

# RX11

## RX11 RELIABILITY MD-11-DZRXA-D

EP-DZRXA-D-DL-A

FEB 1978

COPYRIGHT ©1976

**digital**

FICHE 1 OF 1

Made In U.S.A.

**DZRXA-D SEQ**

The grid contains 100 individual diagrams, each representing a different component or sequence of the DZRXA-D system. The diagrams are organized into 10 rows and 10 columns. The first row is titled 'DZRXA-D SEQ'. The diagrams include various types of technical drawings, such as block diagrams, flowcharts, and data tables. The diagrams are arranged in a regular grid pattern across the page.







IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZRXA-D-D  
PRODUCT NAME: RX11 SYSTEM RELIABILITY TEST  
DATE CREATED: DEC. 21, 1975  
MAINTAINER: DIAGNOSTIC ENGINEERING  
AUTHOR: DAVID L. ADAMS  
REVISED: SEPT. 12, 1975 BY D. L. ADAMS  
NOV. 20, 1975 BY B. P. BURGESS

COPYRIGHT (C) 1975  
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.



TABLE OF CONTENTS

- 1.0 GENERAL PROGRAM INFORMATION
  - 1.1 ABSTRACT
  - 1.2 SYSTEM REQUIREMENTS
    - 1.2.1 HARDWARE
    - 1.2.2 SOFTWARE
- 2.0 OPERATING INSTRUCTIONS
  - 2.0.1 OUTLINE OF OPERATING PROCEDURE
  - 2.1 LOADING PROCEDURE
  - 2.2 STARTING ADDRESSES
  - 2.3 OPERATOR ACTION BEFORE STARTING PROGRAM
    - 2.3.1 DEVICE ADDRESS SELECTION
    - 2.3.2 NON-STANDARD DISKETTE ADDRESS SELECTION
    - 2.3.3 SOFTWARE SWITCH REGISTER (LOC. 176)
    - 2.3.4 TEST PARAMETER SELECTION ("DTESTP" LOC. 1212)
  - 2.4 OPERATOR ACTION TO RUN THE PROGRAM
    - 2.4.1 STARTING THE PROGRAM
    - 2.4.2 OPERATING CONDITIONS
    - 2.4.3 ACT11 & XXDP HOOKS
  - 2.5 PROGRAM OPTIONS
    - 2.5.1 PSEUDO SCOPE LOOP
    - 2.5.2 DISKETTE COMPATABILITY
  - 2.6 RUN TIME
- 3.0 ERROR DETECTION
  - 3.1 ERROR DEFINITIONS
  - 3.2 DEFINITIVE ERROR CODES
  - 3.3 UNEXPECTED OR MISSING ERROR CONDITIONS
  - 3.4 POWER FAILURE
  - 3.5 PROGRAM HUNG
- 4.0 ERROR REPORTING
- 5.0 HALTS



**1.0 GENERAL PROGRAM INFORMATION****1.1 ABSTRACT**

THE RX11 SYSTEM RELIABILITY PROGRAM CONSISTS OF SELECTABLE TESTS THAT CHECK THE OPERATION OF THE RX11 SYSTEM, BY WRITING, READING AND VERIFYING VARIOUS DATA PATTERNS, UNDER VARIOUS (SELECTABLE) HEAD MOVEMENTS. IT CAN TRANSFER DATA AND CHECK FOR ERRORS OVER THE ENTIRE DISKETTE, ALL TRACKS AND SECTORS, OR BETWEEN SEPARATELY SELECTABLE TRACK AND SECTOR ADDRESS LIMITS.

AS WELL AS TRANSFERRING DATA THE PROGRAM ACCUMULATES STATISTICAL DATA ON THE FOLLOWING:

- A. COMPLETED PASSES OF THE PROGRAM
- B. NUMBER OF RESTARTS
- C. NUMBER OF SECTORS WRITTEN/READ
- D. RECOVERABLE AND UNRECOVERABLE FAULTS AS FOLLOWS:

- 1. PARITY ERRORS
- 2. ERROR FLAG ERRORS
- 3. INTERRUPT ERRORS
- 4. SEEK ERRORS
- 5. DATA CRC ERRORS
- 6. CRC NO DATA ERRORS
- 7. DATA NO CRC ERRORS
- 8. DELETED DATA MARK ERRORS
- 9. WRITE ERRORS
- 10. READ ERRORS

- E. TOTAL OF EACH ERROR CODE DETECTED
- F. TOTAL OF TIMES EACH TRACK WAS ACCESSED
- G. TOTAL OF TIMES HEAD MOVED TO A TRACK.
- H. TOTAL OF ERRORS DETECTED PER TRACK PER DRIVE.

THE ABOVE INFORMATION IS REPORTED BY RUNNING THE ERROR DUMP PROGRAM.

**1.2 SYSTEM REQUIREMENTS****1.2.1 HARDWARE REQUIREMENTS**

THE FOLLOWING EQUIPMENT IS REQUIRED.

- A. PDP-11 SERIES OF COMPUTER WITH MIN 8K MEMORY
- B. RX11 FLOPPY DISK SYSTEM INCLUDING DUAL OR SINGLE DRIVE RXD1 AND PDP-11 INTERFACE. (SEE SECTION 2.3 FOR SELECTION OF REGISTER ADDRESSES AND VECTOR ADDRESS)
- C. CONSOLE TELEPRINTER

**1.2.1 SOFTWARE REQUIREMENTS**

THIS PROGRAM ASSUMES THAT THE RX11 INTERFACE DIAGNOSTIC ( MAINDEC - 11 - DZRXB-\* ) HAS BEEN SUCCESSFULLY RUN ON THIS SYSTEM.



## 2.0 OPERATING INSTRUCTIONS

## 2.0.1 OUTLINE OF OPERATING PROCEDURE

THE STANDARD OPERATING PROCEDURE FOR THE RX11 RELIABILITY TEST (TO RUN ALL TESTS ON BOTH DRIVES WITH NO OPERATOR INTERVENTION VIA THE SWITCH REGISTER) IS AS FOLLOWS:

- A. LOAD THE PROGRAM INTO MEMORY
  - 1. IF IT'S BEING LOADED FROM A DISKETTE, REPLACE THE "LIBRARY DISKETTE" WITH A "SCRATCH DISKETTE"
- B. START THE PROGRAM RUNNING AT LOCATION 200
- C. THE PROGRAM WILL TYPE OUT THE FOLLOWING:
  - 1. PROGRAM NAME AND REVISION
  - 2. RX11 REGISTER AND VECTOR ADDRESSES
  - 3. UNITS BEING TESTED
  - 4. "P"ATTERN, "T"EST, AND "S"EQUENCE

IT THEN STARTS RUNNING UNDER THOSE CONDITIONS
- D. IF THERE ARE NO ERRORS, AT THE END OF THE COMPLETED PASS A "D" AND "BELL" IS TYPED, AND THE PROGRAM WILL CONTINUE ON FOR ANOTHER PASS.
- E. HALT THE PROGRAM AS FOLLOWS:
  - 1. IF THERE IS A HARDWARE SWITCH REGISTER, PUT SW 14 UP TO HALT AT THE END OF PASS.
  - 2. IF THERE IS NO SWITCH REGISTER, OR TO HALT AT ANY TIME, HALT THE PROCESSOR.

## 2.1 LOADING PROCEDURE

LOAD THE PROGRAM INTO MEMORY USING STANDARD PROCEDURE FOR BINARY PAPER TAPES.

MAKE SURE THAT THE TOTAL SYSTEM IS READY FOR OPERATION, DISKETTE INSERTED CORRECTLY, DOORS CLOSED ON DRIVES TO BE TESTED, ETC.

## 2.2 STARTING ADDRESSES

THE PROGRAM HAS THREE (3) STARTING LOCATIONS.

## 2.2.1 INITIAL START (LOC. 200)

THIS STARTING LOCATION INITIALIZES THE PROGRAM AS FOLLOWS:

- A. CLEARS ALL ERROR LOGS
- B. RESETS COUNTERS AND CONSTANTS
- C. TYPES OUT RX11 REGISTER AND VECTOR ADDRESSES BEING USED.
- D. TYPES THE DRIVE AND TEST SELECTION IN "DTESTP"
- E. INITIATES THE COLLECTION OF STATISTICAL DATA PERTAINING TO THE OPERATION OF THE RX11 SYSTEM.



## 2.2.2 RESTART (LOC. 202)

THIS STARTING LOCATION DIRECTS THE PROGRAM TO CONTINUE RUNNING USING DRIVE AND TEST SELECTIONS SPECIFIED IN THE PREVIOUS INITIAL START. IT ALSO CONTINUES TO ACCUMULATE STATISTICAL INFORMATION, AMENDING IT TO THE DATA ALREADY COLLECTED PRIOR TO THIS RESTART.

## 2.2.3 ERROR DUMP (LOC. 204)

THIS STARTING ADDRESS DIRECTS THE PROGRAM TO PRINT OUT ON THE CONSOLE TELEPRINTER ALL THE DATA ACCUMULATED FROM THE LAST "INITIAL START" TO THE TIME OF THE PRINTOUT. TYPING OUT THIS INFORMATION DOES NOT DESTROY IT, SO A RESTART CAN BE INITIATED AND INFORMATION WILL CONTINUE TO BE COLLECTED.

## 2.3 OPERATOR ACTION BEFORE STARTING PROGRAM

## 2.3.1 DEVICE ADDRESS SELECTION

LIKE MOST OPTIONS ON THE POP11 THE RX11 INTERFACE CARD HAS JUMPERABLE REGISTER AND VECTOR ADDRESSES. THIS ALLOWS FOR DEVICES WITH THE SAME STANDARD ADDRESSES TO BE JUMPED TO AN OTHER ADDRESS SO THEY WILL RUN WITHOUT CONFLICT.

THE PROGRAM MUST KNOW WHAT ADDRESSES ARE BEING USED, AS IT IS THROUGH THESE REGISTERS AND VECTOR THAT ALL COMMUNICATION BETWEEN THE POP11 AND RX11 IS HANDLED.

IF THE RX11 SYSTEM UNDER TEST IS JUMPED FOR REGISTER ADDRESS OTHER THAN STANDARD, WHICH IS RXCS=177170 AND RXDB=177172. PLACE IN THE MEMORY LOCATION CALLED "RXCS" (LOC. 1206) ITS NEW ADDRESS, AND IN LOCATION "RXDB" (LOC. 1210) ITS NEW ADDRESS. IF THERE IS A NONSTANDARD INTERRUPT VECTOR ADDRESS (STANDARD IS LOC 264) THEN PLACE IN MEMORY LOCATION CALLED "INTVEC" (LOC. 1204) ITS NEW ADDRESS.

IF THESE THREE MEMORY LOCATIONS DO NOT CONTAIN THE ADDRESSES THAT THE INTERFACE BOARD IS WIRED TO THE PROGRAM WILL REPORT "TEST HUNG" AND HALT, OR HALT AT THE VECTOR ADDRESS THAT IS JUMPED IN. (NOTE: THE VECTOR ADDRESSES CAN NOT BE JUMPED FOR LOCATIONS 200 THROUGH 220 AS THESE ARE USED BY THE PROGRAM)



## 2.3.2 NON-STANDARD DISKETTE ADDRESS SELECTION

IF IT IS DESIREABLE TO TEST THE DISKETTES BETWEEN TRACK AND SECTOR ADDRESS LIMITS OTHER THAN BETWEEN THE NORMAL OUTER DIAMETER (OD) AND INNER DIAMETER (ID) TRACK ADDRESSES, AND/OR MINIMUM (FIRST) AND MAXIMUM (LAST) SECTOR ADDRESS. THIS IS DONE BY THE OPERATOR MAKING CHANGES TO TWO (2) MEMORY LOCATIONS BEFORE THE PROGRAM IS STARTED. ONE LOCATION IS CALLED OD WHICH CONTAINS TWO BYTES ONE FOR OD AND THE OTHER ID TRACK ADDRESSES. THE OTHER LOCATION IS CALLED FIRST AND IT TOO CONTAINS TWO BYTES ONE FOR FIRST THE OTHER FOR LAST SECTOR ADDRESS. (IF THESE TWO LOCATIONS ARE LEFT CLEARED THE MAX AND MIN VALUES FOR THE ADDRESSES ARE ASSUMED.)

## A. DEFINITIONS:

OD = ADDRESS OF TRACK AT OUTER DIAMETER (MIN VALUE=0)  
 ID = ADDRESS OF TRACK AT INNER DIAMETER (MAX VALUE=114 [OCTAL])  
 FIRST = ADDRESS OF FIRST SECTOR OF TRACK (MIN VALUE=1)  
 LAST = ADDRESS OF LAST SECTOR OF TRACK (MAX VALUE=32 [OCTAL])

## B. LOCATIONS:

TRACKS LOC. 1200 BITS    14-----8    6-----0  
                                     ID                      OD

SECTORS LOC. 1202 BITS    12-----8    4-----0  
                                     LAST                      FIRST

## C. RESTRICTIONS:

THE CONTENTS OF "OD" MUST BE LESS THAN OR EQUAL TO THE CONTENTS OF "ID"  
 THE CONTENTS OF FIRST MUST BE LESS THAN OR EQUAL TO THE CONTENTS OF "LAST"

IF THESE LOCATIONS ARE CHANGED TO NEW LIMITS, THEN THE PROGRAM WILL ACCESS ONLY THOSE ADDRESSES INCLUSIVE OF AND BETWEEN THESE LIMITS.

## 2.3.3 SOFTWARE SWITCH REGISTER (LOC. 176)

FOR THE PDP 11 PROCESSORS THAT DO NOT HAVE A HARDWARE SWITCH REGISTER, OR IF THE OPERATOR WISHES TO SELECT THE SOFTWARE SWITCH REGISTER, BY PUTTING ALL THE SWITCHES UP TO A "1", LOCATION 176 IS ASSIGNED AS THE SWITCH REGISTER. BITS SET TO A "1" IN THIS LOCATION HAVE THE SAME FUNCTION AS THE CORRESPONDING SWITCH IN THE HARDWARE SWITCH REGISTER. ALL REFERENCES TO THE SWR ARE INDIRECT AND THE PROGRAM ASSIGNS THE CORRECT ADDRESS OF THE SWR AT "INITIAL START". SEE SECTION 2.4.2 FOR THE SELECTION OF OPERATING CONDITIONS.

NOTE: THE LOCATION 176 MUST BE SET UP BEFORE THE START OF THE PROGRAM AS THERE IS NO DYNAMIC CHANGE OF THIS LOCATION AVAILABLE.



2.3.4 TEST PARAMETER SELECTION ("DTESTP" LOC. 1212)

THE DRIVE AND TEST SELECTION MUST BE DONE BEFORE THE PROGRAM STARTS. "DTESTP" (LOCATION 1212) IS WHERE THE BITS ARE SET TO TELL THE PROGRAM WHAT DRIVES ARE WANTED AND WHAT TEST TO RUN, AS INDICATED BELOW. WHEN THE PROGRAM STARTS IT WILL TYPE OUT THE TEST CONDITIONS IT IS RUNNING UNDER.

BIT 15 (1) SELECT DRIVE UNIT 1  
 BIT 14 (1) SELECT DRIVE UNIT 0

THEN SET THE TEST CONDITIONS IN BITS 8 THROUGH 0 AS SHOWN BELOW.

"DTESTP" BITS 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0  
 UI UO NOT USED D P P P T T T S S S

BIT 9 IF BIT 9 IS ON, ALL WRITE/READ FUNCTIONS WILL BE IN THE DELETED DATA MODE.  
 BIT 8,7,6 SELECTS A DATA PATTERN TO BE USED.  
 BIT 5,4,3 SELECTS A TEST TO BE PERFORMED.  
 BIT 2,1,0 SELECTS A HEAD MOVEMENT SEQUENCE.

THE SELECTIONS ARE DEFINED AS FOLLOWS:

P = DATA PATTERN SELECTION

- 0 DEFAULT TO 7
- 1 ZEROS
- 2 ONES
- 3 FLOATING ZERO
- 4 FLOATING ONE
- 5 125
- 6 314
- 7 RANDOM

T = FUNCTIONAL TESTS

- 0 DEFAULT TO 7
- 1 WRITE ONLY
- 2 WRITE/READ
- 3 WRITE/READ CHECK
- 4 READ CHECK ONLY
- 5 READ ONLY (CRC CHECK)
- 6 WRITE/READ CHECK ON ALTERNATING DRIVES \*
- 7 WRITE/READ/READ CHECK \*\*

\* NOTE: TEST 6 WRITES THEN READ CHECKS ANY SELECTED DATA PATTERN USING ANY TRACK SEQUENCE, BUT ONE TRACK AT A TIME. FIRST ON UNIT 0 THEN UNIT 1. WHEN BOTH UNITS HAVE ACCESSED THAT TRACK, IT GOES BACK TO UNIT 0 FOR THE NEXT TRACK, ETC.



## NOTE: THE FIRST HALF OF TEST 7 FORCES THE TRACK SEQUENCE TO INCREMENT UP THROUGH ALL TRACKS DOING WRITE/READ CHECK FUNCTIONS. THIS VERIFIES THAT ALL TRACKS ARE ACCESSABLE. THE SECOND HALF OF THE PASS WILL USE THE SEQUENCE SELECTED BY THE OPERATOR AS INDICATED BELOW, AND ONLY READ CHECK THE DATA JUST WRITTEN. THIS VERIFIES THAT THE DATA CAN BE READ FROM A TRACK AFTER THE HEAD HAS BEEN MOVED AWAY FROM AND BACK TO THAT TRACK. AT THE COMPLETION OF THE PASS THE DELETED DATA BIT IN TEST CONDITIONS IS COMPLEMENTED AND THE NEXT PASS WILL BE RUN UNDER THIS NEW CONDITION.

S \* TRACK SEQUENCING

0	DEFAULT TO 7
1	INCREMENT
2	DECREMENT
3	INCREMENT/DECREMENT
4	BOUNCE
5	DECREASING BOUNCE
6	STROBE
7	RANDOM

IF NO BITS ARE SET (ZEROED "DTESTP:") THEN THE PROGRAM WILL SELECT ALL DRIVE UNITS THAT ARE READY AND DEFAULT TO TEST CONDITIONS AS INDICATED.

THE PROGRAM NEXT PRINTS THE REGISTER AND VECTOR ADDRESSES IT WILL USE IN COMMUNICATING WITH THE RX11 SYSTEM, AS EXPLAINED IN SECTION 2.3.1 THE OPERATOR MUST VERIFY THAT THESE ADDRESSES ARE THE ONES JUMPERED ON THE RX11 INTERFACE BOARD.

2.4 OPERATOR ACTION TO RUN THE PROGRAM

2.4.1 STARTING THE PROGRAM

SET THE DESIRED STARTING ADDRESS INTO THE SWITCH REGISTER, DEPENDING UPON THE TYPE OF CONSOLE AVAILABLE, LOAD ADDRESS, AND PRESS START.

THE PROGRAM WILL TYPE ITS "MAINDEC" NUMBER AND REVISION, AND DEPENDING UPON THE STARTING ADDRESS DO THE FOLLOWING:

SA200 - THE PROGRAM WILL TYPE DRIVE AND TEST PARAMETERS, THE TWO REGISTER ADDRESSES, AND VECTOR ADDRESS. IT WILL THEN BEGIN FUNCTIONAL TESTING, AND INFORMATION COLLECTION.

SA202 - THE PROGRAM WILL CONFIGURE TO CONDITIONS SET IN PREVIOUS "INITIAL START", PRINT OUT THESE CONDITIONS AND CONTINUE FUNCTIONAL TESTING AND DATA COLLECTING. THE ONLY OPERATOR ACTION REQUIRED IS THE DYNAMIC SELECTION OF OPERATING CONDITIONS IN THE SWITCH REGISTER AS REQUIRED.

SA204 - THIS PROGRAM WILL REPORT VIA THE TELEPRINTER ALL DATA COLLECTED. NO OTHER OPERATOR ACTION IS REQUIRED.



## 2.4.2 OPERATING CONDITIONS

THE PROGRAM CHECKS FOR OPERATING CONDITIONS AT VARIOUS POINTS WHILE RUNNING. IF THERE IS A HARDWARE SWR THESE CONDITIONS CAN BE CHANGED AND SET WHILE THE PROGRAM IS RUNNING. IF THE SOFTWARE SWR IS IN USE, THEN THESE CONDITIONS MUST BE SET IN LOCATION 176 BEFORE THE PROGRAM STARTS.

SW15 = HALT ON ERROR  
 SW14 = HALT AT END OF PASS  
 SW13 = DON'T PRINT ERROR MESSAGE  
 SW12 = TYPE ONLY 10 DATA ERRORS  
 SW11 = NO RETRY ON ERROR, LOG HARD ERROR  
 SW08 = NO RECALIBRATION ON SEEK ERRORS  
 SW15-SW0 (1) = SELECT SOFTWARE SWITCH REGISTER

NOTE: IF THERE IS A HARDWARE SWITCH REGISTER, AND THE OPERATOR WANTS THE SOFTWARE SWITCH REGISTER, PUT ALL SWITCHES UP (1) BEFORE STARTING THE PROGRAM AT THE INITIAL START ADDRESS.

THE PROGRAM WILL PRINT A "DRIVE(S)" SELECTED CONFIRMATION MESSAGE THAT IT WAS SUCCESSFUL IN FINDING AT LEAST ONE DRIVE READY (DRY) CONDITION ON AN OPERATOR SELECTED DRIVE OR EITHER OR BOTH IF NO SELECTION WAS MADE. THE DRIVES CONFIRMED AS BEING SELECTED BY THE PROGRAM MAY DIFFER FROM THAT SELECTED BY THE OPERATOR IF THE PROGRAM DETECTED A DRIVE TO BE NOT READY.

IF HOWEVER THERE ARE NO DRIVES IN THE DRIVE READY CONDITION THE PROGRAM WILL TYPE "NO DRIVES READY" AND HALT, AS IT CAN'T FUNCTION WITH NO DRIVES READY. WHEN THE REASON FOR ALL DRIVES TO BE NOT READY IS FOUND AND CORRECTED, THE OPERATOR MAY PRESS CONTINUE AND THE PROGRAM WILL GO BACK TO "INITIAL START" OR HE MAY RELOAD THE STARTING ADDRESS HIMSELF.

AS WELL AS "DRIVE(S)" SELECTED THE PROGRAM WILL TYPE OUT THE TEST PARAMETERS SELECTED OR DEFAULTED TO. IF NON-STANDARD TRACK AND/OR SECTOR ADDRESS LIMITS WERE SELECTED (SEE SECTION 2.3.2 OF THIS DOCUMENT) THESE LIMITS WILL ALSO BE PRINTED OUT FOR VERIFICATION BY THE OPERATOR

## 2.4.3 ACT11 &amp; XXDP HOOKS

THE PROGRAM HAS THE NECESSARY LOCATIONS SET UP FOR OPERATION UNDER ACT11 AND XXDP OPERATION. THE PROGRAM LOOKS AT THE LOADING MEDIA LOCATION AND IF IT CONTAINS THE NUMBER 10, INDICATING THE FLOPPY DISK LOADED THE PROGRAM, WILL TYPE THE FOLLOWING PROMPT MESSAGE & WAIT FOR A USER RESPONSE:

"CAUTION - IF YOU DESIRE TO TEST UNIT 0  
 REPLACE LOAD MEDIUM WITH A SCRATCH DISKETTE  
 THEN PRESS CONTINUE"

THUS, IF THE OPERATOR WISHES TO TEST DRIVE UNIT 0, AFTER LOADING THE PROGRAM, HE REMOVES THE LIBRARY DISKETTE FROM DRIVE 0, INSERTS A SCRATCH DISKETTE INTO DRIVE 0, AND PRESSES THE 'CONTINUE' SWITCH



## 2.5 PROGRAM OPTIONS

THERE ARE A COUPLE OF WAYS THE PROGRAM CAN BE SET UP TO RUN FOR OPTIONAL TESTING.

## 2.5.1 PSEUDO - SCOPE LOOP

BY SETTING THE ADDRESS LIMITS IN OD AND OR FIRST TO ONE OF TWO TRACKS AND OR SECTORS IT IS POSSIBLE TO PRODUCE A FAIRLY TIGHT PSEUDO - SCOPE LOOP. BY RUNNING ONLY A TEST THAT DOES WRITE ONLY OR READ ONLY ITS POSSIBLE TO SCOPE SPECIFIC FUNCTIONS, I.E. FILL BUFFER, WRITE, READ, EMPTY BUFFER.

## 2.5.2 DISKETTE COMPATABILITY

TO CHECK FOR DATA TRANSFER COMPATABILITY BETWEEN DRIVE UNITS USE THE FOLLOWING PROCEDURE. (IT IS ASSUMED THE SYSTEM HAS DUAL DRIVE RXD1. IF NOT YOU MAY TEST COMPATABILITY BETWEEN DIFFERENT RX11 SYSTEMS BY RUNNING THE SAME TESTS ON BOTH.) THIS TEST INSURES PROPER HEAD ALIGNMENT.

- A. HAVE DISKETTES IN BOTH DRIVES AND DRIVES READY.
- B. CLEAR THE OD/ID AND FIRST/LAST MEMORY LOCATION, FOR TOTAL DISKETTE TRANSFERS.
- C. LOAD THE INITIAL START ADDRESS (LOC 200) AND START THE PROGRAM RUNNING.
- D. WHEN ASKED FOR "DRIVE AND TEST CONDITIONS" SELECT BOTH DRIVES, ANY "P"ATTERN OF DATA, "S"EQUENCE #1 (INCREMENT TRACKS TO INSURE ALL HEAD POSITIONS ARE ALIGNED) AND "T"EST 3 (WRITE/READ CHECK). THIS WILL WRITE THEN READ AND VERIFY THE DATA ON BOTH DISKETTES.
- E. ALLOW THE PROGRAM TO RUN FOR AT LEAST 1 COMPLETE PASS.

NOTE: IF RANDOM PATTERN (0 OR 7) IS SELECTED YOU MUST HALT AT THE END OF THE FIRST PASS, AS ADDITIONAL PASSES CHANGES THE DATA AND YOU WILL GET DATA ERRORS WHEN YOU TRY TO REREAD.

TO HALT AT THE END OF PASS PUT SW14 UP (1) WHEN "OPERATING CONDITIONS" ARE REQUESTED BY THE PROGRAM.

- F. AFTER COMPLETION OF THE PASS, PROGRAM HALTED. SWAP DISKETTES, AND AS YOU ONLY WANT TO READ VERIFY THE DATA, START THE PROGRAM AGAIN AT LOC 200 (INITIAL START).
- G. WHEN REQUESTED SELECT THE SAME DRIVES, PATTERN, AND SEQUENCE, BUT SELECT TEST 4 (READ CHECK ONLY).
- H. ALLOW THE PROGRAM TO RUN AS LONG AS YOU WISH TO VERIFY THAT DATA WRITTEN AND CHECKED ON ONE DRIVE CAN BE READ AND VERIFIED ON THE OTHER DRIVE.



## 2.6 RUN TIME

RUN TIME PER PASS DEPENDS UPON NUMBER OF A FUNCTIONS IN THE TEST SELECTED AND THE TRACK SEQUENCE.

EXAMPLES OF RUN TIMES FOLLOW.

(THESE TIMES ARE FOR 1 DRIVE AND COMPLETE DISKETTE, MAXIMUM ID/OD, FIRST/LAST LIMITS, IF OPERATING ON 2 DRIVES DOUBLE THE TIME.)  
 THESE TIMES ARE FOR A PDP 11/05 PROCESSOR AND MAY CHANGE SLIGHTLY FOR A FASTER PROCESSOR.

TEST SELECTION	P	T	S	TIME/PASS
	7	6	5	3 MIN. 30 SEC.
	7	7	4	3 MIN. 47 SEC.
	7	7	7	1 MIN. 23 SEC.
	6	3	4	1 MIN. 46 SEC.
	7	3	1	1 MIN. 30 SEC.
	7	1	1	51 SEC.

NOTE: DUE TO THE SLOW SPEED OF THE LSI 11 PROCESSOR, THE RUN TIME IS ABOUT DOUBLE THAT LISTED ABOVE. TO SPEED UP THE RUNNING OF THIS PROGRAM IN THE LSI 11 YOU CAN CHANGE THE INTERLEAVE FACTOR, OF THE SECTORS, USED IN THE PROGRAM. TO DO THIS CHANGE THE CONTENTS OF LOCATION "THREE" FROM OCTAL 3 TO OCTAL 5.

## 3.0 ERROR DETECTION

## 3.1 PROGRAM DEFINITIONS

ON MOST ERRORS THE PROGRAM WILL TYPE OUT THE CONTENTS OF "STATUS A" AND "STATUS B".

STATUS A IS THE CONTENTS OF THE RXES (ERROR AND STATUS REGISTER) AT THE TIME THE ERROR IS DETECTED. IT SHOWS THE CRC, PAR, ETC. ERRORS.

STATUS B IS THE "DEFINITIVE ERROR CODES" THAT THE RX01 DETECTED, THAT MAY HAVE CAUSED THE ERROR CONDITION. THESE ERROR CODES ARE DEFINED IN SECTION 3.2.

THE PROGRAM HAS DEFINED THE FOLLOWING AS ERRORS.

## 3.1.1 WRITE ERROR

A WRITE ERROR IS A RETRIED READ ERROR IF THE DATA BEING READ IS OF UNKNOWN QUALITY (THE DATA READ IS BEING READ FOR THE FIRST TIME AFTER A WRITE FUNCTION)

## 3.1.2 READ (CRC) ERROR

A READ ERROR IS A RETRIED READ ERROR WHERE THE QUALITY OF THE DATA BEING READ IS KNOWN GOOD. (THE DATA HAS BEEN READ CORRECTLY SOME TIME PREVIOUSLY.)



## 3.1.3 CRC AND DATA ERROR

## 3.1.4 NO CRC ERROR BUT DATA ERROR

## 3.1.5 CRC ERROR BUT NO DATA ERROR

THE ABOVE THREE ERROR ARE DETECTED WHEN THE PROGRAM IS VERIFYING THE DATA READ OFF THE DISKETTE AGAINST THE DATA THAT SHOULD HAVE BEEN READ.

UPON A NON-COMPARISON THE PROGRAM TYPES OUT THE "BYTE" NUMBER IN THE SECTOR, THE DATA READ FROM THE DISKETTE "BAD" AND THE EXPECTED DATA "GOOD".

BYTES      BAD      GOOD  
(THE DATA PATTERNS ARE FORMATTED AS SHOWN)

0      (TRACK ADDRESS; BITS 6 - 0)  
1      (SECTOR ADDRESS; BITS 4 - 0)

BYTES 2 THROUGH 125 CONTAIN THE SELECTED "P"ATTERN.

126      (THE SUM OF ALL BYTES 0 - 125)  
127      THE NEGATIVE OF 2 TIMES BYTE 126)

THE PROGRAM DETECTS A CHECKSUM ERROR BY SUMMING ALL THE DATA READ FROM THE DISKETTE AND COMPARING THAT SUM TO 0.

AT THE END OF THE DATA ERROR TYPEOUT THE PROGRAM TYPES OUT IF THE CHECK SUM ACCUMULATED IS "GOOD" OR HAD "ERRORS". IF BYTES 0 OR 1 HAVE DATA ERRORS THE OPERATOR MUST CHECK THE RESULTS OF THE CHECK SUM. IF IT IS ALSO BAD THEN THERE WAS A TRUE DATA ERROR. IF THE CHECK SUM IS GOOD THEN IT MIGHT BE THAT THE HEAD IS NOT OVER THE TRACK EXPECTED, AND THERE IS A POSITIONING ERROR.

## 3.1.6 SEEK ERROR

A SEEK ERROR IS DEFINED AS NOT A CRC AND NOT A PARITY ERROR. A PROGRAMED RECALIBRATE IS ISSUED TO TRY TO CORRECT EACH SEEK ERROR.

## 3.1.7 PARITY ERROR

A PARITY ERROR RESULTS FROM AN INCORRECT TRANSFER OF A COMMAND WORD FROM THE RX11 INTERFACE TO THE RX01 MICRO-PROCESSOR CONTROLLER.



### 3.2 DEFINITIVE ERROR CODES

THE RXD1 MICROCONTROLLER HAS DEFINED THE ERROR CODES AND MEANINGS WHICH ARE AVAILABLE TO THE PROGRAM BY ISSUING COMMAND #7 "READ THE B-CODE STATUS REGISTER".

A DEFINITIVE ERROR CODE REPRESENTS (WHERE) WITHIN A FUNCTION AN ERROR WAS DETECTED.

THE FOLLOWING ARE THE DEFINITIVE ERROR CODES AND MEANINGS:

- 00 - NO ERROR
- 10 - DRIVE 0 FAILED TO SEE HOME FROM INITIALIZE
- 20 - DRIVE 1 FAILED TO SEE HOME FROM INITIALIZE
- 30 - HOME FOUND WHEN STEPPING OUT 10 TRACKS FROM INIT
- 40 - TRIED TO ACCESS A TRACK GREATER THAN /7(DECIMAL)
- 50 - HOME WAS FOUND BEFORE DESIRED TRACK
- 60 - SELF DIAGNOSTIC ERROR
- 70 - DESIRED SECTOR NOT FOUND AFTER SAMPLING 52 HEADERS
- 100 - WRITE PROTECT ERROR
- 110 - MORE THAN 40US AND NO SEP CLOCK DETECTED
- 120 - A PREAMBLE COULD NOT BE FOUND
- 130 - PREAMBLE FOUND BUT NO ID MARK FOUND IN TIME
- 140 - CRC ERROR ON SUPPOSIDLY GOOD HEADER
- 150 - GOOD HEADER (NO CRC ERROR) BUT TRACK COMPARE ERROR
- 160 - IDAM NOT FOUND IN TIME
- 170 - DATA AM NOT FOUND IN TIME
- 200 - DATA CRC ERROR
- 210 - ALL PARITY ERRORS

### 3.3 UNEXPECTED OR MISSING ERROR CONDITIONS

#### 3.3.1 MISSING DD MARK

AN ERROR WHEN THE PROGRAM WROTE DELETED DATA INFORMATION BUT NO DELETED DATA MARK WAS DETECTED WHEN THE DATA WAS READ.

#### 3.3.2 UNEXPECTED DD MARK

AN ERROR WHEN A DELETED DATA MARK IS DETECTED BUT NO DELETED DATA WAS WRITTEN.



**3.3.3 NO INTERRUPT ON DONE**

THE INTERRUPT ENABLE BIT WAS SET AND THE DONE FLAG WAS SET BUT NO INTERRUPT OCCURRED.

**3.3.4 UNKNOWN INTERRUPT**

IF AN INTERRUPT OCCURS FROM ANY OTHER DEVICE WHILE THIS PROGRAM IS RUNNING IT WILL HALT AT THE INTERRUPT VECTOR LOCATION.

IF AN INTERRUPT OCCURS ON INTERRUPT VECTOR LOCATION 264 (RX11), AND THERE IS NO ERROR, DONE, OR ERROR STATUS CONDITION SET, THEN IT WILL BE TAGGED AS AN UNKNOWN INTERRUPT.

**3.4 POWER FAILURE**

THE PROGRAM TESTS FOR TWO TYPES OF POWER FAILURE, TOTAL SYSTEM POWER LOSS AND RX11 POWER LOSS RESULTING IN A RECALIBRATION OF THE DRIVES.

THE TOTAL SYSTEM POWER FAILURE IS DETECTED BY THE "SYSMAC" SUBROUTINE .SPOWER. WHEN THE POWER IS DETECTED TO BE GOING DOWN, THE REGISTERS ARE SAVED. WHEN THE POWER COMES BACK UP THE REGISTERS ARE RESTORED AND THE MESSAGE "POWER" IS PRINTED. THE PROGRAM THEN AUTOMATICALLY DOES A RESTART.

LOSS OF POWER IN THE RX11 CAUSES A RECALIBRATION OF ALL DRIVES. WHEN THIS HAPPENS, THE "INIT DONE" BIT IS SET IN THE THE RXES REGISTER ALONG WITH THE NORMAL "DONE" FLAG. AT EACH INTERRUPT THE PROGRAM TESTS FOR INIT DONE. IF IT IS FOUND TRUE, THE FUNCTION WAS NOT COMPLETED AND A POWER LOSS MUST HAVE BEEN DETECTED. WHEN THIS HAPPENS THE PROGRAM TYPES OUT "RX11 POWER" AND DOES AN AUTOMATIC RESTART.

IF THERE ARE REPEATED, NOTHING ELSE HAPPENS BUT, RX11 POWER MESSAGES THE INIT DONE FLAG MIGHT BE STUCK ON.

**3.5 PROGRAM HUNG**

THERE ARE MANY PLACES WHERE THE PROGRAM MUST WAIT FOR AN OPERATION TO BE COMPLETED IN THE RX01. THESE ARE WAITING FOR THE "DONE" FLAG TO INDICATE A FUNCTION IS COMPLETED, OR WAITING FOR A TRANSFER REQUEST "TR" FLAG TO SEND OR RECEIVE THE NEXT BYTE OF INFORMATION. IF THE RX11 DOES NOT COME BACK WITH EITHER OR BOTH OF THESE FLAGS THE PROGRAM WOULD HANG UP.

TO INHIBIT THIS FROM HAPPENING THERE ARE TWO SUBROUTINES USED TO CHECK FOR THESE TWO FLAGS.

THE "DONECK" LOOKS FOR THE DONE FLAG AND IF IT IS NOT SEEN WITHIN A SPECIFIC TIME THE "TEST HUNG" MESSAGE IS PRINTED AND THE PROGRAM HALTS. IN REGISTER 3 (177703) IS THE RETURN ADDRESS OF THE TEST THAT IS WAITING FOR THE DONE FLAG.



"TRK" LOOKS FOR THE "TR" FLAG. IF IT IS NOT SEEN WITHIN A SPECIFIC TIME IT ALSO PRINTS THE "TEST HUNG" MESSAGE AND HALTS. IN THE SP (177706) IS THE RETURN ADDRESS OF THE TEST WAITING FOR THE TR FLAG.

THE WAITING TIME FOR THE TWO FLAGS DEPENDS UPON THE HOST PROCESSOR AS INDICATED BELOW:

"DONE" WAIT	11/45 BIPOL 14 SEC.
	11/05 CORE 62 SEC.
"TR" WAIT	11/45 BIPOL .44 SEC.
	11/05 CORE 2 SEC.

4.0 ERROR REPORTING

ALL ERRORS DETECTED WILL BE REPORTED IF SW13 = 0 FOR OPERATING CONDITIONS. IF SW12 = 1 THEN ONLY 10 DATA ERRORS FOR 1 SECTOR WILL BE REPORTED, AND A TOTAL OF DATA ERRORS FOR THAT SECTOR WILL BE REPORTED AT THE END OF THE SECTOR.

THE END OF PASS INDICATOR TYPED, ALSO INDICATES WHETHER ANY ERRORS OCCURED DURING THAT PASS. IF THERE WERE NO ERRORS THEN A "\*" IS PRINTED. IF THERE WERE ERRORS THEN A "-" IS PRINTED. THE TOTAL ACCUMULATED ERRORS AND SYSTEMS OPERATION IS REPORTED BY THE "ERROR DUMP" PROGRAM.

5.0 HALTS

THERE ARE VARIOUS HALT LOCATIONS THROUGHOUT THE PROGRAM. SOME ARE THE RESULTS OF "HARD" ERRORS OTHERS ARE WAITING FOR INTERVENTION. THEY ARE LISTED BELOW:

HALT #	TYPE	DEFINITION
HLT1:	NO ERROR	;CAUTION - LOAD MEDIUM ON UNIT 0
HLT3:	ERROR	;ERRORS FOUND ON RECAL FUNCTIONS
HLT4:	ERROR	;HARD PARITY ERRORS
HLT5:	ERROR	;NO DRIVES READY
HLT6:	ERROR	;SW15 SET ON PARITY ERROR
HLT7:	ERROR	;SW15 SET ON SEEK ERROR
HLT10:	ERROR	;SW15 SET ON CRC GENERATOR ERROR (NO DATES ERROR)
HLT11:	ERROR	;HARD CRC GENERATOR ERROR (NO DATA ERRORS)
HLT12:	ERROR	;SW15 SET ON DATA CRC ERROR
HLT13:	ERROR	;SW15 SET ON MISSING DETECTED DATA ERROR
HLT14:	ERROR	;SW15 SET ON DATA NO CRC ERROR
HLT15:	ERROR	;SW15 SET ON MISSING INTERRUPT AT DONE
HLT16:	NO ERROR	;HALT AT END OF PASS
HLT17:	NO ERROR	;HALT AT END OF ERROR DUMP
HLT20:	ERROR	;PROGRAM HUNG WAITING FOR DONE
HLT21:	ERROR	;PROGRAM HUNG WAITING FOR TR FLAG



FLOW CHART  
\*\*\*\*\*  
RX11 RELIABILITY TEST  
\*\*\*\*\*

COPYRIGHT 1975  
DIGITAL EQUIPMENT CORPORATION  
MAYNARD, MASS. 01754



TABLE OF CONTENTS  
\*\*\*\*\*

PAGE 25	FLOPPY WRITE SEQUENCE
PAGE 52	SUBROUTINE HOME
PAGE 54	SUBROUTINE TRKERR
PAGE 55	SUBROUTINE INITSE(CTOR)
PAGE 56	SUBROUTINE GETSEC(TOR)
PAGE 57	SUBROUTINE ADJSUM
PAGE 58	SUBROUTINE TRCK
PAGE 59	SUBROUTINE COMM0(RD)
PAGE 61	SUBROUTINE DONECK
PAGE 62	ICOND
PAGE 63	SUBROUTINE UILOG
PAGE 64	SUBROUTINE SEKTYP
PAGE 65	SUBROUTINE STATER
PAGE 66	SUBROUTINE DOCHK
PAGE 69	SUBROUTINE EMPBUF(F)
PAGE 73	SUBROUTINE INTSER(V)
PAGE 81	ROCODE
PAGE 82	FLOPPY READ SEQUENCE
PAGE 83	SUBROUTINE GETPAT(TERN)
PAGE 84	SUBROUTINE INITTR(ACKS)
PAGE 85	SUBROUTINE GETTR(ACK)
PAGE 86	SUBROUTINE GETUNI(T)
PAGE 87	FLOPPY READCHECK SEQUENCE
PAGE 88	SUBROUTINE STOP

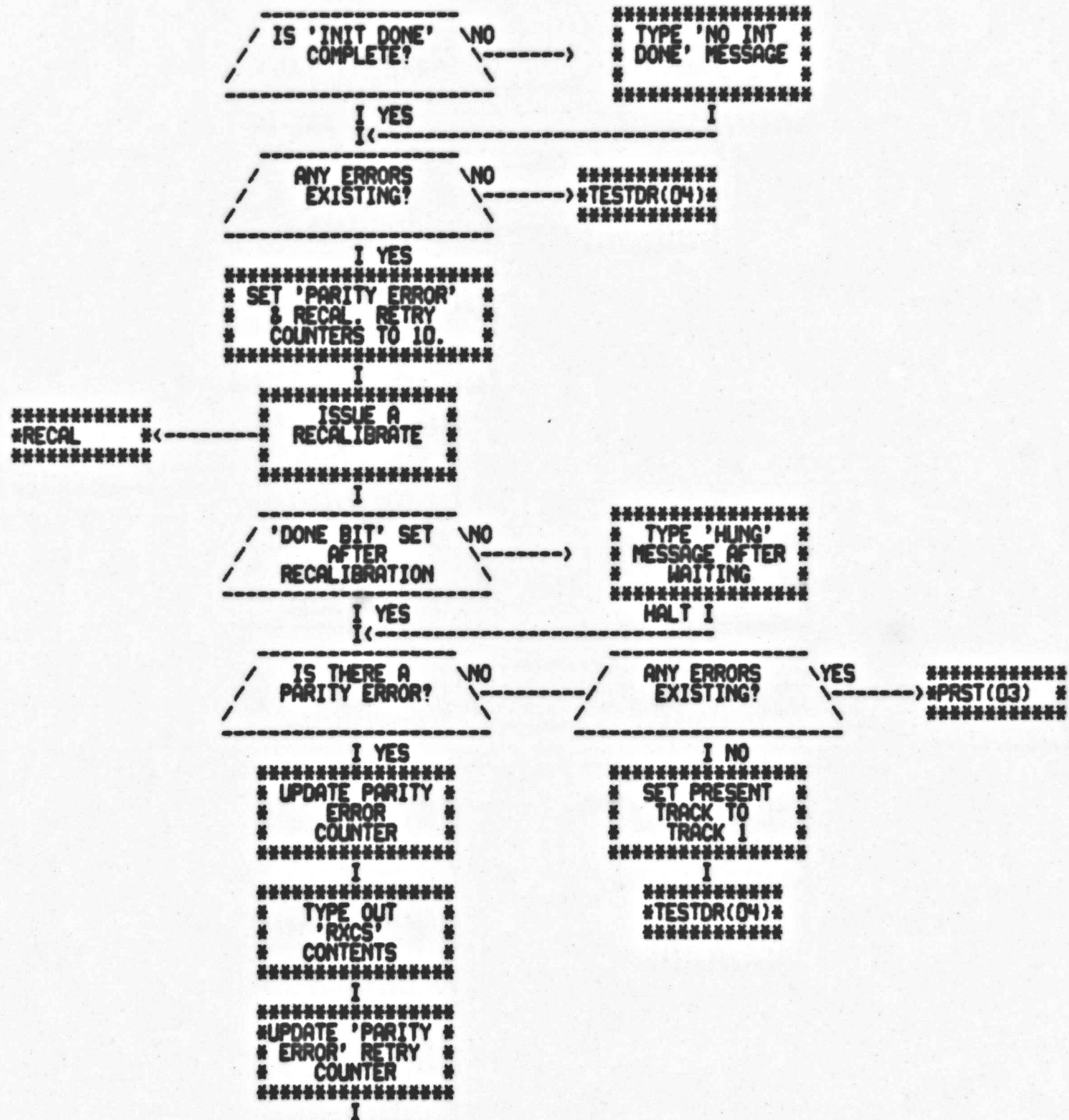


```

*****
#START(03) #
*****
I
*****
# INITIALIZE #
#DEVICE VECTORS #
# 264 & 266 #
*****
I
*****
# INITIALIZE #
# ALL COUNTERS #
# & FLAGS #
*****
I
*****
# ASK USER TO SELECT #
# DRIVE AND TEST #
# CONDITIONS #
*****
I
*****
# 'HALT' #
# WAIT FOR USER #
# TO REPLY #
*****
I
*****
# TYPE: MAINDEC NAME #
#DEVICE 'CSR' & 'DBR' #
# DEVICE VECTOR ADD. #
*****
I
*****
# ASK USER TO SELECT #
# OPERATING #
# CONDITONS #
*****
I
*****
# 'HALT' #
# WAIT FOR USER #
# TO REPLY #
*****
I
-----
/ 'DONE' BIT SET \ NO
/ AFTER \----->
/ RECALIBRATION? \
-----
I YES
I<-----
*****
# TYPE 'HUNG' #
# MESSAGE AFTER #
# WAITING #
*****
HALT I

```







```

-----
/ DO WE STILL \ YES
/ HAVE A PARITY \----->
/ ERROR AFTER IOX? \
-----

```

```

I NO
*****
# TYPE 'PARITY #
# ERROR' MESSAGE #
#
*****
I
*****
# RECAL #
*****

```

```

*****
# UPDATE 'HARD' #
# PARITY ERROR #
# LOG #
*****
I
*****
# TYPE #
# 'UNRECOVERABLE #
# PE' MESSAGE #
*****
I HALT
*****
# START(01) #
*****

```

```

*****
# PRST #
*****
I
*****
# PRINT CONTENTS #
# OF STATUS #
# REGISTERS #
*****
I

```

```

-----
/ IS 'ERROR' CODE \ NO
/ DUE TO A \----->
/ RECALIBRATION? \
-----

```

```

I YES
*****
# UPDATE RECALIBRATION #
# RETRY COUNTER #
#
*****
I

```

```

*****
# SET PRESENT #
# TRACK TO #
# TRACK I #
*****
I
*****
# TESTDR(04) #
*****

```

```

-----
/ HAVE WE \ NO
/ ATTEMPTED THE \----->
/ RECALIBRATION IOX? \
-----

```

```

I YES
*****
# TYPE #
# 'CAN'T RECALIBRATE' #
# MESSAGE #
*****
I HALT
*****
# START(01) #
*****

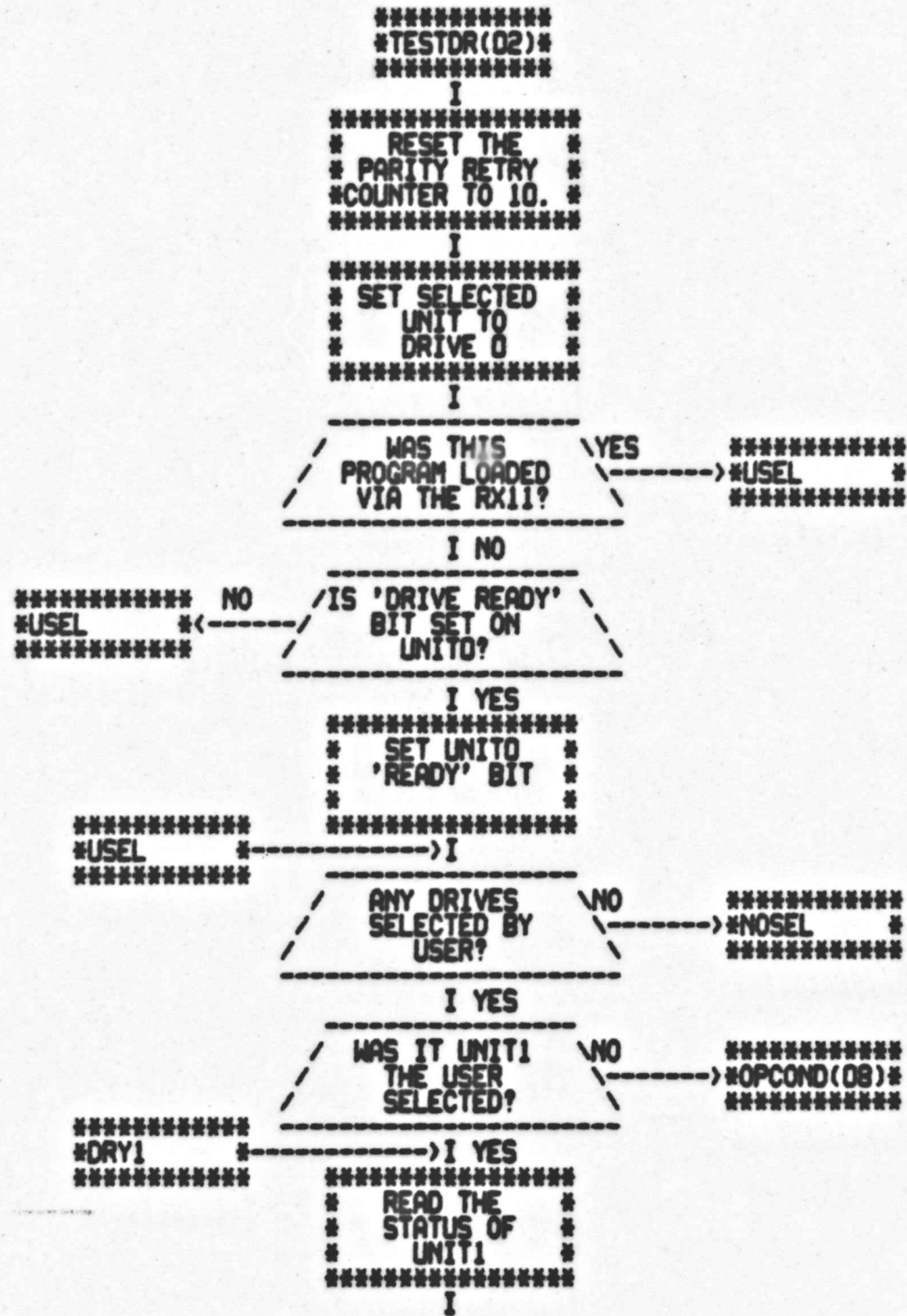
```

```

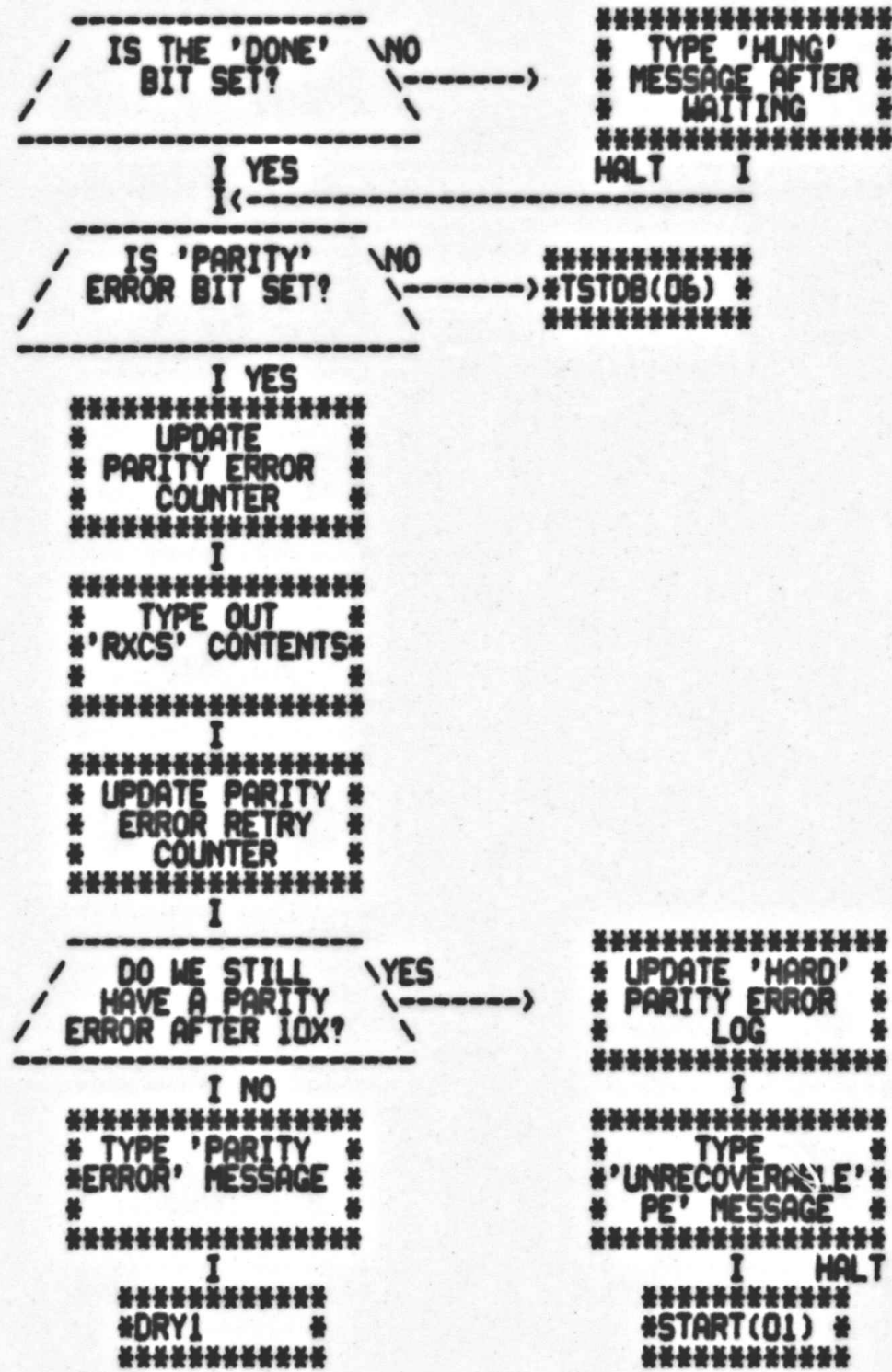
*****
# RECAL #
*****

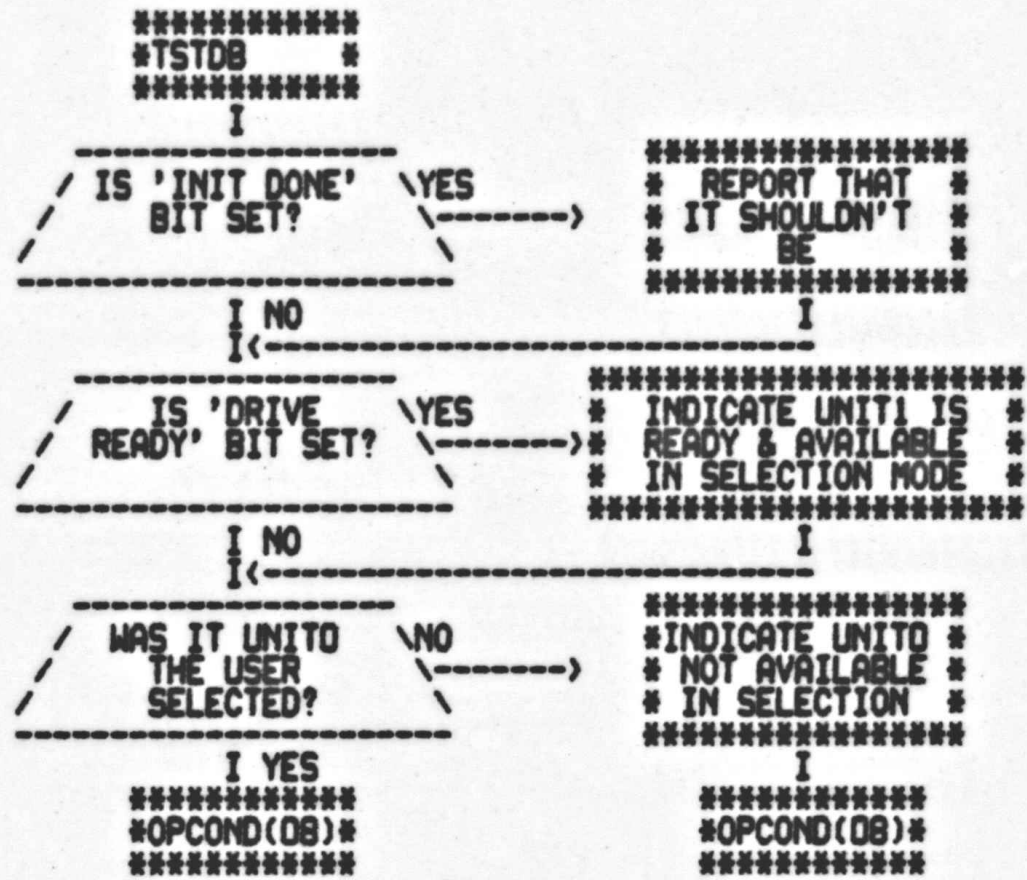
```









