seconds.
TT	K	Message turn-around time.
USCEN	B	Unpacked scenario area.

V. AIRBORNE STATION COMMUNICATIONS SUPERVISOR

This section describes the function and operation of the supervisory logic of the airborne station.

A. ASUPR

The airborne supervisor module controls the logic sequence of the in-flight system. There is no dialogue because the airborne system receives its few externally required parameters from the ground station as part of the time synchronization poll.

The airborne supervisor begins automatically via a vectored transfer of the bootstrap loader. The data-base, flag-base, and procedure-base registers are established, the I/O level logic is initialized; the supervisory level is initialized; the system is set for one hundred fifty prekeys and a printer message is given. When the operator responds with a carriage-return, a header is printed on the secondary terminal, and a flag is set to allow operator keyboard input.

The main logic is now entered. Subroutine MRI is called in order to initialize the modem receiving logic. The receiver is constantly monitored for one of three events. If a reception is started, no character may be spaced in excess of five milliseconds from the previous one. If this time is exceeded, the receiver is momentarily disabled and the procedure is re-started. If the quit flag is detected, the shut-down logic is invoked. If neither of these two events occur, the system waits until the end-of-reception flag (ERMF) is seen. The received poll is analyzed for allowable length and aircraft identifier. If the identifier is incorrect and/or the poll is longer or shorter than specifications require, an appropriate sentinel is printed on the secondary

printer and the procedure re-cycles.

If the length and identifier are acceptable, its classification (by length) is determined. If the operator has signaled a keyboard message for the ground station, a flag is set for requesting a general poll from the ground station, and a flag is set to show a keyboard message is in progress.

The current received poll is decoded and the response is set up by calling subroutine AVCL. If the poll contains a BCS error, a "B" is printed (in upper case for acknowledge; else, lower case). If there is no BCS error, but there is a non-acknowledge, an "N" is printed. In the case of no unusual occurrance, a "." is printed.

The response poll is then transmitted. Upon transmission completion, a "-" is printed on the secondary terminal, and the ground station's poll is awaited.

The logic which decodes a poll and formulates the response (AVCL) first determines whether the poll must be echoed; if so, flag EFT is set to one, for future reference. The receive area and transmission area indices are set up, and the current received mode is saved. The five character message identifier received in the current poll is saved, using subroutine AMOV. The poll is assumed to contain an acknowledge, until a true determination occurs.

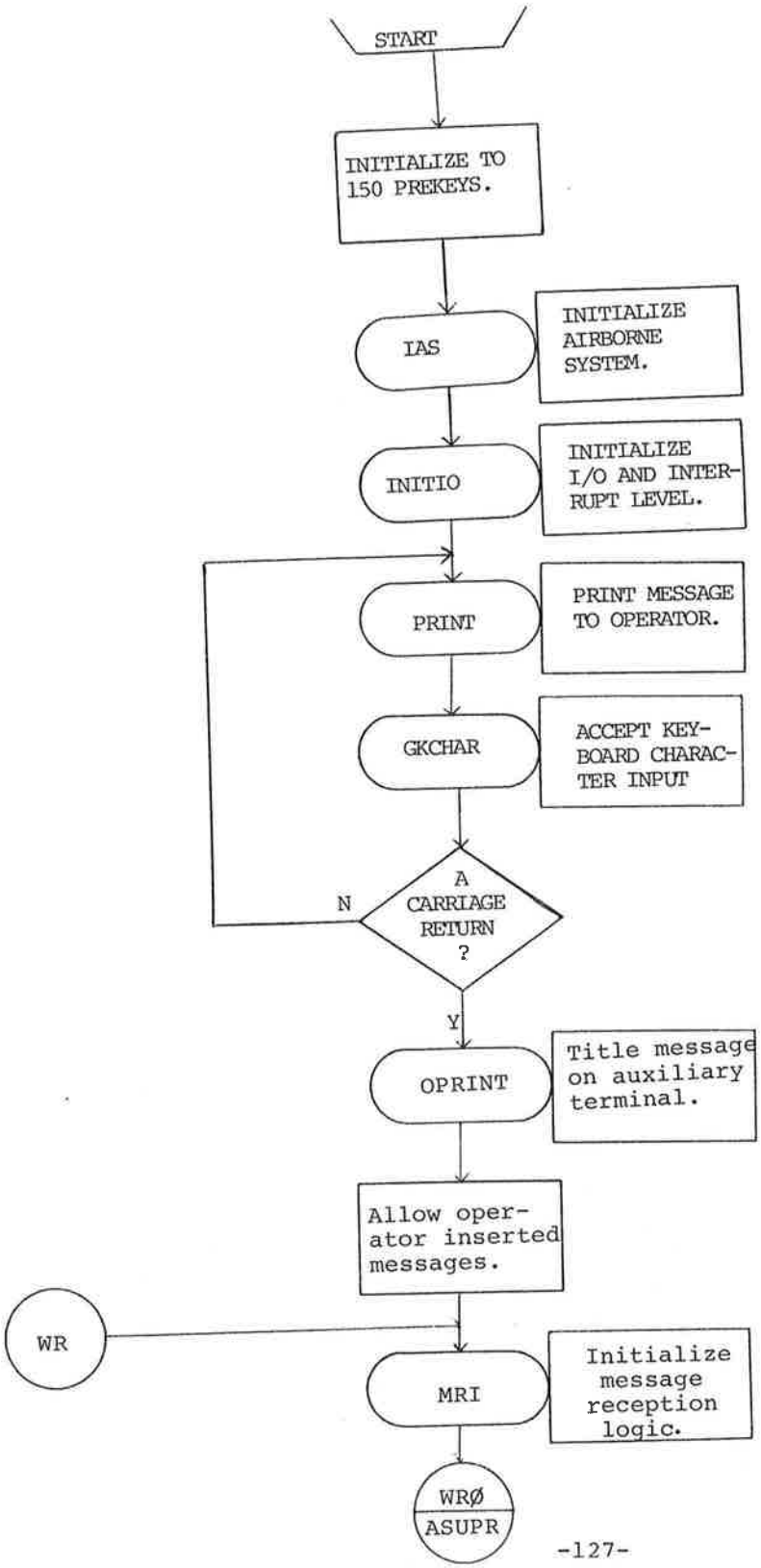
If a BCS error occurred for the current reception, a branch is taken to BCS0, to process the error. If no BCS error occurred, the FAN flag is cleared so that an acknowledge will be sent during the response poll. If the current poll is intended for the printer, it is queued. If it is a time poll, the real-time software clock is set according to the time contained in the text of the current poll. The number of non-acknowledges allowed at the ground station is used to set a BCS error counter. If an acknowledge is seen in the received poll, the appropriate statistical counter (by either echo or non-echo and text length) is tallied, the keyboard message flag is cleared and a message completion record is written on the data acquisition cassette. If an acknowledge is not seen the logic branches to ACKNOT to process a non-acknowledge. If no BCS error occurred (BCS errors predominate over NAKS), the statistical tables are updated by classification length and type and the error is recorded. A NAK-counter is tallied. If it becomes exhausted, "LINK FAIL" is printed on the secondary terminal, all keyboard requests are killed and the counter is re-set full. If the NAK counter is not exhausted, the alpha character of the received message identifier is stepped and processing continues at the point of checking for echo.

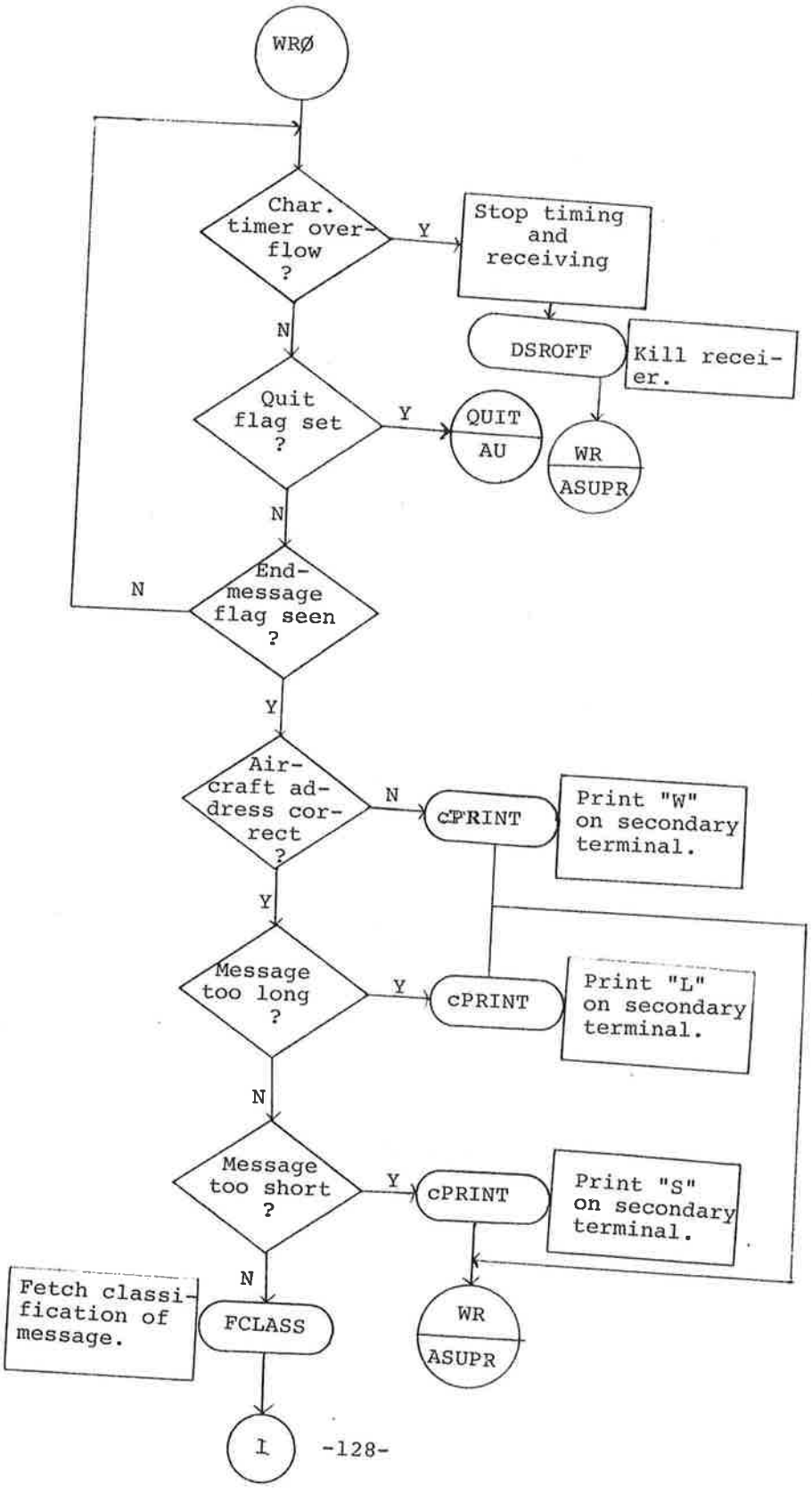
If a BCS error occurred, flag FAN is set to prompt a NAK in the reply poll. The correct statistics are updated and the error is logged on the cassette. If a BCS counter in now exhuasted, "LINK FAIL" is printed on the secondary terminal and any keyboard requests are killed. In either event processing continues at the echo

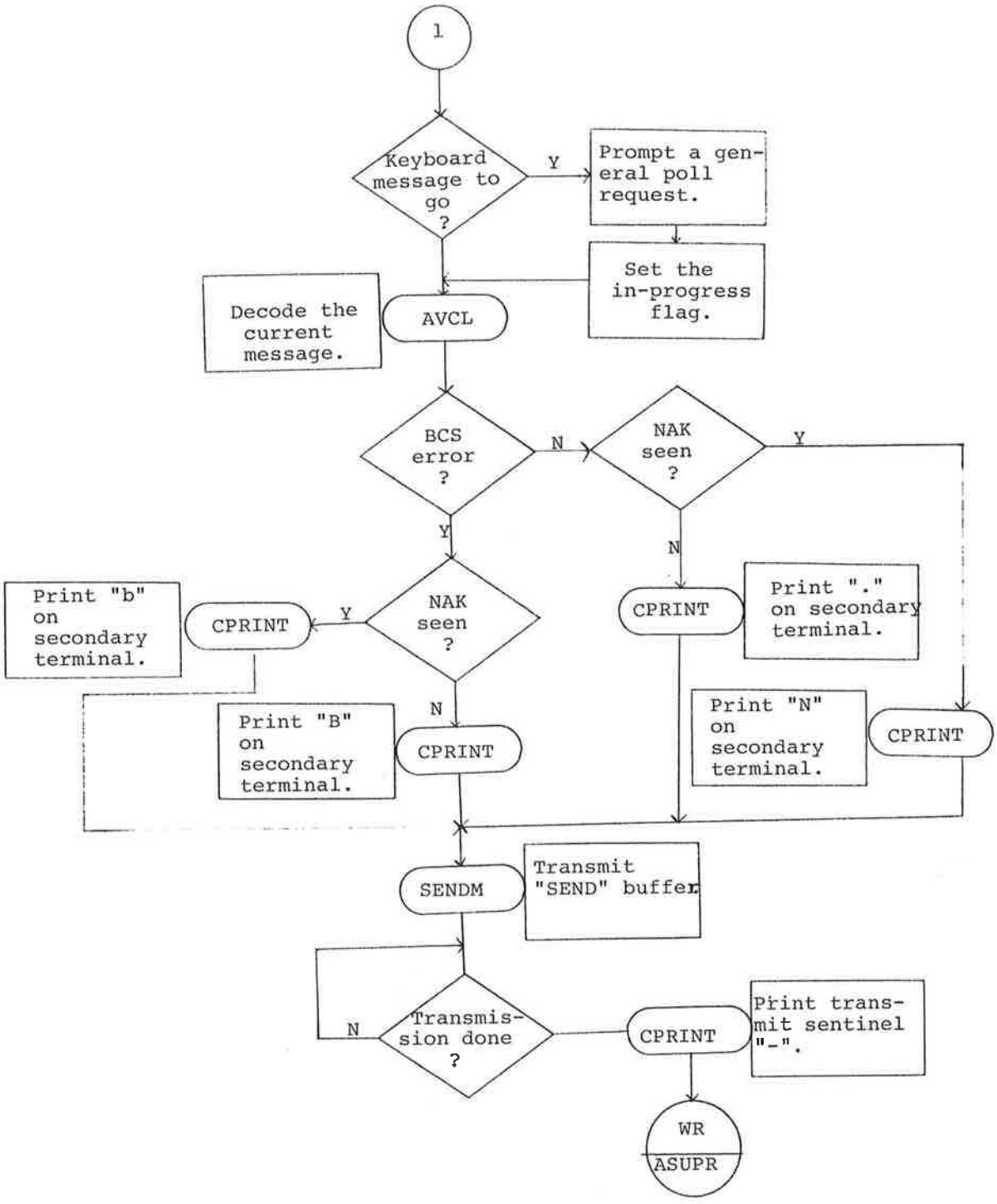
determination.

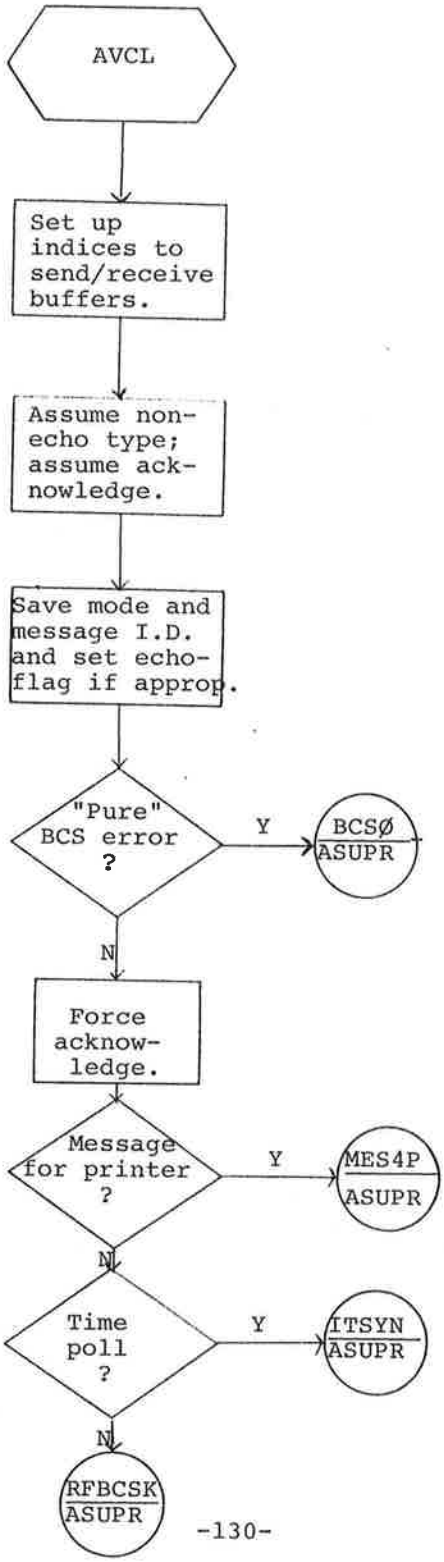
If an echo is required, logic point SUE is entered. If not, a normal mode is entered into the response poll, the preamble is set up using sub-routine PRE. If a keyboard message went out last and was not successfully acknowledged, the caller is re-entered. If a new text keyboard message awaits, the new text is moved into the reply area, the keyboard label character is set, and the caller is re-entered. If no keyboard message needs attention, an "ETX" character is jammed into the reply area and a flag is checked for a general poll request. If one is necessary, the proper label character is moved into the preamble area, and the caller is re-entered.

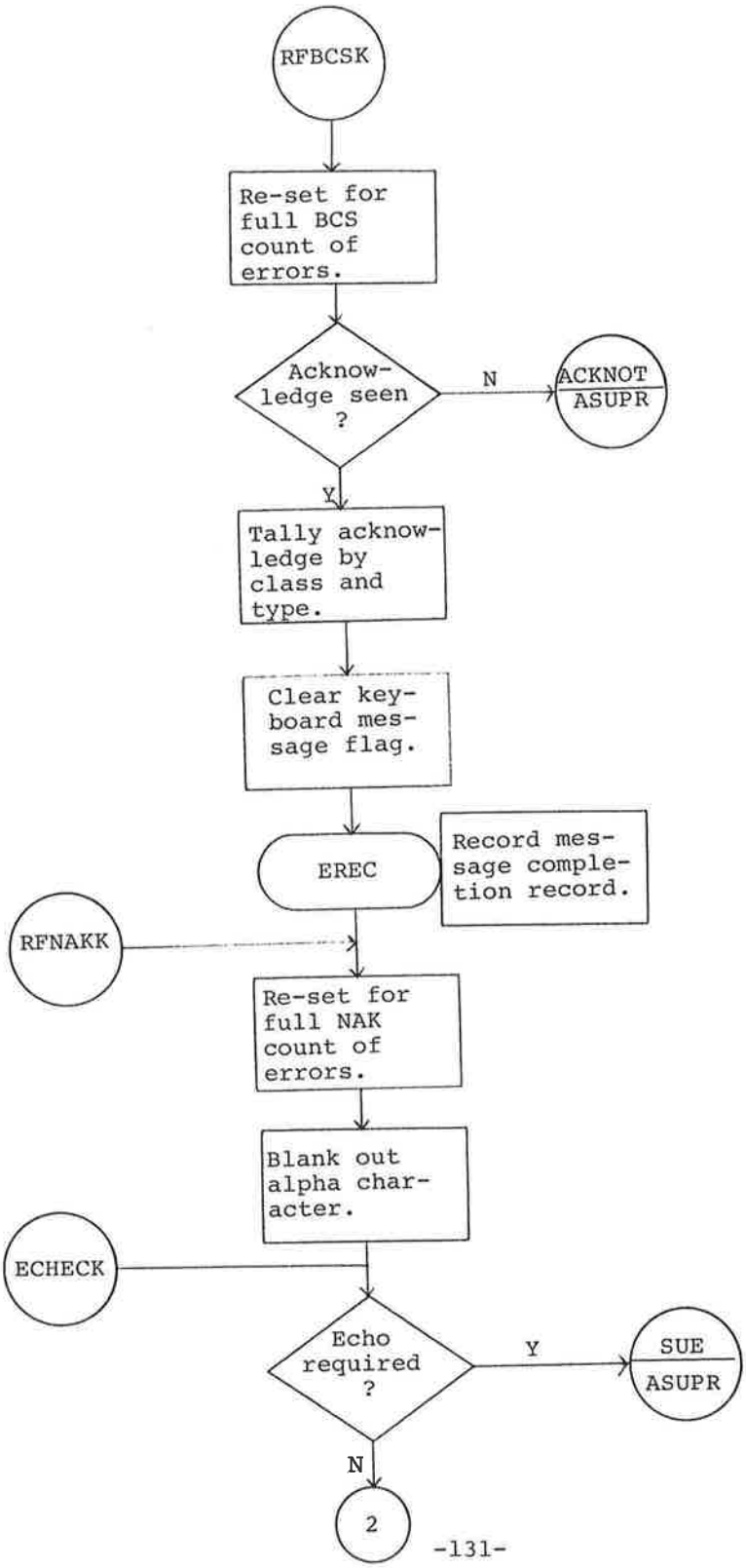
If an echo is required, the preamble is set-up by calling sub-routine PRE, the received text is moved to the reply area using subroutine AMOV, and ETX is forced after the received BCS characters, the echo mode is jammed into the preamble, the text count is forced after the "STX" character, and the caller is re-entered.

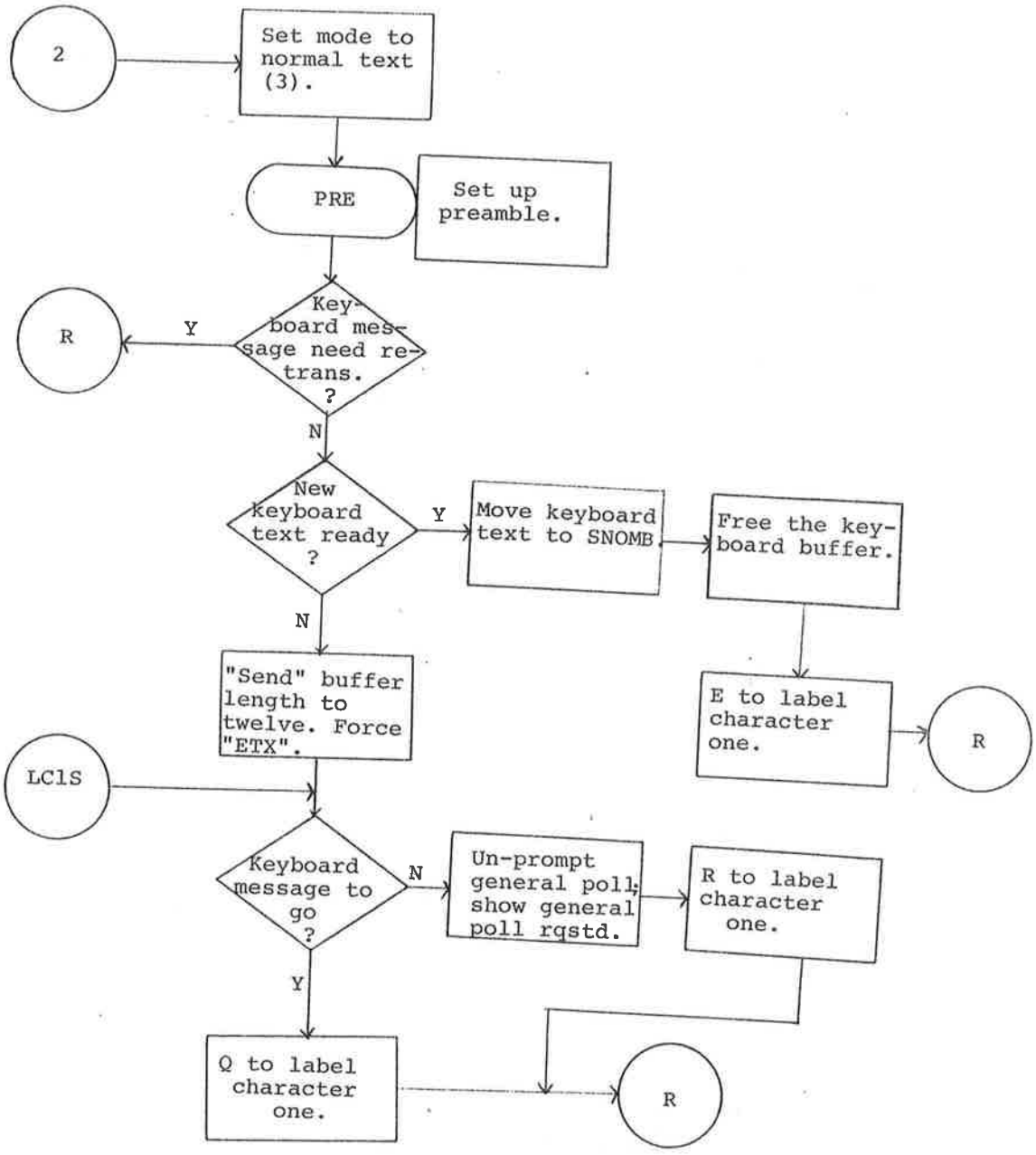

START
INITIALIZE TO 150 PREKEYS.
IAS
INITIALIZE AIRBORNE SYSTEM.
INITIO
INITIALIZE I/O AND INTERRUPT LEVEL.
PRINT
PRINT MESSAGE TO OPERATOR.
GKCHAR
ACCEPT KEYBOARD CHARACTER INPUT
A CARRIAGE RETURN ?
N
Y
OPRINT
Title message on auxiliary terminal.
Allow operator inserted messages.
WR
MRI
Initialize message reception logic.
WRØ
ASUPR

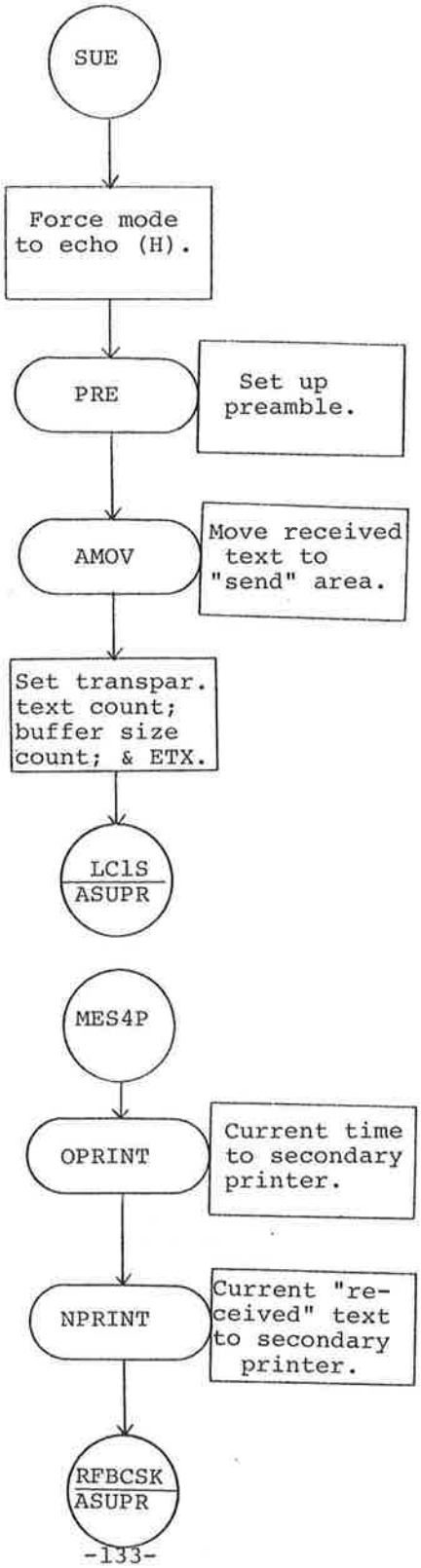

WRØ
Char. timer over-flow ?
Y
Stop timing and receiving
DSROFF
Kill receiver.
N
Quit flag set ?
Y
QUIT AU
WR ASUPR
N
End-message flag seen ?
N
Y
Air-craft address correct ?
N
cPRINT
Print "W" on secondary terminal.
Y
Message too long ?
Y
cPRINT
Print "L" on secondary terminal.
N
Message too short ?
Y
cPRINT
Print "S" on secondary terminal.
N
Fetch classification of message.
FCLASS
WR ASUPR
1

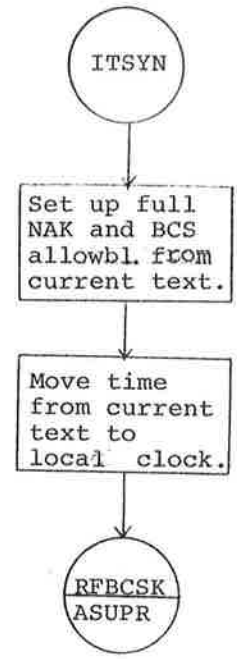

1
Keyboard message to go ?
Y
Prompt a general poll request.
Set the in-progress flag.
Decode the current message.
AVCL
BCS error ?
N
NAK seen ?
Y
Y
N
Print "b" on secondary terminal.
CPRINT
Y
NAK seen ?
CPRINT
Print "." on secondary terminal.
Print "B" on secondary terminal.
N
CPRINT
Print "N" on secondary terminal.
CPRINT
SENDM
Transmit "SEND" buffer
N
Transmission done ?
CPRINT
Print transmit sentinel "_".
WR
ASUPR

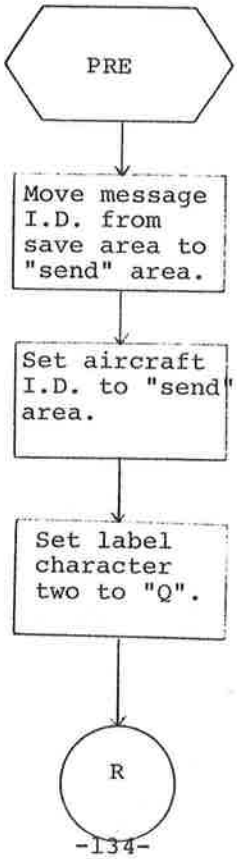

AVCL
Set up indices to send/receive buffers.
Assume non-echo type; assume acknowledge.
Save mode and message I.D. and set echo-flag if approp.
"Pure" BCS error ?
Y
BCSØ ASUPR
N
Force acknowledge.
Message for printer ?
Y
MES4P ASUPR
N
Time poll ?
Y
ITSYN ASUPR
N
RFBCSK ASUPR

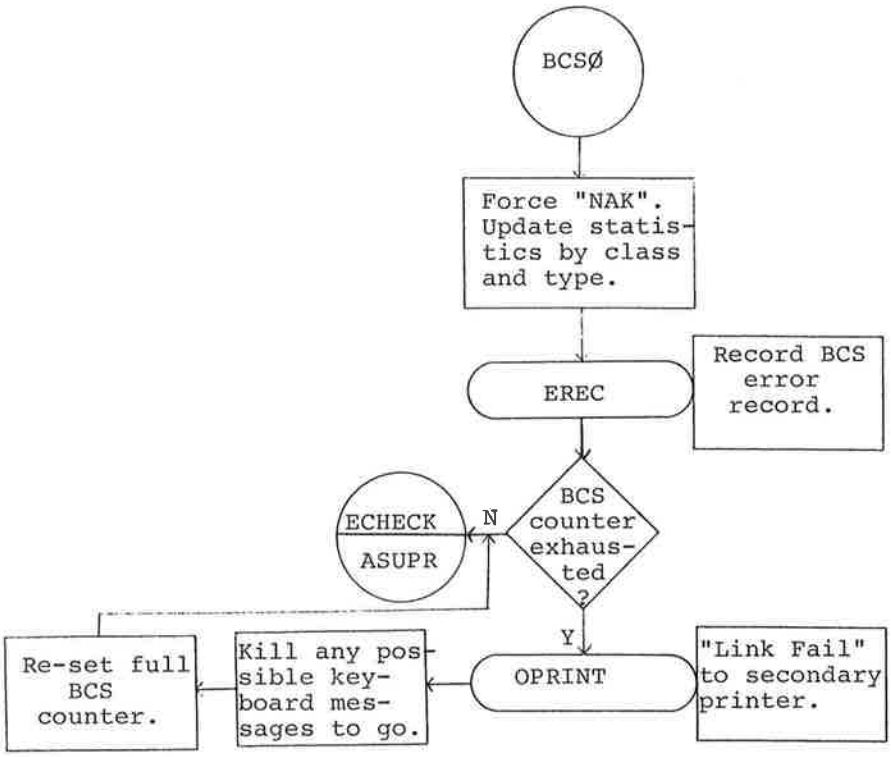

RFBCSK
Re-set for full BCS count of errors.
Acknow-ledge seen ?
N
ACKNOT
ASUPR
Y
Tally acknow-ledge by class and type.
Clear key-board mes-sage flag.
EREC
Record mes-sage comple-tion record.
RFNAKK
Re-set for full NAK count of errors.
Blank out alpha char-acter.
ECHECK
Echo required ?
Y
SUE
ASUPR
N
2

