

RX11

RX11 DIAGNOSTIC
MD-11-DZRXB-D

EP-DZRXB-D-DL-A

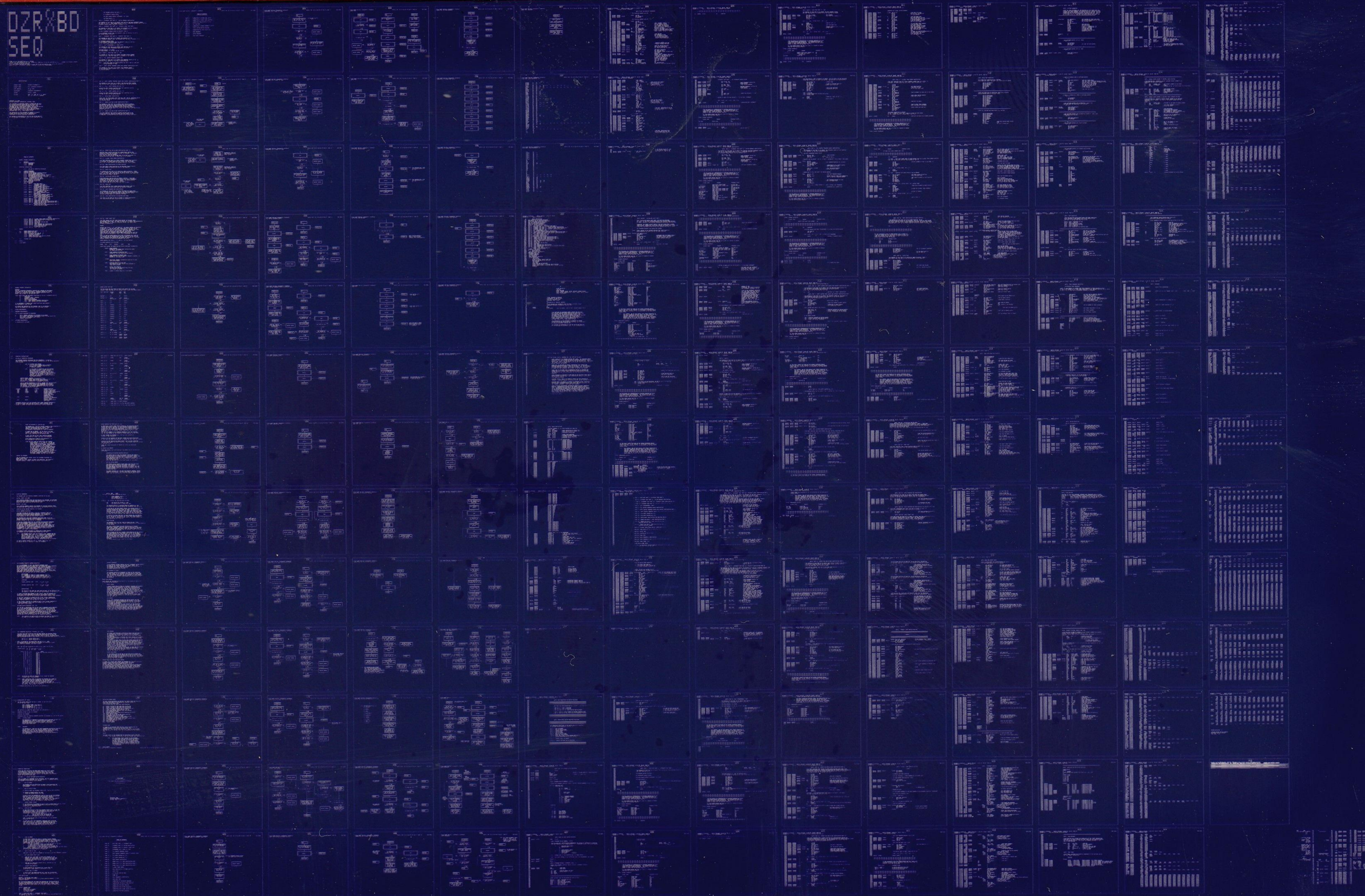
FEB 1976

COPYRIGHT ©1976

digital

FICHE 1 OF 1

Made In U.S.A.



[illegible]

The image displays three 10x10 grids illustrating the evolution of a fractal pattern. The first grid shows a sparse pattern of 'S' characters. The second grid shows a more complex pattern with 'S' and 'E' characters. The third grid shows a highly complex, fractal-like pattern with 'S', 'E', and 'O' characters.

```
LPTSPPL Version 101(2107) Running on MTA140
*START* User WELLSINGHAM A(1,2) Job DZRXB0 Seq. 3592 Date 17-Jan-76 18:27:33 Monitor      RV2250 KI10 SYS#514 *START*
Request created: 17-Jan-76 17:57:10
File: WHTG:DZRXB0.SEQ(055)(404,3722) Created: 09-Jan-76 16:12:28 Printed: 17-Jan-76 18:27:34
QUEUE Switches: /FILE:ASCII /COPIES:1 /SPACING:1 /LIMIT:410 /FORMS:NORMAL
```

IDENTIFICATION

SEQ 0001

PRODUCT CODE: MAINDEC-11-DZRXB-D-D
PRODUCT NAME: RX11 INTERFACE DIAGNOSTIC
DATE CREATED: DEC 21, 1975
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: DAVID L. ADAMS
REVISED: SEPT. 12, 1975 BY D. L. ADAMS
NOV. 10, 1975 BY B. 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)
 - 2.3.4.1 PREREQUISITES OF TESTS
- 2.4 OPERATOR ACTION TO RUN THE PROGRAM
 - 2.4.1 STARTING THE PROGRAM
 - 2.4.2 OPERATING CONDITIONS
- 2.5 TEST DEFINITIONS
 - 2.5.1 PRETEST
 - 2.5.2 TEST 1 - RXCS TEST PART I /
INTERRUPT TEST PART I
 - 2.5.3 TEST 2 - INTERRUPT TEST PART II /
VECTOR ADDRESS VERIFICATION
 - 2.5.4 TEST 3 - INTERRUPT TEST PART III /
PRIORITY LEVEL VERIFICATION PART I
 - 2.5.5 TEST 4 - INTERRUPT TEST PART IV /
PRIORITY VERIFICATION PART II
 - 2.5.6 TEST 5 - INIT (PROGRAMED) / RST
 - 2.5.7 TEST 6 - FILL BUFFER TRANSFER LENGTH
VERIFICATION
 - 2.5.8 TEST 7 - EMPTY BUFFER TRANSFER LENGTH AND
CONTENT VERIFICATION PART I
 - 2.5.9 TEST 10 - EMPTY BUFFER TRANSFER LENGTH AND
CONTENT VERIFICATION PART II
 - 2.5.10 TEST 11 - FILL / EMPTY BUFFER ALL 0'S
 - 2.5.11 TEST 12 - FILL / EMPTY BUFFER ALL 1'S
 - 2.5.12 TEST 13 - DRIVE READY VERIFICATION
 - 2.5.13 TEST 14 - ERROR FLAG AND B-CODE VERIFICATION PART I
 - 2.5.14 TEST 15 - ERROR FLAG AND B-CODE VERIFICATION PART II
/DELETED DATA BIT SETS
 - 2.5.15 TEST 16 - ERROR FLAG AND B-CODE VERIFICATION PART III
/DELETED DATA BIT CLEARS

E01

SEQ 0003

2.5.16 TEST 17 - ILLEGAL TRACK ERROR AND B-CODE VERIFICATION
2.5.17 TEST 20 - SEEK VERIFICATION VIA READ FUNCTION
2.5.18 TEST 21 - WRITE TEST
2.5.19 TEST 22 - INITIALIZE IMPLIED READ
2.5.20 TEST 23 - READ TEST
2.5.21 TEST 24 - DATA TRANSFER AND VERIFICATION
2.5.22 TEST 25 - DATA TRANSFER AND VERIFICATION
 /VIA DELETED DATA MODE
2.5.23 TEST 26 - HEAD "HOME" TEST

3.0 ERRORS

3.1 ERROR HEADING FOR TESTS 1 - 17, 21 - 23
3.2 ERROR OUTPUT PER TEST
3.3 ERROR HEADING FOR TEST 20, 24 - 26
3.3.1 NO ERROR FLAG ERRORS
3.3.2 ERROR FLAG ERRORS
3.3.3 ERRORS RESULTING FROM PREVIOUS ERRORS
3.3.4 DEFINITIVE ERROR CODES

3.4 PROGRAM HUNG

4.0 HALTS

5.0 FLOW CHARTS

1.0 GENERAL PROGRAM INFORMATION

1.1 ABSTRACT
THE RX11 INTERFACE DIAGNOSTIC CONSISTS OF A SERIES OF SELECTABLE TESTS THAT MAY BE RUN INDIVIDUALLY, SEQUENCE THROUGH ALL TESTS, OR START AT A SELECTED TEST AND RUN THROUGH REMAINING TESTS, IN ORDER, THEN GO BACK TO THE SELECTED TEST.

THESE TESTS CHECK OUT THE BASIC FUNCTIONS OF THE RX11 INTERFACE SUCH AS:

- A. DONE FLAG
- B. INTERRUPT LEVEL / ADDRESS
- C. PROGRAM INITIALIZE
- D. READ STATUS REGISTERS
- E. FILL / EMPTY BUFFER TRANSFER VERIFICATION
- F. FILL / EMPTY BUFFER WITH DATA PATTERNS

IT IS NECESSARY TO INSURE THAT THESE FUNCTIONS WORK BEFORE A DATA RELIABILITY TEST IS RUN.

ANY ERRORS ARE REPORTED BY THE PROGRAM, AND IT IS POSSIBLE TO LOOP ON THE ERROR OR A PARTICULAR TEST FOR SCOPE TESTING.

1.2 SYSTEM REQUIREMENTS

1.2.1 HARDWARE REQUIREMENTS

THE FOLLOWING EQUIPMENT IS REQUIRED:

- A. PDP-11 SERIES COMPUTER WITH MINIMUM OF 8K MEMORY
- B. RX11 FLOPPY DISK SYSTEM, INCLUDING A SINGLE OR DUAL DRIVE RX01 AND A PDP-11 INTERFACE CARD (M7B46).
- C. CONSOLE TELEPRINTER

1.2.2 SOFTWARE REQUIREMENTS

NO PREREQUISITE SOFTWARE

2.0 OPERATING INSTRUCTIONS

2.0.1 OUTLINE OF OPERATING PROCEDURE

THE STANDARD RUNNING PROCEDURE FOR THE DIAGNOSTIC (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 IS BEING LOADED FROM A DISKETTE
REPLACE THE "LIBRARY" DISKETTE WITH
A "SCRATCH" DISKETTE.
- NOTE: IF THIS STEP IS FORGOTTEN AND THE PROGRAM
WAS LOADED VIA RXDP (FLOPPY MONITOR) ON
UNIT J WITH UNIT 0 SELECTED BY USER TO
UNDERGO TESTING THE PROGRAM WILL FAILSAFE
THE OPERATION AND PROMPT THE USER AS FOLLOWS:
"CAUTION - IF YOU DESIRE TO TEST UNIT 0
REPLACE LOAD MEDIUM WITH A SCRATCH DISKETTE
THEN PRESS CONTINUE"

CAUTION AGAIN, HOWEVER -----
NOTE 1) WHEN RUNNING THIS PROGRAM ON A SMALL 11 (E.G. /04,
LSI 11, ETC.) WHERE THERE IS NO CONSOLE SWITCH
REGISTER IT IS IMPERATIVE TO REMEMBER THIS SETUP.

NOTE 2) BEFORE PROCEEDING TO STEP B. ENSURE THAT THE FOLLOWING
MODIFIABLE LOCATIONS CONTAIN THE PARAMETERS YOU REQUIRE
FOR TESTING. THE FOLLOWING TABLE DESCRIBES EACH LOCATION
WITH RESPECT TO THE DEFAULT PARAMETERS WHICH WILL BE USED
IF LEFT UNMODIFIED BY THE USER:

LOCATION	LABEL	CONTENTS	PROGRAM REACTION
1200	OD:	0	TRACKS 0,52,53,114(8)
1202	FIRST:	015001	SECTORS 1 THRU 32(8)
1204	KRXVEC:	264	ASSUMES PROPER DEVICE VECTOR
1206	RXCS:	177170	ASSUMES PROPER DEVICE STATUS REGISTER (CALCULATES 'RXDB' ADDRESS FROM)
1212	DTESTP:	0	TESTS BOTH UNITS AUTOMATICALLY SEQUENCES THRU ALL TESTS
1214	BRLEV:	5	ASSUMES PROPER DEVICE 'BR' LEVEL

REFERENCE SECTION 2 OF THIS DOCUMENT FOR A MORE THOROUGH DESCRIPTION
OF EACH OF THESE ITEMS AND HOW TO MODIFY THESE LOCATIONS IF YOU
DESIRE TO CHANGE THE ABOVE MENTIONED DEFAULT TESTING PARAMETERS.

- B. START THE PROGRAM AT LOCATION 200
- C. THE PROGRAM WILL TYPE OUT MAINDEC NUMBER, A TEST PARAMETER OF 0 (USE BOTH DRIVES AND RUN ALL TESTS). THEN TYPE TRACKS TO BE ACCESSED AND SECTOR LIMITS. THE PROGRAM IS NOW RUNNING ALL TESTS IN SEQUENCE.
- D. IF THERE ARE NO ERRORS, AT THE END OF THE PASS (APPROX. 50 SECONDS RUN TIME), A "D" WILL BE TYPED AND IT WILL CONTINUE ON FOR ANOTHER PASS.
- E. TO HALT THE TEST AT ANY TIME (AFTER OR BEFORE COMPLETION OF A PASS) JUST HALT THE PROCESSOR.
- F. AFTER COMPLETING A PASS OF THE DIAGNOSTIC, THE RX11 RELIABILITY TEST MAY BE RUN.
- G. THERE ARE TWO TYPES OF ERROR PRINT OUT FORMATS
 - 1. TESTS PRETEST, 1 - 17, AND 21 - 23 USE THE FORMAT SHOWN IN SECTION 3.1. THE IMPORTANT ADDRESS THERE IS THE "ERADR" (ERROR ADDRESS) GO TO THE LISTING AT THAT LOCATION TO GET MORE INFORMATION ON THE ERROR CONDITION
 - 2. TEST 20, AND 24 - 26 USE THE FORMATT SHOWN IN SECTION 3.3. IN THIS CASE THE "TEST PC" IS THE ADDRESS OF THE TEST BEING RUN WHEN THE ERROR OCCURED. THEN THE VITAL INFORMATION OF THE ERROR IS PRINTED (CONTENTS OF ALL REGISTERS, ADDRESS OF WHERE ON THE DISKETTE THE ERROR OCCURED, AND THE TYPE OF ERROR).

2.1 LOADING THE PROGRAM

LOAD THE PROGRAM INTO MEMORY USING THE STANDARD PROCEDURE FOR BINARY PAPER TAPES.
MAKE SURE THE TOTAL SYSTEM IS READY FOR OPERATION. THE DISKETTES INSERTED PROPERLY, DOORS CLOSED ON DRIVES TO BE TESTED ETC.

2.2 STARTING ADDRESSES

THE PROGRAM HAS TWO STARTING ADDRESS LOCATIONS AS FOLLOWS:

2.2.1 INITIAL START [LOC.200]

THIS STARTING ADDRESS TESTS FOR AND SELECTS THE HARDWARE, OR SOFTWARE SWITCH REGISTER, PRINTS MAINDEC NAME AND REVISION, THE TEST AND DRIVE SELECTION, AND TRACKS AND SECTORS BEING USED.

2.2.2 RESTART [LOC.202]

THIS STARTING ADDRESS DIRECTS THE PROGRAM TO CONTINUE RUNNING USING THE DRIVE AND TEST SELECTIONS SPECIFIED IN THE PREVIOUS INITIAL START.

2.3 OPERATOR ACTION BEFORE STARTING THE PROGRAM

2.3.1 DEVICE ADDRESS SELECTION

LIKE MOST OPTIONS ON THE PDP-11 THE RX11 INTERFACE CARD HAS JUMPERABLE REGISTER AND VECTOR ADDRESSES. THIS ALLOWS FOR DEVICE! 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 REGISTER AND VECTOR ADDRESSES THAT ALL COMMUNICATION BETWEEN THE PDP-11 AND THE RX11 IS HANDLED.

IF THE RX11 SYSTEM UNDER TEST IS JUMPED FOR REGISTER ADDRESSES OTHER THAN STANDARD, WHICH IS RXCS = 177170 AND RXDB = 177172 PLACE IN THE MEMORY LOCATION CALLED "RXCS" [LOC. 1206] ITS NEW ADDRESS. THE PROGRAM ASSUMES THE NEXT EVEN ADDRESS ABOVE THAT OF RXCS, WILL BE THE ADDRESS OF RXDB, SO SETTING THAT ADDRESS IS NOT NECESSARY. IF THERE IS A NONSTANDARD INTERRUPT VECTOR ADDRESS (STANDARD IS LOC. 264) THEN PLACE IN MEMORY LOCATION CALLED "KRXVEC" [LOC. 1204] ITS NEW ADDRESS.

IF EITHER OF THESE LOCATIONS IS LOADED WITH A WRONG ADDRESS, THE PROGRAM WILL GET UNPREDICTABLE ERRORS AND MAY HALT.

NOTE: THE PROGRAM EXPECTS THAT THE PRIORITY LEVEL JUMPERS ARE SET FOR A NORMAL 'BR' LEVEL OF 5 (CONTENTS OF PROGRAM LOCATION 'BRLEV:' IS SET TO 5). IF THE PRIORITY LEVEL JUMPERS ARE SET TO ANY OTHER LEVEL TESTS 3 & 4 WILL REPORT ERRORS, UNLESS PROGRAM LOCATION 'BRLEV:' HAS BEEN PATCHED TO CONTAIN THE RELEVANT 'BR' LEVEL BEFORE EXECUTING THE PROGRAM.

IF THIS IS BEING TESTED ON A LSI 11, TESTS 3 AND 4 WILL NOT BE RUN AS THE LSI 11 HAS ONLY 1 LEVEL OF INTERRUPT.

2.3.2 NON-STANDARD DISKETTE ADDRESS SELECTION

IF IT IS DESIRABLE TO TEST THE DISKETTE BETWEEN TRACK AND SECTOR ADDRESS LIMITS OTHER THAN THE PRESELECTED TRACK ADDRESSES, AND/OR MINIMUM (FIRST) AND MAXIMUM (LAST) SECTOR ADDRESSES. THIS IS DONE BY THE OPERATOR MAKING CHANGES TO TWO MEMORY LOCATIONS BEFORE THE PROGRAM IS STARTED. ONE LOCATION IS CALLED "OD" (LOC. 1200) WHICH CONTAINS THE TWO BYTES FOR INNER AND OUTER TRACK ADDRESSES. THE OTHER LOCATION IS CALLED "FIRST" AND IT CONTAINS THE TWO BYTES FOR THE FIRST AND LAST SECTOR ADDRESSES.

A. DEFINITIONS

OD = ADDRESS OF TRACK AT OUTER DIAMETER (MIN. 0)
 ID = ADDRESS OF TRACK AT INNER DIAMETER (MAX. 114)
 FIRST = ADDRESS OF FIRST SECTOR ON A TRACK (MIN. 1)
 LAST = ADDRESS OF LAST SECTOR ON A TRACK (MAX. 32)

B. LOCATIONS

TRACKS LOCATION 1200	BITS	14----	8	6----	0
		ID		OD	

SECTORS LOCATION 1202	BITS	12----	8	4----	0
		LAST		FIRST	

C. RESTRICTIONS

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

IF THESE LOCATIONS ARE CHANGED TO NEW LIMITS, THEN THE PROGRAM WILL ACCESS ONLY THOSE ADDRESSES INCLUSIVE OF AND BETWEEN THESE LIMITS. THE EXCEPTION TO THIS IS TEST 26 WHICH ALWAYS USES A SPECIAL TRACK SEQUENCE.