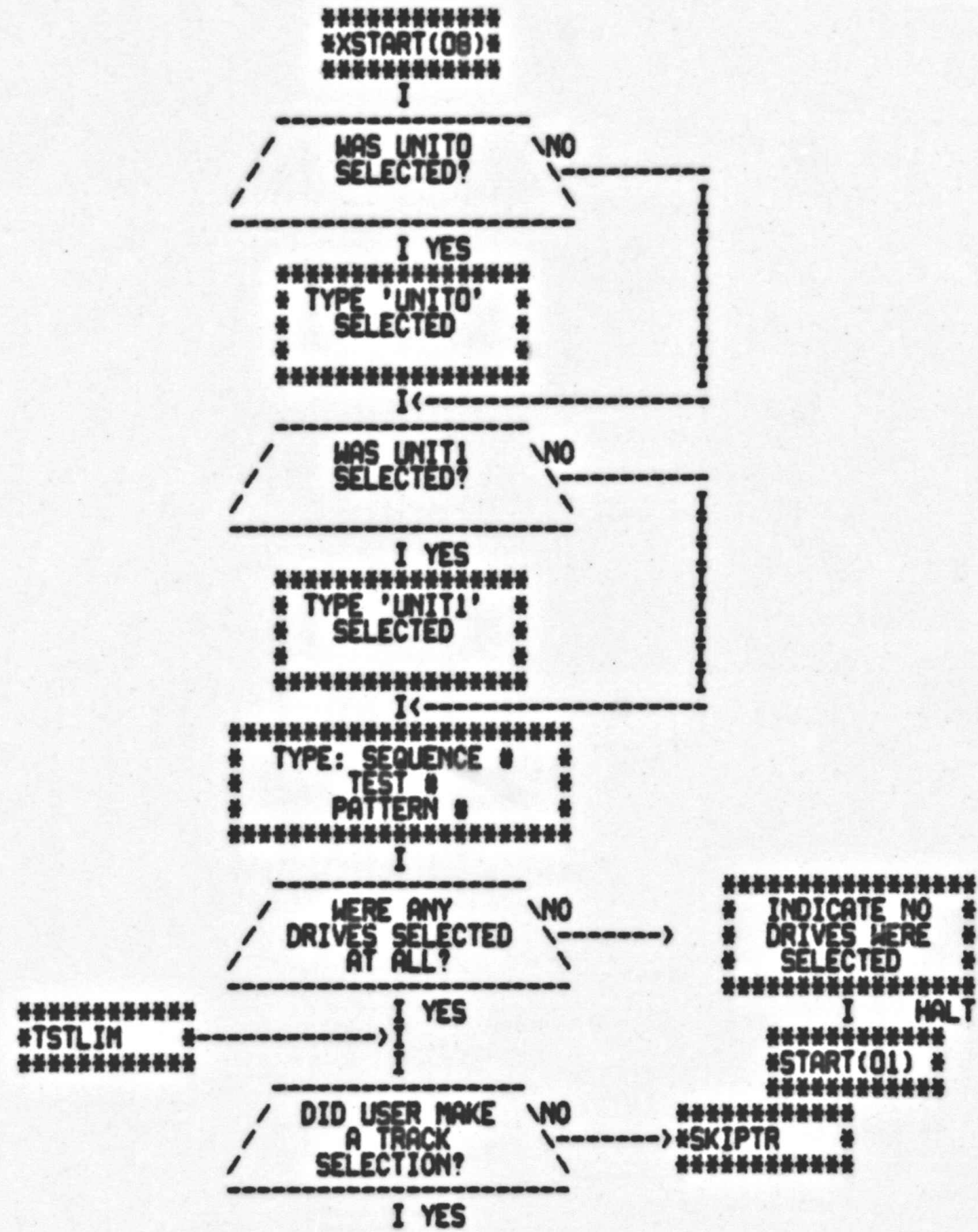


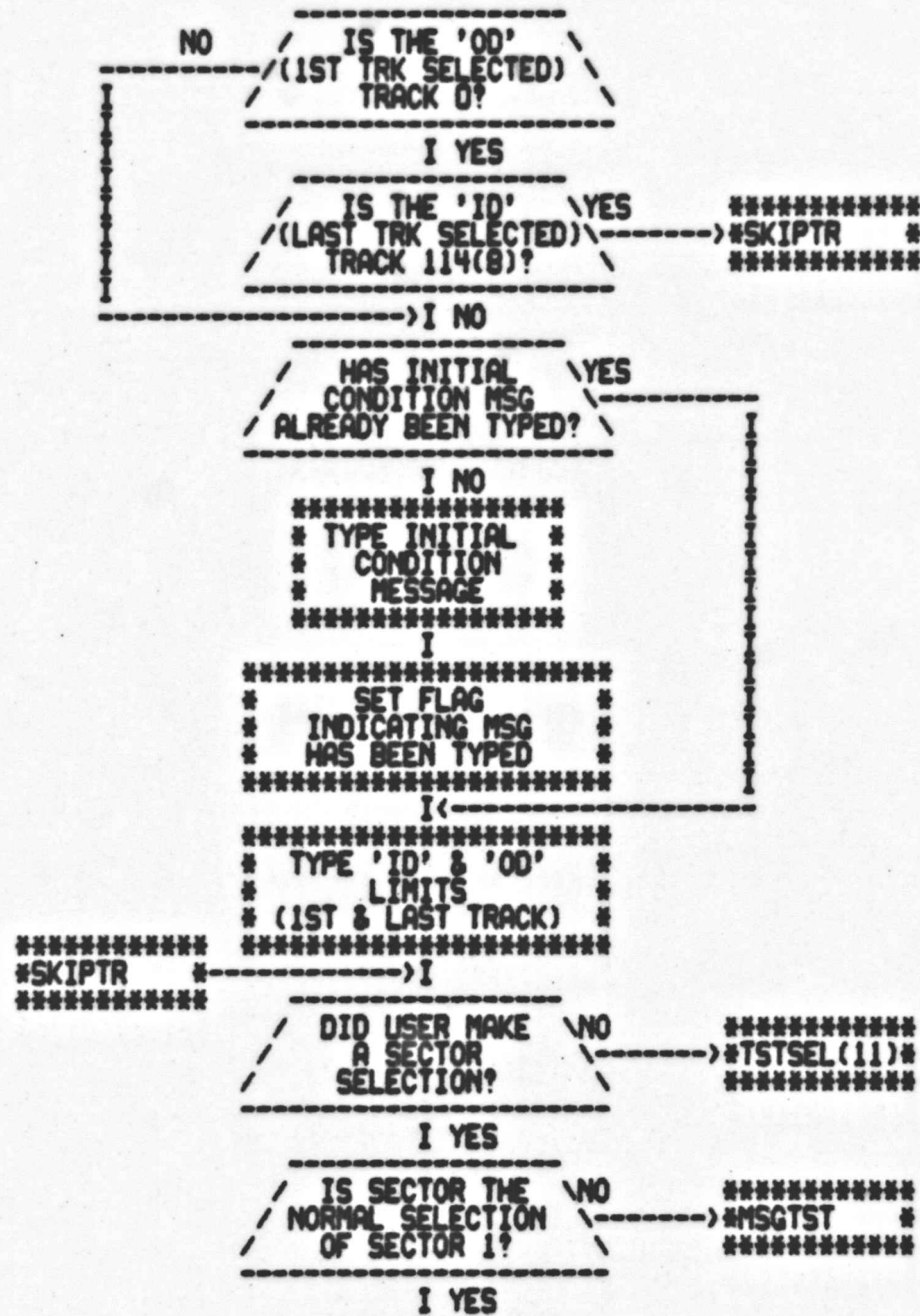




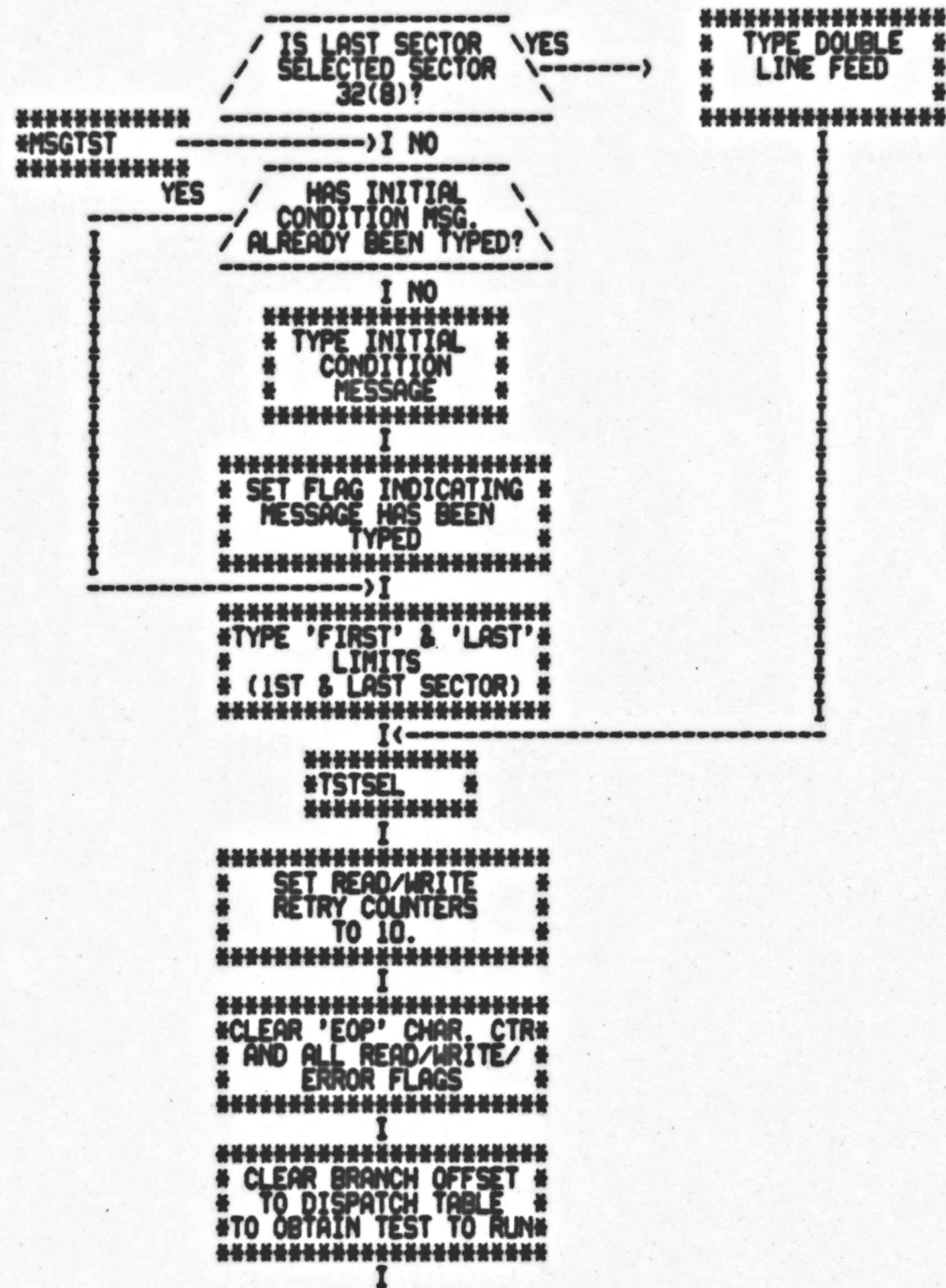
```
*****  
#OPCOND(06)#  
*****  
I  
*****  
# STORE USER #  
# INPUT TEST #  
# CONDITIONS #  
*****  
I  
*****  
# PULL OFF TEST #  
# SEQ #  
# TO BE RUN #  
*****  
I  
*****  
# PULL OFF TEST #  
# TO BE RUN #  
# #  
*****  
I  
*****  
# PULL OFF #  
# PATTERN # #  
# TO BE RUN # #  
*****  
I  
*****  
#XSTART(09)#  
*****
```











```

-----
/DID USR NOT SELCT YES
/ A TEST TO RUN/OR \----->
/DID HE SELECT T=7? \
-----
I NO
*****
# CALCULATE PROPER #
#BRANCH OFFSET TO GET #
# TO SELECTED TEST #
*****
I
*****
# 'JMP' TO #
# APPROPRIATE #
# TEST! #
*****
*****
# FORCE T=7 THEREBY #
# RUNNING WRITE/READ/ #
# READ CHECK TESTING #
*****
I
*****
#TEST7(13) #
*****

```

NOTE: THE APPROPRIATE TEST IS TO ONE OF THE FOLLOWING:

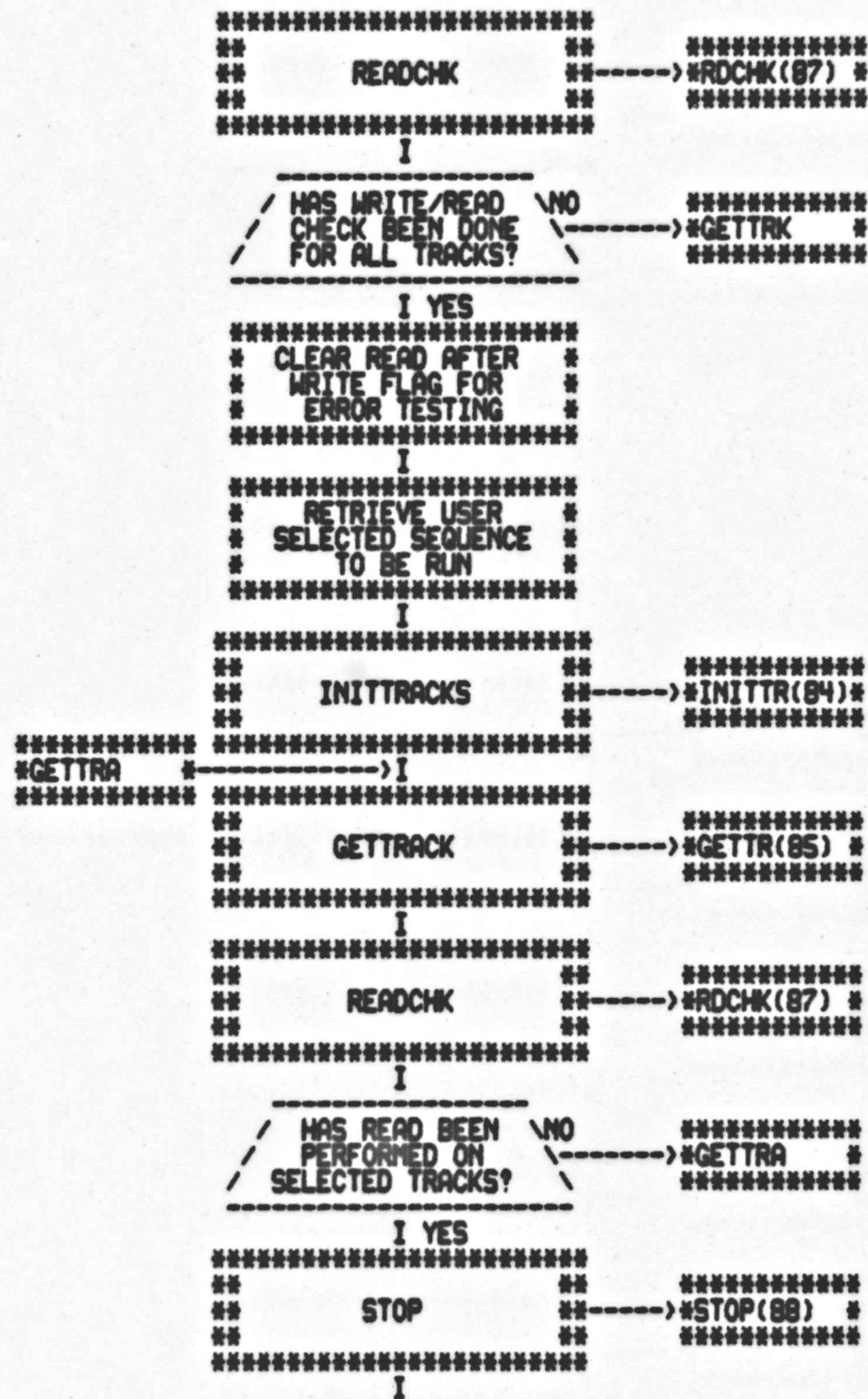
- \*\*\*\*\*  
#TEST7(13) # T=0/7; WRITE/READ/READ CHECK  
\*\*\*\*\*
- \*\*\*\*\*  
#WTRDCK(16) # T=3 ; WRITE/READ CHECK  
\*\*\*\*\*
- \*\*\*\*\*  
#CKREAD(18) # T=4 ; READ CHECK ONLY  
\*\*\*\*\*
- \*\*\*\*\*  
#WTRD(19) # T=2 ; WRITE/READ  
\*\*\*\*\*
- \*\*\*\*\*  
#WRTONL(21) # T=1 ; WRITE ONLY  
\*\*\*\*\*
- \*\*\*\*\*  
#RONLY(22) # T=5 ; READ ONLY  
\*\*\*\*\*
- \*\*\*\*\*  
#DRVSMP(23) # T=6 ; WRITE/READ CHECK ON ALTERNATE  
\*\*\*\*\*  
; DRIVES



```

T=0      T=7
*****
#TEST7(12) #
*****
      I
*****
**          **          *****
**  GETPATTERN  **----->#GETPAT(83)#
**          **          *****
*****
      I
*****
# SAVE ORIGINAL #
# TRACK SEQUENCE #
#   TO BE RUN   #
*****
*****
#TEST7X #----->I
*****
*****
# INITIALLY, FORCE AN #
# 'INCREMENT' SEQUENCE #
# FOR 1ST HALF OF TEST7#
*****
      I
*****
# SET READ AFTER #
# WRITE FLAG FOR #
# ERROR TESTING  #
*****
      I
*****
**          **          *****
**  INITTRACKS  **----->#INITTR(84)#
**          **          *****
*****
      I
*****
**          **          *****
**  GETUNIT     **----->#GETUNI(86)#
**          **          *****
*****
*****
#GETTRK #----->I
*****
*****
**          **          *****
**  GETTRACK    **----->#GETTR(85) #
**          **          *****
*****
      I
*****
**          **          *****
**  WRITE       **----->#WRITE(25) #
**          **          *****
*****
      I

```





```

-----
/ HAVE ALL UNITS \ NO
/ SELECTED BEEN  \-----> *TEST7X *
/ TESTED?        \
-----

```

```

I YES
*****
# RESET STACK #
# POINTER     #
#             #
*****

```

```

-----
/ WAS DELETED   \ NO
/ DATA BIT SET \----->
/ AS A TEST    /
/ CONDITION?   \
-----

```

```

I YES
*****
# CLEAR DELETED #
# DATA BIT IN  #
# TEST CONDITION #
*****

```

```

*****
#TEST7X *
*****

```

```

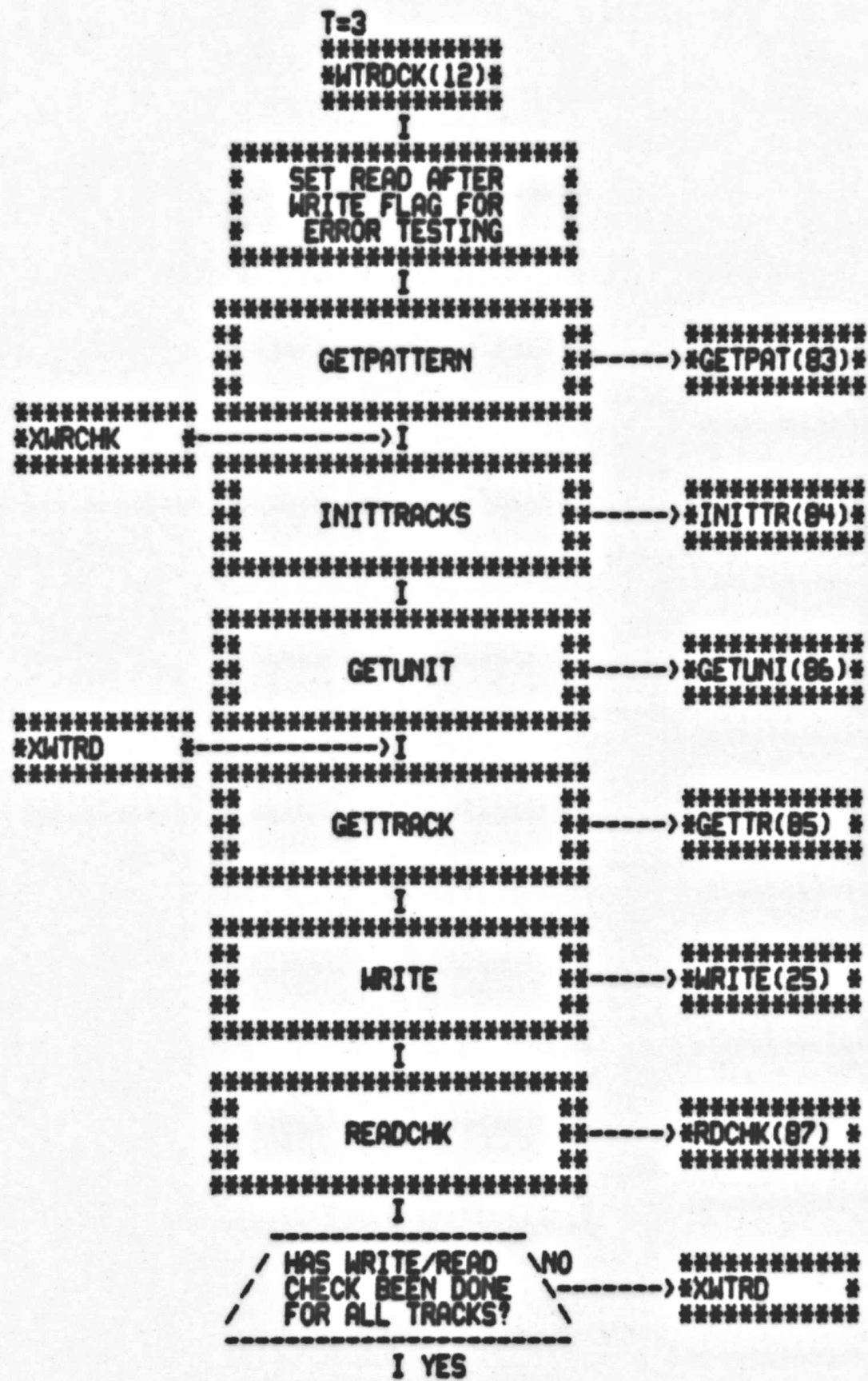
*****
# SET DELETED #
# DATA BIT IN #
# TEST CONDITION #
*****

```

```

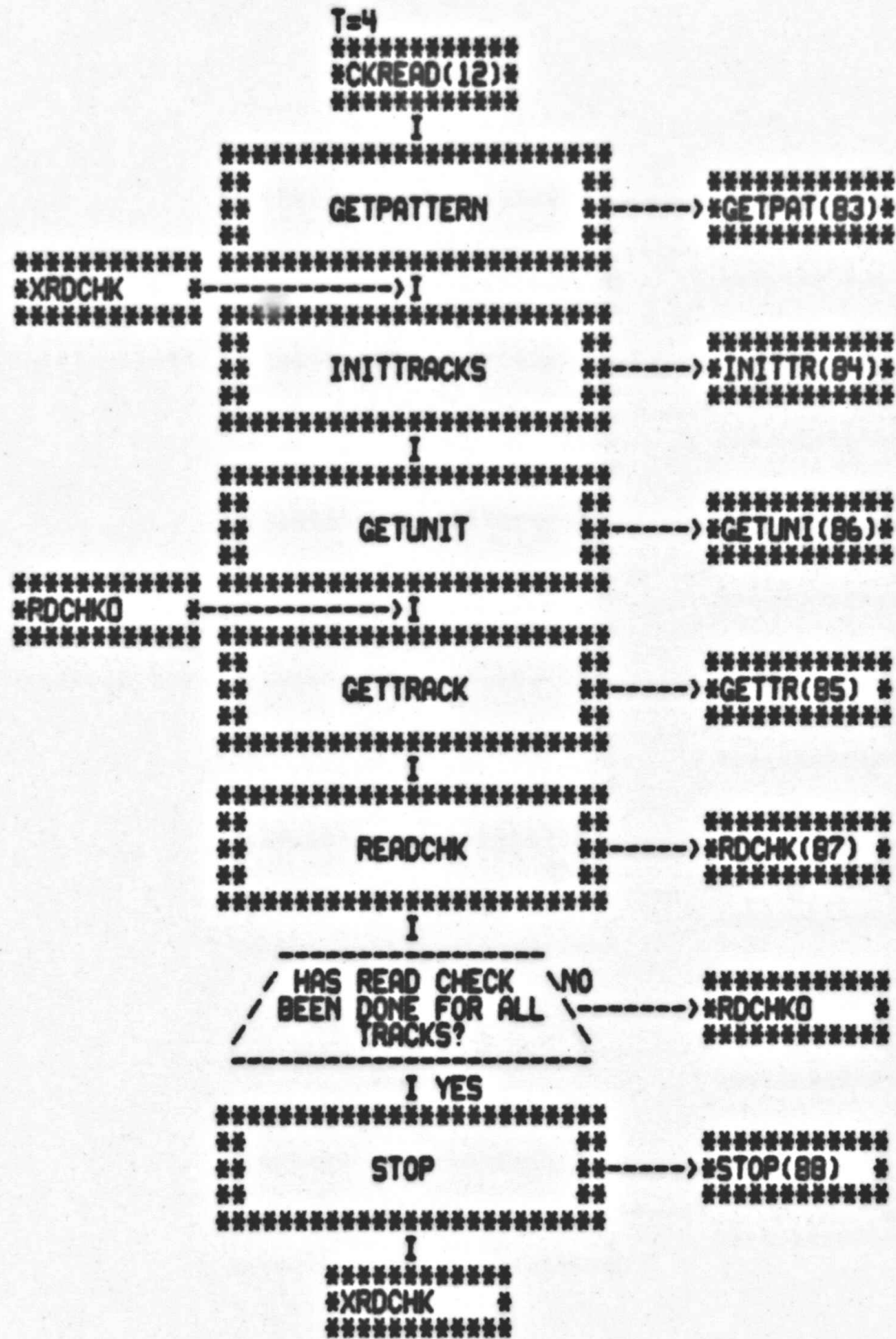
I
*****
#TEST7X *
*****

```

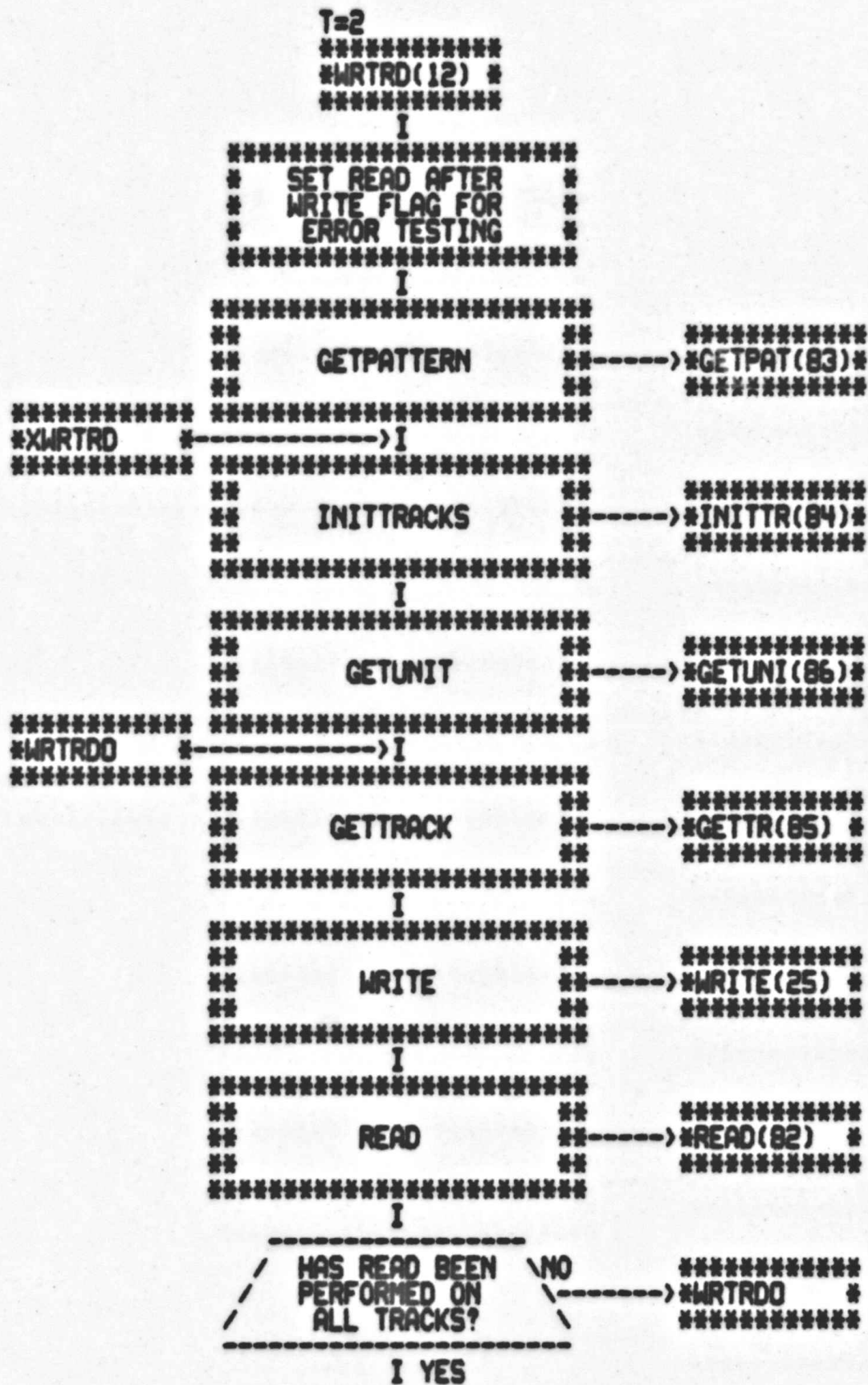




```
#####
#                #
#          STOP          #----->#STOP(88) #
#                #                #
#####
#
#XWRCHK #
#####
```

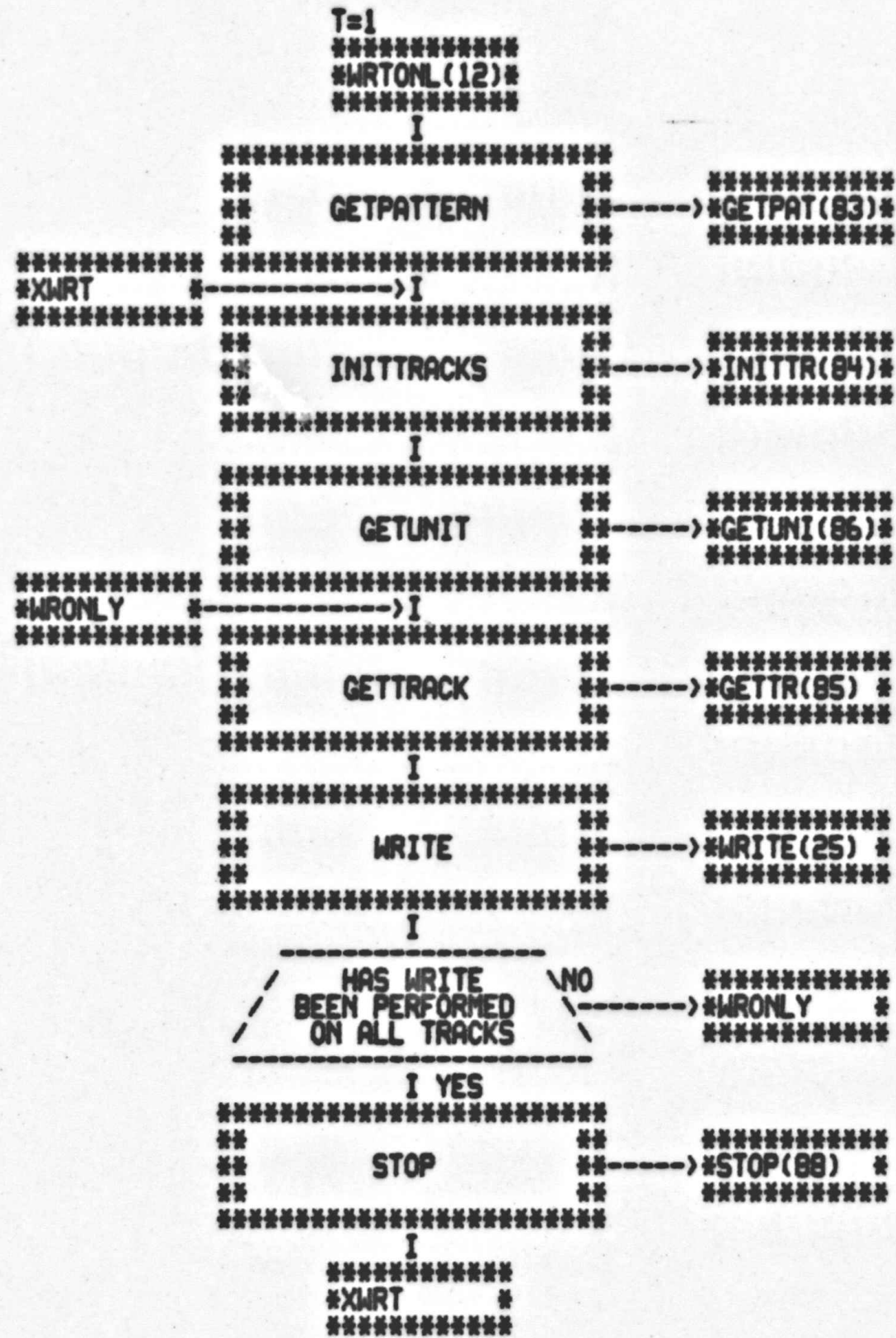


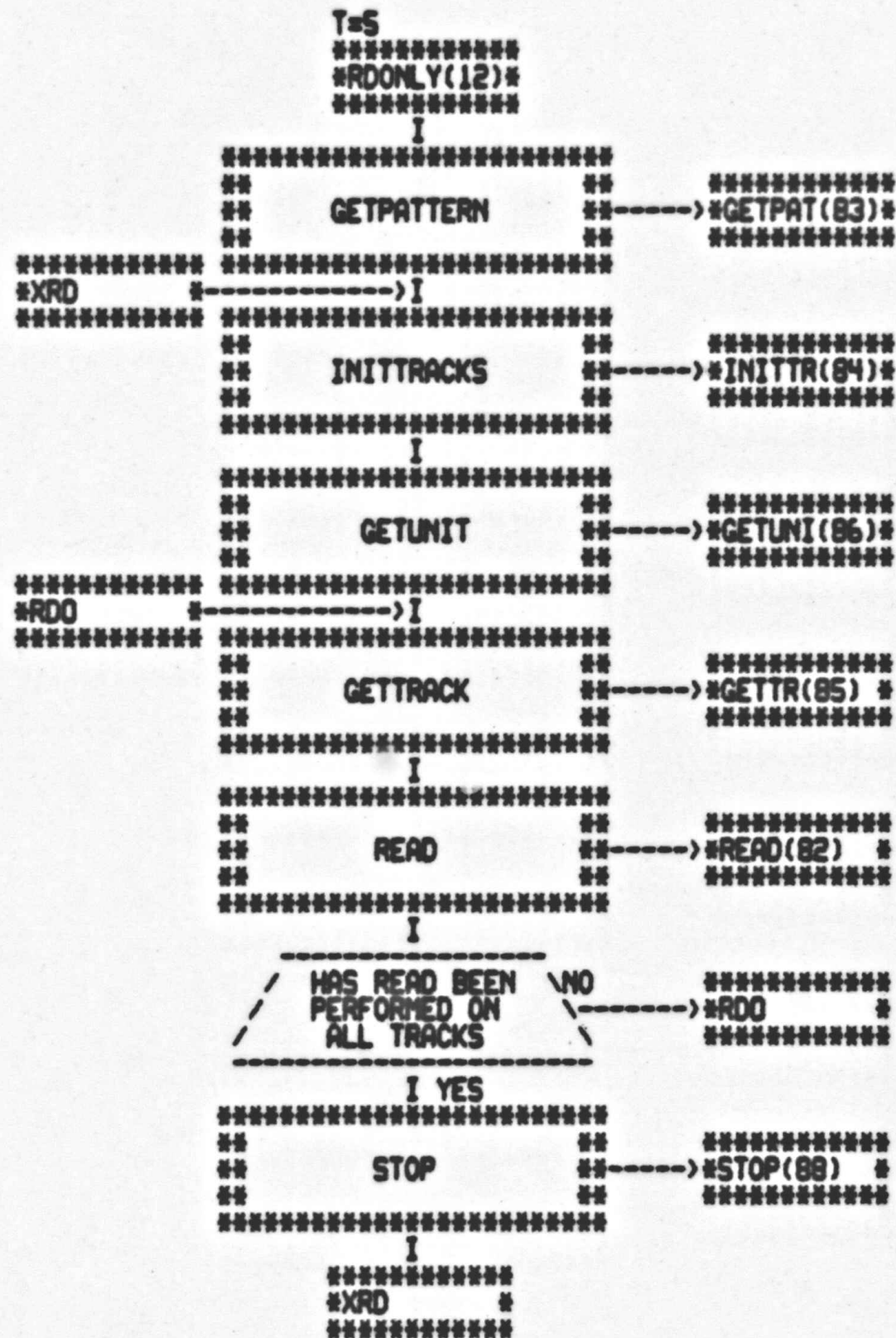




```
*****  
**          STOP          **----->*****  
**          STOP(88)     **          *  
**          *****  
*****  
          I  
*****  
**XWTRD **  
*****
```









```

T=6
*****
#DRVSMP(12)*
*****
I
*****
# SET READ AFTER
# WRITE FLAG FOR
# ERROR TESTING
*****
#SWPO ----->I
*****
# GETPATTERN ----->#GETPAT(83)*
*****
#NEXT ----->I
*****
# INITTRACKS ----->#INITTR(84)*
*****
I
# GETUNIT ----->#GETUNI(86)*
*****
I
# GETTRACK ----->#GETTR(85) *
*****
I
# WRITE ----->#WRITE(25) *
*****
I
# READCHK ----->#RDCHK(87) *
*****
I
# GETUNIT ----->#GETUNI(86)*
*****
I

```

```
*****  
**                               ** *****  
**      WRITE                    **-----> **WRITE(25) **  
**                               ** *****  
*****  
I  
*****  
**                               ** *****  
**      READCHK                   **-----> **RDCHK(87) **  
**                               ** *****  
*****  
I  
-----  
/ HAS WRITE/READ \ NO  
/  CHK BEEN DONE ON  \-----> **NEXT **  
/ ALL TRKS & UNITS? \  
-----  
I YES  
*****  
**                               ** *****  
**      STOP                      **-----> **STOP(88) **  
**                               ** *****  
*****  
I  
*****  
**SWPO **  
*****
```



```

*****
#WRITE *
*****
I
*****
**          **          *****
**  INITSECTOR  **----->#INITSE(55)*
**          **          *****
*****
#XWRITE ----->I
*****
**          **          *****
**  GETSECTOR  **----->#GETSEC(56)*
**          **          *****
*****
#ONEWRT ----->I
*****
#          SET RETRY
#          COUNTERS TO 10.
#          *****
#FILLBU ----->I
*****
**          **          *****
**  ADJSUM     **----->#ADJSUM(57)*
**          **          *****
*****
I
*****
#PUT A RETURN ON STACK* *****
#FOR WHEN BUFR IS FULL*----->#FILLDO(26)*
# & NO ERRORS EXIST * *****
*****
I
*****
#PUT A RETURN ON STACK* *****
# FOR WHEN ERRORS *----->#FILLER(28)*
#OCCUR ON FILLING BUF * *****
*****
I
*****
# INITIALIZE BYTE *
# COUNTER & SET *
# 'PSM' TO LEVEL 4 *
*****
I
*****
# ENABLE 'GO' 'FILL *
# BUFFER' FUNCTION & *
# 'INTERRUPT ENABLE' *
*****
I

```

```

----->I
*****
**          **----->#TRCK(58) #
**      TRCK          **
**          **
*****
I
*****
#  TRANSFER A  #
#  DATA BYTE  #
#              #
*****
I
*****
#  INCREMENT  #
#  BYTE COUNTER  #
#              #
*****

```

NOTE: WE LEAVE THIS LOOP  
WHEN THE RX11 BUFFER  
HAS BEEN FILLED WITH  
128(10) BYTES OF DATA

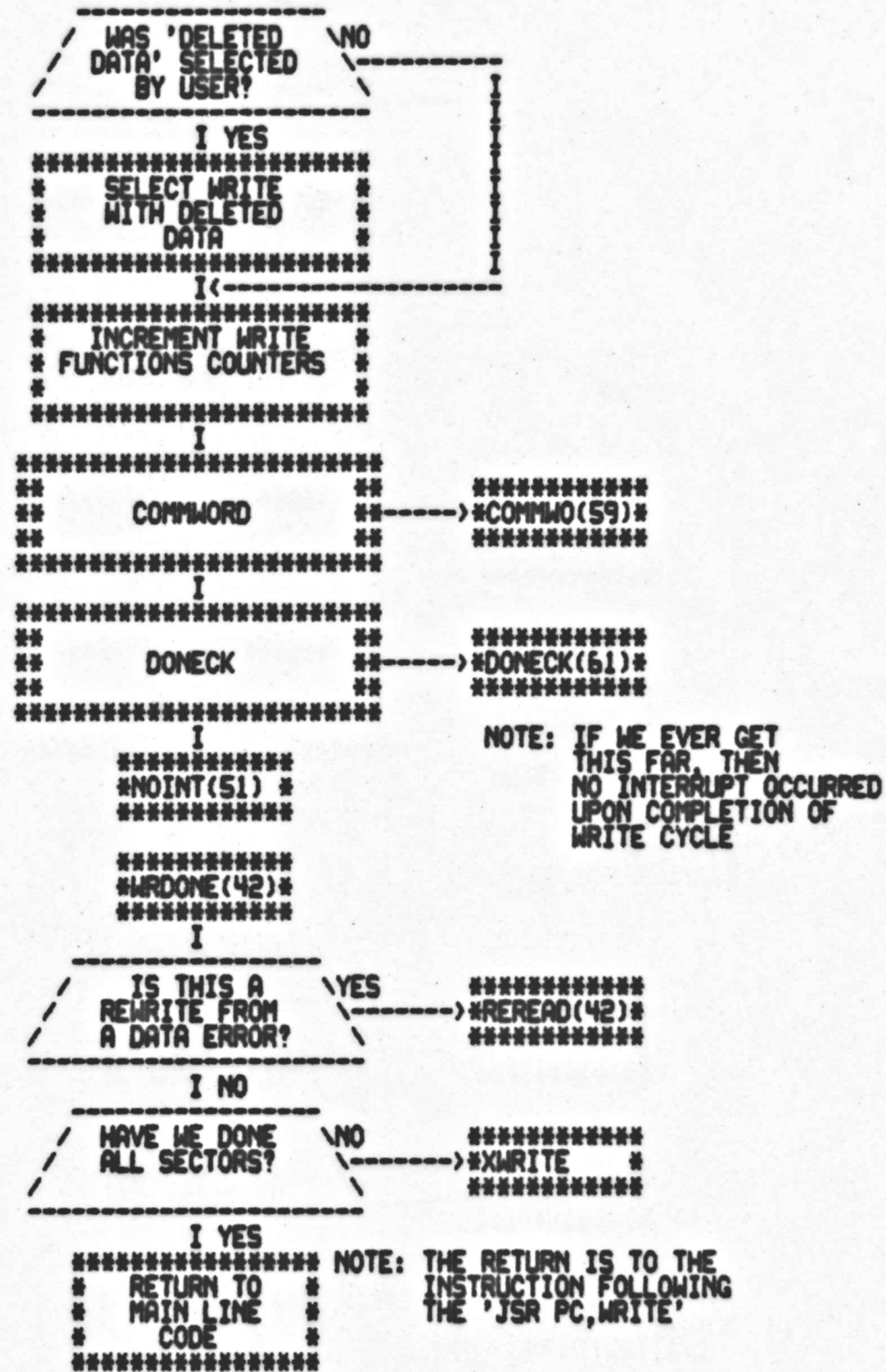
NOTE: WE COME HERE IF  
BUFFER HAS BEEN  
FILLED WITH NO ERRORS

```

*****
#FILLDO #
*****
I
*****
#  SET PARITY  #
#  RETRY COUNTER  #
#  TO 10.      #
#              #
*****
#REWRI #----->I
*****
#PUT A RETURN ON STACK#
#  FOR WRITING DONE  #----->#WRDONE(27)#
#  WITH NO ERRORS    #
#              #
*****
I
*****
#PUT A RETURN ON STACK#
#  FOR WHEN ERRORS  #----->#WRTER(39) #
#  OCCUR DURING WRITING #
#              #
*****
I
*****
#SELECT 'WRITE' #
#  FUNCTION 'GO' #
#  & 'IE' BITS  #
#              #
*****
I

```

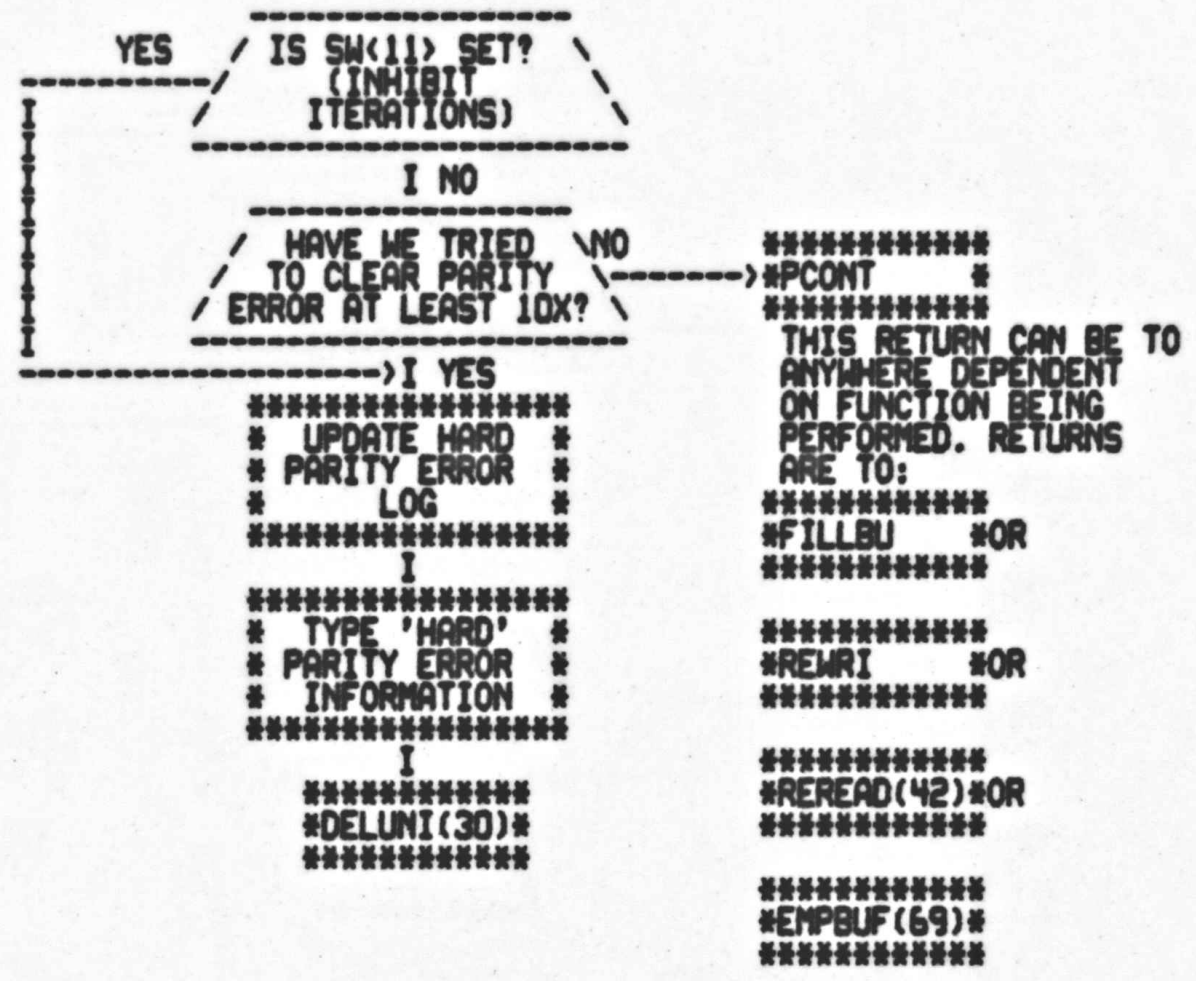


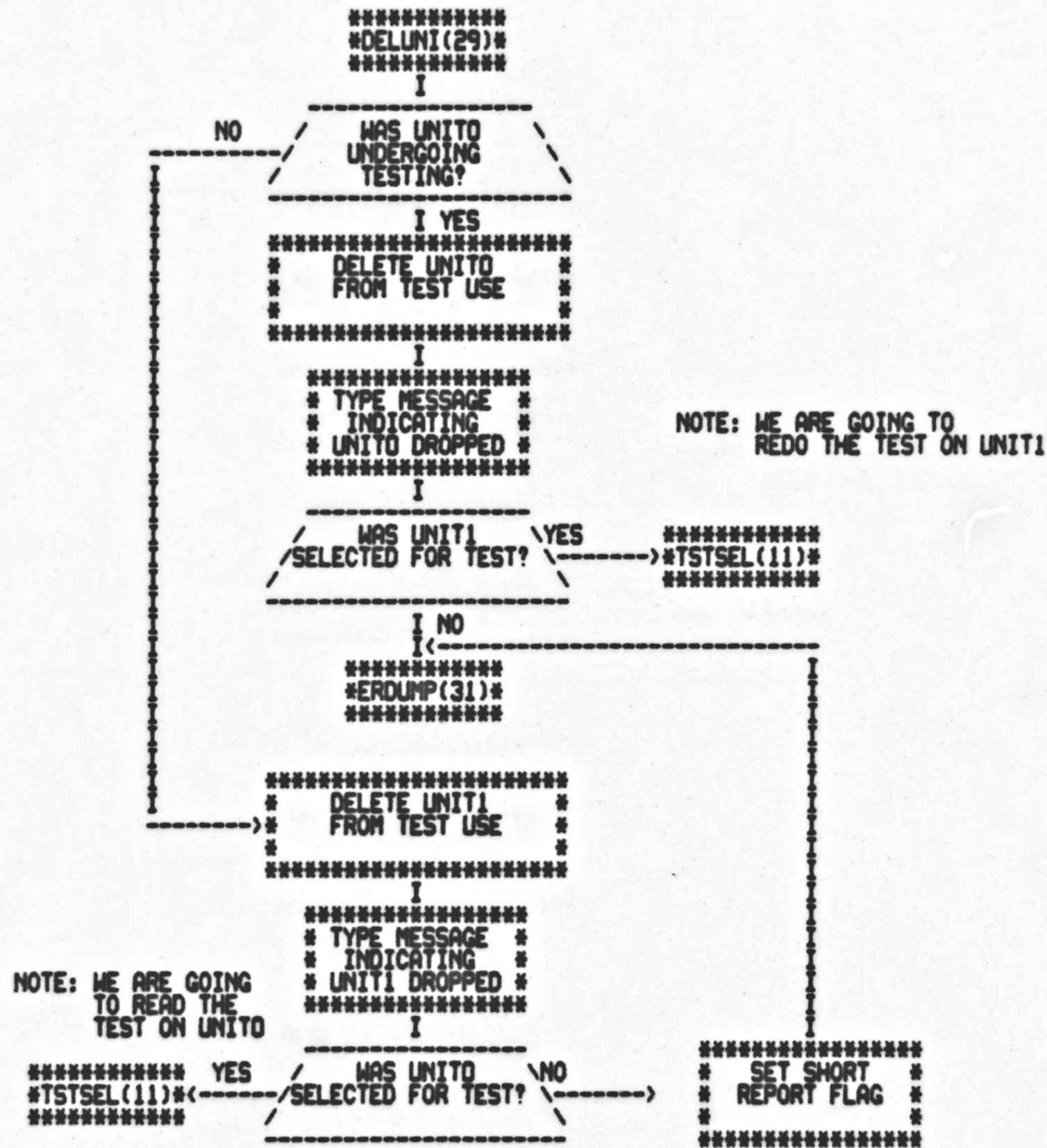




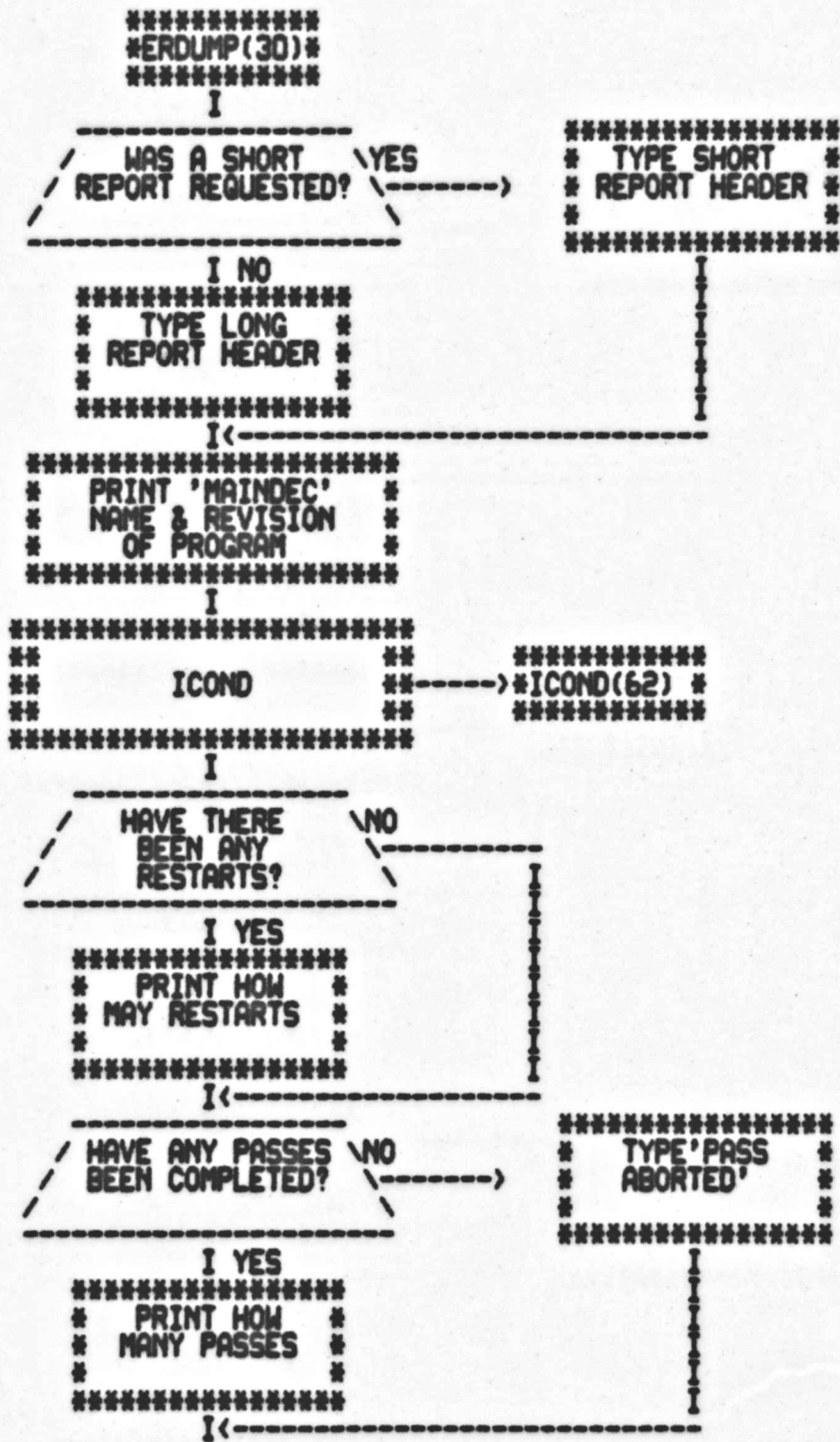


RX11 RELIABILITY TEST  
FLOPPY WRITE SEQUENCE









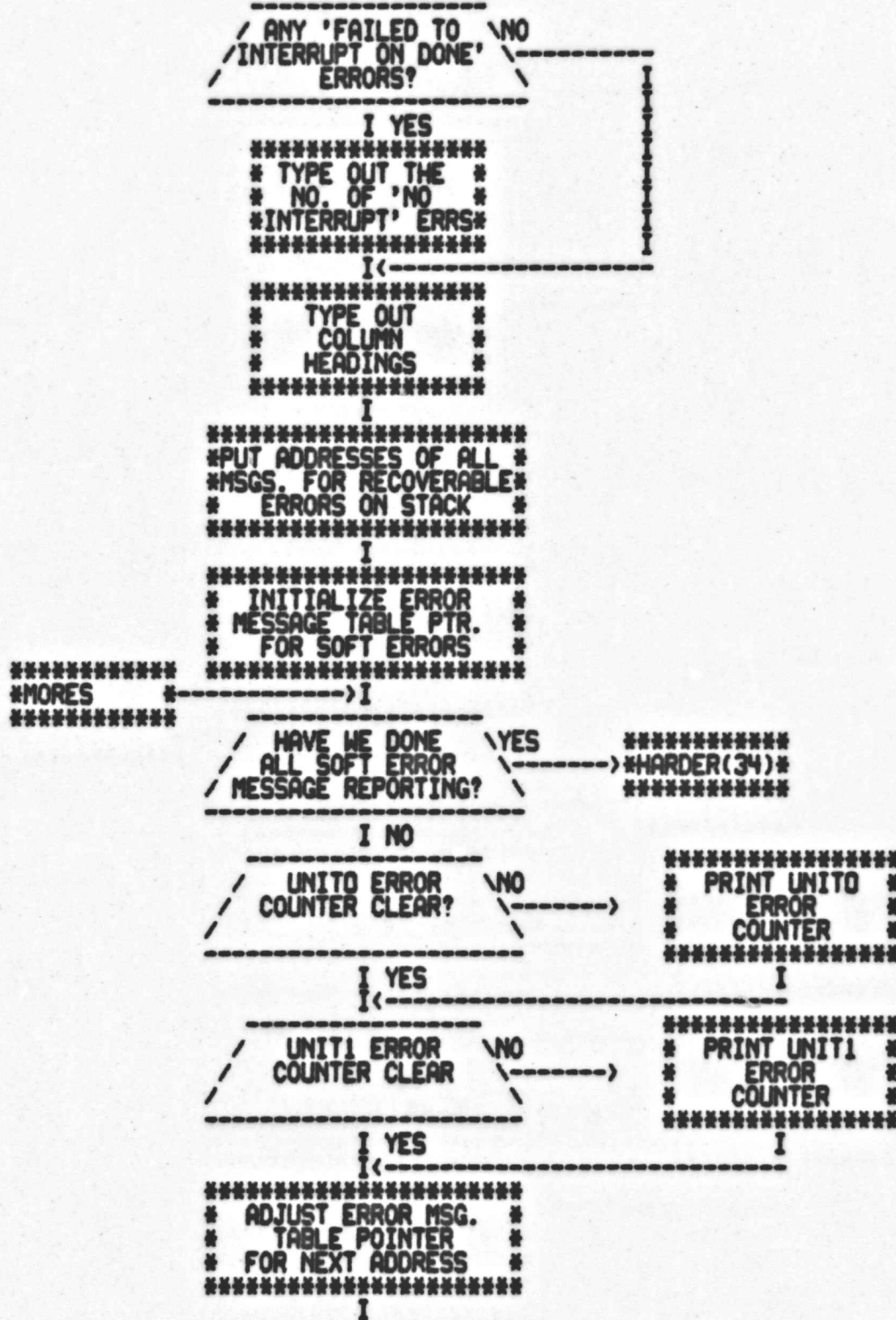
-----  
/ IS THIS TO BE \ YES  
A SHORT REPORT? \-----  
-----  
I NO  
\*\*\*\*\*  
\* PRINT TOTAL \*  
\* NO. OF SECTORS \*  
\* WRITTEN \*  
\*\*\*\*\*  
I  
\*\*\*\*\*  
\* PRINT TOTAL NO. \*  
\* OF SECTORS READ \*  
\* \*  
\*\*\*\*\*  
I<-----