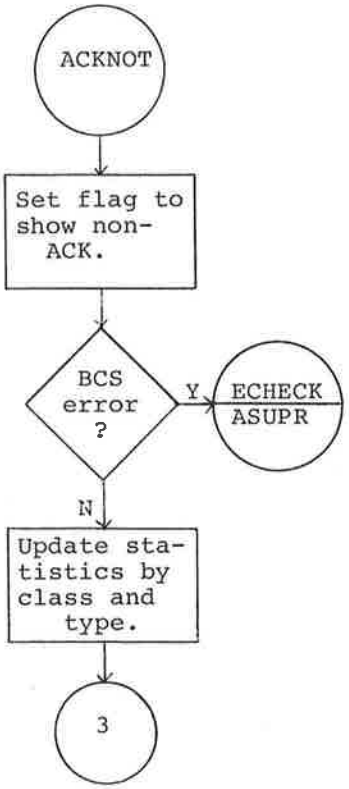

2
Set mode to normal text (3).
PRE
Set up preamble.
R
Y
Keyboard message need retrans. ?
N
New keyboard text ready ?
Y
Move keyboard text to SNOMB
Free the keyboard buffer.
N
"Send" buffer length to twelve. Force "ETX".
E to label character one.
R
LC1S
Keyboard message to go ?
N
Un-prompt general poll; show general poll rqstd.
R to label character one.
Y
Q to label character one.
R

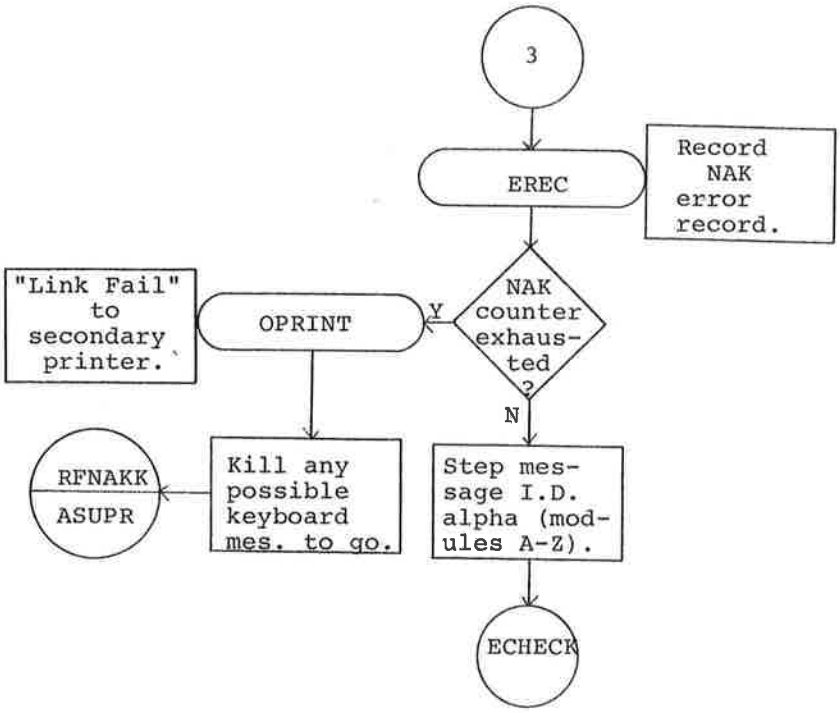

SUE
Force mode to echo (H).
PRE
Set up preamble.
AMOV
Move received text to "send" area.
Set transpar. text count; buffer size count; & ETX.
LC1S
ASUPR
MES4P
OPRINT
Current time to secondary printer.
NPRINT
Current "received" text to secondary printer.
RFBCSK
ASUPR

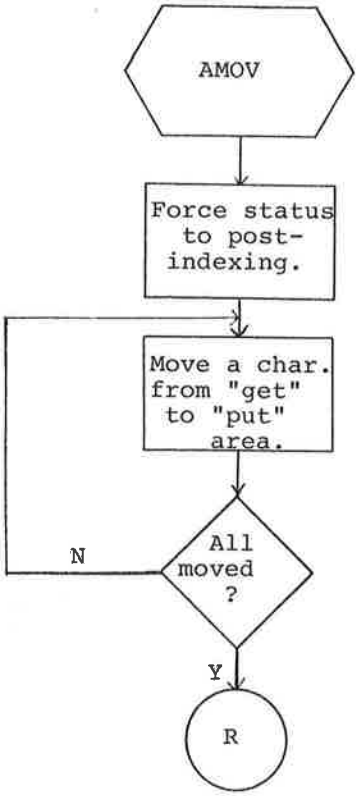

ITSYN
Set up full NAK and BCS allowbl. from current text.
Move time from current text to local clock.
RFBCSK
ASUPR


PRE
Move message I.D. from save area to "send" area.
Set aircraft I.D. to "send" area.
Set label character two to "Q".
R


BCSØ
Force "NAK". Update statistics by class and type.
EREC
Record BCS error record.
BCS counter exhausted ?
N
ECHECK
ASUPR
Y
OPRINT
"Link Fail" to secondary printer.
Kill any possible keyboard messages to go.
Re-set full BCS counter.


ACKNOT
Set flag to show non-ACK.
BCS error ?
Y
ECHECK
ASUPR
N
Update statistics by class and type.
3


3
EREC
Record NAK error record.
NAK counter exhaus-ted ?
Y
OPRINT
"Link Fail" to secondary printer.
N
Step mes-sage I.D. alpha (mod-ules A-Z).
Kill any possible keyboard mes. to go.
RFNAKK
ASUPR
ECHECK


AMOV
Force status to post-indexing.
Move a char. from "get" to "put" area.
All moved ?
N
Y
R

B. ASUPRA

This module is directly analogous to the GSUPRA module in that it is comprised solely of subroutines or functions which serve the airborne supervisor (ASUPR).

Subroutine IAS is directly analogous to the IGS subroutine of GSUPRA. It is a one-shot subroutine, called at system start-up, which is used to initialize variables of both statistical and operational nature. This insures that statistical information is not biased by previous experiments.

Subroutines SENDM, MRI, and MSGREC are identical to the same subroutines employed by the ground station and which reside in module GSUPRA.

C. AU

This module is comprised of utility-type subroutines designed to enable the airborne supervisor to be as logically functional as possible. This section describes the subroutines of the AU module in the order in which they occur in memory.

RTCIH

This subroutine is identical to the subroutine of the same name which resides in module GU, with two exceptions. There is no timer in the airborne system for the start of a message. Therefore, flag TSOM and location SOMTR are not present and their manipulation is not performed. In addition, the airborne system is required to read three factors of aircraft attitude. These three readings -- altitude, pitch, and roll -- are acquired from analogue-to-digital (A/D) conversions. The method of performing the A/D conversions is as follows:

Each time the least significant digit of time-of-day millisecond clock is a three the altitude value is read and stored, and the converter is requested to convert the analogue pitch value. At six, the pitch conversion is read and stored, and the converter is requested to convert the roll value. At nine, the roll conversion is read and stored, and the request is made to convert the altitude value, which is read at three. Therefore, all three attitude factors are completely updated each nine milliseconds, although at three millisecond intervals.

BCSIE-BCSN-BCSFE

This subroutine is absolutely identical to the subroutine of the same name in module GU.

GETAB

This subroutine is absolutely identical to the subroutine of the same name in module GU.

EREC

This subroutine is called whenever the airborne system supervisor determines that it is required to file a record on the data acquisition cassette. The conditions causing an error record to be filed are shown in Appendix G. 'Good' records are also filed using this subroutine. Normal (clear text) and transparent text messages are recorded in separate formats whose characteristics are shown in Appendices E and F.

EXCOM

This subroutine is identical to the subroutine of the same name in module GU.

MVCMP

This subroutine is identical to the subroutine of the same name in module GU.

GET

This subroutine is identical to the subroutine of the same name in module GU.

PUT

This subroutine is identical to the subroutine of the same name in module GU.

ECPUT

This subroutine is identical to the subroutine of the same name in module GU.

FCLASS

This subroutine is identical to the subroutine of the same name in module GU.

QUIT

This subroutine forces all of the on-the-fly statistical information to be encoded and scrolled out to the data acquisition cassette. A system completion message is printed on the primary terminal, and the supervisor is started up again, at the initialization level.

OOPS02

This subroutine is used to move the nine time-of-day characters from the real-time clock area to an output buffer area.

D. ABF-ABD

This module contains all of the flags, tables, variables, constants, and pointers utilized by the airborne station supervisor.

The classes of locations are defined by: system constant (K), experiment constant (C), experiment variable (V), buffer (B), and pointer (P).

FLAG	STATE	IMPLICATION
ACAF	0	A/C I.D. in received poll correct.
	1	A/C I.D. in received poll incorrect.
BCSE	0	Received poll has no length, I.D., nor BCS, error.
	1	Received poll has at least one of above errors.
EFT	0	Poll must not be echoed.
	1	Poll must be echoed.
ERMF	0	End-of-reception character not seen.
	1	End-of-reception character seen (or, don't accept modem char.).
FAN	0	Send an 'ACK'.
	1	Send a 'NAK'.
GETABF	0	'GETAB' subroutine not busy.
	1	'GETAB' subroutine busy.
KBDM	0	No keyboard buffer needs attention.
	1	A keyboard buffer needs attention.
KBDMIP	0	No keyboard buffer is in transmission.
	1	A keyboard buffer is in transmission.
KL1	0	General poll not in request state.
	1	A general poll is requested.
KL2	0	Keyboard text not in output area.
	1	Keyboard text is in output area.

FLAG	STATE	IMPLICATION
KL3	0	Keyboard message not sent at least once.
	1	A keyboard message has been sent at least once.
MD	0	Not transmitting.
	1	Transmitting.
MLFL	0	Received poll not too long.
	1	Received poll too long.
MLFS	0	Received poll not too short.
	1	Received poll too short.
NAKE	0	'ACK' seen.
	1	'ACK' not seen.
QF	0	Continue.
	1	Stop.
SOHF	0	This received char. is an 'SOH'.
	1	This received char. is not an 'SOH'.
STXF	0	An 'STX' not seen.
	1	An 'STX' has been seen.
TC	0	Do not time characters.
	1	Time characters.
TCO	0	Character clock did not overflow.
	1	Character clock overflowed.

FLAG	STATE	IMPLICATION
TSOM	0	Not used.
	1	
TSOMO	0	Not used.
	1	

FLAG	STATE	IMPLICATION
KL3	0	Keyboard message not sent at least once.
	1	A keyboard message has been sent at least once.
MD	0	Not transmitting.
	1	Transmitting.
MLFL	0	Received poll not too long.
	1	Received poll too long.
MLFS	0	Received poll not too short.
	1	Received poll too short.
NAKE	0	'ACK' seen.
	1	'ACK' not seen.
QF	0	Continue.
	1	Stop.
SOHF	0	This received char. is an 'SOH'.
	1	This received char. is not an 'SOH'.
STXF	0	An 'STX' not seen.
	1	An 'STX' has been seen.
TC	0	Do not time characters.
	1	Time characters.
TCO	0	Character clock did not overflow.
	1	Character clock overflowed.

FLAG	STATE	IMPLICATION
TSOM	0	Not used.
	1	
TSOMO	0	Not used.
	1	

LOCATION	CLASS	USE
ABBEG	K	Transfer vector.
BCSCNT	V	Interation counter for BCS calculation.
BCSEEN	V	Calculated BCS of received poll.
CB	P	Points to current keyboard buffer.
CCNT	V	Temporary use by MSGREC.
CHART	V	Character timer clock.
CKBB	V	See CB.
CLASSC	V	Classification code.
CRTCC	V	Current received text char. count.
CTIME	B	Time of day part of record buffer.
ECHOAK	V-B	Number of acknowledgements of echo polls.
ECHOBC	V-B	Number of BCS failures of echo polls.
ECHOBE		Not used.
ECHONK	V-B	Number of non-acknowledgements of echo polls.
ECHONR		Not used.
ECNT	V	Temporary use by MSGREC.
ENALT	V-B	Encoded raw altitude.

LOCATION	CLASS	USE
ENPTCH	V-B	Encoded raw pitch.
ENROLL	V-B	Encoded raw roll.
ERBUF	V	Recording buffer.
HEIGHT	V	Raw altitude A/D value.
KNPK	C	Number of prekeys, as directed ground station.
LBCSW	V	BCS calculation variable.
LSTCC		Not used.
MANB	K-P	Pointer table to MAN1, MAN2, MAN3.
MAN1	B	Keyboard buffer.
MAN2	B	Keyboard buffer.
MAN3	B	Keyboard buffer.
MS	V	Current mode of received poll.
NEAK	V-B	Non-echo: See ECHOBC.
NEBC	V-B	Non-echo: See ECHONK.
NENR		Not used.
NUMBER	V-B	Buffer length count of record buffer.

LOCATION	CLASS	USE
OMSG		Not used.
OMSG0	V-B	M.S. message numeric.
OMSG1	V-B	Next message numeric.
OMSG2	V-B	Next message numeric.
OMSG3	V-B	L.S. message numeric.
OMSG4	V-B	Message alpha.
PITCH	V	Raw pitch A/D value.
RBCSW	V	BCS calculation variable.
RCVMB	B	Reception poll buffer.
RIDC	V-B	Record code.
ROLL	V	Raw roll A/D value.
SNDMB	B	Transmission poll buffer.
SOMTR		Not used.
TODHRA	V-B	M.S. hours.
TODHRB	V-B	L.S. hours.
TODMNA	V-B	M.S. minutes.
TODMNB	V-B	L.S. minutes.

LOCATION	CLASS	USE
TODMSA	V-B	M.S. milliseconds.
TODMSB	V-B	Middle milliseconds.
TODMSC	V-B	L.S. milliseconds.
TODSCA	V-B	M.S. seconds.
TODSCB	V-B	L.S. seconds.
TT		Not used.

VI. SYSTEM INTERRUPT AND I/O LEVEL

This section describes the I/O and system interrupt processing. All of the modules in this logical level are used by all of the programs in the data link project -- ground station software, airborne station software, and both data reduction/analysis programs. The representation SIH is used both to refer to the module of that designation and to be read as "system interrupt handler." The meaning to be inferred is to be taken from the context of use.

A. SIH OVERVIEW

An interrupt occurs and the 960A vectors to the CRU interrupt location. There, the software determines which device has interrupted, clears the hardware interrupt line and sets a software interrupt flag. After an internal timer one millisecond interrupt, however, the clock is reset and is running while interrupts are disabled. It is at this point that the software interrupt flags are interrogated and the various handler routines are called to service the interrupts which have occurred. This scheme simulates a real time system clock in which the clock is running at all times. After all interrupts have been serviced (in less than one millisecond), interrupts are enabled again.

The most complex portion of the I/O has been the development of a scheme to interweave the recording of cassette data, the printing of operator messages and the echoing of keyboard entered operator input. This was necessary because of the single I/O channel allotted to the 733 recorder/printer unit.

The scheme chosen was that of constructing three circular buffers to simulate three devices. All data to be recorded is put into the record buffer (RBUF); all data to be printed is put into the print buffer (PBUF); all data to be echoed as a result of keyboard input is put into the keyboard buffer (KBUF).

When data is entered into any one of these buffers the other two buffers are checked for active I/O status. If neither buffer is actively engaged in I/O, control passes to the recently data stored buffer. This buffer then proceeds to output a character and continues outputting until it is empty or looses its activity to a higher priority buffer. If either buffer is actively engaged in I/O, control may or may not be changed.

A buffer priority for I/O activity has been defined, where KBUF has the highest priority. That is, if either RBUF or PBUF is active when KBUF receives data, control will be passed to KBUF as soon as it is practical to do so. Also, RBUF is activated by a special character stored in the PBUF. Therefore PBUF will actively transfer control to RBUF upon the outputting of a special character. After that RBUF may pass control to KBUF, but not to PBUF unless RBUF lacks data. When any buffer runs out of data it will activate any buffer which has data.

B. BINARY CODED HEX

While recording or playing back data on the TI-733 cassette unit there are certain seven bit binary codes which are used as control characters for the unit. It is therefore imperative that we do not encounter any of these control characters unexpectedly in a data stream. While playing back data which includes a hexadecimal 13, the playback unit will unexpectedly halt. While recording data which includes a hexadecimal 13, the recorder will turn itself off. These control characters are:

11 (playback on), 12 (recorder on), 13 (playback off), 14 (recorder off), and 10 followed by a 31-3F which cause RDC functions.

It should be noted that all common printable ASCII characters (hexadecimal 20-5A), capable of being inputted from the keyboard or read from a normal text message, will not cause any trouble.

Also any seven bit binary numbers or constants which are not listed above will not cause any problems. The problem arises when we are asked to playback or record binary data which includes the above mentioned control characters. Our solution to the problem is to encode any such variable binary data stream into printable ASCII characters. This scheme also allows for the visual inspection of tapes containing a variable binary data stream.

The scheme calls for the encoding of a four bit binary number as one ASCII character. The ASCII character is determined by adding a hexadecimal 30 to the value of the four bit number. To convert the ASCII character back to a four bit binary number, simply subtract a hexadecimal 30 from the ASCII value. Therefore all possible four bit binary values 0-15 map into the printable ASCII characters.

EXAMPLE: One 16 bit word maps into four ascii characters

Hexadecimal		3	F	1	2			
Binary		0011111100010010						
ASCII			N	O				
maps to:								
Hexadecimal	3	3	3	F	3	1	3	2
Binary	00110011001111110011000100110010							
ASCII		3		?		1		2

C. SIH MODULES (10)

MODULE	SUBROUTINES	FUNCTION
1) SIHI		All internal, DMAC, and CRU interrupts are vectered here, flags are set and the interrupts are cleared. Upon 1 MS clock interrupts the flags are examined and the appropriate device services routines are called.
	BRANO	Called by routines running in supervisor mode which wish to halt in worder mode
	DSROFF	To reset the DS receiver
2) SIH	RINT	Handles TTY READ interrupts from playback unit/keyboard
	WINT	Handles TTY WRITE interrupts from the record unit/printer
3) SIHA	RECEND	Flushes the record buffer onto tape
	GKCHAR	Get the next keyboard character input
	PLAY1	Playback 1 record from the AUX TTY
	KREAD	Handles input characters from AUX TTY playback and main TTY keyboard input
4) SIHB	REWINE	Rewind both units, bring them to load point
	PLON1	Set playback on cassette unit #1
	PLON2	Set playback on cassette unit #2
	RANDIO	Called by data reduction program to inhibit any calls to non-existent subroutines, lock out DS interrupts

MODULE	SUBROUTINES	FUNCTION
	OCRLF	Print a CRLF on main TTY
	GETST	Get the status character for the cassette unit
	DELAY3	Delay 3 seconds
5) SIHD	XRINT	Handle DS READ interrupts
	XWINT	Handle DS WRITE interrupts
	DSOUT	Output a buffer through the DS
6) SIHK	PROM	Print keyboard input echo on TTY
	STARTUP	Start up keyboard printing
7) SIHO	KWRITE	Handle AUX TTY WRITE interrupts
	CPRINT	Print 1 character on AUX TTY
8) SDRP	PRINTC	Print 1 character on TTY
	PRINTN	Print N character on TTY
	PRINT	Print 1 buffer on TTY
	RECORD	Record 1 record on tape
	PLAY	Playback 1 record from tape
9) INS	INITIO	Initilization subroutine
10) SDRDF		Flag segment and data base segment

D. FLAGS IN SIH MODULES

1. SYSTEM

INPUT	1 if playback input, 0 if keyboard input
OUTPUT	1 if recorder output, 0 if printer output
PACT	1 if printer buffer has started outputting characters
PACT1	1 if printer is actively printing its buffer
PACT2	1 if printer is actively printing keyboard buffer echoes
LAYON	1 if playback on character has been put in print buffer
LAYO1	1 if PLAY1 is being called to playback the AUX TTY cassette unit

2. INTERRUPT

DSRI	1 if DS READ interrupt occurred
DSWI	1 if DS WRITE interrupt occurred
MTYRI	1 if TTY READ interrupt occurred
MTYWI	1 if TTY READ interrupt occurred
OPYRI	1 if AUXTTY READ interrupt occurred
OPYWI	1 if AUXTTY WRITE interrupt occurred

3. CONDITIONAL

PLAERR	1 if there is a playback time out error
WAITKY	1 if there is a completed keyboard message waiting to set the KBDM flag when the KBDMIP flag goes low
STATC	1 if the next input character is a cassette status character
RECND	1 if recend is calling record
SREC	1 if a tape record on character is in the print buffer

4. USER SET

ACOPIN	Set to 1 to allow the keyboard to accept operator input
SCEN	Set to 1 to allow the decoding of BCH input from tape and to delay the return of play until the tape play-back is completed

5. VARIABLE

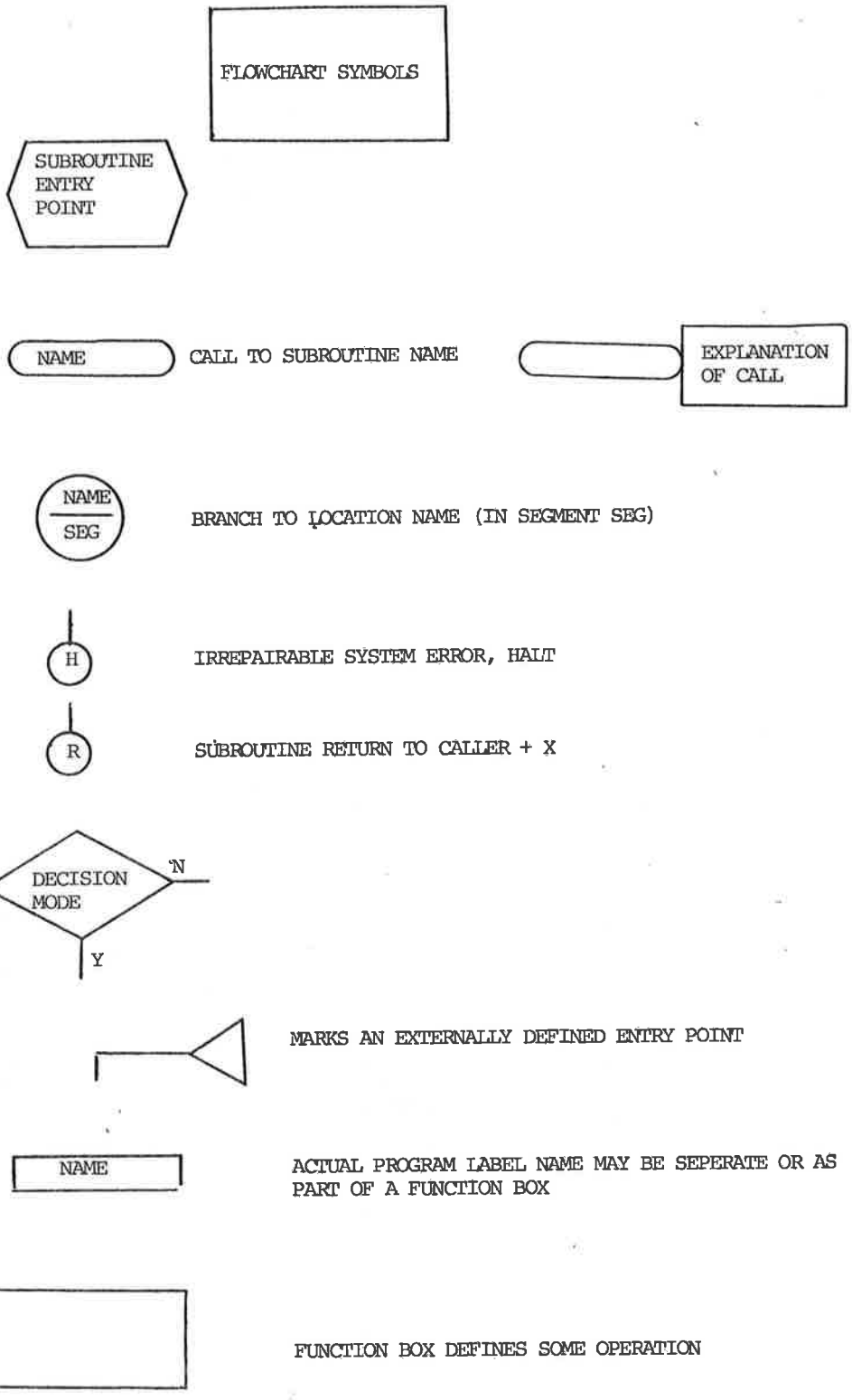
SPREKEY	1 if XWINT is sending prekey
SPREAM	1 if XWINT is sending preamble
TRANS	1 if RINT sees first BCS char of transparent text
TRRH	1 if RINT is decoding right half of BCH transparent text
TRRL	1 if RINT is decoding left half of BCH transparent text
GETBX	1 if KREAD sees an ETX/ETB
KCHAR1	1 if KREAD sees the first char of a keyboard buffer
DUMMY	1 if WINT is printing a dummy character
LIDLE	1 if WINT will not print a dummy character after the next character
PUT13	1 if WINT has just putout an X'13' to end a record
FREC	1 if WINT has just putout an X'14' to turn off the record unit
DUMMY1	1 if KRITE is printing a dummy character on AUX TTY
DUMMY2	1 if PROM is printing a dummy character on AUX TTY (keyboard echo)

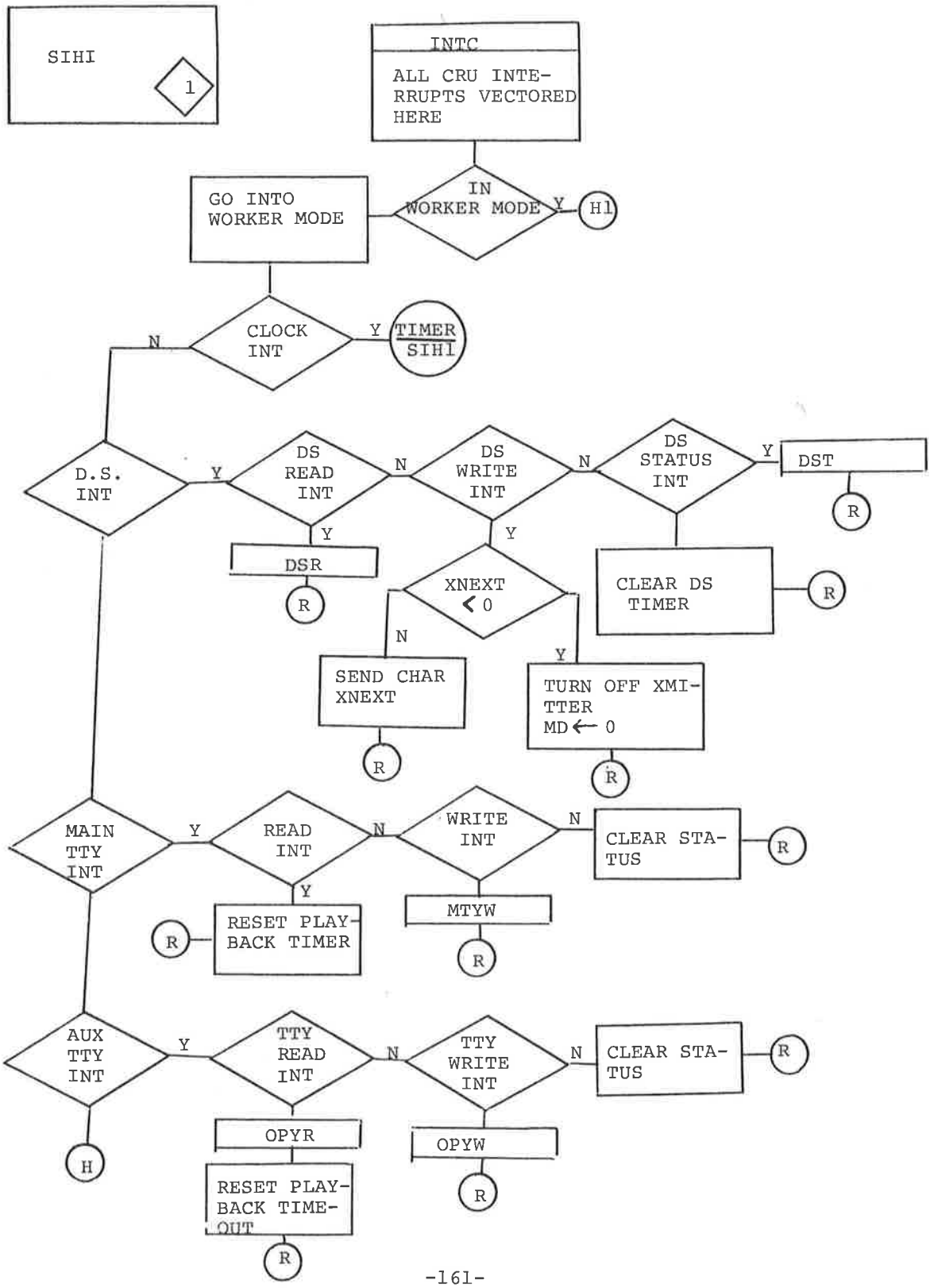
E. SYSTEM VARIABLES

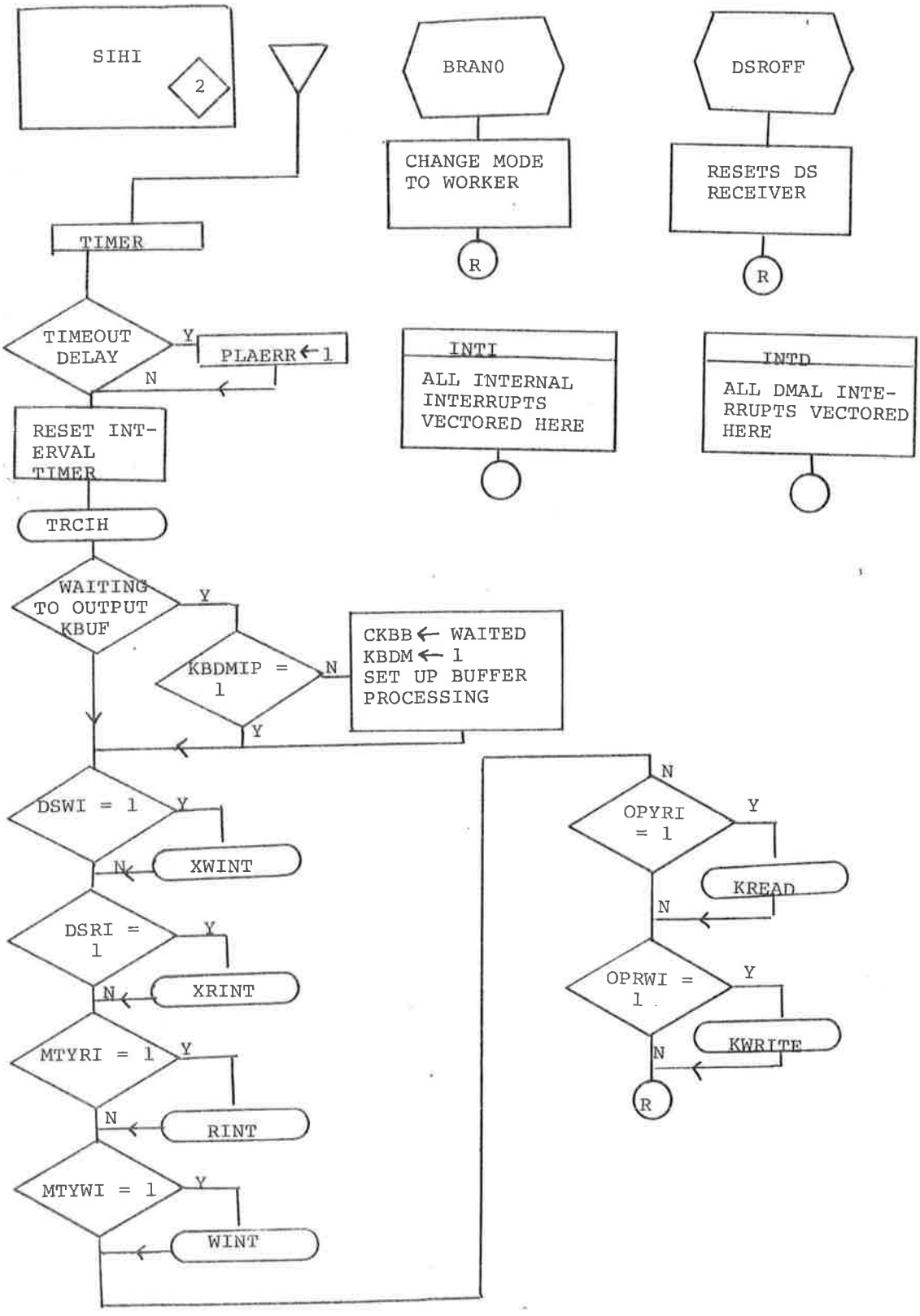
CHARKY	LAST CHAR INPUT FROM MAIN KEYBOARD
CHARST	LAST CASSETTE STATUS CHARACTER RECEIVED
CHARIN	LAST CHAR READ FROM MAIN TTY
CHARO	LAST CHAR WRITTEN FROM MAIN TTY
KOUT	NEXT CHAR TO BE PRINTED AS KEYBOARD ECHO
KCHIN	LAST CHAR READ FROM AUX TTY
CHAR01	LAST CHAR WRITTEN ON AUX TTY
XSTAT	LAST DATA SET INTERRUPT STATUS CHANGE
XNEXT	NEXT CHAR TO BE WRITTEN ON DS
XCHIN	LAST CHAR READ FROM DS
XCHOUT	LAST CHAR WRITTEN ON DS