IF THE "OD" LOCATION IS CLEARED OR SET TO ANY ILLEGAL COMBINATION OF TRACKS, THE PROGRAM WILL CLEAR LOCATION "OD". THE TRACK SEQUENCE WILL THEN BE TRACKS 0, 52, 53, AND 114 (OCTAL) ONLY.

IF THE "FIRST" LOCATION IS CLEARED OR SET TO ANY ILLEGAL COMBINATION OF SECTOR ADDRESS LIMITS THEN THE PROGRAM WILL SET "FIRST" TO 1 AND "LAST" TO 32 (OCTAL).

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 HARDWARE 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 MADE BEFORE THE PROGRAM STARTS.
 LOCATION "DTESTP" (LOC. 1212) IS WHERE THE BITS ARE SET TO TELL THE
 PROGRAM WHAT DRIVES ARE WANTED AND WHAT TESTS TO RUN AS INDICATED BELOW.
 WHEN THE PROGRAM STARTS IT WILL PRINT OUT THE CONDITIONS UNDER WHICH IT IS RUNNING.

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

NOTE: IF NEITHER OF THE ABOVE BITS ARE SET TO A 1, THEN
 THE PROGRAM EXPECTS BOTH DRIVES TO BE READY FOR OPERATION (POWER
 ON, DISKETTES INSERTED, AND DOORS CLOSED).

THEN SET THE TEST SELECTION IN BITS 4,3,2,1,AND 0 AS FOLLOWS:

"DTESTP" BITS	15	14	13-----5	4	3	2	1	0		
	U1	U0	NOT USED	TESTS						
BITS	4	3	2	1	0					TESTS
	0	0	0	0	0					(IF NO TEST SELECTED DEFAULTS TO TEST 1)
	0	0	0	0	1					TEST 1
	0	0	0	1	0					TEST 2
	0	0	0	1	1					TEST 3
	0	0	1	0	0					TEST 4
	0	0	1	0	1					TEST 5
	0	0	1	1	0					TEST 6
	0	0	1	1	1					TEST 7
	0	1	0	0	0					TEST 10
	0	1	0	0	1					TEST 11
	0	1	0	1	0					TEST 12
	0	1	1	0	0					TEST 13
	0	1	1	0	1					TEST 14
	0	1	1	1	0					TEST 15
	0	1	1	1	1					TEST 16
	1	0	0	0	0					TEST 17
	1	0	0	0	1					TEST 20
	1	0	0	1	0					TEST 21
	1	0	0	1	1					TEST 22
	1	0	1	0	0					TEST 23
	1	0	1	0	1					TEST 24
	1	0	1	1	0					TEST 25
	1	0	1	1	1					TEST 26

NOTE1: SELECTION OF TESTS 27 THROUGH 37 WILL CAUSE THE MESSAGE
 "ILLEGAL TEST" TO BE PRINTED.

NOTE2: WHEN A SPECIFIED TEST IS SELECTED THE PROGRAM WILL START AT
 THAT TEST AND THEN RUN THROUGH ALL THE FOLLOWING TESTS UNTIL
 IT COMPLETES TEST 26, INDICATED BY THE EOP TYPE OUT.
 THEN IT WILL GO BACK TO THE TEST SELECTED AND START THE
 NEXT PASS. (IE. IF TEST 24 IS SELECTED THE PROGRAM WILL RUN TEST 24,
 25, AND 26, THEN GO BACK TO TEST 24.)

AN EXPANDED DEFINITION OF THE TESTS IS IN SECTION 2.5

2.3.4.1 PREREQUISITE OF TESTS:

THE FOLLOWING TESTS MUST BE RUN IN ORDER, AS ONE TEST SETS UP FOR THE NEXT TEST.

TEST 6 BEFORE TESTS 7 AND TEST 10
TEST 14 BEFORE TEST 15 AND TEST 16
TEST 16 BEFORE TEST 17
TEST 21 BEFORE TEST 22 AND TEST 23

SEE SECTION 2.5 UNDER THE ABOVE TESTS FOR EXPLANATION

2.4 OPERATOR ACTION TO RUN THE PROGRAM

2.4.1 STARTING THE PROGRAM

DEPENDING UPON THE STARTING ADDRESS SELECTED THE PROGRAM WILL DO THE FOLLOWING:

SA200 (INITIAL START)

THE SELECTION OF HARDWARE OR SOFTWARE SWITCH REGISTER IS MADE THEN THE PROGRAM WILL TYPE ITS IDENTIFICATION NUMBER, THE TEST PARAMETERS SELECTED IN LOCATION "DTESTP", AND TRACKS AND SECTORS BEING TESTED. THE PROGRAM THEN PROCEEDS TO RUN UNDER THOSE CONDITIONS.

SA202 (RESTART)

THE PROGRAM WILL TYPE OUT THE TEST PARAMETERS SELECTED BY THE PREVIOUS INITIAL START, PRINTS THE DISKETTE ADDRESS LIMITS, AND STARTS RUNNING THE TESTS. THE ONLY OPERATOR ACTION REQUIRED IS TO SET THE OPERATING CONDITIONS AS DEFINED IN SECTION 2.4.2, AFTER DEPRESSING THE "LOAD ADRS" SWITCH AND BEFORE DEPRESSING THE START SWITCH.

AFTER THE TEST SELECTION HAS BEEN MADE PRESS THE "CONT" SWITCH. THE PROGRAM WILL THEN ASK FOR OPERATING CONDITIONS. SWITCHES 0 AND 8 THROUGH 15 ARE USED AS INDICATED BELOW. ONCE THEY ARE SET UP AGAIN DEPRESS THE "CONT" SWITCH. THE PROGRAM IS NOW RUNNING UNDER THE SELECTED CONDITIONS.

SW15-SWD (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.

SW15 (1) - HALT ON ERROR

THE PROGRAM HALTS ON DETECTING AN ERROR, AFTER PRINTING THE ERROR MESSAGE. PRESSING "CONT" RESTORES THE NORMAL OPERATION OF THE PROGRAM.

SW14 (1) - HALT AT END OF PASS

AT "END OF PASS" THE PROGRAM TYPES A BELL THEN AN EOP INDICATOR.

"D" MEANS NO ERRORS DURING THE PASS
 "--" MEANS HAD ERRORS DURING THE PASS

IF SW14 IS SET THE PROGRAM WILL HALT. IF SW14 IS OFF THE PROGRAM GOES BACK TO THE TEST SELECTED AND RECYCLES THROUGH TO THE LAST TEST, AT WHICH TIME ANOTHER EOP INDICATOR IS PRINTED. IF THE PROGRAM HALTS DUE TO SW14 THEN PRESS "CONT" WILL RESTORE THE NORMAL FLOW OF THE PROGRAM. IF IT HALTS AT THE END OF A PASS IT WILL TYPE OUT THE NUMBER OF PASSES COMPLETED.

SW13 (1) - INHIBIT ERROR TYPEOUT

AT THE DETECTION OF AN ERROR IF SW13 IS SET NO ERROR PRINT OUT WILL OCCUR. IF SW13 IS OFF THE ERROR INFORMATION IS PRINTED AS DESCRIBED IN SECTION 3.0 ERROR DETECTION

SW12 (1) - LOOP ON TEST

AT THE COMPLETION OF A TEST THE PROGRAM CHECKS SW12. IF SET THE PROGRAM WILL GO BACK TO THE BEGINNING OF THAT TEST AND RERUN IT. THIS PRODUCES A SCOPE LOOP ON A PARTICULAR TEST. THE PROGRAM WILL STAY IN THIS TEST UNTIL:

- A. HALT ON END OF TEST SWITCH IS SET
 - B. LOOP ON TEST SWITCH IS TURNED OFF
- AT WHICH TIME THE PROGRAM WILL GO ON TO THE NEXT TEST.

NOTE: IF SW12 IS SET AND NO TEST SPECIFIED (0) THE PROGRAM WILL LOOP ON TEST 1.

NOTE: TO LOOP ON A TEST THAT REQUIRES A PREVIOUS TEST TO BE RUN FIRST (SECTION 2.3.4). SELECT THE PREREQUISITE TEST AND SET THE "HALT AT END OF TEST" SWITCH. START THE PROGRAM AND WHEN IT HALTS, SELECT THE DESIRED TEST AND SET THE "LOOP ON TEST" SWITCH. THE PROGRAM WILL NOW STAY IN THAT TEST.

SW11 (1) - LOCK ON ERROR

IN SOME TESTS ERRORS CAN OCCUR IN SEVERAL PLACES THROUGH OUT THE TEST. WHEN THE ERROR HAS BEEN REPORTED THE PROGRAM SETS A PC FLAG TO INDICATE WHERE THE ERROR OCCURED. IF SW11 IS SET THE PROGRAM GOES BACK TO THE BEGINNING OF THE TEST RUNNING, AND GOES THROUGH THE TEST UNTIL:

- A. IT FINDS A DIFFERENT ERROR IN AN EARLIER PART OF THE TEST IN WHICH CASE IT WILL LOCK ONTO THAT ERROR.
- B. IT DETECTS THE PC FLAG INDICATING THIS IS WHERE THE ERROR OCCURED. IT THEN GOES BACK TO THE BEGINNING OF THE TEST AGAIN.

THIS LOOP WILL CONTINUE UNTIL HALT ON ERROR SWITCH IS SET OR THE LOCK ON ERROR SWITCH IS TURNED OFF.

SW10 (1) - HALT AT END OF TEST

WHEN SET IT WILL HALT THE PROGRAM AT THE END OF THE TEST PRESENTLY RUNNING.

SW 9 - LIMIT DATA ERROR PRINT OUTS

- (0) - WHEN OFF ONLY THE FIRST 10 DATA BYTE ERRORS WILL BE PRINTED ON A READ CHECK TEST, FOR EACH SECTOR. ANY MORE ERRORS WILL BE TABULATED BUT NOT PRINTED. AN ERROR ON A DIFFERENT SECTOR WILL ALLOW 10 MORE DATA BYTE ERRORS TO BE PRINTED.
- (1) - WHEN SET ALL DATA BYTE ERRORS FOR ALL SECTORS WILL BE PRINTED ON AN ERROR.

SW 8 (1) - INHIBIT RECALIBRATION

NO RECALIBRATION OF THE DRIVES WILL OCCUR UPON THE DETECTION OF A SEEK ERROR IF THIS SWITCH IS SET.

SW 0 (1) - INHIBIT BELL AT ERROR

IF SW0 IS OFF THE ERROR ROUTINE WILL RING THE TELEPRINTER BELL AT EACH ERROR DETECTED. WITH SW0 SET NO BELL WILL RING.

2.5 TEST DEFINITIONS

2.5.1 PRETEST - INITIALIZE [KEY] PART I

EACH TIME THE PROGRAM IS STARTED, BY EITHER STARTING ADDRESS, IT RUNS THROUGH A PRETEST.

KEY INITIALIZE SHOULD SET THE DONE FLAG BECAUSE ANY INITIALIZATION OF THE RX01 MICROPROCESSOR IS AN IMPLIED [READ SECTOR] OF TRACK 1 SECTOR 1. THEREFORE ANY ERROR, EXCEPT PARITY, THAT MAY OCCUR FROM A NORMAL [READ SECTOR] COMMAND MAY OCCUR DURING AN INITIALIZE, CAUSING THE ERROR FLAG TO SET.

PRETEST INSURES THAT:

- A. DONE IS SET
- B. ERROR FLAG IS CLEARED
- C. TR FLAG IS CLEARED
- D. INIT DONE IS SET

2.5.2 TEST 1 - RXCS TEST PART I / INTERRUPT TEST PART I

THE PURPOSE OF THIS TEST IS TO VERIFY THAT WRITING ALL RXCS WRITABLE BITS TO A 0 ARE NOT WRITTEN TO A 1.

THE PROGRAM WRITES THE RXCS = 0

NO INTERRUPTS SHOULD OCCUR

THE RXCS SHOULD REMAIN UNCHANGED = 40 (DONE)

THE RXDB SHOULD = 0

2.5.3 TEST 2 - INTERRUPT TEST PART II / VECTOR ADDRESS VERIFICATION

THE PURPOSE OF THIS TEST IS TO VERIFY THAT WRITING THE RXCS INTERRUPT ENABLE BIT (BIT 6) TO A 1, DOES INDEED WRITE IT TO A 1, THEREFORE BECAUSE DONE IS SET AN INTERRUPT SHOULD OCCUR (THE PDP 11 PRIORITY IS 0)

2.5.4 TEST 3 - INTERRUPT TEST PART III / PRIORITY LEVEL TEST PART I

THE PURPOSE OF THIS TEST IS TO VERIFY THE PRIORITY OF THE INTERRUPT REQUEST LINE. THE PROGRAM SETS THE PDP-11 PRIORITY TO 4

AN RXD1 INTERRUPT SHOULD OCCUR ON PRIORITY LEVEL 5

IF NO INTERRUPT OCCURS THEN THE PRIORITY LEVEL OF THE RX11 IS NOT 5, BUT MAYBE LEVELS 4, 3, 2, OR 1

2.5.5 TEST 4 - INTERRUPT TEST PART IV / PRIORITY TEST PART II

THE PURPOSE OF THIS TEST IS TO VERIFY THE PRIORITY OF THE RX11 INTERRUPT REQUEST LINE. THE PROGRAM SETS THE PDP-11 PRIORITY TO 5.

NO INTERRUPT SHOULD OCCUR

IF AN INTERRUPT DOES OCCUR THEN THE PRIORITY LEVEL OF THE RX11 IS NOT LEVEL 5, BUT MAYBE LEVEL 6, OR 7.

2.5.6 TEST 5 - INIT (PROGRAMMED) B / READ STATUS

THE PURPOSE OF THIS TEST IS TO VERIFY THAT SETTING THE RX11 BIT 14 CAUSES A RXD1 PROGRAMMED SUBSYSTEM INITIALIZE

THE RXCS SHOULD = 40 (DONE)

THE RXDB SHOULD = 4, OR 104, OR 204, OR 304

TEST 5 CONT'D - RXCS TEST PART I: / RST

THE PURPOSE OF THIS TEST IS TO VERIFY THE READ STATUS COMMAND (FUNCTION #12)., AND THAT DONE BIT IS CLEARED BY THE FUNCTION.

2.5.7 TEST 6 - FILL BUFFER TRANSFER LENGTH TEST

THE PURPOSE OF THIS TEST IS TO VERIFY THE TRANSFER LENGTH OF THE FUNCTION "FILL BUFFER" OF THE RXD1 MICROCONTROLLER

NOTE: THIS TEST LOADS THE SECTOR BUFFER FOR TEST 7 AND 10, AND MUST BE RUN PREVIOUS TO THEM.

2.5.8 TEST 7 - EMPTY BUFFER TRANSFER LENGTH AND CONTENT VERIFICATION PART I

THE PURPOSE OF THIS TEST IS TO VERIFY THE TRANSFER LENGTH OF THE FUNCTION "EMPTY BUFFER" AND TO VERIFY THE CONTENTS OF THE SECTOR BUFFER.

2.5.9 TEST 10 - EMPTY BUFFER TRANSFER LENGTH AND CONTENT VERIFICATION PART II

THE PURPOSE OF THIS TEST IS TO VERIFY THE PREVIOUS EMPTY BUFFER TEST DID NOT EMPTY AND DESTROY THE CONTENTS OF THE SECTOR BUFFER.

2.5.10 TEST 11 - FILL / EMPTY BUFFER WITH ALL 0'S

DURING THE EMPTY BUFFER FUNCTION THIS TEST VERIFIES THAT ALL 0'S ARE IN FACT IN THE SECTOR BUFFER.

2.5.11 TEST 12 - FILL / EMPTY BUFFER WITH ALL 1'S

DURING THE EMPTY BUFFER FUNCTION THIS TEST VERIFIES THAT ALL 1'S ARE IN FACT IN THE SECTOR BUFFER.

2.5.12 TEST 13 - DRIVE READY VERIFICATION

TESTS THAT THE DRIVE READY (RDY) BIT WILL SET FOR ALL SELECTED DRIVES. THE RDY BIT WILL BE SET AFTER A READ STATUS FUNCTION DIRECTED TO THE SELECTED DRIVE.

2.5.13 TEST 14 - ERROR FLAG AND B-CODE VERIFICATION PART I

THE PURPOSE OF THIS TEST IS TO VERIFY THAT TRYING TO READ A NON-EXISTANT SECTOR WILL CAUSE AN ERROR AND THE CORRECT ERROR CODE WILL BE PUT INTO THE RXDS WHEN THE STATUS B IS READ.
NOTE: THIS TEST CHECKS FOR PARITY ERROR ON THE READ STATUS B FUNCTION. THE NEXT TWO TESTS (T15 & T16) DO NOT. THIS TEST MUST BE RUN BEFORE TESTS 15 & 16.

2.5.14 TEST 15 - ERROR FLAG AND B-CODE VERIFICATION PART II

THIS TEST VERIFIES THAT TRYING TO WRITE DELETED DATA ON AN ILLEGAL SECTOR WILL PRODUCE AN ERROR AND THE CORRECT B-CODE IS PRODUCED. THE DELETED DATA BIT SHOULD BE SET AFTER THIS TEST.

2.5.15 TEST 16 - ERROR FLAG AND B-CODE VERIFICATION PART III

VERIFIES THAT A WRITE FUNCTION TO A NONEXISTANT SECTOR WILL PRODUCE AN ERROR AND THE CORRECT B-CODE IS PRODUCED. THE DELETED DATA BIT WILL ALSO BE CLEARED.

NOTE: TEST 16 MUST BE RUN BEFORE TEST 17 AS TEST 16 CLEARS THE DELETED DATA BIT AND TEST 17 TESTS THAT IT IS CLEARED.

2.5.16 TEST 17 - ILLEGAL TRACK ERROR VERIFICATION

THIS TEST VERIFIES THAT IF A TRACK ADDRESS LARGER THAN 114(OCTAL) IS ACCESSED, AN ERROR CONDITION WILL OCCUR, AND THE B-CODE WILL = 40. IT ALSO EXPECTS THE DELETED DATA BIT TO BE CLEARED.

2.5.17 TEST 20 - SEEK VERIFICATION VIA READ FUNCTION

THIS TEST DOES A READ FUNCTION ON THE SELECTED TRACKS TESTING FOR SEEK ERRORS ON VARIOUS SECTIONS OF THE DISKETTE.

2.5.18 TEST 21 - WRITE TEST

THE PURPOSE OF THIS TEST IS TO WRITE ALL ONES ON SECTOR 1, TRACK 1, AND TO VERIFY THAT THE DATA IN THE SECTOR BUFFER IS NOT CHANGED.
NOTE: THIS TEST MUST BE RUN BEFORE TESTS 22 & 23 AS THEY CHECK FOR DATA WRITTEN ON TRACK 1 SECTOR 1.

2.5.19 TEST 22 - INITIALIZE IMPLIED READ

AFTER PREVIOUSLY WRITING DATA ON TRACK 1 SECTOR 1, THIS TEST CHANGES THE CONTENTS OF THE SECTOR BUFFER AND DOES A PROGRAMMED INITIALIZE. AT THE END OF AN INIT.(RECAL.) THE SECTOR BUFFER MUST CONTAIN THE DATA FROM TRACK 1 SECTOR 1.
NOTE: UNIT 0 MUST BE ON-LINE FOR THIS TEST TO WORK.

2.5.20 TEST 23 - READ TEST

THIS TEST VERIFIES THAT A READ FUNCTION DOES INFAC LOAD THE SECTOR BUFFER WITH DATA READ FROM THE SELECTED ADDRESS.

2.5.21 TEST 24 - DATA TRANSFER AND VERIFICATION

THE PURPOSE OF THIS TEST IS TO WRITE THEN READ AND CHECK DATA ON ALL SECTORS OF THE SELECTED TRACKS. THE TEST ALTERNATES BETWEEN DRIVES. IF BOTH DRIVES ARE SELECTED, BEFORE CHANGING TRACKS. THE DATA PATTERN USED IS A FLOATING 0 PATTERN.