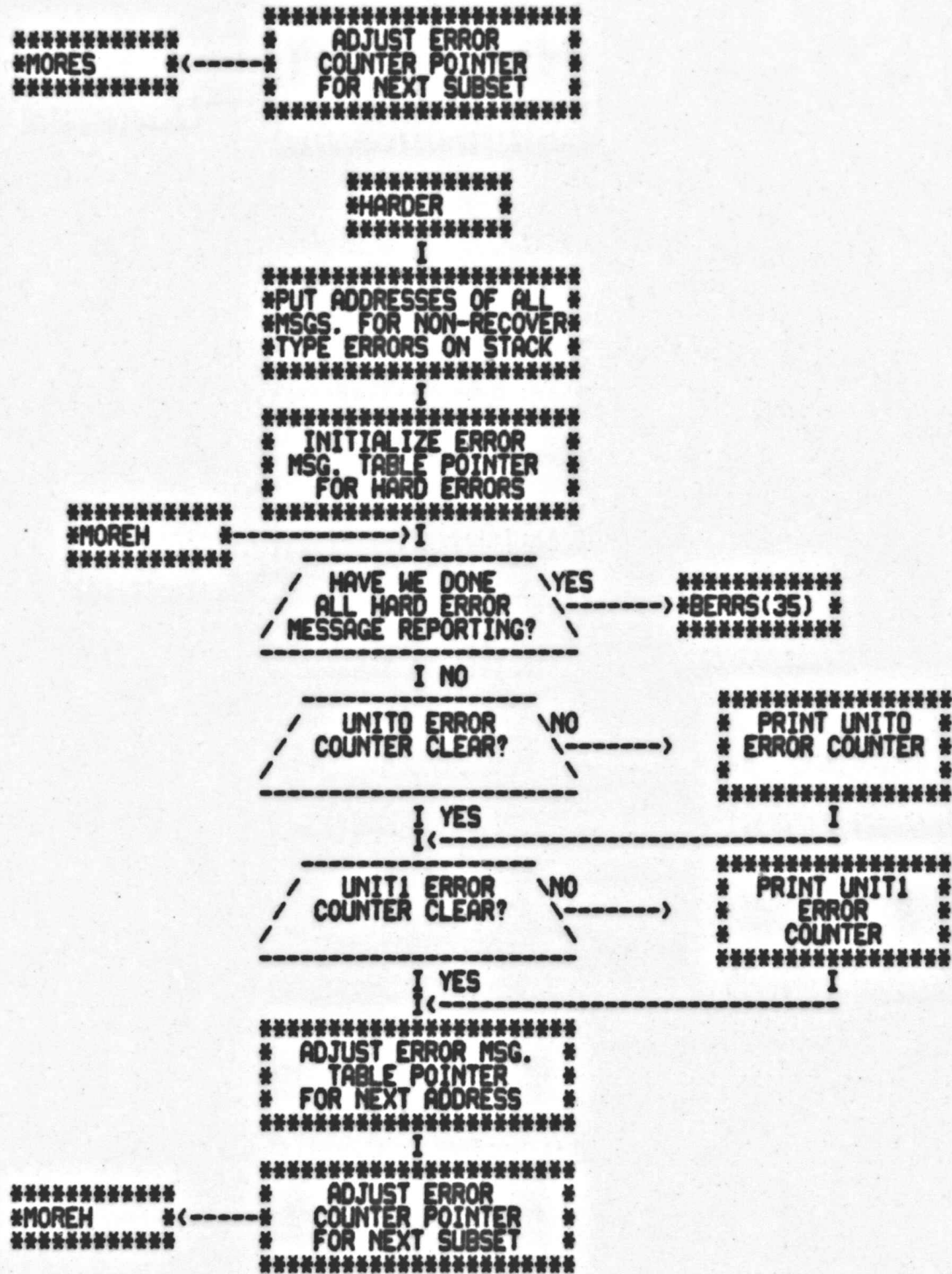
-----  
/ WERE THERE \ NO  
ANY PARITY ERRORS? \-----  
-----  
I YES  
\*\*\*\*\*  
\* TYPE OUT THE \*  
\* NO. OF PARITY \*  
\* ERRORS \*  
\*\*\*\*\*  
I

NO / WERE THERE \ /  
ANY HARD PARITY \ /  
ERRORS? \ /-----  
-----  
I YES  
\*\*\*\*\*  
\* TYPE OUT THE \*  
\* NO. OF HARD \*  
\* PARITY ERRORS \*  
\*\*\*\*\*  
I<-----

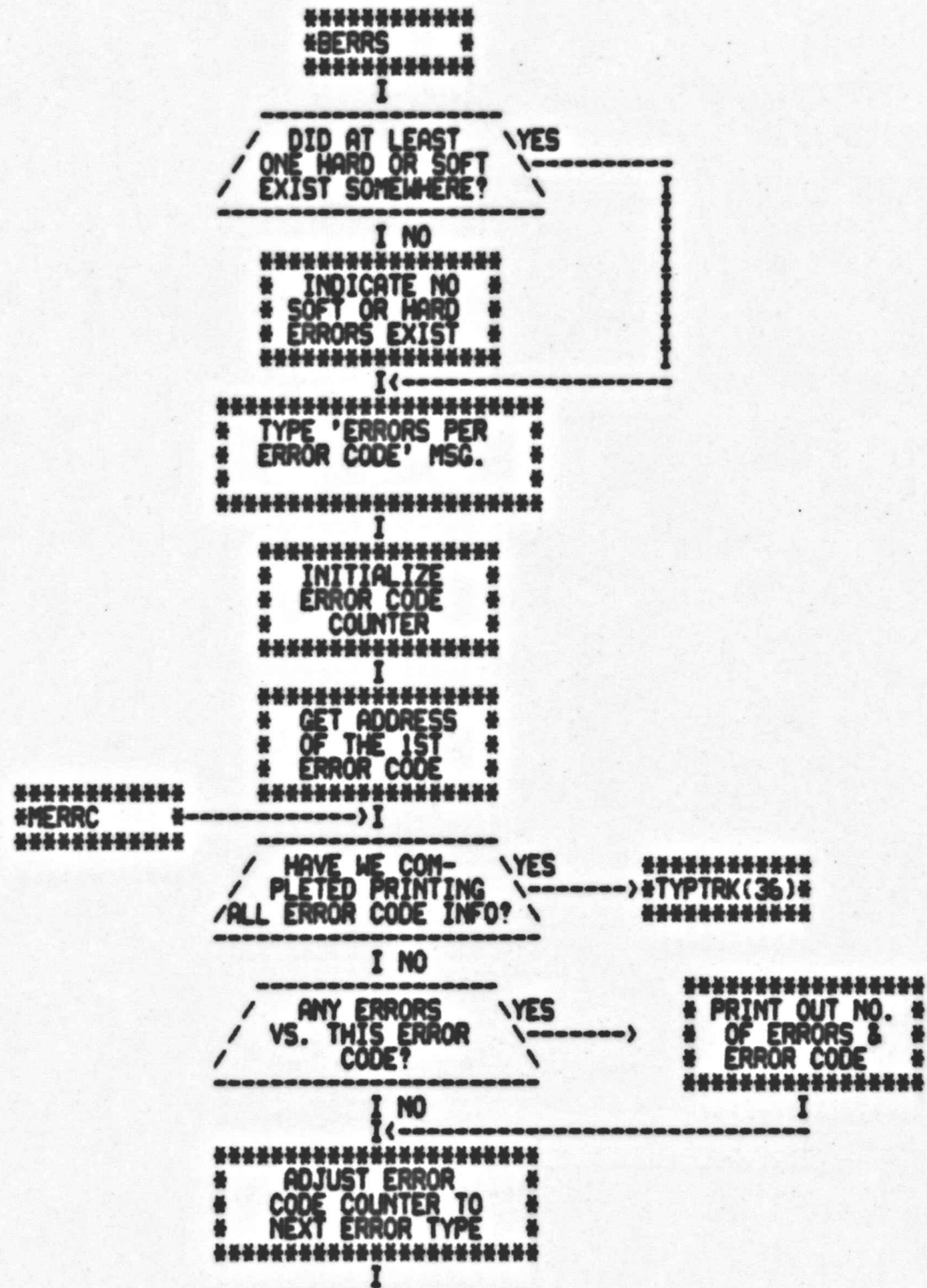
-----  
/ ANY STATUS \ NO  
ERROR FLAG \-----  
ERRORS? \-----  
-----  
I YES  
\*\*\*\*\*  
\* TYPE OUT THE \*  
\* NO. OF STATUS \*  
\* ERRORS \*  
\*\*\*\*\*  
I<-----











```
*****  
*****  
#MERRC * <----- *  
*****  
*****
```

```
*****  
# ADJUST ERROR *  
# CODE TABLE TO *  
# NEXT ADDRESS *  
*****
```

```
*****  
#TYPTRK *  
*****  
I
```

```
-----  
/ DID AT LEAST / YES  
| I TYPE OF 'ERROR |  
| CODE' ERROR EXIST? |  
-----
```

```
I NO  
*****  
# INDICATE 'NO *  
# ERROR CODE' *  
# ERRORS EXIST *  
*****  
I <-----
```

```
-----  
/ ARE WE TYPING / YES  
| A SHORT REPORT? |  
-----
```

```
I NO  
*****  
# TYPE OUT ERROR *  
# COLUMN HEADINGS *  
# *  
*****
```

```
*****  
# CLEAR 'SHORT *  
# REPORT' FLAG *  
# INDICATOR *  
*****  
I  
*****  
# HALT *  
# *  
*****
```

```
I  
*****  
#SET POINTERS TO TRACK*  
# ACCESS TABLE HEAD *  
#MOVEMENT, &08! ERRORS*  
*****  
I <-----
```

NOTE: PROGRAM MUST BE  
RESTARTED AT THIS POINT

```
*****  
#TRKACC *  
*****
```

\*\*\*\*\*NOTE: TRACK IS INITIALLY  
0 & INCREMENTED  
BY 1 THEREAFTER  
TO 114(8)

```
-----  
/ HAS THIS / NO  
| TRACK BEEN |  
| ACCESSED? |  
-----
```

```
I YES  
*****  
# TYPE OUT *  
# THE TRACK *  
# *  
*****  
I
```

```
*****  
# STEP UP ALL *  
# POINTERS TO *  
# NEXT TRACK *  
*****  
I
```

```
-----  
/ HAVE ALL / NO  
| TRACKS BEEN |  
| CHECKED? |  
-----
```

I YES



\*\*\*\*\*  
# TYPE OUT HOW  
# MANY TIMES THIS  
# TRACK WAS ACCESSED  
\*\*\*\*\*

\*\*\*\*\*  
# CLEAR 'SHORT  
# REPORT' FLAG  
# INDICATOR  
\*\*\*\*\*

I  
\*\*\*\*\*  
# TYPE OUT THE  
# HEAD MOVEMENT  
# COUNTER  
\*\*\*\*\*

I  
\*\*\*\*\*  
# HALT  
\*\*\*\*\*

NOTE: PROGRAM MUST BE  
RESTARTED AT THIS POINT

-----  
/ ANY UNIT0 \ NO  
| ERRORS ON THIS |-----> #UITERR #  
TRACK?

\*\*\*\*\*  
#UITERR #  
\*\*\*\*\*

I YES

\*\*\*\*\*  
# TYPE UNIT0  
# PRESENT TRACK  
# ERROR COUNT  
\*\*\*\*\*

\*\*\*\*\*  
#UITERR #  
\*\*\*\*\*

-----  
/ ANY UNIT1 \ NO  
| ERRORS ON THIS |-----> |  
TRACK?

I YES

\*\*\*\*\*  
# TYPE UNIT1  
# PRESENT TRACK  
# ERROR COUNT  
\*\*\*\*\*

I<

\*\*\*\*\*  
# STEP UP ALL  
# POINTERS TO  
# NEXT TRACK  
\*\*\*\*\*

I

-----  
/ HAVE ALL \ NO  
| TRACKS BEEN |-----> #TRKACC #  
CHECKED?

\*\*\*\*\*  
#TRKACC #  
\*\*\*\*\*

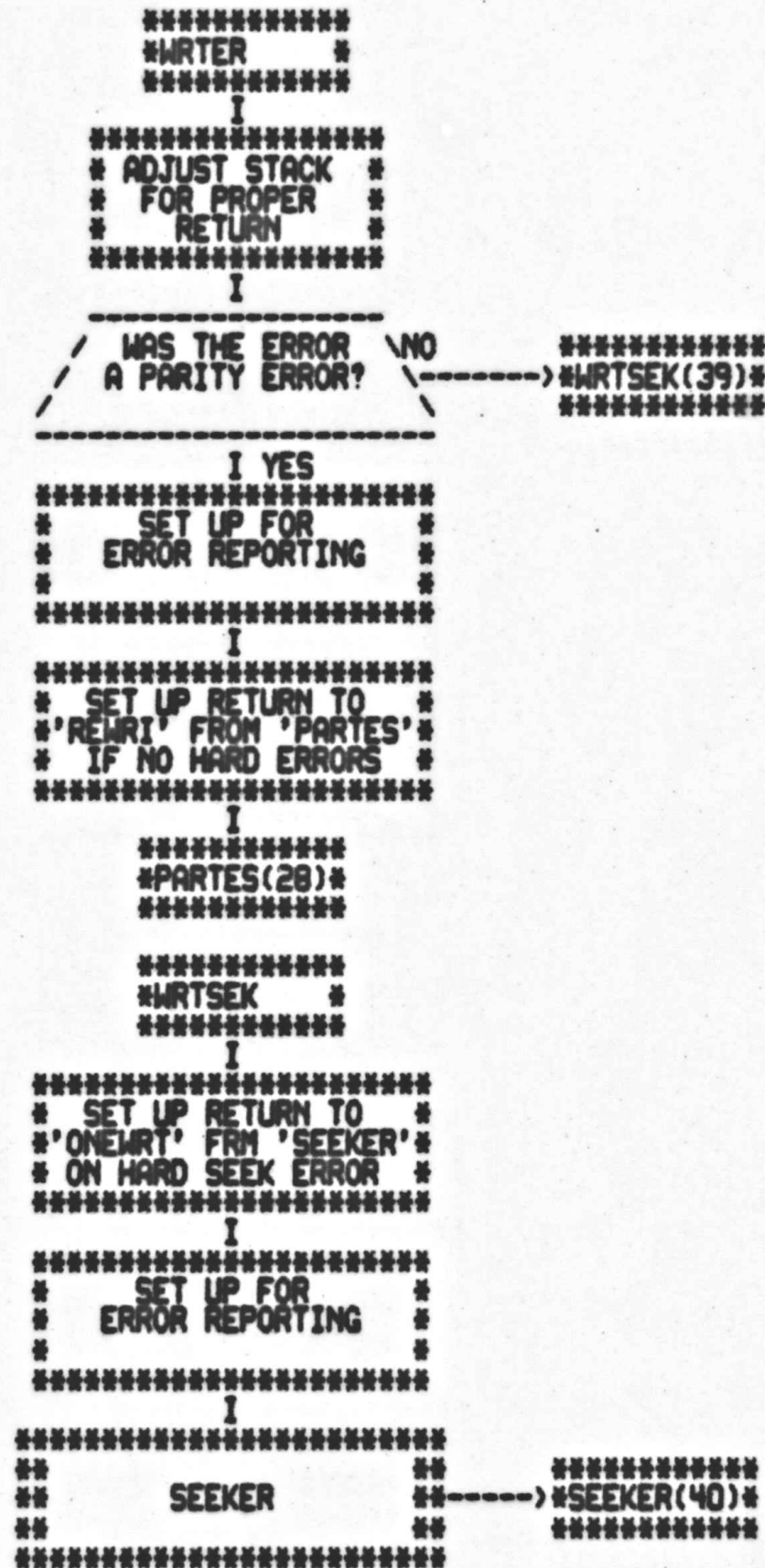
I YES

\*\*\*\*\*  
# CLEAR 'SHORT  
# REPORT' FLAG  
# INDICATOR  
\*\*\*\*\*

I

```
***** NOTE: PROGRAM MUST BE  
*          RESTARTED AT THIS POINT  
*          HALT  
*          *  
*****
```









RX11 RELIABILITY TEST  
FLOPPY WRITE SEQUENCE

\*\*\*\*\*  
# UPDATE SEEK #  
# ERROR RETRY #  
# COUNTER #  
\*\*\*\*\*

I

-----  
/ HAVE 10. SEEK \ YES  
/ ERRORS BEEN \  
/ LOGGED \  
-----

\*\*\*\*\*  
#HARDSK(41)#  
\*\*\*\*\*

I NO

\*\*\*\*\*  
## HOME ##  
## ##  
\*\*\*\*\*

\*\*\*\*\*  
#HOME(52) #  
\*\*\*\*\*

I

\*\*\*\*\*  
#SEKRTY #  
\*\*\*\*\*

THIS RETURN CAN BE TO  
ANYWHERE DEPENDENT  
ON FUNCTION BEING  
RETURNS ARE TO:

\*\*\*\*\*  
#ONEWRT #OR  
\*\*\*\*\*

\*\*\*\*\*  
#REREAD(42)#  
\*\*\*\*\*

\*\*\*\*\*  
#HARDSK #  
\*\*\*\*\*

I

\*\*\*\*\*  
## UILOG ##  
## ##  
\*\*\*\*\*

\*\*\*\*\*  
#UILOG(63) #  
\*\*\*\*\*

I

\*\*\*\*\*  
# UPDATE SEEK #  
# ERROR LOG #  
# #  
\*\*\*\*\*

I

\*\*\*\*\*  
# TYPE 'NON-  
# RECOVERABLE #  
# SEEK ERROR' #  
\*\*\*\*\*

I

```
*****  
**                               **  
**      SEKTYP                    **----->#SEKTYP(64)*  
**                               **  
**                               **  
*****
```

```
I  
*****  
* UPDATE STACK *  
* POINTER FOR *  
* PROPER RETURN *  
*****
```

NOTE: AT THIS POINT WE ARE  
GOING BACK TO DO THE  
NEXT SECTOR

```
I  
*****  
#WRDONE(27)*  
*****
```

```
*****  
#REREAD(29)*  
*****
```

```
I  
*****  
* CLEAR THE CYCLICAL *  
* REDUNDANCY CHECK *  
* FLAG (CRC) *  
*****
```

```
I  
*****  
*PUT A RETURN ON STACK*  
* FOR READING DONE *----->#RDDONE(43)*  
* WITH NO ERRORS *  
*****
```

```
I  
*****  
*PUT A RETURN ON STACK*  
* FOR WHEN ERRORS *----->#RDERR(45) *  
* OCCURRED DURING READ *  
*****
```

```
I  
*****  
* SELECT 'READ' *  
* FUNCTION, 'GO' & *  
* 'IE' BITS *  
*****
```

```
I  
*****  
* UPDATE THE NO. *  
* OF TOTAL READS *  
* COUNTER *  
*****
```

I



```
*****  
**          COMMWORD          **----->*****  
**          COMMWO(59)*****  
**          *****  
*****
```

```
          I  
*****  
**          DONECK          **----->*****  
**          DONECK(61)*****  
**          *****  
*****
```

```
          I  
*****  
**          NOINT(51)*****  
**          *****  
*****
```

NOTE: IF WE EVER GET THIS  
FAR, THEN NO INTERRUPT  
OCCURRED UPON COMPLETION  
OF READ CYCLE

```
*****  
**          RDDONE          *****  
**          *****  
*****
```

```
          I  
-----  
/  HAS READ RETRY  \ YES  *****  
|  BEEN EXECUTED  |----->|  CONT1  |*****  
|  10X?          |----->|  *****  
-----
```

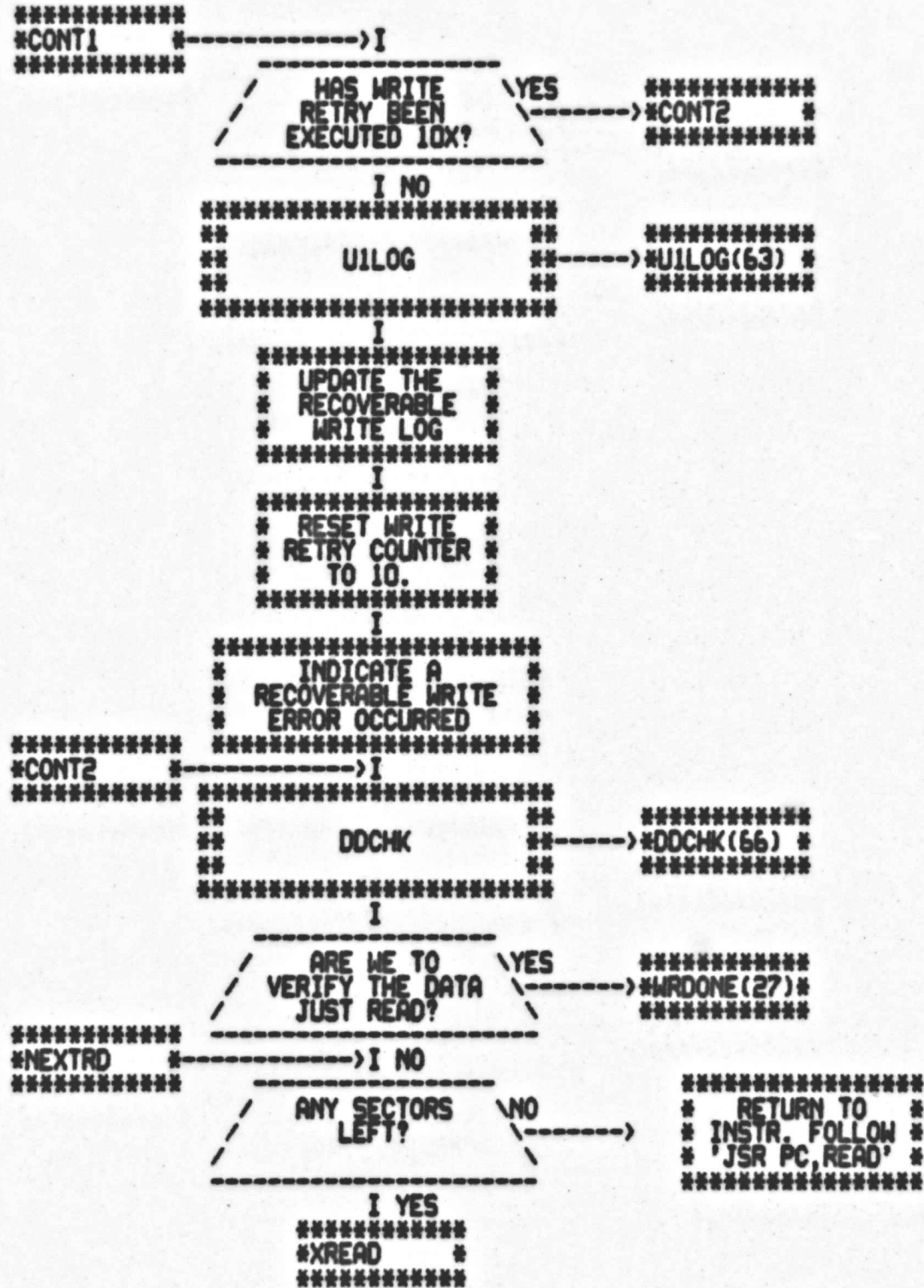
```
          I NO  
*****  
**          UILOG          **----->*****  
**          UILOG(63)*****  
**          *****  
*****
```

```
          I  
*****  
**          UPDATE          *****  
**          RECOVERABLE*****  
**          READ LOG*****  
*****
```

```
          I  
*****  
**          RESET READ*****  
**          RETRY COUNTER*****  
**          TO 10.*****  
*****
```

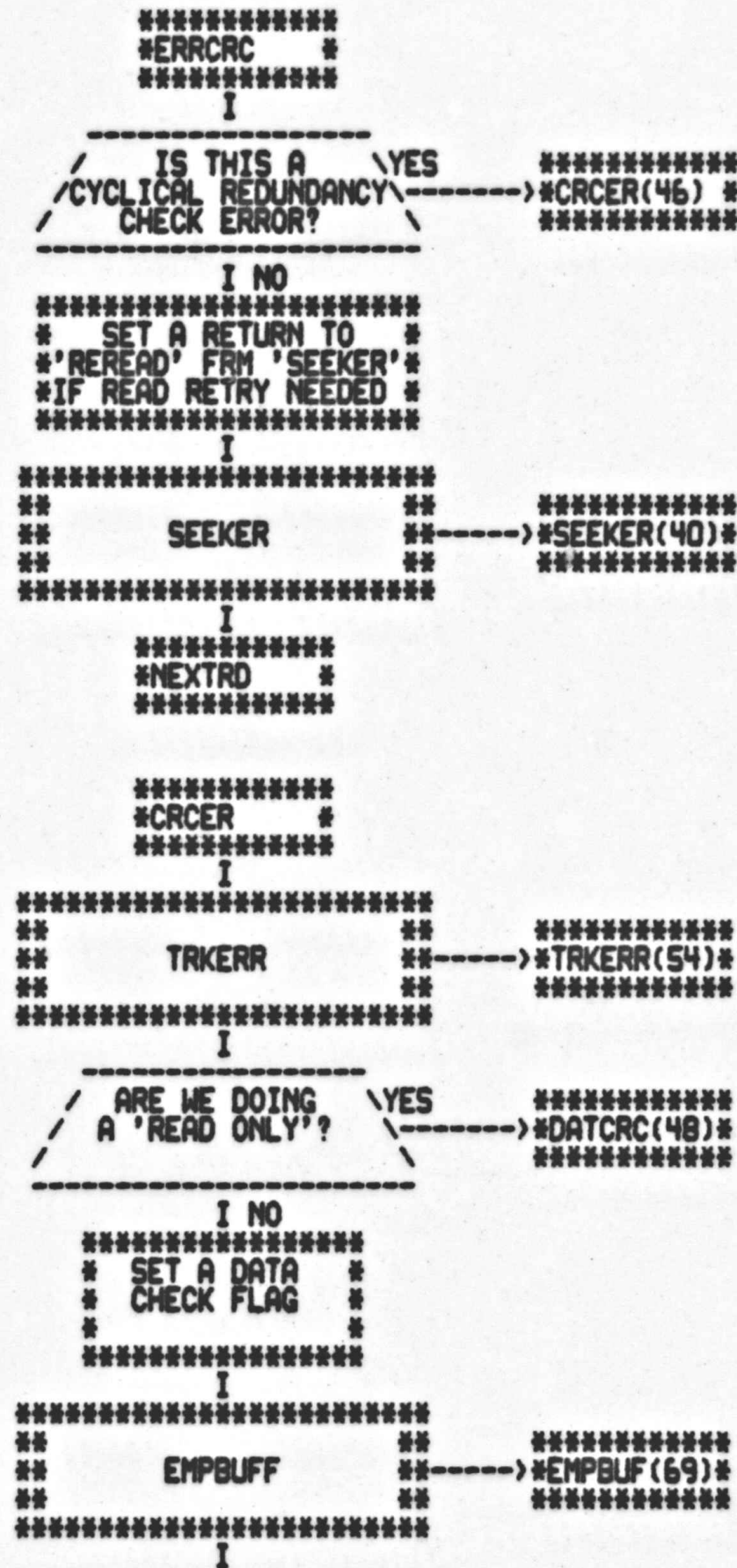
```
          I  
*****  
**          INDICATE A*****  
**          RECOVERABLE RD*****  
**          ERROR OCCURRED*****  
*****
```

I

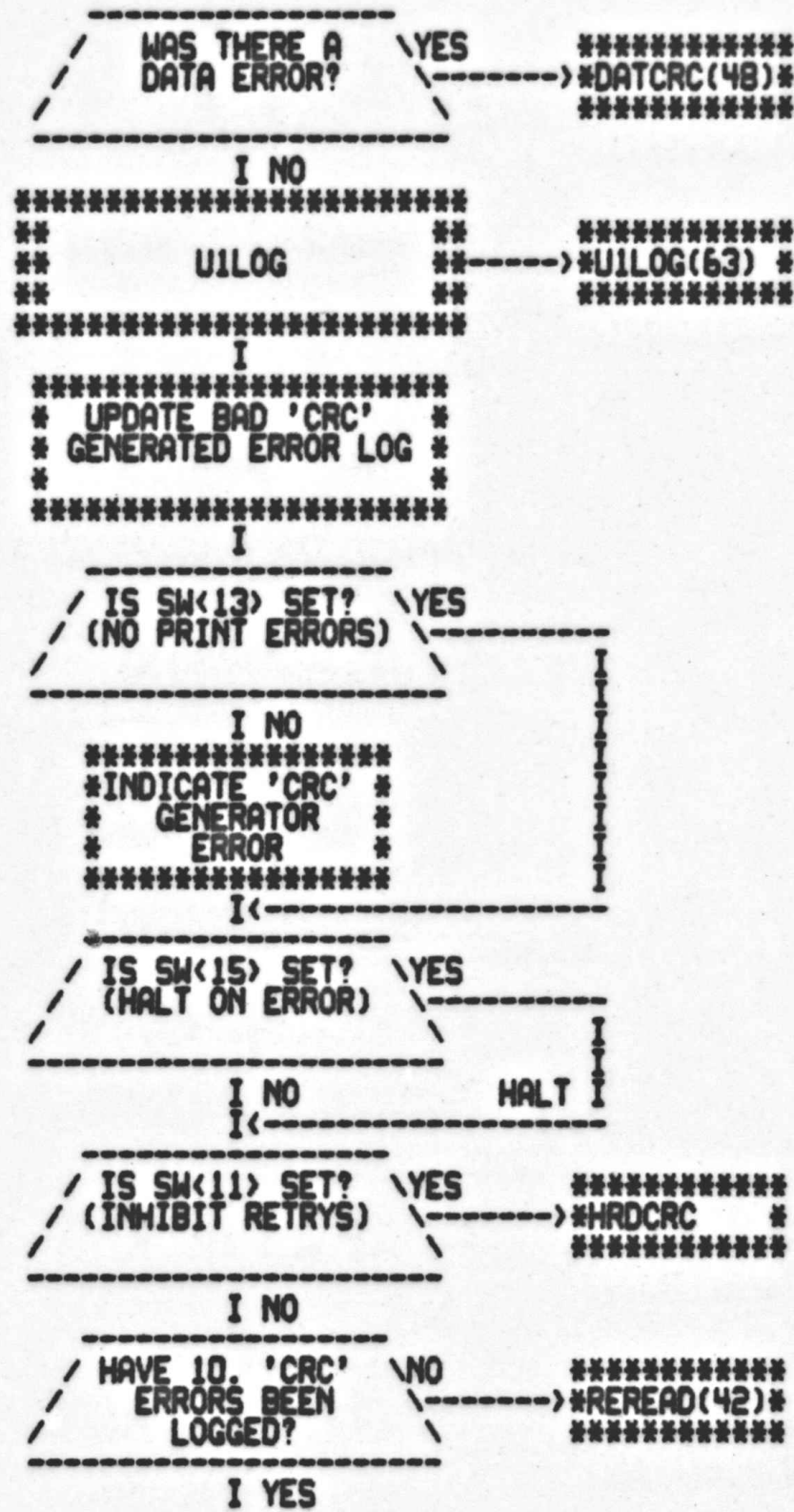




```
*****  
*RDERR *  
*****  
I  
*****  
* UPDATE STACK *  
* FOR PROPER *  
* RETURN *  
*****  
I  
-----  
/ IS THIS A \ NO  
 \ PARITY ERROR? / -----> *****  
----- *ERRCRC(46)*  
*****  
I YES  
*****  
* SET UP FOR *  
* ERROR *  
* REPORTING *  
*****  
I  
*****  
* SET UP RETURN TO *  
* 'REREAD' FRM 'PARTES' *  
* IF HARD ERROR EXISTS *  
*****  
I  
*****  
*PARTES(28)*  
*****
```

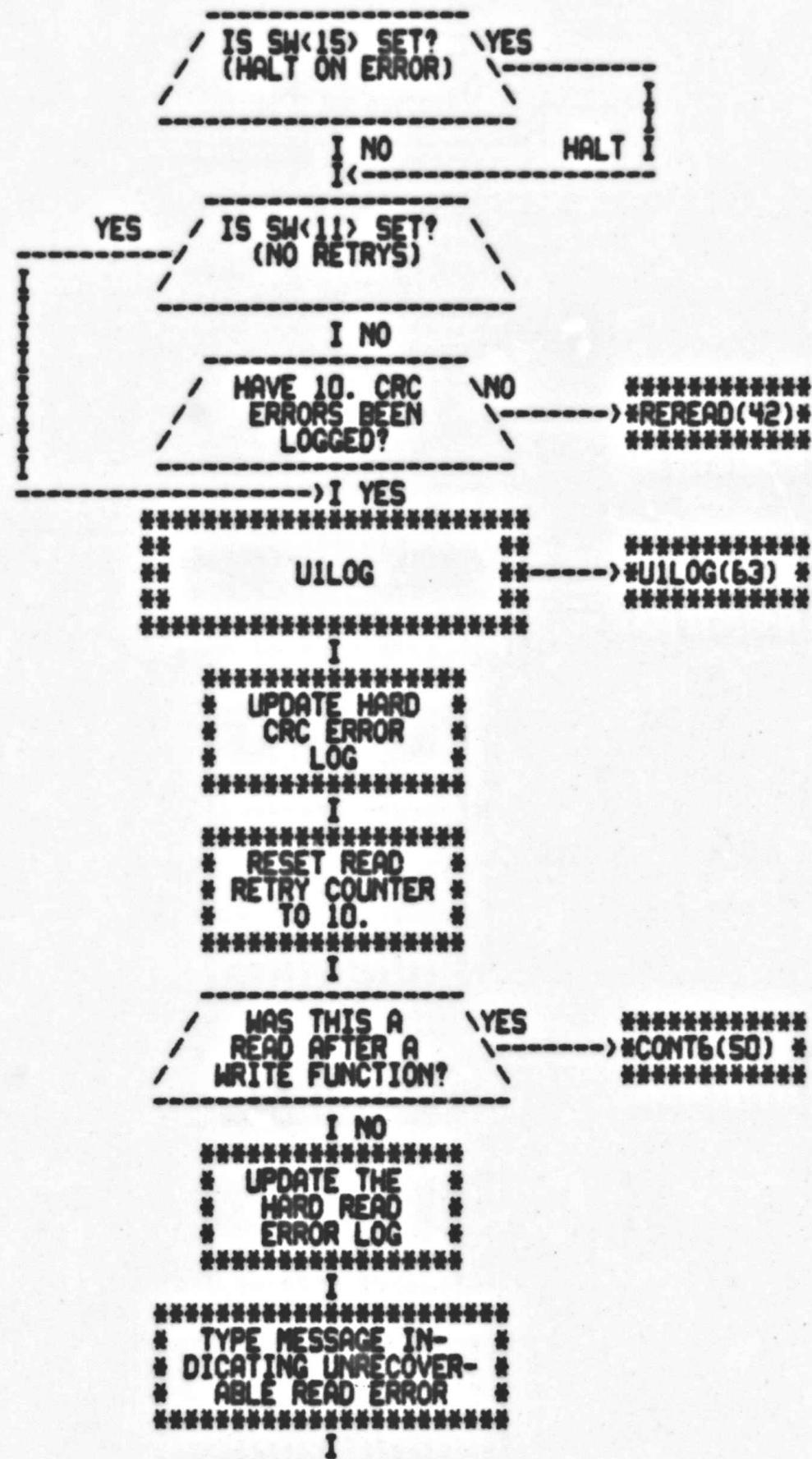


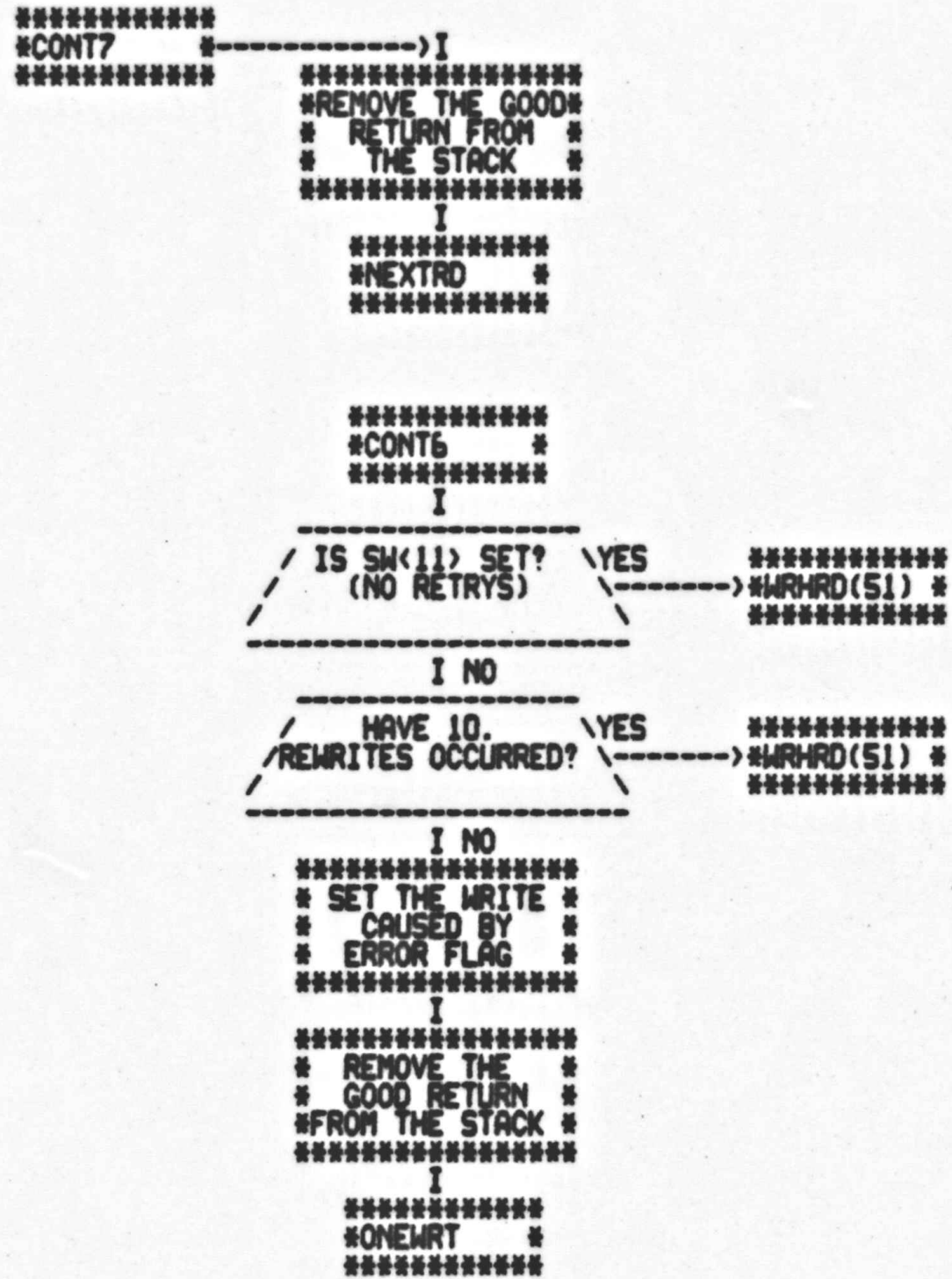




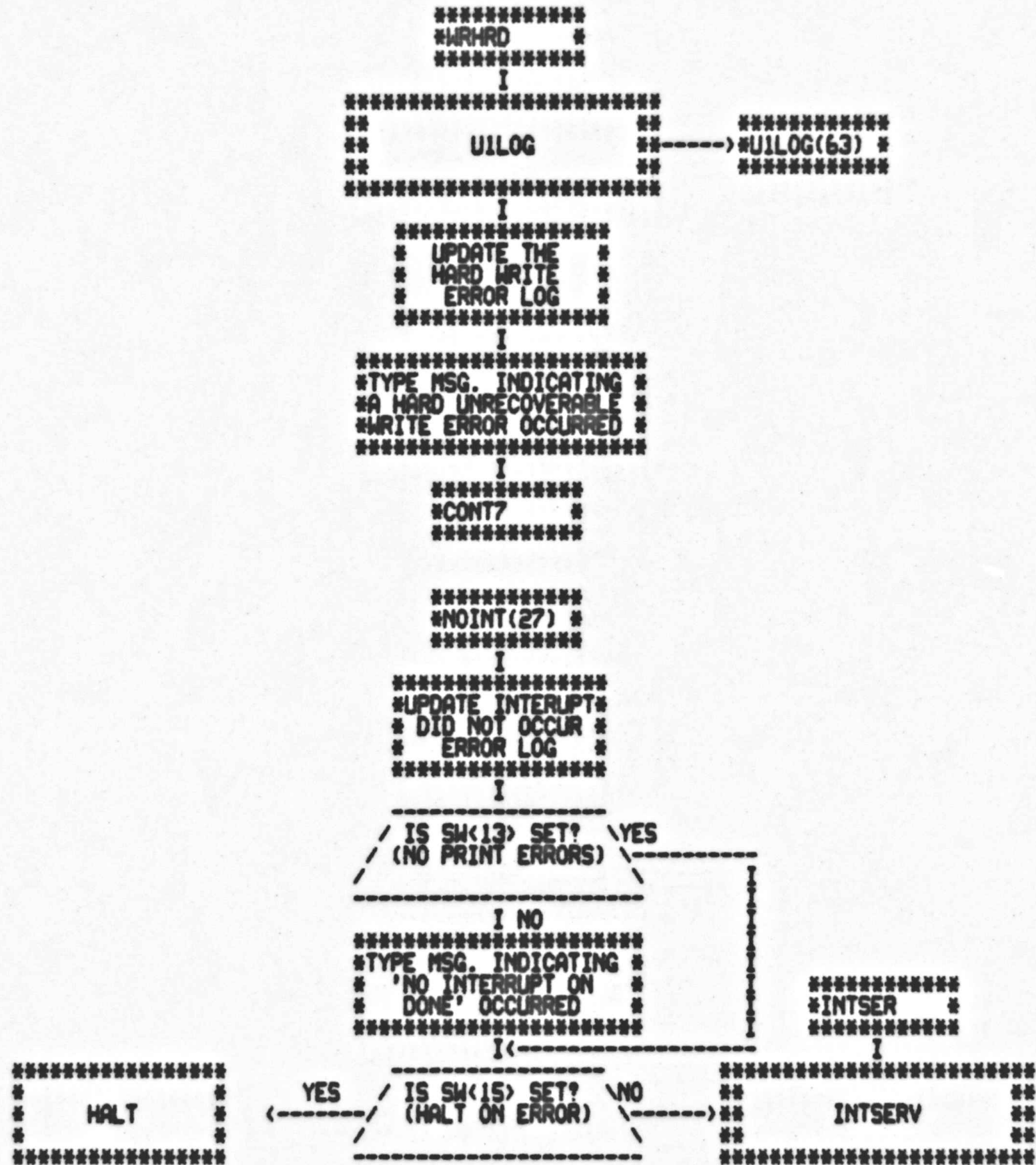


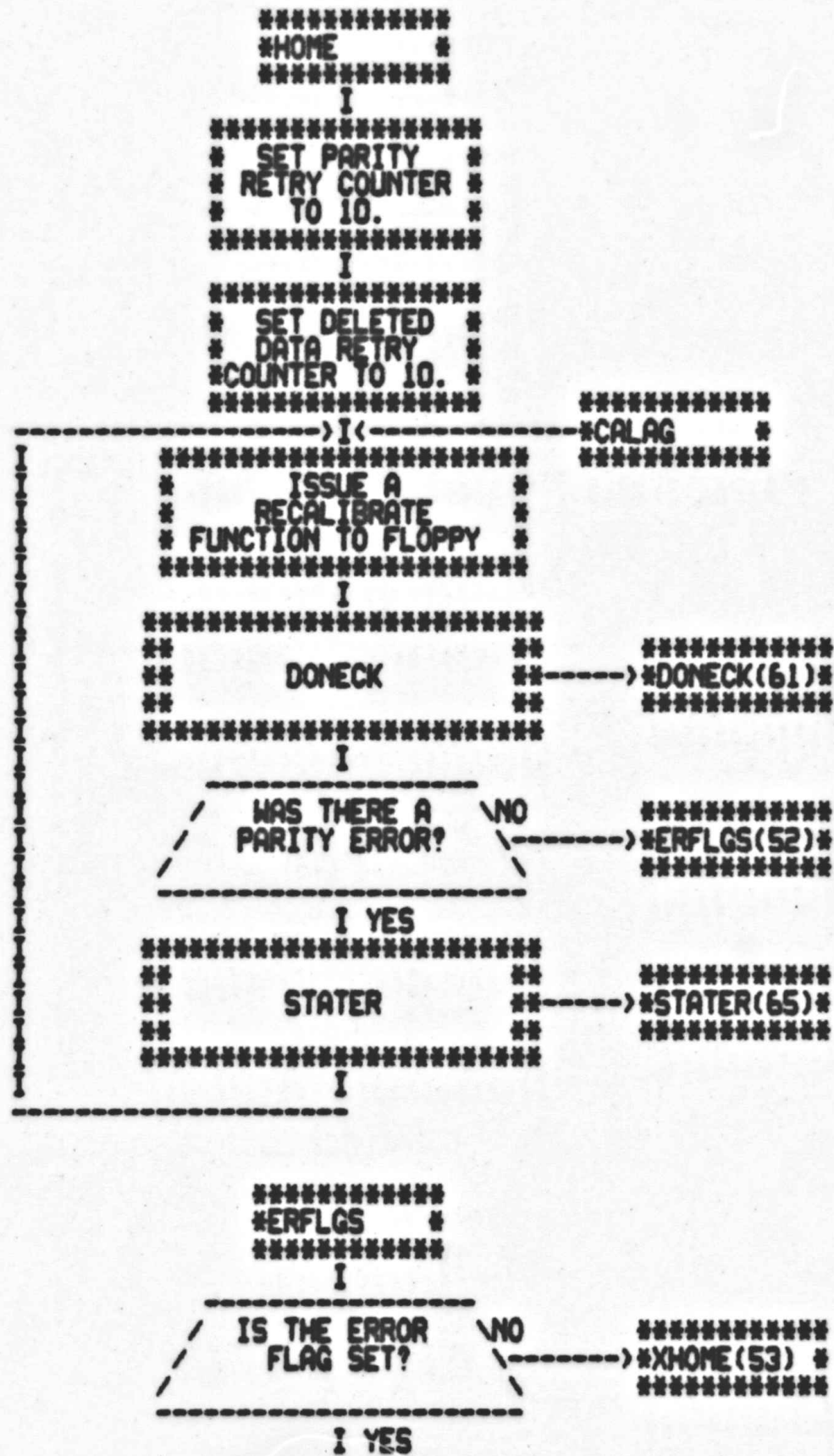








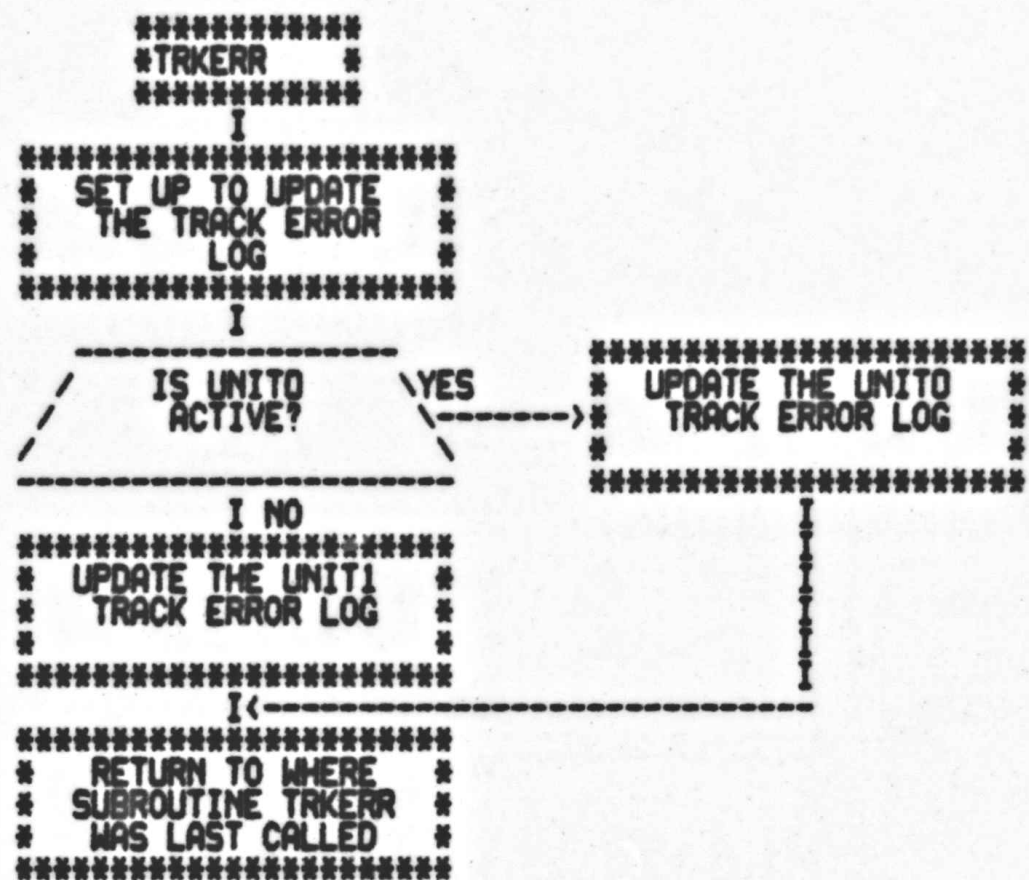






```
*****
**                                     **
**          RDCODE          **----->#RDCODE(81)#
**                                     **
*****
I
-----
/ WAS THE ERROR \ NO          *****
/ CODE 1 OF THE \----->#XHOME(53) #
/ RECALIBRATE   \
/ CODES?       /
-----
I YES
-----
/ HAVE 10. \ NO          *****
/ RECALIBRATE \----->#CALAG #
/ RETRIES OCCURRED? /
-----
I YES
*****
* TYPE MESSAGE INDI- *
* CATING FLOPPY WILL *
* NOT RECALIBRATE *
*****
I
*****
*          *----->#START(01) #
*      HALT          *
*          *
*****

*****
#XHOME #
*****
I
*****
* PRESET PRESENT *
* TRACK TO TRACK 1 *
*          *
*****
I
*****
* RETURN TO *
* WHERE SUBR. *
* HOME WAS CALLED *
*****
```





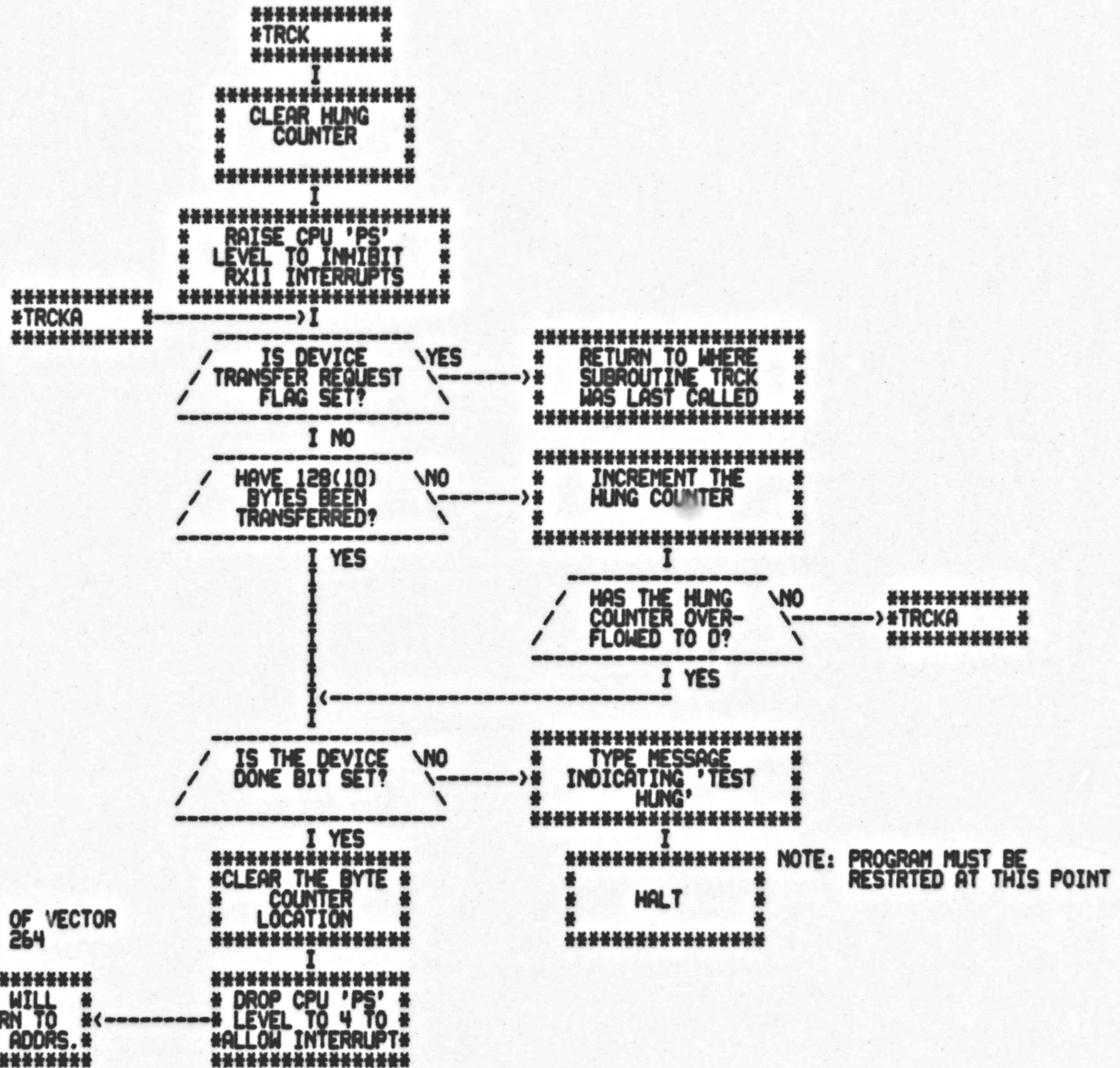
```
*****  
#INITSE #  
*****  
I  
-----  
/ DID USER MAKE \ NO  
ANY SECTOR SELECTIONS? \-----> *****  
----- # INIT. 1ST & LAST #  
# SECTORS TO SECTOR 1 #  
# & 32(8) RESPECTIVELY #  
----- #  
I YES I  
I<-----I  
*****  
# SET UP THE #  
# SECTOR #  
# COUNTER #  
*****  
I  
*****  
# RID HIGH #  
# BYTE OF SECTOR #  
# CTR. OF JUNK #  
*****  
I  
*****  
# INIT. TARGET #  
# SECTOR TO 1ST #  
# SECTOR SELECTED #  
*****  
I  
***** HEAD MOVEMENT  
# SUBTRACT 3 FROM # SETTLING FACTOR  
# THE TARGET SECTOR #  
# (INIT. SUBR. GETSEC) #  
*****  
I  
***** HEAD MOVEMENT  
# INIT. HEAD # SETTLING OFFSET  
# MOVEMENT SETTL #  
# ING OFFSET TO 1 #  
*****  
I  
*****  
# RETURN TO WHERE #  
# SUBR. INITSE(CTOR) #  
# WAS LAST CALLED #  
*****
```

```
*****  
*GETSEC*  
*****  
I  
*****  
* ADD HEAD MOVEMENT *  
* SETTLING FACTOR (3) *  
* TO GET NXT TARG. SECT *  
*****  
I  
-----  
/ IS THE TARGET \ YES  
/ SECTOR LESS THAN \  
/ 32(8) MAX. ALLOWED? \  
-----  
I NO  
*****  
* RESET TARGET *  
* SECTOR TO 1ST *  
* SELECTED SECTOR VALUE *  
*****  
I  
*****  
* ADD THE CURRENT HEAD *  
* MOVEMENT SETTLING *  
* VALUE TO TARGET SECT *  
*****  
I  
*****  
* UPDATE THE HEAD MOVE- *  
* MENT SETTLING OFFSET *  
* BY+1 FOR NXT GO-ROUND *  
*****  
I<-----  
*****  
* RETURN TO WHERE *  
* SUBR. GETSEC(TOR) *  
* WAS LAST CALLED *  
*****
```

NOTE: 1 TOTAL PASS  
THRU THIS ROUTINE  
WILL RUN SECTORS:  
1,4,7,10,13,16,23,26  
  
NEXT TOTAL PASS  
WILL RUN SECTORS:  
2,5,10,13,16,23,26  
  
ETC.  
  