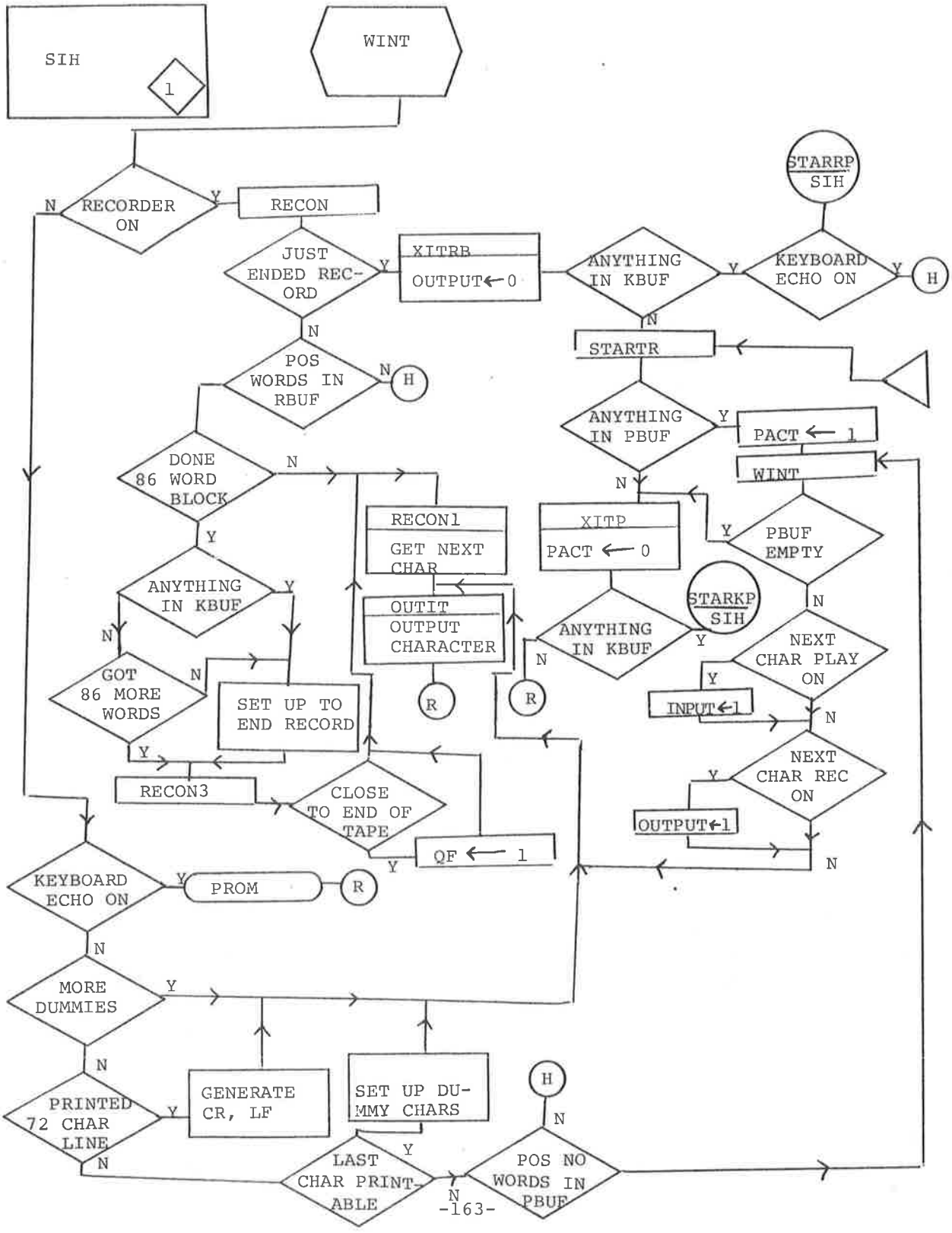
I/O DATA BASE (CIRCULAR BUFFERS)

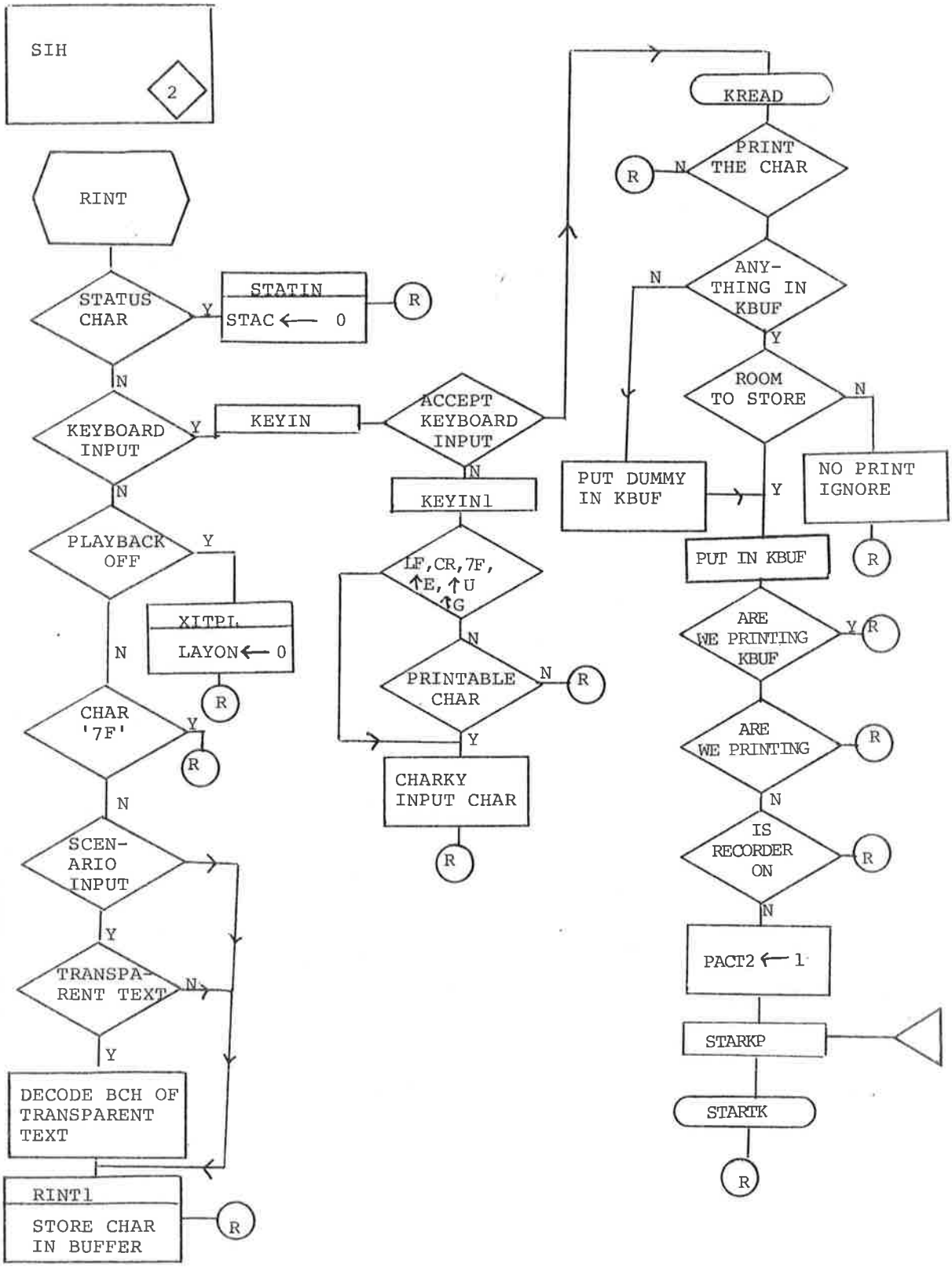
BUFFER	SIZE (DEC)	# WORDS	POINTER	ROUTINE TO FILL BUFFER	ROUTINE TO EMPTY BUFFER	BUFFER FUNCTION
PBUF	500	PNWDS	PPTR	PRINTC PRINT PRINTN	WINT	PRINT BUFFER
RBUF	900	RNWDS	RPTR	RECORD	WINT	RECORD BUFFER
KBUF	232	KNWDS	RPTR	KREAD	WINT PROM	KEYBOARD ECHO BUFFER
PBUF1	300	PNWDS1	PPTR1	OPRINT CPRINT NPRINT	KWRITE	AUX TTY PRINT BUFFER

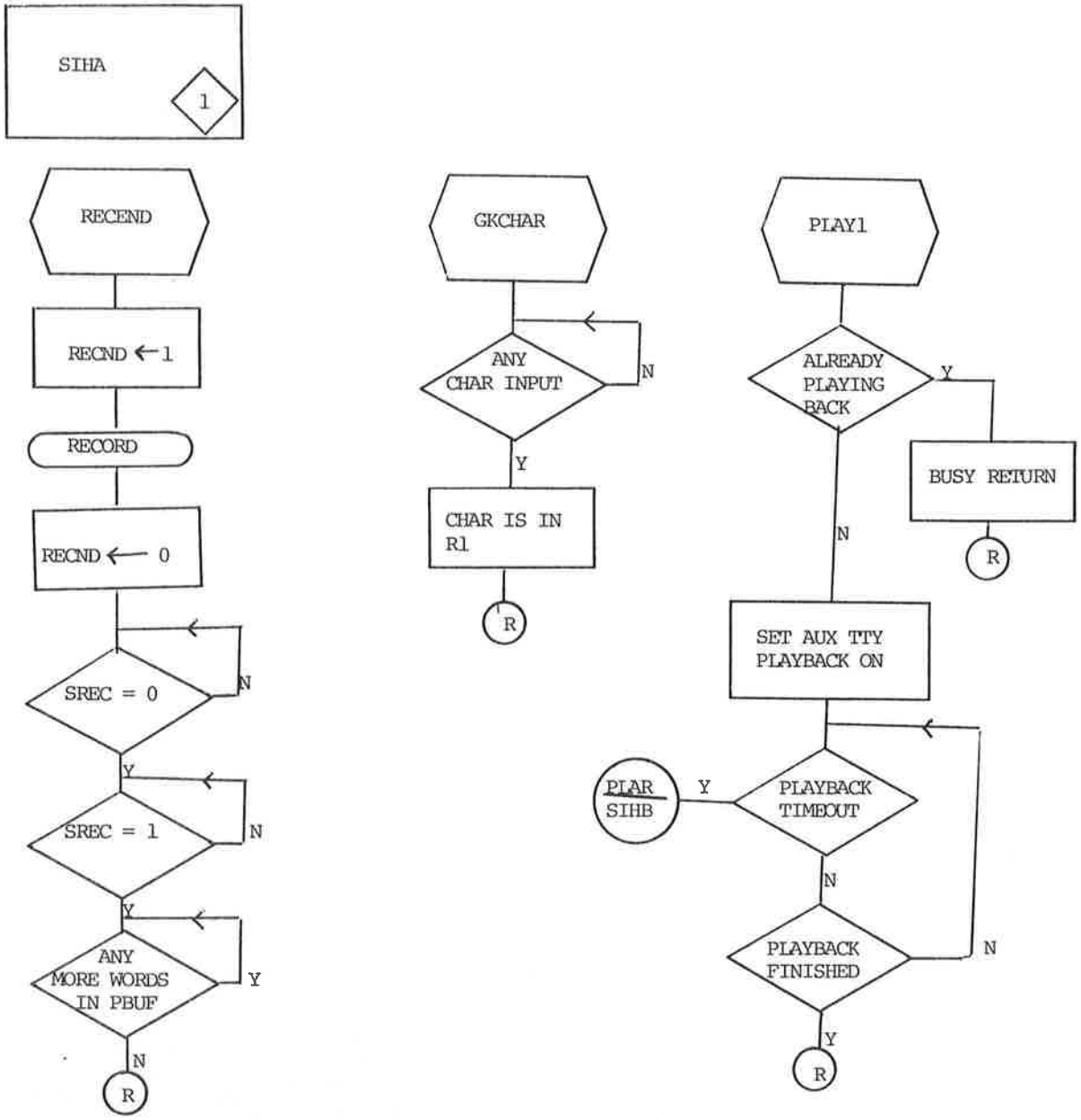

FLOWCHART SYMBOLS
SUBROUTINE ENTRY POINT
NAME
CALL TO SUBROUTINE NAME
EXPLANATION OF CALL
NAME
SEG
BRANCH TO LOCATION NAME (IN SEGMENT SEG)
H
IRREPAIRABLE SYSTEM ERROR, HALT
R
SUBROUTINE RETURN TO CALLER + X
DECISION MODE
N
Y
MARKS AN EXTERNALLY DEFINED ENTRY POINT
NAME
ACTUAL PROGRAM LABEL NAME MAY BE SEPERATE OR AS PART OF A FUNCTION BOX
FUNCTION BOX DEFINES SOME OPERATION

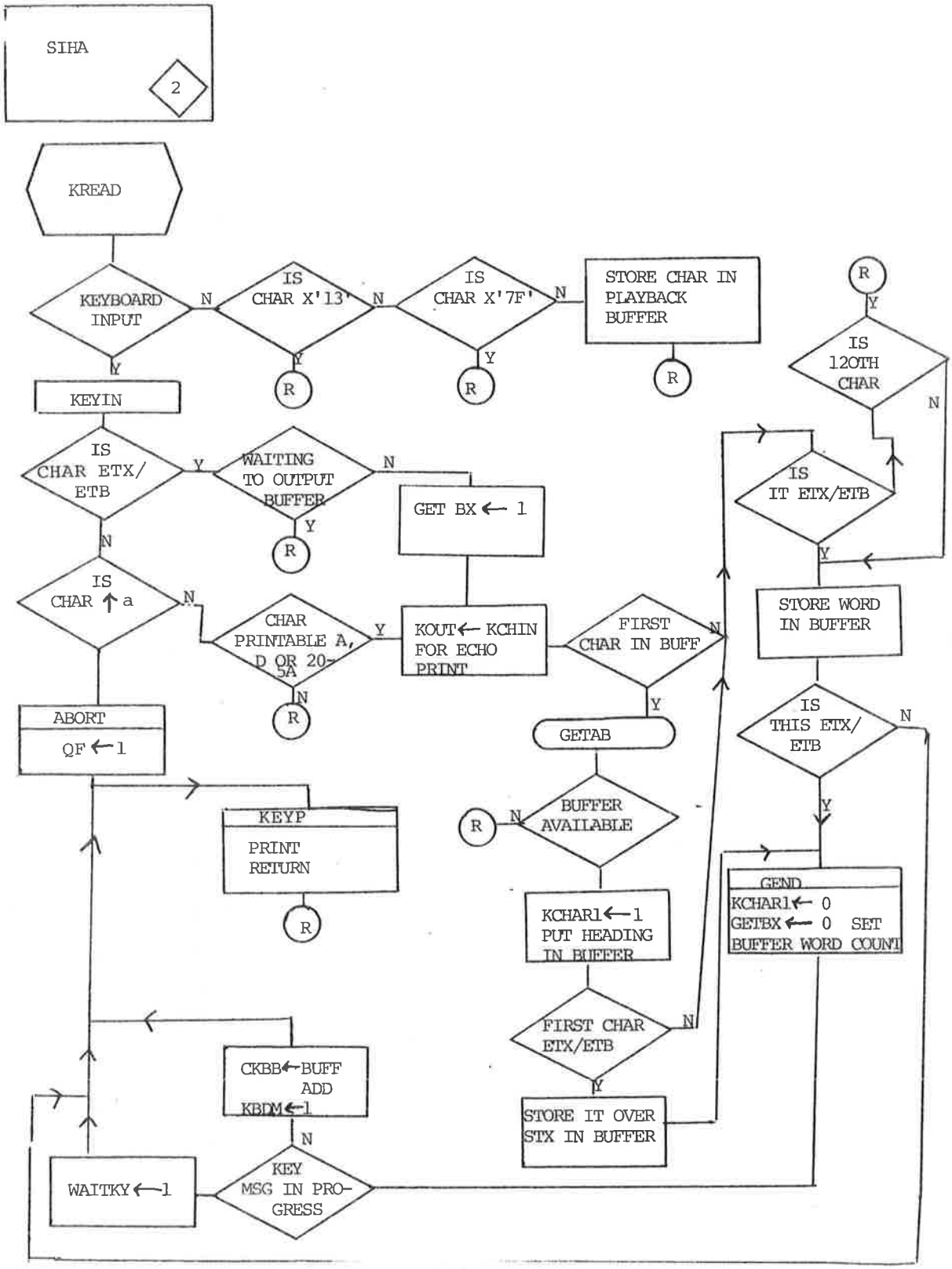

SIHI
1
INTC
ALL CRU INTE-
RRUPTS VECTORED
HERE
GO INTO
WORKER MODE
IN
WORKER MODE
Y
H1
CLOCK
INT
N
Y
TIMER
SIHI
D.S.
INT
Y
DS
READ
INT
N
DS
WRITE
INT
N
DS
STATUS
INT
Y
DST
R
Y
DSR
R
Y
XNEXT
< 0
CLEAR DS
TIMER
R
N
SEND CHAR
XNEXT
R
Y
TURN OFF XMI-
TTER
MD ← 0
R
MAIN
TTY
INT
Y
READ
INT
N
WRITE
INT
N
CLEAR STA-
TUS
R
Y
R
RESET PLAY-
BACK TIMER
MTYW
R
AUX
TTY
INT
Y
TTY
READ
INT
N
TTY
WRITE
INT
N
CLEAR STA-
TUS
R
H
OPYR
RESET PLAY-
BACK TIME-
OUT
R
OPYW
R

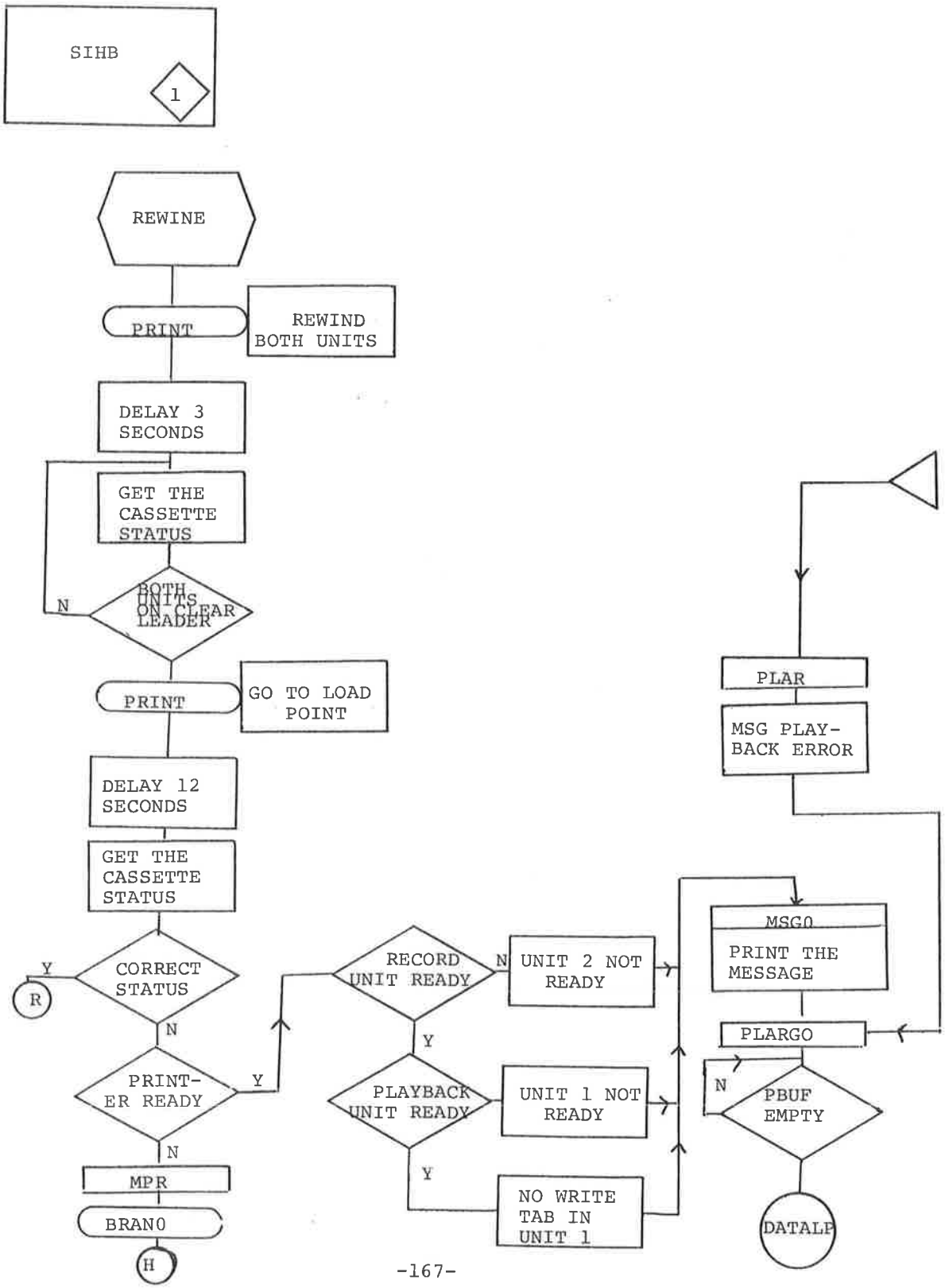

SIHI
2
TIMER
TIMEOUT DELAY
Y
PLAERR ← 1
N
RESET INT-ERVAL TIMER
TRCIH
WAITING TO OUTPUT KBUF
Y
KBDMIP = 1
N
CKBB ← WAITED
KBDM ← 1
SET UP BUFFER PROCESSING
Y
DSWI = 1
Y
XWINT
N
DSRI = 1
Y
XRINT
N
MTYRI = 1
Y
RINT
N
MTYWI = 1
Y
WINT
BRAN0
CHANGE MODE TO WORKER
R
DSROFF
RESETS DS RECEIVER
R
INTI
ALL INTERNAL INTERRUPTS VECTORED HERE
INTD
ALL DMAL INTE-RRUPTS VECTORED HERE
N
OPYRI = 1
Y
KREAD
N
OPRWI = 1
Y
KWRITE
N
R

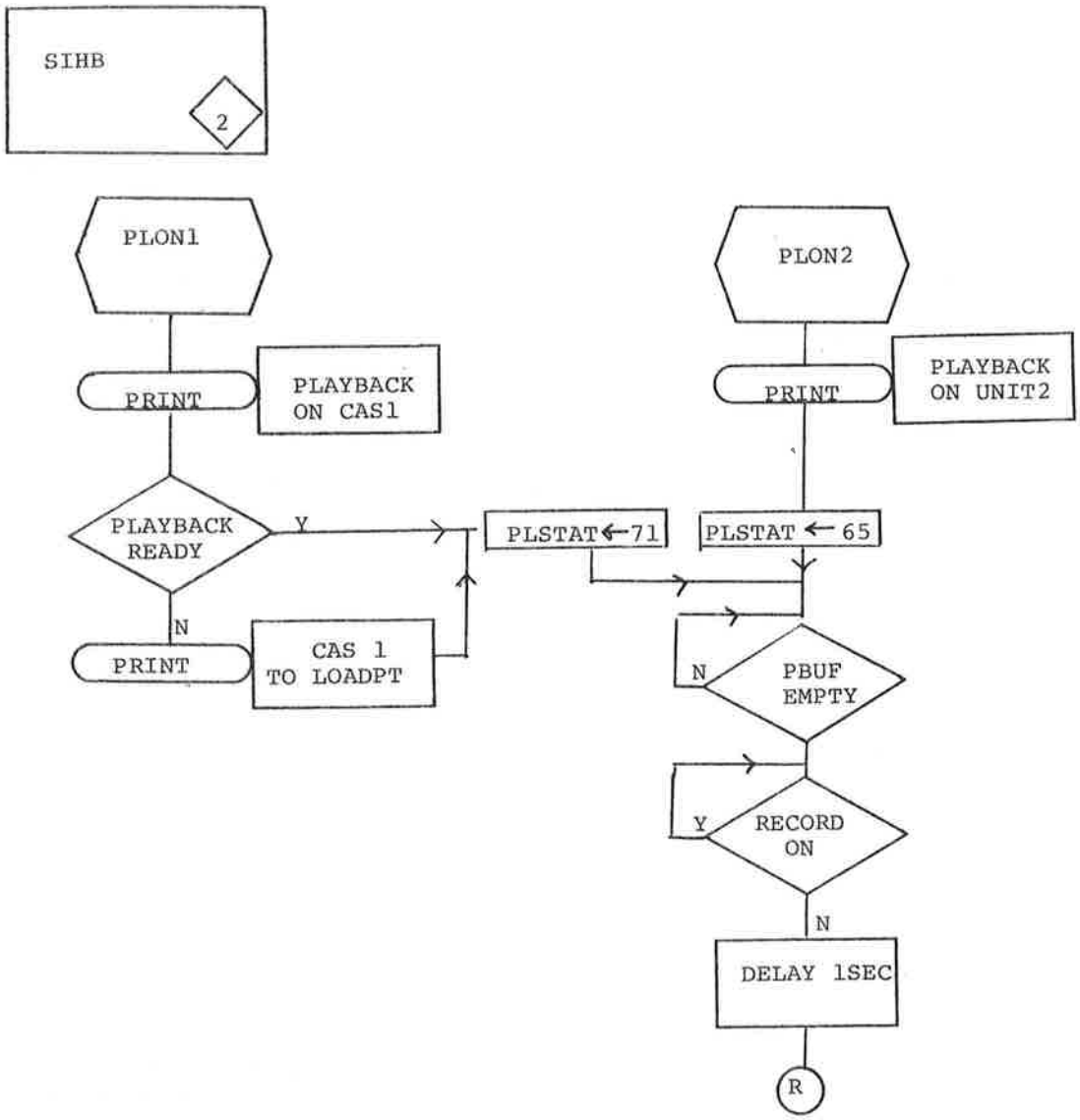

SIH
1
WINT
RECORDER ON
N
Y
RECON
JUST ENDED RECORD
Y
XITRB
OUTPUT ← 0
ANYTHING IN KBUF
Y
KEYBOARD ECHO ON
Y
H
STARRP SIH
N
STARTR
POS WORDS IN RBUF
N
H
N
DONE 86 WORD BLOCK
N
Y
ANYTHING IN KBUF
Y
N
GOT 86 MORE WORDS
N
Y
SET UP TO END RECORD
RECON3
CLOSE TO END OF TAPE
QF ← 1
Y
RECON1
GET NEXT CHAR
OUTIT
OUTPUT CHARACTER
R
R
ANYTHING IN PBUF
Y
PACT ← 1
WINT
N
XITP
PACT ← 0
PBUF EMPTY
Y
N
STARKP SIH
ANYTHING IN KBUF
Y
N
NEXT CHAR PLAY ON
Y
INPUT ← 1
N
NEXT CHAR REC ON
Y
OUTPUT ← 1
N
KEYBOARD ECHO ON
Y
PROM
R
N
MORE DUMMIES
Y
N
PRINTED 72 CHAR LINE
Y
N
GENERATE CR, LF
SET UP DUMMY CHARS
LAST CHAR PRINTABLE
Y
N
H
N
POS NO WORDS IN PBUF

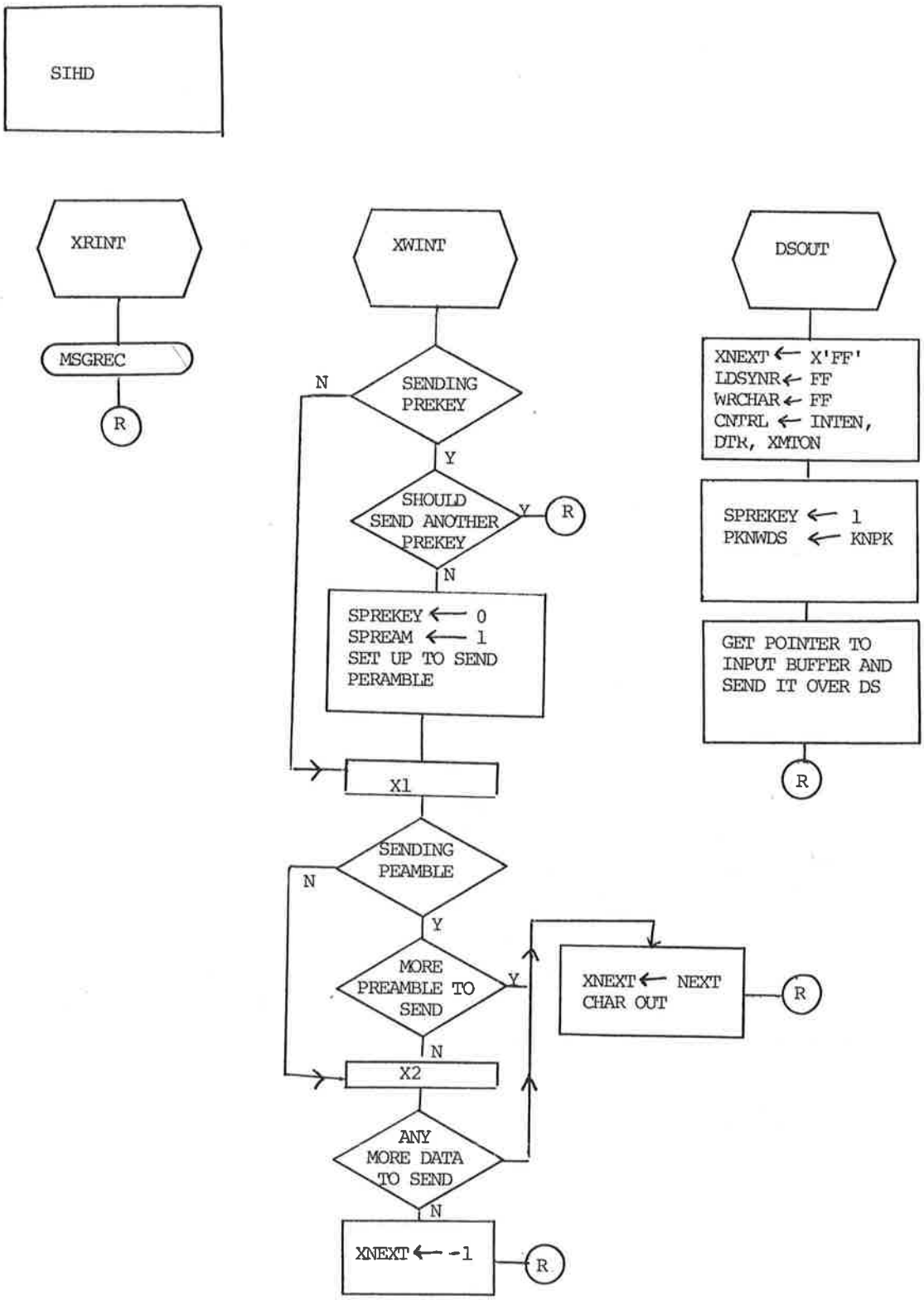

SIH
2
RINT
STATUS CHAR
Y
STATIN
STAC ← 0
R
N
KEYBOARD INPUT
Y
KEYIN
ACCEPT KEYBOARD INPUT
N
KEYIN1
PLAYBACK OFF
Y
XITPL
LAYON ← 0
R
N
CHAR '7F'
Y
R
N
SCEN-ARIO INPUT
Y
TRANSPA-RENT TEXT
N
Y
DECODE BCH OF TRANSPARENT TEXT
RINT1
STORE CHAR IN BUFFER
R
LF,CR,7F, ↑E, ↑U ↑G
N
PRINTABLE CHAR
N
R
Y
CHARKY INPUT CHAR
R
KREAD
R
N
PRINT THE CHAR
N
ANY-THING IN KBUF
Y
ROOM TO STORE
N
NO PRINT IGNORE
R
PUT DUMMY IN KBUF
Y
PUT IN KBUF
ARE WE PRINTING KBUF
Y
R
ARE WE PRINTING
R
N
IS RECORDER ON
R
N
PACT2 ← 1
STARKP
STARTK
R