2.5.22 TEST 25 - DATA VERIFICATION VIA DELETED DATA MODE.

THIS TEST IS THE SAME AS TEST 24 EXCEPT IT CHECKS FOR DELETED DATA INDICATORS AND USES A DATA PATTERN OF FLOATING 1.

2.5.23 TEST 26 - HEAD "HOME" TEST

THIS TEST CHECKS FOR THE "HOME FOUND BEFORE THE DESIRED TRACK WAS REACHED" ERROR CODE. THE HEAD IS MOVED OUT 10 TRACKS THEN DECREMENTED BACK TO TRACK 0. IT TESTS ALL SELECTED DRIVES, AND USES A DATA PATTERN OF RANDOM DATA.

3.0 ERRORS

PRETEST AND TESTS 1 - 17, AND TESTS 21 - 23 HANDLE ERRORS AS INDICATED IN SECTION 3.1. FOR THE MOST PART THESE TESTS DO NOT RELY ON AN INTERRUPT TO INDICATE THE FUNCTION IS COMPLETED. WHEREAS THE OTHER TESTS (TESTS 20, AND 24 - 26) DO READ, WRITE AND READ CHECK FUNCTIONS OVER THE SELECTED TRACK, SECTORS, AND DRIVES. THESE REQUIRE THE INTERRUPT SERVICE AND ERROR DETECTION THAT WAS USED IN THE DATA RELIABILITY TEST. THIS IS DESCRIBED IN SECTION 3.3.

NOTE: IF LOOP ON ERROR SWITCH IS UP THEN THE PROGRAM WILL LOOP ON THE SHORTEST SET OF INSTRUCTIONS THAT WILL KEEP IT IN THE FAILING LOOP. OTHERWISE AFTER REPORTING THE ERROR THE PROGRAM WILL CONTINUE RUNNING THROUGH THE REMAINING ADDRESSES AND TESTS.

3.1 ERROR HEADING FOR TESTS 1 - 17, AND 21 - 23 PLUS PRETEST.

THE ERROR HEADING IS AS FOLLOWS:

ERADR	FAST	FAPT	[BLANK]	GOOD	BAD
-------	------	------	---------	------	-----

UNDER EACH COLUMN THE ERROR ROUTINE PRINTS PERTINENT INFORMATION.

ERADR = ERROR ADDRESS
ADDRESS OF THE ERROR TRAP INSTRUCTION WHERE
THE ERROR WAS DETECTED.

FAST = FIRST ADDRESS OF SELECTED TEST
ADDRESS OF THE TEST SELECTED AND RUNNING

FAPT = FIRST ADDRESS OF PRESENT TEST
ADDRESS OF THE TEST OR SUBTEST PRESENTLY RUNNING, OR
ADDRESS OF THE SCOPE LOOP.

[BLANK] ADDITIONAL GENERAL INFORMATION SUPPLIED BY SOME
TESTS ON AN ERROR.

GOOD = EXPECTED RESULTS OF THE TEST
TEST RESULTS OF WHAT SHOULD HAVE HAPPENED IF
THERE WAS NO ERROR.

BAD = ACTUAL TEST RESULTS
THE DATA THAT WAS RECEIVED FROM THE RXD1,
THAT CAUSED THE ERROR.

PASS = NUMBER OF PASSES MADE UP TO THIS ERROR

3.2 ERROR OUTPUT PER TEST

THE FOLLOWING ARE THE TYPES OF PRINT OUTS UNDER THE COLUMNS
[BLANK], GOOD, AND BAD FOR THE VARIOUS TESTS, USING THIS ERROR FORMAT.

TEST (SECTION)	[BLANK] (R2)	GOOD (R0)	BAD (R1)
----	-----	----	----
PRETEST (1)	N/A	40	(RXCS)
PRETEST (2)	(RXCS) INCL. DO BIT	4 OR 204	(RXCS) NO DO BIT
TEST 1 (1)	N/A	40	(RXCS)
TEST 1 (2)	N/A	0	(RXCS)
TEST 1 (3)	(KRXVEC)	N/A	N/A
TEST 2 (1)	(KRXVEC)	N/A	N/A
TEST 2 (2)	(KRXVEC)	140	(RXCS)
TEST 2 (3)	(KRXVEC)	40	(RXCS)
TEST 2 (4)	(KRXVEC)	40	(RXCS)
TEST 2 (5)	(KRXVEC)	40	(RXCS)
TEST 3 (1)	(KRXVEC)	N/A	N/A
TEST 4 (1)	(KRXVEC)	N/A	N/A
TEST 5 (1)	N/A	40	(RXCS)
TEST 5 (2)	(RXDB) INCL. DO BIT	4 OR 204	(RXDB) NO DO BIT
TEST 5 (3)	N/A	0	(RXCS)
TEST 5 (4)	N/A	40	(RXCS)
TEST 5 (5)	(RXCS) INCL. DO BIT	200	(RXCS) NO DO BIT
TEST 6 (1)	NO. OF XFERS	N/A	N/A
TEST 7 (1)	NO. OF XFERS	EXPEC. DATA	ACTUAL DATA
TEST 10 (1)	NO. OF XFERS	EXPEC. DATA	ACTUAL DATA

TEST 11&12 (1)	[USES TEST 6&7 TO FILL / EMPTY BUFFER]		
TEST 13 (1)	(RXDB)	200	(RXDB) NO DO BIT
TEST 13 (2)	(RXDB)	200	(RXDB) NO DO BIT
TEST 14 (1)	NO. OF TR'S	100040	(RXCS)
TEST 14 (2)	(RXDB)	0	(RXDB) NO DO BIT
TEST 14 (3)	(RXDB)	40	(RXCS)
TEST 14 (4)	N/A	70	(RXDB) ERROR CODE
TEST 15 (1)	NO. OF TR'S	100040	(RXCS)
TEST 15 (2)	N/A	100	(RXDB)
TEST 15 (3)	N/A	70	(RXDB) ERROR CODE
TEST 16 (1)	NO. OF TR'S	100040	(RXCS)
TEST 16 (2)	N/A	0	(RXDB)
TEST 16 (3)	N/A	70	(RXDB) ERROR CODE
TEST 17 (1A)	(RXDB)	0	(RXCS)
TEST 17 (1B)	N/A	100040	(RXCS)
TEST 17 (2)	N/A	0	(RXDB)
TEST 17 (3)	(RXDB)	40	(RXCS)
TEST 17 (4)	N/A	40	(RXDB) ERROR CODE
TEST 21 (1)	(RXES) STATUS A	NO. OF BYTE	(RXDB) STATUS B
TEST 21 (2)	[USES TEST 7 TO EMPTY BUFFER]		
TEST 22	[USES TEST 6 & 7 TO FILL AND EMPTY BUFFER]		
TEST 23	[USES TEST 6 & 21 TO FILL AND CHECK BUFFER]		

3.3 ERROR HEADING FOR TESTS 20, 24 - 26

AS PREVIOUSLY STATED THESE TESTS ACCESS ALL THE SELECTED SECTORS, TRACKS, AND DRIVES, AND RELY ON THE INTERRUPT SERVICE ROUTINE TO INDICATE THAT A FUNCTION IS COMPLETED OR AN ERROR OCCURED. ALL ERRORS, WITH THE EXCEPTIONS WHERE NOTED, WILL TYPE AS ITS FIRST OR SECOND LINE OF THE MESSAGE "ERROR CONDITIONS TEST PC = XXXX PASS = X".

THE TEST PC NUMBER IS THE STARTING ADDRESS OF THE TEST RUNNING, AND THE PASS NUMBER IS THE NUMBER OF PASSES MADE UP TO THE ERROR

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 WAS DETECTED. IT SHOWS THE CRC, PAR, ETC. ERRORS

STATUS B IS THE "DEFINITIVE ERROR CODES" THAT THE RXD1 DETECTED, THAT MAY HAVE CAUSED THE ERROR CONDITION. THESE ERROR CODES ARE DEFINED IN SECTION 3.3.4

THERE ARE THREE CATEGORIES OF ERRORS AS LISTED AND DESCRIBED BELOW.

3.3.1 NO ERROR FLAG ERRORS

THESE ARE ERRORS THAT CAN OCCUR BUT THE ERROR FLAG IN THE RXCS WILL NOT BE SET.

A. UNEXPECTED OR MISSING DELETED DATA BIT

THIS ERROR RESULTS WHEN THE PROGRAM EXPECTS AND DOESN'T SEE THE DD BIT ("D D MARK MISSING"), OR DOESN'T EXPECT AND FINDS THE DELETED DATA BIT SET ("UNEXPECTED D D MARK"). THE PROGRAM WILL TYPE OUT AT WHAT DISKETTE ADDRESS THIS OCCURED THEN CONTINUE TESTING.
NOTE: SEE SECTION 3.3.3 FOR OTHER CAUSES OF THIS ERROR.

B. DATA NO STATUS ERROR

THIS ERROR OCCURS DURING A READ CHECK WHEN THE DATA READ DOES NOT MATCH THE DATA IN THE MEMORY DATA BUFFER, AND THERE WAS NO CRC ERROR INDICATED. THIS MEANS THAT THE DATA WAS PROBABLY READ OFF THE DISKETTE CORRECTLY BUT THE TRANSFER BETWEEN THE SECTOR BUFFER AND THE RXDB IN THE RX11 PRODUCED BAD DATA.

THE ERROR MESSAGE WILL INCLUDE THE DISKETTE ADDRESS, "BYTE" NUMBER IN THE SECTOR, THE DATA READ FROM THE SECTOR BUFFER "BAD", AND THE EXPECTED DATA FROM THE MEMORY BUFFER "GOOD".

BYTE # BAD GOOD
(THE DATA PATTERNS ARE FORMATTED AS SHOWN)

0 (TRACK ADDRESS; BITS 6 - 0)
1 (UNIT NUMBER BIT 7)
(SECTOR ADDRESS BITS 4 - 0)

BYTES 2 - 125 CONTAIN THE SELECTED DATA PATTERN.

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

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

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

IF SWITCH 9 IS DOWN THEN ONLY 10 DATA ERRORS WILL BE PRINTED, AND AT THE END OF THE SECTOR THE "TOTAL READ CHECK ERRORS =" WILL BE TYPED. IF SWITCH 9 IS UP THEN ALL THE DATA ERRORS FOR THAT SECTOR WILL BE TYPED OUT.

C. 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 "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 RESTARTS.

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

D. UNKNOWN INTERRUPT

IF AN INTERRUPT OCCURS THROUGH THE RX11 INTERRUPT VECTOR ADDRESS AND NONE OF THE STATUS BITS ARE SET (DONE, ERROR, ETC.) THE PROGRAM WILL TYPE "UNKNOWN INTERRUPT" AND RETURN BACK TO THE PROGRAM TO CONTINUE THE FUNCTION. THE ERROR HEADING IS NOT PRINTED.

E. NO INTERRUPT AT DONE

THE PROGRAM EXPECTS AN INTERRUPT AT DONE ON THE FUNCTIONS OF THESE TESTS. IF AN INTERRUPT DOES NOT OCCUR AT DONE TIME THEN THE PROGRAM WILL TYPE OUT "NO INTERRUPT AT DONE ERROR" THEN GO INTO THE INTERRUPT SERVICE ROUTINE AS IF AN INTERRUPT DID OCCUR. AT THIS POINT OTHER ERRORS MAY BE PRINTED IF ANY ARE DETECTED.

3.3.2 ERROR FLAG ERRORS

THESE ERRORS ARE DETECTED AS THE RESULTS OF THE ERROR BIT BEING SET IN THE RXCS AT AN INTERRUPT.

A. PARITY ERROR

A PARITY ERROR RESULTS FROM AN INCORRECT TRANSFER OF A COMMAND WORD FROM THE RX11 INTERFACE TO THE RX01 MICRO-PROCESSOR CONTROLLER. THE PROGRAM WILL TYPE OUT THE CONTENTS OF THE COMMAND STATUS REGISTER (RXCS) SHOWING THE FUNCTION THAT FAILED, THE ADDRESS OF THE ERROR, CONTENTS OF STATUS A (RXES) WITH THE PARITY BIT SET, CONTENTS OF STATUS B (RXDB) WITH THE DEFINITIVE ERROR CODE OF 210 SET. THEN A "READ, WRITE, FILL BUFFER OR EMPTY BUFFER PARITY ERROR" WILL BE PRINTED. IF A PARITY ERROR OCCURS ON A "READ DEFINITIVE ERROR CODE" FUNCTION, THEN THE CONTENTS OF THE RXCS AND "PARITY ERROR" WILL BE TYPED OUT.

B. CRC ERRORS

ON ALL DATA TRANSFERS BETWEEN THE SECTOR BUFFER AND THE DISKETTE, A CRC WORD IS GENERATED AND CHECKED. IF AN ERROR IS DETECTED BY THE MICRO-PROCESSOR IN THIS CRC WORD THEN A CRC ERROR IS GENERATED. THE PROGRAM AGAIN TYPES OUT THE CONTENTS OF THE REGISTERS (RXCS CONTAINS FUNCTION, STATUS A WITH "CRC ERR" BIT SET, STATUS B WITH AN ERROR CODE OF 200). THEN IF IT IS A READ ONLY FUNCTION, OR A READ CHECK FUNCTION AND THERE WERE DATA ERRORS IT WILL TYPE OUT "DATA CRC ERRORS" THEN PRINT THE BAD BYTES IF ANY. IF IT WAS A READ CHECK FUNCTION AND THERE WERE NO DATA ERRORS IT WILL PRINT "CRC ERROR NO DATA ERROR".

C. SEEK ERRORS

ANY ERROR THAT PRODUCES A DEFINITIVE ERROR CODE BUT DOES NOT SET AN ERROR BIT IN STATUS A (RXDB AT END OF FUNCTION) IS LABELED A SEEK ERROR. SEE SECTION 3.3.4 FOR ERROR CODES AND MEANINGS.

THE SAME INFORMATION IS PRINTED FOR THESE ERRORS AS IN PARITY, OR CRC ERRORS, EXCEPT IT STATES THAT IT IS A "WRITE OR READ SEEK ERROR".

IF SWITCH B IS DOWN THEN AT EACH SEEK ERROR FOUND THE PROGRAM DOES AN INITIALIZE OF THE RXD1 SO IT WILL RECALIBRATE TO A KNOWN (HOME) POSITION. THE PROGRAM THEN GOES ON TO THE NEXT SECTOR OR TRACK AND CONTINUES TESTING, IF THE LOOP ON ERROR SWITCH IS OFF. (SEE SECTION 3.3.3 FOR ERRORS CAUSED BY PREVIOUS ERRORS.) IF THE LOOP ON ERROR SWITCH IS UP IT WILL RETRY THE FUNCTION AT THE SAME ADDRESS.

IF SWITCH B IS UP THEN NO "INITIALIZE" IS DONE AND THE PROGRAM LOOKS AT THE OTHER SWITCHES FOR OPERATING CONDITIONS. SEEK ERRORS ALSO PRINT THE TRACK ADDRESS THAT THE HEAD MOVED FROM AT THE TIME OF THE ERROR.

D. ERROR FLAG ERROR

IF THE ERROR FLAG IS NOT SET IN THE RXCS AND AN ERROR BIT IS SET IN STATUS A OR AN ERROR CODE IS SET IN STATUS B THEN THERE WAS AN ERROR BUT THE ERROR FLAG WAS NOT SET. THE MESSAGE "ERROR FLAG ERROR" IS PRINTED THEN THE PROGRAM CONTINUES TO TYPE OUT THE TYPE OF ERROR.

3.3.3 ERRORS RESULTING FROM PREVIOUS ERRORS

IF THERE IS A "WRITE SEEK ERROR" THE PROGRAM WILL GO ON TO THE NEXT ADDRESS WITHOUT WRITING ON THE ADDRESS WHERE THE ERROR OCCURED. (UNLESS THE LOOP ON ERROR SWITCH IS UP AND THE SEEK ERROR IS RECOVERED.) IF THE WRITE FUNCTION IS FOLLOWED BY A READ CHECK FUNCTION AND THE READ DOES NOT HAVE A SEEK ERROR AT THE SAME ADDRESS. THEN THERE MAY BE DATA ERRORS, OR UNEXPECTED OR MISSING DELETED DATA BIT ERRORS RESULTING FROM NO DATA BEING WRITTEN ON THAT ADDRESS BY THE PREVIOUS WRITE FUNCTION.

3.3.4 DEFINITIVE ERROR CODES

THE RX01 MICRO-PROCESSOR HAS DEFINED THE ERROR CODES AND MEANINGS WHICH ARE AVAILABLE TO THE PROGRAM BY ISSUING COMMAND #7 "READ DEFINITIVE ERROR CODE"
THE FOLLOWING ARE THE CODES AND THEIR MEANINGS

- 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 FOR INIT.
- 40 - TRIED TO ACCESS A TRACK GREATER THEN 76
- 50 - HOME FOUND BEFORE DESIRED TRACK WAS REACHED
- 60 - SELF DIAGNOSTIC ERROR
- 70 - DESIRED SECTOR NOT FOUND AFTER SAMPLING 52 HEADERS
- 100 - WRITE PROTECT ERROR
- 110 - MORE THEN 40 US 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 A HEADER, NO ERROR FLAG
- 150 - GOOD HEADER (NO CRC ERROR) BUT TRACK COMPARE ERROR
- 160 - ID ADDRESS MARK NOT FOUND IN TIME
- 170 - DATA MARK NOT FOUND IN TIME
- 200 - DATA CRC ERROR
- 210 - PARITY ERRORS

3.4 PROGRAM HUNG

IF THERE IS NO RESPONSE FROM THE RX11 WHILE WAITING FOR THE TRANSFER REQUEST (TR) FLAG OR THE DONE FLAG. THE PROGRAM WILL TYPE "DEVICE TEST HUNG 2 PC" (ONLY IF SW13 IS OFF) AND THEN GO ON TO THE NEXT TEST, OR THE BEGINNING OF THE PRESENT TEST.

4.0 HALTS

THE ONLY HALTS IN THE PROGRAM ARE THE SELECTABLE HALTS (EOP, EOT, AT ERROR), THE ILLEGAL VECTOR HALTS, AND THE ILLEGAL TEST SELECTION HALT.

NOTE: ONE ADDITIONAL 'HALT' EXISTS IN THE PROGRAM. IT OCCURS WHEN THE USER HAS LOADED HIS PROGRAM VIA THE 'RXDP' MONITOR (ON UNIT 0) AND ALSO REQUIRES TESTING OF UNIT 0. A PROMPT MESSAGE IS TYPED REMINDING THE USER TO REPLACE HIS LOAD MEDIUM WITH A SCRATCH DISKETTE BEFORE GOING ON. THE PROGRAM WILL WAIT FOR THE 'CONTINUE' SWITCH TO BE DEPRESSED.