UNTIL SECTOR 32(8)  
HAS BEEN REACHED  
THEREBY, ALL SECTORS  
HAVING BEEN DONE  
  
NOTE: THIS PROCEDURE IS  
USED TO ALLOW FOR  
HEAD MOVEMENT  
SETTLING, SHUTDOWN  
TIME, ETC.



```
*****  
*ADJSUM *  
*****  
      I   BUFADR  
*****  
* GET TARGET TRACK *  
* GET TARGET SECTOR *  
*(LOW & HI BYTE RESP.)*  
*****  
      I   CKSUM  
*****  
*   GET THE   *  
*   PATTERN   *  
* CHECKSUM   *  
*****  
      I  
*****  
* ADD TRACK + SECTOR *  
* ADDRESSES TO THE *  
* PATTERN CHECKSUM *  
*****  
      I   LOBYTE  
*****  
*PLACE THE TOTAL POS. *  
*CHECKSUM * INTO LAST *  
*LOCATION OF DATA BUFF*  
*****  
      I  
*****  
* FORM THE NEG- *  
* ATIVE CKSUM *  
*   VALUE   *  
*****  
      I   HIBYTE  
*****  
*PLACE THE TOTAL NEG. *  
*CHECKSUM * INTO LAST *  
*LOCATION OF DATA BUFF*  
*****  
      I   BUFADR  
*****  
* INIT. DATA BUFFER *  
* STARTING ADDRESS *  
*   INTO RD   *  
*****  
      I  
*****  
* RETURN TO WHERE *  
* SUBROUTINE ADJSUM *  
* WAS LAST CALLED *  
*****
```





```
*****  
#COMMWO #  
*****  
I TARGET  
*****  
# GET TARGET #  
# TRACK VALUE #  
#  
*****  
I  
*****  
#MULTIPLY TARGET#  
# TRACK VALUE #  
# BY 4 #  
*****  
I  
*****  
# ADD IN ADDRESS #  
# OF TRACK ACCESS #  
# COUNTER #  
*****  
I  
*****  
# UPDATE THE TRACK #  
# ACCESS COUNTER & #  
#PICK UP UNIT SELECTED#  
*****  
I  
*****  
# SEND UNIT #  
#SELECTION VALUE#  
# TO DRIVE #  
*****  
I  
*****  
** TRCK **----->#TRCK(58) #  
** **----->#TRCK(58) #  
*****  
I TSECTR  
*****  
# SEND TARGET #  
# SECTOR TO #  
# DRIVE #  
*****  
I  
*****  
** TRCK **----->#TRCK(58) #  
** **----->#TRCK(58) #  
*****  
I
```

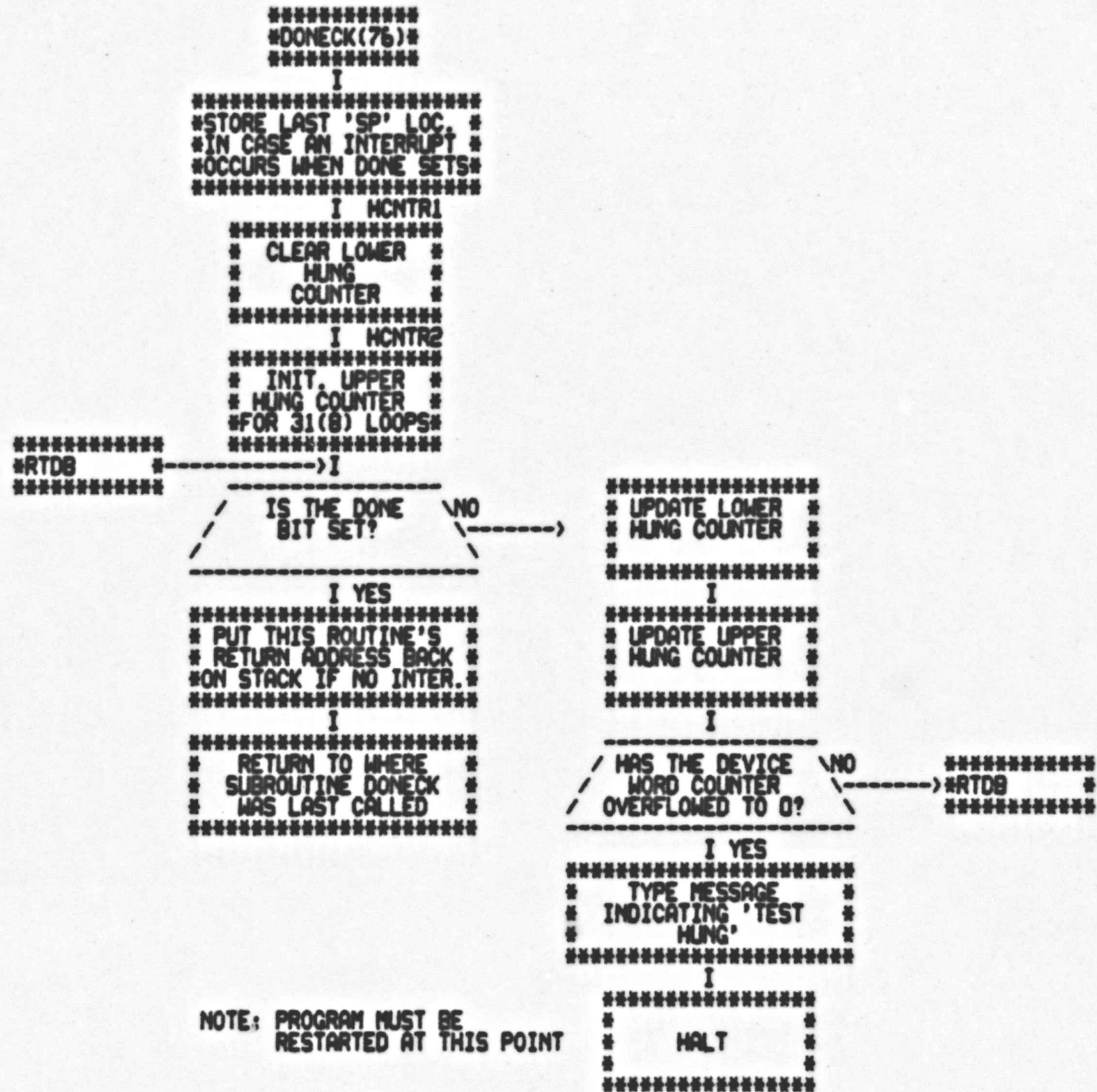
RX11 RELIABILITY TEST  
SUBROUTINE COMMWO(RD)

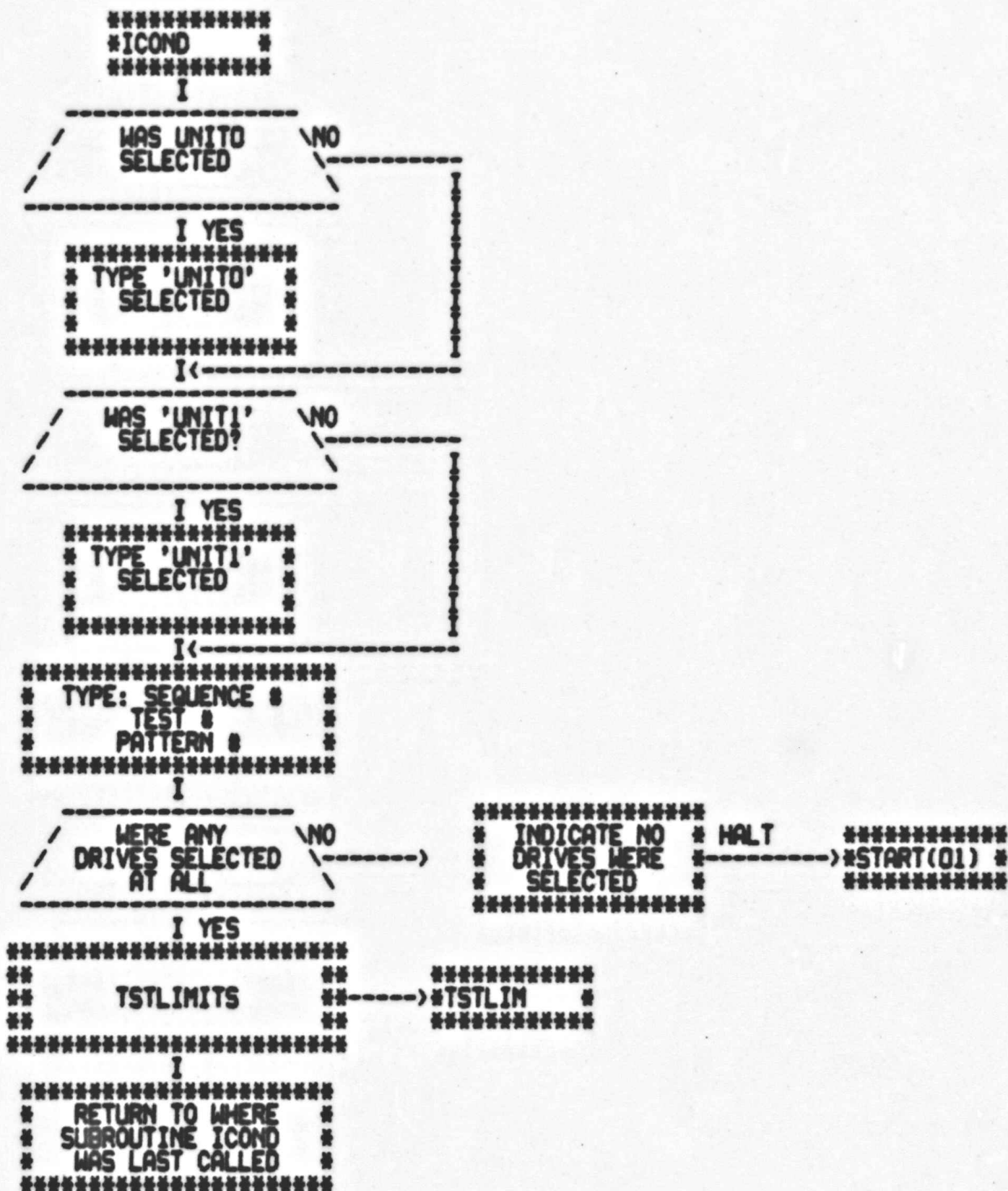
```

                                TARGET
*****
* SEND TARGET *
* TRACK TO *
* DRIVE *
*****
I
*****
* LOWER CPU 'PS' *
* LEVEL TO 4 TO *
* ALLOW AN INTERRUPT *
*****
I
*****
* RETURN TO WHERE *
* SUBR. COMMWO(RD) *
* WAS LAST CALLED *
*****
```



RX11 RELIABILITY TEST  
SUBROUTINE DONECK







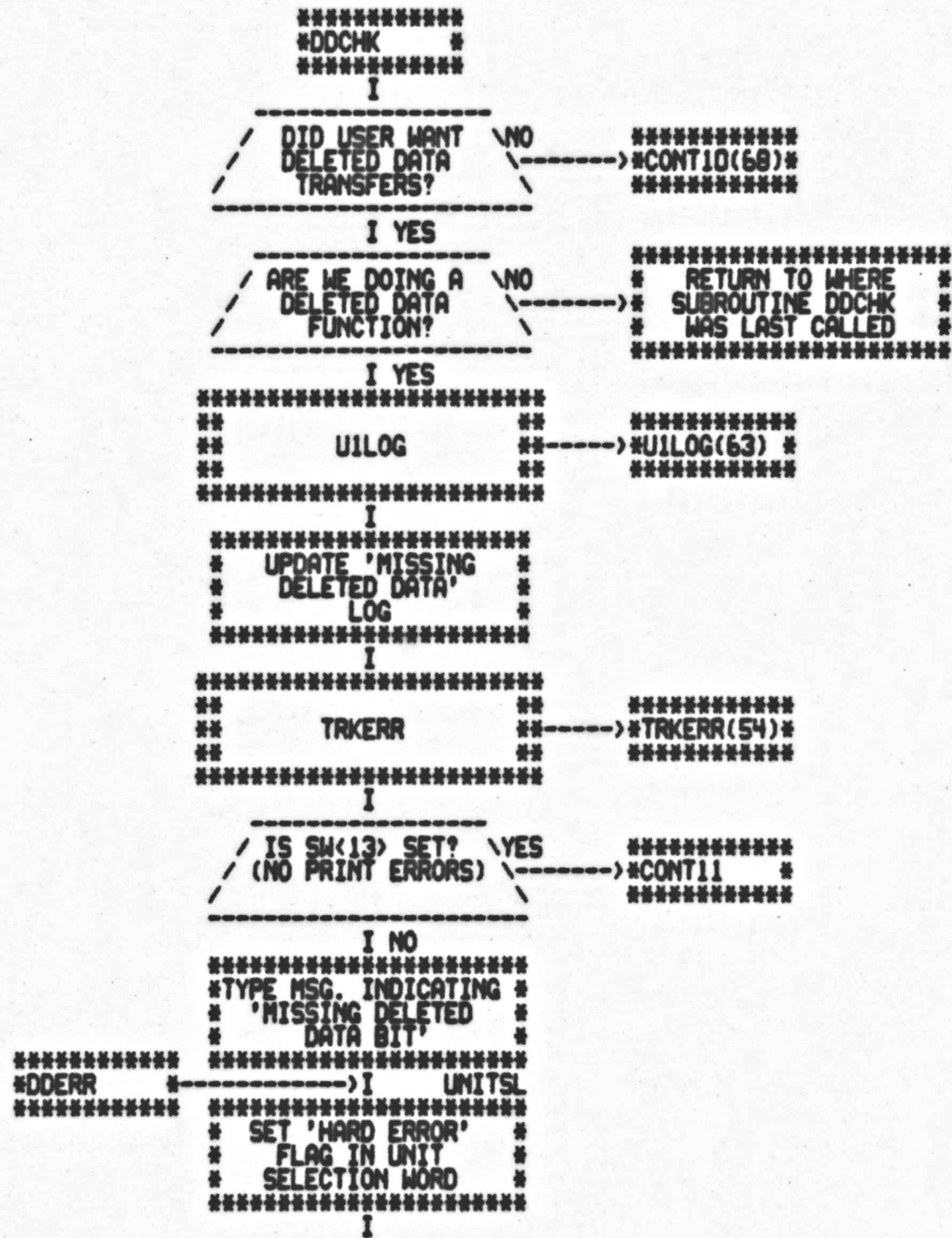
```
*****  
#UILOG #  
*****  
I  
-----  
/ IS UNITS THE \ YES  
/ DRIVE UNDERGOING \----->  
/ TEST? \  
-----  
I NO  
*****  
# SET UP FOR ADDR. #  
# OF UNIT1 ERROR LOGS #  
#(UNIT1 IN OPERATION) #  
*****  
I  
*****  
# RETURN TO WHERE #  
# SUBROUTINE UILOG #  
# WAS LAST CALLED #  
*****
```

```
*****  
#SEKTYP #  
*****  
I  
*****  
#TYPE MSG. INDICATING #  
# A 'SEEK ERROR' #  
# OCCURRED #  
*****  
I  
*****  
# TYPE THE ADDRESS #  
# OF THE TRACK FROM #  
# WHICH SEEK STARTED #  
*****  
I  
*****  
# RETURN TO WHERE #  
# SUBROUTINE SEKTYP #  
# WAS LAST CALLED #  
*****
```

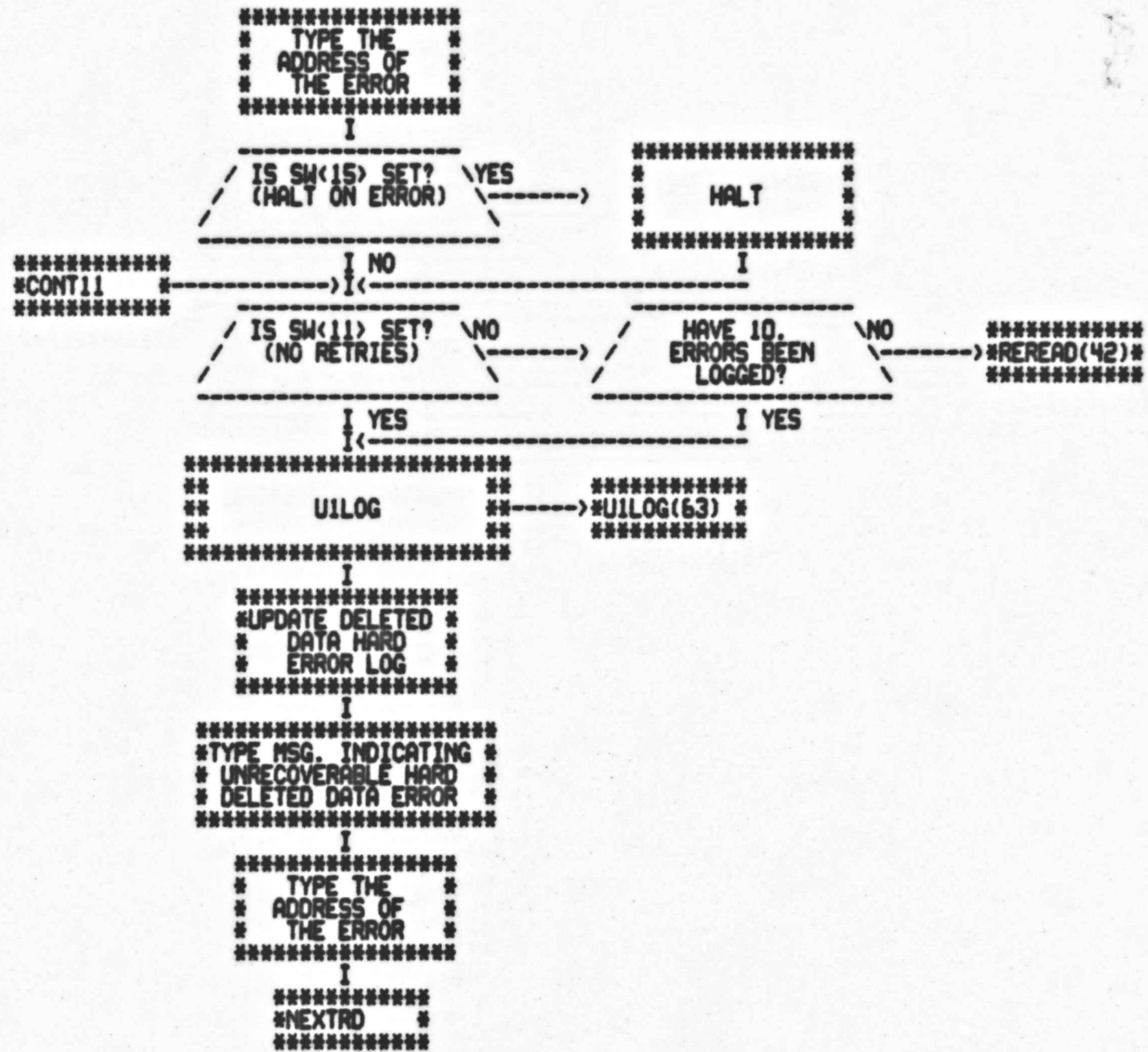


```
*****  
*STATER(76)*  
*****  
I PARLOG  
*****  
* UPDATE THE *  
* PARITY ERROR *  
* COUNTER *  
*****  
I  
*****  
* TYPE OUT THE *  
* CONTENTS OF THE *  
* DEVICE STATUS REG. *  
*****  
I  
-----  
/ HAVE 10. PARITY \ NO  
  ERRORS BEEN      /  
  LOGGED?          /-----  
-----  
I YES  
I HPARLG  
*****  
* UPDATE THE *  
* HARD PARITY *  
* ERROR LOG *  
*****  
I  
*****  
*TYPE MSG. INDICATING *  
* 'UNRECOVERABLE *  
* PARITY ERROR' *  
*****  
HALT I  
*****  
*START(01) *  
*****
```

-----> \*\*\*\*\*  
\*TYPE MSG. INDICATING \*  
\* A 'PARITY ERROR' \*  
\* OCCURRED \*  
\*\*\*\*\*  
I  
\*\*\*\*\*  
\* RETURN TO WHERE \*  
\* SUBROUTINE STATER \*  
\* WAS LAST CALLED \*  
\*\*\*\*\*







```
*****  
#CONT10 #  
*****  
I  
-----  
/ ARE WE DOING \ NO  
/ A DELETED DATA \-----> *****  
/ FUNCTION? \-----> # RETURN TO WHERE #  
# SUBROUTINE DDCHK #  
# WAS LAST CALLED #  
-----  
I YES  
*****  
** UILOG **-----> *****  
** UILOG(63) **  
** *****  
*****  
I  
*****  
#UPDATE THE UN- #  
# EXPECTED 'DD' #  
#BIT SET ERR LOG#  
*****  
I  
*****  
# SET 'HARD ERROR' #  
# FLAG IN UNIT #  
# SELECTION WORD #  
*****  
I  
*****  
** TRKERR **-----> *****  
** TRKERR(54)**  
** *****  
*****  
I  
-----  
/ IS SW<13> SET? \ YES  
/ (NO PRINT ERRORS) \-----> *****  
#CONT11 #  
-----  
I NO  
*****  
#TYPE MSG. INDICATING #  
# 'UNEXPECTED' DELETED #  
# DATA BIT OCCURRED #  
*****  
I  
*****  
#DDERR #  
*****
```



```

*****
#EMPBUF(29)#
*****
I
*****
# CLEAR THE #
# BYTE & ERROR #
# COUNTERS #
*****
I
*****
# SET A 1ST ERROR #
# FLAG (BIT7 OF R1) #
*****
I
*****
** ADJSUM **----->#ADJSUM(57)#
*****
I
*****
# CLEAR THE #
# CHECKSUM #
# HOLDER #
*****
I EMPDNE
*****
# PUT A RETURN ON THE #
# STACK FOR BUFFER #
# EMPTY DONE & NO ERRS #
*****
I EMPE(76)
*****
# PUT A RETURN ON THE #
# STACK IF ERRORS EX- #
# ISTED ON BUFFER EMPTY#
*****
I
*****
# SET CPU 'PS' LEVEL #
# TO 4 TO ALLOW INTER- #
# RUPT ON BUFFER EMPTY #
*****
I
*****
#SET 'INTERRUPT ENAB',#
# 'EMPTY BUFFER' AND #
# 'GO' BITS #
*****
I

```

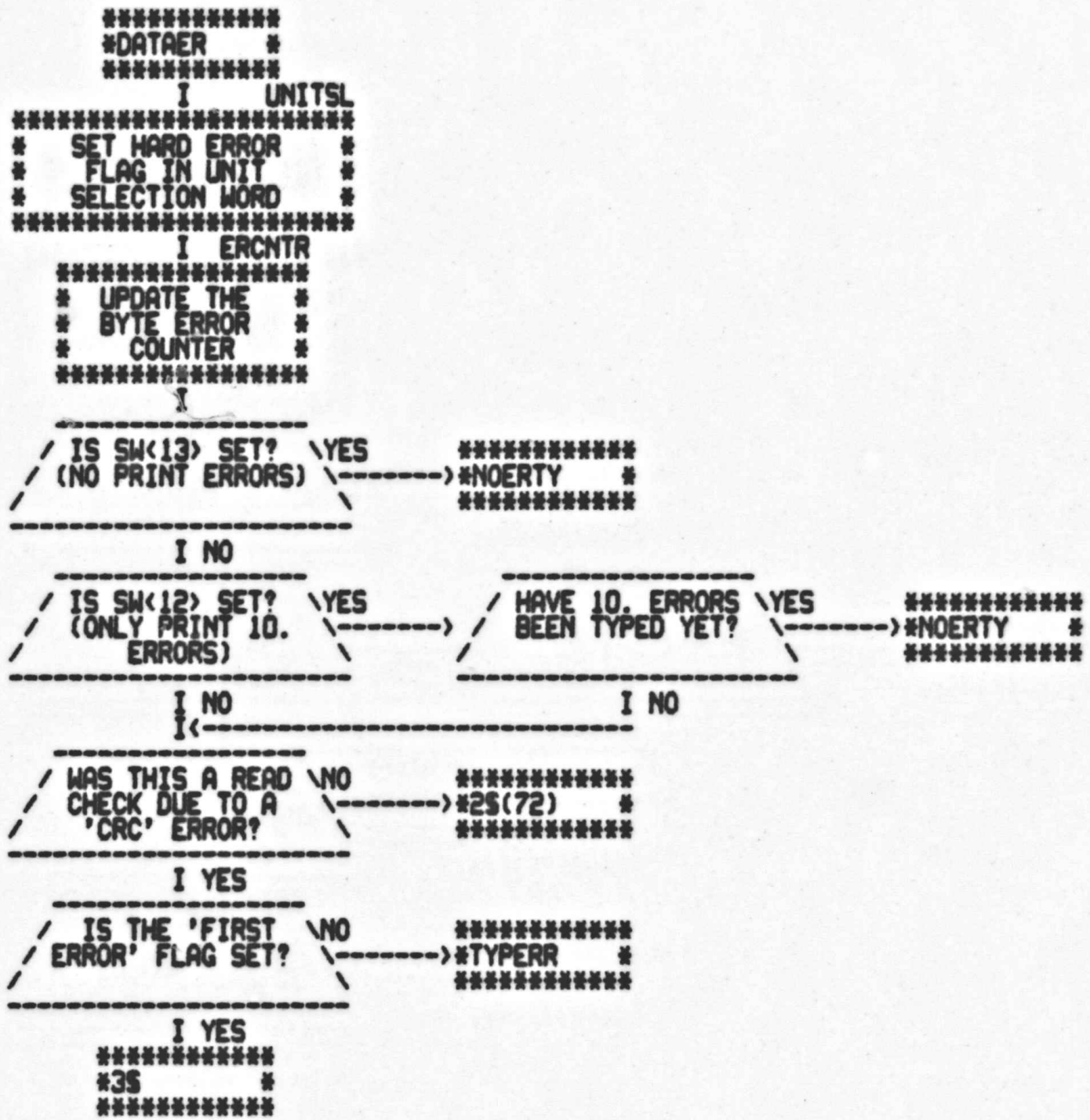
```

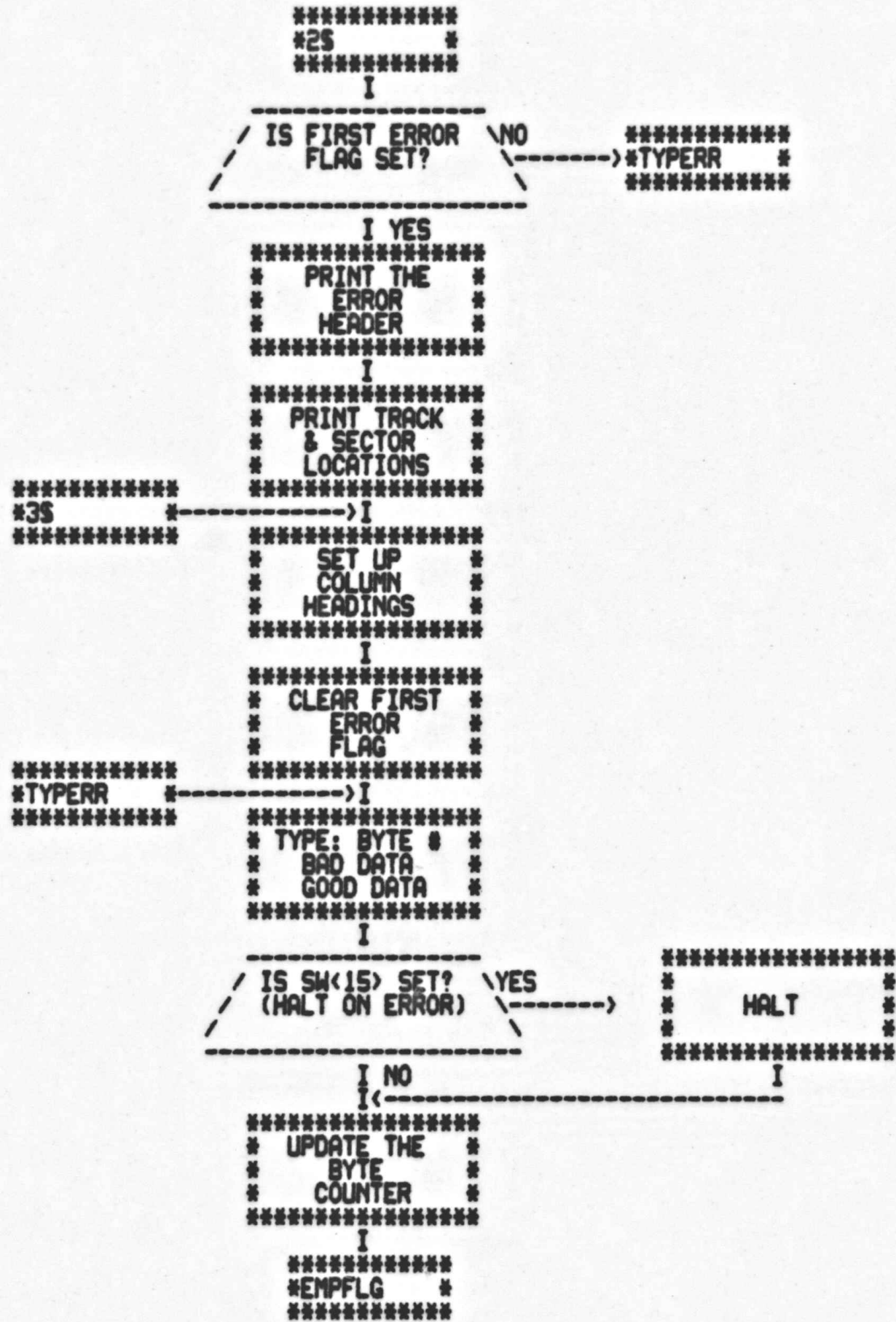
-----)I<-----
*****
*****EMPFLG*****
*****
*****
**          **          *****
**      TRCK          **----->#TRCK(58) **
**          **          *****
*****
I
*****
*  UNLOAD 1 BYTE  *
*  FROM THE DISKETTE *
*  DATA BUFFER (RXDB) *
*****
I
*****
*  ADD THIS BYTE *
*  VALUE TO THE *
*  CHECKSUM *
*****
I
-----
/ IS DATA EMPTIED \ NO
/ FROM BUFFER SAME AS \----->#DATAER(71)#
/ THAT FROM PREVIOUSLY \
-----
I YES
*****
*  UPDATE THE *
*  BYTE *
*  COUNTER *
*****
I
-----

```

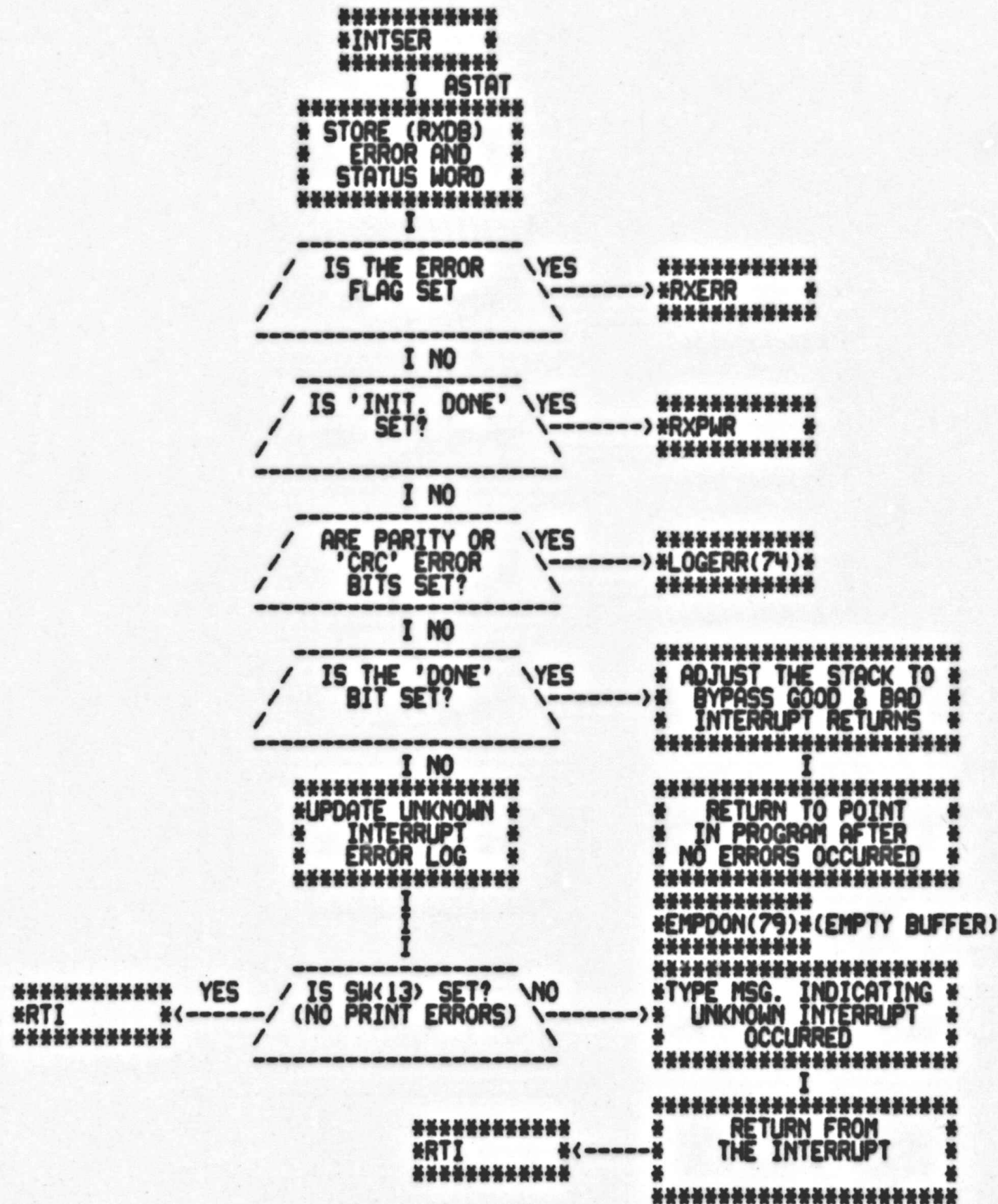
NOTE: THE LOOP FROM SUBROUTINE 'TRCK' TO 'UPDATE THE BYTE COUNTER' CONTINUES TILL 128(10) BYTES OF DATA HAVE BEEN FULFILLED, AT WHICH POINT IF THE 'IE' BIT HAD BEEN SET, AN INTERRUPT TO SUBROUTINE 'INTSER(V)' WOULD HAVE OCCURRED.

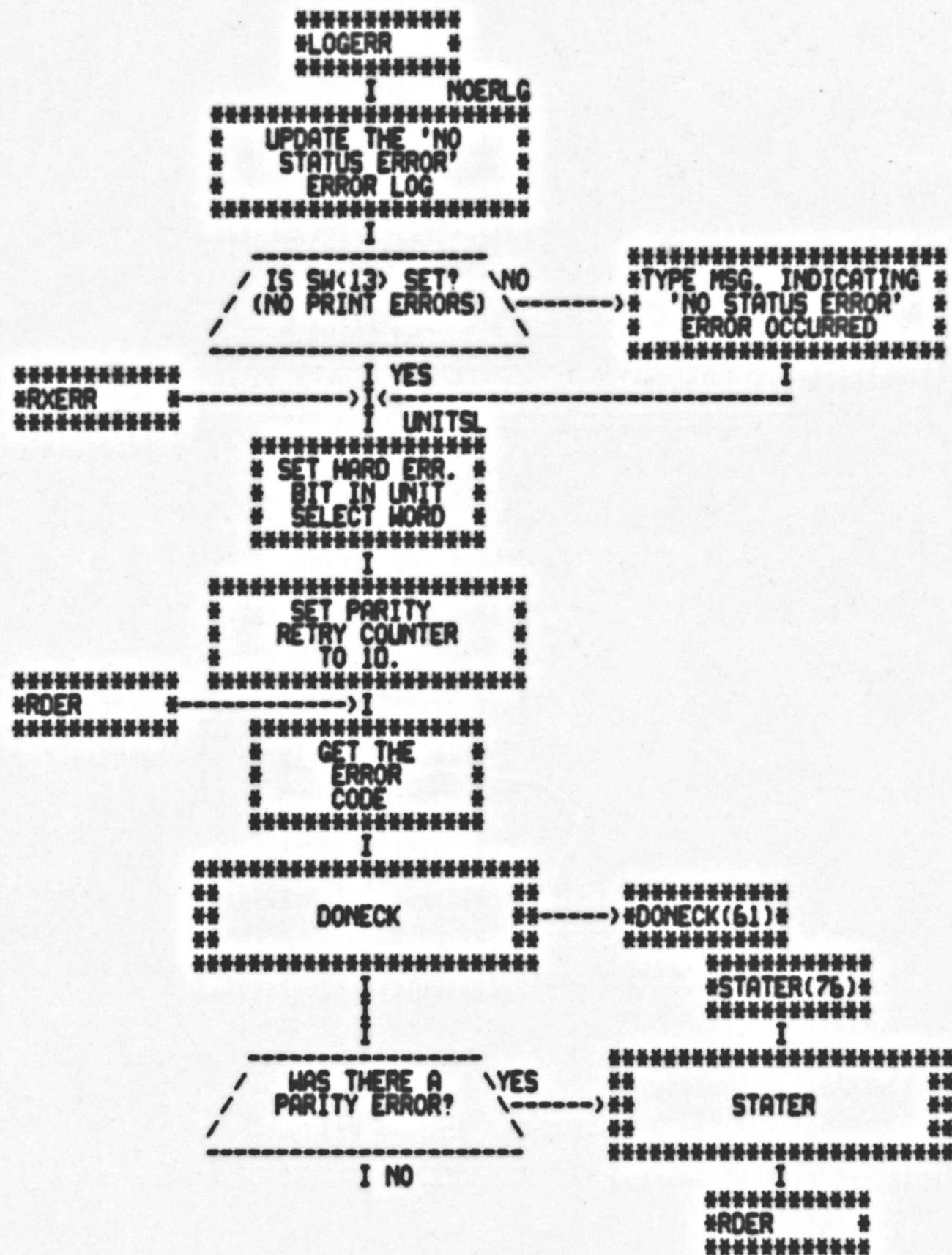








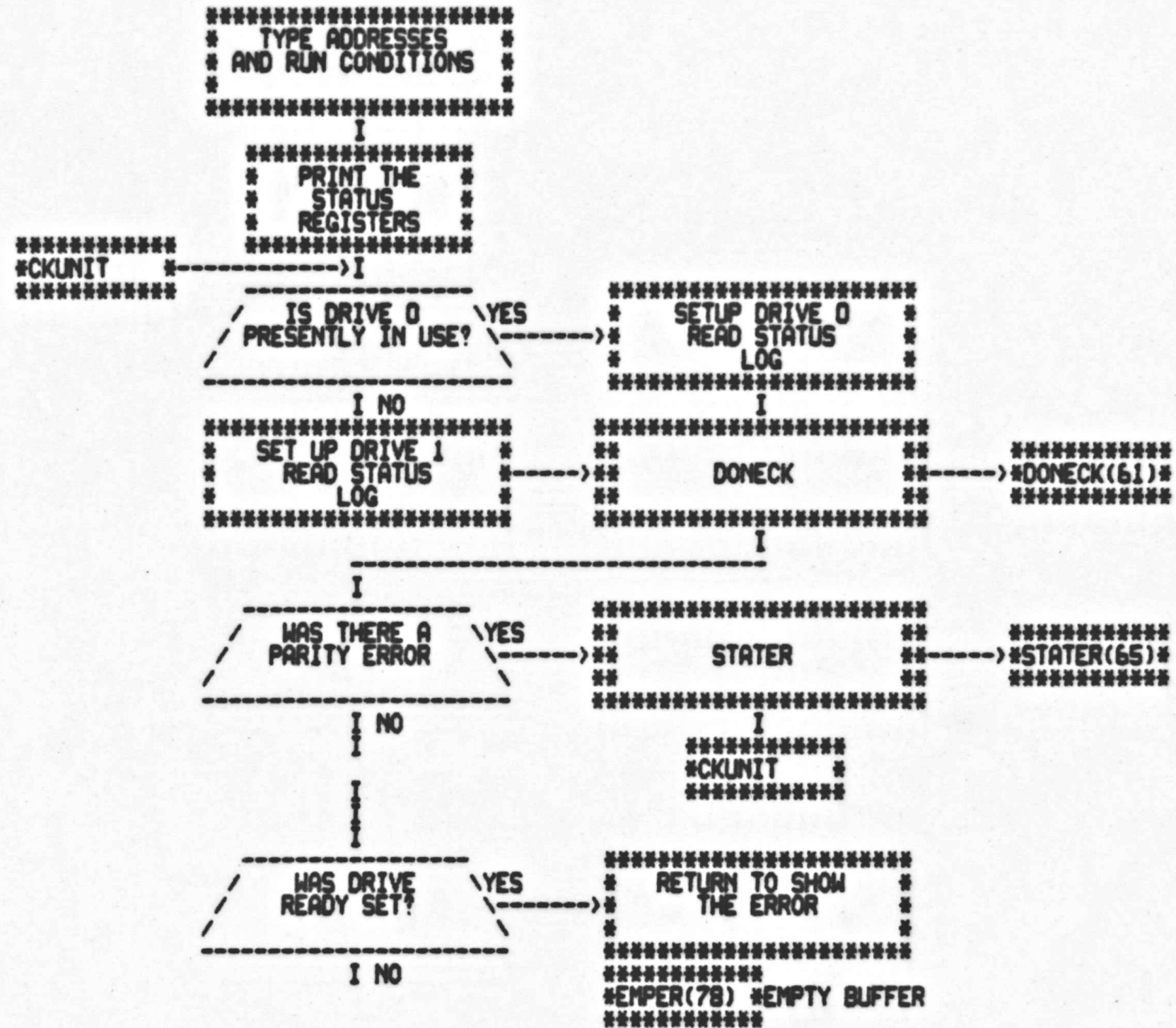






```

                                BSTAT
*****
*   SAVE THE                     *
*   ERROR CODE                   *
*****
                                I
-----
/ IS THERE A DEFINITE \ NO -----> *****
/   ERROR CODE?   \   >#NOPRNT *
-----
                                I YES
*****
*   ADJUST ERROR                 *
*   CODE TO OBTAIN              *
*   TABLE EVEN ADDR.          *
*****
                                I
*****
*   OBTAIN THE                   *
*   CORRECT ERROR               *
*   CODE LOG                    *
*****
                                I
*****
*   UPDATE THE                   *
*   CORRECT ERROR               *
*   CODE LOG                    *
*****
*****
#NOPRNT #-----> I
*****
-----
/ IS SW<13> SET? \ YES -----> *****
/ (NO PRINT ERRORS) \ >#CKUNIT *
-----
                                I NO
*****
*   TYPE ERROR                   *
*   HEADER                       *
*****
                                I
*****
*   TYPE PASSES                  *
*   COMPLETED AT               *
*   TIME OF ERROR               *
*****
                                I
*****
*   TYPE COMMAND                 *
*   STATUS REGISTER             *
*   FUNCTION ETC.              *
*****
                                I
```





```
*****  
#TYPE MSG. INDICATING #  
# DRIVE WASN'T #  
# READY #  
*****
```

I

```
*****  
** DELUNIT **----->#DELUNI(30)#  
** **  
*****
```

I

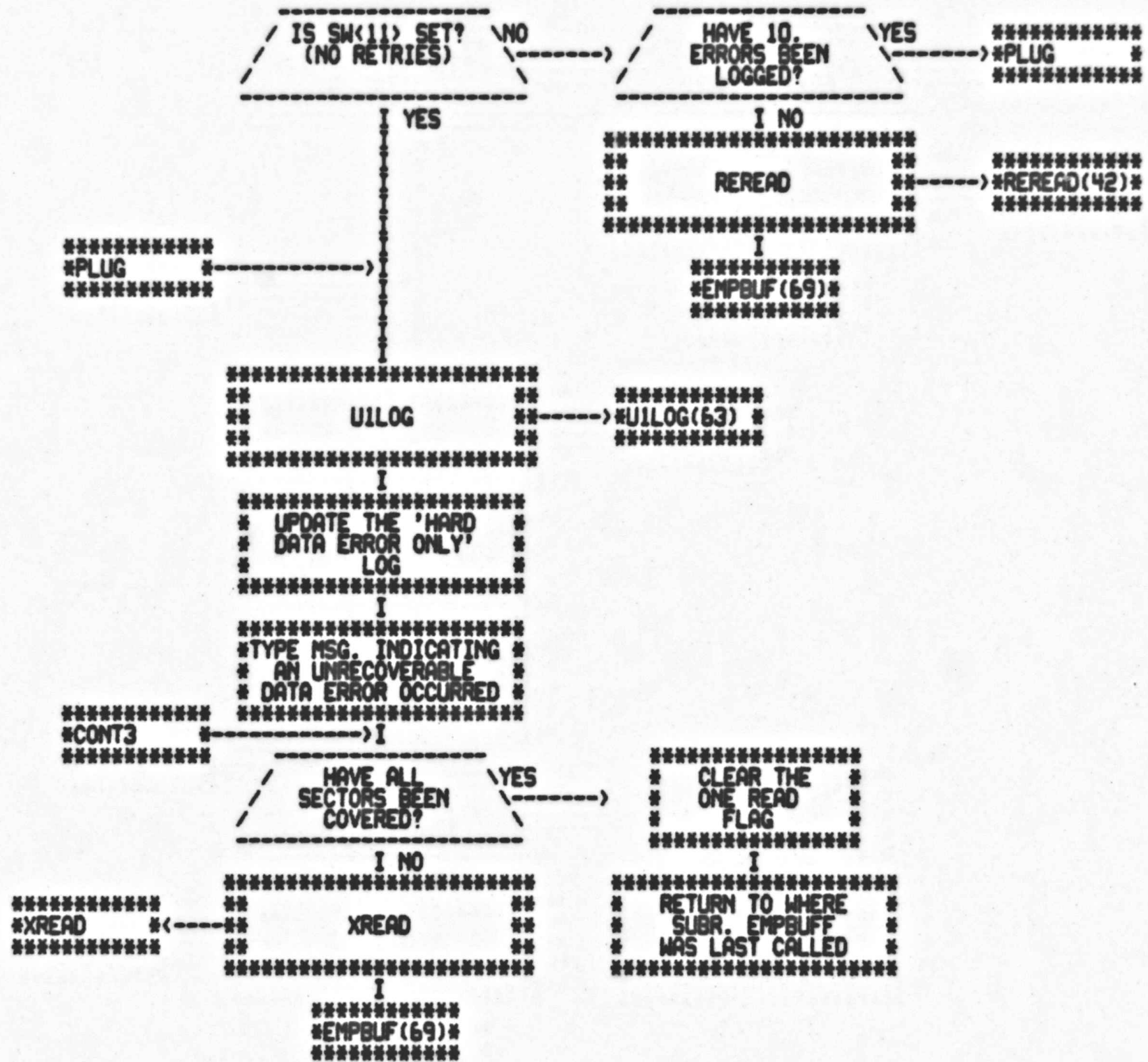
```
*****  
# RETURN TO REDO #  
# THE ERROR & SHOW #  
# IT #  
*****
```

```
*****  
#EMPER(78) #EMPTY BUFFER  
*****
```

```
*****  
#EMPER(76) #  
*****  
I  
*****  
# CORRECT STACK FOR #  
# PROPER RETURN FROM #  
# 'PARTST' #  
*****  
I  
*****  
# SET UP MESSAGE #  
# ADDRESSES FOR #  
# ERROR REPORTING #  
*****  
I PCONT  
*****  
# SET UP A 'NOT HARD #  
# ERROR' RETURN FROM #  
# 'PARTST' THRU 'PCONT' #  
*****  
I  
*****  
#PARTES(28) #  
*****
```



```
*****  
#EMPDOON(73)*  
*****  
I  
-----  
/ WAS THIS RDCHECK \ YES *****  
CAUSED BY A CRC  ----->#CRCER(46) *  
ERROR              \ NO *****  
-----  
I NO  
-----  
/ WERE THERE \ NO *****  
ANY OTHER ERRORS? ----->#CONT3 *  
-----  
I YES  
-----  
/ IS SW<13> SET? \ YES *****  
(NO PRINT ERRORS) ----->#EMPLOG *  
-----  
I NO  
*****  
# PRINT THE TOTAL *  
# DATA ERROR COUNT *  
*****  
I  
*****  
# INDICATE IF THE *  
# CHECKSUM WAS *  
# GOOD OR BAD *  
*****  
*****  
#EMPLOG *----->I  
*****  
** UILOG **----->#UILOG(63) *  
**-----  
I  
*****  
# UPDATE THE *  
# DATA ERROR *  
# LOG *  
*****  
I  
*****  
** TRKERR **----->#TRKERR(54)*  
**-----  
*****  
I
```