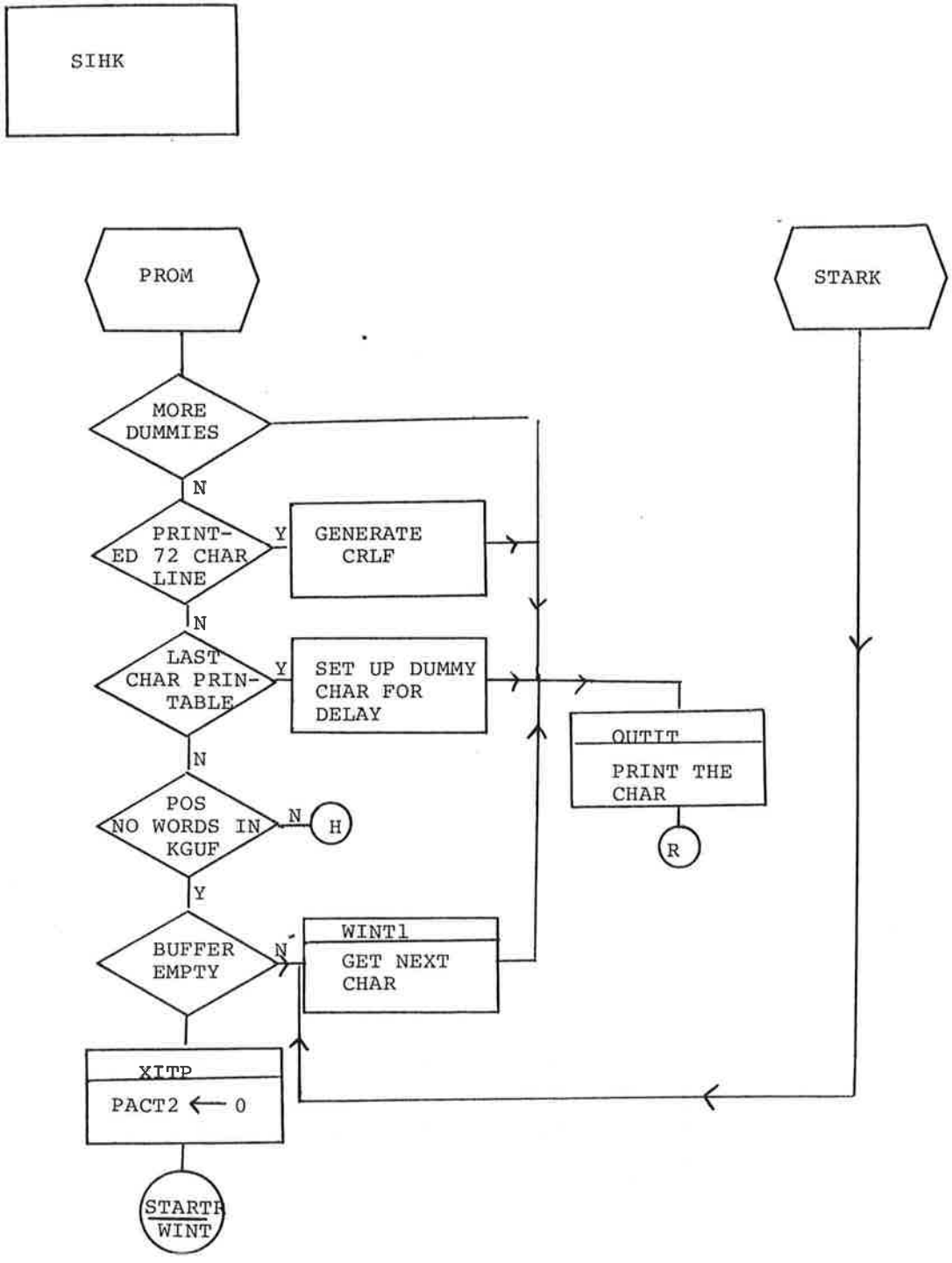

SIHA
1
RECEND
RECND ← 1
RECORD
RECND ← 0
SREC = 0
N
Y
SREC = 1
N
Y
ANY
MORE WORDS
IN PBUF
Y
N
R
GKCHAR
ANY
CHAR INPUT
N
Y
CHAR IS IN
R1
R
PLAY1
ALREADY
PLAYING
BACK
Y
BUSY RETURN
R
N
SET AUX TTY
PLAYBACK ON
PLAR
SIHB
Y
PLAYBACK
TIMEOUT
N
PLAYBACK
FINISHED
N
Y
R

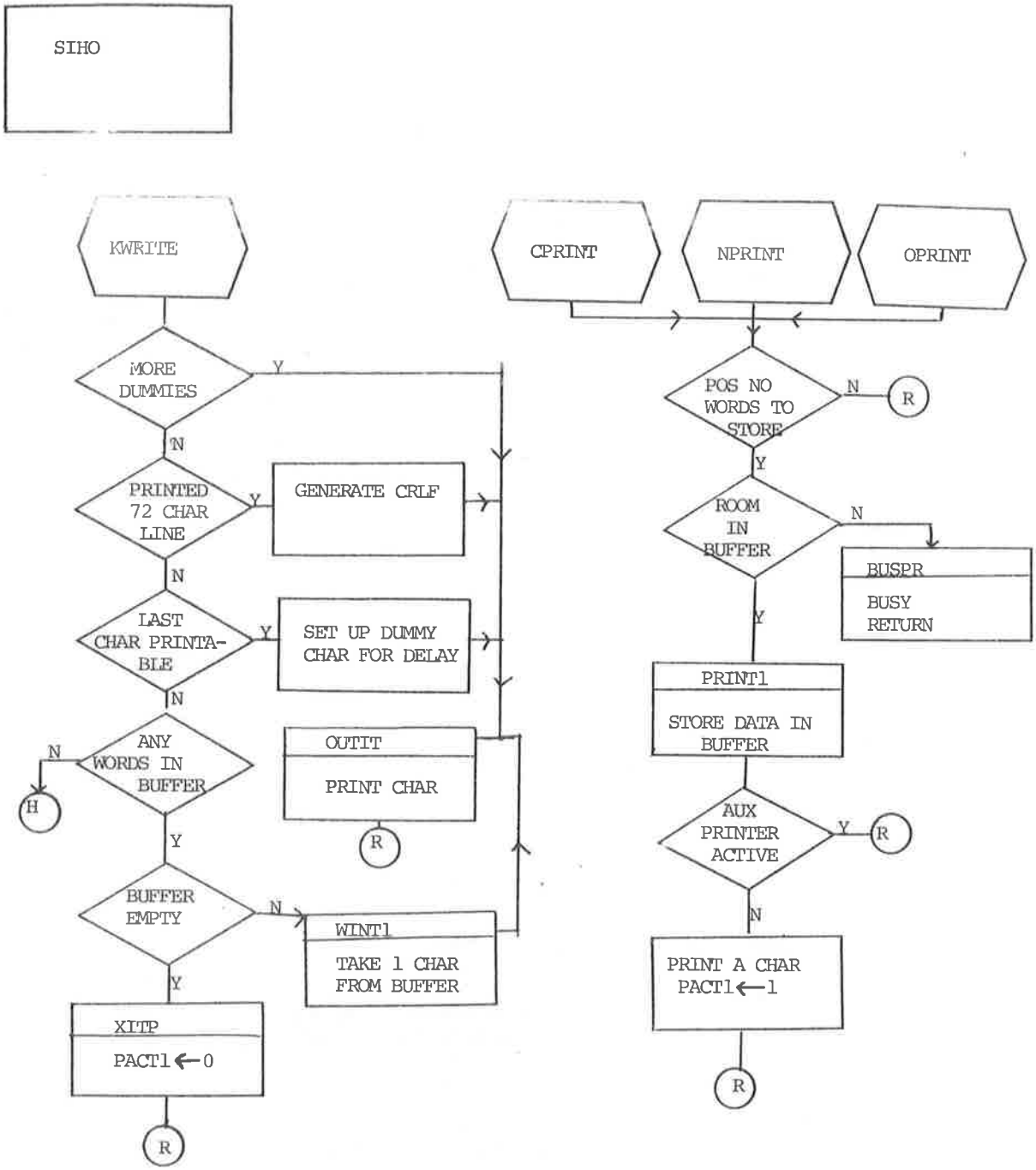

SIHA
2
KREAD
KEYBOARD INPUT
N
IS CHAR X'13'
N
IS CHAR X'7F'
N
STORE CHAR IN PLAYBACK BUFFER
Y
R
KEYIN
IS CHAR ETX/ ETB
Y
WAITING TO OUTPUT BUFFER
N
GET BX ← 1
N
IS CHAR ↑a
N
CHAR PRINTABLE A, D OR 20-5A
Y
KOUT ← KCHIN FOR ECHO PRINT
FIRST CHAR IN BUFF
N
Y
ABORT
QF ← 1
GETAB
KEYP
PRINT RETURN
BUFFER AVAILABLE
N
KCHAR1 ← 1
PUT HEADING IN BUFFER
FIRST CHAR ETX/ETB
N
Y
STORE IT OVER STX IN BUFFER
CKBB ← BUFF ADD
KBDM ← 1
WAITKY ← 1
KEY MSG IN PRO-GRESS
IS 120TH CHAR
N
IS IT ETX/ETB
STORE WORD IN BUFFER
IS THIS ETX/ ETB
N
GEND
KCHAR1 ← 0
GETBX ← 0 SET BUFFER WORD COUNT

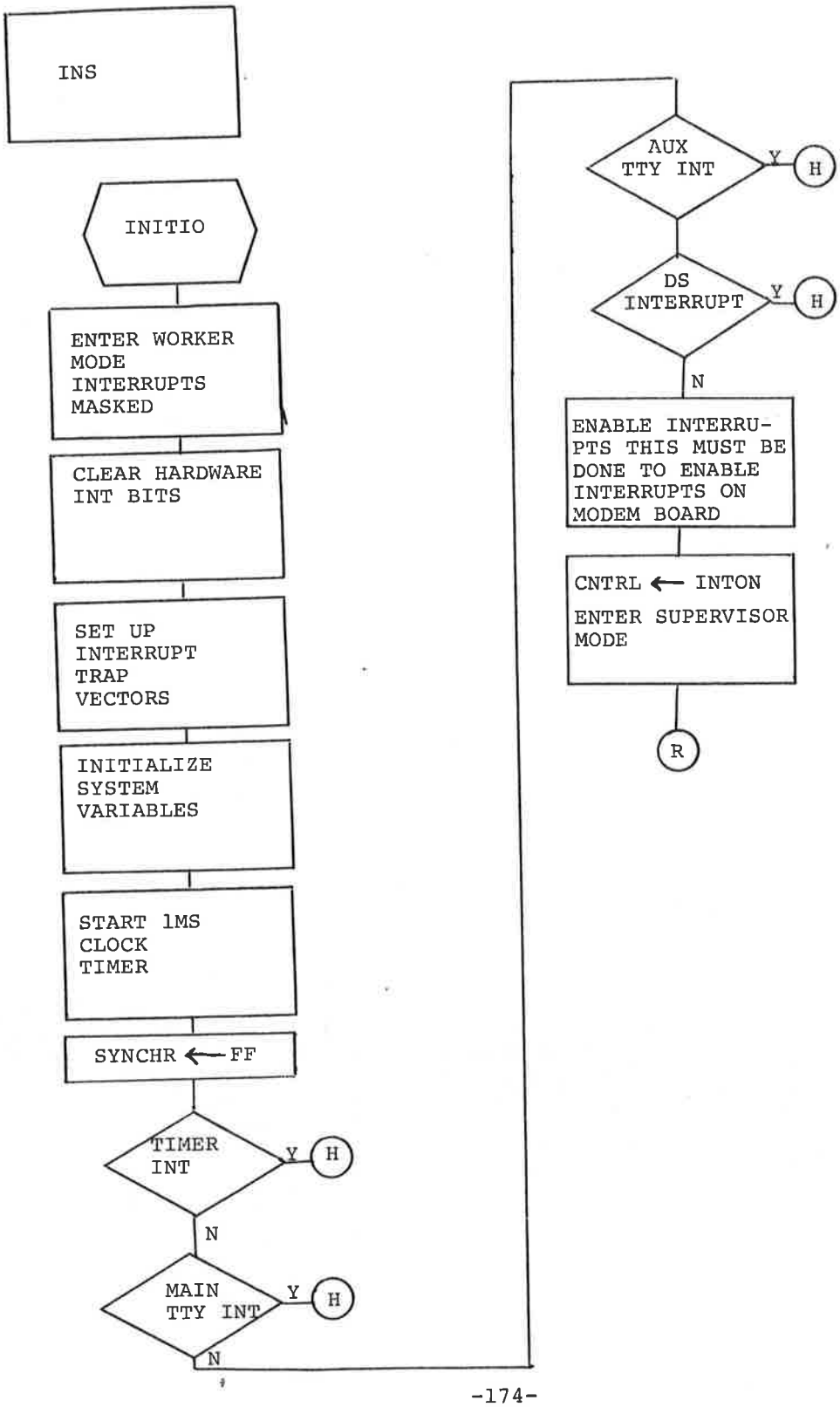

SIHB
1
REWINE
PRINT
REWIND BOTH UNITS
DELAY 3 SECONDS
GET THE CASSETTE STATUS
BOTH UNITS ON-CLEAR LEADER
N
PRINT
GO TO LOAD POINT
DELAY 12 SECONDS
GET THE CASSETTE STATUS
CORRECT STATUS
Y
R
N
PRINT-ER READY
Y
N
MPR
BRAN0
H
RECORD UNIT READY
N
UNIT 2 NOT READY
Y
PLAYBACK UNIT READY
UNIT 1 NOT READY
Y
NO WRITE TAB IN UNIT 1
PLAR
MSG PLAY-BACK ERROR
MSG0
PRINT THE MESSAGE
PLARGO
N
PBUF EMPTY
DATALP


SIHB
2
PLON1
PRINT
PLAYBACK ON CAS1
PLAYBACK READY
Y
N
PRINT
CAS 1 TO LOADPT
PLSTAT←71
PLON2
PRINT
PLAYBACK ON UNIT2
PLSTAT ← 65
N
PBUF EMPTY
Y
RECORD ON
N
DELAY 1SEC
R


SIHD
XRINT
MSGREC
R
XWINT
SENDING PREKEY
N
Y
SHOULD SEND ANOTHER PREKEY
Y
R
N
SPREKEY ← 0
SPREAM ← 1
SET UP TO SEND PERAMBLE
X1
SENDING PEAMBLE
N
Y
MORE PREAMBLE TO SEND
Y
XNEXT ← NEXT CHAR OUT
R
N
X2
ANY MORE DATA TO SEND
N
XNEXT ← -1
R
DSOUT
XNEXT ← X'FF'
LDSYNR ← FF
WRCHAR ← FF
CNTRL ← INTEN, DTR, XMTON
SPREKEY ← 1
PKNWDS ← KNPK
GET POINTER TO INPUT BUFFER AND SEND IT OVER DS
R


SIHK
PROM
STARK
MORE DUMMIES
N
PRINT-ED 72 CHAR LINE
Y
GENERATE CRLF
N
LAST CHAR PRIN-TABLE
Y
SET UP DUMMY CHAR FOR DELAY
N
POS NO WORDS IN KGUF
N
H
Y
BUFFER EMPTY
N
WINT1
GET NEXT CHAR
OUTIT
PRINT THE CHAR
R
XITP
PACT2 ← 0
STARTR WINT


SIHO
KWRITE
MORE DUMMIES
Y
N
PRINTED 72 CHAR LINE
Y
GENERATE CRLF
N
LAST CHAR PRINTA-BLE
Y
SET UP DUMMY CHAR FOR DELAY
N
ANY WORDS IN BUFFER
N
H
Y
OUTIT
PRINT CHAR
R
BUFFER EMPTY
N
WINT1
TAKE 1 CHAR FROM BUFFER
Y
XITP
PACT1 ← 0
R
CPRINT
NPRINT
OPRINT
POS NO WORDS TO STORE
N
R
Y
ROOM IN BUFFER
N
BUSPR
BUSY RETURN
Y
PRINT1
STORE DATA IN BUFFER
AUX PRINTER ACTIVE
Y
R
N
PRINT A CHAR
PACT1 ← 1
R


INS
INITIO
ENTER WORKER MODE INTERRUPTS MASKED
CLEAR HARDWARE INT BITS
SET UP INTERRUPT TRAP VECTORS
INITIALIZE SYSTEM VARIABLES
START 1MS CLOCK TIMER
SYNCHR ← FF
TIMER INT
Y
H
N
MAIN TTY INT
Y
H
N
AUX TTY INT
Y
H
DS INTERRUPT
Y
H
N
ENABLE INTERRUPTS THIS MUST BE DONE TO ENABLE INTERRUPTS ON MODEM BOARD
CNTRL ← INTON
ENTER SUPERVISOR MODE
R

baud rate. Therefore, to print at our 1200 baud data transfer rate, we must send the character to be printed followed by three dummy non-printing characters (X'007F'). This scheme gives us an effective printing rate of 300 baud, the maximum rate at which the print unit will function.

F. T.I. 733 TERMINAL

The TI733 is a silent terminal which contains a dual cassette drive and a keyboard/printer unit. It is referred to in the documentation as the TTY. The system calls for two 733 units in each station. These units are referred to as the TTY (or main TTY in slot EF0) and the AUX TTY (Auxilliary TTY in slot EF5).

The system has two stations with two terminals in each station. One terminal is a 300 baud unit while the other three are 1200 baud units. The baud rate refers to the data transfer rate of the unit and becomes significant when an excessive amount of cassette recording is done. Therefore, it is recommended that the 300 baud TTY be used as the airborne station's AUX TTY as this unit has minimal I/O transfer and is normally not used in data reduction. Note that the 300 and 1200 baud terminals have different cabling.

The terminal has two separate devices (keyboard/printer and playback/record) controlled by one I/O channel. Therefore, at any time there can be input from or output to only one device. That is, you may accept input from either the keyboard or the playback unit; you may output to either the printer or the record unit.

Although cassette functions take place at the 1200 baud rate, the printer unit is only capable of printing characters at a 300 baud rate.

CABLE DIAGRAM

From TI 960 full duplex synchronous communication module (TI NO 966755)

to McDonnell Douglas Data Link ground modem (Model MDL 310, MSK)

TI EDGE CONNECTOR PIN NO.	TI SIGNAL	DESCRIPTION	RS 232 SIGNAL	DB 25P PIN NO
A	SGRND	PROTECTIVE GROUND	AA	1
27	XMTDE	TRANSMITTED DATA	BA	2
28	RTSE	REQUEST TO SEND(T)	CA	4
22	CTSE	CLEAR TO SEND(T)	CB	5
		SIGNAL GROUND	AB	7
32	SCT	TRANSMIT CLOCK	DB	15
		PUSH TO TALK OR TRANSMIT/RECEIVE CONTROL	N/A	18
				DB25S PIN NO
		PROTECTIVE GROUND	AA	1
29	RCDE	RECEIVED DATA	BB	3
31	DSRE	REQUEST TO SEND(R)	CC	4
25	DTRE	CLEAR TO SEND(R)	CD	5
		SIGNAL GROUND	AB	7
26	SCR	DATA CLOCK(R)	DD	17

JUMPERS

Furthermore TI edge connector is jumpered as follows (C-3), (D-4), (E-5), (H-7), (K-9), (M-11) for options 1a, 2a, 3a, 4a, 5a, 6.

CABLE DIAGRAM

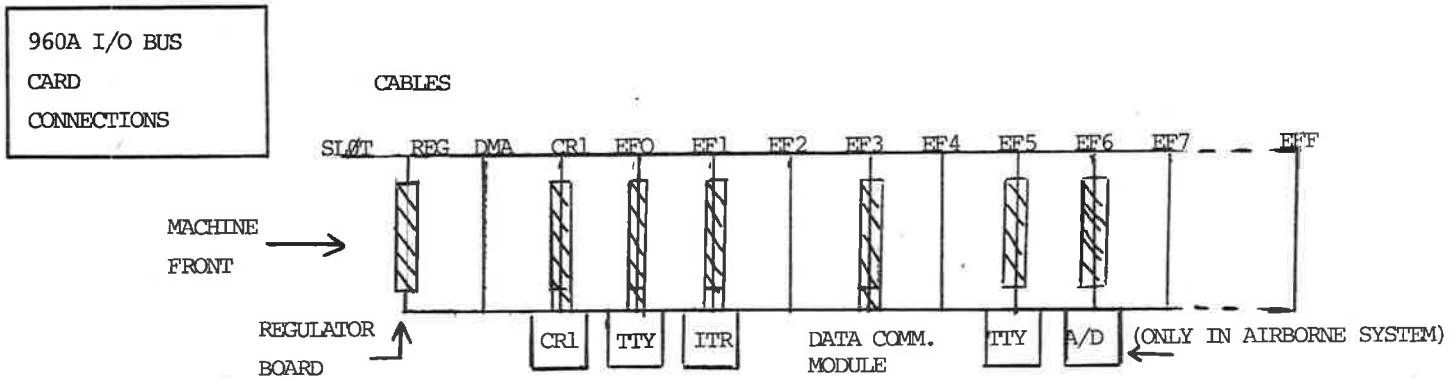
From TI 960 Full Duplex synchronous communication module (TI NO 966755) to ICC Milgo modem 24 synchronous data set

TI EDGE CONNECTOR PIN NO.	TI SIGNAL	DESCRIPTION	RS232 SIGNAL	DBM-25P PIN NO
22	CTSE	CLEAR TO SEND INPUT LINE FROM MODEM	CB	5
25	DTRE	DATA TERMINAL READY LINE TO MODEM	CD	20
26	SCR	SERIAL CLOCK RECEIVER LINE FROM MODEM	DD	17
27	XMTDE	TRANSMITTED DATA LINE TO MODEM	BA	2
28	RTSE	REQUEST TO SEND LINE TO MODEM	CA	4
29	RCVDE	RECEIVED DATA LINE FROM MODEM	BB	3
30	DCDE	DATA CARRIER DETECT LINE FROM MODEM	CF	8
31	DSRE	DATA SET READY LINE FROM MODEM	CC	6
32	SCT	SERIAL CLOCK TRANSMIT FROM MODEM	DB	15
A	SGRND	SYSTEM GROUND	AA	1
$\overline{R}$		SIGNAL GROUND	AB	7

Furthermore TI edge connector is jumpered as follows (C-3), (D-4), (E-5), (H-7), (K-9), (M-11) for options 1a, 2a, 3a, 4a, 5a, 6.

TI EDGE CONNECTOR JUMPERS

PINS		OPTION
C-3	INTERNAL +15V	1a
D-4	INTERNAL -15V	2a
E-5	TRANSMIT CLOCK LINE SYNCHRONIZATION	3a
H-7	TRUE SENSE RCVD DATA	4a
K-9	TRUE SENSE TRANSMITTED DATA	5a
M-11	PARITY DISABLED	6



1) ALL COMPONENTS ON PC BOARDS FACE FRONT

2) TTY BOARDS ARE JUMPERED E8-E10 FOR 300 BAUD
E8-E9 FOR 1200 BAUD

3) 1200 BAUD TTY MUST HAVE TI CABLE NO 959372-2

4) 300 BAUD TTY MUST HAVE TI CABLE NO 959372-1

5) INTERFACE CARDS

CR1 CRU INTERFACE CARD TI 226806-0001

TTY FULL DUPLEX TTY/EYA MODULE TI 961642-000X

ITR INTERVAL TIMER TI 214113-001

A/D A/D CONVERTER TI 214103-000X

SYNCHRONOUS COMMUNICATION MODULE TI 966752-0001

VII. DATA REDUCTION AND ANALYSIS

This section describes the function and operation of the two data analysis programs. The first program is commonly referred to as the Data Reduction program and the second as the Graph program. The first provides the capability of printing every record of the ground and airborne experimental cassettes. In addition, a matrix of statistical information is provided as a run summary. The second program provides a time versus event graphical representation of significant occurrences during an experimental run. It also produces a statistical summarization.

Data Reduction (DR) is composed of four assembly modules (MAIN, FIN, PRINRE and UTIL) that form a stand alone executable program when linked with the system interrupt handler (SIH) modules.

DR has two input data streams, each read from a cassette via a Silent 700 terminal. These data streams are recorded by the ground station and the air station respectively during an experimental run of the baseline system. Both data streams contain two types of information: transaction data and statistical data. The transaction data recorded by the ground station consists of a message transaction record (MTR; noting time, unusual conditions, message length, mode, etc.) for every message sent, and the actual text of any message with an error condition (BCS check, no response, NAK). For transaction data, the airborne station records an Uplink Transaction Record (UTR; containing aircraft attitude, time, message length, mode, etc.) for every successfully uplinked message, and an error record (containing the same information plus the received text) for every uplinked message received with an error condition. The last record recorded by each station is statistical information including, for each of four message length ranges, a count of occurrences of each of eight conditions (echo message with ACK, ..., non-echo message with BCS error); and, for echo type messages, a bit error count.

DR has two major tasks. The first is to read through the transaction data, correlating information from the air and ground

recordings, and printing it in a form indicating the sequence of events that occurred during the test. (How much of the transaction data is actually printed is controlled by print options entered via computer panel data switches.) (See Figure I for details of merging the air and ground transaction files.) While it passes through the transaction data, DR also counts the number of successful/unsuccessful transactions, and accumulates total transaction time for successful transactions. This task is handled by assembly modules MAIN (for merging the transaction files) and PRINRE (for formating and printing the transaction records), which both use utility routines from the module UTIL.

The second major task of DR is to use the accumulated statistical information for computing average transaction time for successful transactions, and the percentage of occurrence of various conditions. This task is handled by module FIN using utility routines from UTIL. An outline of these computations follows:

1) Average Transaction Time =

$$\frac{\text{total time for successful transactions}}{\text{number of successful transactions}}$$

2) The ground and air systems count each message as having exactly one of the following conditions: ACK, NAK, NO RESPONSE, BCS ERROR. Hence for each message length

Number of echo msgs = Number of echo ACK + No. of echo NAK + No. of echo no response + No. of echo BCR errors

Then

$$\text{Echo \% ACK} = \frac{\text{No. of echo ACK}}{\text{No. of echo msgs.}} \cdot 100,$$

and similarly for the other echo and non-echo conditions. Notice that echo % ACK + echo % NAK + echo % no response + echo % BCS ERROR = 100%, and similarly for the non-echo conditions

3) On echo type messages, the ground system compares the original copy with the echoed copy (text and preamble) and count is kept of the total number of bits echoed, and the total number echoed incorrectly.

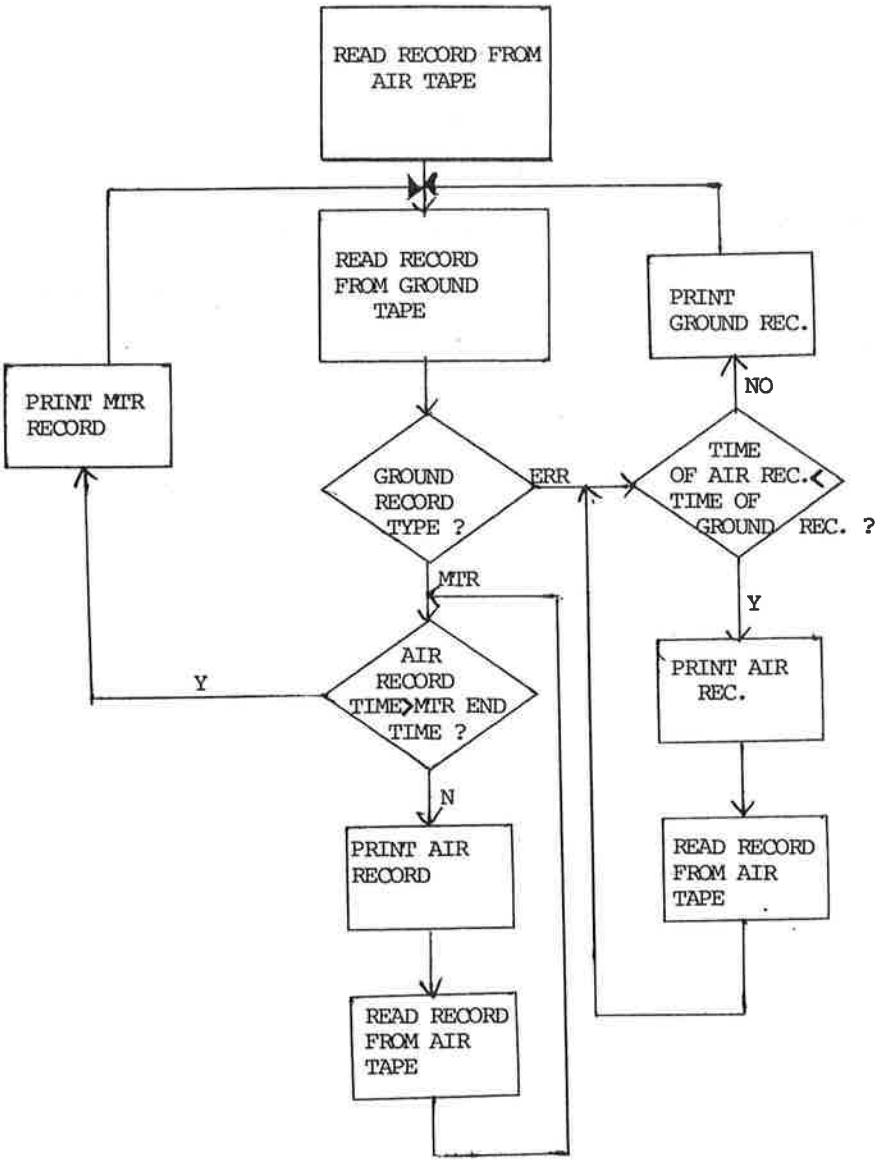
Then

$$\text{Echo bit error \%} = \frac{\text{No. bits echoed incorrectly}}{\text{No. bits echoed}} \cdot 100$$

FIGURE 1

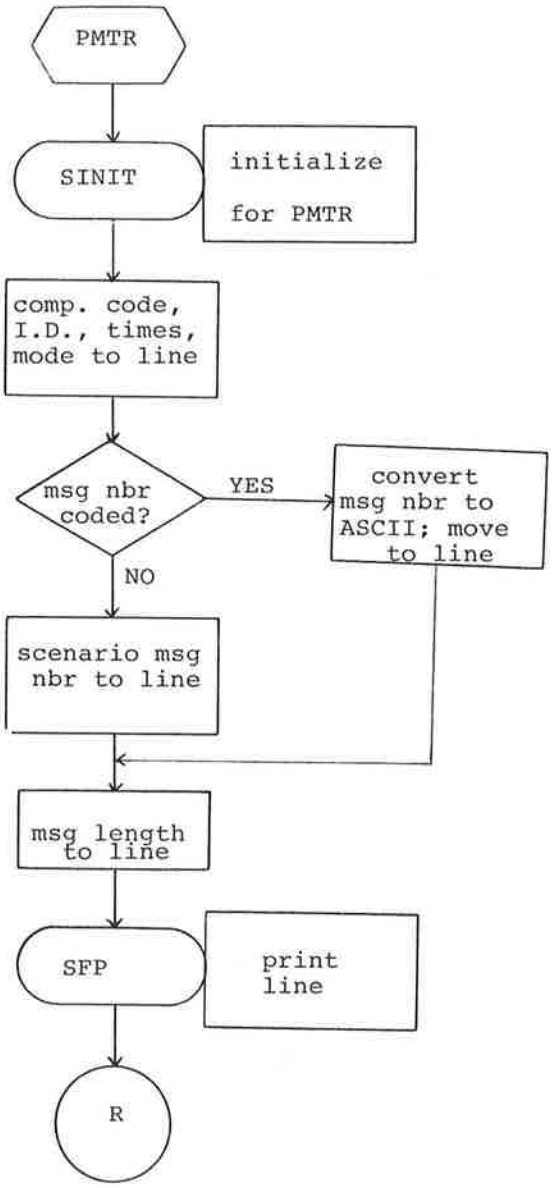
Merge of air and ground transaction files.