5.0 FLOW CHARTS
FLOW CHART FOR RX11 DIAGNOSTIC INTERFACE

FLOW CHART

FLOW CHART FOR RX11 DIAGNOSTIC INTERFACE

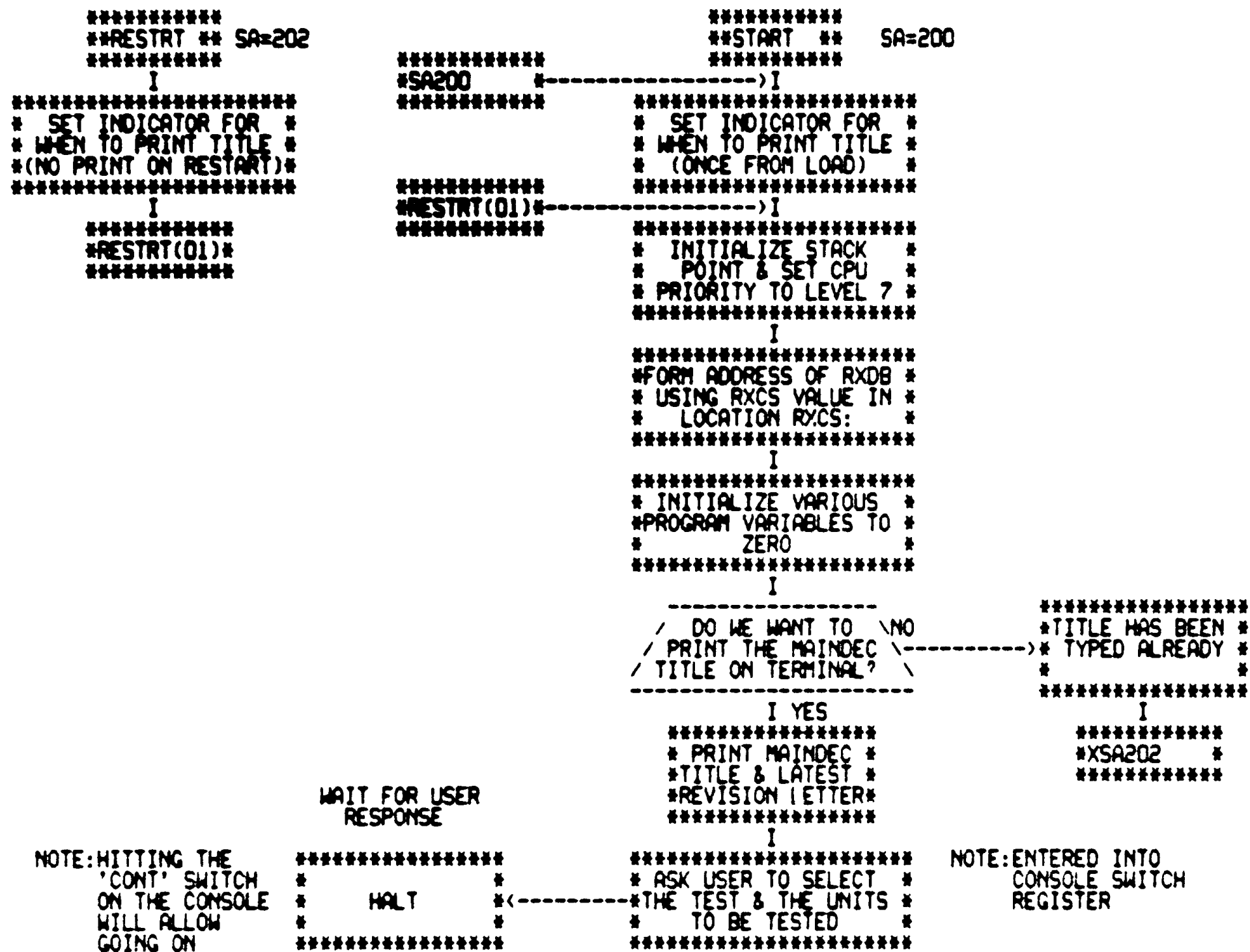
COPYRIGHT 1975
DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASS. 01754

TABLE OF CONTENTS

PAGE 08	RXCS TEST PART 1 & INTERRUPT PART 1
PAGE 09	INTERRUPT PART 2 & VECTOR VERIFICATION
PAGE 11	INTERRUPT PART 3 & PRIORITY CHECK PART 1
PAGE 13	PROGRAMMED INIT & READ 'B' CODE
PAGE 16	FILL BUFFER TRANSFER LENGTH
PAGE 17	EMPTY BUFFER TRANSFER LENGTH & CONTENTS CHECK
PAGE 18	FILL/EMPTY BUFFER ALL 0'S
PAGE 19	FILL EMPTY BUFFER ALL 1'S
PAGE 20	DRIVE READY VERIFICATION
PAGE 21	ERROR FLAG & 'B' CODE VERIFICATION PART 1
PAGE 22	ERROR FLAG & 'B' CODE VERIFICATION PART 2
PAGE 23	ERROR FLAG & 'B' CODE VERIFICATION PART 3
PAGE 24	ILLEGAL TRACK ERROR
PAGE 25	SEEK VERIFICATION VIA READ FUNCTION
PAGE 26	WRITE TEST
PAGE 28	INITIALIZE IMPLIED READ
PAGE 30	READ TEST
PAGE 31	DATA TRANSFER & VERIFICATION
PAGE 32	HEAD HOME TEST
PAGE 33	SUBROUTINE TO CALCULATE CHECKSUM
PAGE 34	SUBROUTINES FOR END OF PASS PRINTOUT
PAGE 35	SUBROUTINE FOR DONE CHECKING
PAGE 36	SUBROUTINE FOR PATTERN SELECTION
PAGE 37	SUBROUTINE FOR ILLEGAL SECTOR SELECTION
PAGE 39	SUBROUTINE FOR READ SEQUENCE

TABLE OF CONTENTS

PAGE 40	SUBROUTINE TO ACCESS DUAL UNITS
PAGE 42	SUBROUTINE TO VERIFY DATA WRITTEN
PAGE 44	SUBROUTINE FOR SELECTION OF UNIT
PAGE 45	EOP ROUTINES
PAGE 46	SUBROUTINE TO ISSUE COMMAND TO DRIVE
PAGE 47	TRACK INITIALIZATION ROUTINE
PAGE 48	SUBROUTINE FOR TRACK SELECTION
PAGE 49	READ SEQUENCE
PAGE 50	WRITE SEQUENCE
PAGE 52	READ CHECK SEQUENCE



 * STORE INFO. * PROGRAM WILL PICK UP
 * ENTERED INTO * HERE AFTER DEPRESSING
 * SW. REGISTER * 'CONT' SWITCH

WAIT FOR USER
 RESPONSE

NOTE: HITTING THE
 'CONT' SWITCH*
 ON THE FRONT *
 CONSOLE WILL *
 ALLOW GOING *
 ON

 * ASK USER TO SELECT *
 * OPERATING CONDITIONS *

NOTE: ENTERED INTO
 CONSOLE SWITCH
 REGISTER

 * ZERO THE UNITS * PROGRAM WILL PICK UP
 * SELECTED WORD * HERE AFTER DEPRESSING
 * (UNITSEL) * 'CONT' SWITCH

WERE ANY DRIVES
 SELECTED?

 * SET UNITS SEL- *
 * ECTED BITS IN *
 * UNITS SEL. WORD *

I YES

WAS UNIT 0
 SELECTED?

 * XSA202 *

I YES

WAS PROGRAM
 LOADED VIA UNIT0?

I NO

 * SET UNIT0 *
 * SELECTED BIT IN *
 * UNIT SEL. WORD *

I

 * SET I5 *

NO

WAIT FOR USER
 RESPONSE

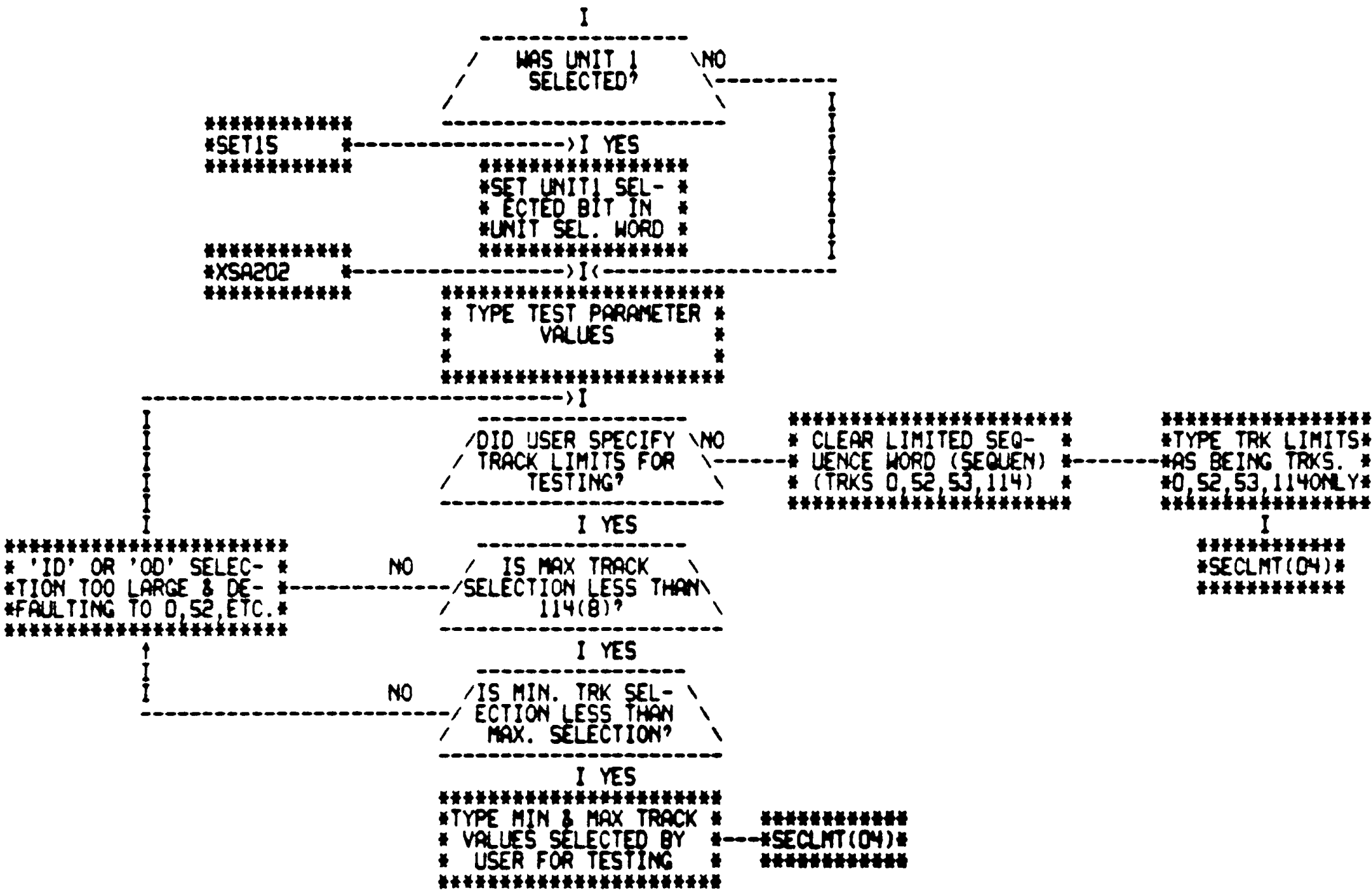
 * HALT *

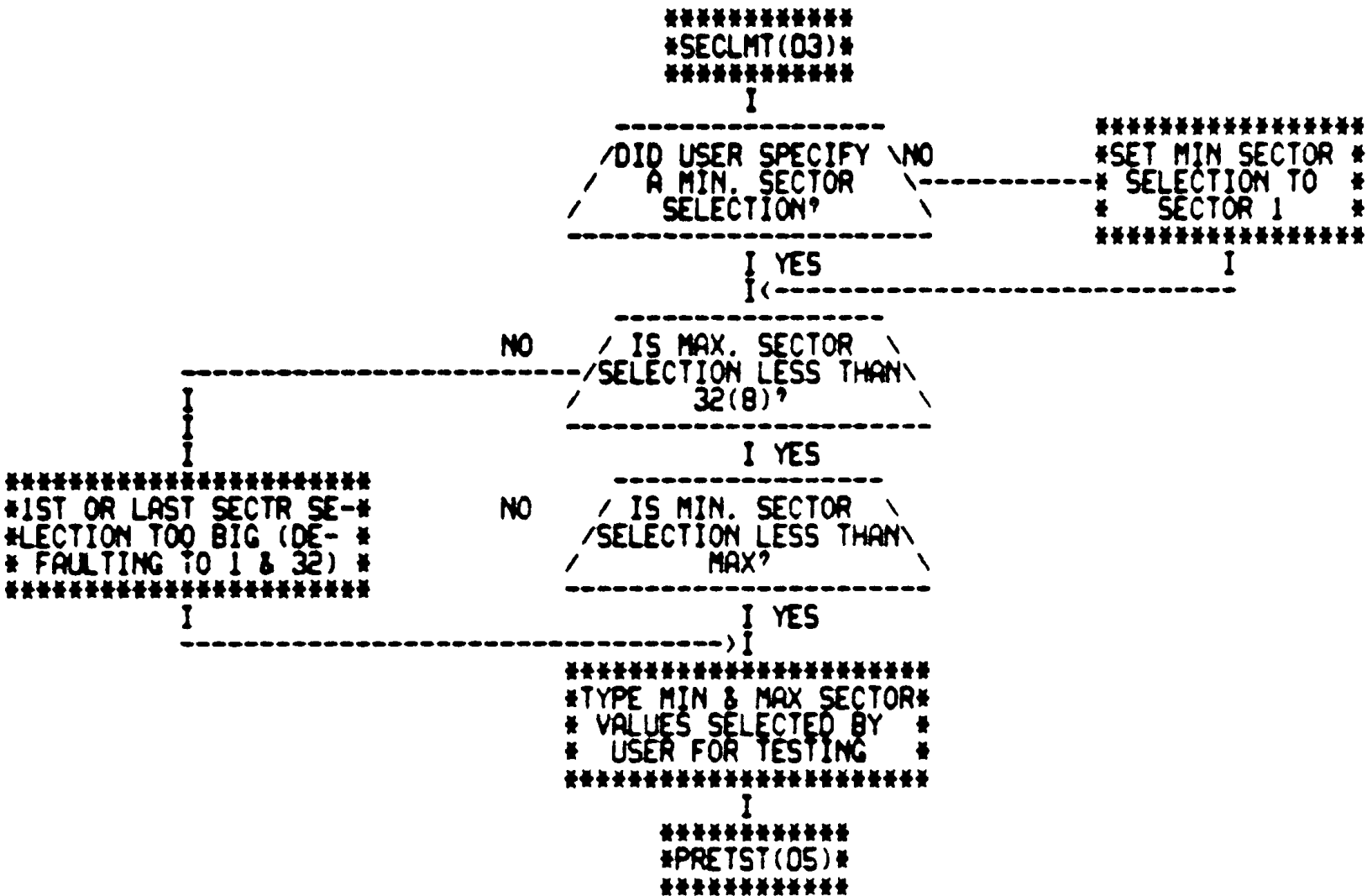
 * INFORM USER TO REMOVE *
 * LOAD MEDIUM FROM *
 * UNIT0 *