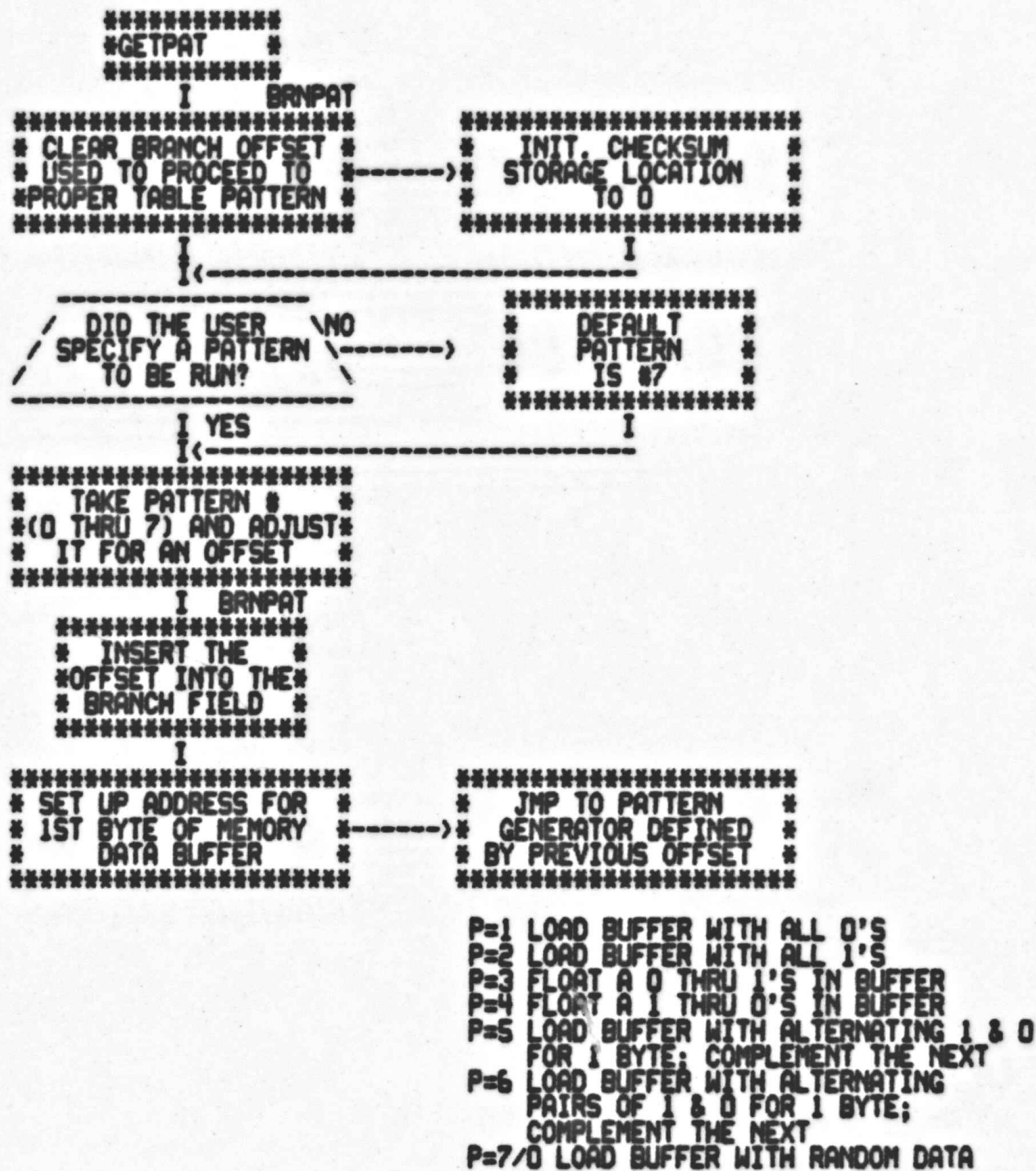












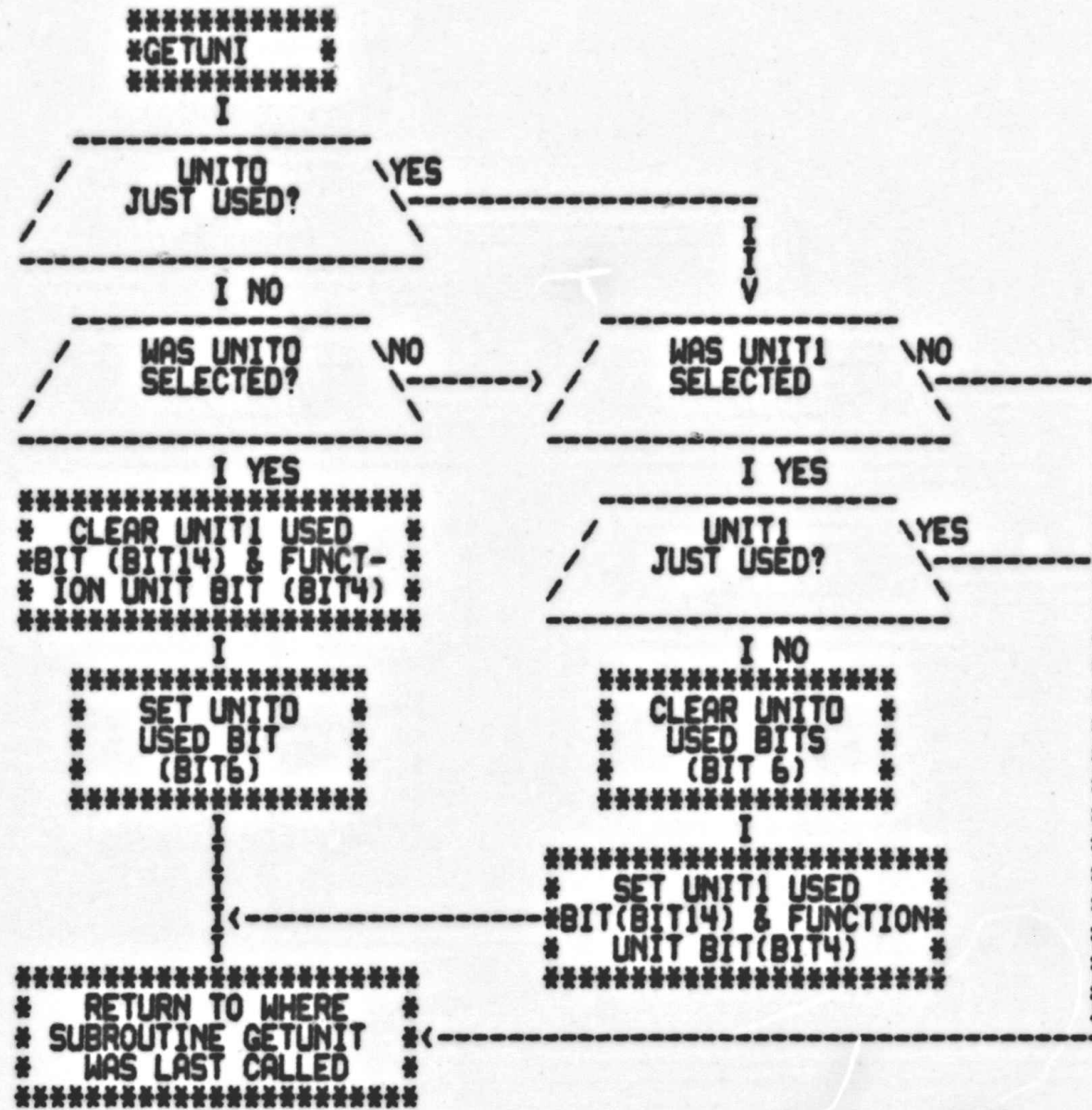
NOTE: WHEN THE PATTERN HAS BEEN COMPLETELY GENERATED  
RETURN WILL BE TO THE INSTRUCTION FOLLOWING THE  
LAST CALL TO GETPAT(TERN)

```
*****  
#INITTR #  
*****  
I  
*****  
# CLEAR 1ST TIME #  
#USED BITS IN LOC. 00:#  
# (BITS 15 & BIT07) #  
*****  
I  
-----  
/ DID THE USER \ NO  
 \ SPECIFY TRACKS? /-----> * SET STARTING TRACK *  
----- * TO 0; SET LAST *  
 * TRACK TO 114(8) *  
----- *  
*****  
I YES I  
I <----- I  
I TARGET  
*****  
# INIT. TARGET #  
# TRACK TO #  
# 1ST TRACK #  
*****  
XID I XOD  
*****  
# SET UP WORKING #  
# LOCS. FOR UPDATING #  
# 1ST TO LAST TRACKS #  
*****  
I  
*****  
# SET UP TOTAL # OF #  
# TRACK MOVEMENTS #  
# (00-ID) #  
*****  
I  
*****  
# SET 1ST TIME #  
# USED BITS IN #  
# LOC. 00 #  
*****  
I  
*****  
# RETURN TO WHERE #  
# SUBR. INITTR(ACKS) #  
# WAS LAST CALLED #  
*****
```



```
*****  
#GETTR #  
*****  
I PRSTRK  
*****  
* SET PRESENT TRACK *  
* TO THE 1ST TARGET *  
* TRACK *  
*****  
I BRNTRK  
*****  
* CLEAR BRANCH OFFSET *  
* USED TO PROCEED TO *  
* PROPER TABLE SEQUENCE *  
*****  
I  
-----  
/ DID THE USER \ NO  
/ SPECIFY A TRACK \-----> * DEFAULT SEQUENCE *  
/ SEQUENCE TO RUN? \ * IS #7 *  
----- *  
I YES I  
I <----- I  
I BRNTRK  
*****  
* INSERT THE OFFSET *  
* INTO THE BRANCH *  
* FIELD *  
*****  
I  
***** *  
* BRANCH TO SOME- *-----> * JMP TO TRACK SEQ. *  
* WHERE IN TABLE * * DEFINED BY PREVI- *  
* SELECTED BY OFFSET * * OUS OFFSET *  
***** *  
*****  
S=1 INCREMENT 1ST TO LAST & RETURN  
S=2 INCREMENT LAST TO 1ST  
S=3 INCREMENT 1ST TO LAST,  
THEN LAST TO 1ST  
S=4 BOUNCE BACK & FOURTH  
BETWEEN 1ST & LAST ONLY  
S=5 BOUNCE BACK & FOURTH BETWEEN  
DECREASING (LAST DOWN) &  
INCREASING (1ST UP)  
S=6 BOUNCE BETWEEN DECREASING  
(LAST DOWN) & 1ST  
S=7 RANDOM TRACK SELECTION
```

RX11 RELIABILITY TEST  
SUBROUTINE GETUNI(T)



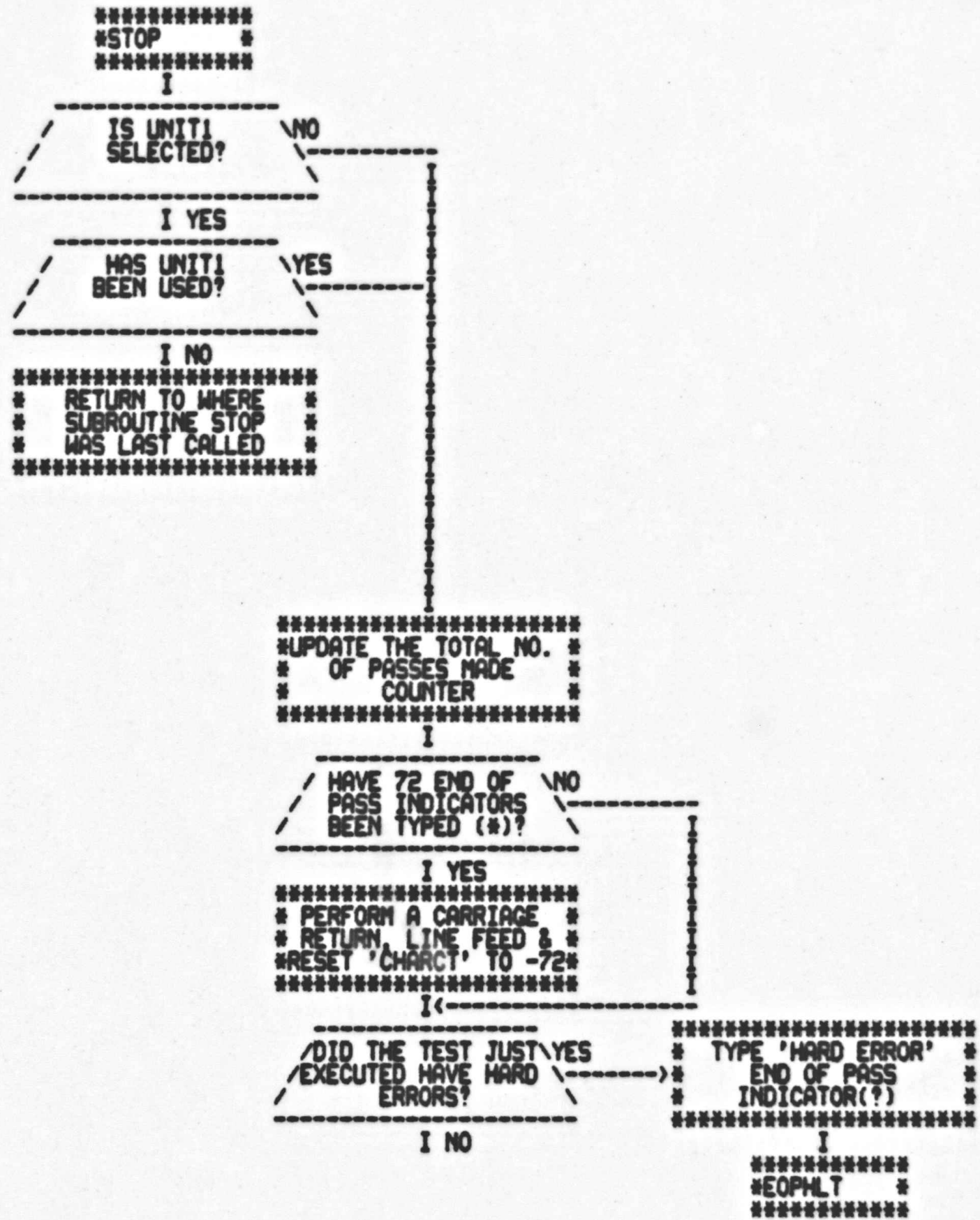


RX11 RELIABILITY TEST  
FLOPPY READCHECK SEQUENCE

```

*****
#RDCHK #
*****
I
*****
# SET FLAG TO #
# INDICATE READ #
# I SECTOR #
*****
I
*****
# READ #
#----->#READ(82) #
*****
I *
*****
#EMPBUF(69)# * IF NO PROBLEM OCCURRED
***** # DURING THE READ CYCLE
# RETURN TO HERE

```





```
*****
* TYPE 'ERROR FREE' *
* END OF PASS INDI- *
* CATOR (*) *
*****
#EOPHLT ----->I
*****
/ IS 'END OF PASS' \ NO
/ HALT SWITCH \ ----->#CONT22
/ SET? (SW<14>) \
-----
I YES
*****
* TYPE NO. OF *
* TEST PASSES *
* COMPLETED *
*****
I
*****
* HALT *
*****
I CHARCT
*****
* CLEAR HOLDER *
* OF 'EOP' INDI- *
* CATOR COUNTS *
*****
#CONT22 ----->I UNITSL
*****
* CLEAR BITS INDI- *
* CATING UNITS 0 & 1 *
* WERE USED *
*****
I
-----
/ ARE WE DOING \ NO
/ A RANDOM DATA \ ----->#HERE(90)
/ PATTERN? \
-----
I YES
*****
** GETPATTERN ** ----->#GETPAT(83)**
*****
I
-----
/ WERE WE \ NO
/ LOADED VIA ACT11? \ ----->#HERE(90)
-----
I YES
```

NOTE: DEPRESSING  
'CONT' SWITCH WILL  
ALLOW GOING ON

E09

RX11 RELIABILITY TEST  
SUBROUTINE STOP

```
*****  
* RETURN VIA *  
* ACT11 *  
* MONITOR *  
*****
```

```
*****  
*HERE *  
*****
```

```
I  
*****  
* RETURN UNDER PRO- *  
* GRAM CONTROL TO *  
* TEST JUST EXECUTED *  
*****
```



RX11 RELIABILITY TEST  
FLOW CHART CROSS REFERENCE LIST

ADJSUM	25	57	69						
ASTAT	73								
BERRS	44								
BRNPAT	83								
BRNTRK	85								
BSTAT	75								
BUFAOR	57								
CALAG	53								
CHARCT	89								
CKREAD	12	18							
CKSUM	57								
CKUNIT	75	76	76						
COMMO	27	43	59						
CONT1	43	44							
CONT10	66	68							
CONT11	66	67	68						
CONT2	44	44							
CONT22	89	89							
CONT3	79	80							
CONT6	49	50							
CONT7	50	51							
CRCER	46	46	79						
DATAER	70	71							
DATCRC	46	47	48						
DDCHK	44	66							
DDERR	66	68							
DELLNI	23	30	77						
DONECK	27	43	52	61	74	76	81		
DRVSWP	12	23							
DRY1	04	05							
EMPBUF	29	46	69	80	80	87			
EMPDNE	69								
EMPDON	73	79							
EMPER	69	76	77	78					
EMPFLG	70	72							
EMPLOC	79	79							
EOPHLT	88	89							
ERCNTR	71								
ERDUMP	30	31							
ERFLGS	52	52							
ERRCRC	45	46							
FILLBU	25	29							
FILLDO	25	26							
FILLER	25	28							
GETPAT	13	16	18	19	21	22	23	83	89
GETSEC	25	56	82						
GETTR	13	14	16	18	19	21	22	23	85
GETTRA	14	14							
GETTRK	13	14							
GETUNI	13	16	18	19	21	22	23	23	86
HARDER	33								
HARDSK	40	34							







MAINDEC-11-DZRXA-D MACY11 27(663) 17-NOV-75 15:06  
DZRXP.P11 TABLE OF CONTENTS

SEQ 0111

44	BASIC DEFINITIONS
2051	TYPE ROUTINE
2143	BINARY TO OCTAL (ASCII) AND TYPE
2221	SAVE AND RESTORE RD-RS ROUTINES
2267	TRAP DECODER
2283	TRAP TABLE
2299	POWER DOWN AND UP ROUTINES
2341	SINGLE LENGTH BINARY TO DECIMAL ASCII ROUTINE
2359	DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE





55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108

000000  
000001  
000002  
000003  
000004  
000005  
000006  
000007

000000  
000040  
000100  
000140  
000200  
000240  
000300  
000340

100000  
040000  
020000  
010000  
004000  
002000  
001000  
000400  
000200  
000100  
000040  
000020  
000010  
000004  
000002  
000001

100000  
040000

.\*GENERAL PURPOSE REGISTER DEFINITIONS

R0 = X0 :: GENERAL REGISTER  
R1 = X1 :: GENERAL REGISTER  
R2 = X2 :: GENERAL REGISTER  
R3 = X3 :: GENERAL REGISTER  
R4 = X4 :: GENERAL REGISTER  
R5 = X5 :: GENERAL REGISTER  
R6 = X6 :: GENERAL REGISTER  
R7 = X7 :: GENERAL REGISTER  
.EQUIV R6, SP :: STACK POINTER  
.EQUIV R7, PC :: PROGRAM COUNTER

.\*PRIORITY LEVEL DEFINITIONS

PR0 = 0 :: PRIORITY LEVEL 0  
PR1 = 40 :: PRIORITY LEVEL 1  
PR2 = 100 :: PRIORITY LEVEL 2  
PR3 = 140 :: PRIORITY LEVEL 3  
PR4 = 200 :: PRIORITY LEVEL 4  
PR5 = 240 :: PRIORITY LEVEL 5  
PR6 = 300 :: PRIORITY LEVEL 6  
PR7 = 340 :: PRIORITY LEVEL 7

.\*"SWITCH REGISTER" SWITCH DEFINITIONS

SW15 = 100000  
SW14 = 40000  
SW13 = 20000  
SW12 = 10000  
SW11 = 4000  
SW10 = 2000  
SW09 = 1000  
SW08 = 400  
SW07 = 200  
SW06 = 100  
SW05 = 40  
SW04 = 20  
SW03 = 10  
SW02 = 4  
SW01 = 2  
SW00 = 1  
.EQUIV SW09, SW9  
.EQUIV SW08, SW8  
.EQUIV SW07, SW7  
.EQUIV SW06, SW6  
.EQUIV SW05, SW5  
.EQUIV SW04, SW4  
.EQUIV SW03, SW3  
.EQUIV SW02, SW2  
.EQUIV SW01, SW1  
.EQUIV SW00, SW0

.\*DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15 = 100000  
BIT14 = 40000



109 020000  
 110 010000  
 111 004000  
 112 002000  
 113 001000  
 114 000400  
 115 000200  
 116 000100  
 117 000040  
 118 000020  
 119 000010  
 120 000004  
 121 000002  
 122 000001  
 123  
 124  
 125  
 126  
 127  
 128  
 129  
 130  
 131  
 132  
 133  
 134  
 135  
 136  
 137  
 138  
 139  
 140  
 141  
 142  
 143  
 144  
 145  
 146  
 147  
 148  
 149  
 150  
 151  
 152  
 153  
 154  
 155  
 156  
 157  
 158  
 159  
 160  
 161  
 162

000004  
 000010  
 000014  
 000014  
 000014  
 000014  
 000020  
 000024  
 000030  
 000034  
 000060  
 000064  
 000240

BIT13= 20000  
 BIT12= 10000  
 BIT11= 4000  
 BIT10= 2000  
 BIT09= 1000  
 BIT08= 400  
 BIT07= 200  
 BIT06= 100  
 BIT05= 40  
 BIT04= 20  
 BIT03= 10  
 BIT02= 4  
 BIT01= 2  
 BIT00= 1  
 .EQUIV BIT09,BIT9  
 .EQUIV BIT08,BIT8  
 .EQUIV BIT07,BIT7  
 .EQUIV BIT06,BIT6  
 .EQUIV BIT05,BIT5  
 .EQUIV BIT04,BIT4  
 .EQUIV BIT03,BIT3  
 .EQUIV BIT02,BIT2  
 .EQUIV BIT01,BIT1  
 .EQUIV BIT00,BIT0

.\*BASIC "CPU" TRAP VECTOR ADDRESSES  
 ERRVEC= 4 : TIME OUT AND OTHER ERRORS  
 RESVEC= 10 : RESERVED AND ILLEGAL INSTRUCTIONS  
 TBITVEC= 14 : "T" BIT  
 TRTVEC= 14 : TRACE TRAP  
 BPTVEC= 14 : BREAKPOINT TRAP (BPT)  
 IOTVEC= 20 : INPUT/OUTPUT TRAP (IOT) \*\*SCOPE\*\*  
 PWRVEC= 24 : POWER FAIL  
 EMTVEC= 30 : EMULATOR TRAP (EMT) \*\*ERROR\*\*  
 TRAPVEC= 34 : "TRAP" TRAP  
 TKVEC= 60 : TTY KEYBOARD VECTOR  
 TPVEC= 64 : TTY PRINTER VECTOR  
 PIRQVEC= 240 : PROGRAM INTERRUPT REQUEST VECTOR

;SPECIAL EQUATES

RDOSTAT = 13  
 RDISTAT = 33  
 RDER = 17  
 DONEBIT = 40  
 FBIE = 101  
 EBIE = 103  
 WRTIE = 105  
 RDIE = 107  
 WTDIE = 115  
 RECAL = 40001  
 PR4 = 200

```

163          000340          PR7      =340
164          000000          OPEN     =0
165
166          000000          . =0
167 000000 000000 000000          .WORD 0,0
168
169          000024          . =24
170 000024 012734          $PWRDN
171 000026 000340          340
172
173          000034          . =34
174 000034 012676          $TRAP          ;ADDRESS OF TRAP SERVICE
175 000036 000340          340
176
177          000046          . =46
178 000046 010674          LOGICAL          ;ACT11 EOP HOOK
179
180          000052          . =52
181 000052 000000          .WORD 0
182
183          000174          . =174
184 000174 000000          DISREG:         0
185 000176 000000          SWREG:         0
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202          000200          . =200
203 000200 000402          BR 1$
204 000202 000403          BR 2$
205 000204 000404          BR 3$
206 000206 000137 001220          1$: JMP SA200          ;OPERATOR SELECTED CONDITIONS
207 000212 000137 002532          2$: JMP RESTART      ;RESTART PROGRAM WITH PREVIOUS CONDITIONS
208 000216 000137 017622          3$: JMP ERDUMP        ;STATISTICAL ERROR PRINT OUT
209
210
211
212
213
214
215
216

```

\*\*\*\*\*

; STARTING ADDRESSES

```

; INITIAL START =200 /CLEARS ALL ERROR LOGS, RESETS COUNTERS, AND ALLOWS FOR
; /SELECTION OF DRIVES AND TEST CONDITIONS
; RESTART =202 /USES PREVIOUS INITIAL START DRIVES AND TEST
; /SELECTION AND CONTINUES TO ACCUMULATE STATISTICAL DATA
; ERROR REPORT =204 /PRINTS OUT ALL RUN AND ERROR CONDITIONS
; /ACUMULATED OVER THE RUN OF THE PROGRAM.

```

\*\*\*\*\*

```

; THE FOLLOWING LOCATIONS "OD", "ID", "FIRST" AND "LAST"
; MAY BE CHANGED BY THE OPERATOR MANUALLY HOWEVER FOLLOWING THESE RESTRICTIONS.
; (IF THESE LOCATIONS ARE LEFT CLEARED, MAX AND MIN VALUES ARE ASSUMED)
;

```





271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324

IF THE RX11 SYSTEM UNDER TEST IS JUMPERED FOR REGISTER  
ADDRESSES OTHER THAN STANDARD: RXCS = 177170  
RXDB = 177172  
PLACE IN THESE TWO WORDS THE CORRESPONDING NEW ADDRESSES FOR  
THE REGISTERS. THE PROGRAM WILL TYPE OUT THE REGISTER  
ADDRESSES FOR VERIFICATION BY THE OPERATOR. IF THE ADDRESSES  
ARE INCORRECT (I.E. - DON'T MATCH THE HARDWARE) AND THE PROGRAM  
IS RUN WITH THE INCORRECT ADDRESSES, A 'HUNG' CONDITION WILL BE  
REPORTED AND THE PROGRAM WILL 'HALT' AS THE RX11 WILL NOT BE  
ABLE TO RESPOND TO COMMANDS.

\*\*\*\*\*

001206 177170  
001210 177172

RXCS: 177170  
RXDB: 177172

BIT ASSIGNMENT IN THE RXCS REGISTER.

- KEY: R - READ ONLY BIT
- W - WRITE ONLY BIT
- 15 - R - ERROR FLAG
- 14 - W - INITIALIZE (RECALIBRATE)
- 13 - NOT USED
- 12 - NOT USED
- 11 - R - TRANSFER REQUEST (TR) FLAG
- 10 - R/W - INTERRUPT ENABLE
- 9 - R - DONE FLAG
- 8 - W - UNIT SELECT
- 7 - W - FUNCTION
- 6 - W - FUNCTION
- 5 - W - FUNCTION
- 4 - W - FUNCTION
- 3 - W - FUNCTION
- 2 - W - FUNCTION
- 1 - W - GO
- 0 - W - GO

FUNCTION CODES:

- 0 + GO = FILL BUFFER
- 2 + GO = EMPTY BUFFER
- 4 + GO = WRITE SECTOR
- 6 + GO = READ SECTOR
- 10 (NOT USED)
- 12 + GO = READ STATUS "A"
- 14 + GO = WRITE DELETED DATA
- 16 + GO = READ STATUS "B" (CODES)

THE FOLLOWING BIT ASSIGNMENTS REPRESENTS THE STATUS AT THE END OF A  
FUNCTION (EXCEPT FUNCTION "READ STATUS B") DISPLAYED IN THE  
RX DATA BUFFER (RXDB).

- 15 - NOT USED
- 10 - NOT USED
- 8 - NOT USED
- 7 - SELECTED DRIVE READY \*



370  
371  
372  
373  
374  
375  
376  
377  
378

001212 000000  
001214 177570  
001216 177570

DTESTP: .WORD 0 ;TEST SELECTION WORD  
SMR: .WORD DSMR ;ADDRESS OF SWITCH REGISTER  
DISPLAY: .WORD DDISP ; ADDRESS OF DISPLAY REGISTER

0 - DELETED DATA  
1 - WRITE PROTECT ERROR (WHEN AVAILABLE)  
2 - INITIALIZE DONE \*\*  
3 - PARITY ERROR  
4 - CRC ERROR  
\* VISIBLE ONLY IF THE FUNCTION WAS A #12 "READ STATUS A"  
\*\* VISIBLE ONLY AFTER AN INITIALIZE [KEY] OR [PROGRAMMED] WAS ISSUED.

\*\*\*\*\*

:START OF OPERATOR SELECTABLE TEST, DATA PATTERNS, AND DRIVE UNIT CONDITIONS  
:SET THE TEST CONDITIONS WANTED (OR LEAVE 0) IN LOCATION CALLED  
:"DTESTP" (LOC. 1212), BEFORE STARTING THE PROGRAM. WHEN THE PROGRAM  
:STARTS IT WILL TYPE OUT THE TEST CONDITIONS IT IS OPERATING UNDER.

SWITCH REGISTER BITS

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0  
U1 U0 NOT USED D P P P T T T S S S

U1 SELECT DRIVE UNIT 1  
U0 SELECT DRIVE UNIT 0  
IF NEITHER DRIVE IS SPECIFIED, PROGRAM WILL SELECT ALL DRIVES  
THAT ARE READY

D = DELETED DATA FUNCTIONS

IF THIS SWITCH IS ON ALL READ AND WRITE FUNCTIONS WILL BE IN THE  
DELETED DATA MODE

P = DATA PATTERN SELECTION  
0 DO PATTERN 7 (RANDOM)  
1 ZEROS  
2 ONES  
3 FLOATING ZERO  
4 FLOATING ONE  
5 125  
6 314  
7 RANDOM

T = FUNCTIONAL TESTS  
0 DO TEST 7 (WRITE/READ/READ CHECK)  
1 WRITE ONLY  
2 WRITE/READ

379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432

WRITE/READ CHECK  
READ CHECK ONLY  
READ ONLY  
WRITE/READ CHECK ON ALTERNATE DRIVES  
WRITE/READ/READ CHECK \*

WRITE/READ CHECK  
READ CHECK ONLY  
READ ONLY  
WRITE/READ CHECK ON ALTERNATE DRIVES  
WRITE/READ/READ CHECK \*

: \* NOTE: THE FIRST HALF OF TEST 7 FORCES THE TRACK SEQUENCE  
: TO INCRIMENT UP THROUGH ALL THE TRACKS DOING WRITE / READ CHECK FUNCTIONS.  
: THIS VERIFIES THAT ALL THE TRACKS ARE ACCESSABLE.  
: THE SECOND HALF OF THE PASS WILL USE THE SEQUENCE SELECTED BY THE  
: OPERATOR AS INDICATED BELOW, AND ONLY READ CHECK THE DATA JUST WRITTEN.  
: THIS VERIFIES THAT THE DATA CAN BE READ FROM A TRACK AFTER THE HEAD HAS BEEN  
: MOVED AWAY FROM AND BACK TO THAT TRACK. AT THE COMPLETION OF THE PASS  
: THE DELETED DATA BIT IN TEST CONDITIONS IS COMPLEMENTED AND THE NEXT  
: PASS WILL BE RUN UNDER THIS NEW CONDITION.

S = TRACK SEQUENCING  
D-INCR/DECR  
DO SEQUENCE 7 (RANDOM)  
INCREMENT  
DECREMENT  
INCREMENT/DECREMENT  
BOUNCE  
DECREASING BOUNCE  
STROBE  
RANDOM

\*\*\*\*\*

: SET THE OPERATING CONDITIONS IN THE SWITCH REGISTER (HARDWARE).  
: OR THE SOFTWARE SWITCH REGISTER (LOC. 176) BEFORE STARTING THE PROGRAM.

: SWITCHES DO THE FOLLOWING WHEN SET TO "1".

SW 15 =HALT ON ERROR  
SW 14 =HALT AT END OF TEST  
SW 13 =DON'T PRINT ERROR MESSAGE  
SW 12 =TYPE ONLY 10 DATA ERRORS  
SW 11 =NO RETRY ON ERROR. LOG HARD ERROR  
  
SW 8 =NO RECALIBRATION ON SEEK ERRORS  
  
SW15 - SW0 =SELECT SOFTWARE SWITCH REGISTER

\*\*\*\*\*

```
001220 012706 001200 SA200: MOV #STACK, SP
001224 012746 000340 MOV #PR7, -(SP)
001230 012746 001236 MOV #2S, -(SP)
001234 000002 RTI
001236 012737 001256 000004 2S: MOV #3S, 4 ;SET TIME OUT VECTOR
```







487	001514	000000			MLT1:	HALT	:WAIT FOR USER RESPONSE
488	001516	105777	177466		NOACT:	TSTB DRXDB	:IS DRY BIT SET FOR UNIT 0 AFTER RECAL
489	001522	100003				BPL 2\$	:IF NOT SKIP NEXT INSTRUCTION
490	001524	052737	000200	010464		BIS #BIT7,UNITSEL	:YES,SET UNIT 0 READY BIT
491	001532	032737	140000	001212	2\$:	BIT #140000,DTESTP	:WERE ANY DRIVES SPECIFIED
492	001540	001445				BEQ NOSEL	:NO,SEE IF UNIT 1 IS READY
493	001542	005737	001212			TST DTESTP	:WAS UNIT 1 SELECTED
494	001546	100044				BPL OPCOND	:NO,IT MUST BE UNIT 0
495	001550	004737	001572			JSR PC,DRY1	:SEE IF UNIT 1 IS READY
496	001554	032737	040000	001212		BIT #40000,DTESTP	:WAS UNIT 0 SELECTED
497	001562	001036				BNE OPCOND	:YES,CONTINUE
498	001564	105037	010464			CLRB UNITSEL	:NO,CLEAR UNIT 0 READY BIT
499	001570	000433				BR OPCOND	
500							
501	001572	012777	000033	177406	DRY1:	MOV #RD1STAT,DRXCS	:READ STATUS OF UNIT 1
502	001600	004737	006714			JSR PC,DONECK	:WAIT FOR DONE BIT
503	001604	132777	000002	177376		BITB #2,DRXDB	:IS PARITY ERROR BIT SET
504	001612	001411				BEQ 2\$	:NO,CONTINUE
505	001614	004737	001760			JSR PC,STATER	:YES,GO HANDLE PARITY ERROR
506	001620	000764				BR DRY1	:REISSUE FUNCTION
507	001622	032777	000004	177360		BIT #BIT2,DRXDB	:INIT DONE SHOULD NOT BE SET
508	001630	001402				BEQ 2\$	
509	001632	104400	014521			TYPE ,MINIT2	:INIT DONE WAS SET PRINT ERROR
510	001636	105777	177346		2\$:	TSTB DRXDB	:IS UNIT 1 DRY BIT SET
511	001642	100003				BPL 3\$	:IF NOT SKIP NEXT INSTRUCTION
512	001644	052737	100000	010464		BIS #BIT15,UNITSEL	:YES,SET UNIT 1 READY BIT
513	001652	000207			3\$:	RTS PC	
514							
515	001654	004737	001572		NOSEL:	JSR PC,DRY1	:TEST UNIT 1 FOR DRY SET
516	001660	013737	001212	011134	OPCOND:	MOV DTESTP,SEQUEN	:PUT INITIAL CONDITIONS IN SEQUENCE
517	001666	042737	177770	011134		BIC #177770,SEQUEN	:CLEAR ALL BUT SEQUENCE BITS
518	001674	013737	001212	002052		MOV DTESTP,TEST	:PUT INITIAL CONDITIONS IN TEST
519	001702	042737	177707	002052		BIC #177707,TEST	:CLEAR ALL BUT TEST BITS
520	001710	006037	002052			ROR TEST	:RIGHT JUSTIFY TEST BITS
521	001714	006037	002052			ROR TEST	
522	001720	006037	002052			ROR TEST	
523	001724	013737	001212	010200		MOV DTESTP,PAT	:PUT INITIAL CONDITIONS IN PATTERN
524	001732	006137	010200			ROL PAT	:RIGHT JUSTIFY PATTERN BITS
525	001736	006137	010200			ROL PAT	
526	001742	000337	010200			SWAB PAT	
527	001746	042737	177770	010200		BIC #177770,PAT	:CLEAR ALL BUT PATTERN BITS
528	001754	000137	002554			JMP XSTART	
529							
530	001760	005237	015564		STATER:	INC PARLOG	:INC PARITY ERROR COUNTER
531	001764	104400	014121			TYPE ,MRXCS	:PRINT THE RXCS CONTENTS
532	001770	017746	177212			DRXCS,-(SP)	:SAVE DRXCS FOR TYPEOUT
533	001774	104402			MOV		:GO TYPE--OCTAL ASCII
534	001776	003			TYPOS	3	:TYPE 3 DIGIT(S)
535	001777	000			.BYTE	0	:SUPPRESS LEADING ZEROS
536	002000	005337	015554		.BYTE	DEC P2RETRY	:HAVE 10 ERRORS OCCURED
537	002004	001405				BEQ 1\$	:YES,LOG HARD ERROR
538	002006	104400	014751			TYPE ,MPAR	:NO,TYPE PARITY ERROR
539	002012	104400	014004			TYPE ,NCRLF	
540	002016	000207				RTS PC	:RETURN



541	002020	005237	015566	1S:	INC HPARLOG	;INC HARD PARITY ERROR LOG
542	002024	104400	014707		TYPE ,MUNREC	;PRINT UNRECOVERABLE PARITY ERROR
543	002030	104400	014751		TYPE ,MPAR	
544	002034	104400	014004		TYPE ,MCRLF	
545	002040	104400	014634		TYPE ,MHALT4	
546	002044	000000		HLT4:	HALT	;HALT BECAUSE OF HARD PARITY ERROR
547	002046	000137	001220		JMP SA200	;IF CONTINUE IS PRESSED GO TO START
548	002052	000000		TEST:	0	
549						
550	002054	005002		TSTLIMITS:	CLR R2	;MESSAGE FLAG
551	002056	005737	001200		TST 00	;TEST FOR NO SELECTION OF TRACKS
552	002062	001434			BEQ 1S	;IT WILL BE STANDARD LIMITS
553	002064	105737	001200		TSTB 00	;TEST FOR 0 00 TRACK
554	002070	001425			BEQ 2S	;YES,CHECK FOR STANDARD ID TRACK
555	002072	004737	002266	5S:	JSR PC,TSTMSG	;HAS INITIAL MESSAGE BEEN TYPED
556	002076	104400	014442		TYPE ,MOD	;TYPE BOTH ID AND 00 LIMITS
557	002102	113737	001200	002114	MOVB 00,7S	
558	002110	004537	013374		JSR RS,SGLDEC	;TYPE SINGLE DECIMAL WORD
559	002114	000000		7S:	OPEN	
560	002116	104400	014001		TYPE ,DBLSP	
561	002122	104400	014446		TYPE ,MID	
562	002126	113737	001201	002140	MOVB 10,10S	
563	002134	004537	013374		JSR RS,SGLDEC	
564	002140	000000		10S:	OPEN	
565	002142	000404			BR 1S	
566	002144	123727	001201	000114	CMPB ID,#114	;TEST FOR MAXIMUM ID LIMIT
567	002152	001347		2S:	BNE 5S	
568	002154	005737	001202	1S:	TST FIRST	;TEST FOR NO SELECTION OF SECTORS
569	002160	001441			BEQ 13S	;IT WILL BE STANDARD LIMITS
570	002162	123727	001202	000001	CMPB FIRST,#1	;TEST FOR NORMAL FIRST SECTOR LIMIT
571	002170	001427			BEQ 4S	;YES,CHECK LAST SECTOR LIMIT
572	002172	004737	002266	6S:	JSR PC,TSTMSG	
573	002176	104400	014001		TYPE ,DBLSP	
574	002202	104400	014452		TYPE ,MFIRST	;TYPE BOTH FIRST AND LAST LIMITS
575	002206	113737	001202	002220	MOVB FIRST,11S	
576	002214	004537	013374		JSR RS,SGLDEC	
577	002220	000000		11S:	OPEN	
578	002222	104400	014001		TYPE ,DBLSP	
579	002226	104400	014461		TYPE ,MLAST	
580	002232	113737	001203	002244	MOVB LAST,12S	
581	002240	004537	013374		JSR RS,SGLDEC	
582	002244	000000		12S:	OPEN	
583	002246	000404			BR 3S	
584	002250	123727	001203	000032	CMPB LAST,#32	;TEST FOR NORMAL LAST SECTOR LIMIT
585	002256	001345		4S:	BNE 6S	
586	002260	104400	014172	3S:	TYPE ,DBLLF	
587	002264	000207		13S:	RTS PC	
588						
589						
590	002266	005702		TSTMSG:	TST R2	;TEST MESSAGE FLAG
591	002270	001005			BNE 1S	
592	002272	104400	014400		TYPE ,MNONSTD	;MESSAGE HEADING HAD NOT BEEN TYPED
593	002276	104400	014004		TYPE ,MCRLF	
594	002302	005202			INC R2	;SET THE MESSAGE FLAG

```

595 002304 000207 1S: RTS PC
596
597 002306 104400 014004 ICOND: TYPE ,MCRLF
598 002312 104400 014351 TYPE ,MICON ;TYPE INITIAL CONDITIONS
599 002316 105737 010464 TSTB UNITSEL ;TEST FOR DRIVE SELECTION
600 002322 100002 BPL 1S
601 002324 104400 014040 TYPE ,MUNIT0 ;TYPE UNIT 0 SELECTED
602 002330 005737 010464 1S: TST UNITSEL
603 002334 100002 BPL 2S
604 002336 104400 014050 TYPE ,MUNIT1 ;TYPE UNIT 1 SELECTED
605 002342 104400 013772 2S: TYPE ,TAB
606 002346 004737 007572 JSR PC,TYPSEL ;TYPE TEST SELECTIONS
607 002352 104400 014004 TYPE ,MCRLF
608 002356 005737 010464 TST UNITSEL ;WERE ANY DRIVES READY
609 002362 001005 BNE CONT12 ;YES,CONTINUE
610 002364 104400 013750 TYPE ,MNO DRY ;NO,TYPE NO DRIVES MESSAGE
611 002370 000000 HLT5: HALT ;NO DRIVES CAN'T CONTINUE
612 002372 000137 001220 JMP SA200 ;IF CONTINUE IS PRESSED RESTART
613 002376 004737 002054 CONT12: JSR PC,TSTLIMITS ;TYPE NONSTANDARD TRK AND SEC LIMITS
614 002402 000207 RTS PC
615
616 002404 032777 000400 176602 HOME: BIT #5WB,#5WR ;TEST THE NO RECAL SWITCH
617 002412 001046 BNE RTN ;RETURN IF THE SWITCH IS SET
618 002414 012737 000012 015554 MOV #10.,P2RETRY ;SET UP THE PARITY RETRY COUNTER
619 002422 012737 000012 015550 MOV #10.,DDRETRY ;USE THE DD RETRY COUNTER FOR RECAL RETRIES
620 002430 012777 040001 176550 2S: MOV #RECAL,#RXCS ;ISSUE RECAL FUNCTION
621 002436 004737 006714 JSR PC,DONECK ;WAIT FOR DONE FLAG
622 002442 032777 000002 176540 BIT #2,#RXDB ;WAS THERE A PARITY ERROR
623 002450 001403 BEQ 1S ;NO,CHECK FOR ANY OTHER ERROR
624 002452 004737 001760 JSR PC,STATER ;YES,GO REPORT IT
625 002456 000764 BR 2S ;RETRY RECAL
626 002460 005777 176522 1S: TST #RXCS ;IS THE ERROR FLAG SET
627 002464 100016 BPL XHOME ;IF NOT RETURN
628 002466 004737 007760 JSR PC,RDCODE ;YES,PRINT STATUS REGISTERS
629 002472 123727 007652 000040 CMPB BSTAT,#40 ;IS THE B CODE LESS THAN CODE 40
630 ;(RECAL CODES ARE 10,20,AND 30)
631 002500 002010 BGE XHOME ;IF NOT RETURN
632 002502 005337 015550 DEC DDRETRY ;HAVE 10 ERRORS OCCURED
633 002506 001350 BNE 2S ;NO,RETRY
634 002510 104400 014645 TYPE ,MHALT3 ;YES,CAN'T CONTINUE AS RX11 WILL NOT RECAL
635 002514 000000 HLT3: HALT
636 002516 000137 001220 JMP SA200 ;IF CONT. SWITCH IS PRESSED GO TO START
637 002522 012737 000001 011126 XHOME: MOV #1,PRESTRK ;RESET THE PRESENT TRACK TO TRACK 1
638 002530 000207 RTN: RTS PC ;RETURN
639

```

\*\*\*\*\*

```

;DECODE TEST BITS OF INITIAL CONDITIONS
;BITS 3,4,AND 5 OF THE INITIAL SWR SELECTED THE TEST WHICH
;IS TO BE PERFORMED. THEY ARE AS FOLLOWS:

```

```

;
;   BITS          TESTS
; 5 4 3

```

```

640
641
642
643
644
645
646
647
648

```



649				:	0	0	0
650				:	0	0	0
651				:	0	0	0
652				:	0	0	0
653				:	0	0	0
654				:	0	0	0
655				:	0	0	0
656				:	0	0	0
657				:	0	0	0
658				:	0	0	0
659				:	0	0	0
660				:	0	0	0

NO TEST SELECTED (DEFAULT TO TEST 7)  
WRITE ONLY  
WRITE FOLLOWED BY READ  
WRITE FOLLOWED BY READ AND VERIFY  
READ AND VERIFY ONLY  
READ ONLY (CRC CHECK)  
READ AND VERIFY DELETED DATA  
WRITE FOLLOWED BY READ FOLLOWED BY ANOTHER  
READ AND VERIFY

\*\*\*\*\*

661	002532	012706	001200	RESTART:	MOV #STACK, SP		:RESET THE STACK POINTER
662	002536	012746	000340		MOV #PR7, -(SP)		:AND INTERRUPT LEVEL
663	002542	012746	002550		MOV #18, -(SP)		
664	002546	000002			RTI		
665	002550	005237	017600	IS:	INC RESTCNTR		:INC RESTART COUNTER
666	002554	004737	002306	XSTART:	JSR PC, ICOND		:TYPE OUT INITIAL CONDITIONS
667	002560	005037	010706	TESTSEL:	CLR CHARCT		:CLEAR EOP CHARACTER COUNTER
668	002564	012737	000012		MOV #10., RDRETRY	015544	:SET UP READ AND WRITE RETRY COUNTERS
669	002572	012737	000012		MOV #10., WTRTRY	015546	
670	002600	005037	003746		CLR ERWRT		:CLEAR WRITE CAUSED BY ERROR FLAG
671	002604	005037	003744		CLR RDAFTWT		:CLEAR READ AFTER WRITE FLAG
672	002610	142737	000377		BICB #377, @BRONTEST	002646	:CLEAR OUT BRANCH OFFSET
673	002616	005737	002052		TST TEST		:IF NO TEST SPECIFIED FORCE TEST 7
674	002622	001003			BNE IS		
675	002624	012737	000007		MOV #7, TEST	002052	
676	002632	013704	002052	IS:	MOV TEST, R4		
677	002636	005304			DEC R4		:ADJUST TEST BITS FOR CORRECT
678	002640	006304			ASL R4		:BRANCH OFFSET
679	002642	150437	002646		BISB R4, @BRONTEST		:INSERT OFFSET TO BRANCH INST.
680	002646	000777		BRONTEST:	BR		:BRANCH BY TEST OFFSET
681	002650	000137	002704		JMP WRONLY		:WRITE ONLY FUNCTION
682	002654	000137	003752		JMP WRTRD		:WRITE THEN READ FUNCTION
683	002660	000137	005260		JMP WTRDCK		:WRITE THEN READ VERIFY
684	002664	000137	006236		JMP RDCHK		:READ VERIFY
685	002670	000137	006276		JMP RDNLY		:READ ONLY FUNCTION
686	002674	000137	006336		JMP DRVSNP		:WRITE, AND READ VERIFY ON ALTERNATING DRIVES
687	002700	000137	006422		JMP TEST7		:WRITE, READ, FOLLOWED BY READ VERIFY

\*\*\*\*\*

:THIS IS A WRITE ONLY FUNCTION USING DATA PATTERN AND  
:TRACK SEQUENCE SPECIFIED BY INITIAL SWR SETTINGS  
:T = 1

697	002704	004737	010072	WRONLY:	JSR PC, GETPATTERN		:SET UP SOFTWARE DATA BUFFER
698	002710	004737	010710	XWRONLY:	JSR PC, INITTRACKS		:SET UP ID, OD, AND TRACK COUNTER
699	002714	004737	010374		JSR PC, GETUNIT		:SET UP DRIVE UNIT SELECTION
700	002720	004737	011020	IS:	JSR PC, GETTRACK		:PICK UP NEXT TRACK
701	002724	004737	002744		JSR PC, WRITE		:DO THE WRITE FUNCTION
702	002730	005337	011122		DEC TRKCNTR		:TEST TRACK COUNTER

703	002734	001371			BNE IS	
704	002736	004737	010512		JSR PC, STOP	;CHECK FOR LAST DRIVE AND EOP
705	002742	000762			BR XWRONLY	;NEXT PASS
706						
707	002744	004737	011762		WRITE: JSR PC, INITSECTOR	;SET UP FIRST, LAST AND SECTOR COUNTER
708	002750	004737	012062		XWRITE: JSR PC, GETSECTOR	;PICK UP NEXT SECTOR
709	002754	012737	000012	015562	MOV #10., SRETRY	
710	002762	012737	000012	015556	ONEWRT: MOV #10., PRETRY	;SET RETRY COUNTER
711	002770	004737	005460		FILLBUF: JSR PC, ADJSUM	;ADJUST DATA BUFFER AND CHECK SUM FOR ADDRESSES
712	002774	012746	003220		MOV #FILLDONE, -(SP)	;PUT GOOD RETURN ON STACK
713	003000	012746	003046		MOV #FILLER, -(SP)	;PUT ERROR RETURN ON STACK
714	003004	005037	006226		CLR BYTECNTR	
715	003010	012746	000200		MOV #PR4, -(SP)	
716	003014	012746	003022		MOV #IS, -(SP)	
717	003020	000002			RTI	
718	003022	012777	000101	176156	IS: MOV #FBIE, JRXCS	;EXICUTE FILLBUFFER COMMAND
719	003030	004737	006606		FILLFLAG: JSR PC, TRCK	;TEST FOR TRANSFER REQUEST FLAG
720	003034	112077	176150		XFRBYTE: MOVB (RD)+, JRXDB	;TRANSFER DATA BYTE
721	003040	005237	006226		INC BYTECNTR	
722	003044	000771			BR FILLFLAG	;WAIT FOR NEXT TR FLAG
723						
724	003046	005726			FILLER: TST (SP)+	;REMOVE THE DONE RETURN FROM THE STACK
725	003050	012737	014320	003114	MOV #FIL, PTP1+2	;PUT ADDR OF FILLBUF MESSAGE IN PAR ERR TYP0UT 1
726	003056	012737	014320	003202	MOV #FIL, PTP2+2	;PUT ADDR OF FILLBUF MESSAGE IN PAR ERR TYP0UT 2
727	003064	012737	002770	003162	MOV #FILLBUF, PCONT+2	;IF NOT HARD ERR RETURN THROUGH PCONT TO FILLBUF
728	003072	000137	003076		JMP PARTEST	;PRINT OUT PAR ERR AND TEST CONDITIONS FOR RETRY
729						
730						
731	003076	005237	015564		PARTEST: INC PARLOG	;INCREMENT PARITY ERROR
732	003102	032777	020000	176104	BIT #SW13, JSWR	;TEST DON'T PRINT ERROR SWITCH
733	003110	001006			BNE CONT4	
734	003112	104400	000000		PTYP1: TYPE , OPEN	;PRINT THE PARITY ERROR MESSAGE
735	003116	104400	014751		TYPE ,MPAR	
736	003122	104400	014004		TYPE ,MCRLF	
737	003126	005777	176062		CONT4: TST JSWR	;TEST HALT ON ERROR SWITCH
738	003132	100001			BPL CONT13	
739	003134	000000			HLT6: HALT	;HALT ON ERROR
740	003136	032777	004000	176050	CONT13: BIT #SW11, JSWR	;TEST NO RETRY SWITCH
741	003144	001007			BNE CONT5	;IF SET LOG HARD ERROR
742	003146	005337	015556		DEC PRETRY	;DECREMENT RETRY COUNTER
743	003152	005737	015556		TST PRETRY	;HAVE 10 ERRORS OCCURED
744	003156	001402			BEQ CONT5	;IF CLEARED LOG HARD ERROR
745	003160	000137	002770		PCONT: JMP FILLBUF	;RETURN TO RETRY TEST THROUGH HERE
746	003164	005237	015566		CONT5: INC HPARLOG	;INC.HARD PARITY ERROR COUNTER
747	003170	104400	014172		TYPE ,DELLF	
748	003174	104400	014707		TYPE ,MUNREC	;TYPE HARD PARITY ERROR
749	003200	104400	000000		PTYP2: TYPE , OPEN	
750	003204	104400	014751		TYPE ,MPAR	
751	003210	104400	014004		TYPE ,MCRLF	
752	003214	000137	007654		JMP DELUNIT	;GO DELETE THE UNIT,CAN NOT CONTINUE
753						
754						
755						
756						

; SWITCH 9 OF INITIAL TEST CONDITIONS SWITCH SETTINGS IS THE DELETED DATA  
 ; FUNCTION INDICATOR. WHEN THIS BIT IS SET ALL WRITE / READ FUNCTIONS WILL  
 ; SET AND CHECK THE DELETED DATA BIT ON THE DISKETTE.



```

757 ;*****
758
759 003220 012737 000012 015556 FILLDONE: MOV #10,PRETRY ;SET UP RETRY COUNTER
760 003226 012746 003310 REWRITE: MOV #WRTDONE,-(SP) ;SET GOOD RETURN ON STACK
761 003232 012746 003336 MOV #WRITER,-(SP) ;SET ERROR RETURN ON STACK
762 003236 112737 000105 003742 MOV# #WRTIE,FUNCTION ;SET FUNCTION WORD TO WRITE
763 003244 032737 001000 001212 BIT #BIT9,DTESTP ;TEST FOR WRITE DELETED DATA
764 003252 001403 BEQ IS
765 003254 112737 000115 003742 MOV# #WTDIE,FUNCTION
766 003262 062737 000001 017570 IS: ADD #1,WTCNTR ;INC TOTAL WRITE FUNCTIONS COUNTER
767 003270 005537 017572 ADC WTCNTR+2 ;DOUBLE PRECISION COUNTER
768 003274 004737 003644 JSR PC,COMMORD ;TRANSFER COMMAND TO DRIVE
769 003300 004737 006714 JSR PC,DONECK ;TEST FOR DONE FLAG
770 003304 000137 006776 JMP NOINTER ;NO INTERRUPT ERROR
771
772 003310 005737 003746 WRTDONE: TST ERWRT ;IS THIS A REWRITE FROM A DATA ERROR
773 003314 001004 BNE IS ;YES GO REREAD THIS SECTOR
774 003316 005337 012052 DEC SECCNTR ;NO TEST SECTOR COUNTER
775 003322 001003 BNE 2$ ;NOT LAST SECTOR GO TO NEXT ONE
776 003324 000207 RTS PC
777 003326 000137 004070 IS: JMP REREAD ;REREAD THE SECTOR
778 003332 000137 002750 2$: JMP XWRITE
779
780 003336 005726 WRITER: TST (SP)+ ;REMOVE THE DONE RETURN FROM THE STACK
781 003340 032737 000002 007650 BIT #BIT1,ASTAT ;IS THIS A PARITY ERROR
782 003346 001413 BEQ WRTSEK ;NO, IT MUST BE A SEEK ERROR
783 ;PARITY ERROR DURING A WRITE FUNCTION
784 003350 012737 014742 003114 MOV #MWRITE,PTYP1+2 ;PUT ADDR OF WRITE MESSAGE IN PAR ER TYP0UT 1
785 003356 012737 014742 003202 MOV #MWRITE,PTYP2+2 ;PUT ADDR OF WRITE MESSAGE IN PAR ER TYP0UT 2
786 003364 012737 003226 003162 MOV #REWRITE,PCONT+2 ;IF NOT HARD ERR RETURN THROUGH PCONT TO REWRITE
787 003372 000137 003076 JMP PARTEST ;GO INC LOG AND TEST FOR RETRY
788
789 ;SEEK ERROR DURING A WRITE FUNCTION
790 003376 012737 002762 003516 WRTSEK: MOV #ONEWRT,SEKRTY+2 ;SETUP FOR WRT RETRY ON SEEK ERROR
791 ;(AFTER A RECAL. THE CONTENTS OF SECTOR 1,
792 ;TRACK 1 ARE LOADED INTO THE SECTOR BUFFER.
793 ;TO REWRITE THE CORRECT DATA THE PROGRAM
794 ;MUST REFILL THE SECTOR BUFFER.
795 003404 012737 014742 003454 MOV #MWRITE,STYP1+2 ;PUT ADDR OF WRITE MESSAGE IN SEEK ER TYPEOUT 1
796 003412 012737 014742 003544 MOV #MWRITE,STYP2+2 ;PUT ADDR OF WRITE MESSAGE IN SEEK ER TYPEOUT 2
797 003420 004737 003424 JSR PC,SEEKER ;RECORD SEEK ERROR
798
799 003424 012703 015576 SEEKER: MOV #ZSEKLOG,R3 ;SETUP INC INSTRUCTION FOR UNIT 0 LOG
800 003430 004737 017604 JSR PC,U1LOG ;TEST FOR UNIT 1 LOG
801 003434 005213 INC (R3) ;INCREMENT SEEK ERROR LOG
802 003436 004737 003612 JSR PC,TRKERR ;INC ERROR PER TRACK COUNTERS
803 003442 032777 020000 175544 BIT #SW13,ASWR ;CHECK DON'T PRINT ERROR SWITCH
804 003450 001004 BNE SWHLT1
805 003452 104400 014742 STYP1: TYPE #WRITE ;PRINT WRITE (READ) SEEK ERROR
806 003456 004737 003560 JSR PC,SEKTYP
807 003462 005777 175526 SWHLT1: TST ASWR ;TEST THE HALT ON ERROR SWITCH
808 003466 100001 BPL CONT14
809 003470 000000 HLT7: HALT ;HALT ON THE ERROR
810 003472 032777 004000 175514 CONT14: BIT #SW11,ASWR ;CHECK THE NO RETRY SWITCH