Error type transaction records contain the time the error is recognized, and are recorded in the order they occurred. Message transaction records contain the start time and end time for the transaction, and are recorded by the ground system after any error records pertaining to the transaction. The air and ground transaction files are merged with respect to these time fields using the following logic:


READ RECORD FROM AIR TAPE
READ RECORD FROM GROUND TAPE
PRINT GROUND REC.
NO
PRINT MTR RECORD
GROUND RECORD TYPE ?
ERR
TIME OF AIR REC. < TIME OF GROUND REC. ?
MTR
Y
AIR RECORD TIME > MTR END TIME ?
Y
PRINT AIR REC.
N
PRINT AIR RECORD
READ RECORD FROM AIR TAPE
READ RECORD FROM AIR TAPE

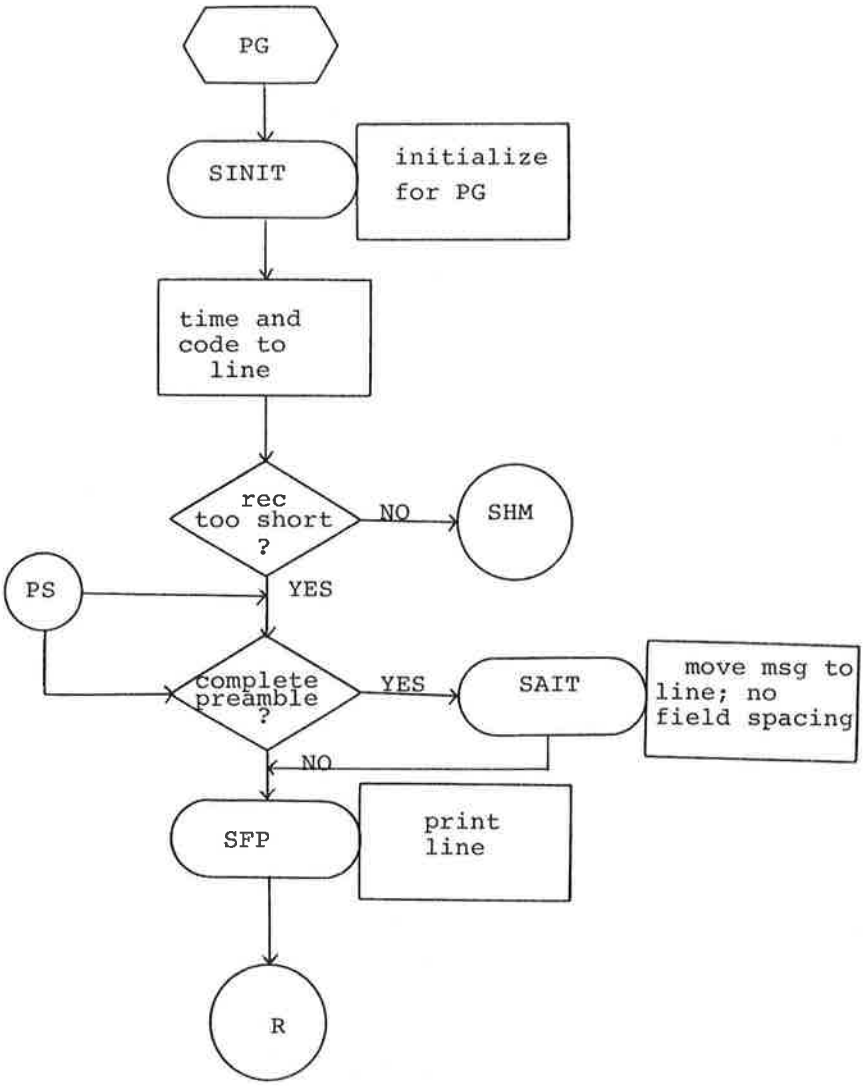
PMTR

Print Msg Transaction Record from MTR Dseg



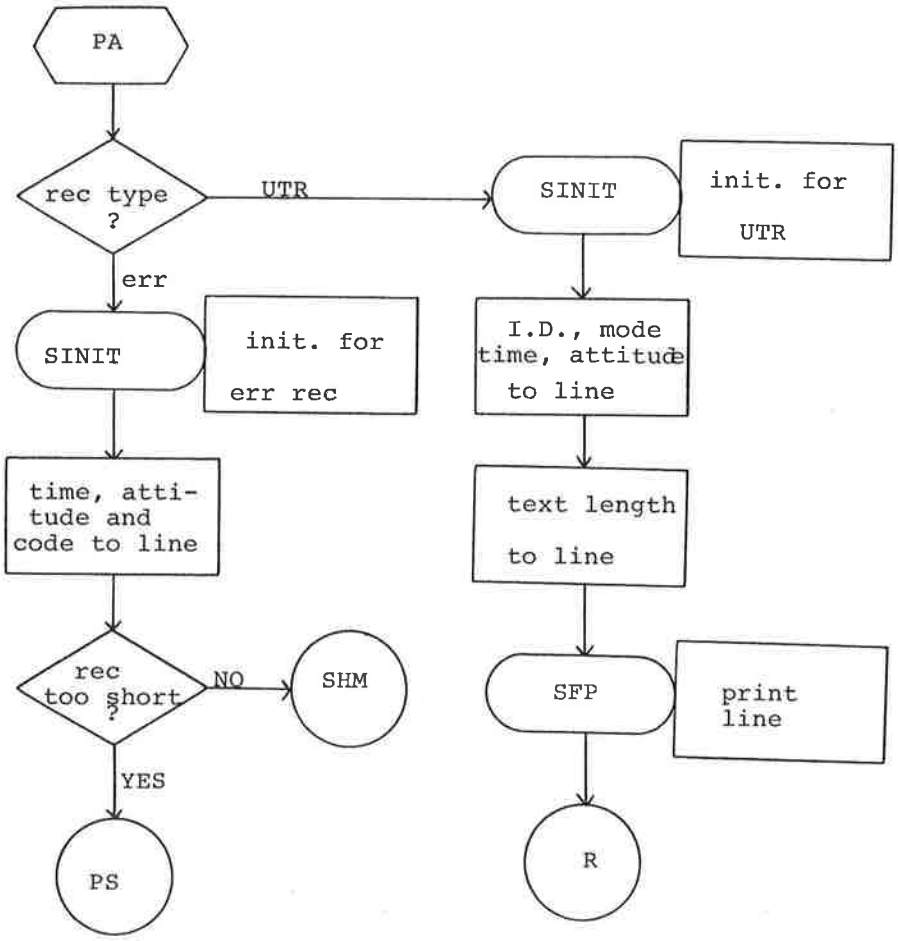
PG

Print grnd err record from GREC Dseg



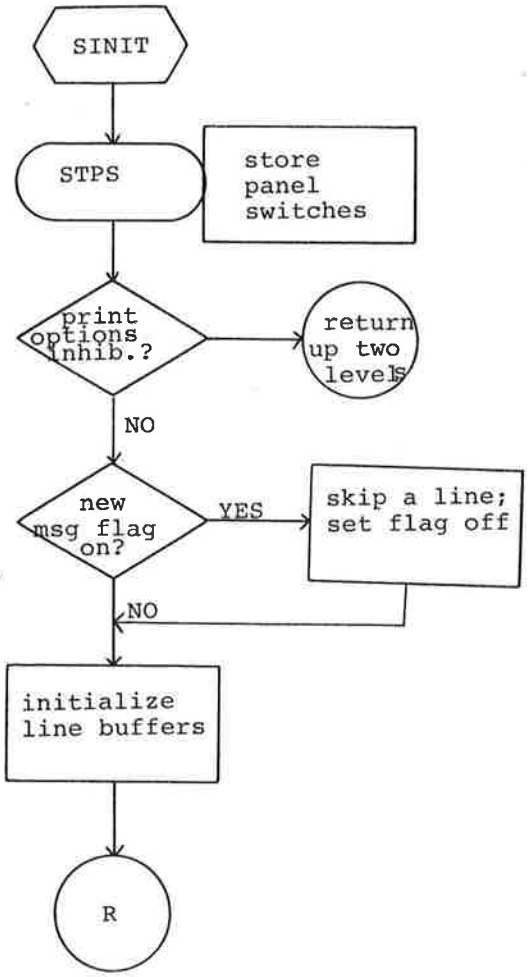
PA

Print air err record or UTR from AREC Dseg



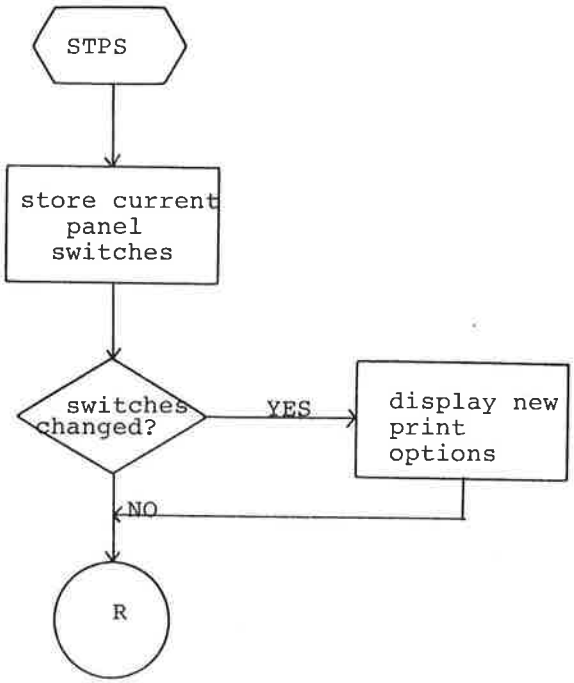
SINIT

Initialize for PMTR, PA, PG



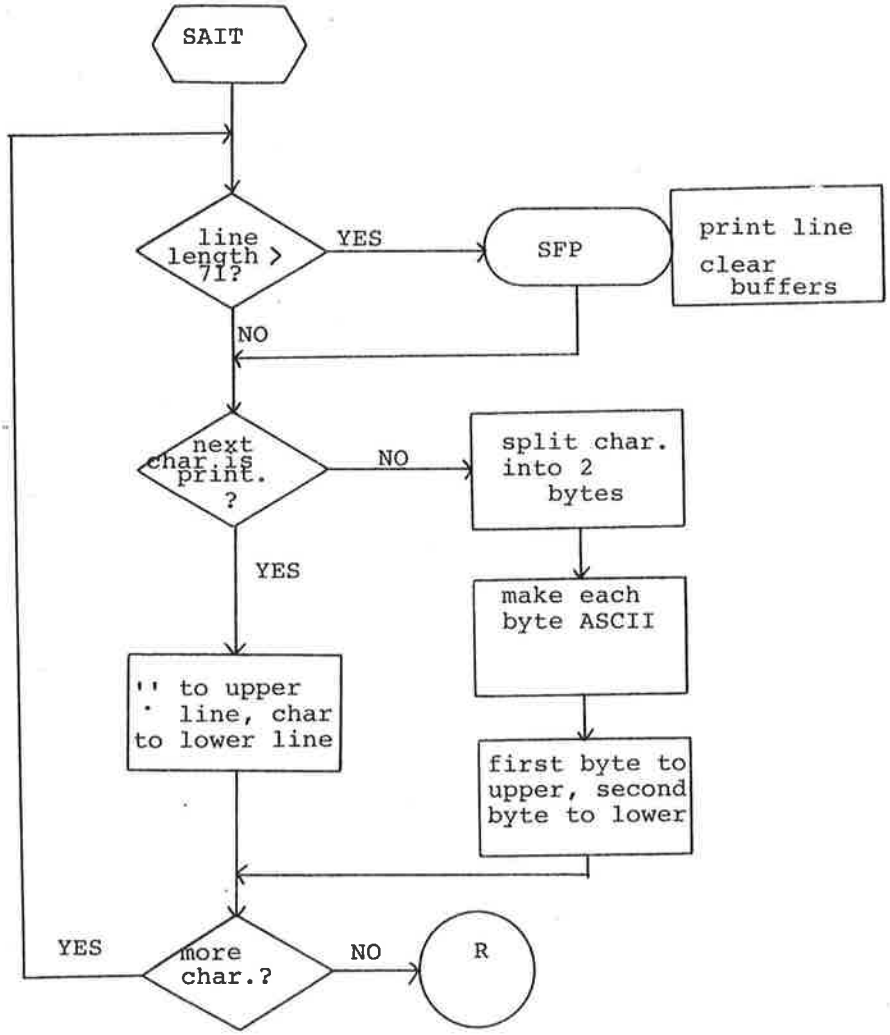
STPS

Get panel switches, display print options



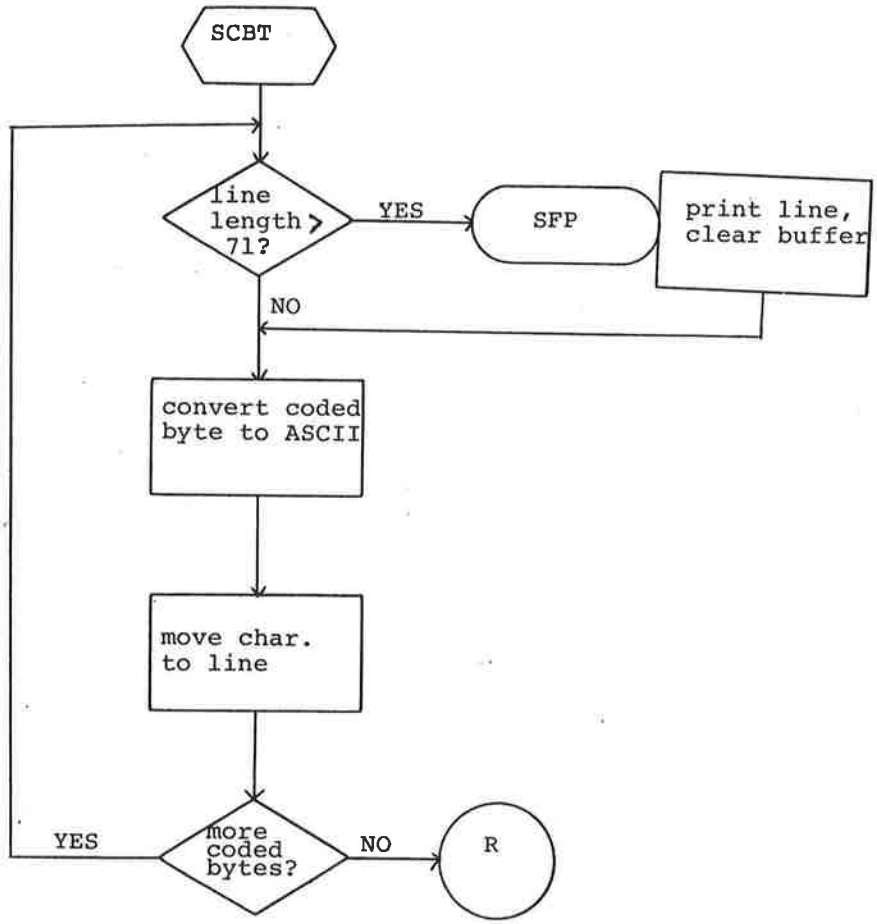
SAIT

move to line in as is, double line format



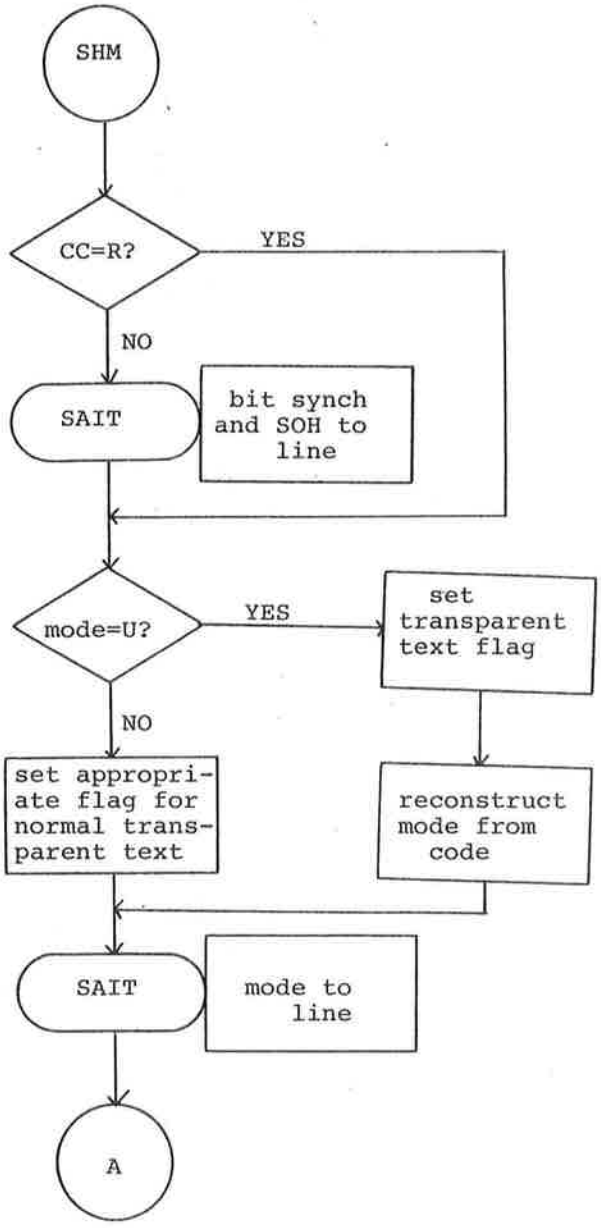
SCBT

move coded binary to line in single line format

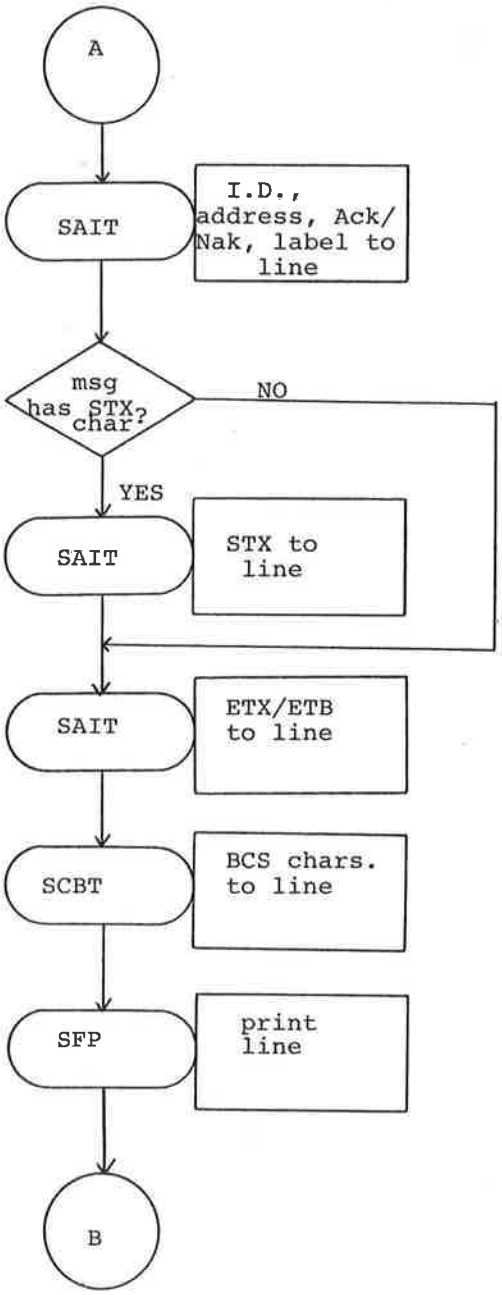


SHM

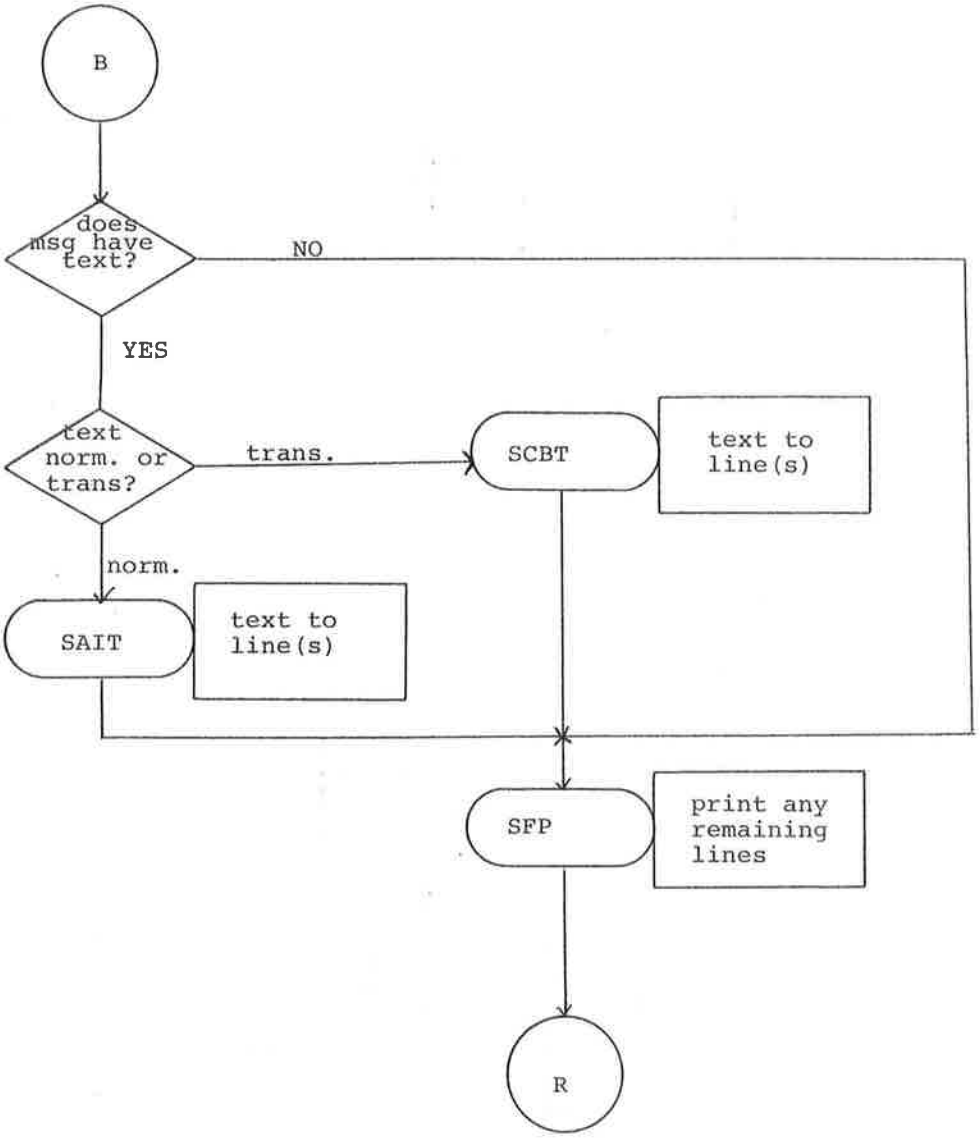
Format error msg line for printing. Used by PA and PG



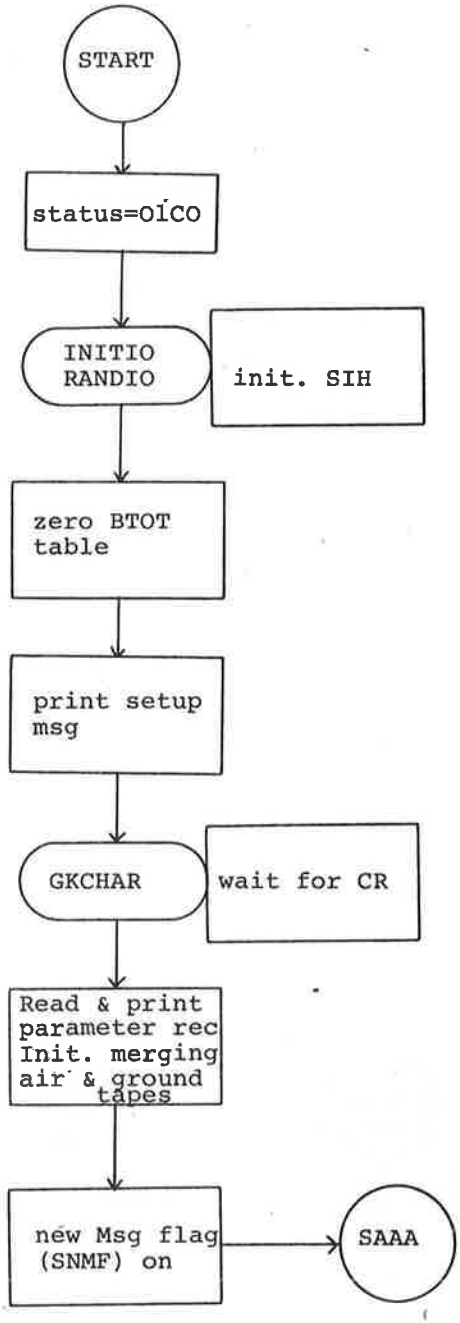
SHM continued



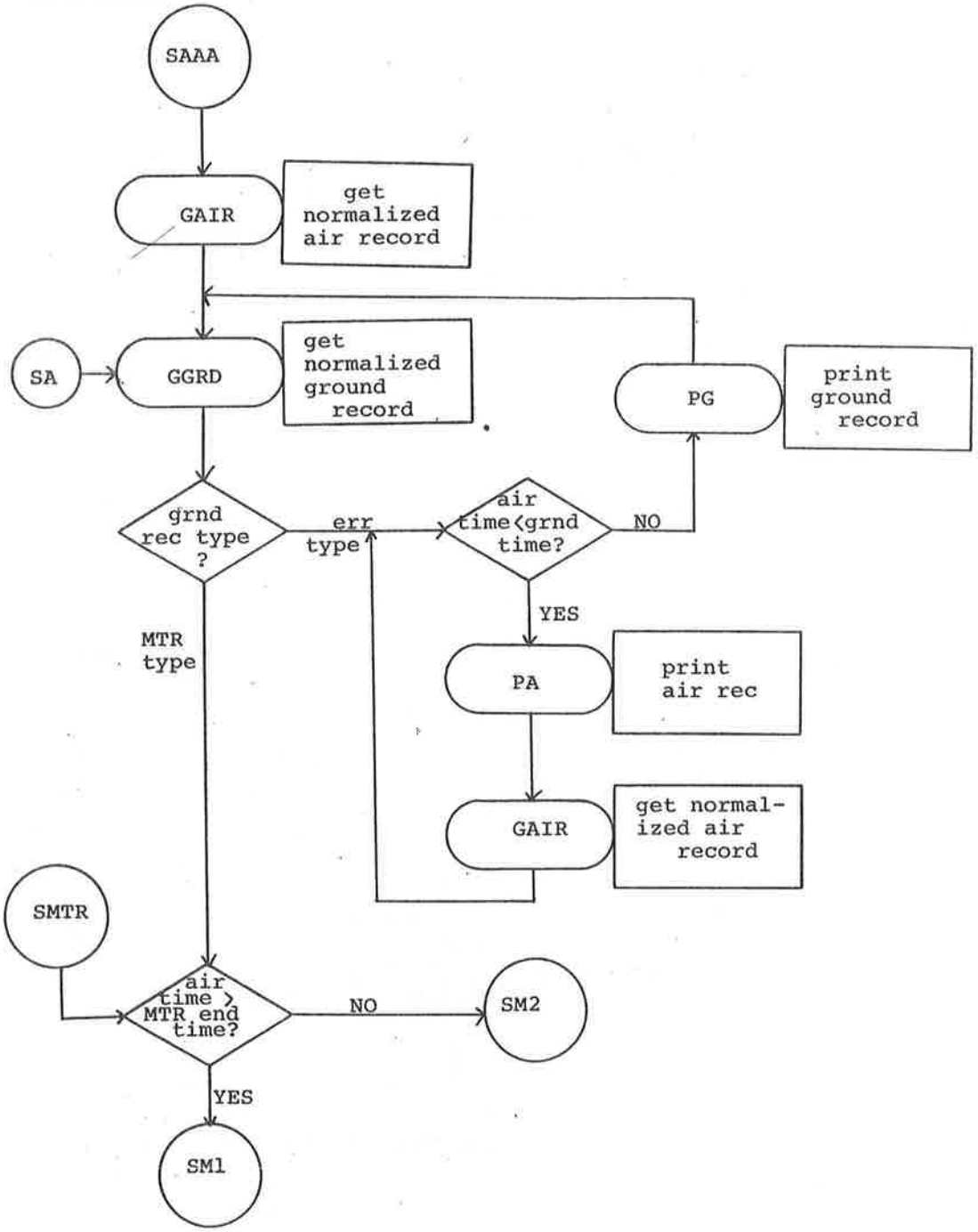
SHM continued



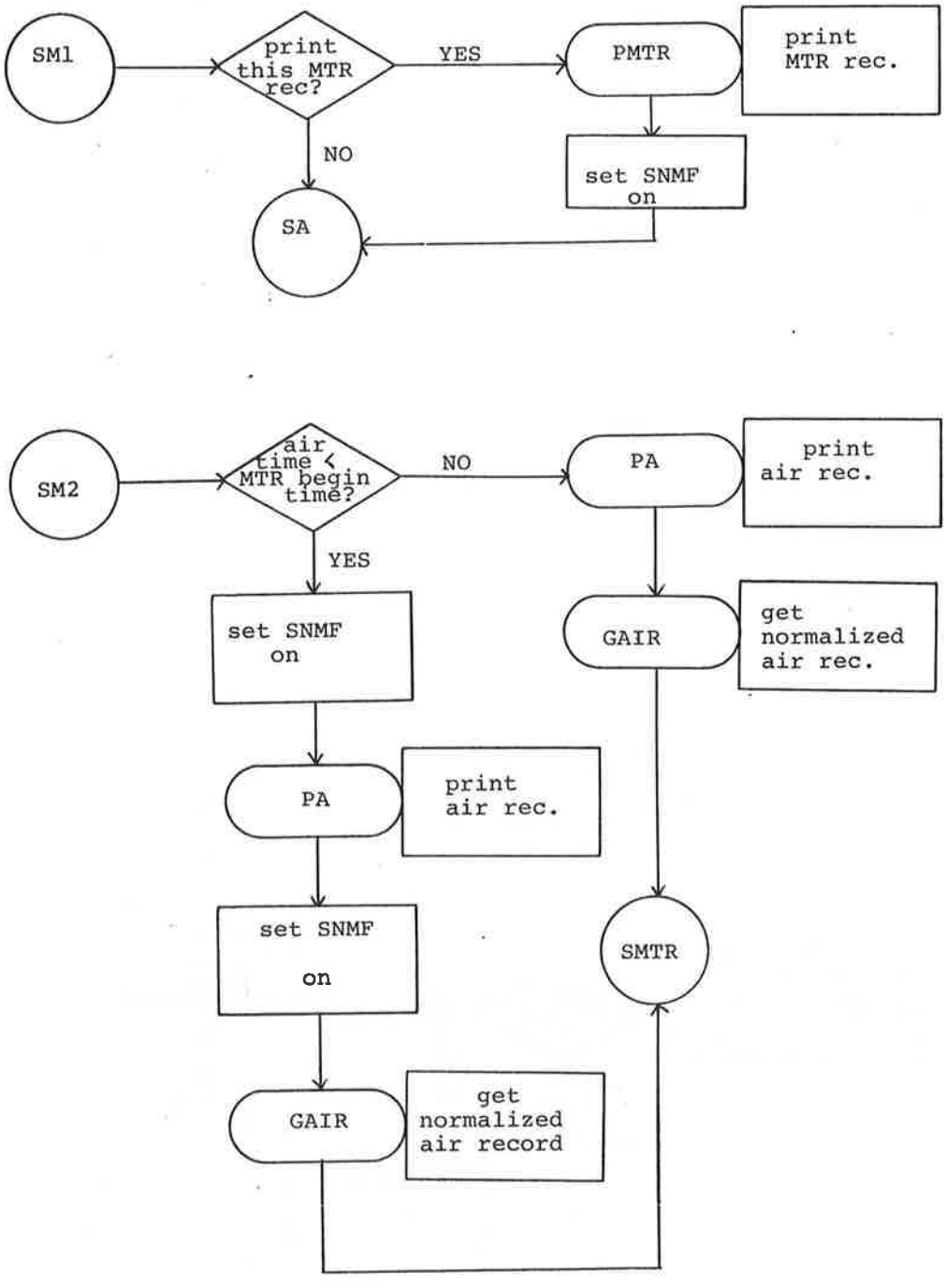
Main Sequence Logic Initialize



Main Sequence Logic Process transactions

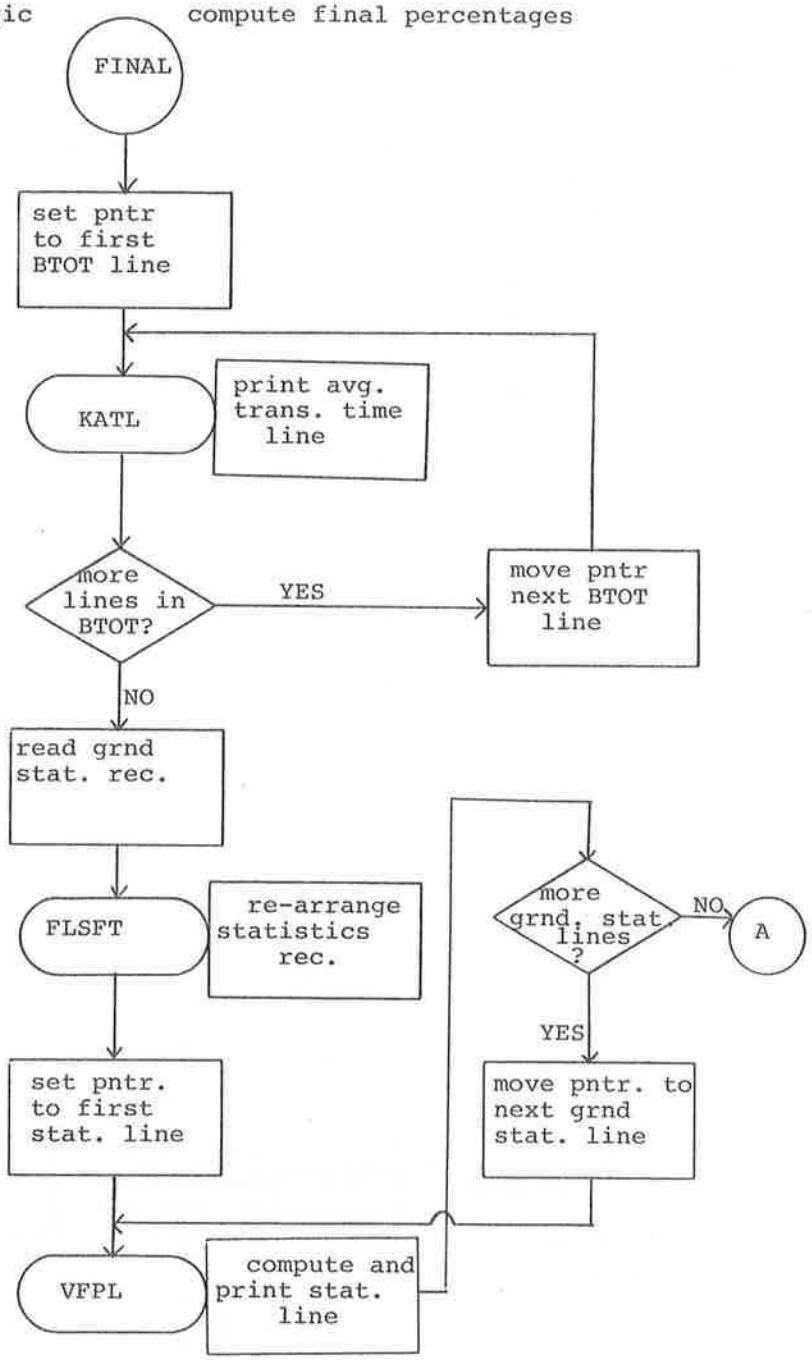


Main Sequence Logic Process Transactions (cont.)