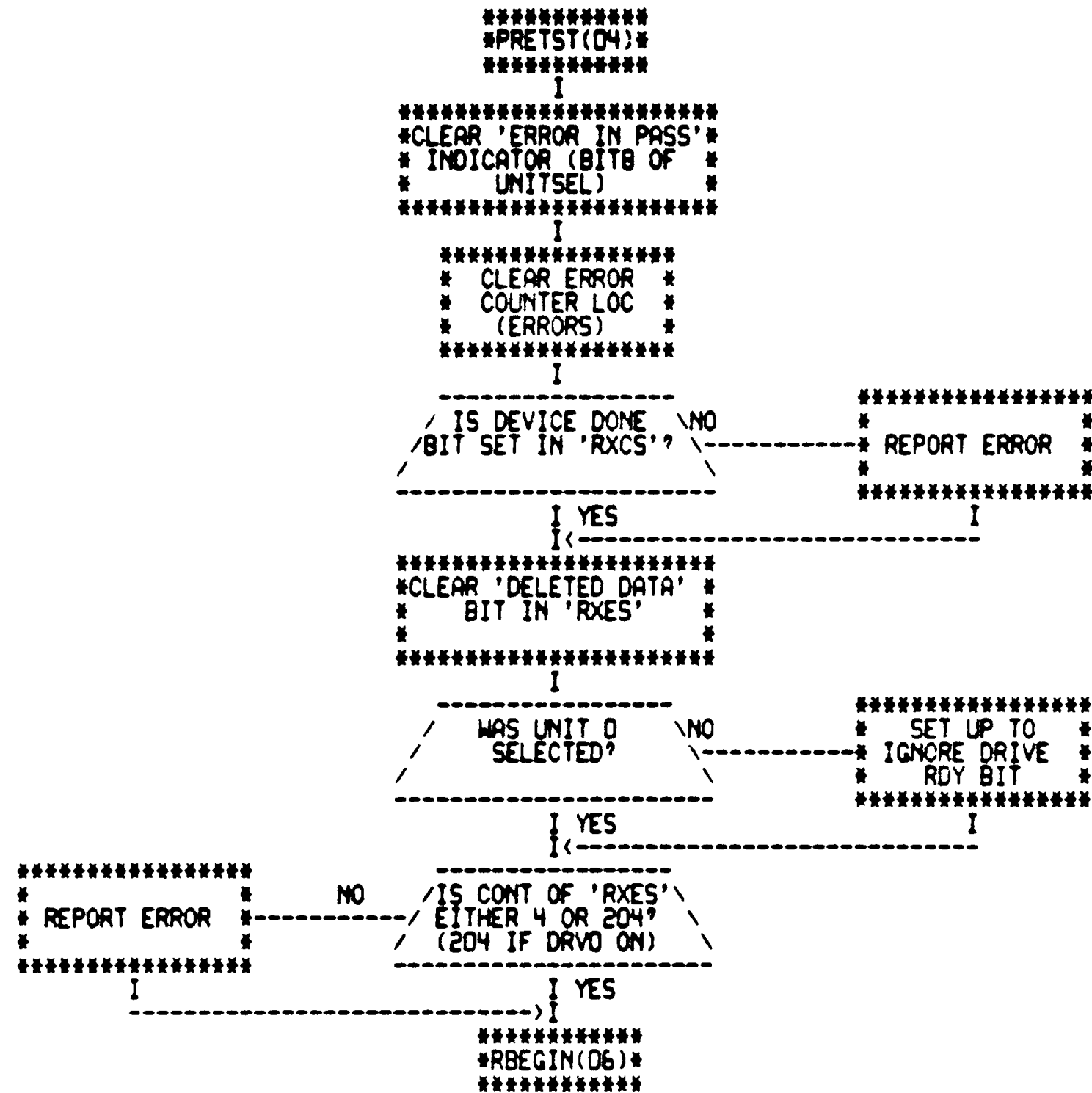
YES

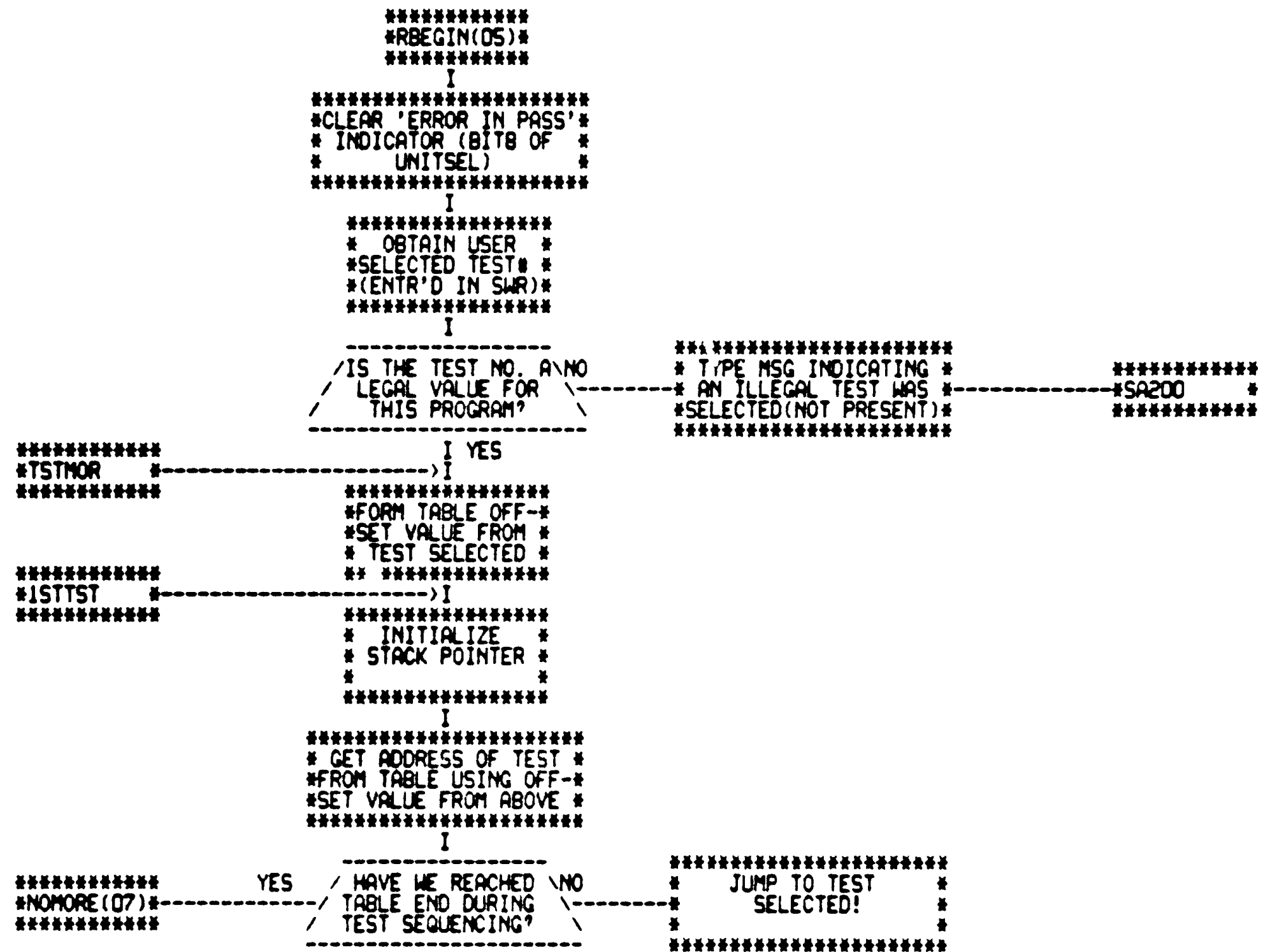
NOTE: HITTING THE
 'CONT' SWITCH
 ON THE CONSOLE
 WILL ALLOW GOING
 ON

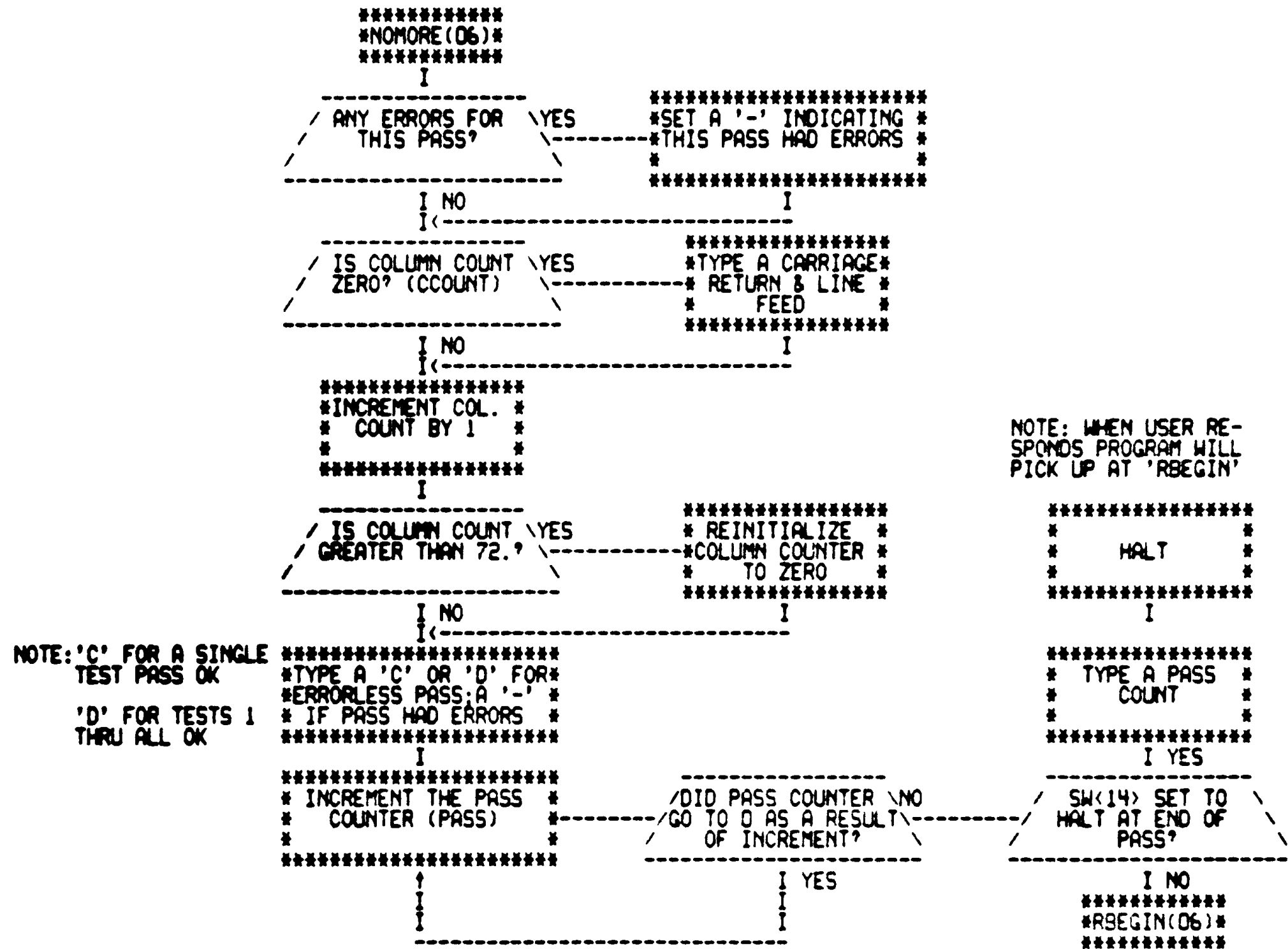
PROGRAM WILL PICK
 UP HERE AFTER DE-
 PRESSING THE 'CONT'
 SWITCH

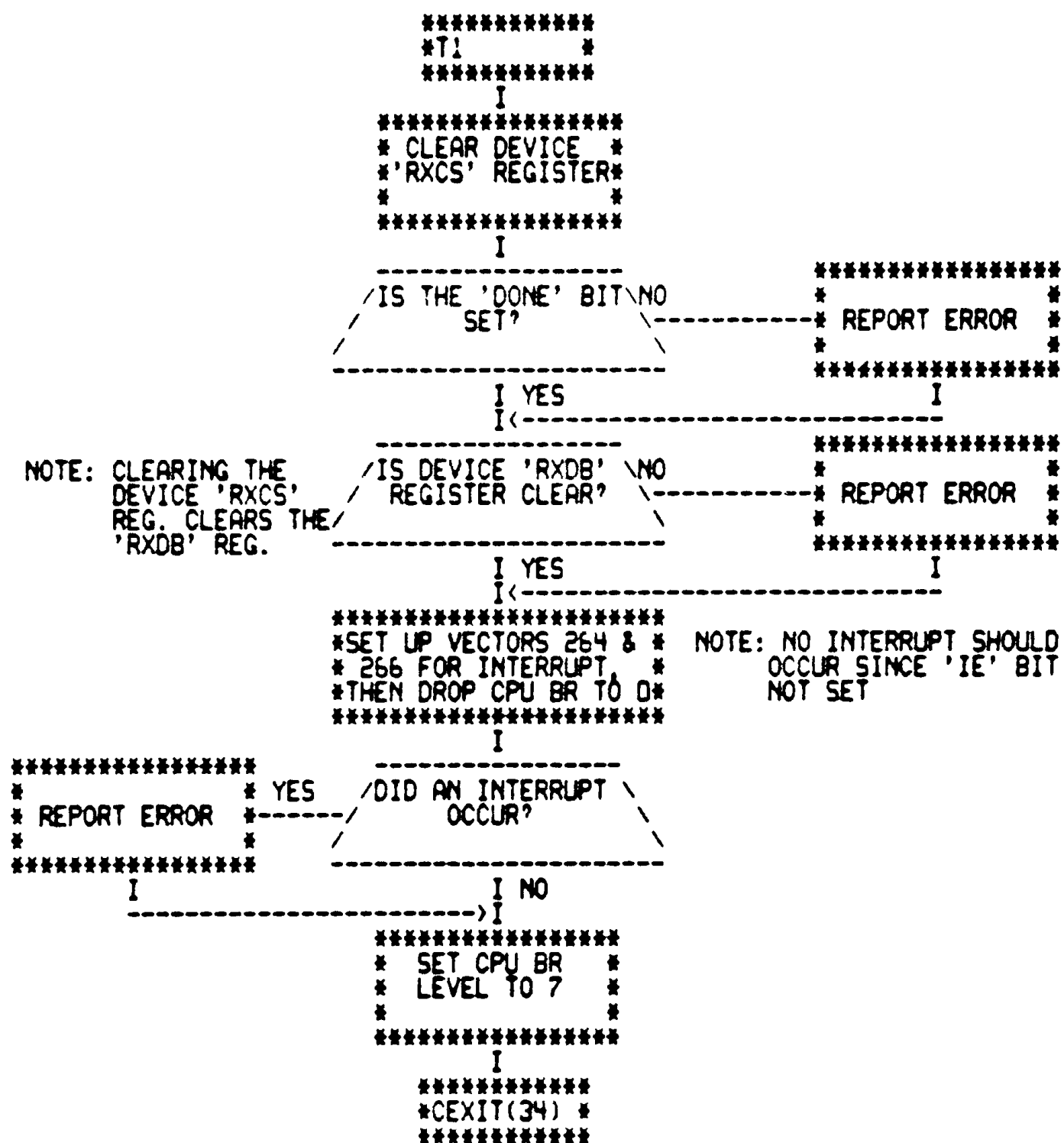


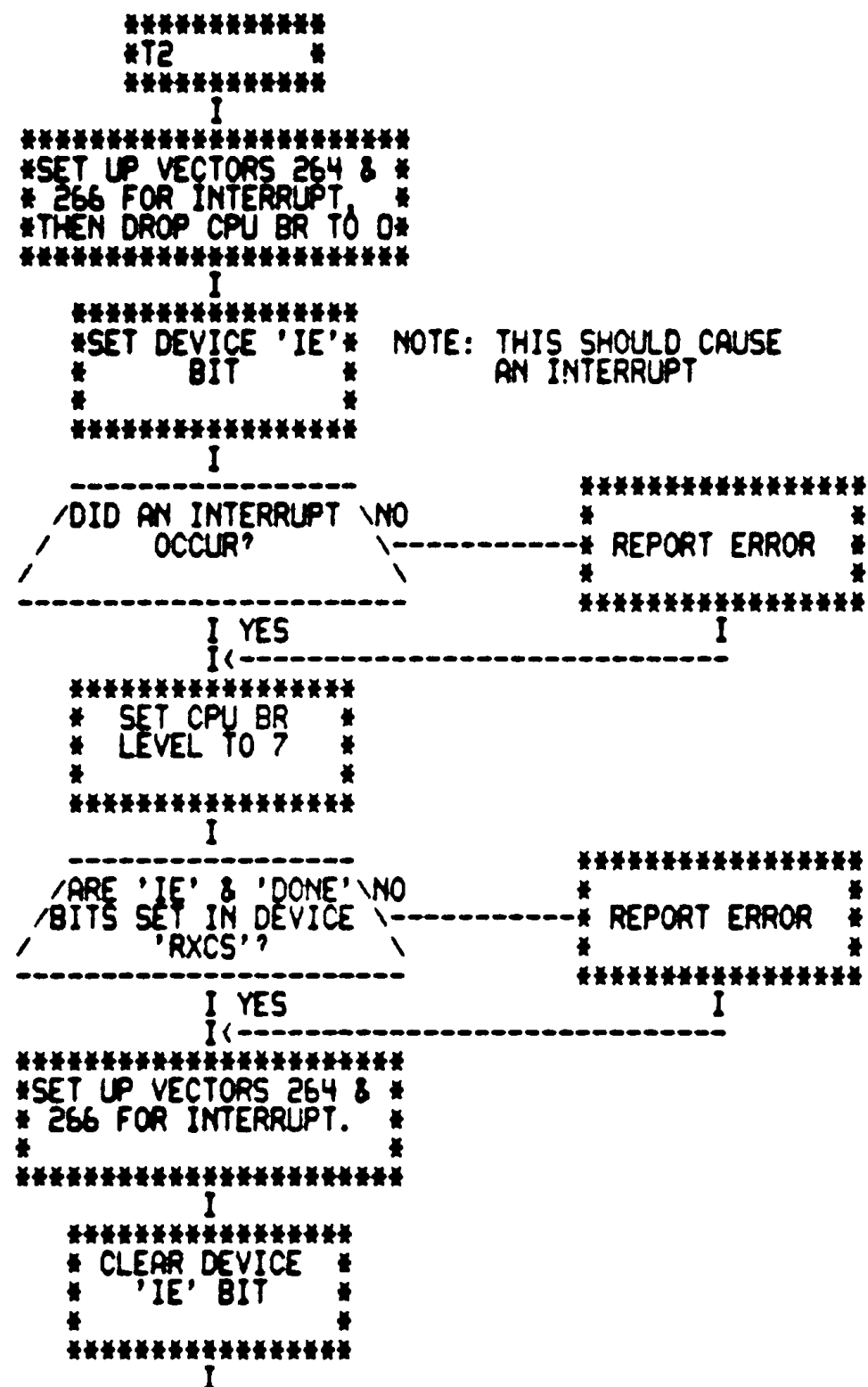


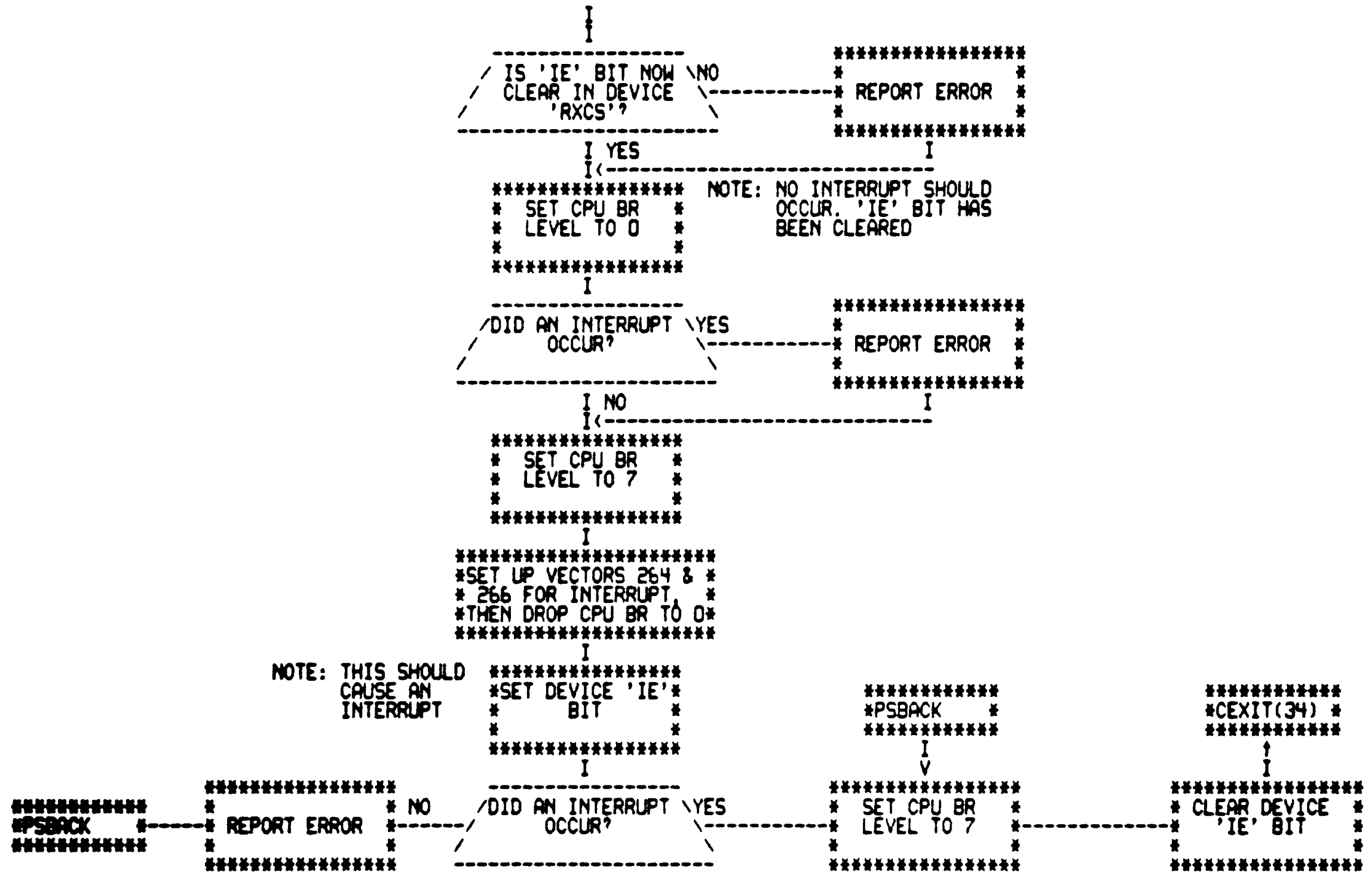












```

*****
*T3
*****
I
*****
*SET UP VECTORS 264 & *
* 266 FOR INTERRUPT *
*
*****
I
*****
* CALCULATE & SET CPU *
*BR LEVEL TO A VALUE 1*
* LESS THAN DEVICE *
*****
I
*****
*SET DEVICE 'IE' *
* BIT *
*
*****
NOTE: THIS SHOULD CAUSE
AN INTERRUPT
I
-----
/ DID AN INTERRUPT \ NO
\ OCCUR? /
-----
I YES
I <-----
*****
* SET CPU BR *
* LEVEL TO 7 *
*
*****
I
*****
* CLEAR DEVICE *
* 'IE' BIT *
*
*****
I
*****
*CEXIT(34) *
*****