```



811	003500	001007			BNE HARDSK		: IF SET LOG HARD SEEK ERROR
812	003502	005337	015562		DEC SRETRY		: HAVE 10 ERRORS BEEN LOGED
813	003506	001404			BEQ HARDSK		: YES LOG HARD ERROR
814	003510	004737	002404		JSR PC, HOME		: RECALIBRATE DRIVES ON SEEK ERROR
815	003514	000137	002762		JMP ONEWRT	SEKRTY:	: NO RETRY WRITE COMMAND (READ COMAND)
816	003520	012703	015636		MOV #ZHSEKLOG, R3	HARDSK:	: SET INC. INST. FOR UNIT 0 ERR LOG
817	003524	004737	017604		JSR PC, ULOG		: TEST FOR UNIT 1 ERROR LOG
818	003530	005213			INC (R3)		: HARD SEEK ERROR
819	003532	104400	014172		TYPE ,DBLLF		
820	003536	104400	014707		TYPE ,MUNREC		: TYPE UNRECOVERABLE SEEK ERROR
821	003542	104400	014742		TYPE ,MWRITE	STYP2:	: ON WRITE (READ) COMMAND
822	003546	004737	003560		JSR PC, SEKTYP		
823	003552	062706	000002		ADD #2, SP		: REMOVE SEEK ERROR FROM STACK POINTER
824	003556	000207			RTS PC		: RETURN TO NEXT SECTOR BY DONE RETURN ON STACK
825							
826	003560	104400	014727		TYPE ,MSEEK	SEKTYP:	: TYPE SEEK ERROR
827	003564	104400	013521		TYPE ,MPRES		: TYPE ADDRESS OF TRACK MOVED FROM
828	003570	013737	011126	003602	MOV PRESTRK, 1S		
829	003576	004537	013374		JSR R5, SGLDEC		
830	003602	000000			OPEN	1S:	
831	003604	104400	014004		TYPE ,MCRLF		
832	003610	000207			RTS PC		
833							
834	003612	013705	011124		MOV TARGET, R5	TRKERR:	: SET UP TO INC ERROR PER TRACK COUNTER
835	003616	006305			ASL R5		: ADJUST FOR EVEN ADDRESS
836	003620	062705	017104		ADD #UDTRK, R5		: ADDIN ADDRESS OF UNIT 0 LOG
837	003624	032737	000020	010464	BIT #BIT4, UNITSEL		: CHECK THE UNIT SELECTION BIT
838	003632	001402			BEQ 1S		: IF CLEARED UNIT 0 IS ACTIVE
839	003634	062705	017336		ADD #UITRK, R5		: ADJUST FOR UNIT 1 LOG
840	003640	005215			INC (R5)	1S:	: INC THE CORRECT COUNTER
841	003642	000207			RTS PC		
842							
843	003644	013705	011124		MOV TARGET, R5	COMMAND:	: GET TRACK NUMBER
844	003650	006305			ASL R5		: MULTIPLY BY 4 TO DOUBLE PRECISION
845	003652	006305			ASL R5		: INTERLEAVE COUNTER LOCATIONS
846	003654	062705	015734		ADD #TKACC, R5		: ADD ON ADDRESS OF TRACK ACCESS COUNTER
847	003660	062715	000001		ADD #1, (R5)		: INCREMENT THE COUNTER
848	003664	005565	000002		ADC 2(R5)		: ADD CARRY TO HIGH ORDER WORD
849	003670	153737	010464	003742	BISB UNITSEL, FUNCTION		: SET UNIT SELECTION BIT IN COMMAND WORD
850	003676	013777	003742	175302	MOV FUNCTION, ARXCS		: SEND OUT COMMAND TO DRIVE
851	003704	004737	006606		JSR PC, TRCK		: WAIT FOR TR FLAG
852	003710	113777	012054	175272	MOVB TSECTOR, ARXDB		: SEND OUT TARGET SECTOR
853	003716	004737	006606		JSR PC, TRCK		: WAIT FOR TR FLAG
854	003722	113777	011124	175260	MOVB TARGET, ARXDB		: SEND OUT TARGET TEACK
855	003730	005046			CLR -(SP)		: LOWER INTERRUPT LEVEL TO ALLOW AN INTERRUPT
856	003732	012746	003740		MOV #1S, -(SP)		
857	003736	000002			RTI		
858	003740	000207			RTS PC	1S:	
859							
860	003742	000000			FUNCTION:		
861	003744	000000			RDAFTWT:		: READ AFTER WRITE FUNCTION FLAG
862	003746	000000			ERWRT:		: WRITE CAUSED BY DATA ERROR FLAG
863	003750	000000			DATAACK:		: DATA CHECK ON CRC ERROR FLAG
864							



```

865 ;*****
866
867 ;THIS IS A WRITE ALL SECTORS FOLLOWED BY A READ ALL SECTORS
868 ;WITH DATA PATTERNS AND HEAD SEQUENCING SET BY INITIAL SWR
869 ; T = 2
870
871 003752 005137 003744 WRTRD: COM RDAFTWT ;SET READ AFTER WRITE FLAG
872 003756 004737 010072 JSR PC,GETPATTERN
873 003762 004737 010710 XWRTRD: JSR PC,INITTRACKS
874 003766 004737 010374 JSR PC,GETUNIT
875 003772 004737 011020 IS: JSR PC,GETTRACK
876 003776 004737 002744 JSR PC,WRITE
877 004002 004737 004022 JSR PC,READ
878 004006 005337 011122 DEC TRKCNTR
879 004012 001367 BNE IS
880 004014 004737 010512 JSR PC,STOP
881 004020 000760 BR XWRTRD
882
883 ;*****
884
885 ;READ DATA FROM THE DISKETTE
886
887
888 004022 004737 011762 READ: JSR PC,INITSECTOR
889 004026 004737 012062 XREAD: JSR PC,GETSECTOR
890 004032 012737 000012 015550 MOV #10.,DDRETRY ;SET UP RETRY COUNTERS FOR DELETED DATA
891 004040 012737 000012 015552 MOV #10.,DATARETRY ;DATA ERROR
892 004046 012737 000012 015556 MOV #10.,PRETRY ;PARITY
893 004054 012737 000012 015562 MOV #10.,SRETRY ;SEEK
894 004062 012737 000012 015560 MOV #10.,CRETRY ;AND CRC ERRORS
895 004070 005037 003750 REREAD: CLR DATAACK ;CLEAR CRC DATA CHECK FLAG
896 004074 012746 004140 MOV #RDDONE,-(SP) ;SET GOOD RETURN ON STACK
897 004100 012746 004302 MOV #RDERR,-(SP) ;SET READ ERROR RETURN ON STACK
898 004104 112737 000107 003742 MOV# #RDIE,FUNCTION
899 004112 062737 000001 017574 ADD #1,RDCNTR ;INC TOTAL READ FUNCTIONS COUNTER
900 004120 005537 017576 ADC RDCNTR+2 ;DOUBLE PRECISION COUNTER
901 004124 004737 003644 JSR PC,COMMAND
902 004130 004737 006714 JSR PC,DONECK ;TEST FOR DONE
903 004134 000137 006776 JMP NOINTER ;NO INTERRUPT ON DONE
904
905
906
907 004140 022737 000012 015544 RDDONE: CMP #10.,RDRETRY ;IS READ RETRY EQUAL TO 10
908 004146 001420 BEQ CONT1 ;YES,NO ERRORS OCCURED
909 004150 012703 015612 MOV #ZRDLOG,R3 ;SET INC. INST. FOR UNIT 0 ERROR LOG
910 004154 004737 017604 JSR PC,U1LOG ;TEST FOR UNIT 1 ERROR LOG
911 004160 005213 INC (R3) ;INC RECOVERABLE READ LOG
912 004162 012737 000012 015544 MOV #10.,RDRETRY ;RESET READ RETRY COUNTER
913 004170 104400 014172 TYPE ,DBLLF
914 004174 104400 014561 TYPE ,MREC ;TYPE RECOVERABLE READ ERROR
915 004200 104400 014626 TYPE ,MREAD
916 004204 104400 014004 TYPE ,MCRLF
917 004210 022737 000012 015546 CONT1: CMP #10.,WTRTRY ;IS WRITE RETRY EQUAL TO 10
918 004216 001420 BEQ CONT2 ;YES,NO ERRORS OCCURED

```



919	004220	012703	015616			MOV #ZWRLOG,R3	;SET INC.INST.FOR UNIT 0 ERROR LOG
920	004224	004737	017604			JSR PC,U1LOG	;TEST FOR UNIT 1 LOG
921	004230	005213				INC (R3)	;INC RECOVERABLE WRITE LOG
922	004232	012737	000012	015546		MOV #10,WRETRY	;RESET THE WRITE RETRY COUNTER
923	004240	104400	014172			TYPE ,DBLLF	
924	004244	104400	014561			TYPE ,MREC	;TYPE RECOVERABLE WRITE ERROR
925	004250	104400	014742			TYPE ,MWRITE	
926	004254	104400	014004			TYPE ,MCRLF	
927	004260	004737	005014		CONT2:	JSR PC,DDCHK	;CHECK FOR DELETED DATA INDICATOR
928	004264	005701				TST R1	;BIT 15 OF R1 IS READ 1 SECTOR FLAG
929	004266	100001				BPL NEXTRD	
930	004270	000207				RTS PC	;IF SET,GO VERIFY DATA JUST READ
931	004272	005337	012052		NEXTRD:	DEC SECCNTR	
932	004276	001253				BNE XREAD	
933	004300	000207				RTS PC	;READ FUNCTION IS DONE
934							
935	004302	005726			RDERR:	TST (SP)+	;REMOVE THE DONE RETURN FROM THE STACK
936	004304	032737	000002	007650		BIT #BIT1,ASTAT	;IS THIS A PARITY ERROR
937	004312	001413				BEG 1\$	;NO, SEE IF ITS A CRC ERROR
938							;PARITY ERROR DURING A READ FUNCTION
939	004314	012737	014626	003114		MOV #MREAD,PTYP1+2	;PUT ADDR OF READ MESSAGE IN PAR ERR TYPEOUT 1
940	004322	012737	014626	003202		MOV #MREAD,PTYP2+2	;PUT ADDR OF READ MESSAGE IN PAR ER TYPEOUT 2
941	004330	012737	004070	003162		MOV #REREAD,PCONT+2	;IF HARD ERR RETURN THROUGH PCONT TO REREAD
942	004336	000137	003076			JMP PARTEST	;RECORD PARITY ERROR AND RETRY FUNCTION
943	004342	032737	000001	007650	1\$:	BIT #BIT0,ASTAT	;IS THIS A CRC ERROR
944	004350	001014				BNE CRCER	;YES GO TEST AND LOG IT
945							;SEEK ERROR DURING A READ FUNCTION
946	004352	012737	004070	003516		MOV #REREAD,SEKRTY+2	;SET SEEK CONTINUE FOR READ RETRY
947	004360	012737	014626	003454		MOV #MREAD,STYP1+2	;SET ADDR OF READ MESSAGE IN SEEK ER TYPEOUT 1
948	004366	012737	014626	003544		MOV #MREAD,STYP2+2	;SET ADDR OF READ MESSAGE IN SEEK ER TYPEOUT 2
949	004374	004737	003424			JSR PC,SEEKER	;RECORD SEEK ERROR
950	004400	000734				BR NEXTRD	;GO TO NEXT SECTOR,CAN'T READ THIS ONE
951							;CRC ERROR DETECTED WHILE READING
952	004402	004737	003612		CRCER:	JSR PC,TRKERR	;INC ERROR PER TRACK COUNTER
953	004406	005701				TST R1	;IF READ ONLY, REPORT DATA CRC ERROR
954	004410	100057				BPL DATACRC	
955	004412	005237	003750			INC DATAK	;SET DATA CHECK FLAG
956	004416	004737	005340			JSR PC,EMPBUFF	;CHECK FOR A DATA ERROR
957	004422	005737	006234			TST ERCNTR	;WAS THERE A DATA ERROR
958	004426	001050				BNE DATACRC	;YES,REREAD AND/OR REWRITE THE DATA
959	004430	012703	015606			MOV #ZCRCBAD,R3	;SET INC.INST.FOR UNIT 0 ERROR LOG
960	004434	004737	017604			JSR PC,U1LOG	;TEST FOR UNIT 1 ERROR LOGS
961	004440	005213				INC (R3)	;NO,INC BAD CRC GENERATOR ERROR
962	004442	032777	020000	174544		BIT #SW13,2SWR	;TEST DON'T SWITCH
963	004450	001004				BNE 2\$	
964	004452	104400	014576			TYPE ,MBADCRC	;TYPE CRC GENERATOR ERROR
965	004456	104400	014004			TYPE ,MCRLF	
966	004462	005777	174526		2\$:	TST 2SWR	;TEST HALT ON ERROR SWITCH
967	004466	100001				BPL CONT15	
968	004470	000000			HLT10:	HALT	;HALT ON ERROR
969	004472	032777	004000	174514	CONT15:	BIT #SW11,2SWR	;CHECK NO RETRY SWITCH
970	004500	001005				BNE 3\$	;IF SET LOG HARD ERROR
971	004502	005337	015560			DEC CRETRY	;HAVE 10 ERRORS BEEN LOGED
972	004506	001402				BEG 3\$	;YES,LOG HARD ERROR



973	004510	000137	004070		JMP REREAD	:NO GO REREAD DATA
974	004514	012703	015646	3S:	MOV #ZHCRCBAD,R3	:SET INC.INST.FOR UNIT 0 ERROR LOG
975	004520	004737	017604		JSR PC,U1LOG	:TEST FOR UNIT 1 ERROR LOGS
976	004524	005213			INC (R3)	
977	004526	104400	014172		TYPE ,DBLLF	
978	004532	104400	014576		TYPE ,MBADCRC	:TYPE HARD CRC GENERATOR ERROR
979	004536	104400	014004		TYPE ,MCRLF	
980	004542	104400	014656		TYPE ,MHALT11	:AND HALT AS THE CRC GENERATOR DOESN'T WORK
981	004546	000000		HLT11:	HALT	
982						
983	004550	012703	015602	DATA CRC:	:DATA CRC ERROR REREAD AND/OR REWRITE TO GET GOOD DATA	
984	004554	004737	017604		MOV #ZCRCLOG,R3	:SET INC.INST.FOR UNIT 0 ERROR LOG
985	004560	005213			JSR PC,U1LOG	:TEST FOR UNIT 1 ERROR LOGS
986	004562	032777	020000	174424	INC (R3)	:TRUE DATA CRC ERROR
987	004570	001004			BIT #SW13,JSWR	:TEST DON'T PRINT ERROR SWITCH
988	004572	104400	014670		BNE 4S	
989	004576	104400	014004		TYPE ,MCRC	:TYPE DATA CRC ERROR
990	004602	005777	174406	4S:	TYPE ,MCRLF	
991	004606	100001			TST JSWR	:TEST HALT ON ERROR SWITCH
992	004610	000000		HLT12:	BPL CONT16	
993	004612	032777	004000	174374	HALT	:HALT ON ERROR
994	004620	001005		CONT16:	BIT #SW11,JSWR	:TEST NO RETRY SWITCH
995	004622	005337	015544		BNE 5S	:IF SET LOG HARD ERROR
996	004626	001402			DEC RDRETRY	:HAVE 10 ERRORS OCCURED
997	004630	000137	004070		BEQ 5S	:YES LOG HARD ERROR OR REWRITE DATA
998	004634	012703	015642	5S:	JMP REREAD	:NO GO REREAD THIS SECTOR
999	004640	004737	017604		MOV #ZHCRCLOG,R3	:SET INC.INST.FOR UNIT 0 ERROR LOG
1000	004644	005213			JSR PC,U1LOG	:TEST FOR UNIT 1 ERROR LOGS
1001	004646	012737	000012	015544	INC (R3)	:INC HARD CRC ERROR LOG
1002	004654	005737	003744		MOV #10, RDRETRY	:RESET READ RETRY COUNTER
1003	004660	001021			TST RDAFTMT	:IS THIS A READ AFTER WRITE FUNCTION
1004	004662	012703	015652		BNE CONT6	:YES GO REWRITE IT
1005	004666	004737	017604		MOV #ZHRDLOG,R3	:NO SET INC.INST.FOR UNIT 0 ERROR LOG
1006	004672	005213			JSR PC,U1LOG	:TEST FOR UNIT 1 ERROR LOGS
1007	004674	104400	014172		INC (R3)	:READ ONLY, HARD READ ERROR
1008	004700	104400	014707		TYPE ,DBLLF	
1009	004704	104400	014626		TYPE ,MUNREC	:TYPE UNRECOVERABLE READ ERROR
1010	004710	104400	014004	CONT7:	TYPE ,MREAD	
1011	004714	062706	000002		TYPE ,MCRLF	
1012	004720	000137	004272		ADD #2,SP	:REMOVE READ DONE ADDRESS FROM STACK
1013	004724	032777	004000	174262	JMP NEXTRD	:READ NEXT SECTOR CAN'T READ THIS ONE
1014	004732	001011		CONT6:	BIT #SW11,JSWR	:TEST NO RETRY SWITCH
1015	004734	005337	015546		BNE 7S	:IF SET LOG HARD ERROR
1016	004740	001406			DEC WTRETRY	:HAVE 10 REWRITES BEEN DONE
1017	004742	005137	003746		BEQ 7S	:YES LOG HARD WRITE ERROR
1018	004746	062706	000002		COM ERWRT	:NO SET THE WRITE CAUSED BY ERROR FLAG
1019	004752	000137	002762		ADD #2,SP	:REMOVE RDDONE ADDRESS FROM STACK
1020	004756	012703	015656	7S:	JMP ONEWRT	:REWRITE SECTOR THEN REREAD TO CHECK FOR ERROR
1021	004762	004737	017604		MOV #ZHWRTLOG,R3	:SET INC.INST.FOR UNIT 0 ERROR LOG
1022	004766	005213			JSR PC,U1LOG	:TEST FOR UNIT 1 ERROR LOGS
1023	004770	012737	000012	015546	INC (R3)	:HARD WRITE ERROR
1024	004776	104400	014172		MOV #10, WTRETRY	:RESET WRITE RETRY COUNTER
1025	005002	104400	014707		TYPE ,DBLLF	
1026	005006	104400	014742		TYPE ,MUNREC	:TYPE UNRECOVERABLE WRITE ERROR
					TYPE ,MWRITE	



```

1027 005012 000736 BR CONT7
1028
1029
1030
1031 005014 032737 001000 001212 DDCHK: BIT #BIT9,DTESTP ;TEST BIT 9 AS TO DELETED DATA TRANSFER
1032 005022 001470 BEQ CONT10
1033 005024 132737 000100 007650 BITB #BIT6,ASTAT ;THIS IS A DELETED DATA FUNCTION
1034 005032 001111 BNE RETURN ;DD BIT SHOULD BE SET
1035 005034 012703 015626 MOV #ZDDMIS,R3 ;SET INC INST FOR UNIT 0 ERROR LOG
1036 005040 004737 017604 JSR PC,U1LOG ;TEST FOR UNIT 1 ERROR LOGS
1037 005044 005213 INC (R3) ;INC MISSING DELETED DATA LOG
1038 005046 004737 003612 JSR PC,TRKERR ;INC ERROR PER TRACK COUNTER
1039 005052 032777 020000 174134 BIT #SW13,DSWR ;TEST DON'T PRINT ERROR SWITCH
1040 005060 001011 BNE CONT11
1041 005062 104400 013440 TYPE ,MDDMIS ;TYPE MISSING DELETED DATA BIT
1042 005066 052737 000400 010464 DDERR: BIS #BIT8,UNITSEL ;SET HAD ERROR FLAG
1043 005074 004737 007476 JSR PC,TYPADR ;TYPE ADDRESS OF ERROR
1044 005100 104400 014004 TYPE ,MCRLF
1045 005104 005777 174104 CONT11: TST DSWR ;TEST HALT ON ERROR SWITCH
1046 005110 100001 BPL CONT17
1047 005112 000000 HALT ;HALT ON DELETED DATA ERROR
1048 005114 032777 004000 174072 HLT13: BIT #SW11,DSWR ;TEST NO RETRY SWITCH
1049 005122 001005 BNE 45 ;IF SET LOG HARD ERROR
1050 005124 005337 015550 DEC DDRETRY ;HAVE 10 ERRORS BEEN LOGED
1051 005130 001402 BEQ 45 ;YES LOG HARD ERROR
1052 005132 000137 004070 JMP REREAD ;NO, REREAD SECTOR
1053 005136 012703 015666 45: MOV #ZDDLOG,R3 ;SET INC INST FOR UNIT 0 ERROR LOG
1054 005142 004737 017604 JSR PC,U1LOG ;TEST FOR UNIT 1 ERROR LOGS
1055 005146 005213 INC (R3)
1056 005150 104400 014172 TYPE ,DBLLF
1057 005154 104400 014707 TYPE ,MUNREC ;TYPE UNRECOVERABLE DELETED DATA ERROR
1058 005160 104400 014547 TYPE ,MDDER
1059 005164 104400 014004 TYPE ,MCRLF
1060 005170 004737 007476 JSR PC,TYPADR
1061 005174 104400 014004 TYPE ,MCRLF
1062 005200 000137 004272 JMP NEXTSD ;READ NEXT SECTOR
1063 005204 032737 000100 007650 CONT10: BIT #BIT6,ASTAT ;THIS IS NOT A DELETED DATA TRANSFER
1064 005212 001421 BEQ RETURN
1065 005214 012703 015632 MOV #ZUNXDD,R3 ;SET INC INST FOR UNIT 0 ERROR LOG
1066 005220 004737 017604 JSR PC,U1LOG ;TEST FOR UNIT 1 ERROR LOGS
1067 005224 005213 INC (R3) ;UNEXPECTED DD BIT SET
1068 005226 052737 000400 010464 BIS #BIT8,UNITSEL ;SET HAD ERROR FLAG
1069 005234 004737 003612 JSR PC,TRKERR ;INC ERROR PER TRACK COUNTER
1070 005240 032777 020000 173746 BIT #SW13,DSWR ;TEST DON'T PRINT ERROR SWITCH
1071 005246 001316 BNE CONT11
1072 005250 104400 013414 TYPE ,MUNXDD ;TYPE UNEXPECTED DELETED DATA BIT
1073 005254 000704 BR DDERR
1074 005256 000207 RETURN: RTS PC
1075
1076
1077
1078 ;*****
1079 ;WRITE ALL SECTORS,READ AND VERIFY ALL SECTORS
1080

```



```

1081                                     ; T = 3
1082
1083 005260 005137 003744      WTRDCK:      COM RDAFTWT      ;SET READ AFTER WRITE FLAG
1084 005264 004737 010072      JSR PC,GETPATTERN
1085 005270 004737 010710      XWTRDCK:    JSR PC,INITTRACKS
1086 005274 004737 010374      JSR PC,GETUNIT
1087 005300 004737 011020      1S:        JSR PC,GETTRACK
1088 005304 004737 002744      JSR PC,WRITE
1089 005310 004737 005330      JSR PC,READCHK
1090 005314 005337 011122      DEC TRKCNTR
1091 005320 001367              BNE 1S
1092 005322 004737 010512      JSR PC,STOP
1093 005326 000760              BR XWTRDCK
1094
1095
1096 ;*****
1097
1098 ;READ A SECTOR,EMPTY THE SECTOR BUFFER AND VERIFY
1099 ;THE DATA READ AGAINST CORE DATA BUFFER
1100
1101 005330 052701 100000      READCHK:    BIS #BIT15,R1      ;SET READ ONE SECTOR FLAG
1102 005334 004737 004022      JSR PC,READ      ;GO READ ONE SECTOR
1103 005340 005737 012052      EMPBUFF:    TST SECCNTR      ;IF CLEARED NO SECTOR WAS FOUND
1104 005344 001002              BNE 2S
1105 005346 000137 006224      JMP EXIT      ;GO TO NEXT TRACK CAN'T READ THIS ONE
1106 005352 005037 006226      2S:        CLR BYTECNTR     ;CLEAR THE BYTE AND ERROR COUNTERS
1107 005356 005037 006234      CLR ERCNTR
1108 005362 052701 000200      BIS #BIT7,R1      ;R1 BIT 7 IS USED AS FIRST ERROR FLAG
1109 005366 004737 005460      JSR PC,ADJSUM     ;ADJUST DATA AND CK SUM FOR ADDRESSES
1110 005372 005037 005550      CLR CKSUM        ;SET UP FOR CHECK SUM ACCUMULATION
1111 005376 012746 006010      MOV #EMPDOONE,-(SP) ;SET UP RETURN ADDRESSES
1112 005402 012746 005552      MOV #EMPER,-(SP)
1113 005406 005046              CLR -(SP)        ;LOWER INTERRUPT LEVEL
1114 005410 012746 005416      MOV #1S,-(SP)
1115 005414 000002              RTI
1116 005416 012777 000103 173562 1S:      MOV #EBIE,ARXCS  ;LOAD EMPTY BUFFER FUNCTION
1117 005424 004737 006606      EMPFLAG:    JSR PC,TRCK      ;TEST FOR TR FLAG
1118
1119 005430 117737 173554 006230 006230 CKBYTE:    MOVB ARXDB,BADBYTE ;SAVE BYTE FROM DISKETTE
1120 005436 063737 006230 005550      ADD BADBYTE,CKSUM ;ACCUMULATE CHECK SUM
1121 005444 123720 006230      CMPB BADBYTE,(R0)+ ;COMPARE AGAINST GOOD BYTE
1122 005450 001054              BNE DATAER     ;IF NOT EQUAL GO TO DATAER
1123 005452 005237 006226      INC BYTECNTR
1124 005456 000762              BR EMPFLAG      ;GET NEXT BYTE
1125
1126 005460 113737 011124 015344 ADJSUM:    MOVB TARGET,BUFADR ;SET FIRST AND SECOND BYTES WITH ADDRESSES
1127 005466 113737 012054 015345      MOVB TSECTOR,BUFADR+1
1128 005474 013737 010372 005550      MOV SUM,CKSUM    ;GET THE PATTERN SUM
1129 005502 063737 011124 005550      ADD TARGET,CKSUM ;ADD TRACK ADDRESS TO CHECK SUM
1130 005510 063737 012054 005550      ADD TSECTOR,CKSUM ;ADD SECTOR ADDRESS TO CHECK SUM
1131 005516 113737 005550 015542      MOVB CKSUM,BUFADR+176 ;INSERT CHECK SUM TO DATA BUFFER
1132 005524 106337 005550      ASL 9,CKSUM      ;GENERATE NEGITIVE CHECK SUM
1133 005530 105437 005550      NEGB CKSUM
1134 005534 113737 005550 015543      MOVB CKSUM,BUFADR+177 ;INSERT NEG,SUM INTO DATA BUFFER

```

```

1135 005542 012700 015344      MOV #BUFADR,RO      ;SET ADDRESS OF BYTE IN RO
1136 005546 000207      RTS PC              ;RETURN
1137
1138 005550 000000      CKSUM:             0
1139
1140 005552 005726      EMPER:             TST (SP)+           ;REMOVE THE DONE RETURN FROM THE STACK
1141 005554 012737 014334 003114      MOV #EMPTY,PTYP1+2 ;PUT ADDR OF EMPTYBUF MESSAGE IN PAR ER TYP0UT 1
1142 005556 012737 014334 003202      MOV #EMPTY,PTYP2+2 ;PUT ADDR OF EMPTYBUF MESSAGE IN PAR ER TYP0UT 2
1143 005570 012737 005340 003162      MOV #EMPBUFF,PCONT+2 ;NOT HARD ERR RETURN THROUGH PCONT TO EMPTYBUF
1144 005576 000137 003076      JMP PARTEST        ;REPORT PARITY ERROR
1145
1146 005602 052737 000400 010464 DATAER:          BIS #BITS,UNITSEL ;SET THE HAD ERROR FLAG
1147 005610 005237 006234      INC ERCNTR         ;INC THE BYTE ERROR COUNTER
1148 005614 032777 020000 173372      BIT #SW13,SWR      ;TEST PRINT ERROR SW IN SWR
1149 005622 001062      BNE NOERTYP        ;DON'T PRINT THE ERROR
1150 005624 032777 010000 173362      BIT #SW12,SWR      ;TEST PRINT 10 ERRORS SWITCH
1151 005632 001404      BEQ 1S             ;PRINT ALL ERRORS
1152 005634 023727 006234 000012      CMP ERCNTR,#10.    ;HAVE 10 ERRORS BEEN TYPED
1153 005642 003052      BGT NOERTYP        ;YES,DON'T PRINT ANY MORE
1154 005644 005737 003750      1S:                TST DATAK         ;WAS THIS A READ CHECK DUE TO CRC ERROR
1155 005650 001403      BEQ 2S             ;NO REPORT DATA ERRORS
1156 005652 105701      TSTB R1            ;YES TEST FIRST ERROR FLAG
1157 005654 100017      BPL TYPERR         ;TYPE THE DATA CRC ERROR
1158 005656 000412      BR 3S
1159 005660 105701      2S:                TSTB R1            ;TEST FIRST ERROR FLAG
1160 005662 100014      BPL TYPERR
1161 005664 104400 014004      TYPE ,MCRLF
1162 005670 104400 013461      TYPE ,MDEHDR
1163 005674 104400 014004      TYPE ,MCRLF
1164 005700 004737 007476      JSR PC,TYPADR      ;PRINT TRACK AND SECTOR LOCATIONS
1165 005704 104400 013550      TYPE ,MCOLMUN      ;SET UP COLMUN HEADINGS
1166 005710 042701 000200      BIC #BIT7,R1       ;CLEAR FIRST ERROR FLAG
1167 005714 013737 006226 005726 TYPERR:          MOV BYTECNTR,1S    ;PRINT BYTE NUMBER
1168 005722 004537 013374      JSR R5,SGLDEC
1169 005726 000000      1S:                OPEN
1170 005730 104400 014001      TYPE ,DELSP
1171 005734 013746 006230      MOV BADBYTE,-(SP)  ;PRINT BYTE READ FROM DISKETTE
1172 005740 104402      TYP0S
1173 005742 000003      .WORD 3
1174 005744 104400 014001      TYPE ,DELSP
1175 005750 114037 006232      MOVB -(RO),GOODBYTE ;GET GOOD BYTE
1176 005754 005200      INC RO             ;RETURN RO TO NEXT BYTE IN BUFFER
1177 005756 013746 006232      MOV GOODBYTE,-(SP)
1178 005762 104403      TYP0N             ;PRINT GOOD DATA
1179 005764 104400 014004      TYPE ,MCRLF
1180 005770 005777 173220      NOERTYP:          TST SWR            ;TEST HALT ON ERROR SWITCH
1181 005774 100001      BPL CONT20
1182 005776 000000      HLT14:            HALT
1183 006000 005237 006226      CONT20:          INC BYTECNTR
1184 006004 000137 005424      JMP EMPFLAG
1185
1186 006010 005737 003750      EMPDONE:          TST DATAK         ;WAS THIS READCHECK CAUSED BY CRC ERROR
1187 006014 001401      BEQ 1S             ;NO,CONTINUE
1188 006016 000207      RTS PC             ;YES,RETURN TO CRC ERROR HANDLER

```



```

1189 006020 005737 006234 1S: TST ERCNTR ;WAS THERE ERRORS
1190 006024 001467 BEQ CONT3 ;NO ERRORS
1191 006026 032777 020000 173160 BIT #SW13,JSWR ;YES,TEST DON'T PRINT SWITCH
1192 006034 001024 BNE 2S ;DON'T PRINT THE ERROR
1193 006036 104400 014265 TYPE ,MERC ;PRINT THE TOTAL DATA ERROR COUNT
1194 006042 013737 006234 006054 MOV ERCNTR,3S
1195 006050 004537 013374 JSR RS,SGLDEC
1196 006054 000000 3S: OPEN
1197 006056 104400 015261 TYPE ,MSUM ;INDICATE IF CHECK SUM WAS GOOD OR HAD ERRORS
1198 006062 105737 005550 TSTB CKSUM
1199 006066 001403 BEQ 4S
1200 006070 104400 015021 TYPE ,MERS
1201 006074 000402 BR 5S
1202 006076 104400 015254 4S: TYPE ,MG00D
1203 006102 104400 014004 5S: TYPE ,MCRLF
1204 006106 012703 015622 2S: MOV #ZDATALOG,R3 ;SET INC.INST.FOR UNIT 0 ERROR LOG
1205 006112 004737 017604 JSR PC,U1LOG ;TEST FOR UNIT 1 ERROR LOGS
1206 006116 005213 INC (R3) ;INC DATA ERROR LOG
1207 006120 004737 003612 JSR PC,TRKERR ;INC ERROR PER TRACK COUNTER
1208 006124 032777 004000 173062 BIT #SW11,JSWR ;TEST NO RETRY SWITCH
1209 006132 001007 BNE 6S ;IF SET LOG HARD ERROR
1210 006134 005337 015552 DEC DATARETRY ;HAVE 10 ERRORS BEEN LOGED
1211 006140 001404 BEQ 6S
1212 006142 004737 004070 JSR PC,REREAD ;NO,GO REREAD THE DATA
1213 006146 000137 005340 JMP EMPBUFF ;GO RECHECK THE DATA
1214 006152 012703 015662 6S: MOV #ZHATALOG,R3 ;YES,SET INC.INST.FOR UNIT 0 ERROR LOG
1215 006156 004737 017604 JSR PC,U1LOG ;TEST FOR UNIT 1 ERROR LOGS
1216 006162 005213 INC (R3) ;INC HARD DATA NO STATUS ERROR
1217 006164 104400 014172 TYPE ,DALLF
1218 006170 104400 014707 TYPE ,MUNREC ;TYPE UNRECOVERABLE DATA NO
1219 006174 104400 013461 TYPE ,MDERHOR ;STATUS ERROR
1220 006200 104400 014004 TYPE ,MCRLF
1221 006204 005337 012052 CONT3: DEC SECCNTR
1222 006210 001405 BEQ EXIT
1223 006212 004737 004026 JSR PC,XREAD ;READ THE NEXT SECTOR
1224 006216 000137 005340 JMP EMPBUFF
1225 006222 005001 3S: CLR R1 ;CLEAR THE ONE READ FLAG
1226 006224 000207 EXIT: RTS PC
1227
1228 006226 000000 BYTECNTR: 0
1229 006230 000000 BADBYTE: 0
1230 006232 000000 GOODBYTE: 0
1231 006234 000000 ERCNTR: 0
1232
1233 ;*****
1234
1235 ;READ AND VERIFY ALL SECTORS WRITTEN BY
1236 ;PREVIOUS WRITE FUNCTION
1237
1238 ; T = 4
1239
1240 006236 004737 010072 RDCHK: JSR PC,GETPATTERN
1241 006242 004737 010710 XRDCHK: JSR PC,INITTRACKS
1242 006246 004737 010374 JSR PC,GETUNIT

```

1243 006252 004737 011020  
1244 006256 004737 005330  
1245 006262 005337 011122  
1246 006266 001371  
1247 006270 004737 010512  
1248 006274 000762

1S: JSR PC,GETTRACK  
JSR PC,READCHK  
DEC TRKCNT  
BNE 1S  
JSR PC,STOP  
BR XRDCHK

\*\*\*\*\*

:DO A READ ONLY FUNCTION ON ALL SECTORS  
:THIS DOES NOT VERIFY THE DATA, ONLY TESTS FOR CRC ERRORS  
: T = 5

1257 006276 004737 010072  
1258 006302 004737 010710  
1259 006306 004737 010374  
1260 006312 004737 011020  
1261 006316 004737 004022  
1262 006322 005337 011122  
1263 006326 001371  
1264 006330 004737 010512  
1265 006334 000762

RONLY: JSR PC,GETPATTERN  
XRONLY: JSR PC,INITTRACKS  
JSR PC,GETUNIT  
1S: JSR PC,GETTRACK  
JSR PC,READ  
DEC TRKCNT  
BNE 1S  
JSR PC,STOP  
BR XRONLY

\*\*\*\*\*

:WRITE AND READ CHECK ANY DATA PATTERN,AND ANY TRACK SEQUENCE  
:BUT ALTERNATE BETWEEN DRIVE UNITS ON EACH TRACK SELECTED.  
: T = 6

1275 006336 005137 003744  
1276 006342 004737 010072  
1277 006346 004737 010710  
1278 006352 004737 010374  
1279 006356 004737 011020  
1280 006362 004737 002744  
1281 006366 004737 005330  
1282 006372 004737 010374  
1283 006376 004737 002744  
1284 006402 004737 005330  
1285 006406 005337 011122  
1286 006412 001357  
1287 006414 004737 010512  
1288 006420 000752

DRVSWP: COM RDAFTWT ;SET THE READ AFTER WRITE FLAG  
JSR PC,GETPATTERN  
2S: JSR PC,INITTRACKS  
1S: JSR PC,GETUNIT ;SET UP FOR UNIT 0  
JSR PC,GETTRACK ;GET THE TRACK TO BE ACCESSED  
JSR PC,WRITE  
JSR PC,READCHK  
JSR PC,GETUNIT ;NOW GO TO UNIT 1 ON THE SAME TRACK  
JSR PC,WRITE  
JSR PC,READCHK  
DEC TRKCNT  
BNE 1S  
JSR PC,STOP  
BR 2S

\*\*\*\*\*

:THE FIRST HALF OF TEST 7 FORCES THE TRACK SEQUENCE  
:TO INCRIMENT UP THROUGH ALL THE TRACKS DOING WRITE / READ CHECK FUNCTIONS.  
:THIS VERIFIES THAT ALL THE TRACKS ARE ACCESSABLE.  
:THE SECOND HALF OF THE PASS WILL USE THE SEQUENCE SELECTED BY THE  
:OPERATOR AS INDICATED BELOW,AND ONLY READ CHECK THE DATA JUST WRITTEN.

1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296



```

1297
1298
1299
1300
1301
1302
1303 006422 004737 010072 TEST7: JSR PC,GETPATTERN
1304 006426 013737 011134 006604 MOV SEQUEN,SEGSV ;SAVE THE SELECTED SEQUENCE
1305 006434 012737 000001 011134 TEST7X: MOV #1,SEQUEN ;FORCE INCREMENT SEQUENCE
1306 006442 005137 003744 COM RDAFTWT ;SET READ AFTER WRITE FLAG FOR ERROR TESTING
1307 006446 004737 010710 JSR PC,INITTRACKS
1308 006452 004737 010374 JSR PC,GETUNIT
1309 006456 004737 011020 1S: JSR PC,GETTRACK
1310 006462 004737 002744 JSR PC,WRITE
1311 006466 004737 005330 JSR PC,READCHK
1312 006472 005337 011122 DEC TRKCNT ;IS THE FIRST HALF OF THE PASS DONE
1313 006476 001367 BNE 1S ;NO,CONTINUE WRITING AND READING
1314 006500 005037 003744 CLR RDAFTWT ;YES,NOW DO A READ ONLY
1315 006504 013737 006604 011134 MOV SEGSV,SEQUEN ;USING SELECTED SEQUENCE
1316 006512 004737 010710 JSR PC,INITTRACKS
1317 006516 004737 011020 2S: JSR PC,GETTRACK
1318 006522 004737 005330 JSR PC,READCHK
1319 006526 005337 011122 DEC TRKCNT ;IS THIS HALF OF THE PASS DONE
1320 006532 001371 BNE 2S ;NO,CONTINUE READ CHECKING
1321 006534 004737 010512 JSR PC,STOP ;YES,TEST FOR END OF PASS CONDITIONS
1322 006540 032737 040100 010464 BIT #40100,UNITSEL ;HAVE ALL SELECTED UNITS BEEN USED
1323 006546 001332 BNE TEST7X ;IF A USED BIT IS STILL SET GO TO NEXT DRIVE
1324 006550 012706 001200 MOV #1200,SP ;RESET STACK ADDRESS FOR ACT11 EOP
1325 006554 032737 001000 001212 BIT #BIT9,DTESTP ;NEXT PASS,COMPLEMENT D D BIT
1326 006562 001404 BEQ 3S
1327 006564 042737 001000 001212 BIC #BIT9,DTESTP ;IT WAS ON CLEAR IT
1328 006572 000720 BR TEST7X
1329 006574 052737 001000 001212 3S: BIS #BIT9,DTESTP ;IT WAS OFF SET IT
1330 006602 000714 BR TEST7X
1331
1332 006604 000000 SEGSV: 0
1333 ;*****
1334
1335 ;TEST FOR TRANSFER FLAG AND PROGRAM NOT HUNG
1336
1337 006606 005037 006772 TRCK: CLR HCNTR1 ;CLEAR HUNG COUNTER
1338 006612 012746 000340 MOV #PR7,-(SP) ;RAISE INTERRUPT LEVEL TO INHIBIT INTERRUPTS
1339 006616 012746 006624 MOV #3S,-(SP)
1340 006622 000002 RTI
1341 006624 105777 172356 3S: TSTB 3RXCS ;TEST FOR TR FLAG
1342 006630 100001 BPL 1S ;SKIP NEXT INST. IF NOT SET
1343 006632 000207 RTS PC ;RETURN TO TRANSFER DATA
1344 006634 023727 006226 000200 1S: CMP BYTECNTR,#128. ;HAVE 128 BYTES BEEN TRANSFERED
1345 006642 001403 BEQ 2S ;ALLOW AN INTERRUPT IF EQUAL
1346 006644 005237 006772 INC HCNTR1 ;IF THE HUNG COUNTER OVERFLOWS EITHER
1347 ;THERE WAS NO TR FLAG OR NO INTERRUPT OCCURED
1348 ;AFTER 128 BYTES TRANSFERED
1349 006650 001365 BNE 3S ;NO OVERFLOW RETEST FOR TR FLAG
1350 006652 132777 000040 172326 2S: BITB #DONEBIT,3RXCS ;TEST FOR A DONE FLAG WHICH WOULD INDICATE A

```

```

;THIS VERIFIES THAT THE DATA CAN BE READ FROM A TRACK AFTER THE HEAD HAS BEEN
;MOVED AWAY FROM AND BACK TO THAT TRACK. AT THE COMPLETION OF THE PASS
;THE DELETED DATA BIT IN TEST CONDITIONS IS COMPLEMENTED AND THE NEXT
;PASS WILL BE RUN UNDER THIS NEW CONDITION.
; T = 0 OR 7

```

```

1351
1352
1353 006660 001407          BEQ 45
1354 006662 005037 006226  CLR BYTECNTR
1355
1356 006666 005726          TST (SP)+
1357 006670 005046          CLR -(SP)
1358 006672 012746 006700  MOV #45,-(SP)
1359 006676 000002          RTI
1360 006700 005237 006772  45: INC HCNTR1
1361 006704 001362          BNE 25
1362 006706 104400 014364  TYPE ,MHUNG
1363 006712 000000          HLT21: HALT
1364
1365
1366
1367 006714 012603          DONECK: ;TEST FOR DONE FLAG AND PROGRAM HUNG
1368
1369 006716 005037 006772  MOV (SP)+,R3
1370 006722 012737 177740 006774  CLR HCNTR1
1371 006730 032777 000040 172250 25: MOV #177740,HCNTR2
1372 006736 001402          BIT #DONEBIT,ARXCS
1373 006740 010346          BEQ 15
1374 006742 000207          MOV R3,-(SP)
1375 006744 062737 000001 006772 15: RTS PC
1376 006752 005537 006774          ADD #1,HCNTR1
1377 006756 005737 006774          ADC HCNTR2
1378 006762 001362          TST HCNTR2
1379 006764 104400 014364          BNE 25
1380 006770 000000          HLT20: TYPE ,MHUNG
1381
1382
1383 006772 000000          HCNTR1: 0
1384 006774 177740          HCNTR2: 177740
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402 007036 117737 172146 007650 NOINTER: INC INTER
1403 007044 005777 172136          BIT #SW13,ASWR
1404 007050 100446          BNE 15
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500

```

```

;POSSABLE ERROR CONDITION EXISTS WHILE
;TRANSFERING COMMANDS OR DATA
;IF DONE NOT SET TYPE HUNG
;CLEAR BYTE COUNTER SO IT WON'T INTERFERE
;WITH TESTING FOR DONE.
;REMOVE THIS RETURN ADDRESS FROM STACK
;LOWER INTERRUPT LEVEL TO ACCEPT AN INTERUPT
;THE INTERUPT WILL RETURN THROUGH PREVIOUSLY
;SET ADDRESSES
;ALLOW SOME TIME FOR FLAG TO SET

;PROGRAM HUNG, SP (177706) CONTAINS RETURN
;ADDRESS OF WAITING LOOP

;TAKE THE RETURN ADDRESS OF THE STACK INCASE
;THE PROGRAM IS EXPECTING AN INTERUPT AT DCNE
;CLEAR THE LOWER HUNG COUNTER
;SET UPER COUNTER FOR 31 FULL LOOPS
;TEST FOR DONE FLAG
;IF NOT SET SKIP RETURN
;PUT RETURN ADDRESS BACK ON THE STACK
;RETURN
;WHEN THE DOUBLE WORD COUNTER OVERFLOWES
;THE PROGRAM MUST BE HUNG

;IF NO OVERFLOW RETEST FOR DONE

;PROGRAM HUNG, CONTENTS OF R3 (177703) HOLDES
;ADDRESS OF WAITING TEST.

;AN INTERUPT DID NOT OCCURE AT A FUNCTION DONE FLAG.

;INC INTERUPT ERROR COUNTER
;TEST DON'T PRINT ERROR SWITCH

;TYPE NO INTERUPT ON DONE ERROR

;TEST HALT ON ERROR SWITCH

;HALT ON ERROR
;JSR TO INTSERV AS IF IT WAS AN INTERUPT

;*****

;INTERUPT SERVICE

;SAVE THE ERROR AND STATUS WORD
;TEST THE ERROR FLAG
;THERE WAS AN ERROR GO REPORT IT

```



1405	007052	032737	000004	007650		BIT #BIT2,ASTAT		: IS INIT DONE SET
1406	007060	001402				BEQ 25		: NO CONTINUE
1407	007062	000137	007462			JMP RXPWR		: YES, REPORT POWER FAILED AND RESTART
1408	007066	032737	000003	007650	25:	BIT #3,ASTAT		: ARE PAR OR CRC BITS SET
1409	007074	001022				BNE 15		: YES GO LOG ERROR
1410	007076	132777	000040	172102		BITB #DONEBIT,ARXCS		: IS DONE SET
1411	007104	001013				BNE 35		: IF SET RETURN TO TEST
1412	007106	005237	015572			INC LKNINT		: INC UNKNOWN INTERRUPT ERROR LOG
1413	007112	032777	020000	172074		BIT #SW13,ASWR		: TEST DON'T PRINT ERROR SWITCH
1414	007120	001004				BNE 45		: DON'T PRINT
1415	007122	104400	014244			TYPE ,MKNINT		: TYPE UNKNOWN INTERRUPT
1416	007126	104400	014004			TYPE ,MCLRF		
1417	007132	000002			45:	RTI		: RETURN FROM THE INTERRUPT
1418	007134	062706	000006		35:	ADD #6,SP		: BYPASS INTERRUPT POINTERS ON STACK
1419	007140	000207				RTS PC		: RETURN TO PROGRAM
1420	007142	005237	015570		15:	INC NOERLOG		: NO STATUS ERROR FLAG ERROR
1421	007146	032777	020000	172040		BIT #SW13,ASWR		: TEST DON'T PRINT ERROR SWITCH
1422	007154	001004				BNE RXERROR		
1423	007156	104400	014766			TYPE ,MNOFLAG		: TYPE NO STATUS ERROR ERROR
1424	007162	104400	014004			TYPE ,MCLRF		
1425								
1426	007166	052737	000400	010464	RXERROR:	BIS #BIT8,UNITSEL		: SET HAD ERROR FLAG
1427	007174	012737	000012	015554		MOV #10,P2RETRY		: SET RETRY COUNTER
1428	007202	012777	000017	171776	25:	MOV #ORDER,ARXCS		: GET THE ERROR CODE
1429	007210	004737	006714			JSR PC,DONECK		: TEST FOR DONE FLAG
1430	007214	032777	000002	171766		BIT #2,ARXDB		: WAS THERE A PARITY ERROR
1431	007222	001403				BEQ 15		: NO CONTINUE
1432	007224	004737	001760			JSR PC,STATER		: YES GO REPORT THE PARITY ERROR
1433	007230	000764				BR 25		: REISSUE THE FUNCTION
1434	007232	117737	171752	007652	15:	MOVB ARXDB,BSTAT		: SAVE THE ERROR CODE IN B STATUS
1435	007240	005737	007652			TST BSTAT		: IS THERE A DEFINITE CODE
1436	007244	001407				BEQ NOPRNT		: NO CONTINUE
1437	007246	013705	007652			MOV BSTAT,RS		: ADJUST ERROR CODE TO PRODUCE AN EVEN ADDR
1438	007252	006005				ROR RS		: OF THE CORRESPONDING COUNTER
1439	007254	006005				ROR RS		
1440	007256	062705	015670			ADD #ERCODE-2,RS		: ADD ON ADDR OF ERROR LOG -2, AS THERE IS NO 0
1441								: ERROR CODE. THE CONTENTS OF RS WILL READJUST
1442								: ADDRESS TO CORRECT LOG.
1443	007262	005215				INC (RS)		: INC THE CORRECT ERROR CODE COUNTER
1444	007264	032777	020000	171722	NOPRNT:	BIT #SW13,ASWR		: TEST PRINT ERROR SWITCH IN SWR
1445	007272	001032				BNE 25		
1446	007274	104400	014004			TYPE ,MCLRF		
1447	007300	104400	014015			TYPE ,MERHEADER		: TYPE ERROR HEADER
1448	007304	104400	015074			TYPE ,MPASS		: TYPE PASSES COMPLETED AT ERROR
1449	007310	013737	017602	007322		MOV PASCNTR,15		
1450	007316	004537	013374			JSR RS,SGLDEC		
1451	007322	000000			15:	OPEN		
1452	007324	104400	014004			TYPE ,MCLRF		
1453	007330	104400	014121			TYPE ,MRXCS		: TYPE COMMAND STATUS REGISTER
1454	007334	013746	003742			FUNCTION,-(SP)		: SAVE FUNCTION FOR TYPEOUT
1455	007340	104402						: GO TYPE--OCTAL ASCII
1456	007342	003						: TYPE 3 DIGIT(S)
1457	007343	000						: SUPPRESS LEADING ZEROS
1458	007344	004737	007476			JSR PC,TYPADR		: TYPE ADDRESSES AND RUN CONDITIONS



1459	007350	104400	014004			TYPE ,MCRLF	
1460	007354	004737	010030			JSR PC,TYP CODE	;PRINT THE STATUS REGISTERS
1461	007360	032737	000020	003742	2S:	BIT #BIT4,FUNCTION	;WHAT DRIVE IS BEING USED
1462	007366	001006				BNE 6S	
1463	007370	012777	000013	171610		MOV #RD0STAT,ARXCS	;DRIVE 0 BEING USED
1464	007376	004737	006714		5S:	JSR PC,DONECK	;TEST FOR DONE FLAG
1465	007402	000404				BR 7S	
1466	007404	012777	000033	171574	6S:	MOV #RD1STAT,ARXCS	;DRIVE 1 BEING USED
1467	007412	000771				BR 5S	
1468	007414	032777	000002	171566	7S:	BIT #2,ARXDB	;WAS THERE A PARITY ERROR
1469	007422	001403				BEQ 3S	;NO CONTINUE
1470	007424	004737	001760			JSR PC,STATER	;YES REPORT THE ERROR
1471	007430	000753				BR 2S	;REISSUE THE COMMAND
1472	007432	105777	171552		3S:	TSTB ARXDB	;WAS DRIVE READY SET
1473	007436	100406				BMI 4S	;YES RETURN
1474	007440	104400	013750			TYPE ,MNO DRY	
1475	007444	104400	014004			TYPE ,MCRLF	
1476	007450	004737	007654			JSR PC,DELUNIT	;NO GO DELETE THAT UNIT
1477	007454	062706	000004		4S:	ADD #4,SP	;MOVE ERROR RETURN TO TOP OF STACK
1478	007460	000207				RTS PC	
1479							
1480	007462	104400	015276		RXPWR:	TYPE ,MRX11	;ONLY THE RX11 POWER HAS FAILED
1481	007466	104400	013070			TYPE ,SPOWER	;PRINT POWER FAILED
1482	007472	000137	002532			JMP RESTART	;GO TO RESTART
1483							
1484							
1485	007476	104400	013507		TYPADR:	TYPE ,MTRK	;TYPE TRACK ADDRESS
1486	007502	013737	011124	007514		MOV TARGET,3S	
1487	007510	004537	013374			JSR RS,SGLDEC	
1488	007514	000000			3S:	OPEN	
1489	007516	104400	013536			TYPE ,MSECT	;TYPE SECTOR ADDRESS
1490	007522	013737	012054	007542		MOV TSECTOR,2S	
1491	007530	042737	177740	007542		BIC #177740,2S	;CLEAR ALL BUT SECTOR ADDRESS
1492	007536	004537	013374			JSR RS,SGLDEC	
1493	007542	000000			2S:	OPEN	
1494	007544	104400	014001			TYPE ,DBLSP	
1495	007550	032737	000020	010464		BIT #BIT4,UNITSEL	;WHICH DRIVE IS BEING USED
1496	007556	001003				BNE 1S	
1497	007560	104400	014040			TYPE ,MUNIT0	;TYPE UNIT 0
1498	007564	000402				BR TYPSEL	
1499	007566	104400	014050		1S:	TYPE ,MUNIT1	;TYPE UNIT 1
1500	007572	104400	014176		TYPSEL:	TYPE ,MPAT	;TYPE PATTERN CODE
1501	007576	013746	010200			PAT,-(SP)	::SAVE PAT FOR TYPEOUT
1502	007602	104402				MOV	::GO TYPE--OCTAL ASCII
1503	007604	001				TYPOS	::TYPE 1 DIGIT(S)
1504	007606	000				.BYTE	::SUPPRESS LEADING ZEROS
1505	007606	104400	014001			.BYTE	
1506	007612	104400	014202			TYPE ,DBLSP	;TYPE TEST CODE
1507	007616	013746	002052			TYPE ,MTEST	
1508	007622	104402				TEST,-(SP)	::SAVE TEST FOR TYPEOUT
1509	007624	001				MOV	::GO TYPE--OCTAL ASCII
1510	007625	000				TYPOS	::TYPE 1 DIGIT(S)
1511	007626	104400	014001			.BYTE	::SUPPRESS LEADING ZEROS
1512	007632	104400	014206			.BYTE	
						TYPE ,DBLSP	
						TYPE ,MSEQ	;TYPE SEQUENCE CODE



```

1513 007636 013746 011134      MOV      SEQUEN,-(SP)      ;;SAVE SEQUEN FOR TYPEOUT
1514 007642 104402              TYPOS    ;;GO TYPE--OCTAL ASCII
1515 007644      001          .BYTE   1      ;;TYPE 1 DIGIT(S)
1516 007645      000          .BYTE   0      ;;SUPPRESS LEADING ZEROS
1517 007646 000207              RTS PC
1518
1519 007650 000000      ASTAT:   0
1520 007652 000000      BSTAT:   0
1521
1522 007654 132737 000020 003742 DELUNIT:  BITB #BIT4,FUNCTION ;TEST FOR PRESENTLY USED UNIT
1523 007662 001016              BNE 3$          ;UNIT 0 HAS BEEN DELETED FROM USE
1524 007664 104400 014040      TYPE ,MUNIT0
1525 007670 104400 014060      TYPE ,MDELET
1526 007674 104400 014172      TYPE ,DBLLF
1527 007700 042737 000200 010464  BIC #BIT7,UNITSEL ;CLEAR UNIT 0 SELECTION BIT
1528 007706 005737 010464      TST UNITSEL    ;WAS UNIT 1 SELECTED FOR USE
1529 007712 100020              BPL 1$          ;UNIT 1 NOT SELECTED GO DUMP ERROR REPORT
1530 007714 000137 002560      JMP TESTSEL    ;CONTINUE ON OTHER UNIT AT BEGINING OF TEST
1531 007720 104400 014050      TYPE ,MUNIT1  ;UNIT 1 HAS BEEN DELETED FROM USE
1532 007724 104400 014060      TYPE ,MDELET
1533 007730 104400 014172      TYPE ,DBLLF
1534 007734 042737 100000 010464  BIC #BIT15,UNITSEL ;CLEAR UNIT 1 SELECTION BIT
1535 007742 105737 010464      TSTB UNITSEL  ;WAS UNIT 0 SELECTED FOR USE
1536 007746 100762              BMI 2$          ;YES CONTINUE ON UNIT 0
1537 007750 005137 021254      COM SHORTRPT ;SET SHORT REPORT FLAG
1538 007754 000137 017622      JMP ERDUMP    ;CAN'T CONTINUE TYPE OUT STATISTICAL REPORT
1539
1540
1541 007760 117737 171224 007650 RDCODE:  MOVB #RDXB,ASTAT ;SAVE THE A STATUS
1542 007766 012777 000017 171212 2$:  MOV #RDR, #RXC ;READ THE B STATUS REGISTER
1543 007774 004737 006714              JSR PC,DONECK ;WAIT FOR DONE FLAG
1544 010000 032777 000002 171202      BIT #2,RDXB   ;WAS THERE A PARITY ERROR
1545 010006 001403              BEQ 1$          ;NO,CONTINUE
1546 010010 004737 001760              JSR PC,STATER ;YES,REPORT THE PARITY ERROR
1547 010014 000764              BR 2$          ;RETRY READING STATUS B
1548 010016 117737 171166 007652 1$:  MOVB #RDXB,BSTAT ;SAVE THE B STATUS CODES
1549 010024 104400 014004      TYPE ,MCRLF
1550 010030 104400 014142      TYPE ,MASTAT  ;TYPE THE CONTENTS OF THE TWO STATUS REGISTERS
1551 010034 013746 007650      TYP CODE:  ASTAT,-(SP)  ;;SAVE ASTAT FOR TYPEOUT
1552 010040 104402              TYPOS    ;;GO TYPE--OCTAL ASCII
1553 010042      003          .BYTE   3      ;;TYPE 3 DIGIT(S)
1554 010043      000          .BYTE   0      ;;SUPPRESS LEADING ZEROS
1555 010044 104400 013772      TYPE ,TAB
1556 010050 104400 014156      TYPE ,MBSTAT
1557 010054 013746 007652      MOV      BSTAT,-(SP)  ;;SAVE BSTAT FOR TYPEOUT
1558 010060 104402              TYPOS    ;;GO TYPE--OCTAL ASCII
1559 010062      003          .BYTE   3      ;;TYPE 3 DIGIT(S)
1560 010063      000          .BYTE   0      ;;SUPPRESS LEADING ZEROS
1561 010064 104400 014004      TYPE ,MCRLF
1562 010070 000207              RTS PC
1563
1564
1565
1566
;*****

```

1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620

: DECODES THE DATA PATTERN SELECTED IN THE SWR  
: BITS 6,7, AND 8 OF THE INITIAL SWR SELECTED THE DATA PATTERN  
: TO BE USED AS FOLLOWS:

BITS			DATA PATTERN
8	7	6	
0	0	0	NO PATTERN SPECIFIED (FORCE RANDOM DATA)
0	0	1	ALL ZEROS
0	1	0	ALL ONES
0	1	1	FLOATING ZERO
1	0	0	FLOATING ONE
1	0	1	ALTERNATING BITS
1	1	0	ALTERNATING PAIRS OF BITS
1	1	1	RANDOM

: NOTE: ALL DATA PATTERNS WILL BE MODIFIED SO THE FIRST BYTE WILL  
: CONTAIN THE TRACK ADDRESS. THE SECOND BYTE WILL CONTAIN THE UNIT  
: NUMBER AND SECTOR ADDRESS IN WHICH THE DATA IS WRITTEN. THE LAST  
: TWO BYTES CONTAIN THE CHECK SUM NUMBERS.