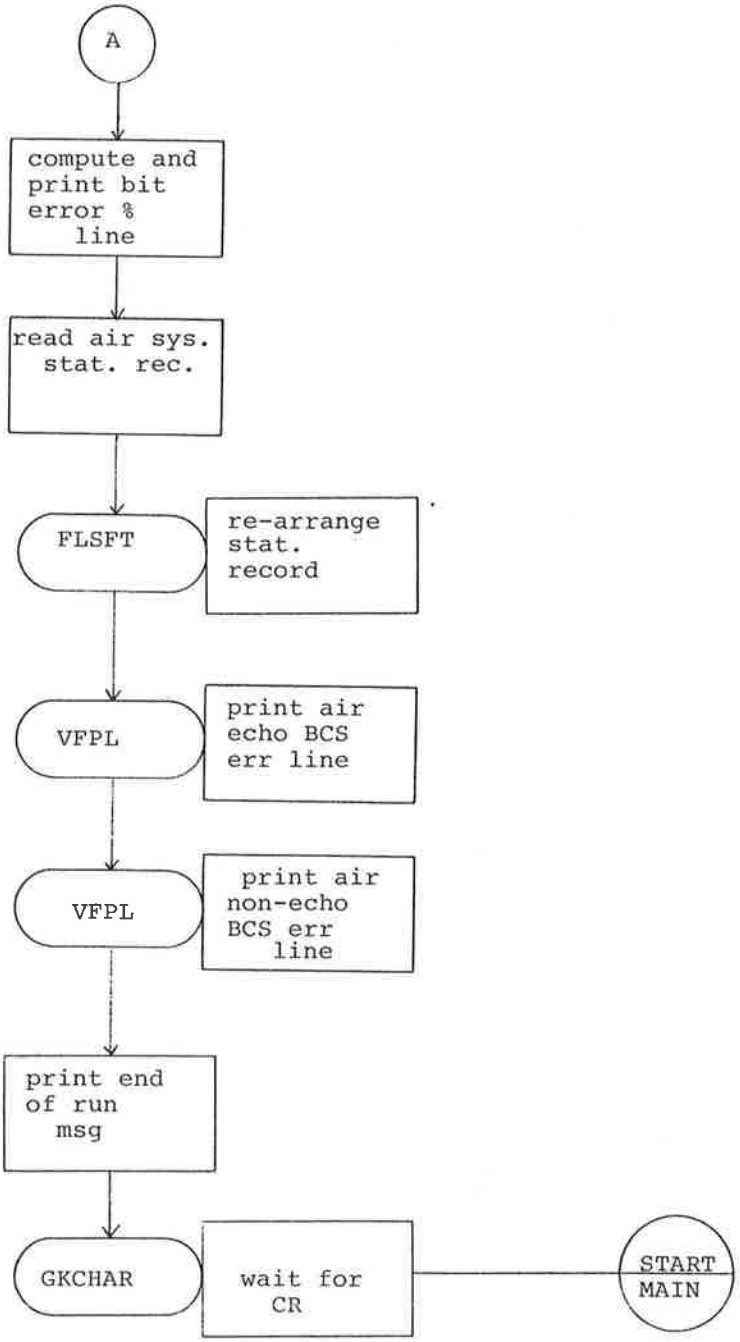


Main Sequence Logic

compute final percentages

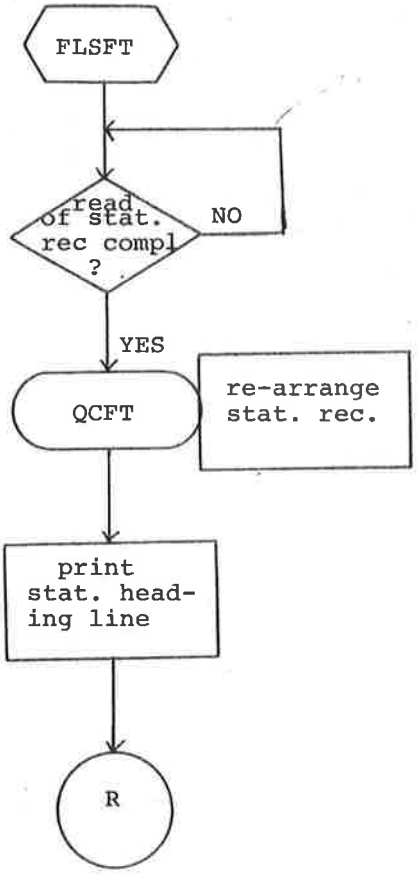


Main Sequence Logic Compute final percentages FIN



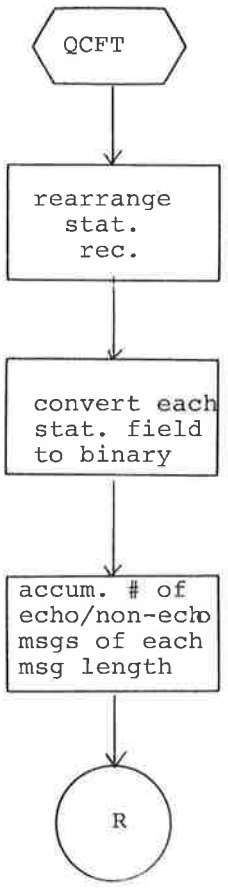
FLSFT

Massage statistics record, create and print statistics header lines



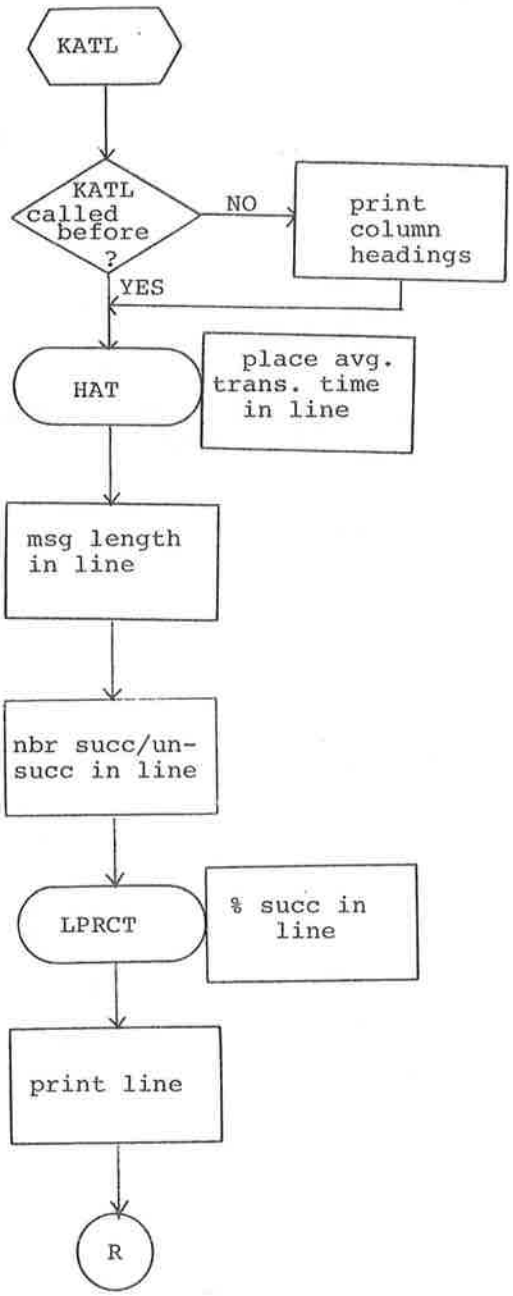
QCFT

Rearrange Statistical record and convert it to binary
Accumulate echo/non-echo totals for each msg length



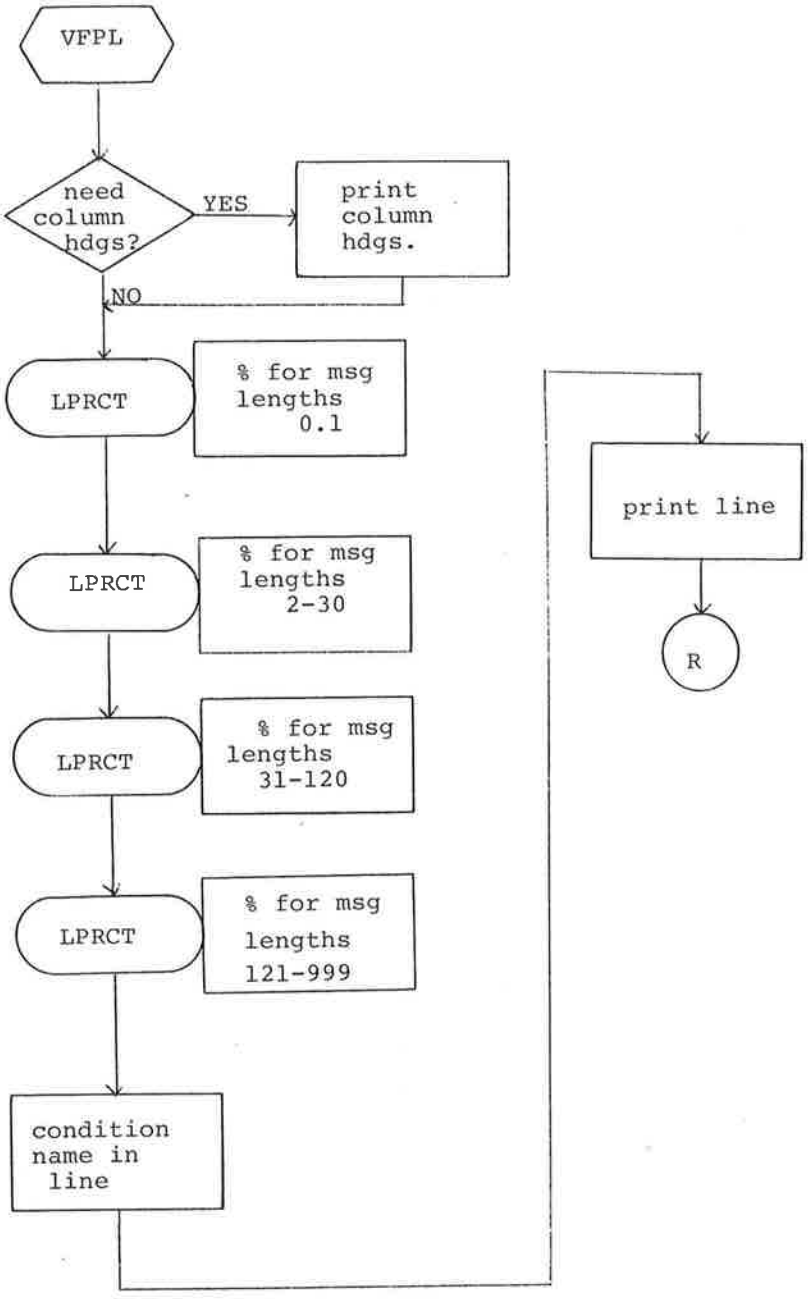
KATL

Compute and print average trans. time/ % successful line



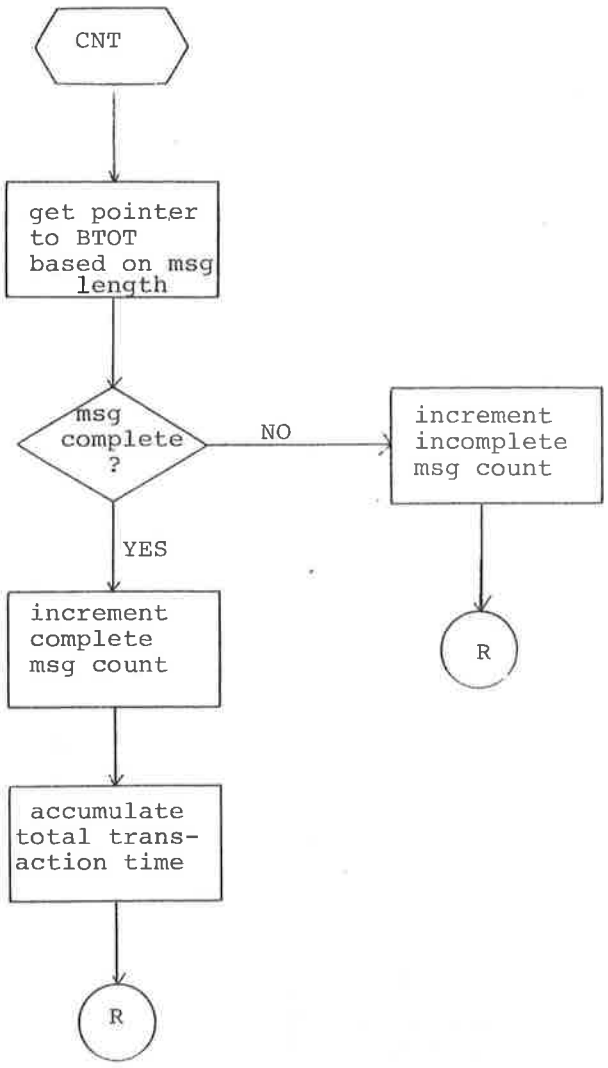
VFPL

Create and print condition occurrence line



CNT

Accumulate statistics from MTR records



GRAPH is a stand alone program which is designed to be used in conjunction with the Data Reduction Program. GRAPH accepts the same input tapes, but generates a graphic display of the system performance. It accepts one input tape and, therefore, can be run with a single ASR-733 terminal.

GRAPH was designed to provide:

1) A graphical representation of all significant system events, associating them with time and other parameters.
2) Two separate graphs for each experiment. Each graph will be representative of the frame of reference from which it originated (ground/air station).
3) General statistics which can be used as a standard by which all experiments can be compared.
4) Sufficient information so that every message transaction can be traced from origination to conclusion.

The graphic output consists of a family of square waves printed in a vertical direction. The wave length is calibrated in units of time while the height is either high (In Communications), or low (Out Of Communications). Since this wave form is representative of a (ground/air) station's frame of reference, the air and ground graphs will be of slightly different waves. That is, at some point in time the ground station will decide that it has a no-response condition and its graph will go low. At the same point in time, the air station may have responded to the last ground communication, and its graph will still be high.

Thus, a graph will exhibit a square wave form which moves from high to low (and vice versa) values to indicate its present state of communications.

The distance of the wave crest (or trough) is a semi-linear function and is computed as follows. Each wave crest consists of at least two lines. One line displays the starting message transaction number, and the other displays the final message transaction number. Each minute of wave duration (the difference in real time between the starting and final message transactions) then causes an additional line of '*' to be printed. Therefore, for any large number of minutes duration, this scaling method approaches linearity. All significant events which occur during the experiment are displayed. A significant event is considered to be one of the following:

1) Any error message transaction (any non-complete message transaction).

2) The last message completion transaction (MCT) before the error message.

3) The first MCT after the error message.

4) The first MCT of the experiment.

5) The last MCT of the experiment.

Furthermore, all significant events are associated with a 'Time

of Day', an 'Elapsed Time', and relevant message transaction information. The Time of Day is the time at which the event is recognized and recorded by the experimental Data Link System. Elapsed Time is the difference in time between the Time of Day being printed and the previously printed Time of Day. Message transaction information is slightly difference for air and ground graphs. The ground heading is:

'MSG# MODE SCEN#',

while the air heading is:

'MSG# MODE NAK ATTITUDE'.

The "MSG#' is the four ASCII digits and one alpha character which are assigned to the message transaction by the ground system at transmission time. The 'MODE' is the one character mode associated with each message.

The 'SCEN#' is the number of the scenario text which was imbedded in the record. (01-16, K=keyboard message).

The air graph has 'NAK' and 'ATTITUDE' columns. The NAK column contains an 'N', if the message contained a NAK. The attitude information, represented as three 12 bit words is taken directly from the aircraft A/D converters at the time when the message is recorded.

The troughs of the square wave are indicators of error messages. The first line of each trough has associated with it a five character message transaction number and a one character error code (B, I, W, L, N). The air graph trough also has a mode associated with it. The ground graph displays any non-MCT record as an error, while the air graph allows N errors to be treated as MCT's, since the receipt of a NAK in a message only implies something wrong with the previous message transmission.

GRAPH STATISTICS

There are three time calculations which indicate how station time was allocated between processing MCT's and error transmissions. The 'DURATION' of the experiment is the elapsed time from the first MCT until the last MCT. It is a measure of the time that the station was in the communications loop. Graphically, it is the elapsed time from the initial wave high until the final wave low.

'OUT OF COM' is the sum of the elapsed times of all wave troughs. It is a measure of how much time the station spent in processing non-productive error messages.

'IN COM' equals DURATION-OUT OF COM. It is the sum of the elapsed times of all wave crests. It measures how much time the station spent in processing productive MCT's.

There are four row headings in the remaining statistics. The headings are self-explanatory. 'ALL MSGS' are divided into three mutually exclusive groups. '#MC MSGS' represent MCT's. '#NAK MSGS' represent all other messages (errors).

The three column headings are also self-explanatory. 'TOTAL' represents the actual number for each row. '%ALL MSGS' and 'MSGS/HOUR' are also calculated for each row.

The ground graph has an additional row. '#GIVE UPS' represents the number of times that a selected message was not successfully

transmitted, but was given up on because it incurred more than the allowable parameter number of error messages. This category is a subset of the "BAD MSGS" row.

GRAPH is composed of three assembly modules (Grapxx, Mathxx, and Gradxx) that form a stand alone executable program when linked with the System Interrupt Handler. Graph is a load and go program which will automatically self start after loading.

It prints its initial dialogue:

'READY INPUT TAPE AND
TYPE 'RETURN' OR
TYPE 'S' (STATISTICS ONLY)
WHEN READY'.

The operator must bring the input tape to the ready position on the playback cassette unit of the ASR-733 terminal. He may then type a carriage return to enable the graphic printout and statistics output. Or, he may type an 'S' to only enable the statistics output.

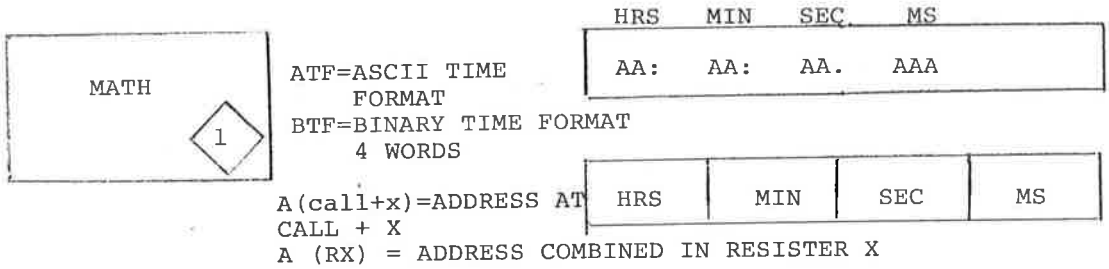
If a hardware read error occurs during the program execution, GRAPH will print,

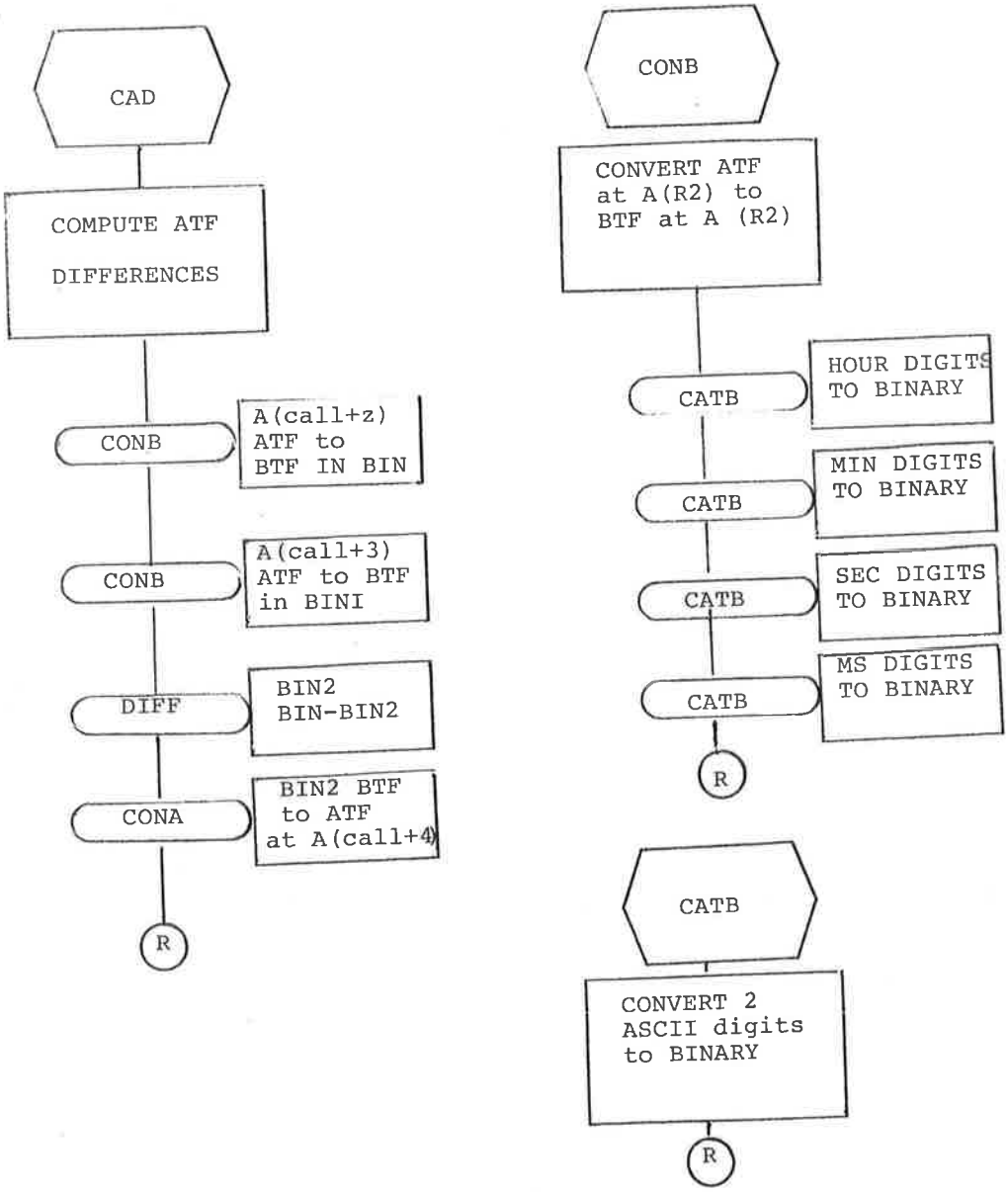
'PLAY ERROR',

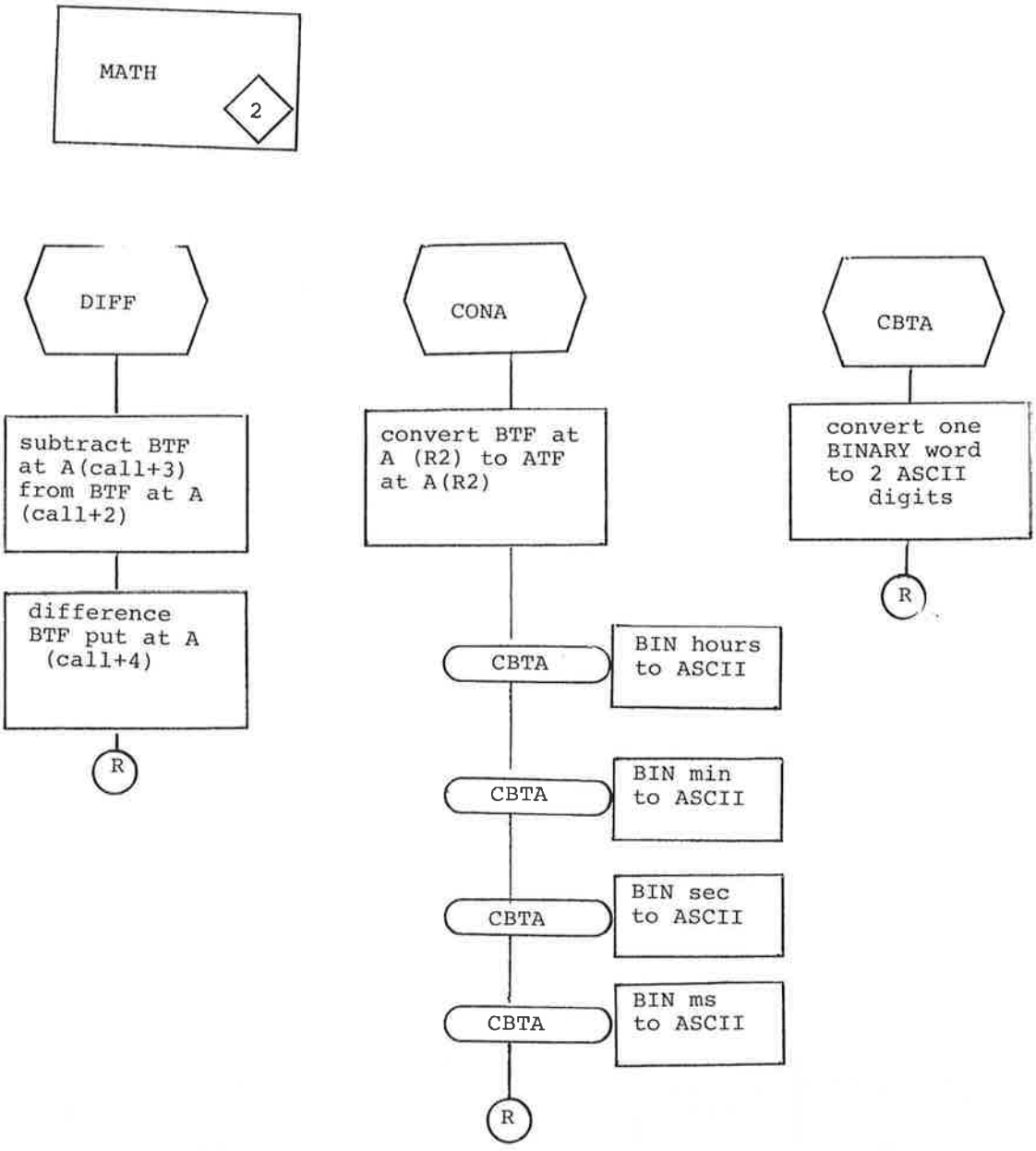
and the program will automatically restart. The operator must then manually turn the playback unit. The operator then responds to the dialogue as above.

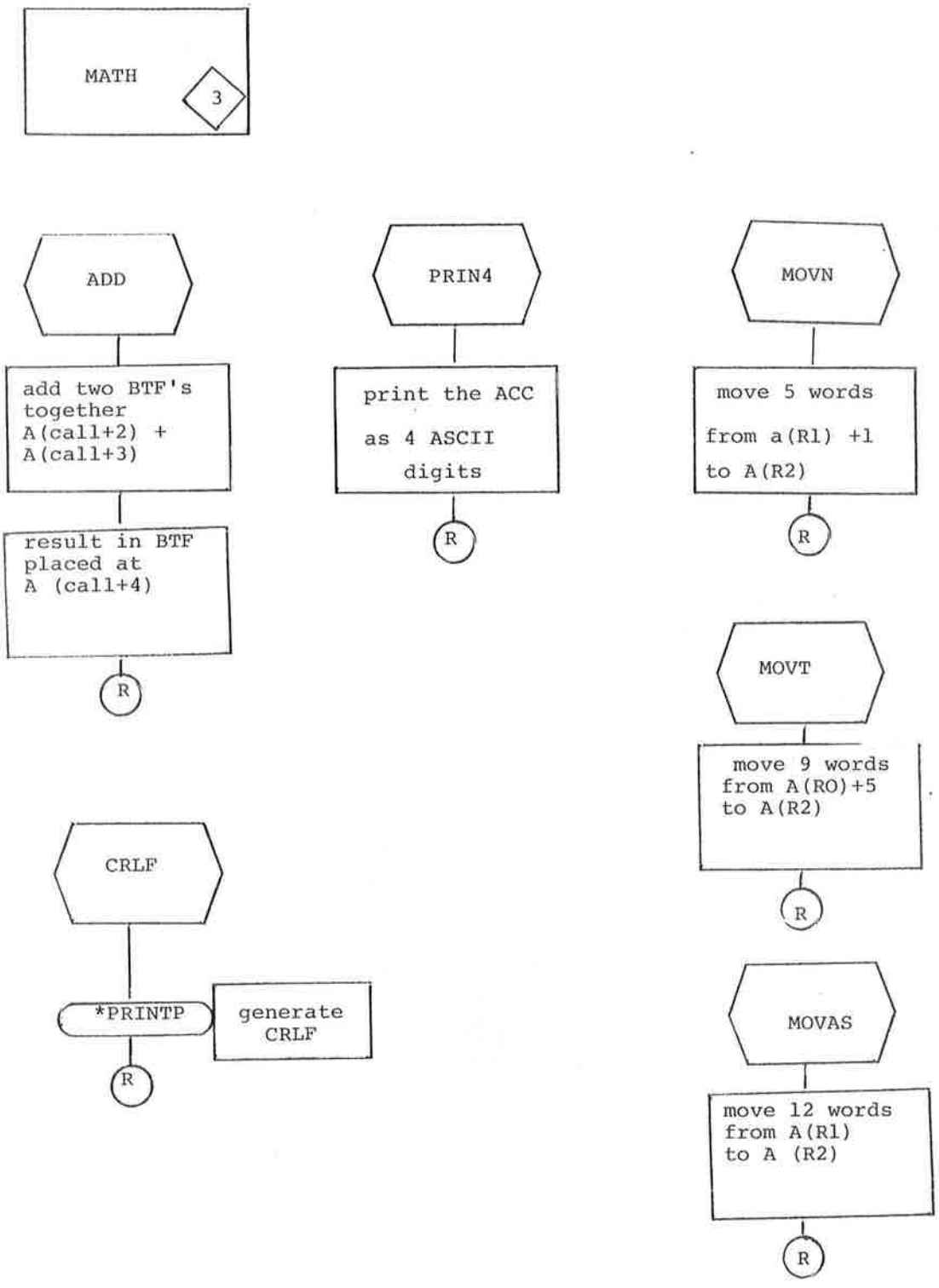
It should be noted that such any error condition is a result of misalignment between the tape cassette and the playback unit read head. If the misalignment is slight, the tape may be read correctly during the next attempted read. However, it would perhaps be better to switch cassette units.