```

```

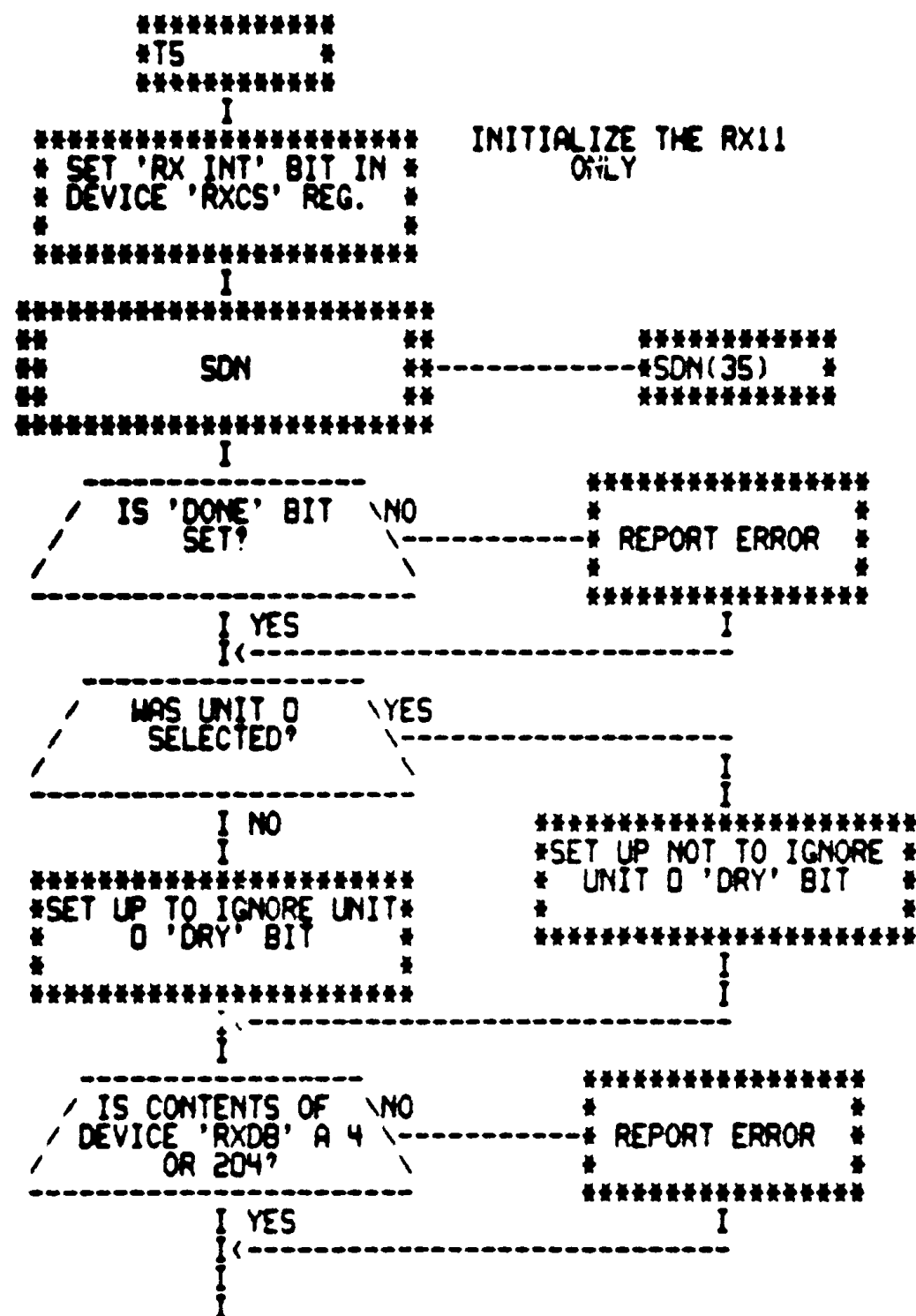
*****
*
* REPORT ERROR *
*
*****

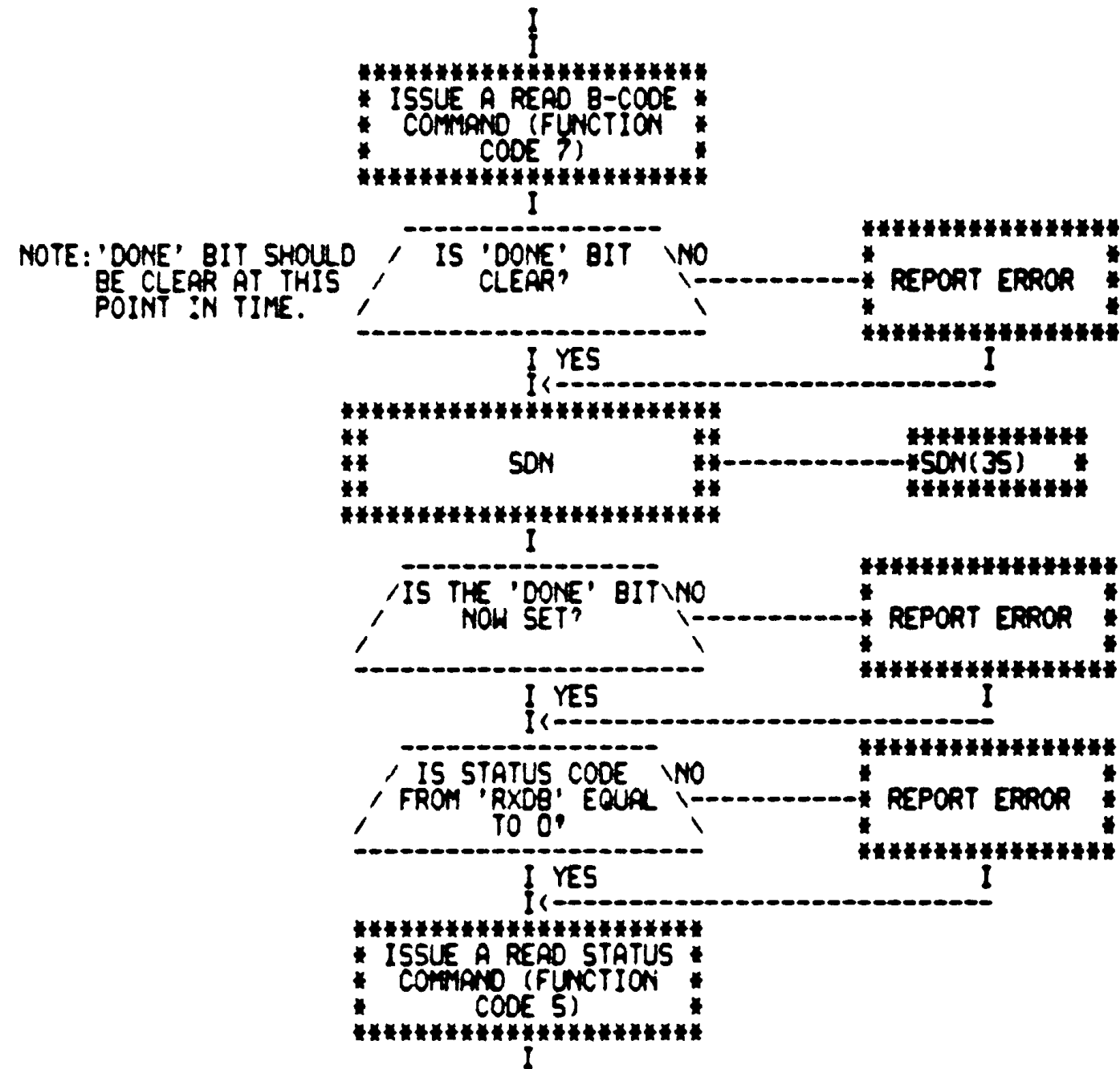
```

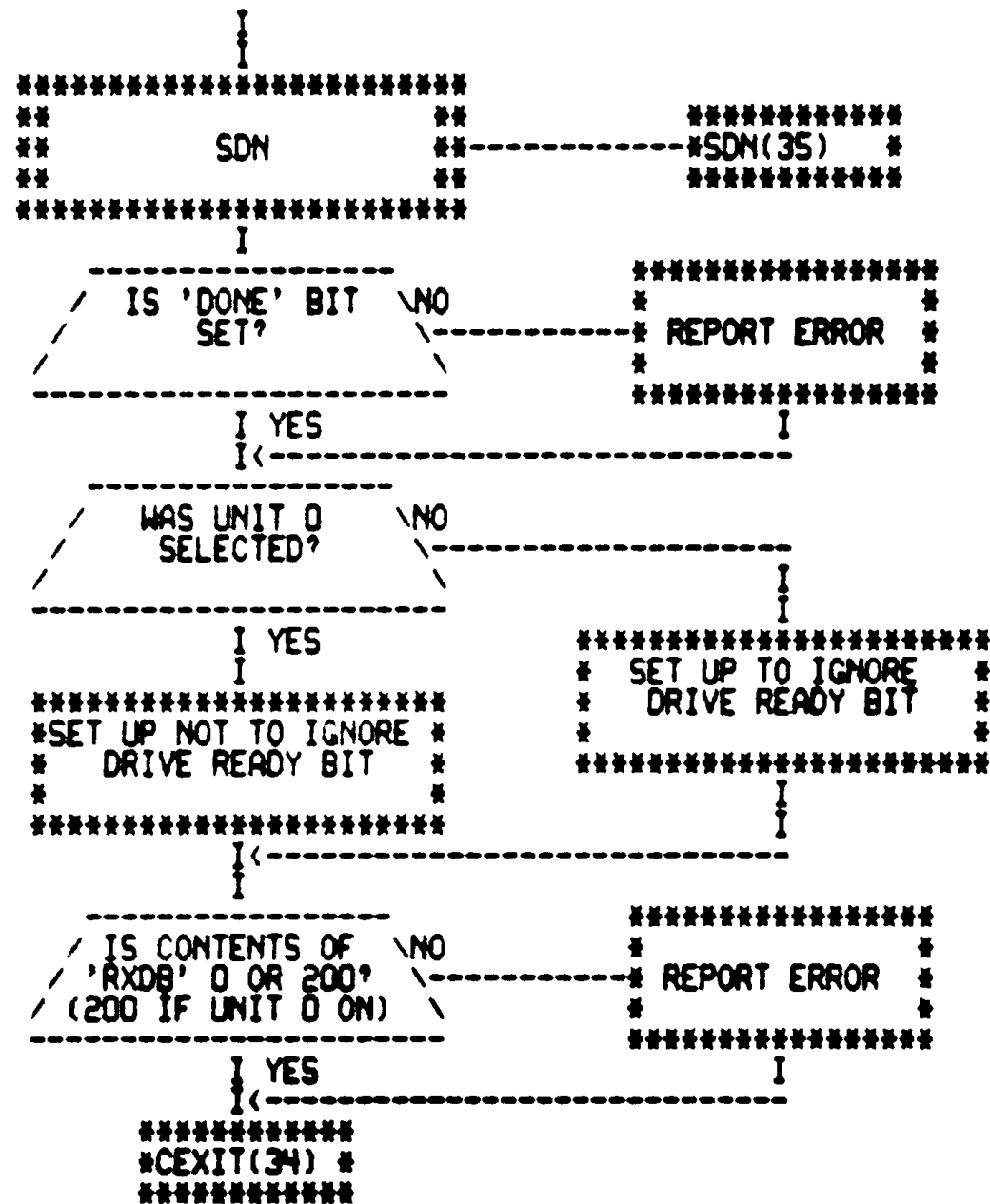
```

*****
*T4
*****
I
*****
*SET UP VECTORS 264 & *
* 266 FOR INTERRUPT *
*
*****
I
*****
* CALCULATE & SET CPU *
*BR LEVEL TO VALUE DE-*
*FINED IN LOC. 'BRLEV'*
*****
I
*****
*SET DEVICE 'IE'*
*   BIT   *
*
*****
I
NOTE: NO INTERRUPT SHOULD OCCUR
      CPU SAME LEVEL AS DEVICE
-----
/ DID AN INTERRUPT \ YES
/   OCCUR?         \ -----
\                   \
\                   \
-----
I NO
I <-----
*****
* SET CPU BR *
* LEVEL TO 7 *
*
*****
I
*****
* CLEAR DEVICE *
* 'IE' BIT *
*
*****
I
*****
*CEXIT(34) *
*****

```







```

*****
*T6FILL(16)*
*****
I
*****
*SET FILL BUFFER*
*FUNCTION & 'GO'*
*   BIT   *
*****
I
*****
**          **-----**GETPAT(36)*
**   GETPAT   **
**          **
*****
I
*****
* SET UP ADDRESS FOR *
* START OF BUFFER FILL *
*   (BUFADR)   *
*****
I
*****
**          **-----**FBEB(16) *
**   FBEB    **
**          **
*****
I
*****
* RETURN TO WHERE THE *
* SUBROUTINE T6FILL WAS *
*   LAST CALLED   *
*****

*****
*BT1280(16)*
*****
I
/-----\
/HAVE 128(10) BEEN NO
/TRANSFERRED? \
I YES
I<-----I
*****
*RETURN TO WHERE SUB- *
* ROUTINE 'FBEB' WAS *
*   LAST CALLED   *
*****

```

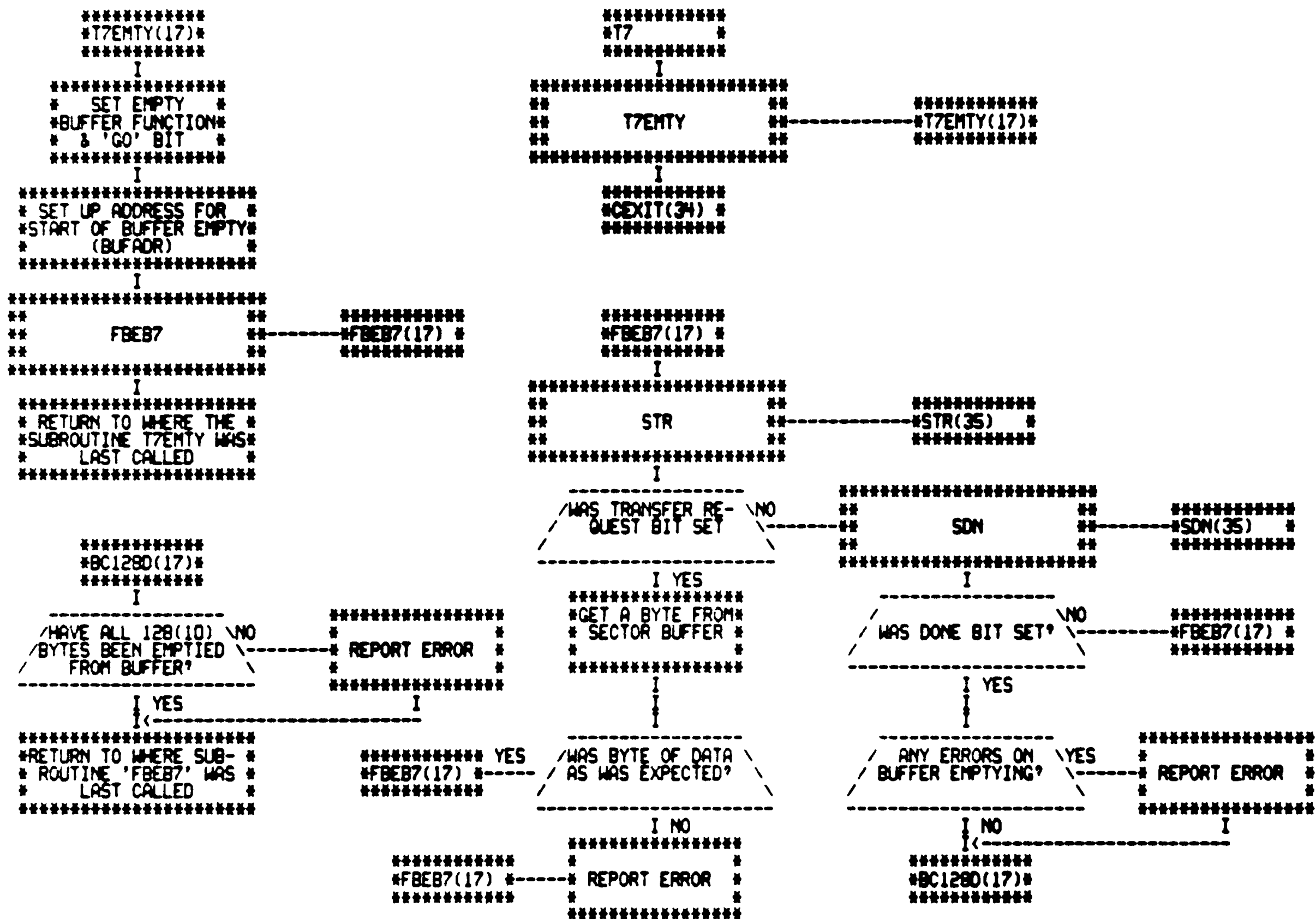
```

*****
*T6      *
*****
I
*****
**          **-----**T6FILL(16)*
**   T6FILL   **
**          **
*****
I
*****
*CEXIT(34) *
*****

*****
*FBEB(16) *
*****
I
*****
**          **-----**STR(35) *
**   STR     **
**          **
*****
I
/-----\
/ WAS TRANSFER \ NO
/ REQUEST BIT SET? \
I YES
*****
* FILL BUFFER *
* WITH 1 BYTE *
*****
I
*****
* KEEP TRACK OF *
* BYTE COUNT   *
*****
I
*****
*FBEB(16) *
*****

*****
**          **-----**SDN(35) *
**   SDN    **
**          **
*****
I
/-----\
/ WAS 'DONE' BIT \ NO
/ SET? \
I YES
/-----\
/ ANY ERRORS ON \ YES
/ BUFFER FILLING? \
I NO
I<-----I
*****
*BT1280(16)*
*****