:\*\*\*\*\*

```

1591 010072 142737 000377 010140 GETPATTERN: BICB #377,2#BRONPAT ;CLEAR BRANCH OFFSET
1592 010100 005037 010372          CLR SUM ;SET UP FOR ACCUMULATION OF CHECK SUM
1593 010104 005737 010200          TST PAT ;IF NO PATTERN SPECIFIED FORCE PATTERN 7
1594 010110 001003          BNE 1$
1595 010112 012737 000007 010200 1$: MOV #7,PAT ;GET PATTERN BITS
1596 010120 013704 010200          MOV PAT,R4 ;ADJUST FOR CORRECT OFFSET
1597 010124 005304          DEC R4
1598 010126 006304          ASL R4
1599 010130 150437 010140          BISB R4,2#BRONPAT ;INSERT OFFSET
1600 010134 012704 015346          MOV #BUFADR+2,R4 ;SET UP ADDRESS OF FIRST BYTE
1601 010140 000777          BR ;BRANCH BY OFFSET SELECTED
1602 010142 000137 010202          JMP DATA0 ;000 DATA BYTE
1603 010146 000137 010214          JMP DATA1 ;377 DATA BYTE
1604 010152 000137 010224          JMP FLOAT0 ;FLOAT A 0 THROUGH ALL 1'S
1605 010156 000137 010266          JMP FLOAT1 ;FLOAT A 1 THROUGH ALL 0'S
1606 010162 000137 010274          JMP PAT125 ;125/052 DATA WORD
1607 010166 000137 010314          JMP PAT314 ;314/063 DATA WORD
1608 010172 000137 010324          JMP RANDATA ;RANDOM DATA BYTE
1609
1610
1611 010176 000000          DATABYTE: 0
1612 010200 000000          PAT: 0
1613
1614
1615 ;*****
1616
1617 ;LOAD SOFTWARE BUFFER WITH ALL ZEROS
1618 ; P = 1
1619
1620 010202 005037 010176          DATA0: CLR DATABYTE

```



```

1621 010206 004737 010344 PATGEN: JSR PC,LOAD ;GO LOAD THE DATA BUFFER
1622 010212 000775 BR PATGEN
1623
1624 ;*****
1625
1626 ;LOAD SOFTWARE BUFFER WITH ALL ONES
1627 ; P = 2
1628
1629
1630 010214 112737 000377 010176 DATA1: MOVB #377,DATABYTE
1631 010222 000771 BR PATGEN
1632
1633 ;*****
1634
1635 ;FLOAT A 0 THROUGH ONES IN SOFTWARE BUFFER
1636 ; P = 3
1637
1638
1639 010224 112737 000376 010176 FLOAT0: MOVB #376,DATABYTE ;SET UP A ONES FIELD
1640 010232 000261 XPATGEN: SEC ;SET THE C BIT TO ROTATE THROUGH THE DATA
1641 010234 012702 000000 1S: MOV #0,R2 ;CLR R2 (CAN'T USE "CLR" AS IT CLEARS "C" BIT)
1642 010240 103001 BCC 2S ;BR IF THE "C" BIT IS CLEARED
1643 010242 005202 INC R2 ;SET R2 IF NOT
1644 010244 004737 010344 2S: JSR PC,LOAD ;GO LOAD THE DATA BUFFER
1645 010250 000241 CLC
1646 010252 005702 TST R2 ;IS R2 NONZERO
1647 010254 001401 BEQ 3S
1648 010256 000261 SEC ;YES, SET THE "C" BIT
1649 010260 106137 010176 3S: ROLB DATABYTE
1650 010264 000763 BR 1S
1651
1652 ;*****
1653
1654 ;FLOAT A 1 THROUGH ALL ZEROS IN SOFTWARE BUFFER
1655 ; P = 4
1656
1657
1658 010266 005037 010176 FLOAT1: CLR DATABYTE
1659 010272 000757 BR XPATGEN
1660
1661 ;*****
1662
1663 ;LOAD SOFTWARE BUFFER WITH ALTERNATING 1 AND 0 FOR
1664 ;ONE BYTE AND THE COMPLIMENT INTO THE NEXT
1665 ; P = 5
1666
1667
1668 010274 112737 000125 010176 PAT125: MOVB #125,DATABYTE
1669 010302 004737 010344 XXPATGEN: JSR PC,LOAD
1670 010306 105137 010176 COMB DATABYTE
1671 010312 000773 BR XXPATGEN
1672
1673 ;*****
1674
    
```

```

1675                                     ;LOAD SOFTWARE BUFFER WITH ALTERNATING PAIRS OF 1 AND 0 AND
1676                                     ;COMPLIMENT INTO THE NEXT
1677                                     ; P = 6
1678
1679 010314 112737 000314 010176 PAT314:      MOVB #314,DATABYTE
1680 010322 000767                                     BR XXPATGEN
1681
1682                                     ;*****
1683
1684                                     ;LOAD SOFTWARE BUFFER WITH RANDOM DATA PATTERN
1685                                     ; P = 0 OR 7
1686
1687 010324 004737 011666 010176 RANDATA:      JSR PC,RANGEN          ;GET RANDOM NUMBER
1688 010330 113737 011760 010176      MOVB RANUM,DATABYTE
1689 010336 004737 010344      JSR PC,LOAD
1690 010342 000770      BR RANDATA
1691
1692 010344 063737 010176 010372 LOAD:          ADD DATABYTE,SUM      ;ACCUMULATE THE PATTERN CHECK SUM
1693 010352 113724 010176      MOVB DATABYTE,(R4)+   ;LOAD THE DATA BUFFER
1694 010356 022704 015542      CMP #BUFADR+176,R4   ;HAVE 124 BYTES BEEN GENERATED
1695 010362 001401      BEQ 15                ;IF YES, RETURN TO TEST
1696 010364 000207      RTS PC                ;IF NO, RETURN TO PATTERN GENERATOR
1697 010366 005726 15:          TST (SP)+             ;TAKE PATTERN RETURN ADDRESS OF STACK
1698 010370 000207      RTS PC                ;RETURN TO TEST
1699
1700 010372 000000 SUM:          0
1701
1702                                     ;*****
1703
1704                                     ;TEST FOR SELECTED UNITS, DRIVE READY, AND USED CONDITIONS
1705                                     ;ALSO CONTAINS A "HAD ERROR" FLAG TO BE TESTED AT EOP.
1706                                     ;THE BITS IN UNITSEL ARE USED AS FOLLOWS
1707
1708                                     ;BIT15 =UNIT 1 SELECTED VIA SWR
1709                                     ;BIT14 =UNIT 1 USED BIT
1710                                     ;BIT8  =THIS PASS HAD AN ERROR
1711                                     ;BIT7  =UNIT 0 SELECTED VIA SWR
1712                                     ;BIT6  =UNIT 0 USED BIT
1713                                     ;BIT4  =UNIT SELECTION FOR FUNCTION WORD
1714
1715                                     ;*****
1716
1717 010374 032737 000100 010464 GETUNIT:      BIT #BIT6,UNITSEL    ;WAS UNIT 0 JUST USED
1718 010402 001012      BNE 15                ;UNIT 0 USED CHECK UNIT 1
1719 010404 105737 010464      TSTB UNITSEL         ;WAS UNIT 0 SELECTED
1720 010410 100007      BPL 15                ;NO GO TO UNIT 1
1721 010412 042737 040020 010464      BIC #40020,UNITSEL  ;CLEAR UNIT 1 USED BIT AND FUNCTION UNIT BIT
1722 010420 052737 000100 010464      BIS #BIT6,UNITSEL   ;SET UNIT 0 USED BIT
1723 010426 000207      RTS PC
1724 010430 005737 010464 15:          TST UNITSEL         ;WAS UNIT 1 SELECTED
1725 010434 100012      BPL 25                ;NO RETURN
1726 010436 032737 040000 010464      BIT #BIT14,UNITSEL ;WAS UNIT 1 BEEN USED
1727 010444 001006      BNE 25                ;YES RETURN
1728 010446 042737 000100 010464      BIC #BIT6,UNITSEL  ;CLEAR UNIT 0 USED BIT

```



```

1729 010454 052737 040020 010464          BIS #40020,UNITSEL      ;SET UNIT 1 USED BIT AND FUNCTION UNIT BIT
1730 010462 000207          RTS PC
1731
1732
1733
1734 010464 000000          UNITSEL:      0
1735
1736          ;TEST THAT ALL UNITS HAVE BEEN ACCESSED
1737
1738 010466 005737 010464          DONE:      TST UNITSEL          ;IS UNIT 1 SELECTED
1739 010472 100006          BPL IS          ;NO RETURN
1740 010474 032737 040000 010464          BIT #BIT14,UNITSEL      ;YES HAS IT BEEN USED
1741 010502 001002          BNE IS          ;YES RETURN
1742 010504 062706 000002          ADD #2,SP          ;BYPASS NOT DONE RETURE ON STACK
1743 010510 000207          IS:      RTS PC
1744
1745
1746 010512 004737 010466          STOP:      JSR PC,DONE          ;HAVE ALL DRIVES BEEN USED
1747 010516 005237 017602          INC PASCNTR          ;KEEP TOTAL NUMBER OF PASSES
1748 010522 005737 010706          TST CHARCT          ;HAVE 72 EOP INDICATORS BEEN TYPED
1749 010526 001005          BNE 3$
1750 010530 104400 014004          TYPE ,MCRLF          ;YES DO A CRLF
1751 010534 012737 177670 010706          MOV #72,CHARCT          ;RESET CHARACTER COUNTER
1752 010542 005237 010706          3$:      INC CHARCT
1753 010546 032737 000400 010464          BIT #BIT8,UNITSEL      ;TEST HAD ERROR FLAG
1754 010554 001403          BEQ 2$          ;NO ERROR PRINT *
1755 010556 104400 014011          TYPE ,HEREOP          ;HAD ERROR PRINT ?
1756 010562 000404          BR IS
1757 010564 104400 014007          2$:      TYPE ,MEOP          ;PRINT EOP INDICATOR
1758 010570 104400 014013          TYPE ,MABELL
1759 010574 032777 040000 170412          1$:      BIT #SW14,JSWR          ;TEST EOP HALT SWITCH
1760 010602 001417          BEQ CONT22
1761 010604 104400 014004          TYPE ,MCRLF
1762 010610 104400 015074          TYPE ,MPASS          ;AT HALT ON PASS PRINT NUMBER OF PASSES COMPLETE
1763 010614 013737 017602 010626          MOV PASCNTR,6$
1764 010622 004537 013374          JSR RS,SGLDEC
1765 010626 000000          6$:      OPEN
1766 010630 104400 014004          TYPE ,MCRLF
1767 010634 000000          HLT16:      HALT
1768 010636 005037 010706          CLR CHARCT          ;THIS CAUSES A CRLF AFTER HALT AT E.O.P.
1769 010642 042737 040500 010464          CONT22:      BIC #40500,UNITSEL      ;CLEAR USED BITS, AND "HAD ERROR" BIT
1770 010650 022737 000007 010200          CMP #7,PAT          ;IF RANDOM DATA GET NEW DATA FOR BUFFER
1771 010656 001012          BNE HERE
1772 010660 004737 010072          JSR PC,GETPATTERN
1773 010664 013705 000042          2$:      MOV #42,RS          ;ACT11 END OF PASS HOOKS
1774 010670 001405          BEQ HERE
1775 010672 000005          RESET
1776 010674 004715          LOGICAL:      JSR PC,(RS)
1777 010676 000240          NOP
1778 010700 000240          NOP
1779 010702 000240          NOP
1780 010704 000207          HERE:      RTS PC
1781
1782 010706 000000          CHARCT:      0
    
```

1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836

\*\*\*\*\*

; INITIALIZE TRACK SEQUENCE

```

INITTRACK:      BIC #100200,0D      ;CLEAR FIRST USED BITS
                TST 00              ;TEST CONTENTS OF ID,00 FOR 0
                BNE IS              ;ID,00 SPECIFIED USE THEM
                MOV #114, ID         ;NONE SPECIFIED SET ID TO MAXIMUM
                CLRB 00              ;SET 00 TO MINIMUM
                MOV 00, TARGET        ;INIT 00 AS PRESENT TRACK
                CLR XID               ;INIT WORKING ID AND 00 LOCATIONS
                MOV #ID, XID
                CLR X0D
                MOV 00, X0D
                MOV XID, TRKCNTR      ;SET UP NUMBER OF TRACK MOVEMENTS
                SUB X0D, TRKCNTR
                INC TRKCNTR
                BIS #100200,0D        ;SET FIRST TIME BITS IN ID,00
                RTS PC

```

\*\*\*\*\*

```

; TEST FOR HEAD SEQUENCE SELECTED BY INITIAL SWR SETTING
; BITS 0, 1, AND 2 OF THE INITIAL SWR SELECTED THE TRACK SEQUENCING
; TO BE FOLLOWED AS INITIATED BELOW

```

BITS			SEQUENCE
0	0	0	NO SEQUENCE SPECIFIED (DEFAULT TO SEQ 7)
0	0	1	INCREMENT FROM 00 TO ID
0	1	0	DECREMENT FROM ID TO 00
0	1	1	DO PREVIOUS 2 SEQUENCES
1	0	0	BOUNCE BETWEEN ID AND 00
1	0	1	DECREASING BOUNCE
1	1	0	STROBE BETWEEN 00 AND DECREMENTING ID
1	1	1	RANDOM TRACK SELECTION

\*\*\*\*\*

```

GETTRACK:      MOV #TARGET, PRESTRK ;RESET TO PRESENT TRACK
                BICB #377, 2#BRONTRK ;CLEAR OUT BRANCH OFFSET
                TST SEQUEN           ;IF NO SEQUENCE SPECIFIED FORCE SEQUENCE 7
                BNE IS
                MOV #7, SEQUEN
                MOV SEQUEN, R4
                DEC R4
                ASL R4
                BISB R4, 2#BRONTRK   ;THIS BR INST. IS MODIFIED BY THE TEST CONDITIONS
                ;SELECTED , TO BRANCH TO ONE OF THE SEQ. BELOW

```



```

1837 011064 000777          BRONTRK:          BR          ; BRANCH BY OFFSET SELECTED
1838 011066 000137 011136    JMP SEQ1         ; INCREMENT
1839 011072 000137 011172    JMP SEQ2         ; DECREMENT
1840 011076 000137 011224    JMP SEQ3         ; INCREMENT/DECREMENT
1841 011102 000137 011270    JMP SEQ4         ; BOUNCE ID TO OD
1842 011106 000137 011354    JMP SEQ5         ; DECREASING BOUNCE
1843 011112 000137 011506    JMP SEQ6         ; STROBE
1844 011116 000137 011566    JMP SEQ7         ; RANDOM
1845
1846 011122 000000          TRKCNTR:        0
1847 011124 000000          TARGET:        0
1848 011126 000000          PRESTRK:       0
1849 011130 000000          XOD:           0
1850 011132 000000          XID:           0
1851 011134 000000          SEQUEN:        0
1852
1853 ;*****
1854
1855 ; INCREMENT FROM OD+1 TO ID AND RETURN TO OD
1856 ; S = 1
1857
1858 011136 042737 100200 001200 SEQ1:          BIC #100200,OD ; CLEAR FIRST TIME BITS
1859 011144 123737 011132 011126    CMPB XID,PRESTRK ; PRESENT TRACK EQUAL TO ID
1860 011152 001004          BNE IS         ; NO GET NEW TRACK
1861 011154 113737 001200 011124    MOVB OD,TARGET  ; YES RETURN TO OD
1862 011162 000461          BR NEWTRK      ; NEWTRK INCREMENTS THE TRACK MOVED TO COUNTER
1863 ; AND RETURNS TO NEXT TRACK
1864 011164 005237 011124          IS:           INC TARGET    ; ADD 1 TO TARGET TRACK
1865 011170 000456          BR NEWTRK
1866
1867 ;*****
1868
1869 ; DECREMENT FROM ID TO OD
1870 ; S = 2
1871
1872 011172 105737 001201          SEQ2:          TSTB ID        ; FIRST TIME BIT SET
1873 011176 100007          BPL IS         ; NO GET NEXT TRACK
1874 011200 042737 100200 001200    BIC #100200,OD ; YES CLEAR FIRST TIME BITS
1875 011206 113737 011132 011124    MOVB XID,TARGET ; GO TO ID
1876 011214 000444          BR NEWTRK
1877 011216 005337 011124          IS:           DEC TARGET    ; MOVE TO NEXT TRACK
1878 011222 000441          BR NEWTRK
1879
1880 ;*****
1881
1882 ; INCREMENT THEN DECREMENT TRACKS
1883 ; S = 3
1884
1885 011224 105737 001200          SEQ3:          TSTB OD        ; CHECK FIRST TIME BIT
1886 011230 100007          BPL IS         ; NOT FIRST TIME THROUGH
1887 011232 042737 100200 001200    BIC #100200,OD ; CLEAR FIRST TIME BITS
1888 011240 005337 011122          DEC TRKCNTR
1889 011244 006337 011122          ASL TRKCNTR    ; RESET TRACK COUNTER TO DOUBLE THE COUNT
1890 011250 013700 011132          IS:           MOV XID,RO

```

```

1891 011254 163700 011130          SUB XOD,RO          ;GET DIFFERENCE BETWEEN ID AND OD
1892 011260 163700 011122          SUB TRKCNTR,RO     ;IS TRACK COUNTER HALF DONE
1893 011264 002342                   BGE SEQ2           ;YES GO DECREMENT TRACKS
1894 011266 000723                   BR SEQ1            ;NO CONTINUE INCREMENTING TRACKS
1895
1896
1897
1898
1899
1900
1901
1902 011270 042737 100200 001200 SEQ4:          BIC #100200,OD     ;CLEAR THE FIRST TIME BITS
1903 011276 123737 011126 001200          CMPB PRESTRK,OD   ;DID IT JUST DO OD
1904 011304 001404                   BEQ IDNEXT        ;YES
1905 011306 113737 001200 011124          MOVB OD,TARGET   ;NO GO TO OD TRACK
1906 011314 000404                   BR NEWTRK
1907 011316 113737 001201 011124 IDNEXT:       MOVB ID,TARGET    ;GO DO ID TRACK
1908 011324 000400                   BR NEWTRK
1909
1910
1911
1912
1913
1914
1915 011326 013705 011124          NEWTRK:           MOV TARGET,R5     ;GET TRACK ADDRESS
1916 011332 006305                   ASL R5            ;MULTIPLY BY 4 TO DOUBLE PRECISION
1917 011334 006305                   ASL R5            ;INTERLEAVE COUNTER ADDRESSES
1918 011336 062705 016420          ADD #HDMOVE,R5   ;ADD ON ADDRESS OF HEAD MOVE COUNTER
1919 011342 062715 000001          ADD #1,(R5)      ;INC COUNTER
1920 011346 005565 000002          ADC 2(R5)        ;ADD CARRY TO HIGH ORDER WORD
1921 011352 000207                   RTS PC            ;RETURN TO ACCESS NEXT TRACK
1922
1923
1924
1925
1926
1927
1928
1929 011354 105737 001201          SEQ5:            TSTB ID           ;TEST FIRST TIME BIT OF ID
1930 011360 100011                   BPL 15            ;ADJUST FOR ONE EXTRA MOVEMENT TO TRK OD
1931 011362 005237 011122          INC TRKCNTR      ;CLEAR FIRST TIME BIT OF ID
1932 011366 142737 000200 001201          BICB #200,ID     ;MOVE TO ID
1933 011374 113737 011132 011124          MOVB XID,TARGET
1934 011402 000751                   BR NEWTRK
1935 011404 105737 001200          15:             TSTB OD           ;TEST FIRST TIME BIT OF OD
1936 011410 100007                   BPL 25            ;CLEAR FIRST TIME BIT OF OD
1937 011412 142737 000200 001200          BICB #200,OD
1938 011420 113737 011130 011124          MOVB XOD,TARGET  ;MOVE TO OD
1939 011426 000737                   BR NEWTRK
1940 011430 132737 000001 011122 25:           BITB #1,TRKCNTR  ;TEST COUNTER FOR ODD OR EVEN
1941 011436 001006                   BNE 35            ;ODD MOVE TO NEW ID
1942 011440 005237 011130          INC XOD          ;EVEN ADD 1 TO OD
1943 011444 113737 011130 011124          MOVB XOD,TARGET
1944 011452 000405                   BR LASTTRK       ;SEE IF LAST TRACK IS BEING ACCESSED

```

\*\*\*\*\*

;BOUNCE BETWEEN ID AND OD ONLY  
; S = 4

\*\*\*\*\*

;INCREMENT THE "HEAD MOVED TO" COUNTERS AND RETURN  
;THROUGH HERE FROM ALL TRACK SEQUENCES

\*\*\*\*\*

;BOUNCE BETWEEN DECREASING ID AND INCREASING OD  
; S = 5



```

1945 011454 005337 011132 3S: DEC XID ;SUBTRACT 1 FROM ID
1946 011460 113737 011132 011124 MOVB XID,TARGET
1947 011466 123727 011122 000001 LASTTRK: CMPB TRKCNT, #1
1948 011474 001003 BNE XRETURN
1949 011476 113737 001200 011124 XRETURN: MOVB OD,TARGET ;THIS IS LAST TRACK RETURN TO OD
1950 011504 000710 BR NEWTRK
1951 ;*****
1952 ;
1953 ;STROBE BETWEEN OD AND DECREASING ID
1954 ; S = 6
1955 ;
1956 ;
1957 011506 105737 001200 SEQ6: TSTB OD ;CHECK FIRST TIME BIT
1958 011512 100007 BPL 1S ;NOT FIRST TIME
1959 011514 042737 100200 001200 BIC #100200,OD ;WAS FIRST TIME,CLEAR THE BITS
1960 011522 005337 011122 DEC TRKCNT ;RESET COUNTER TO DOUBLE THE NUMBER OF TRACKS
1961 011526 006337 011122 ASL TRKCNT
1962 011532 123737 011126 001200 1S: CMPB PRESTRK,OD ;WAS OD JUST ACCESSED
1963 011540 001006 BNE ODNEXT ;NO GO TO OD
1964 011542 113737 011132 011124 MOVB XID,TARGET ;DECREASING ID IS NEXT TRACK
1965 011550 005337 011132 DEC XID
1966 011554 000664 BR NEWTRK
1967 011556 113737 001200 011124 ODNEXT: MOVB OD,TARGET ;OD IS NEXT TRACK
1968 011564 000660 BR NEWTRK
1969
1970
1971 ;*****
1972 ;RANDOM SEQUENCING OF TRACKS
1973 ;FOR PROPER USE OF RANDOM TRACK SEQUENCING, THE OD/ID LIMITS
1974 ;SHOULD NOT BE SET FOR LESS THAN HALF THE TRACKS.
1975 ; S = 0 OR 7
1976 ;
1977 ;
1978 011566 042737 100200 001200 SEQ7: BIC #100200,OD ;CLEAR THE FIRST TIME BITS
1979 011574 004737 011666 JSR PC,RANGEN ;GET A RANDOM NUMBER
1980 011600 042737 177600 011760 BIC #177600,RANUM ;CLEAR ALL BUT LOW 7 BITS
1981 011606 123737 011760 011132 IDCOMP: CMPB RANUM,XID ;IS RANUM LARGER THAN ID ADDRESS
1982 011614 003404 BLE ODCOMP ;NO,RANUM IS LESS OR EQUAL TO ID
1983 011616 163737 011132 011760 SUB XID,RANUM ;YES,SUBTRACT ID FROM IT
1984 011624 000770 BR IDCOMP ;SEE IF RANUM IS NOW OK
1985 011626 123737 011760 011130 ODCOMP: CMPB RANUM,XOD ;IS RANUM SMALLER THAN OD ADDRESS
1986 011634 002004 BGE PRESCHK ;NO,RANUM IS GREATER OR EQUAL TO OD
1987 011636 063737 011130 011760 ADD XOD,RANUM ;YES,ADD OD TO RANUM
1988 011644 000770 BR ODCOMP ;SEE IF RANUM IS NOW OK
1989 011646 123737 011760 011126 PRESCHK: CMPB RANUM,PRESTRK ;IF RANUM EQUALS PRESENT TRACK
1990 011654 001744 BEQ SEQ7 ;GET ANOTHER RANDOM NUMBER
1991 011656 013737 011760 011124 MOV RANUM,TARGET ;RANUM OK PUT IT IN TARGET TRACK
1992 011664 000620 BR NEWTRK
1993
1994 011666 012700 000001 RANGEN: MOV #1,RO
1995 011672 063700 011754 ADD RAN1,RO
1996 011676 063700 011756 ADD RAN2,RO
1997 011702 042700 170000 BIC #170000,RO
1998 011706 000241 CLC

```

1999 011710 006100  
 2000 011712 006100  
 2001 011714 010037 011754  
 2002 011720 005000  
 2003 011722 013700 011756  
 2004 011726 006000  
 2005 011730 006000  
 2006 011732 063700 011754  
 2007 011736 042700 170000  
 2008 011742 010037 011756  
 2009 011746 010037 011760  
 2010 011752 000207

ROL RO  
 ROL RO  
 MOV RO,RAN1  
 CLR RO  
 MOV RAN2,RO  
 ROR RO  
 ROR RO  
 ADD RAN1,RO  
 BIC #170000,RO  
 MOV RO,RAN2  
 MOV RO,RANUM  
 RTS PC

2011 011754 000000  
 2012 011756 000000  
 2013 011760 000000  
 2014  
 2015  
 2016  
 2017  
 2018  
 2019

RAN1: 0  
 RAN2: 0  
 RANUM: 0

\*\*\*\*\*

;SECTOR INITIALIZATION AND SELECTION

2020 011762 005737 001202  
 2021 011766 001005  
 2022 011770 005237 001202  
 2023 011774 112737 000032 001203  
 2024 012002 113737 001203 012052  
 2025 012010 163737 001202 012052  
 2026 012016 005237 012052  
 2027 012022 105037 012053  
 2028 012026 113737 001202 012054  
 2029 012034 163737 012060 012054  
 2030  
 2031 012042 012737 000001 012056  
 2032 012050 000207  
 2033  
 2034 012052 000000  
 2035 012054 000000  
 2036 012056 000000  
 2037 012060 000003  
 2038

INITSECTOR: TST FIRST ;TEST FIRST AND LAST FOR 0  
 BNE 1\$ ;SECTORS SPECIFIED USE THEM  
 INC FIRST ;NONE SPECIFIED SET FIRST TO 1  
 MOV #32,LAST ;SET LAST TO MAXIMUM  
 1\$: MOVB LAST,SECCNTR ;SET UP SECTOR COUNTER  
 SUB FIRST,SECCNTR  
 INC SECCNTR  
 CLAB SECCNTR+1  
 MOVB FIRST,TSECTOR ;PUT FIRST SECTOR IN TARGET SECTOR  
 SUB THREE,TSECTOR ;SUB 3 FROM TSECTOR AS FIRST TIME THROUGH  
 ;IT GETS ADDED BACK ON.  
 ;SET INTERLEAVE OFFSET  
 MOV #1,INTLEAV  
 RTS PC

SECCNTR: 0  
 TSECTOR: 0  
 INTLEAV: 0  
 THREE: 3

2039 012062 063737 012060 012054  
 2040 012070 123737 001203 012054  
 2041 012076 002010  
 2042 012100 113737 001202 012054  
 2043 012106 063737 012056 012054  
 2044 012114 005237 012056  
 2045 012120 000207  
 2046  
 2047  
 2048  
 2049  
 2050  
 2051  
 2052

GETSECTOR: ADD THREE,TSECTOR ;ADD 3 FOR INTERLEAVING  
 CMPB LAST,TSECTOR  
 BGE 1\$ ;NEW SECTOR IS WITHIN LIMITS  
 MOVB FIRST,TSECTOR ;RESET TARGET SECTOR TO INTERLEAVE  
 ADD INTLEAV,TSECTOR ;ADD ON INTERLEAVE OFFSET VALUE  
 INC INTLEAV ;UP DATE THE OFFSET VALUE  
 1\$: RTS PC

\*\*\*\*\*

.SBTTL TYPE ROUTINE

;\*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.



2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106

012122 105737 012345  
012126 100002  
012130 000000  
012132 000407  
012134 010046  
012136 017600 000002  
012142 112046  
012144 001005  
012146 005726  
012150 012600  
012152 062716 000002  
012156 000002  
012160 122716 000011  
012164 001426  
012166 122716 000200  
012172 001004  
012174 005726  
012176 104400  
012200 012347  
012202 000757  
012204 004737 012266  
012210 123726 012344  
012214 001352  
012216 013746 012342  
012222 105366 000001  
012226 002770  
012230 004737 012266  
012234 105337 012332  
012240 000770  
  
012242 112716 000040  
012246 004737 012266  
012252 132737 000007 012332  
012260 001372  
012262 005726  
012264 000726  
012266 105777 000044  
012272 100375

```

;#THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
;#NOTE1: SNULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
;#NOTE2: SFILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
;#NOTE3: SFILLC CONTAINS THE CHARACTER TO FILL AFTER.
;#
;#CALL:
;#1) USING A TRAP INSTRUCTION
;# TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
;#OR
;# TYPE
;# MESADR
;#
STYPE: TSTB STPFLG ;; IS THERE A TERMINAL?
BPL 15 ;; BR IF YES
HALT ;; HALT HERE IF NO TERMINAL
BR 35 LEAVE
1S: MOV RO, -(SP) ;; SAVE RO
MOV 32(SP), RO ;; GET ADDRESS OF ASCIZ STRING
2S: MOVB (RO)+, -(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
BNE 45 ;; BR IF IT ISN'T THE TERMINATOR
TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
60S: MOV (SP)+, RO ;; RESTORE RO
3S: ADD #2, (SP) ;; ADJUST RETURN PC
RTI ;; RETURN
4S: CMPB #THT, (SP) ;; BRANCH IF <HT>
BEQ 85
CMPB #TCRLF, (SP) ;; BRANCH IF NOT <CRLF>
BNE 55
TST (SP)+ ;; POP <CR><LF> EQUIV
TYPE ;; TYPE A CR AND LF
BR 25 ;; GET NEXT CHARACTER
5S: JSR PC, STYPEC ;; GO TYPE THIS CHARACTER
6S: CMPB SFILLC, (SP)+ ;; IS IT TIME FOR FILLER CHARS.?
BNE 25 ;; IF NO GO GET NEXT CHAR.
MOV SNULL, -(SP) ;; GET # OF FILLER CHARS. NEEDED
AND THE NULL CHAR.
7S: DECB 1(SP) ;; DOES A NULL NEED TO BE TYPED?
BLT 6S ;; BR IF NO--GO POP THE NULL OFF OF STACK
JSR PC, STYPEC ;; GO TYPE A NULL
DECB SCHARCNT ;; DO NOT COUNT AS A COUNT
BR 7S ;; LOOP

;HORIZONTAL TAB PROCESSOR
8S: MOVB #40, (SP) ;; REPLACE TAB WITH SPACE
9S: JSR PC, STYPEC ;; TYPE A SPACE
BITB #7, SCHARCNT ;; BRANCH IF NOT AT
BNE 9S ;; TAB STOP
TST (SP)+ ;; POP SPACE OFF STACK
BR 25 ;; GET NEXT CHARACTER
STYPEC: TSTB STPS ;; WAIT UNTIL PRINTER IS READY
BPL STYPEC

```

```

2107 012274 116677 000002 000036      MOVB 2(SP),STPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
2108 012302 122766 000015 000002      CMPB #15,2(SP)      ;;BRANCH IF
2109 012310 001003                      BNE 15              ;;NOT <CR>
2110 012312 105037 012332      CLAB $CHARCNT      ;;
2111 012316 000406                      BR STYPEX          ;;EXIT
2112 012320 122766 000012 000002 1S:  CMPB #12,2(SP)      ;;BRANCH IF
2113 012326 002002                      BGE STYPEX        ;;<LF>
2114 012330 105227                      INCB (PC)+        ;;INC SPACE
2115 012332 000000                      SCHARCNT: .WORD 0 ;;COUNT
2116 012334 000207                      STYPEX: RTS PC
2117                      ;; EQUATES
2118                      THT=11
2119                      TCRLF=200
2120
2121 012336 177564      STPS: .WORD 177564  ;;TTY PRINTER STATUS REG. ADDRESS
2122 012340 177566      STPB: .WORD 177566  ;;TTY PRINTER BUFFER REG. ADDRESS
2123 012342 000      $NULL: .BYTE 0     ;;CONTAINS NULL CHARACTER FOR FILLS
2124 012343 002      $FILLS: .BYTE 2    ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
2125 012344 012      $FILLC: .BYTE 12   ;;INSERT FILL CHARS. AFTER A "LINE FEED"
2126 012345 000      STPFLG: .BYTE 0    ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
2127 012346 077      $QUES: .ASCII "?"  ;;QUESTION MARK
2128 012347 015 000  $CRLF: .ASCIIZ <15> ;;CARRIAGE RETURN
2129 012351 012 000  $LF: .ASCIIZ <12>  ;;LINEFEED
2130                      .EVEN
2131                      ;*****
2132                      .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
2133
2134                      ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
2135                      ;*OCTAL (ASCII) NUMBER AND TYPE IT.
2136                      ;*STYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
2137                      ;*CALL:
2138                      ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2139                      ;*      TYPOS      ;;CALL FOR TYPEOUT
2140                      ;*      .BYTE  N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2141                      ;*      .BYTE  M      ;;M=1 OR 0
2142                      ;*                      ;;1=TYPE LEADING ZEROS
2143                      ;*                      ;;0=SUPPRESS LEADING ZEROS
2144                      ;*
2145                      ;*STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2146                      ;*STYPOS OR STYPOC
2147                      ;*CALL:
2148                      ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2149                      ;*      TYPON      ;;CALL FOR TYPEOUT
2150                      ;*
2151                      ;*STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2152                      ;*CALL:
2153                      ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2154                      ;*      TYPOC      ;;CALL FOR TYPEOUT
2155
2157 012354 017646 000000 012577  STYPOS: MOV 2(SP),-(SP)  ;;PICKUP THE MODE
2158 012360 116637 000001          MOVB 1(SP),SOFILL  ;;LOAD ZERO FILL SWITCH
2159 012366 112637 012601          MOVB (SP)+,SOMODE+1 ;;NUMBER OF DIGITS TO TYPE
2160 012372 062716 000002          ADD #2,(SP)      ;;ADJUST RETURN ADDRESS

```



```

2161 012376 000406          BR          STYPON
2162 012400 112737 000001 012577 STYPOC: MOVB   #1, SOFILL
2163 012406 112737 000006 012601          MOVB   #6, SOMODE+1
2164 012414 112737 000005 012576 STYPON: MOVB   #5, SOCNT
2165 012422 010346          MOV    R3, -(SP)
2166 012424 010446          MOV    R4, -(SP)
2167 012426 010546          MOV    R5, -(SP)
2168 012430 113704 012601          MOVB   SOMODE+1, R4
2169 012434 005404          NEG    R4
2170 012436 062704 000006          ADD    #6, R4
2171 012442 110437 012600          MOVB   R4, SOMODE
2172 012446 113704 012577          MOVB   SOFILL, R4
2173 012452 016605 000012          MOV    12(SP), R5
2174 012456 005003          CLR    R3
2175 012460 006105          1S:   ROL    R5
2176 012462 000404          BR     3S
2177 012464 006105          2S:   ROL    R5
2178 012466 006105          ROL    R5
2179 012470 006105          ROL    R5
2180 012472 010503          MOV    R5, R3
2181 012474 006103          3S:   ROL    R3
2182 012476 105337 012600          DECB  SOMODE
2183 012502 100016          BPL   7S
2184 012504 042703 177770          BIC   #177770, R3
2185 012510 001002          BNE   4S
2186 012512 005704          TST   R4
2187 012514 001403          BEQ   5S
2188 012516 005204          4S:   INC    R4
2189 012520 052703 000060          BIS   #'0, R3
2190 012524 052703 000040          5S:   BIS   #' , R3
2191 012530 110337 012574          MOVB  R3, #5
2192 012534 104400 012574          TYPE  #5
2193 012540 105337 012576          7S:   DECB  SOCNT
2194 012544 003347          BGT   2S
2195 012546 002402          BLT   6S
2196 012550 005204          INC   R4
2197 012552 000744          BR    2S
2198 012554 012605          6S:   MOV   (SP)+, R5
2199 012556 012604          MOV   (SP)+, R4
2200 012560 012603          MOV   (SP)+, R3
2201 012562 016666 000002 000004          MOV   2(SP), 4(SP)
2202 012570 012616          MOV   (SP)+, (SP)
2203 012572 000002          RTI
2204 012574 000          8S:   .BYTE 0
2205 012575 000          .BYTE 0
2206 012576 000          SOCNT: .BYTE 0
2207 012577 000          SOFILL: .BYTE 0
2208 012600 000000          SOMODE: .WORD 0
2209
2210
2211          .SBTTL SAVE AND RESTORE R0-R5 ROUTINES
2212
2213          ;#SAVE R0-R5
2214          ;#CALL:

```

```

::: SET THE ZERO FILL SWITCH
::: SET FOR SIX(6) DIGITS
::: SET THE ITERATION COUNT
::: SAVE R3
::: SAVE R4
::: SAVE R5
::: GET THE NUMBER OF DIGITS TO TYPE
::: SUBTRACT IT FOR MAX. ALLOWED
::: SAVE IT FOR USE
::: GET THE ZERO FILL SWITCH
::: PICKUP THE INPUT NUMBER
::: CLEAR THE OUTPUT WORD
::: ROTATE MSB INTO "C"
::: GO DO MSB
::: FORM THIS DIGIT
::: GET LSB OF THIS DIGIT
::: TYPE THIS DIGIT?
::: BR IF NO
::: GET RID OF JUNK
::: TEST FOR 0
::: SUPPRESS THIS 0?
::: BR IF YES
::: DON'T SUPPRESS ANYMORE 0'S
::: MAKE THIS DIGIT ASCII
::: MAKE ASCII IF NOT ALREADY
::: SAVE FOR TYPING
::: GO TYPE THIS DIGIT
::: COUNT BY 1
::: BR IF MORE TO DO
::: BR IF DONE
::: INSURE LAST DIGIT ISN'T A BLANK
::: GO DO THE LAST DIGIT
::: RESTORE R5
::: RESTORE R4
::: RESTORE R3
::: SET THE STACK FOR RETURNING
::: RETURN
::: STORAGE FOR ASCII DIGIT
::: TERMINATOR FOR TYPE ROUTINE
::: OCTAL DIGIT COUNTER
::: ZERO FILL SWITCH
::: NUMBER OF DIGITS TO TYPE

```

```

2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228 012602
2229 012602 010046
2230 012604 010146
2231 012606 010246
2232 012610 010346
2233 012612 010446
2234 012614 010546
2235 012616 016646 000022
2236 012622 016646 000022
2237 012626 016646 000022
2238 012632 016646 000022
2239 012636 000002
2240
2241
2242
2243
2244 012640
2245 012640 012666 000022
2246 012644 012666 000022
2247 012650 012666 000022
2248 012654 012666 000022
2249 012660 012605
2250 012662 012604
2251 012664 012603
2252 012666 012602
2253 012670 012601
2254 012672 012600
2255 012674 000002
2256
2257
2258
2259
2260
2261
2262
2263
2264 012676 010046
2265 012700 016600 000002
2266 012704 005740
2267 012706 111000
2268 012710 006300

```

```

;# SAVREG
;#UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
;#
;#TOP---(+16)
;# +2---(+18)
;# +4---R5
;# +6---R4
;# +8---R3
;#+10---R2
;#+12---R1
;#+14---R0

```

```

$SAVREG:
MOV RO,-(SP) ;: PUSH RO ON STACK
MOV R1,-(SP) ;: PUSH R1 ON STACK
MOV R2,-(SP) ;: PUSH R2 ON STACK
MOV R3,-(SP) ;: PUSH R3 ON STACK
MOV R4,-(SP) ;: PUSH R4 ON STACK
MOV R5,-(SP) ;: PUSH R5 ON STACK
MOV 22(SP),-(SP) ;: SAVE PS OF MAIN FLOW
MOV 22(SP),-(SP) ;: SAVE PC OF MAIN FLOW
MOV 22(SP),-(SP) ;: SAVE PS OF CALL
MOV 22(SP),-(SP) ;: SAVE PC OF CALL
RTI

```

```

;#RESTORE RO-R5
;#CALL:
;# RESREG
$RESREG:
MOV (SP)+,22(SP) ;: RESTORE PC OF CALL
MOV (SP)+,22(SP) ;: RESTORE PS OF CALL
MOV (SP)+,22(SP) ;: RESTORE PC OF MAIN FLOW
MOV (SP)+,22(SP) ;: RESTORE PS OF MAIN FLOW
MOV (SP)+,R5 ;: POP STACK INTO R5
MOV (SP)+,R4 ;: POP STACK INTO R4
MOV (SP)+,R3 ;: POP STACK INTO R3
MOV (SP)+,R2 ;: POP STACK INTO R2
MOV (SP)+,R1 ;: POP STACK INTO R1
MOV (SP)+,R0 ;: POP STACK INTO R0
RTI

```

\*\*\*\*\*

.SBTTL TRAP DECODER

```

;#THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
;#AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
;#OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
;#GO TO THAT ROUTINE.

```

```

$TRAP: MOV RO,-(SP) ;: SAVE RO
MOV 2(SP),RO ;: GET TRAP ADDRESS
TST -(RO) ;: BACKUP BY 2
MOVB (RO),RO ;: GET RIGHT BYTE OF TRAP
ASL RO ;: POSITION FOR INDEXING

```



2269 012712 016000 012720  
 2270 012716 000200  
 2271  
 2272  
 2273  
 2274  
 2275  
 2276  
 2277  
 2278  
 2279  
 2280 012720  
 2281 012720 012122  
 2282 012722 012400  
 2283 012724 012354  
 2284 012726 012414  
 2285 012730 012602  
 2286 012732 012640  
 2287  
 2288  
 2289  
 2290  
 2291  
 2292 012734 012737 013062 000024  
 2293 012742 012737 000340 000026  
 2294 012750 010046  
 2295 012752 010146  
 2296 012754 010246  
 2297 012756 010346  
 2298 012760 010446  
 2299 012762 010546  
 2300 012764 010637 013066  
 2301 012770 012737 013002 000024  
 2302 012776 000000  
 2303 013000 000776  
 2304  
 2305  
 2306 013002 013706 013066  
 2307 013006 005037 013066  
 2308 013012 005237 013066  
 2309 013016 001375  
 2310 013020 012605  
 2311 013022 012604  
 2312 013024 012603  
 2313 013026 012602  
 2314 013030 012601  
 2315 013032 012600  
 2316 013034 012737 012734 000024  
 2317 013042 012737 000340 000026  
 2318 013050 104400  
 2319 013052 013070  
 2320 013054 012716  
 2321 013056 002532  
 2322 013060 000002

```

MOV STRPAD(RO),RO ;;INDEX TO TABLE
RTS RO ;;GO TO ROUTINE

.SBTTL TRAP TABLE
; *THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
; *BY THE "TRAP" INSTRUCTION.
:
: ROUTINE
: -----
STRPAD:
STYPER ;;CALL=TYPE TRAP+0(104400) TTY TYPEOUT ROUTINE
STYPOC ;;CALL=TYPOC TRAP+1(104401) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
STYPOS ;;CALL=TYPOS TRAP+2(104402) TYPE OCTAL NUMBER (NO LEADING ZEROS)
STYPON ;;CALL=TYPON TRAP+3(104403) TYPE OCTAL NUMBER (AS PER LAST CALL)
SSAVREG ;;CALL=SAVREG TRAP+4(104404) SAVE RO-R5 ROUTINE
SRESREG ;;CALL=RESREG TRAP+5(104405) RESTORE RO-R5 ROUTINE
;*****

.SBTTL POWER DOWN AND UP ROUTINES

:POWER DOWN ROUTINE
SPWRDN: MOV $SILLUP, @#PWRVEC ;;SET FOR FAST UP
MOV #340, @#PWRVEC+2 ;;PRIO:7
MOV RO, -(SP) ;;PUSH RO ON STACK
MOV R1, -(SP) ;;PUSH R1 ON STACK
MOV R2, -(SP) ;;PUSH R2 ON STACK
MOV R3, -(SP) ;;PUSH R3 ON STACK
MOV R4, -(SP) ;;PUSH R4 ON STACK
MOV R5, -(SP) ;;PUSH R5 ON STACK
MOV SP, SSAVR6 ;;SAVE SP
MOV #SPWRUP, @#PWRVEC ;;SET UP VECTOR
HALT
BR -2 ;;HANG UP

:POWER UP ROUTINE
SPWRUP: MOV SSAVR6, SP ;;GET SP
CLR SSAVR6 ;;WAIT LOOP FOR THE TTY
IS: INC SSAVR6 ;;WAIT FOR THE INC
BNE IS OF WORD
MOV (SP)+, R5 ;;POP STACK INTO R5
MOV (SP)+, R4 ;;POP STACK INTO R4
MOV (SP)+, R3 ;;POP STACK INTO R3
MOV (SP)+, R2 ;;POP STACK INTO R2
MOV (SP)+, R1 ;;POP STACK INTO R1
MOV (SP)+, RO ;;POP STACK INTO RO
MOV #SPWRDN, @#PWRVEC ;;SET UP THE POWER DOWN VECTOR
MOV #340, @#PWRVEC+2 ;;PRIO:7
TYPE ;;REPORT THE POWER FAILURE
SPWRMG: .WORD SPOWER ;;POWER FAIL MESSAGE POINTER
MOV (PC)+, (SP) ;;RESTART AT RESTART
SPWRAD: .WORD RESTART ;;RESTART ADDRESS
RTI

```



```

2323 013062 000000          $ILLUP: HALT          ;; THE POWER UP SEQUENCE WAS STARTED
2324 013064 000776          BR          .-2        ;; BEFORE THE POWER DOWN WAS COMPLETE
2325 013066 000000          $SAVR6: 0           ;; PUT THE SP HERE
2326 013070 005015 047520 042527 $POWER: .ASCIZ <15><12>"POWER"
2327 013076 000122
2328
2329          .EVEN
2330          ;*****
2331          .SBTTL SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE
2332
2333          ;*THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
2334          ;*UNSIGNED DECIMAL ASCIZ NUMBER.
2335          ;*CALL
2336          ;*      MOV      NUMBER, -(SP)      ;; PUT BINARY NUMBER ON THE STACK
2337          ;*      JSR      PC, @#$SDB2D      ;; CALL
2338          ;*      RETURN     ;; ADDRESS OF THE 1ST ASCIZ CHAR. IS ON THE STACK
2339
2340
2341 013100 016637 000002 013124 $SDB2D: MOV      2(SP), 1$      ;; SAVE BINARY NUMBER
2342 013106 012746 013124          MOV      #1$, -(SP)      ;; SET POINTER
2343 013112 004737 013130          JSR      PC, @#$SDB2D      ;; CALL DOUBLE LENGTH CONVERT
2344 013116 012666 000002          MOV      (SP)+, 2(SP)      ;; PICKUP POINTER
2345 013122 000207          RTS      PC              ;; RETURN
2346 013124 000000 000000          1$:      .WORD      0, 0
2347          ;*****
2348
2349          .SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
2350
2351          ;*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
2352          ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
2353          ;*POSITIVE.
2354          ;*CALL
2355          ;*      MOV      #PNTR, -(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
2356          ;*      JSR      PC, @#$SDB2D      ;; CALL
2357          ;*      RETURN     ;; THE FIRST ADDRESS OF ASCIZ
2358          ;*                      ;; IS ON THE STACK
2359
2360
2361 013130 104404          SDB2D: SAVREG          ;; SAVE REGISTERS
2362 013132 016602 000002          MOV      2(SP), R2      ;; PICKUP THE DATA POINTER
2363 013136 012700 013310          MOV      #$DECL, R0     ;; GET ADDRESS OF "DECL" STRING
2364 013142 010066 000002          MOV      R0, 2(SP)      ;; PUT ADDRESS OF ASCIZ STRING ON STACK
2365 013146 012201          MOV      (R2)+, R1      ;; PICKUP THE BINARY NUMBER
2366 013150 012202          MOV      (R2)+, R2
2367 013152 012737 000012 013226          MOV      #10, R5        ;; SET UP TO DO 10 CONVERSIONS
2368 013160 012704 013240          MOV      #$TNPWR, R4     ;; ADDRESS OF TEN POWER
2369 013164 012705 013242          MOV      #$TNPWR+2, R5
2370 013170 005003          1$:      CLR      R3              ;; CLEAR PARTIAL
2371 013172 161401          2$:      SUB      (R4), R1      ;; SUBTRACT TEN POWER
2372 013174 005602          SBC      R2
2373 013176 161502          SUB      (R5), R2
2374 013200 002402          BLT     3$              ;; BR IF TEN POWER TO LARGE
2375 013202 005203          INC     R3              ;; ADD 1 TO PARTIAL
2376 013204 000772          BR      2$              ;; LOOP
    
```



2377	013206	062401	
2378	013210	005502	
2379	013212	062402	
2380	013214	022525	
2381	013216	052703	000060
2382	013222	110320	
2383	013224	005327	
2384	013226	000000	
2385	013230	001357	
2386	013232	105020	
2387	013234	104405	
2388	013236	000207	
2389	013240	145000	
2390	013242	035632	
2391	013244	160400	
2392	013246	002765	
2393	013250	113200	
2394	013252	000230	
2395	013254	041100	
2396	013256	000017	
2397	013260	103240	
2398	013262	000001	
2399	013264	023420	
2400	013266	000000	
2401	013270	001750	
2402	013272	000000	
2403	013274	000144	
2404	013276	000000	
2405	013300	000012	
2406	013302	000000	
2407	013304	000001	
2408	013306	000000	
2409	013310	000014	
2410			
2411			
2412			
2413			
2414			
2415			
2416			
2417			
2418	013324	010046	
2419	013326	016600	000004
2420	013332	010037	013364
2421	013336	105710	
2422	013340	001406	
2423	013342	122710	000060
2424	013346	001005	
2425	013350	112720	000040
2426	013354	000770	
2427	013356	112740	000060
2428	013362	104400	
2429	013364	000000	
2430	013366	012600	

```

3S:  ADD (R4)+,R1      ;;RESTORE SUBTRACTED VALUE
      ADC R2
      ADD (R4)+,R2
      CMP (R5)+,(R5)+  ;;MOVE TO NEXT TEN POWER
      BIS #'0,R3      ;;CHANGE PARTIAL TO ASCII
      MOVB R3,(R0)+   ;;SAVE IT
      DEC (PC)+       ;;DONE?

4S:  .WORD 0
      BNE IS         ;;BR IF NO
      CLRB (R0)+     ;;TERMINATOR
      RESREG        ;;RESTORE REGISTERS
      RTS PC        ;;RETURN

STNPHR: 145000      ;;1.0E09
        35632
        160400     ;;1.0E08
        2765
        113200     ;;1.0E07
        230
        041100     ;;1.0E06
        17
        103240     ;;1.0E05
        1
        23420      ;;1.0E04
        0
        1750       ;;1.0E03
        0
        144        ;;1.0E02
        0
        12         ;;1.0E01
        0
        1          ;;1.0E00
        0

SDECVL: .BLKB 12.      ;;RESERVE STORAGE FOR ASCIZ STRING

```

```

*****
:TYPE NUMERICAL ASCIZ STRING,RIGHT JUSTIFIED
:REPLACING LEADING ZEROS WITH SPACES.
:FIRST ADDRESS OF ASCIZ STRING MUST BE ON TOP OF THE STACK

```

```

RTJUST: MOV RO,-(SP)   ;SAVE RO
        MOV 4(SP),RO  ;PICK UP ADDRESS OF ASCIZ STRING
        MOV RO,3S    ;SAVE ADDRESS FOR TYPE OUT
1S:     TSTB (RO)     ;IS THIS THE TERMINATOR
        BEQ 2S       ;IF YES TYPE IT OUT
        CMPB #'0,(RO) ;IS IT A ZERO
        BNE 4S       ;IF NO GO PRINT IT
        MOVB #' ,(RO)+ ;IF YES REPLACE IT WITH A SPACE
        BR 1S        ;TEST NEXT CHAR.
2S:     MOVB #'0,-(RO) ;STRING OFF ALL ZEROS,PUT BACK THE LAST ONE
4S:     TYPE          ;TYPE THE STRING
3S:     OPEN
        MOV (SP)+,RO ;RESTORE RO

```

```

2431 013370 012616          MOV (SP)+,(SP)      ;RESTORE THE STACK
2432 013372 000207          RTS PC              ;RETURN
2433
2434
2435
2436
2437 013374 012546          SGLDEC:           MOV (RS)+,-(SP)    ;PUT NUMBER TO BE TYPED ON STACK
2438 013376 004737 013100    JSR PC,@#S82D     ;CONVERT NUMBER TO DECIMAL
2439 013402 062716 000005    ADD #5,(SP)       ;MOVE ADDRESS OF ASCII STRING OVER BY 5 BYTES
2440
2441 013406 004737 013324    JSR PC,RTJUST     ;TO TYPE SINGLE DECIMAL NUMBER
2442 013412 000205          RTS RS            ;TYPE THE DECIMAL NUMBER
2443
2444
2445 013414 047125 054105 042520 MUXDD:           .ASCII "UNEXPECTED D D MARK"
2446 013422 052103 042105 042040
2447 013430 042040 046440 051101
2448 013436 000113
2449
2450 013440 020104 020104 040515 MDDMIS:         .ASCII "D D MARK MISSING"
2451 013446 045522 046440 051511
2452 013454 044523 043516 000
2453
2454
2455 013461 004 052101 026101 MDERHDR:        .ASCII "DATA, NO STATUS ERROR"
2456 013466 047040 020117 052123
2457 013474 052101 051525 042440
2458 013502 051122 051117 000
2459
2460 013507 0040 047117 052040 MTRK:           .ASCII " ON TRACK"
2461 013514 040522 045503 000
2462
2463
2464 013521 0040 043040 047522 MPRES:           .ASCII " FROM TRACK"
2465 013526 020115 051124 041501
2466 013534 000113
2467
2468 013536 027440 051440 041505 MSECT:          .ASCII " / SECTOR"
2469 013544 047524 000122
2470
2471 013550 005015 041040 052131 MCOLMUN:       .ASCII <15><12>" BYTE BAD GOOD"<15><12>
2472 013556 020105 041040 042101
2473 013564 020040 047507 042117
2474 013572 005015 000
2475
2476 013575 0015 041412 052501 DLOAD: .ASCII <15><12>"CAUTION - IF YOU DESIRE TO TEST UNIT 0"
2477 013602 044524 047117 026440
2478 013610 044440 020106 047531
2479 013616 020125 042504 044523
2480 013624 042522 052040 020117
2481 013632 042524 052123 052440
2482 013640 044516 020124 060
2483 013645 0015 051012 050105 .ASCII <15><12>"REPLACE LOAD MEDIUM WITH A SCRATCH DISKETTE"
2484 013652 040514 042503 046040
2485 013660 040517 020104 042515