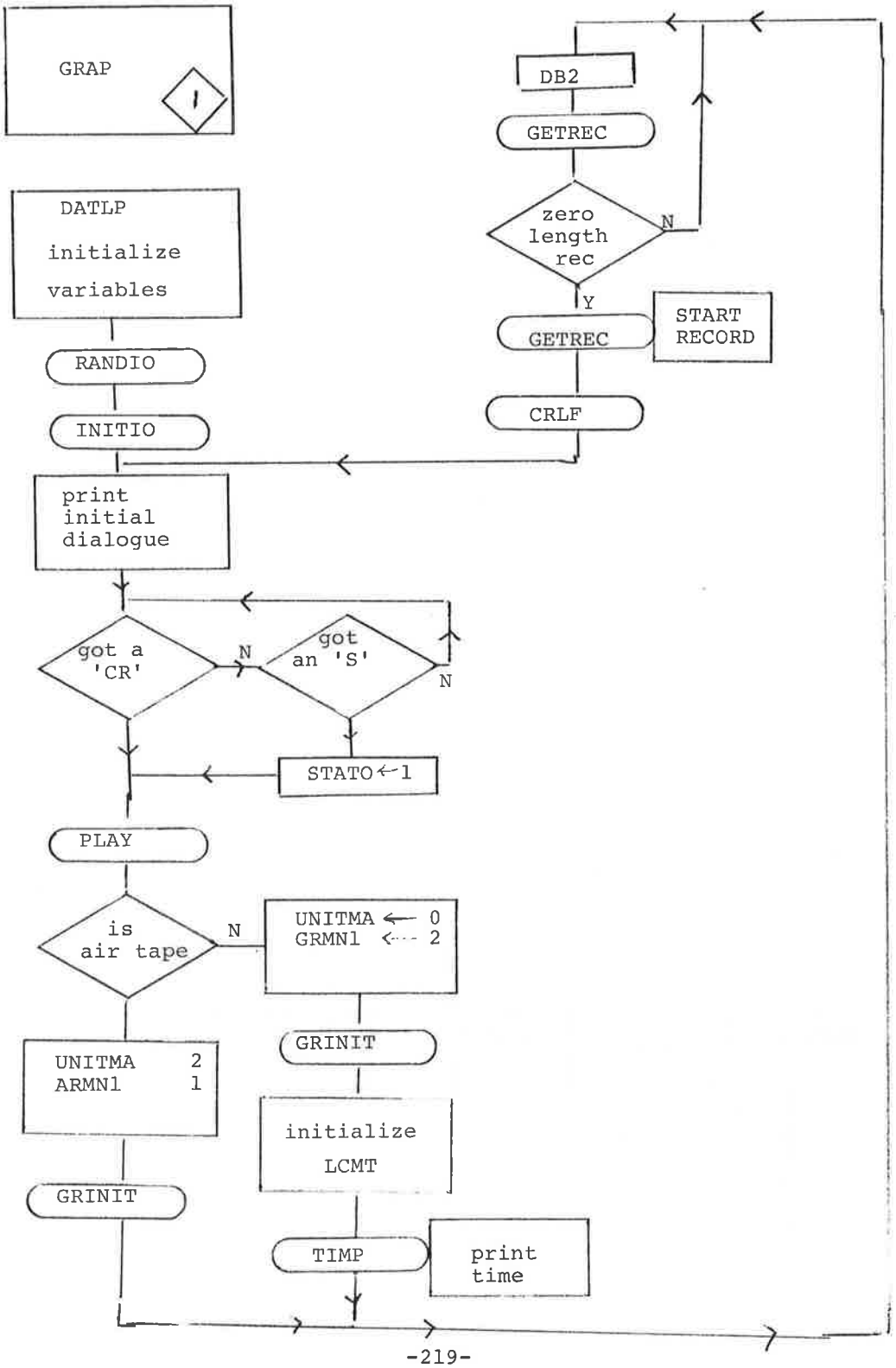
The program requires a 960A Computer, an Interval Timer Board (Slot EF1) and an ASR-733 terminal (Slot EF0).

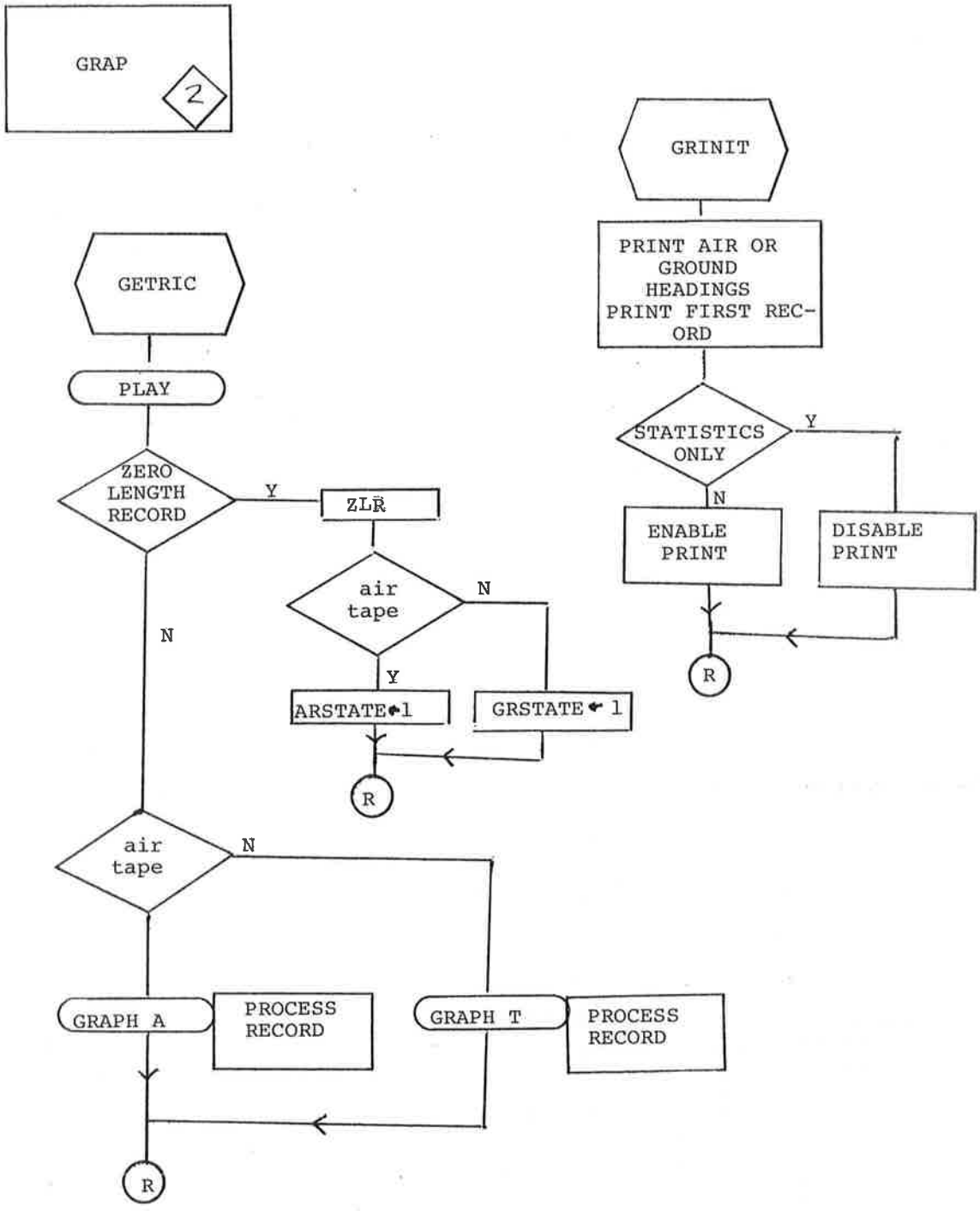

MATH
1
ATF=ASCII TIME FORMAT
BTF=BINARY TIME FORMAT
4 WORDS
A(call+x)=ADDRESS AT
CALL + X
A (RX) = ADDRESS COMBINED IN RESISTER X
HRS MIN SEC. MS
AA: AA: AA. AAA
HRS
MIN
SEC
MS

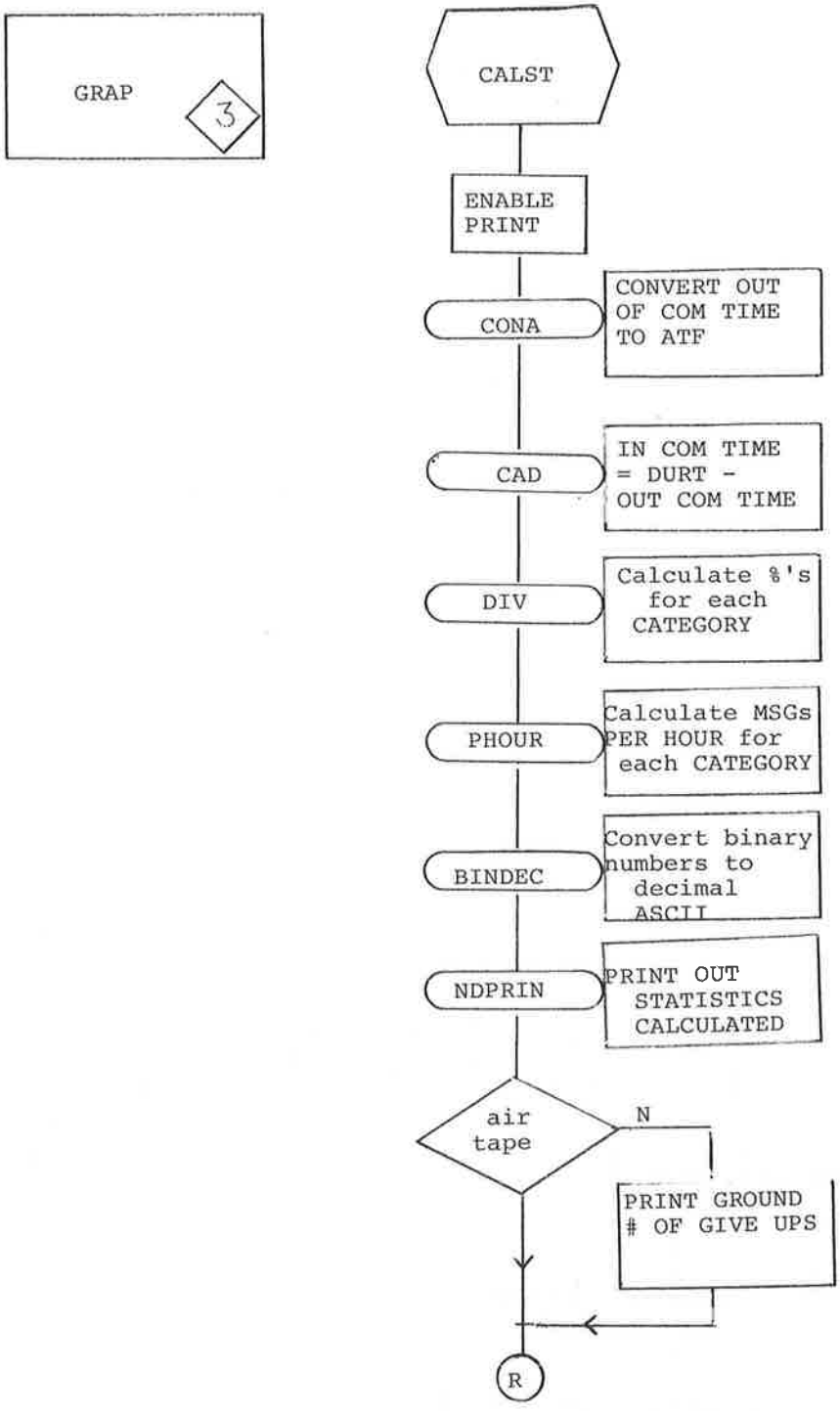

CAD
COMPUTE ATF DIFFERENCES
CONB
A(call+z) ATF to BTF IN BIN
CONB
A(call+3) ATF to BTF in BINI
DIFF
BIN2 BIN-BIN2
CONA
BIN2 BTF to ATF at A(call+4)
R
CONB
CONVERT ATF at A(R2) to BTF at A (R2)
CATB
HOUR DIGITS TO BINARY
CATB
MIN DIGITS TO BINARY
CATB
SEC DIGITS TO BINARY
CATB
MS DIGITS TO BINARY
R
CATB
CONVERT 2 ASCII digits to BINARY
R

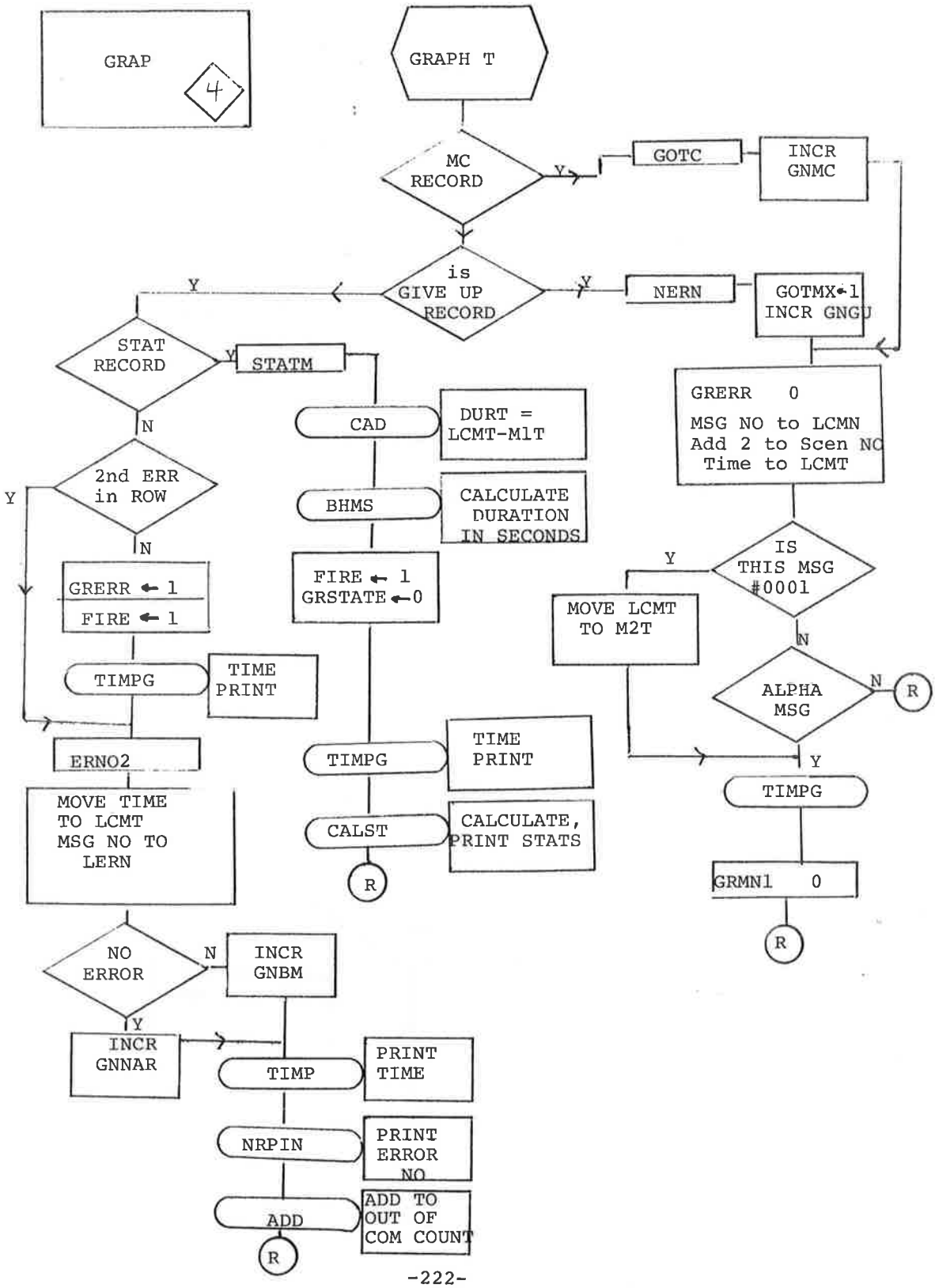

MATH
2
DIFF
subtract BTF at A(call+3) from BTF at A (call+2)
difference BTF put at A (call+4)
R
CONA
convert BTF at A (R2) to ATF at A(R2)
CBTA
BIN hours to ASCII
CBTA
BIN min to ASCII
CBTA
BIN sec to ASCII
CBTA
BIN ms to ASCII
R
CBTA
convert one BINARY word to 2 ASCII digits
R

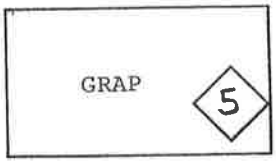

MATH
3
ADD
add two BTF's
together
A(call+2) +
A(call+3)
result in BTF
placed at
A (call+4)
R
PRIN4
print the ACC
as 4 ASCII
digits
R
MOVN
move 5 words
from a(R1) +1
to A(R2)
R
MOVT
move 9 words
from A(RO)+5
to A(R2)
R
CRLF
*PRINTP
generate
CRLF
R
MOVAS
move 12 words
from A(R1)
to A (R2)
R

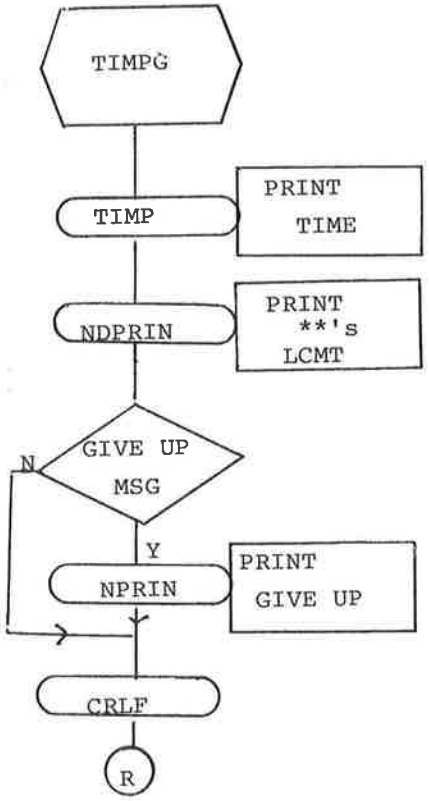

GRAP
1
DB2
GETREC
zero length rec
N
Y
GETREC
START RECORD
CRLF
DATLP
initialize variables
RANDIO
INITIO
print initial dialogue
got a 'CR'
N
got an 'S'
N
STATO←1
PLAY
is air tape
N
UNITMA ← 0
GRMN1 ← 2
UNITMA 2
ARMN1 1
GRINIT
initialize LCMT
GRINIT
TIMP
print time

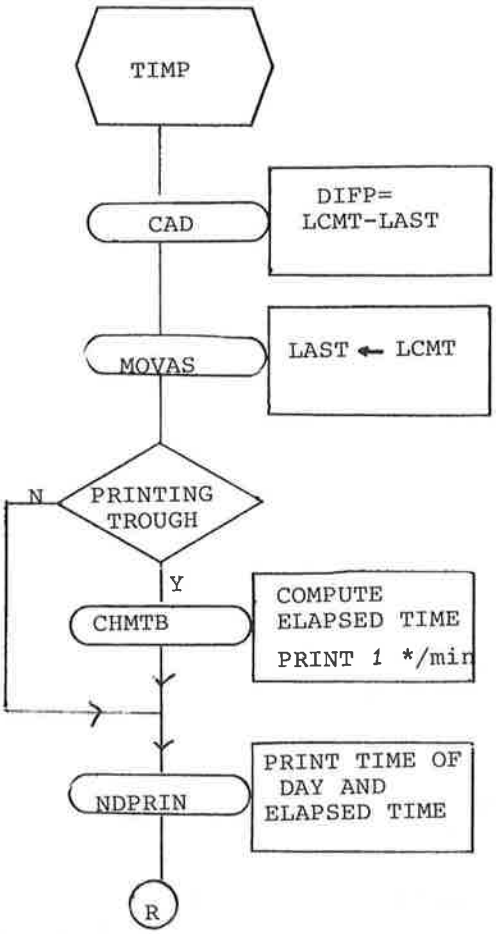

GRAP
2
GRINIT
PRINT AIR OR GROUND HEADINGS PRINT FIRST RECORD
STATISTICS ONLY
Y
N
ENABLE PRINT
DISABLE PRINT
R
GETRIC
PLAY
ZERO LENGTH RECORD
Y
ZLR
air tape
N
Y
ARSTATE←1
GRSTATE ← 1
R
N
air tape
N
GRAPH A
PROCESS RECORD
GRAPH T
PROCESS RECORD
R

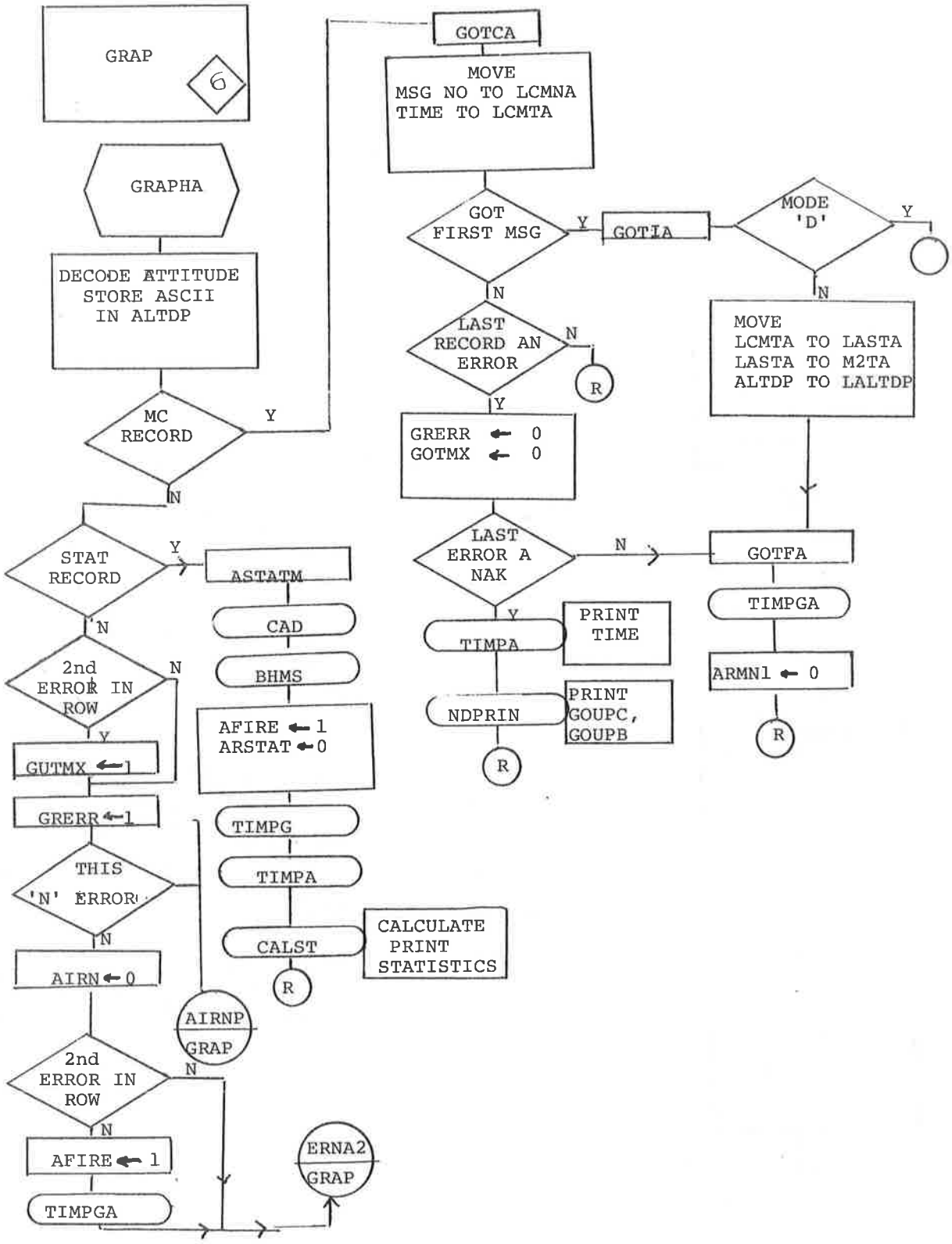

GRAP
3
CALST
ENABLE PRINT
CONA
CONVERT OUT OF COM TIME TO ATF
CAD
IN COM TIME = DURT - OUT COM TIME
DIV
Calculate %'s for each CATEGORY
PHOUR
Calculate MSGs PER HOUR for each CATEGORY
BINDEC
Convert binary numbers to decimal ASCII
NDPRIN
PRINT OUT STATISTICS CALCULATED
air tape
N
PRINT GROUND # OF GIVE UPS
R

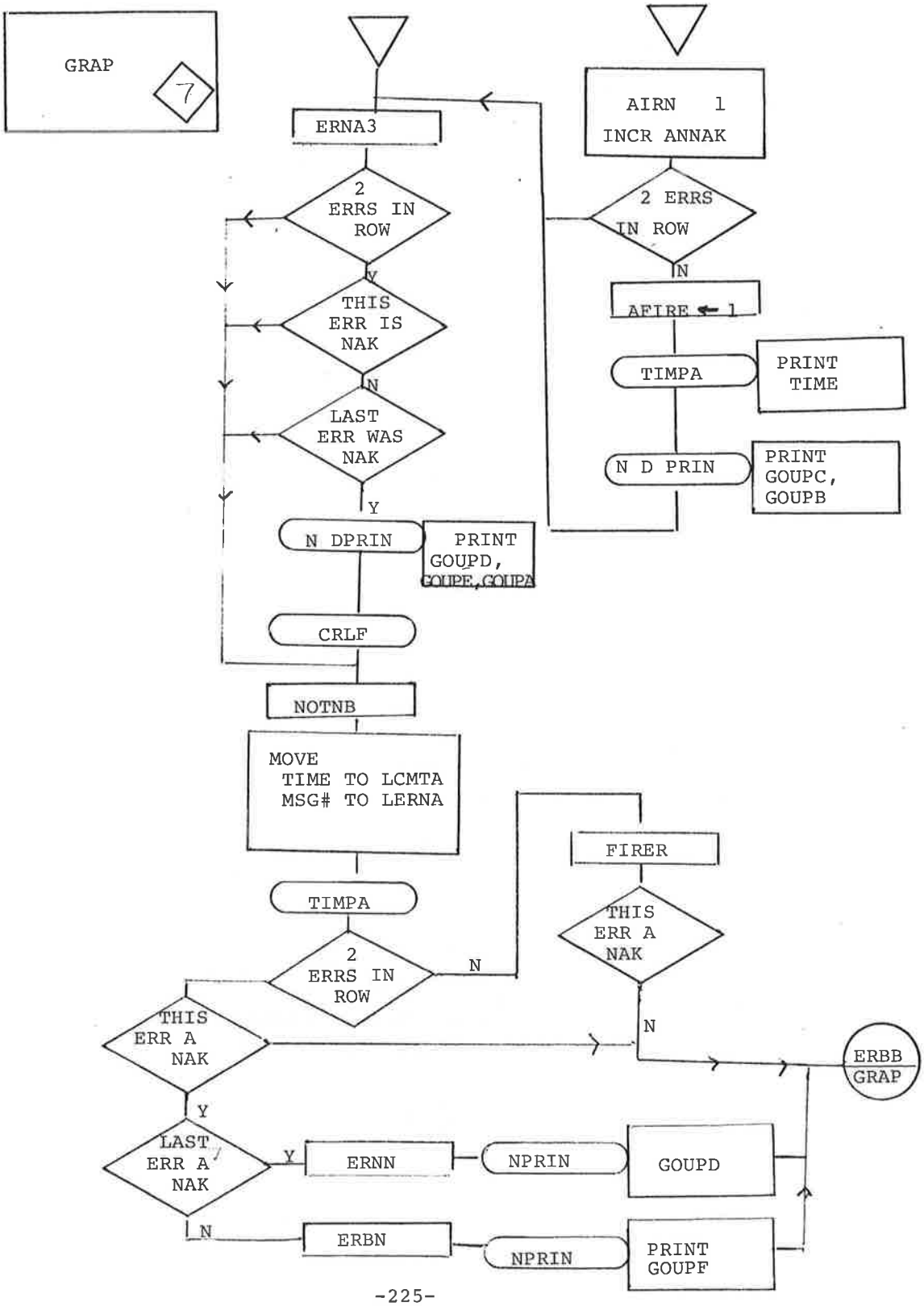

GRAP
4
GRAPH T
MC RECORD
Y
GOTC
INCR GNMC
is GIVE UP RECORD
Y
NERN
GOTMX←1 INCR GNGU
Y
STAT RECORD
Y
STATM
N
2nd ERR in ROW
Y
N
GRERR ← 1
FIRE ← 1
TIMPG
TIME PRINT
ERNO2
MOVE TIME TO LCMT MSG NO TO LERN
NO ERROR
N
INCR GNBM
Y
INCR GNNAR
TIMP
PRINT TIME
NRPIN
PRINT ERROR NO
ADD
ADD TO OUT OF COM COUNT
R
CAD
DURT = LCMT-M1T
BHMS
CALCULATE DURATION IN SECONDS
FIRE ← 1
GRSTATE ←0
TIMPG
TIME PRINT
CALST
CALCULATE, PRINT STATS
R
GRERR 0
MSG NO to LCMN
Add 2 to Scen NO
Time to LCMT
IS THIS MSG #0001
Y
MOVE LCMT TO M2T
N
ALPHA MSG
N
R
Y
TIMPG
GRMN1 0
R