```

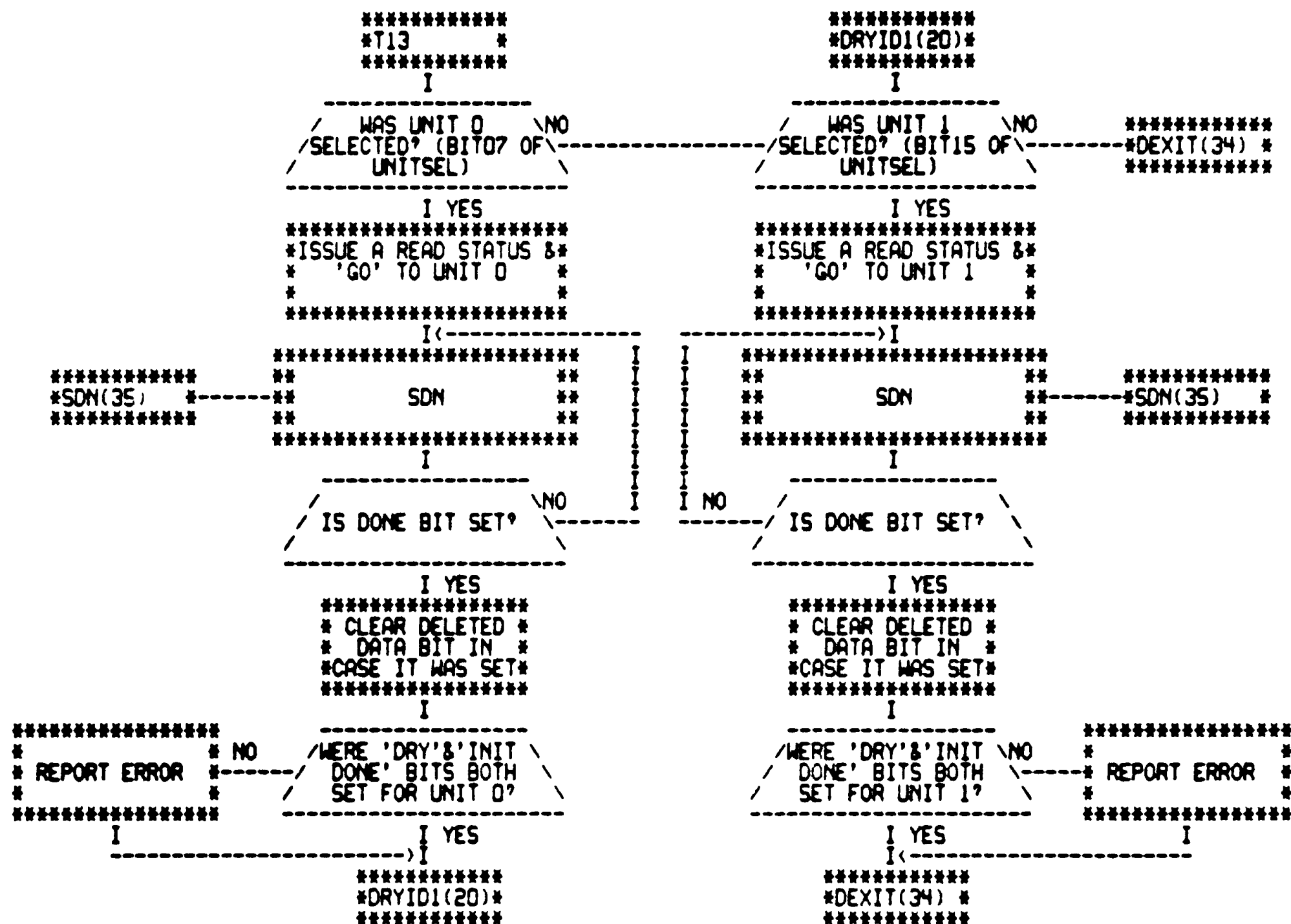


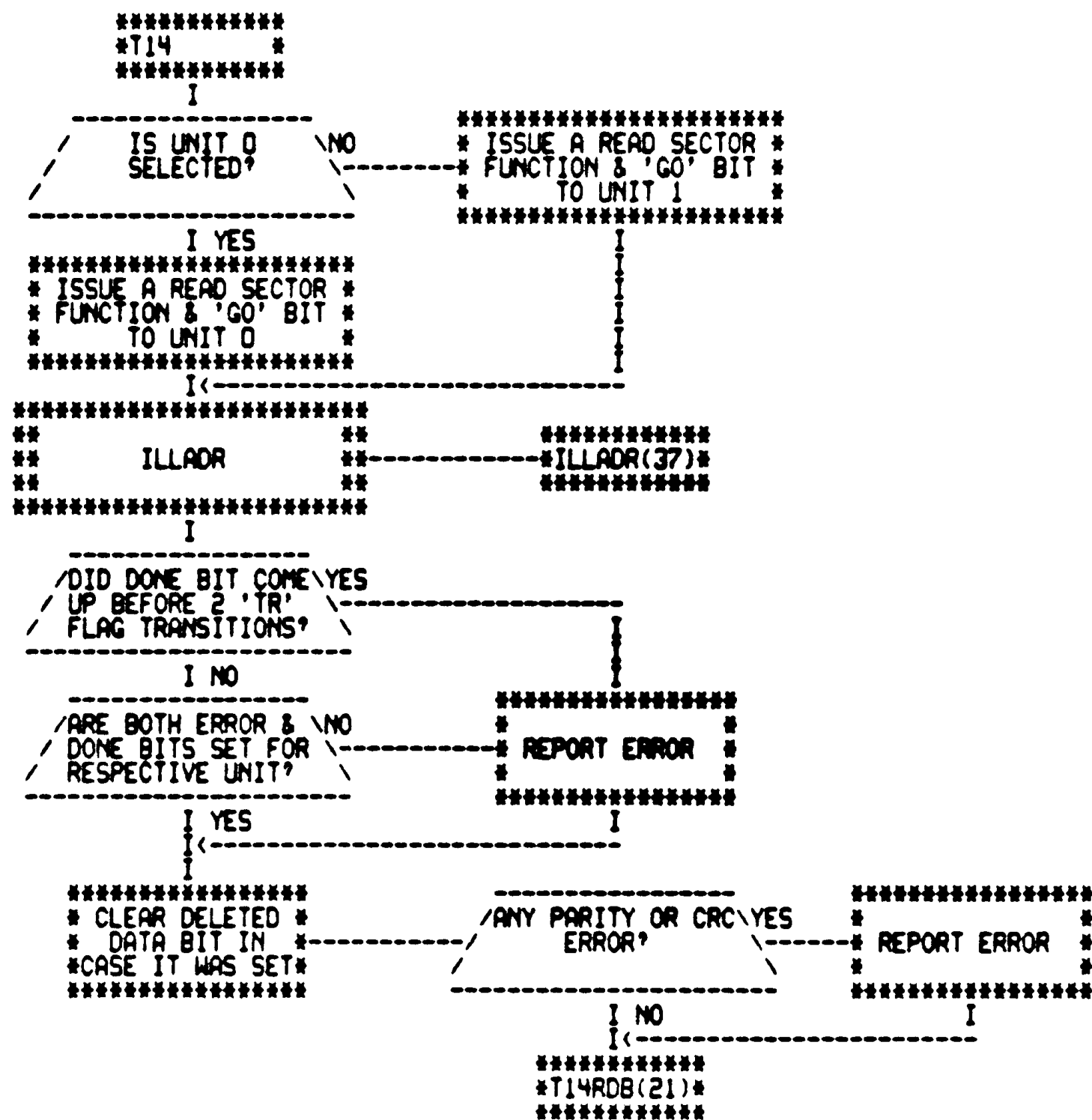
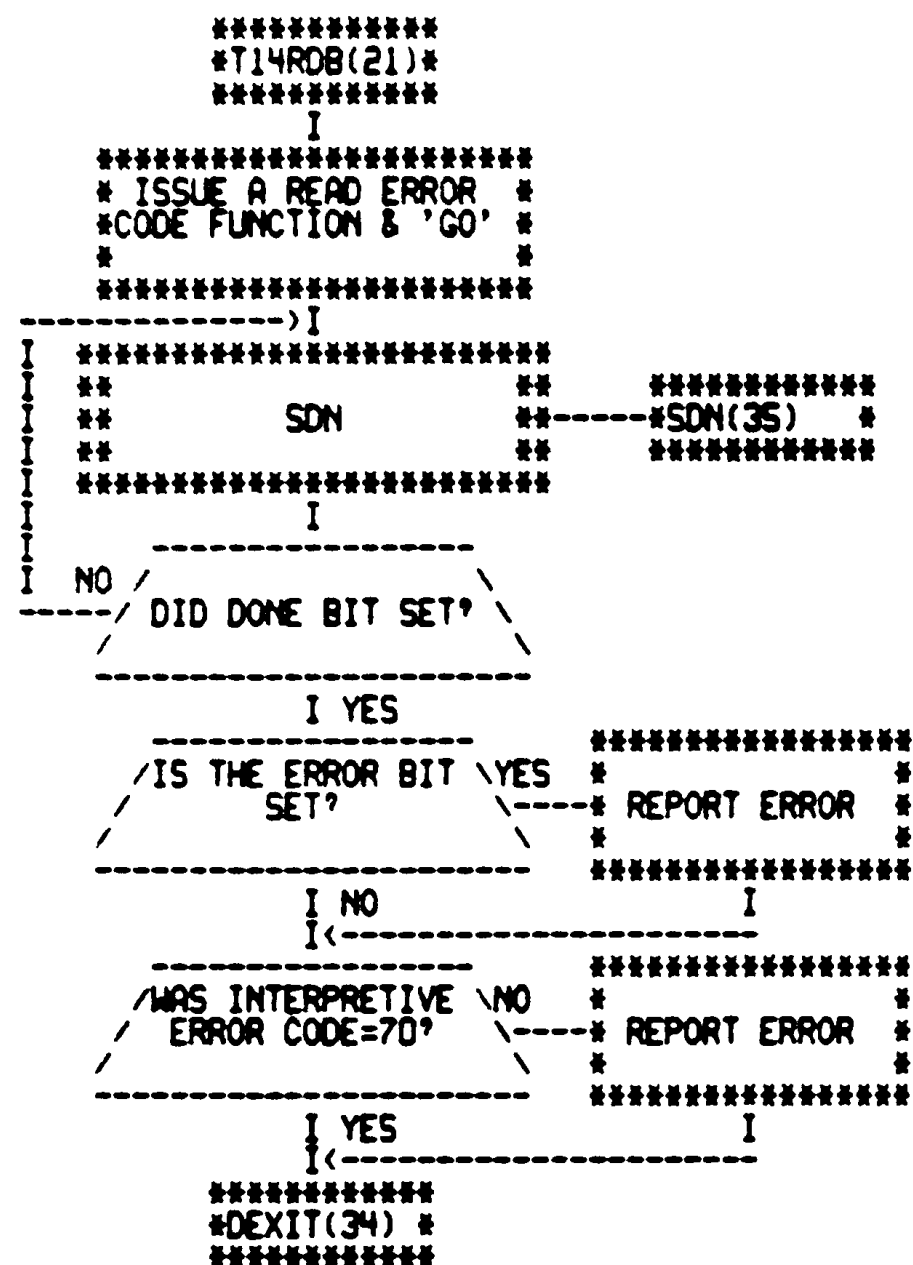
```
*****  
#T11 *  
*****  
I  
*****  
* SET UP PATTERN WORD *  
*FOR EMPTY/FILL BUFFER*  
* WITH O'S (PAT) *  
*****  
I  
*****  
** **  
** T6FILL **-----#T6FILL(16)#  
** **  
*****  
I  
*****  
** **  
** T7EMPTY **-----#T7EMPTY(17)#  
** **  
*****  
I  
*****  
#CEXIT(34) #  
*****
```

```

*****
*T12      *
*****
I
*****
* SET UP PATTERN WORD *
*FOR EMPTY/FILL BUFFER*
*  WITH 1'S (PAT)    *
*****
I
*****
**
**          T6FILL          **-----**T6FILL(16)**
**
**
*****
I
*****
**
**          T7EMPTY         **-----**T7EMPTY(17)**
**
**
*****
I
*****
*CEXIT(34) *
*****

```



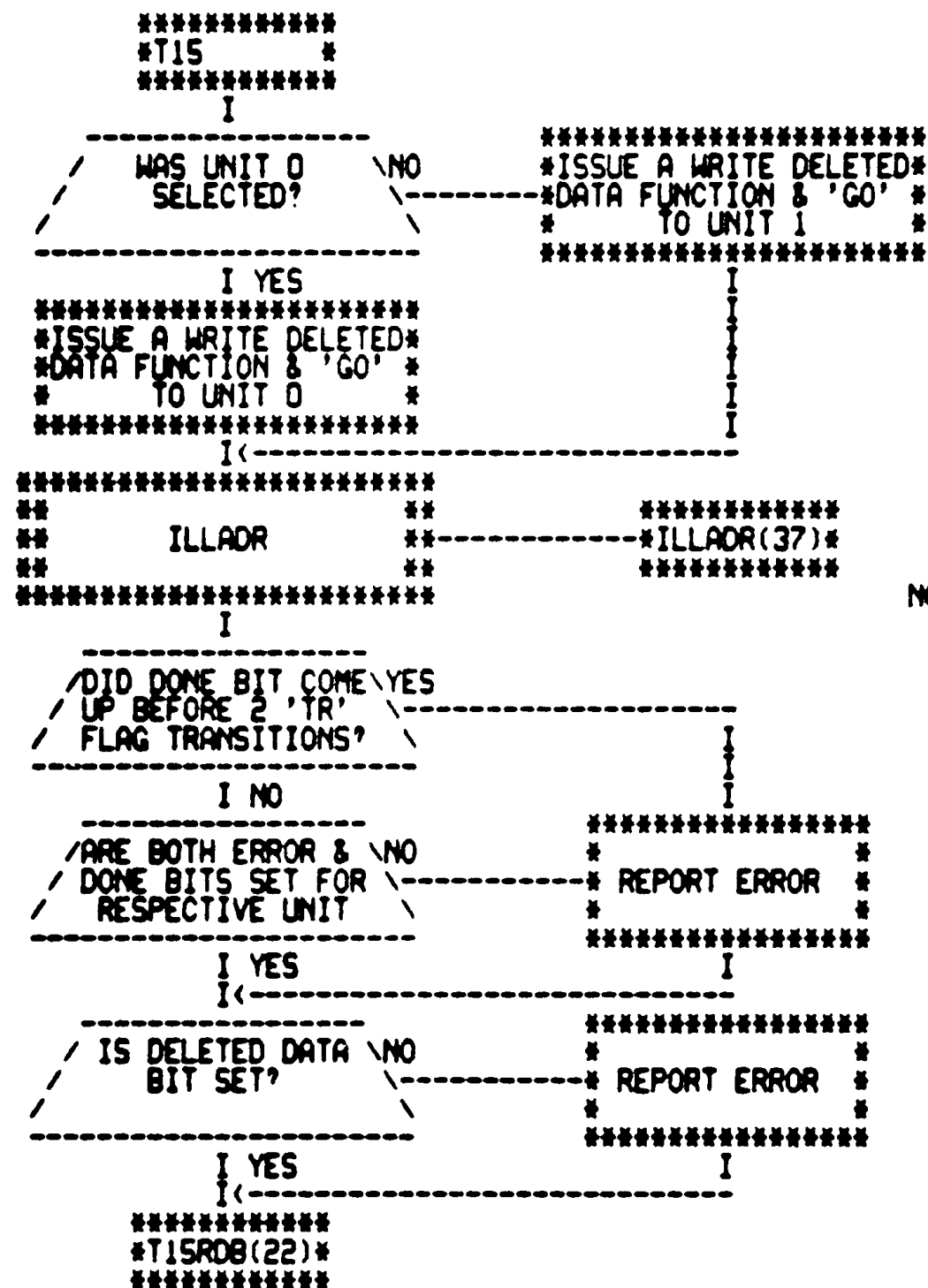
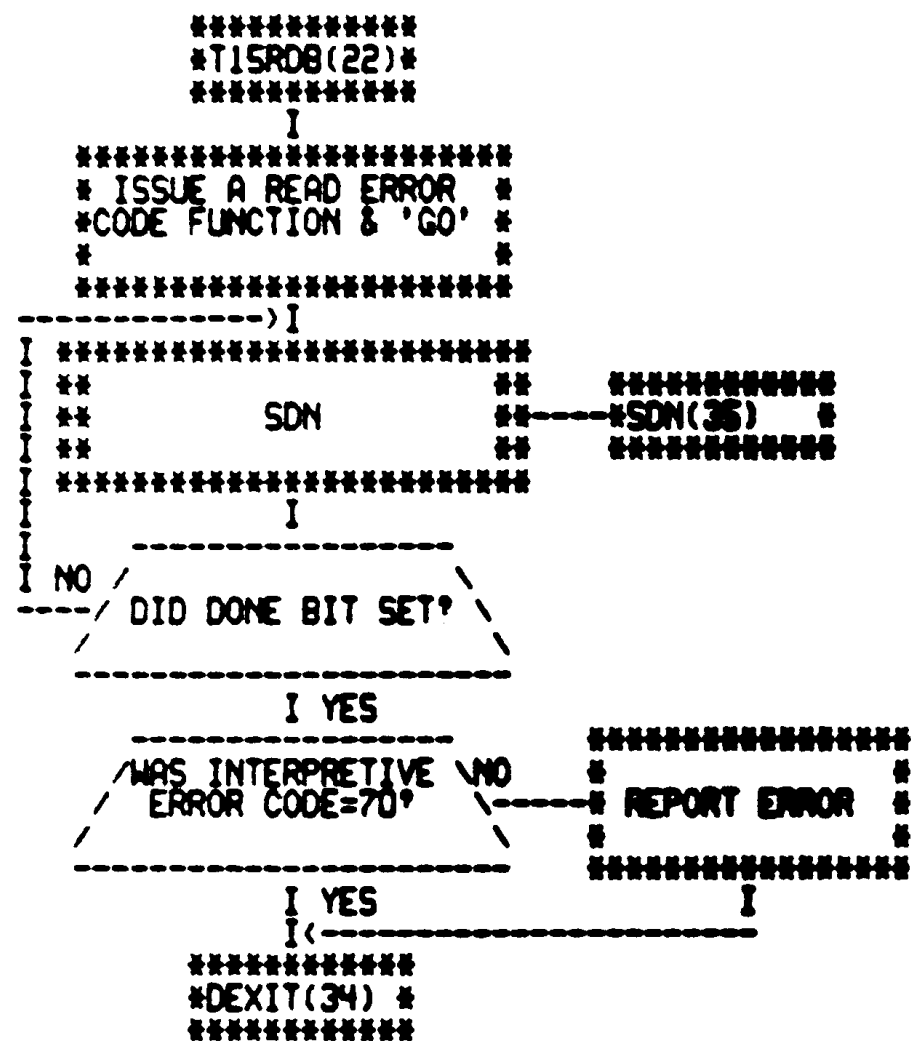


FLOW CHART FOR RX11 DIAGNOSTIC INTERFACE
ERROR FLAG & 'B' CODE VERIFICATION PART 2

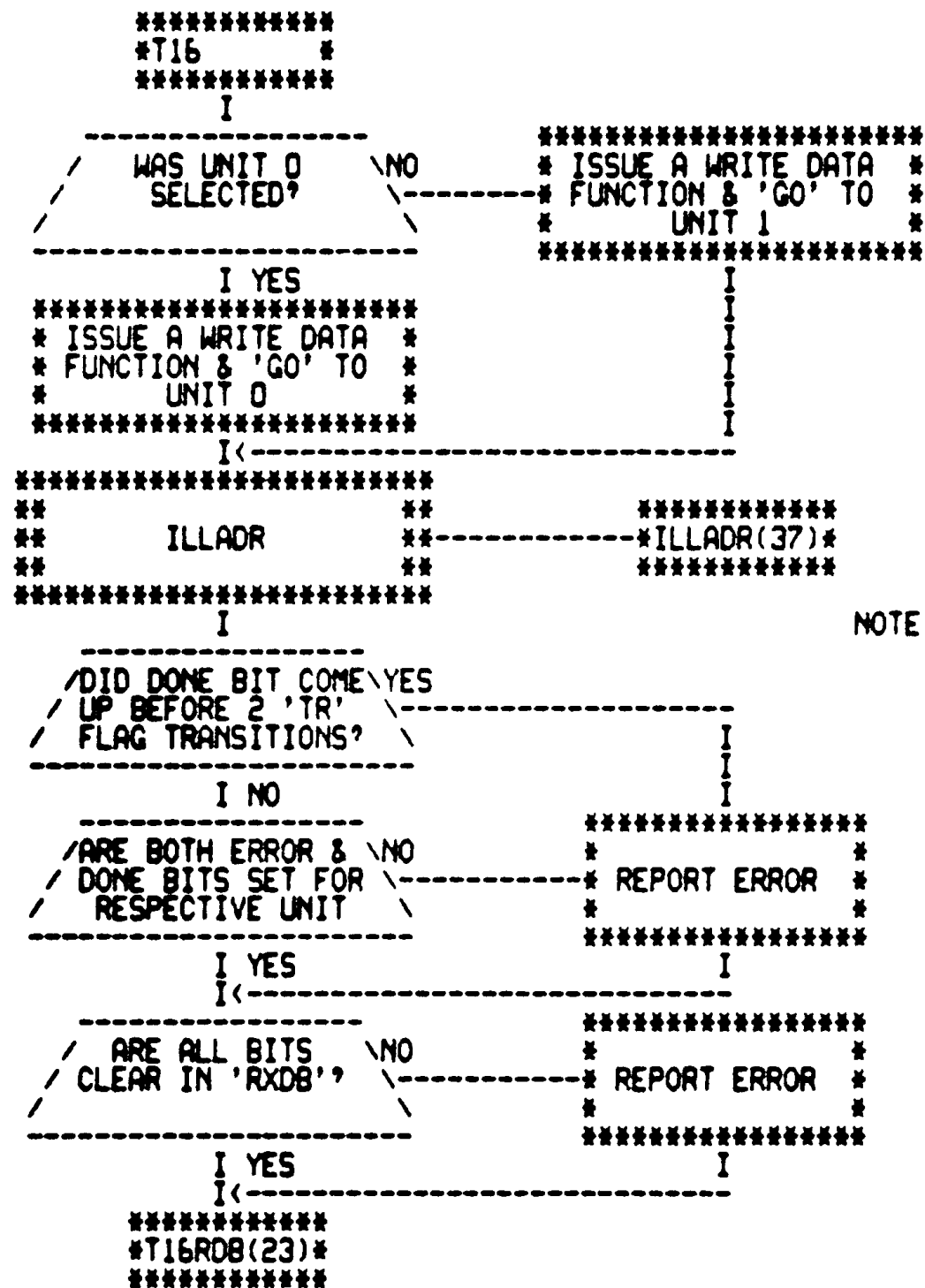
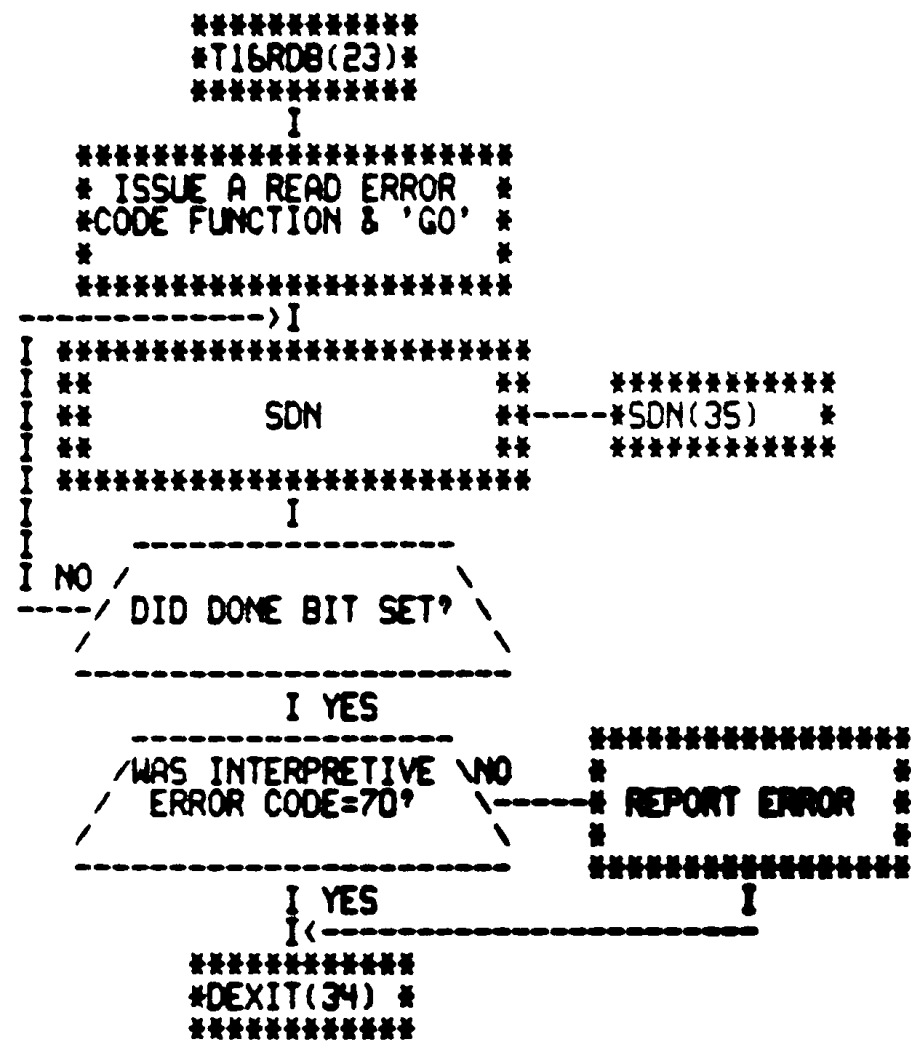
K04