```



013666	044504	046525	053440		
013674	052111	020110	020101		
013702	041523	040522	041524		
013710	020110	044504	045523		
013716	052105	042524			
013722	005015	044124	047105		.ASCIZ <15><12>"THEN PRESS CONTINUE"
013730	050040	042522	051523		
013736	041440	047117	044524		
013744	052516	000105			
013750	047516	042040	044522	MNODRY:	.ASCIZ "NO DRIVES READY"<15><12>
013756	042526	020123	042522		
013764	042101	006531	000012		
013772	020040	020040	020040	TAB:	.ASCIZ <40><40><40><40><40><40>
014000	000				
014001	040	000040		DBLSP:	.ASCIZ <40><40>
014004	005015	000		MCRLF:	.ASCIZ <15><12>
014007	104	000		MEOP:	.ASCIZ "D"
014011	055	000		MEREOP:	.ASCIZ "--"
014013	007	000		MABELL:	.ASCIZ <07>
014015	105	051122	051117	MERHEADER:	.ASCIZ "ERROR CONDITIONS "
014022	041440	047117	044504		
014030	044524	047117	020123		
014036	000040				
014040	047125	052111	030040	MUNIT0:	.ASCIZ "UNIT 0 "
014046	000040				
014050	047125	052111	030440	MUNIT1:	.ASCIZ "UNIT 1 "
014056	000040				
014060	040510	020123	042502	MDELET:	.ASCIZ "HAS BEEN DELETED."
014066	047105	042040	046105		
014074	052105	042105	000056		
014102	044440	052116	053040	MINTVEC:	.ASCIZ " INT VECTOR = "
014110	041505	047524	020122		
014116	020075	000			
014121	122	041530	020123	MRXCS:	.ASCIZ "RXCS = "
014126	020075	000			
014131	040	054122	041104	MRXDB:	.ASCIZ " RXDB = "
014136	036440	000040			
014142	052123	052101	051525	MASTAT:	.ASCIZ "STATUS A = "
014150	040440	036440	000040		

014156	052123	052101	051525	MBSTAT:	.ASCIZ "STATUS B = "
014164	041040	036440	000040		
014172	005015	000012		DBLLF:	.ASCIZ <15><12><12>
014176	036520	000040		MPAT:	.ASCIZ "P= "
014202	036524	000040		MTEST:	.ASCIZ "T= "
014206	036523	000040		MSEQ:	.ASCIZ "S= "
014212	047516	044440	052116	MINTER:	.ASCIZ "NO INTERRUPT AT DONE ERROR"
014220	051105	050125	020124		
014226	052101	042040	047117		
014234	020105	051105	047522		
014242	000122				
014244	047125	047113	053517	MUKNINT:	.ASCIZ "UNKNOWN INTERRUPT"
014252	020116	047111	042524		
014260	052522	052120	000		
014265	124	052117	046101	MERCT:	.ASCIZ "TOTAL READ CHECK ERRORS = "
014272	051040	040505	020104		
014300	044103	041505	020113		
014306	051105	047522	051522		
014314	036440	000040			
014320	044506	046114	052502	MFIL:	.ASCIZ "FILLBUFFER "
014326	043106	051105	000040		
014334	046505	052120	041131	MEMPTY:	.ASCIZ "EMPTYBUFFER "
014342	043125	042506	020122		
014350	000				
014351	104	044522	042526	MICON:	.ASCIZ "DRIVE(S) "
014356	051450	020051	000040		
014364	042524	052123	044040	MHUNG:	.ASCIZ "TEST HUNG"<15><12>
014372	047125	006507	000012		
014400	047516	051516	040524	MNONSTD:	.ASCIZ "NONSTANDARD TRACK / SECTOR LIMITS"
014406	042116	051101	020104		
014414	051124	041501	020113		
014422	020057	042523	052103		
014430	051117	046040	046511		
014436	052111	000123			
014442	042117	000075		MOD:	.ASCIZ "OD="
014446	042111	000075		MID:	.ASCIZ "ID="
014452	044506	051522	036524	MFIRST:	.ASCIZ "FIRST="
014460	000				



014461	114	051501	036524	MLAST:	.ASCIZ "LAST="
014466	000				
014467	111	044516	027124	MINIT1:	.ASCIZ "INIT.DONE NOT SET ERROR" <15><12>
014474	047504	042516	047040		
014502	052117	051440	052105		
014510	042440	051122	051117		
014516	005015	000			
014521	111	044516	027124	MINIT2:	.ASCIZ "INIT.DONE SET ERROR" <15><12>
014526	047504	042516	051440		
014534	052105	042440	051122		
014542	051117	005015	000		
014547	104	042040	042440	MDDER:	.ASCIZ "D D ERROR"
014554	051122	051117	000		
014561	122	041505	053117	MREC:	.ASCIZ "RECOVERABLE "
014566	051105	041101	042514		
014574	000040				
014576	051103	020103	051105	MBADCR:	.ASCIZ "CRC ERROR NO DATA ERROR"
014604	047522	020122	047516		
014612	042040	052101	020101		
014620	051105	047522	000122		
014626	042522	042101	000040	MREAD:	.ASCIZ "READ "
014634	040510	052114	032040	MHALT4:	.ASCIZ "HALT 4" <15><12>
014642	005015	000			
014645	110	042071	020124	MHALT3:	.ASCIZ "HALT 3" <15><12>
014652	006463	00012			
014656	040510	052114	030440	MHALT11:	.ASCIZ "HALT 11" <15><12>
014664	006461	000012			
014670	040504	040524	041440	MCRC:	.ASCIZ "DATA CRC ERROR"
014676	041522	042440	051122		
014704	051117	000			
014707	040	047125	042522	MUNREC:	.ASCIZ " UNRECOVERABLE "
014714	047503	042526	040522		
014722	046102	020105	000		
014727	123	042505	020113	MSEEK:	.ASCIZ "SEEK ERROR"
014734	051105	047522	000122		
014742	051127	052111	020105	MWRITE:	.ASCIZ "WRITE "
014750	000				
014751	120	051101	052111	MPAR:	.ASCIZ "PARITY ERROR"

2647	014756	020131	051105	047522		
2648	014764	000122				
2649						
2650	014766	051105	047522	020122	MNOFLAG:	.ASCIZ "ERROR FLAG ERROR"
2651	014774	046106	043501	042440		
2652	015002	051122	051117	000		
2653						
2654	015007	122	030530	020061	MDUMP:	.ASCIZ "RX11 DUMP"
2655	015014	052504	050115	000		
2656						
2657	015021	105	051122	051117	MERS:	.ASCIZ "ERRORS"
2658	015026	000123				
2659						
2660	015030	044123	051117	020124	MSHORT:	.ASCIZ "SHORT DUMP"
2661	015036	052504	050115	000		
2662						
2663	015043	122	051505	040524	MRESTART:	.ASCIZ "RESTARTS = "
2664	015050	052122	020123	020075		
2665	015056	000				
2666						
2667	015057	120	051501	020123	MNOPAS:	.ASCIZ "PASS ABORTED"
2668	015064	041101	051117	042524		
2669	015072	000104				
2670						
2671	015074	040520	051523	051505	MPASS:	.ASCIZ "PASSES = "
2672	015102	036440	000040			
2673						
2674	015106	051127	052111	042524	MWRTE:	.ASCIZ "WRITTEN "
2675	015114	020116	000			
2676						
2677	015117	057	000040		MSLASH:	.ASCIZ "/"
2678						
2679	015122	000040			SPACE:	.ASCIZ "<40>"
2680						
2681	015124	051105	047522	051522	MCODES:	.ASCIZ "ERRORS PER ERROR CODES"<15><12>
2682	015132	050040	051105	042440		
2683	015140	051122	051117	041440		
2684	015146	042117	051505	005015		
2685	015154	000				
2686						
2687	015155	124	040522	045503	MTKLOG:	.ASCIZ "TRACK ACCESSED MOVED TO UNIT 0 UNIT 1 ERRORS"<15><12>
2688	015162	020040	041501	042503		
2689	015170	051523	042105	020040		
2690	015176	046440	053117	042105		
2691	015204	052040	020117	020040		
2692	015212	052440	044516	020124		
2693	015220	020060	052440	044516		
2694	015226	020124	020061	051105		
2695	015234	047522	051522	005015		
2696	015242	000				
2697						
2698	015243	050	047516	042516	MNONE:	.ASCIZ "(NONE)"<15><12>
2699	015250	006451	000012			
2700						



2701	015254	047507	042117	000	MG00D:	.ASCIZ "GOOD"
2702						
2703	015261	040	041440	042510	MSUM:	.ASCIZ " CHECK SUM "
2704	015266	045503	051440	046525		
2705	015274	000040				
2706						
2707	015276	005015	054122	030461	MRX11:	.ASCIZ <15><12>"RX11 / RXV11"
2708	015304	027440	051040	053130		
2709	015312	030461	000			
2710						
2711	015315	015	046412	044501	MREV:	.ASCIZ <15><12> "MAINDEC-11-DZRXA-D" <15><12>
2712	015322	042116	041505	030455		
2713	015330	026461	055104	054122		
2714	015336	026501	006504	000012		

.EVEN

\*\*\*\*\*

; THE FOLLOWING LOCATIONS ARE USED FOR DATA STORAGE, RETRY COUNTERS  
; ACCESS COUNTERS ETC.

2723 015344 000200

BUFADR: .BLKB 200

; RETRY COUNTERS

2726	015544	000012
2727	015546	000012
2728	015550	000012
2729	015552	000012
2730	015554	000012
2731	015556	000012
2732	015560	000012
2733	015562	000012

RDRETRY:	10.
WTRETRY:	10.
DDRETRY:	10.
DATARETRY:	10.
P2RETRY:	10.
PRETRY:	10.
CRETRY:	10.
SRETRY:	10.

; FOR SETUP PURPOSE RDRETRY MUST BE FIRST ON RETRY LIST

; FOR SETUP PURPOSE SRETRY MUST BE LAST ON RETRY LIST

; GENERAL ERROR COUNTERS

2737	015564	000000
2738	015566	000000
2739	015570	000000
2740	015572	000000
2741	015574	000000

PARLOG:	0
HPARLOG:	0
NOERLOG:	0
UKNINT:	0
INTER:	0

; FOR SETUP PURPOSE PARLOG MUST BE FIRST ON COUNTERS LIST

; DRIVE RELATED ERROR COUNTERS

2743	015576	000000
2744	015600	000000
2745	015602	000000
2746	015604	000000
2747	015606	000000
2748	015610	000000
2749	015612	000000
2750	015614	000000
2751	015616	000000
2752	015620	000000
2753	015622	000000
2754	015624	000000

ZSEKLOG:	0
SEKLOG:	0
ZCRCLOG:	0
CRCLOG:	0
ZCRCBAD:	0
CRCBAD:	0
ZRDLOG:	0
RDLOG:	0
ZWRTLOG:	0
WRTLOG:	0
ZDATALOG:	0
DATALOG:	0

; "Z" LOGS ARE FOR DRIVE UNIT "ZERO"  
; NON"Z" LOGS ARE FOR DRIVE UNIT 1

\*\*\* NOTE: \*\*\*  
; ERROR CODES MUST NOT BE CHANGED  
; FROM THIS ORDER. ERROR DUMP  
; ASSUMES TAGS OF LOGS ARE IN ORDER SHOWN.

2755 015626 000000  
 2756 015630 000000  
 2757 015632 000000  
 2758 015634 000000  
 2759 015636 000000  
 2760 015640 000000  
 2761 015642 000000  
 2762 015644 000000  
 2763 015646 000000  
 2764 015650 000000  
 2765 015652 000000  
 2766 015654 000000  
 2767 015656 000000  
 2768 015660 000000  
 2769 015662 000000  
 2770 015664 000000  
 2771 015666 000000  
 2772 015670 000000  
 2773  
 2774  
 2775 015672 000021  
 2776  
 2777  
 2778  
 2779  
 2780  
 2781  
 2782  
 2783  
 2784  
 2785  
 2786  
 2787  
 2788 015734 000232  
 2789  
 2790  
 2791 016420 000232  
 2792  
 2793  
 2794  
 2795  
 2796 017104 000115  
 2797  
 2798  
 2799 017336 000115  
 2800  
 2801  
 2802 017570 000000 000000  
 2803 017574 000000 000000  
 2804  
 2805  
 2806 017600 000000  
 2807 017602 000000  
 2808

ZDDMIS: 0  
 DDMIS: 0  
 ZUNXDD: 0  
 UNXDD: 0  
 ZHSEKLOG: 0  
 HSEKLOG: 0  
 ZHCRCLOG: 0  
 HCRCLOG: 0  
 ZHCRCBAD: 0  
 HCRCBAD: 0  
 ZHRDLOG: 0  
 HRDLOG: 0  
 ZHMRTLOG: 0  
 HMRTLOG: 0  
 ZHDATALOG: 0  
 HDATALOG: 0  
 ZHDDLOG: 0  
 HDDLOG: 0

ERCODE: ;ERROR CODES  
 .BLKW 17.

;THE FOLLOWING 2 BLOCKS OF WORDS ARE TRACK ACCESS COUNTER AND  
 ;HEAD MOVED TO TRACK COUNTERS. THEY ARE DOUBLE PRECISION  
 ;COUNTERS IN THE FOLLOWING FORMATT:

;LOC.X LOW ORDER WORD TRACK 0  
 ;LOC.X+2 HIGH ORDER WORD TRACK 0  
 ;LOC.X+4 LOW ORDER WORD TRACK 1  
 ;LOC.X+6 HIGH ORDER WORD TRACK 1  
 ;ETC.

TKACC: ;TOTAL ACCESS / TRACK  
 .BLKW 232

HDMOVE: ;TOTAL HEAD MOVEMENT TO TRACK  
 .BLKW 232

;THE FOLLOWING 2 BLOCKS OF COUNTERS ARE SINGLE PRECISION

UOTRK: ;ERROR PER TRACK ON UNIT 0  
 .BLKW 115

UITRK: ;ERROR PER TRACK ON UNIT 1  
 .BLKW 115

;TOTAL WRITE AND READ FUNCTIONS DOUBLE PRECISION COUNTERS

WTCNTR: .WORD 0,0  
 RDCNTR: .WORD 0,0

;RESTART AND PASSES COMPLETED COUNTERS

RESTCNTR: 0  
 PASCNTR: 0

;FOR SETUP PURPOSE PASCNTR MUST BE PLACED LAST



```

2809 017604 032737 000020 010464 UILOG:      BIT #BIT4,UNITSEL      ;TEST FOR UNIT IN OPERATION
2810 017612 001402                      BEQ 1$                ;IF BIT 4 IS 0 RETURN AND INC. UNIT 0 COUNTERS
2811 017614 062703 000002                      ADD #2,R3             ;ADJUST THE CONTENTS OF R3 FOR
2812                                     ;THE ADDRESS OF UNIT 1 ERROR LOG COUNTERS.
2813 017620 000207                      1$:                  RTS PC
2814                                     ;
2815                                     ;STATISTICAL ERROR REPORT ROUTEEN
2816                                     ;PRINTS ALL ERROR LOGS AND OPERATING CONDITIONS
2817                                     ;
2818                                     ;*****
2819                                     ;
2820 017622 012706 001200 ERDUMP:          MOV #STACK,SP
2821 017626 104400 014172                      TYPE ,DBLLF
2822 017632 005737 021254                      TST SHORTRPT        ;IS SHORT REPORT REQUESTED
2823 017636 001405                      BEQ 1$                ;NO,PRINT LONG REPORT
2824 017640 104400 013772                      TYPE ,TAB
2825 017644 104400 015030                      TYPE ,MSHORT        ;YES,TYPE SHORT REPOR HEADER
2826 017650 000404                      BR 2$
2827 017652 104400 013772                      1$:                  TYPE ,TAB
2828 017656 104400 015007                      TYPE ,MDUMP         ;TYPE LONG REPORT HEADER
2829 017662 104400 015315                      2$:                  TYPE ,MREV          ;PRINT NAME AND REVISION OF PROGRAM
2830 017666 004737 002306                      JSR PC,ICOND        ;TYPE OUT SELECTED DRIVES AND TESTS
2831 017672 005737 017600                      TST RESTCNTR        ;HAVE THERE BEEN ANY RESTARTS
2832 017676 001412                      BEQ 3$                ;NO
2833 017700 104400 015043                      TYPE ,MRESTART      ;YES,PRINT OUT HOW MANY
2834 017704 013737 017600 017716          MOV RESTCNTR,12$
2835 017712 004537 013374                      JSR R5,SGLDEC
2836 017716 000000                      12$:                 OPEN
2837 017720 104400 014172                      TYPE ,DBLLF
2838 017724 005737 017602                      3$:                  TST PASCNTR        ;HAVE ANY PASSES BEEN COMPLETED
2839 017730 001005                      BNE 4$                ;YES,PRINT OUT HOW MANY
2840 017732 104400 015057                      TYPE ,MNOPAS        ;NO,TYPE PASS ABORTED
2841 017736 104400 014172                      TYPE ,DBLLF
2842 017742 000412                      BR 5$
2843 017744 104400 015074                      4$:                  TYPE ,MPASS         ;TYPE OUT NUMBER OF PASSES
2844 017750 013737 017602 017762          MOV PASCNTR,7$
2845 017756 004537 013374                      JSR R5,SGLDEC
2846 017762 000000                      7$:                  OPEN
2847 017764 104400 014004                      TYPE ,MCRLF
2848 017770 005737 021254                      5$:                  TST SHORTRPT        ;IS THIS A SHORT REPORT
2849 017774 001030                      BNE AERRS           ;YES,DON'T PRINT WRITE/READ TOTALS
2850                                     ;
2851                                     ;PRINTS TOTAL NUMBER OF SECTORS WRITTEN AND/OR
2852                                     ;READ FOR THE DURATION OF RUN TIME.
2853 017776 104400 015106                      TYPE ,MWRN          ;PRINT WRT/RD MESSAGE
2854 020002 104400 015117                      TYPE ,MSLASH
2855 020006 104400 014626                      TYPE ,MREAD
2856 020012 104400 014001                      TYPE ,DBLSP
2857 020016 012746 017570                      MOV #WTCNTR,-(SP)   ;ADDRESS OF TOTAL WRITES ON STACK
2858 020022 004737 013130                      JSR PC,@#SDG2D      ;TYPE TOTAL WRITES
2859 020026 004737 013324                      JSR PC,RTJUST
2860 020032 104400 015117                      TYPE ,MSLASH
2861 020036 012746 017574                      MOV #RDCNTR,-(SP)   ;ADDRESS OF READS ON STACK
2862 020042 004737 013130                      JSR PC,@#SDG2D      ;TYPE TOTAL READS

```

```

2863 020046 004737 013324 JSR PC,RTJUST
2864 020052 104400 014172 TYPE ,DBLLF
2865
2866 ;TYPE OUT MICRO CPU RELATED ERRORS
2867 020056 104400 014004 AERRS: TYPE ,MCRLF
2868 020062 005737 015564 TST PARLOG ;WERE THERE ANY PARITY ERRORS
2869 020066 001435 BEQ 1$
2870 020070 013737 015564 020102 MOV PARLOG,20$ ;YES,TYPE OUT THE NUMBER
2871 020076 004537 013374 JSR RS,SGLDEC
2872 020102 000000 20$: OPEN
2873 020104 104400 014001 TYPE ,DBLSP
2874 020110 104400 014751 TYPE ,MPAR
2875 020114 104400 014004 TYPE ,MCRLF
2876 020120 005737 015566 TST HPARLOG ;WERE THERE ANY HARD PARITY ERRORS
2877 020124 001416 BEQ 1$
2878 020126 013737 015566 020140 MOV HPARLOG,21$ ;YES,TYPE OUT THE COUNT
2879 020134 004537 013374 JSR RS,SGLDEC
2880 020140 000000 21$: OPEN
2881 020142 104400 014001 TYPE ,DBLSP
2882 020144 104400 014707 TYPE ,MUNREC
2883 020152 104400 014751 TYPE ,MPAR
2884 020156 104400 014004 TYPE ,MCRLF
2885 020162 005737 015570 1$: TST NOERLOG ;ANY STATUS ERROR FLAG ERRORS
2886 020166 001414 BEQ 3$ ;IF NONE CHECK NEXT ERROR
2887 020170 013737 015570 020202 MOV NOERLOG,24$ ;YES,TYPE OUT COUNT
2888 020176 004537 013374 JSR RS,SGLDEC
2889 020202 000000 24$: OPEN
2890 020204 104400 014001 TYPE ,DBLSP
2891 020210 104400 014766 TYPE ,MNOFLAG
2892 020214 104400 014004 TYPE ,MCRLF
2893 020220 005737 015574 3$: TST INTER ;ANY NO INTERRUPT ON DONE ERRORS
2894 020224 001414 BEQ 4$ ;IF NONE CHECK NEXT ERROR GROUP
2895 020226 013737 015574 020240 MOV INTER,25$ ;YES,PRINT OUT NUMBER
2896 020234 004537 013374 JSR RS,SGLDEC
2897 020240 000000 25$: OPEN
2898 020242 104400 014001 TYPE ,DBLSP
2899 020246 104400 014212 TYPE ,MINTER
2900 020252 104400 014004 TYPE ,MCRLF
2901 ;TYPES DRIVE RELATED ERRORS
2902 020256 104400 014172 4$: TYPE ,DBLLF
2903 020262 104400 014040 TYPE ,MUNITO ;TYPE OUT COLUMN HEADINGS
2904 020266 104400 014050 TYPE ,MUNITI
2905 020272 104400 015122 TYPE ,SPACE
2906 020276 104400 015021 TYPE ,MERS
2907 020302 104400 014004 TYPE ,MCRLF
2908 020306 005002 CLR R2 ;R2 USED AS AN ERROR PRINTED FLAG
2909 020310 013746 021256 MOV ZERO,-(SP) ;PUT ADDR OF ZERO MARK ON STACK
2910 020314 012746 013414 MOV @MUNXDD,-(SP) ;PUT ADDRESSES OF ALL RECOVERABLE
2911 020320 012746 013440 MOV @MDDMIS,-(SP) ;ERROR MESSAGES ON THE STACK
2912 020324 012746 013461 MOV @MDEHDA,-(SP)
2913 020330 012746 014742 MOV @MWRITE,-(SP)
2914 020334 012746 014626 MOV @MREAD,-(SP)
2915 020340 012746 014576 MOV @MBADCRC,-(SP)
2916 020344 012746 014670 MOV @MCRC,-(SP)

```



2917	020350	012746	014727		MOV #MSEEK, -(SP)	
2918	020354	012701	015576		MOV #ZSEKLOG, R1	: PUT FIRST ERROR COUNTER IN REGISTER
2919	020360	005716		7S:	TST (SP)	: GET ADDRESS OF MESSAGE
2920	020362	001436			BEQ 5S	: IF 0 FINISHED SOFT ERRS DO HARD ERRS
2921	020364	005721			TST (R1)+	: IS DRIVE 0 COUNTER CLEAR
2922	020366	001005			BNE 6S	: NO, GO PRINT ERROR COUNTER
2923	020370	005711			TST (R1)	: IS DRIVE 1 COUNTER CLEAR
2924	020372	001003			BNE 6S	: NO, GO PRINT ERROR COUNTER
2925	020374	005721			TST (R1)+	: ADJUST R1 FOR NEXT UNIT 0 COUNTER
2926	020376	005726			TST (SP)+	: ADJUST STACK FOR NEXT ADDRESS
2927	020400	000767			BR 7S	: GO TEST NEXT PAIR OF COUNTERS
2928	020402	005202		6S:	INC R2	: MAKE ERROR PRINTED FLAG NON ZERO
2929	020404	014137	020414		MOV -(R1), 30S	: TYPE CONTENTS OF UNIT 0 COUNTER
2930	020410	004537	013374		JSR RS, SGLDEC	
2931	020414	000000		30S:	OPEN	
2932	020416	104400	014001		TYPE ,DBLSP	
2933	020422	005721			TST (R1)+	: ADJUST R1 FOR UNIT 1 COUNTER
2934	020424	012137	020434		MOV (R1)+, 31S	: TYPE COUNTER, GET ADDR OF NEXT IN R1
2935	020430	004537	013374		JSR RS, SGLDEC	
2936	020434	000000		31S:	OPEN	
2937	020436	104400	014001		TYPE ,DBLSP	
2938	020442	012637	020450		MOV (SP)+, 13S	
2939	020446	104400			TYPE	: TYPE ERROR MESSAGE FROM STACK
2940	020450	000000		13S:	OPEN	
2941	020452	104400	014004		TYPE ,MCRLF	
2942	020456	000740			BR 7S	: GET NEXT COUNTER
2943	020460	104400	014004	5S:	TYPE ,MCRLF	
2944	020464	013746	021256		MOV ZERO, -(SP)	: PUT ADDR OF ZERO MARK ON THE STACK
2945	020470	012746	014547		MOV #MDDR, -(SP)	: PUT ADDRESS OF ALL HARD ERROR
2946	020474	012746	013461		MOV #MDDRHDR, -(SP)	: MESSAGES ON THE STACK
2947	020500	012746	014742		MOV #MWRITE, -(SP)	
2948	020504	012746	014626		MOV #MREAD, -(SP)	
2949	020510	012746	014576		MOV #MBADCRC, -(SP)	
2950	020514	012746	014670		MOV #MCRC, -(SP)	
2951	020520	012746	014727		MOV #MSEEK, -(SP)	
2952	020524	012701	015636		MOV #ZHSEKLOG, R1	: PUT FIRST HARD ERROR COUNTER IN R1
2953	020530	005716		12S:	TST (SP)	: GET ADDRESS OF MESSAGE FROM STACK
2954	020532	001437			BEQ BERRS	: IF 0 FINISHED ERROR LOGS
2955	020534	005721			TST (R1)+	: IS DRIVE 0 COUNTER CLEAR
2956	020536	001005			BNE 11S	: NO, PRINT ERROR COUNTER AND MESSAGE
2957	020540	005711			TST (R1)	: IS DRIVE 1 COUNTER CLEAR
2958	020542	001003			BNE 11S	: NO, GO PRINT THE ERROR
2959	020544	005721			TST (R1)+	: ADJUST R1 FOR NEXT UNIT 0 LOG
2960	020546	005726			TST (SP)+	: ADJUST STACK FOR NEXT ADDRESS
2961	020550	000767			BR 12S	
2962	020552	014137	020562	11S:	MOV -(R1), 32S	: TYPE CONTENTS OF UNIT 0 COUNTER
2963	020556	004537	013374		JSR RS, SGLDEC	
2964	020562	000000		32S:	OPEN	
2965	020564	104400	014001		TYPE ,DBLSP	
2966	020570	005721			TST (R1)+	: ADJUST R1 FOR UNIT 1 COUNTER
2967	020572	012137	020602		MOV (R1)+, 33S	: TYPE UNIT 1 LOG
2968	020576	004537	013374		JSR RS, SGLDEC	
2969	020602	000000		33S:	OPEN	
2970	020604	104400	014001		TYPE ,DBLSP	



2971	020610	104400	014707		TYPE ,MUNREC	;TYPE UNRECOVERABLE
2972	020614	012637	020622		MOV (SP)+,14\$	
2973	020620	104400			TYPE	;ERROR MESSAGE FROM STACK
2974	020622	000000		14\$:	OPEN	
2975	020624	104400	014004		TYPE ,MCRLF	
2976	020630	000737			BR 12\$	;GET NEXT LOG
2977					;PRINTS ERRORS PER ERROR CODES	
2978	020632	005702		BERRS:	TST R2	;WAS AN ERROR PRINTED IN THE LAST GROUP
2979	020634	001004			BNE 5\$	;IF R2 IS NONZERO BRANCH
2980	020636	104400	013772		TYPE ,TAB	
2981	020642	104400	015243		TYPE ,MNONE	;TYPE NONE
2982	020646	005002		5\$:	CLR R2	;CLEAR ERROR PRINTED FLAG AGAIN
2983	020650	104400	014172		TYPE ,DBLLF	
2984	020654	104400	015124		TYPE ,MCODES	;TYPE ERROR PER ERROR CODE
2985	020660	012700	000001		MOV #1,R0	;SET UP CODE COUNTER
2986	020664	012701	015672		MOV #ERCODE,R1	;GET ADDR OF FIRST ERROR CODE
2987	020670	020027	000022	2\$:	CMP R0,#22	;IS THE LAST CODE PRINTED
2988	020674	001001			BNE 3\$	;NO TEST THE NEXT ONE
2989	020676	000427			BR TYPTRK	;YES GO PRINT TRACK INFORMATION
2990	020700	005721		3\$:	TST (R1)+	;TEST FOR 0 WORD GET NEXT ADDRESS
2991	020702	001002			BNE 1\$	;IF NOT 0 TYPE COUNT AND CODE NUMBER
2992	020704	005200			INC R0	;INC CODE COUNTER
2993	020706	000770			BR 2\$	
2994	020710	014137	020720	1\$:	MOV -(R1),4\$	;PUT CONTENTS OF COUNTER IN OPEN
2995	020714	004537	013374		JSR R5,SGLDEC	;FOR TYPE OUT
2996	020720	000000		4\$:	OPEN	
2997	020722	104400	013772		TYPE ,TAB	
2998						
2999	020726	010002			MOV R0,R2	;PUT CODE NUMBER IN REGISTER TO
3000	020730	006302			ASL R2	;TYPE IT OUT IN THE FROM OF ERROR CODE
3001	020732	006302			ASL R2	
3002	020734	006302			ASL R2	
3003	020736	010246			MOV R2,-(SP)	::SAVE R2 FOR TYPEOUT
3004	020740	104402			TYPOS	::GO TYPE--OCTAL ASCII
3005	020742	004			.BYTE 4	::TYPE 4 DIGIT(S)
3006	020743	001			.BYTE 1	::TYPE LEADING ZEROS
3007	020744	104400	014004		TYPE ,MCRLF	
3008	020750	005200			INC R0	;SET UP FOR NEXT COUNTER
3009	020752	005721			TST (R1)+	;GET NEXT ERROR COUNTER
3010	020754	000745			BR 2\$	
3011					;PRINTS TRACK ACCESS, HEAD MOVEMENT, AND ERRORS PER TRACK	
3012	020756	005702		TYPTRK:	TST R2	;WAS AN ERROR PRINTED IN LAST GROUP
3013	020760	001004			BNE 5\$	;IF PRINTED ERROR FLAG NONZERO BRANCH
3014	020762	104400	013772		TYPE ,TAB	
3015	020766	104400	015243		TYPE ,MNONE	;TYPE NONE
3016	020772	104400	014172		TYPE ,DBLLF	
3017	020776	005737	021254	5\$:	TST SHORTRPT	;IF SHORT REPORT DON'T TYPE TRACK ACCESS OR ERRO
3018	021002	001121			BNE 6\$	
3019	021004	104400	015155		TYPE ,MTKLOG	;TYPE FORMAT OF ERROR PRINT OUT
3020	021010	005037	021070		CLR 2\$	;TRACK ADDRESS COUNTER
3021	021014	012704	015730		MOV #TKACC-4,R4	;SET UP ADDRESS OF TRACK ACCESSED
3022	021020	012703	016414		MOV #HDMOVE-4,R3	;ADDRESS OF HEAD MOVEMENT COUNTER
3023	021024	012702	017104		MOV #UOTRK,R2	;UNIT 0 ERROR PER TRACK COUNTER
3024	021030	012701	017336		MOV #UITRK,R1	;UNIT 1 ERROR PER TRACK COUNTER



3025	021034	062704	000004	1S:	ADD #4,R4	;ADJUST FOR ADDRESS OF NEXT COUNTER
3026	021040	005714			TST (R4)	;HAS THIS TRACK BEEN ACCESSED
3027	021042	001010			BNE 10S	;YES,PRINT OUT THE COUNTERS
3028	021044	005764	000002		TST 2(R4)	;TEST UPPER WORD OF ACCESS COUNTER
3029	021050	001005			BNE 10S	
3030	021052	062703	000004		ADD #4,R3	;THIS TRACK HAS NOT BEEN USED. ADJUST
3031	021056	005722			TST(R2)+	;REGISTERS TO POINT TO NEXT TRACK COUNTERS
3032	021060	005721			TST (R1)+	
3033	021062	000463			BR 12S	;TEST FOR LAST TRACK
3034	021064	004537	013374	10S:	JSR R5,SGLDEC	;TYPE TRACK NUMBER
3035	021070	000000		2S:	OPEN	;THIS LOCATION USED AS TRACK ADDR COUNTER
3036	021072	104400	015122		TYPE ,SPACE	
3037	021076	010446			MOV R4,-(SP)	;TYPE TRACK ACCESSED COUNT
3038	021100	004737	013130		JSR PC,@#5DB2D	
3039	021104	004737	013324		JSR PC,RTJUST	
3040	021110	104400	015122		TYPE ,SPACE	
3041	021114	062703	000004		ADD #4,R3	
3042	021120	010346			MOV R3,-(SP)	;TYPE HEAD MOVED TO COUNT
3043	021122	004737	013130		JSR PC,@#5DB2D	
3044	021126	004737	013324		JSR PC,RTJUST	
3045	021132	104400	014001		TYPE ,DBLSP	
3046	021136	104400	015122		TYPE ,SPACE	
3047	021142	005712			TST (R2)	;ARE THERE ANY UNIT 0 ERRORS
3048	021144	001421			BEQ 20S	;NO,CHECK UNIT 1
3049	021146	012237	021156		MOV (R2)+,3S	;TYPE UNIT 0 ERRORS
3050	021152	004537	013374		JSR R5,SGLDEC	
3051	021156	000000		3S:	OPEN	
3052	021160	005711			TST (R1)	;ARE THERE ANY UNIT 1 ERRORS
3053	021162	001420			BEQ 7S	;NO,GO ADJUST POINTERS
3054	021164	104400	015122		TYPE ,SPACE	
3055	021170	104400	014001	22S:	TYPE ,DBLSP	
3056	021174	012137	021204		MOV (R1)+,4S	;TYPE UNIT 1 ERRORS
3057	021200	004537	013374		JSR R5,SGLDEC	
3058	021204	000000		4S:	OPEN	
3059	021206	000407			BR 11S	;GO TO NEXT TRACK
3060						
3061	021210	005722		20S:	TST (R2)+	;NO ERRORS,ADJUST REG.FOR NEXT COUNTER
3062	021212	005711			TST (R1)	;ARE THERE ANY UNIT 1 ERRORS
3063	021214	001403			BEQ 7S	;NO,GO TO NEXT TRACK
3064	021216	104400	013772		TYPE ,TAB	;YES,TYPE TAB FOR PLACEMENT
3065	021222	000762			BR 22S	;GO TYPE THE NUMBER
3066	021224	005721		7S:	TST (R1)+	;NO ERRORS,ADJUST REG. FOR NEXT COUNTER
3067	021226	104400	014004	11S:	TYPE ,MCRLF	
3068	021232	005237	021070	12S:	INC 2S	
3069	021236	023727	021070	000115	CMP 2S,#115	;HAVE ALL TRACK ERRORS BEEN TYPED
3070	021244	001273			BNE 1S	;NO,TYPE THE NEXT ONE
3071	021246	005037	021254	6S:	CLR SHORTRPT	;SET UP FOR LONG REPORT
3072	021252	000000		HLT17:	HALT	;YES DONE
3073						
3074	021254	000000		SHORTRPT:	0	
3075	021256	000000		ZERO:	0	
3076						
3077		000001				

.END















NRX11	015276	1480	2707*											
MSECT	013536	1489	2467*											
MSECK	014727	826	2640*	2917	2951									
MSEQ	014206	1512	2549*											
MSHORT	015030	2660*	2825											
MSLASH	015117	2677*	2854	2860										
MSUM	015261	1197	2703*											
MTEST	014202	1506	2547*											
MTKLOG	015155	2687*	3019											
MTRK	013507	1485	2459*											
MUKNIN	014244	1415	2557*											
MUNITO	014040	601	1497	1524	2517*	2903								
MUNITI	014050	604	1499	1531	2520*	2904								
MUNREC	014707	542	748	820	1008	1025	1057	1218	2636*	2882	2971			
MUNX00	013414	1072	2444*	2910										
MURITE	014742	784	785	795	796	805	821	925	1026	2643*	2913	2947		
MURTEN	015106	2674*	2853											
NEWTRK	011326	1862	1865	1876	1878	1906	1908	1915*	1934	1939	1950	1966	1968	1992
NEXTRD	004272	929	931*	950	1012	1062								
NOACT	001516	482	488*											
NOERLO	015570	1420*	2739*	2885	2887									
NOERTY	005770	1149	1153	1180*										
NOINTE	006776	770	903	1388*										
NOFRNT	007264	1436	1444*											
NOSEL	001654	492	515*											
OD	001200	237*	238	551	553	557	1789*	1790	1793*	1794	1798	1802*	1858*	1861
		1874*	1885	1887*	1902*	1903	1905	1935	1937*	1949	1957	1959*	1962	1967
		1978*												
ODCOMP	011626	1982	1985*	1988										
ODNEXT	011556	1963	1967*											
ONEWRT	002762	710*	790	815	1019									
OPCOND	001660	494	497	499	516*									
OPEN =	000000	164*	559	564	577	582	734	749	830	1169	1196	1451	1488	1493
		1765	2429	2836	2846	2872	2880	2889	2897	2931	2936	2940	2964	2969
		2974	2996	3035	3051	3058								
PARLOG	015564	445	530*	731*	2737*	2868	2870							
PARTES	003076	728	731*	787	942	1144								
PASCNT	017602	447	1449	1747*	1763	2807*	2838	2844						
PAT	010200	523*	524*	525*	526*	527*	1501	1593	1595*	1596	1612*	1770		
PATGEN	010206	1621*	1622	1631										
PAT125	010274	1606	1668*											
PAT314	010314	1607	1679*											
PC =%	000007	66*	469*	475*	495*	502*	505*	513*	515*	540*	555*	572*	587*	595*
		606*	613*	614*	621*	624*	628*	638*	666*	697*	698*	699*	700*	701*
		704*	707*	708*	711*	719*	768*	769*	776*	797*	800*	802*	806*	814*
		817*	822*	824*	832*	841*	851*	853*	858*	872*	873*	874*	875*	876*
		877*	880*	888*	889*	901*	902*	910*	920*	927*	930*	933*	949*	952*
		956*	960*	975*	984*	999*	1005*	1021*	1036*	1038*	1043*	1054*	1060*	1066*
		1069*	1074*	1084*	1085*	1086*	1087*	1088*	1089*	1092*	1102*	1109*	1117*	1136*
		1164*	1188*	1205*	1207*	1212*	1215*	1223*	1226*	1240*	1241*	1242*	1243*	1244*
		1247*	1257*	1258*	1259*	1260*	1261*	1264*	1276*	1277*	1278*	1279*	1280*	1281*
		1282*	1283*	1284*	1287*	1303*	1307*	1308*	1309*	1310*	1311*	1316*	1317*	1318*
		1321*	1343*	1374*	1396*	1419*	1429*	1432*	1458*	1460*	1464*	1470*	1476*	1478*
		1517*	1543*	1546*	1562*	1621*	1644*	1669*	1687*	1689*	1696*	1698*	1723*	1730*





		1403	1410	1428*	1463*	1466*	1542*							
RXDB	001210	284*	460	470	488	503	507	510	622	720*	852*	854*	1119	1402
RXERR0	007166	1430	1434	1468	1472	1541	1544	1548						
RXPWR	007462	1404	1422	1426*										
RO	=%000000	1407	1480*											
		57*	720	1121	1135*	1175	1176*	1890*	1891*	1892*	1994*	1995*	1996*	1997*
		1999*	2000*	2001	2002*	2003*	2004*	2005*	2006*	2007*	2008	2009	2070	2071*
		2072	2075*	2228	2253*	2264	2265*	2266	2267*	2268*	2269*	2270*	2294	2315*
		2363*	2364	2382*	2386*	2418	2419*	2420	2421	2423	2425*	2427*	2430*	2985*
		2987	2992*	2999	3008*									
R1	=%000001	58*	928	953	1101*	1108*	1156	1159	1166*	1225*	2229	2252*	2295	2314*
		2365*	2371*	2377*	2918*	2921	2923	2925	2929	2933	2934	2952*	2955	2957
		2959	2962	2966	2967	2986*	2990	2994	3009	3024*	3032	3052	3056	3062
		3066												
R2	=%000002	59*	550*	590	594*	1641*	1643*	1646	2230	2251*	2296	2313*	2362*	2365
		2366*	2372*	2373*	2378*	2379*	2908*	2928*	2978	2982*	2999*	3000*	3001*	3002*
		3003	3012	3023*	3031	3047	3049	3061						
R3	=%000003	60*	440*	441*	442*	799*	801*	816*	818*	909*	911*	919*	921*	959*
		961*	974*	976*	983*	985*	998*	1000*	1004*	1006*	1020*	1022*	1035*	1037*
		1053*	1055*	1065*	1067*	1204*	1206*	1214*	1216*	1367*	1373	2165	2174*	2180*
		2181*	2184*	2189*	2190*	2191	2200*	2231	2250*	2297	2312*	2370*	2375*	2381*
		2382	2811*	3022*	3030*	3041*	3042							
R4	=%000004	61*	676*	677*	678*	679	1596*	1597*	1598*	1599	1600*	1693*	1694	1831*
		1832*	1833*	1834	2166	2168*	2169*	2170*	2171	2172*	2186	2188*	2196*	2199*
		2232	2249*	2298	2311*	2368*	2371	2377	2379	3021*	3025*	3026	3028	3037
R5	=%000005	62*	445*	446*	447	449*	450*	451	558*	563*	576*	581*	829*	834*
		835*	836*	839*	840*	843*	844*	845*	846*	847*	848*	1168*	1195*	1437*
		1438*	1439*	1440*	1443*	1450*	1487*	1492*	1764*	1773*	1776	1915*	1916*	1917*
		1918*	1919*	1920*	2167	2173*	2175*	2177*	2178*	2179*	2180	2198*	2233	2248*
		2299	2310*	2369*	2373	2380	2436	2441*	2835*	2845*	2871*	2879*	2888*	2896*
		2930*	2935*	2963*	2968*	2995*	3034*	3050*	3057*					
R6	=%000006	63*	65											
R7	=%000007	64*	66											
SAVREG=	104404	2285*	2361											
SA200	001220	206	428*	547	612	636								
SECCNT	012052	774*	931*	1103	1221*	2024*	2025*	2026*	2027*	2034*				
SEEKER	003424	797	799*	949										
SEKLOG	015600	2744*												
SEKRTY	003514	790*	815*	946*										
SEKTYP	003560	806	822	826*										
SEGSAV	006604	1304*	1315	1332*										
SEQUEN	011134	516*	517*	1304	1305*	1315*	1513	1828	1830*	1831	1851*			
SEQ1	011136	1838	1858*	1894										
SEQ2	011172	1839	1872*	1893										
SEQ3	011224	1840	1885*											
SEQ4	011270	1841	1902*											
SEQ5	011354	1842	1929*											
SEQ6	011506	1843	1957*											
SEQ7	011566	1844	1978*											
SGLDEC	013374	558	563	1990										
		2845	2871	2879	581	829	1168	1195	1450	1487	1492	1764	2436*	2835
		1537*	2822	2848	2888	2896	2930	2935	2963	2968	2995	3034	3050	3057
SHORTR	021254	65*	428*	429*	430*	436	455*	460*	465*	532*	661*	662*	663*	712*
SP	=%000006	713*	715*	716*	724	760*	761*	780	823*	855*	856*	896*	897*	935





	1905*	1907*	1915	1933*	1938*	1943*	1946*	1949*	1964*	1967*	1991*				
TBITVE= 000014	137#														
TCRLF = 000200	2080	2119#													
TEST 002052	518#	519#	520*	521*	522*	548#	673	675*	676	1507					
TESTDR 001466	474	476#													
TESTSE 002560	667#	1530													
TEST7 006422	687	1303#													
TEST7X 006434	1305#	1323	1328	1330											
THREE 012060	2029	2037#	2039												
THT = 000011	2078	2118#													
TKACC 015734	846	2788#	3021												
TKVEC = 000060	144#														
TPVEC = 000064	145#														
TRAPVE= 000034	143#														
TRCK 006606	719	851	853	1117	1337#										
TRKCNT 011122	702#	878#	1090*	1245#	1262*	1285*	1312*	1319*	1799*	1800*	1801*	1846#	1888*		
	1889#	1892	1931*	1940	1947	1960*	1961*								
TRKERR 003612	802	834#	952	1038	1069	1207									
TRTVEC= 000014	138#														
TSECTO 012054	852	1127	1130	1490	2028*	2029*	2035#	2039*	2040	2042*	2043*				
TSTLIM 002054	550#	613													
TSTMSG 002266	555	572	590#												
TYPADR 007476	1043	1060	1164	1458	1485#										
TYP000 010030	1460	1550#													
TYPE = 104400	453	454	459	464	472	483	509	531	538	539	542	543	544		
	545	556	560	561	573	574	578	579	586	592	593	597	598		
	601	604	605	607	610	634	734	735	736	747	748	749	750		
	751	805	819	820	821	826	827	831	913	914	915	916	923		
	924	925	926	964	965	977	978	979	980	988	989	989	1007	1008	
	1009	1010	1024	1025	1026	1041	1044	1056	1057	1058	1059	1061	1072		
	1161	1162	1163	1165	1170	1174	1179	1193	1197	1200	1202	1203	1217		
	1218	1219	1220	1362	1379	1391	1392	1415	1416	1423	1424	1446	1447		
	1448	1452	1453	1459	1474	1475	1480	1481	1485	1489	1494	1497	1499		
	1500	1505	1506	1511	1512	1524	1525	1526	1531	1532	1533	1549	1550		
	1555	1556	1561	1750	1755	1757	1758	1761	1762	1766	2083	2192	2281#		
	2318	2428	2821	2824	2825	2827	2828	2829	2833	2837	2840	2841	2843		
	2847	2853	2854	2855	2856	2860	2864	2867	2873	2874	2875	2881	2882		
	2883	2884	2890	2891	2892	2898	2899	2900	2902	2903	2904	2905	2906		
	2907	2932	2937	2939	2941	2943	2965	2970	2971	2973	2975	2980	2981		
	2983	2984	2997	3007	3014	3015	3016	3019	3036	3040	3045	3046	3054		
	3055	3064	3067												
	1157	1160	1167#												
TYPERR 005714	2282#														
TYP0C = 104401	1178	2284#													
TYP0N = 104403	456	461	466	533	1172	1455	1502	1508	1514	1552	1558	2283#	3004		
TYP0S = 104402	606	1498	1500#												
TYPSEL 007572	2989	3012#													
TYPTRK 020756	1412#	2740#													
UKNINT 015572	477#	490#	498#	512*	599	602	608	837	849	1042*	1068*	1146#	1322		
UNITSE 010464	1426#	1495	1527*	1528	1534#	1535	1717	1719	1721*	1722*	1724	1726	1728*		
	1729#	1734#	1738	1740	1753	1769#	2809								
UNXDD 015634	2758#														
UOTRK 017104	836	2796#	3023												
UILOG 017604	800	817	910	920	960	975	984	999	1005	1021	1036	1054	1066		











ADC	767	848	900	1376	1920	2378									
ADD	766	823	836	839	846	847	899	1011	1018	1120	1129	1130	1375	1418	1440
	477	1692	1742	1918	1919	1987	1995	1996	2006	2039	2043	2076	2160	2170	2377
ASL	2379	2438	2811	3025	3030	3041									
ASLB	678	835	844	845	1598	1833	1889	1916	1917	1961	2268	3000	3001	3002	
BCC	1132														
BEG	1642														
	434	492	504	508	537	552	554	569	571	623	744	764	782	813	838
	978	918	937	972	996	1016	1032	1051	1064	1151	1155	1187	1190	1199	1211
	1222	1326	1345	1353	1372	1406	1431	1436	1469	1545	1647	1695	1754	1760	1774
	1404	1990	2079	2187	2422	2810	2823	2832	2869	2877	2886	2894	2920	2954	3048
BGE	3053	3063													
BGT	631	1893	1986	2041	2113										
BIC	1153	2194													
	517	519	527	1166	1327	1491	1527	1534	1721	1728	1769	1789	1858	1874	1887
	1902	1959	1978	1980	1997	2007	2184								
BICB	672	1591	1827	1932	1937										
BIS	490	512	1042	1068	1101	1108	1146	1329	1426	1722	1729	1802	2189	2190	2381
BISB	679	849	1599	1834											
BIT	470	491	496	507	616	622	732	740	763	781	803	810	837	936	943
	962	969	986	993	1013	1031	1039	1048	1063	1070	1148	1150	1191	1208	1322
	1325	1371	1389	1405	1408	1413	1421	1430	1444	1461	1468	1495	1544	1717	1726
	1740	1753	1759	2809											
BITB	503	1033	1350	1410	1522	1940	2101								
BLE	1982														
BLT	2092	2195	2374												
BMI	1404	1473	1536												
BNE	448	452	471	482	497	567	585	591	609	617	633	674	703	733	741
	773	775	804	811	879	932	944	958	963	970	987	994	1003	1014	1034
	1040	1049	1071	1091	1104	1122	1149	1192	1209	1246	1263	1286	1313	1320	1323
	1349	1361	1378	1390	1409	1411	1414	1422	1445	1462	1496	1523	1594	1718	1727
	1741	1749	1771	1791	1829	1860	1941	1948	1963	2021	2073	2081	2088	2102	2109
	2185	2309	2385	2424	2839	2849	2922	2924	2956	2958	2979	2988	2991	3013	3018
	3027	3029	3070												
BPL	474	489	494	511	600	603	627	738	808	929	954	967	991	1046	1157
	1160	1181	1342	1394	1529	1720	1725	1739	1873	1886	1930	1936	1958	2067	2106
	2183														
BR	203	204	205	435	499	506	565	583	625	680	705	722	881	950	1027
	1073	1093	1124	1158	1201	1248	1265	1288	1328	1330	1433	1465	1467	1471	1498
	1547	1601	1622	1631	1650	1659	1671	1680	1690	1756	1837	1862	1865	1876	1878
	1894	1906	1908	1934	1939	1944	1950	1966	1968	1984	1988	1992	2069	2085	2095
	2104	2111	2161	2176	2197	2303	2324	2376	2426	2826	2842	2927	2942	2961	2976
	2989	2993	3010	3033	3059	3065									
CLC	1645	1998													
CLR	446	477	550	667	670	671	714	855	895	1106	1107	1110	1113	1225	1314
	1337	1354	1357	1369	1592	1620	1658	1768	1795	1797	2002	2174	2307	2370	2908
	2982	3020	3071												
CLRB	498	1793	2027	2110	2386										
CMP	433	436	447	451	907	917	1152	1344	1694	1770	2380	2987	3069		
CMPB	481	566	570	584	629	1121	1859	1903	1947	1962	1981	1985	1989	2040	2078
	2080	2087	2108	2112	2423										
COM	871	1017	1083	1275	1306	1537									
COMB	1670														
DEC	536	632	677	702	742	774	812	878	931	971	995	1015	1050	1090	1210







ROL	524	525	1999	2000	2175	2177	2178	2179	2181						
ROLB	1649														
ROR	520	521	522	1438	1439	2004	2005								
RTI	431	664	717	857	1115	1340	1359	1417	2077	2203	2238	2254	2322		
RTS	513	540	587	595	614	638	776	824	832	841	858	930	933	1074	1136
	1188	1226	1343	1374	1419	1478	1517	1562	1696	1698	1723	1730	1743	1780	1803
	1921	2010	2032	2045	2116	2270	2345	2388	2432	2441	2813				
SBC	2372														
SEC	1640	1648													
SUB	1800	1891	1892	1983	2025	2029	2371	2373							
SWAB	526														
TRAP	2272	2282	2283	2284	2285	2286									
TST	473	493	551	568	590	602	608	626	673	724	737	743	772	780	807
	928	935	953	957	966	990	1002	1045	1103	1140	1154	1180	1186	1189	1356
	1377	1393	1403	1435	1528	1593	1646	1697	1724	1738	1748	1790	1828	2020	2074
	2082	2103	2186	2266	2822	2831	2838	2848	2868	2876	2885	2893	2919	2921	2923
	2925	2926	2933	2953	2955	2957	2959	2960	2966	2978	2990	3009	3012	3017	3026
	3028	3031	3032	3047	3052	3061	3062	3066							
TSTB	488	510	553	599	1156	1159	1198	1341	1472	1535	1719	1872	1885	1929	1935
	1957	2066	2105	2421											
.ASCII	2127	2129	2182												
.ASCIIZ	2128	2129	2326	2444	2449	2454	2459	2463	2467	2470	2490	2495	2499	2502	2504
	2506	2508	2510	2512	2517	2520	2523	2527	2531	2534	2537	2540	2543	2545	2547
	2549	2551	2557	2561	2567	2570	2574	2577	2580	2587	2589	2591	2594	2597	2603
	2608	2611	2615	2620	2622	2623	2628	2631	2636	2640	2643	2646	2650	2654	2657
	2660	2663	2667	2671	2674	2677	2679	2681	2687	2698	2701	2703	2707	2711	
.BLKB	2409	2723													
.BLKW	2775	2788	2791	2796	2799										
.BYTE	457	458	462	463	467	468	534	535	1456	1457	1503	1504	1509	1510	1515
	1516	1553	1554	1559	1560	2123	2124	2125	2126	2204	2205	2206	2207	3005	3006
.ENABL	4														
.END	3077														
.ENDC	15	47	133	147	189	212	244	264	282	342	407	425	458	459	463
	464	468	469	535	536	641	660	692	758	866	884	1079	1097	1234	1252
	1270	1292	1334	1399	1457	1458	1504	1505	1510	1511	1516	1517	1554	1555	1560
	1561	1566	1588	1616	1625	1634	1653	1662	1674	1683	1703	1716	1785	1808	1825
	1854	1868	1881	1898	1911	1925	1953	1973	2017	2049	2072	2132	2210	2256	2265
	2268	2281	2282	2283	2284	2285	2286	2287	2288	2300	2310	2320	2322	2329	2330
	2348	2412	2719	2819	3006	3007									
.EQUIV	47	48	50	65	66	95	96	97	98	99	100	101	102	103	104
	123	124	125	126	127	128	129	130	131	132					
.EVEN	2130	2328	2716												
.IF	11	45	105	133	188	211	243	263	281	341	406	424	457	458	462
	463	467	468	534	535	640	659	691	757	865	883	1078	1096	1233	1251
	1269	1291	1333	1398	1456	1457	1503	1504	1509	1510	1515	1516	1553	1554	1559
	1560	1565	1587	1615	1624	1633	1652	1661	1673	1682	1702	1715	1784	1807	1824
	1853	1867	1880	1897	1910	1924	1952	1972	2016	2048	2072	2131	2209	2255	2264
	2268	2272	2282	2283	2284	2285	2286	2287	2300	2310	2318	2320	2322	2326	2329
	2347	2411	2718	2818	3005	3006									
.IFF	47	189	212	244	264	282	342	407	425	457	458	462	463	468	535
	641	660	692	758	866	884	1079	1097	1234	1252	1270	1292	1334	1399	1457
	1504	1510	1516	1554	1560	1566	1588	1616	1625	1634	1653	1662	1674	1683	1703
	1716	1785	1808	1825	1854	1868	1881	1898	1911	1925	1953	1973	2017	2049	2132
	2210	2256	2265	2288	2320	2330	2348	2412	2719	3006	3007				



.IIF	10 2122	15 2123	20 2124	21 2125	456 2126	461 2127	466 2128	533 2129	1455 2130	1502 2281	1508 2282	1514 2283	1552 2284	1558 2285	2121 2286
.IRP	3004 2228	2248	2294	2310											
.LIST	3	147	168	2272	2281	2282	2283	2284	2285	2286	2287				
.MACRO	2272														
.MCALL	5	6	147												
.NLIST	2	147	168	2272	2281	2282	2283	2284	2285	2286	2287				
.REPT	168														
.SBTTL	43	2050	2133	2211	2257	2273	2289	2331	2349						
.TITLE	10														
.WORD	167 2803	181	337	338	339	1173	2115	2121	2122	2208	2319	2321	2346	2384	2802

ERRORS DETECTED: 0

\*DZRAD, DZRAD/CRF/SOL=DZRAD  
RUN-TIME: 37 27 5 SECONDS  
CORE USED: 12K