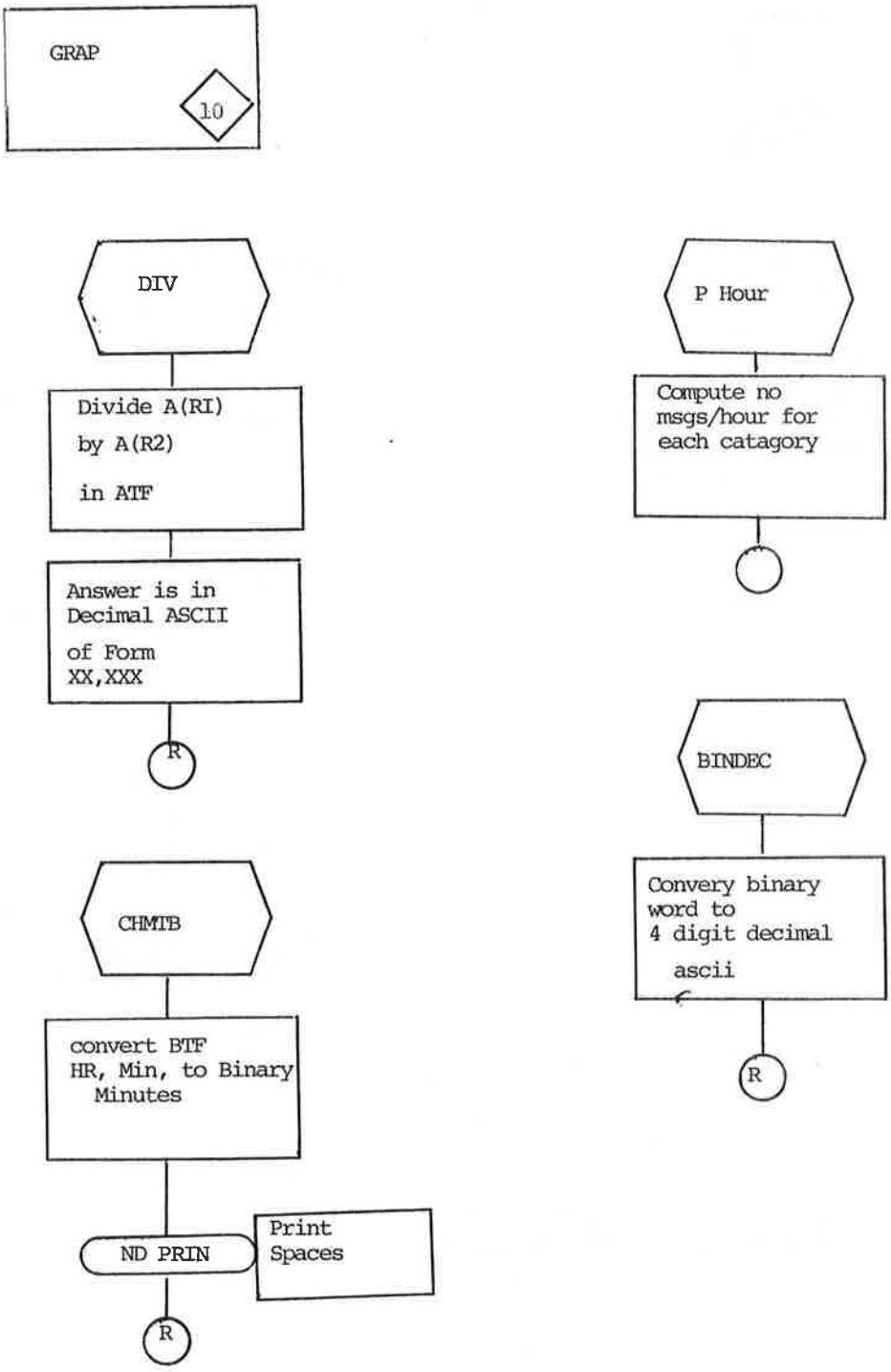

GRAP
5

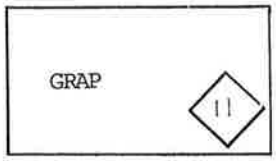

TIMPG
TIMP
PRINT TIME
NDPRIN
PRINT **'s LCMT
GIVE UP MSG
N
Y
NPRIN
PRINT GIVE UP
CRLF
R

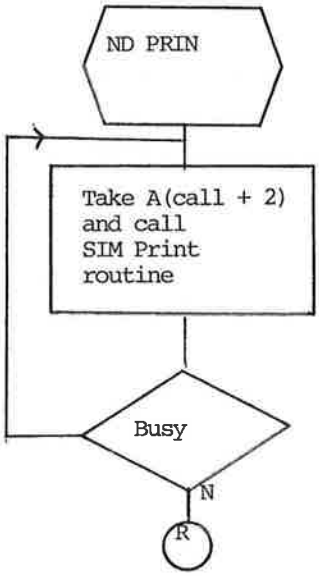

TIMP
CAD
DIFP= LCMT-LAST
MOVAS
LAST ← LCMT
N
PRINTING TROUGH
Y
CHMTB
COMPUTE ELAPSED TIME
PRINT 1 */min
NDPRIN
PRINT TIME OF DAY AND ELAPSED TIME
R

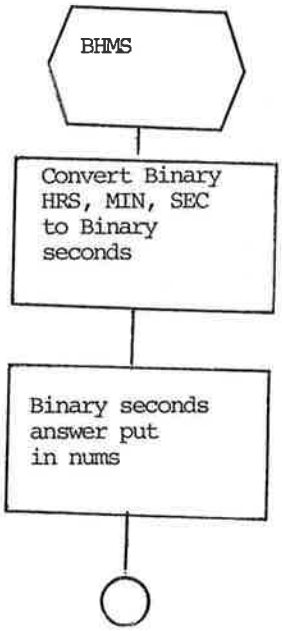

GRAP
6
GRAPHA
DECODE ATTITUDE
STORE ASCII
IN ALTDP
MC
RECORD
Y
N
STAT
RECORD
Y
ASTATM
CAD
BHMS
AFIRE ← 1
ARSTAT ← 0
TIMPG
TIMPA
CALST
CALCULATE
PRINT
STATISTICS
R
N
2nd
ERROR IN
ROW
Y
GUTMX ← 1
GRERR ← 1
THIS
'N' ERROR
N
AIRN ← 0
AIRNP
GRAP
2nd
ERROR IN
ROW
N
AFIRE ← 1
TIMPGA
ERNA2
GRAP
GOTCA
MOVE
MSG NO TO LCMNA
TIME TO LCMTA
GOT
FIRST MSG
Y
GOTIA
N
LAST
RECORD AN
ERROR
N
R
Y
GRERR ← 0
GOTMX ← 0
LAST
ERROR A
NAK
N
Y
TIMPA
PRINT
TIME
NDPRINT
PRINT
GOUPC,
GOUPB
R
MODE
'D'
Y
N
MOVE
LCMTA TO LASTA
LASTA TO M2TA
ALTDP TO LALTDP
GOTFA
TIMPGA
ARMN1 ← 0
R


GRAP
7
ERNA3
2 ERRS IN ROW
Y
THIS ERR IS NAK
N
LAST ERR WAS NAK
Y
N DPRIN
PRINT GOUPD, GOUPE, GOUPA
CRLF
NOTNB
MOVE TIME TO LCMTA
MSG# TO LERNA
TIMPA
2 ERRS IN ROW
N
THIS ERR A NAK
Y
LAST ERR A NAK
Y
ERNN
NPRIN
GOUPD
N
ERBN
NPRIN
PRINT GOUPF
AIRN 1
INCR ANNAK
2 ERRS IN ROW
N
AFIRE ← 1
TIMPA
PRINT TIME
N D PRIN
PRINT GOUPC, GOUPB
FIRER
THIS ERR A NAK
N
ERBB
GRAP


GRAP
10
DIV
Divide A(RI)
by A(R2)
in ATF
Answer is in
Decimal ASCII
of Form
XX,XXX
R
P Hour
Compute no
msgs/hour for
each catagory
BINDEC
Convery binary
word to
4 digit decimal
ascii
R
CHMTB
convert BTF
HR, Min, to Binary
Minutes
ND PRIN
Print
Spaces
R


GRAP
11


ND PRIN
Take A(call + 2)
and call
SIM Print
routine
Busy
N
R


BHMS
Convert Binary
HRS, MIN, SEC
to Binary
seconds
Binary seconds
answer put
in nums

VARIABLES CONTAINED IN GRAD DATA SEGMENT

NAME	USE air/ground	Format, explanation
LCMNA	A	ASCII, last MC message number
LCMN	G	
LCMTA	A	ATF, last MC time
LCMT	G	
LERNA	A	ASCII, last error message number
LERN	G	
LASTA	A	ATF, last time printed
LAST	G	
LALTDP	A	ASCII, last attitude information
ALTDP	A	ASCII, current attitude information
M1TA	A	ATF, first message time
M1T	G	
OUTCBA	A	BTF, elapsed error time
OUTCB	G	

FLAGS USED IN GRAPH MODULES

The following flags are used by GRAPHT, the ground tape analyser module, or GRAPHA, the air tape analyser module.

FLAGS	INTERPRETATION
GOTMX	1 if GRAPHT has received a give-up record 1 if 2 air error messages received in a row
GRERR ARERR	1 if last record was an error 1 if last air record was a NAK
UNITMA	1 if air tape being processed
STATO	2 if statistics only print
AIRN	2 if current air error is a NAK
GRMN1 ARMN1	1 if GRAPHT is expecting first message record 1 if GRAPHA is expecting first message record.
GRSTAT ARSTAT	1 if GRAPHT is expecting a statistics record 1 if GRAPHA is expecting a statistics record
FIRE AFIRE	1 if GRAPHT must print a trough 1 if GRAPHA must print a trough

APPENDIX A

This appendix presents a visual representation of two typical error records -- a "normal" one and a "transparent" one. There are special characters to represent the "non-print" characters inherent to the preambles and post-ambles.

11130011720=00900?B+½½¼21018A01AQQ%THE QUICK BROWN FOX JUMPED OVER THE LAZY DOF¢ $B_1B_2B_3B_4$

Time	11:13:00.117		
Altitude*	20=	(20D)	
Pitch*	009		
Roll*	00?	(F)	
Sentinel	B	(BCS error)	
Preamble	+*½½¼21018A01AQQ%		
Text	THE QUICK BROWN FOX		
	JUMPED OVER THE		
	LAZY DOF		
Postamble	¢B_1B_2		
Calculated BCS	B_3B_4		

NOTE: In the preamble - - ½ represents the non-printing SYN char.

- - ½ " " " SOH "
- - A " " " ACK "
- - % " " " STX "

In the postamble- - ¢ " " " ETX/ETB char.

- -B_1 " " encoded BCS "
- -B_2 " " " " "
- -B_3 " " " " "
- -B_4 " " " " "

* Not present in ground error records.

2359598017?=›‹:??9N+*½½¼HOl98bQQ%09414243??=›454‹4

4¢$B_1B_2B_3B_4$

Time	23:59:59.801	
Altitude	7?=	(7FD)
Pitch	›‹:	(ECA)
Roll	??9	(FF 9)
Sentinel	N	(NAK)
Preamble	+*½½¼HOl98bQQ%	
Text (Encoded)	(9 14243??=›454‹44	
Postamble	¢B_1B_2	
Calculated BCS	B_3B_4	

Note - - See previous page for preamble and postamble definitions.

Note - - Decoding of the text produces (in binary):

(09)	00001001	Byte - Count
(41)	01000001	ASCII A
(42)	01000010	ASCII B
(43)	01000011	ASCII C
(??)	11111111	Transparent FF
(=)	11011110	Transparent DE
(45)	01000101	ASCII E
(4)	01001110	ASCII N
(44)	01000100	ASCII D

APPENDIX B

This appendix shows a visual representation of a typical airborne transaction record (UTR). The record has exactly this appearance under off-line play-back conditions. That is, the entire record is composed of "printing" characters.

145859997l?900<017#9702B2l?

Time	14:58:59.997	
Altitude	l?9	(lF9)
Pitch	00<	(E)
Roll	017	
UTR Sentinel	#	
Message I.D.	9702B	
Mode	2	
Text Count	l?	(lF)

APPENDIX C

This appendix gives a visual representation of a typical statistics record. Each character is the representation of encoding a four-bit byte. This is exactly the appearance that the record would have if it were to be off-line played-back and printed. In this type of record (statistics) all recorded characters are in the set of "printing" characters.

```
09261298300;900;900;900;900000001000000010003000000
040003000200000000000000000000000005<00000000000000
1=10000005> 0000150> 000056;800009?=800>;00;=00;=00;8
000000000001000000010000000000060000000000000000
```

Time		09:26:12.983	
Echo Acknowledge	Class 1	00;9	(B9)
	2	00;9	(B9)
	3	00;9	(B9)
	4	00;9	(B9)
Echo Non-akcnowledge	Class 1	0000	
	2	0001	
	3	0000	
	4	0001	
Echo Non-response	Class 1	0003	
	2	0000	
	3	0004	
	4	0003	
Echo BCS Error	Class 1	0002	
	2	0000	
	3	0000	
	4	0000	
Echo Total Bit Errors	Class 1	00000000	
	2	0000005<	(5E)
	3	0000000	
	4	000001=1	(1D1)
Echo Total Text Chars.	Class 1	0000005>	(5C)

	Class 2	0000150>	(150C)
	3	000056;8	(56B8)
	4	00009?=8	(9FD8)
Non-Echo Acknowledge	Class 1	00>;	(CB)
	2	00;=	(BD)
	3	00;=	(BD)
	4	00;8	(B8)
Non-Echo Non-acknowledge	Class 1	0000	
	2	0000	
	3	0001	
	4	0000	
Non-echo Non-response	Class 1	0001	
	2	0000	
	3	0000	
	4	0006	
Non-echo BCS Error	Class 1	0000	
	2	0000	
	3	0000	
	4	0000	

APPENDIX D

This appendix is a visual representation of a typical ground message transaction record (MTR). The entire record is composed of "printing" characters.

C9876A0825386910825394062;3:

MTR Class	C	
Message I.D.	9876A	
Start Time	082538691	
End Time	082539406	
Text Count	2;	(2B)
Mode	3	
Scenario Message Number	:	(A)

APPENDIX E

This appendix describes the three types of ground station data acquisition cassette records.

GROUND DATA ACQUISITION TAPE

RECORD TYPES: (1) TRANSACTION (MTR); (2) ERROR; (3) STATISTICS

TYPE (1): TRANSACTION (MTR)

MTR CLASS	1	ASCII CHAR.	NOTE 10
MESSAGE I.D.	5	ASCII CHARS.	NNNNA (FOUR NUMERIC--ONE ALPHA)
START TIME	9	ASCII CHARS.	HHMMSSUUU
END TIME	9	ASCII CHARS.	HHMMSSUUU
TEXT COUNT	2	ENCODED CHARS.	NOTE 2 (COUNT OF ORIGINATING MSG.)
MODE	1	ASCII CHAR.	1, 2, 3, D, OR H (ORIGINATING MSG.)
SCENARIO MSG. #	1	ENCODED CHAR.	NOTE 11

TYPE (2): ERROR

TIME	9	ASCII CHARS.	HHMMSSUUU
SENTINEL	1	ASCII CHAR.	NOTE 10
PREAMBLE	12	OR 17 CHARS.	NOTE 12
TEXT	≤	235 CHARS.	NOTE 4
POSTAMBLE		3 CHARS.	NOTE 5
CALC. BCS		2 CHARS.	NOTE 13

TYPE (3): STATISTICS

SEE AIRBORNE DESCRIPTION (APPENDIX F).

NOTE 1:

The aircraft attitude components are the result of a twelve-bit analogue-to-digital conversion. The raw twelve-bit values are recorded by splitting them into three four-bit bytes. Each byte is added to hexadecimal thirty and recorded most-significant byte first, then middle significant, and, lastly, least significant. This method of encoding produces the following character set:

BYTE	ENCODES TO	CHARACTER
0000		0
0001		1
0010		2
0011		3
0100		4
0101		5
0110		6
0111		7
1000		8
1001		9
1010		:
1011		;
1100		>
1101		=
1110		<
1111		?

NOTE 2:

The text count is eight bits encoded as two four-bit bytes (Note 1).

NOTE 3:

The first five characters are recorded exactly as received UNLESS any belongs to the set hexadecimal 10, 11, 12, 13, or 14, or if the eighth bit (most significant) is one. If the MSB is one, the character is recorded as hexadecimal 7E, which is the diacritical mark (~). If the character is hexadecimal 10 through 14, it is recorded as lower case A through E, respectively. The sixth character is recorded as seen, if it is an acceptable mode; else, a U is recorded, followed by the encoded representation (Note 2) of the character seen. The final eleven characters (up to, and including, the STX) are recorded as seen, unless the special cases of hexadecimal 10 through 14 or the most significant bit is one are seen, in which case the same substitution (as previously discussed) takes place.

NOTE 4:

If the text is "normal" (mode 2 or mode 3), it is recorded as seen, with hexadecimal 7E or lower case A through E substitution (Note 3), if necessary. If the text is "transparent" (modes 1, D, H, or U), the entire text field is recorded in encoded format (Note 2).

NOTE 5:

The ETX character is recorded as seen, with hexadecimal 7E or lower case A through E substitution, as required. The received BCS characters are recorded by the encoding process (Note 2).

NOTE 6:

The BCS characters calculated during the reception process are recorded by the encoding process (Note 2).

NOTE 7:

The statistics record is preceded by an empty record (end-of-file) as a signal to the data analysis program. The record is made up of single-precision event counters for occurrences of echo messages of class 1 (text length of 0 or 1 char.), class 2 (text length of 2-20 chars.), class 3 (text length of 31-120 chars.), and class 4 (text length in excess of 120 chars.). The subclasses are ACK seen, NAK (not ACK) seen, non-response, and BCS error detected, for both echo-type and non-echo-type messages. In addition, there are double-precision counters for total echo text characters and total echo bits--in-error (classified as above).

NOTE 8:

The single precision word is encoded as four four-bit bytes. The scheme parallels that of Note 1.

NOTE 9:

These values are not applicable in the airborne system. All values are zeroed and then recorded as in Note 8.

NOTE 10:

B - - "Pure" BCS error

C - - Completion (successfully).

I - - Bit-Compare error.

L - - Message too long.

N - - NAK (not ACK) detected.

R - - No response to poll from ground.

S - - Message too short.

W - - Aircraft I.D. received improperly.

NOTE 11:

Since the number of messages in the canned scenario is sixteen, the encoding is to record the "true" scenario number less one. That is, message one is recorded as message zero, message two as message one, . . ., message sixteen as message fifteen (Note 1). The data analysis resolves this technique for proper visual display. The symbol "K" is used to denote a non-scenario message - - time poll - general poll - system entry poll - keyboard message.

NOTE 12:

If the error recording is caused by a non-response (R-type), then the first five characters as described in Note 3, are not applicable. The message is recorded as sent, where the preamble is twelve characters. In all other cases, Note 3 applies fully.

NOTE 13:

If the case is a non-response (R-type) these two characters are forced to zero, prior to recording. In all other cases, Note 6 applies fully.

APPENDIX F

This appendix describes the three types of airborne data acquistion cassette records. All notes in this appendix refer to the notes in Appendix E.

AIRBORNE DATA ACQUISITION TAPE

RECORD TYPES: (1) TRANSACTION (UTR): (2) ERROR: (3) STATISTICS

TYPE (1): TRANSACTION (UTR)

TIME	9	ASCII CHARS.	HHMMSSUUU
ALTITUDE	3	ENCODED CHARS.	NOTE 1
PITCH	3	ENCODED CHARS.	NOTE 1
ROLL	3	ENCODED CHARS.	NOTE 1
SENTINEL	1	ASCII CHAR.	#
MESSAGE I.D.	5	ASCII CHARS.	NNNNA (FOUR NUMERIC--ONE ALPHA)
MODE	1	ASCII CHAR.	1, 2, 3, D, OR H
TEXT COUNT	2	ENCODED CHARS.	NOTE 2

TYPE (2): ERROR

TIME	9	ASCII CHARS.	HHMMSSUUU
ALTITUDE	3	ENCODED CHARS.	NOTE 1
PITCH	3	ENCODED CHARS.	NOTE 1
ROLL	3	ENCODED CHARS.	NOTE 1
SENTINEL	1	ASCII CHAR.	B (BCS ERROR) OR N (NAK SEEN)
PREAMBLE	17	CHARS.	NOTE 3
TEXT	$\leq$	220 CHARS.	NOTE 4
POSTAMBLE	3	CHARS.	NOTE 5
CALC. BCS	2	CHARS.	NOTE 6

TYPE (3): STATISTICS (NOTE 7)

TIME		9	ASCII CHARS.	HHMMSSUUU
ECHO-ACK.	1	4	ENCODED CHARS.	NOTE 8
"	2	4	"	"
"	3	4	"	"
"	4	4	"	"
ECHO-NAK.	1	4	"	"
"	2	4	"	"
"	3	4	"	"
"	4	4	"	"
ECHO-NR	1	4	"	NOTE 9
"	2	4	"	"
"	3	4	"	"
"	4	4	"	"
ECHO-BCS	1	4	"	NOTE 8
"	2	4	"	"
"	2	4	"	"
"	3	4	"	"
"	4	4	"	"
ECHO-BE	1	8	"	NOTE 9
"	2	8	"	"
"	3	8	"	"
"	4	8	"	"
ECHO-TC	1	8	"	"
"	2	8	"	"
"	3	8	"	"
"	4	8	"	"

NON-ECHO-ACK.	1	4	ENCODED CHARS.	NOTE 8
"	2		"	"
"	3		"	"
"	4		"	"
NON-ECHO-NAK.	1		"	"
"	2		"	"
"	3		"	"
"	4		"	"
NON-ECHO-NR	1		"	NOTE 9
"	2		"	"
"	3		"	"
"	4		"	"
NON-ECHO-BCS	1		"	NOTE 8
"	2		"	"
"	3		"	"
"	4		"	"

APPENDIX G

This appendix shows the status terminal codes and data acquisition cassette "error" record codes.

GLOSSARY OF GROUND STATION STATUS TERMINAL CODES

CHARACTER	CASE	INDICATION
.	N/A	Message received with acknowledge and good BCS.
B	Upper	BCS error received with acknowledge.
B	Lower	BCS error received without acknowledge.
I	Upper	Bit-compare error received with acknowledge.
I	Lower	Bit-compare error received without acknowledge.
L	Upper	Message received too long with acknowledge.
L	Lower	Message received too long without acknowledge.
N	Upper	Message received without acknowledge, but good BCS.
R	Upper	No response received.
S	Upper	Message received too short with acknowledge.
S	Lower	Message received too short without acknowledge.
W	Upper	Message received with improper address with acknowledge.
W	Lower	Message received with improper address without acknowledge.
-	N/A	Message sent.

GLOSSARY OF AIRBORNE STATION STATUS TERMINAL CODES

CHARACTER	CASE	INDICATION
.	N/A	Message received with acknowledge and good BCS.
B	Upper	BCS error received with acknowledge.
B	Lower	BCS error received without acknowledge.
L	Upper	Message received too long.
N	Upper	Message received without acknowledge.
S	Upper	Message received too short.
W	Upper	Message received with improper address.
-	N/A	Message sent.

GLOSSARY OF GROUND STATION RECORDED ERROR INDICATORS

B	Upper	Message received with BCS error.
I	Upper	Message received with bit-compare error.
L	Upper	Message received too long.
N	Upper	Message received without acknowledge.
R	Upper	No response received.
S	Upper	Message received too short.
W	Upper	Message received with improper aircraft address.

GLOSSARY OF AIRBORNE STATION RECORDED ERROR INDICATORS

B	Upper	Message received with BCS error.
N	Upper	Message received without acknowledge.

APPENDIX H

This appendix describes the scenario cassette tape which contains the sixteen pre-created messages. Also described is the method of generation of the scenario cassette tape.

SCENARIO TAPE FORMAT

The Data Link system uses a scenario tape as its input. The tape must be constructed according to the following rules:

1) There must be sixteen scenario message records
2) Each record must be prefixed with the following eleven character heading:

   M0000 01(ACK) QQ

   Where M is the ASCII mode character 1,2,3, or H.
3) Each record is terminated by an ETX character.
4) Modes 2,3 must have only 7-bit ASCII characters as text.
5) Modes 1,H must have their 8-bit text characters encoded in binary coded hexadecimal. These modes also must have a text count character which occurs as the first text word. This count is equal to the number of text characters to follow, plus one.
6) The maximum number of text characters allowable is 220.

SCENARIO TAPE GENERATION

1) Using EDIT-960, running under PSM, create a source of the scenario tape. Each character should be packed into a halfword. The pseudo-operation 'DATA' can be used to easily convert a text string into halfword storage. Each record must be ended by the two halfword characters '137F'. Each record must start on a full word boundary. All modes 1 or H text must be encoded.

   To aid in editing, '7F' may be used as a null character to fill halfword boundaries. This character is always ignored by the generation program.

2) Using SALM, assemble the source and generate a binary output tape.

3) Lead that binary tape at location 3000. This will destroy the PSM monitor.

4) Plug an interval timer board into slot EF1.

5) Load the program 'USER DIAGNOSTIC' at location 0100.

6) The program will self-start and will print out: 'USER'.

7) Set panel switches 0-13 down. Set switch 11 to 1. Then flip switches 14, 15. The program will print out:

   'SCENARIO
    BIN AT LOC'

8) Ready an output tape on the record unit.

9) Type '3000', carriage return.

10) The program will now interpret the binary scenario table which was loaded in step 3. It will then record the scenario tape for use on the Data Link system.

11) When the recording is complete, the program will print: 'USER'. The tape is complete. The following steps will verify the data which was recorded.

12) Rewind the tape, bring it to ready, and change the record unit to the playback unit.

13) Set switch 8 to 1, and flip switches 14, 15. The program will print:

'CASSETTE UNIT
HOW MANY RECORDS (HEX)?'

Then type '10' which is hexadecimal for 16. The program will print:

'PLAYBACK RECORD (P/R)?'

Then type 'P'. The program will then print:

'PRINT (Y/N)?'

Then type 'Y'.

14) The program will now read the tape and output a listing of the scenario tape as the Data Link system would interpret it. All ASCII characters are printed as their ASCII equivalent. All non-ASCII characters are printed as a 16-bit word of the form '(00XX)'.

After each record has been printed (ending with ETX), a word count is printed and an automatic CRLF is generated. This text count is printed in hexadecimal and counts the

the number of words read from each record. This does not correspond to the number of characters in each record because of the various halfword and encoding methods employed in generating the tape.

15) The program will print 'USER' when it is completed.

16) At this time ODT may be entered at location '27E1'. The binary scenario table may now be changed if necessary. Any superfluous entries may be modified by inserting the '7F' ignore character.

17) The USER DIAGNOSTIC program may be manually restarted at location 0100 and another scenario tape generated.

USER DIAGNOSTIC PROGRAM

The USER'S Diagnostic Program is used to generate and dump scenario tapes. It is built by linking SIH with the following six assembly modules: UPSEG, USERA, USAUX, USERS, USERD and CALODT. It is a load and go program loaded at location 0100. It requires that an internal timer board be plugged into slot EF1.

```
USER
CASSETTE UNIT
HOW MANY RECORDS (HEX)?10
PLAYBACK/RECORD (P/R) ?P
PRINT (Y/N)?Y

20000 01(0006)00(0002)(000D)(000A)LA INTERNAT AIRPORT(000D)(000A)INFO ZU
LU CIELING UNLIMITED(000D)(000A)VIS 3 MILES IN HAZE AND SMOKE(000D)(000A
)TEMP 61 DEW POINT 54 WIND 270(000D)(000A)AT 14 ALTIMITER 30.10 ILS APPR
OACHES(000D)(000A)RUNWAYS 24 LEFT AND 24 RIGHT(000D)(000A)DEPARTURE RUNW
AYS 25 LEFT AND 25 RIGHT(0003)
0045
20000 01(0006)00(0003)
000C
30000 01(0006)00(0002)(000D)(000A)LA 246 CLEARED TO SAN(000D)(000A)FRANC
SCO AIRPORT VIA VENTURA(000D)(000A)TWO DEPARTURE SALINAS TRANSITION(000D
)(000A)JET 98 SANTA CRUZ INTERSECTION(000D)(000A)WOODSIDE 141R,WOODSIDE
VICTOR 25(000D)(000A)SAN FRANCISCO MAINTAIN FLIGHT LEVEL(000D)(000A)280,
SQUAWK 4701 DM DEPARTURE(0003)
0044
30000 01(0006)00(0003)
000C
20000 01(0006)00(0002)(000D)(000A)LAX GROUND 121.8              (0003)
002B
30000 01(0006)00(0002)(000D)(000A)TAXI TO RUNWAY 25 LEFT      (0003)
0029
20000 01(0006)00(0002)(000D)(000A)UA 246 B727 AC 7619 ZFW 127.1(000D)(00
0A)ATOGW 146.1 MAC 22.4 PSGRS 18F(000D)(000A)104Y LOWER ROTATING BEACON
HAS(000D)(000A)ONE LIGHT OUT. FUEL 20.(0003)
0085
30000 01(0006)00(0002)(000D)(000A)UA 246 DEPARTURE REPORT RECEIVED(000D)
(000A)DEPARTED LAX AT 20 MINUTES(000D)(000A)AFTER 12A NOTHING TO REPORT
ON(000D)(000A)AIRCRAFT CONDITION NOW   (0003)
0085
H0000 01(0006)00(0002)(0001)(0003)
000E
10000 01(0006)00(0002)(0001)(0003)
000E
H0000 01(0006)00(0002)(001E)(000D)(000A)WIND 275 AT 16 SQUAWK IDENT(0003
)
002B
10000 01(0006)00(0002)(001E)(000D)(000A)LAX DEPARTURE 125.2          (0003
)
002B
H0000 01(0006)00(0002)(0078)(000D)(000A)SQUAK IDENT  RADAR CONTACT (000D
)(000A)MAINTAIN HEADING 250 RADAR VECTORS(000D)(000A)FOR VTU(000D)(000A)
CLIMB TO AND MAINTAIN 6000(000D)(000A)TURB. AT 5000   (0003)
0030
10000 01(0006)00(0002)(0078)(000D)(000A)TRAFFIC 11 O,CLOCK 3 MILES 4000(
000D)(000A)FEET SOUTHEAST BOUND SLOW(000D)(000A)PROCEED DIRECT VTU(000D)
(000A)TRFIC 3 OCLK 4 MILES 6000 FT SBND FST(0003)
0085
H0000 01(0006)00(0002)(00DC)(000D)(000A)CLEAR OF TRAFFIC(000D)(000A)LAX
CENTER 125.0(000D)(000A)SQUAWK IDENT  RADAR CONTACT(000D)(000A)CLIMB TO
AND MAINTAIN FLIGHT(000D)(000A)LEVEL 280(000D)(000A)EXPEDITE THROUGH FLI
GHT LEVEL 90(000D)(000A)OAK CENTER 128.7(000D)(000A)SQUAWK IDENT  RADAR
CONTACT(000D)(000A)DESCEND TO AND MAINTAIN  7000  (0003)
0045
10000 01(0006)00(0002)(00DC)(000D)(000A)OAK CENTER 126.8(000D)(000A)SQUA
```

```
WK IDENT  RADAR CONTACT(000D)(000A)SFO ALTIMITER 30.15(000D)(000A)PROCEE
D DIRECT OAK DIRECT SFO(000D)(000A)SFO APPROACH 123.85(000D)(000A)SQUAWK
 IDENT  RADAR CONTACT(000D)(000A)RADAR VECTORS FOR ILS RUNWY 28(000D)(00
0A)LEFT APPROACH(000D)(000A)TURN RIGHT HEADING 35(0003)
0030
USER
```

APPENDIX I

This appendix shows the generalized message format and describes some of the special pre-amble characters. For the description of the specialized message formats (system entry poll, general poll, time re-synchronization poll, etc.) the reader is referred to the system specification.

MESSAGE FORMAT

FUNCTION	ASCII CODE	HEX	CHARACTERS
Prekey[1]		00	1-150
Bit Sync[1]	+	2B	1
Bit Sync[1]	*	2A	1
Character Sync[1]	+	2B	1
Character Sync[1]	+	2B	1
Bit Sync	+	2B	1
Bit Sync	*	2A	1
Character Sync	SYN	16	1
Character Sync	SYN	16	1
Start of Heading	SOH	01	1
Code	See Next Page		1
Message Number	Numeric		4
Message Alpha	Alpha		2
Address	Alphanumeric		2
Technical Acknowledge-ment	ACK/NAK	06/15	1
Label Character One	See Next Page		1
Label Character Two	See Next Page		1
Start of Text	STX	02	1
Text			235 max.
End of Text	ETX or ETB	03/17	1
Block Check Sequence			1
Block Check Sequence			1

[1]These are timing, modem, and modem controller characters. All characters following these are passed to the computer.

UPLINK MESSAGE LABELS

Label 1	Label 2
O System Entry	O System Entry
E MSG for Printer	Q Other Messages
T Resynch Real Time Clock	
Q Other Messages	

DOWNLINK MESSAGE LABELS

Label 1	Label 2
O System Entry	O System Entry
E MSG for Printer	Q Other Messages
R Request for Special General Poll	
Q Other Messages	

MODE CHARACTER

MODE
1 Transparent Text (echo test)
2 Normal Text
3 Normal Text (echo test)
H Transparent text
D System Entry

Message Labels and Mode Characters

REPORT OF INVENTIONS APPENDIX J

After diligent review of the work performed under this contract, no new innovation, discovery, improvement, or invention was made.