DECFL0 VER 00.10 06-NOV-75 08:47 PAGE 22

SEQ 0048



NOTE: THE ILLEGAL SECTOR
ADDRESS WAS 0 BUT
'DO' BIT SHOULD
STILL BE SET



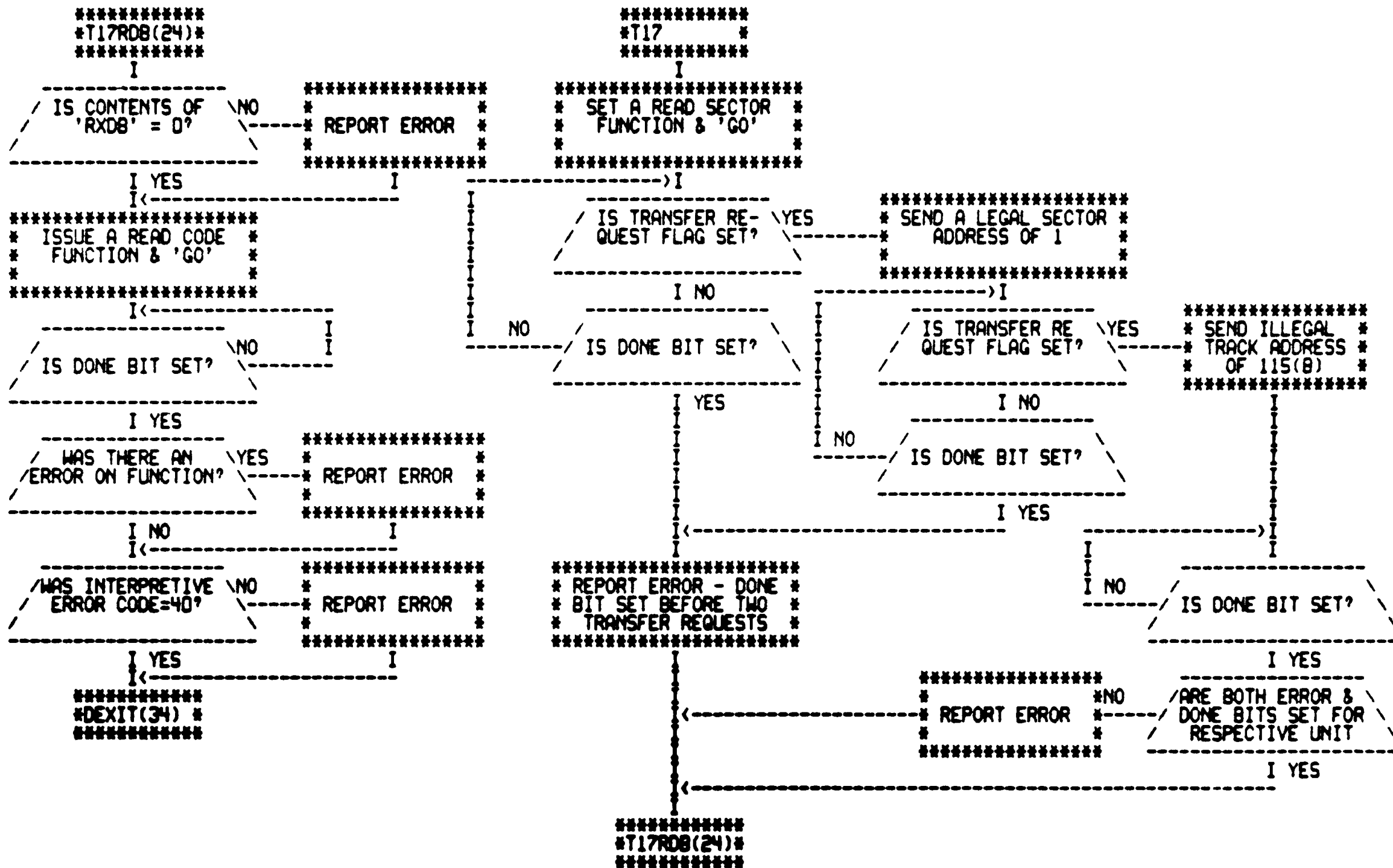
NOTE: THE ILLEGAL SECTOR
ADDRESS WAS 0

FLOW CHART FOR RX11 DIAGNOSTIC INTERFACE
ILLEGAL TRACK ERROR

MO4

DECFL0 VER 00.10 06-NOV-75 08:47 PAGE 24

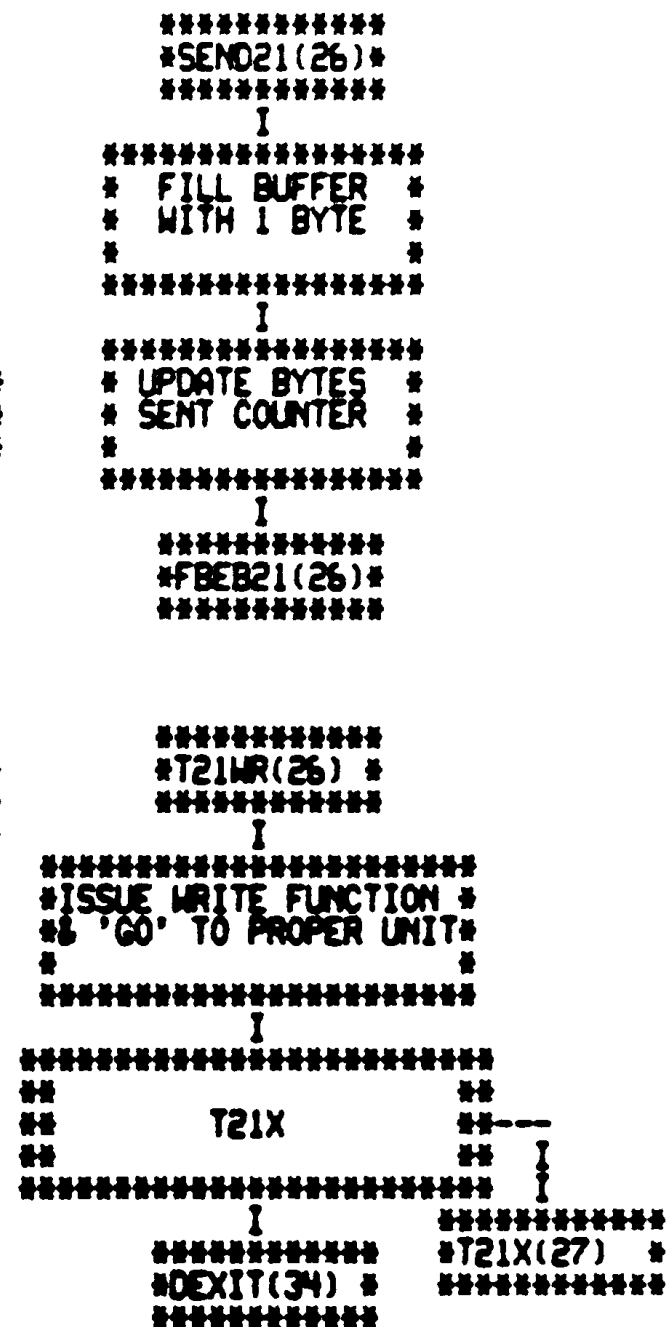
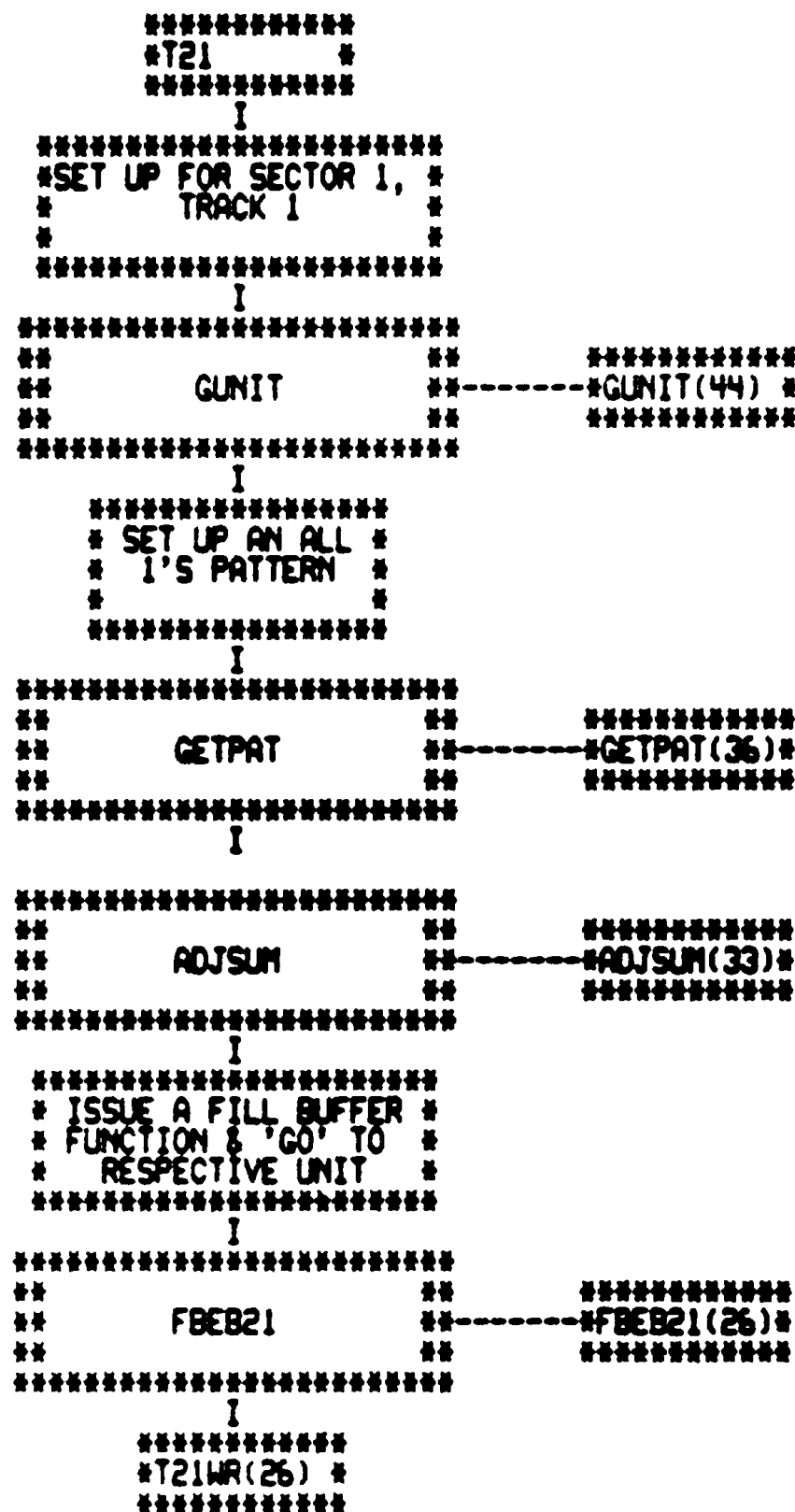
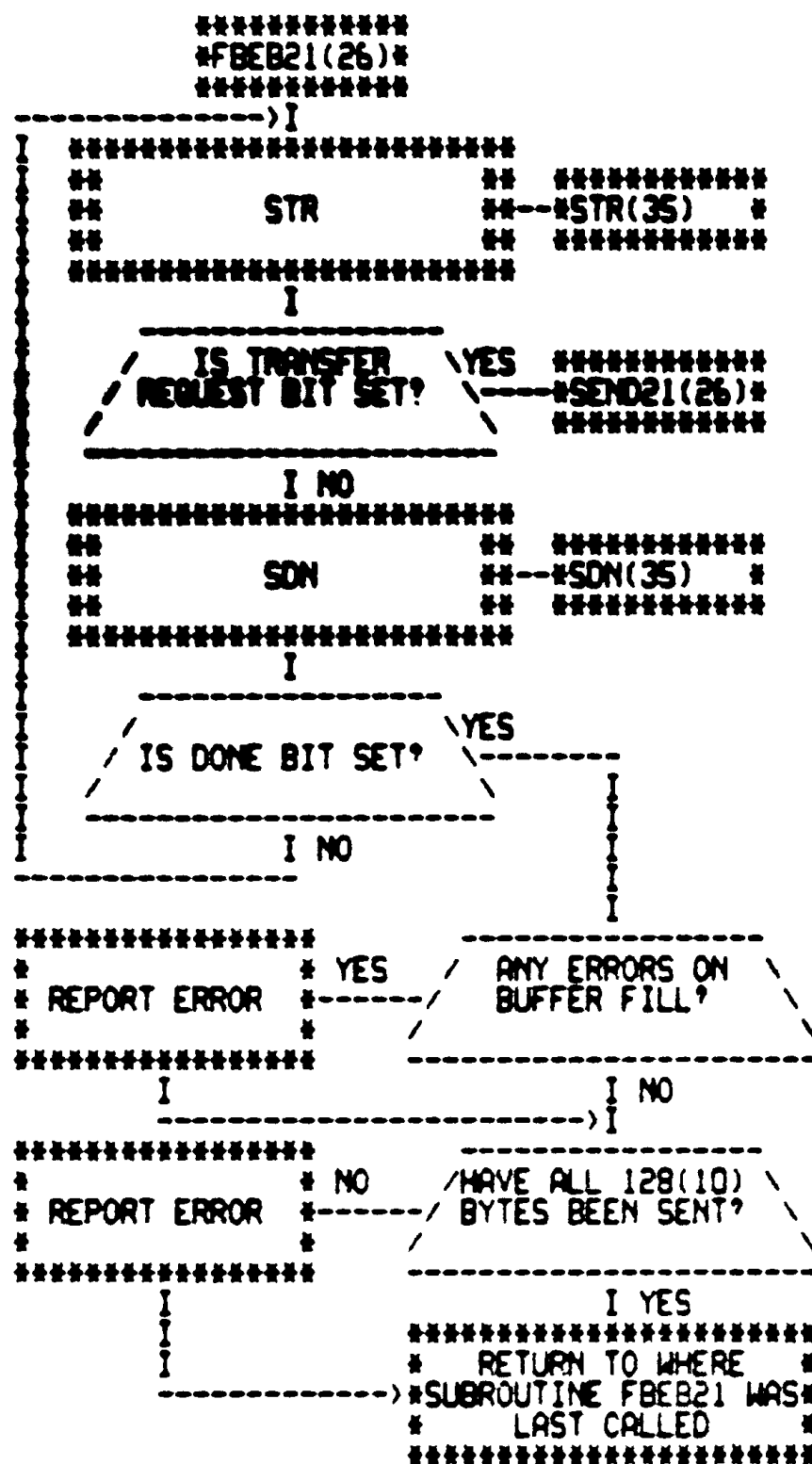
SEQ 0050

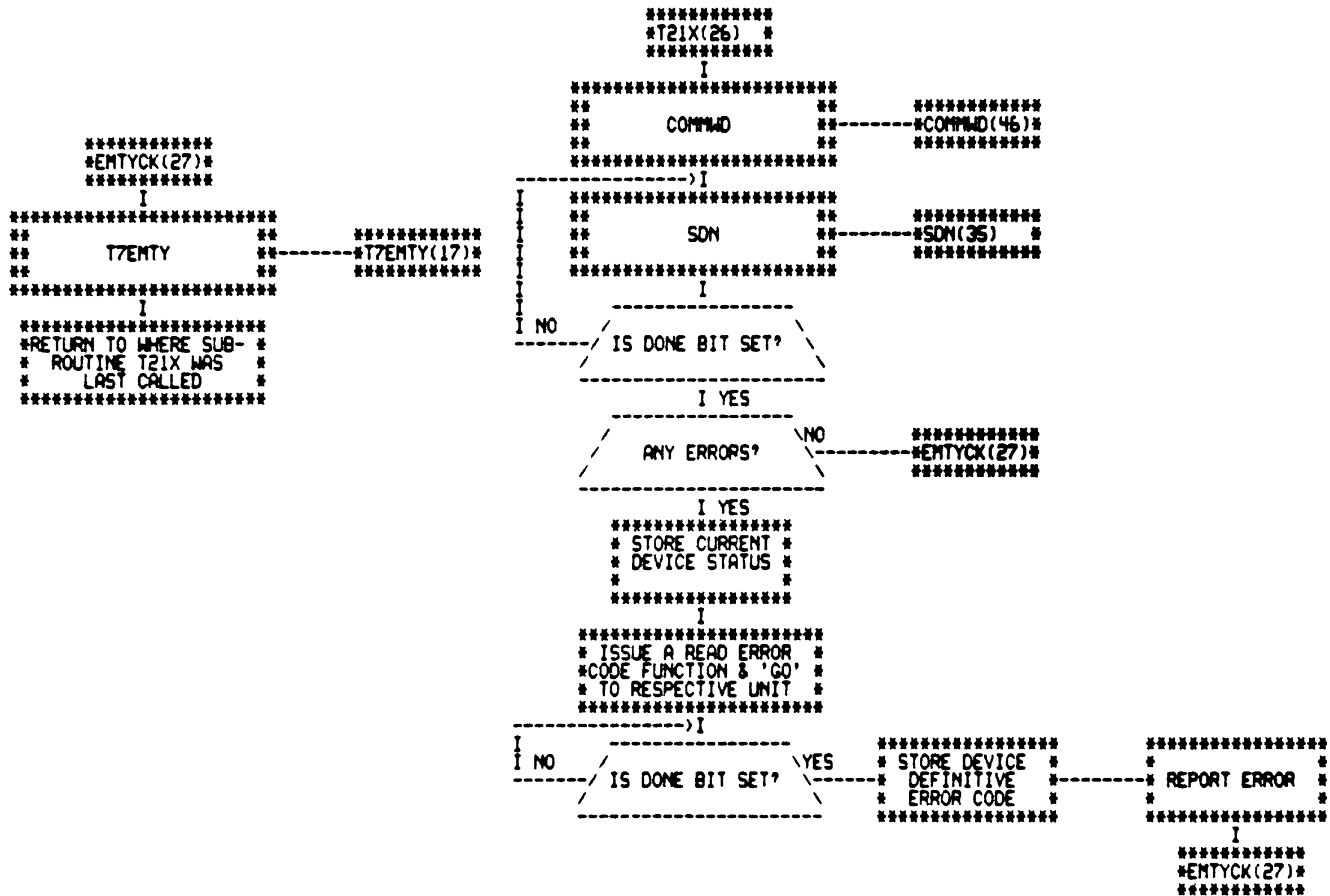


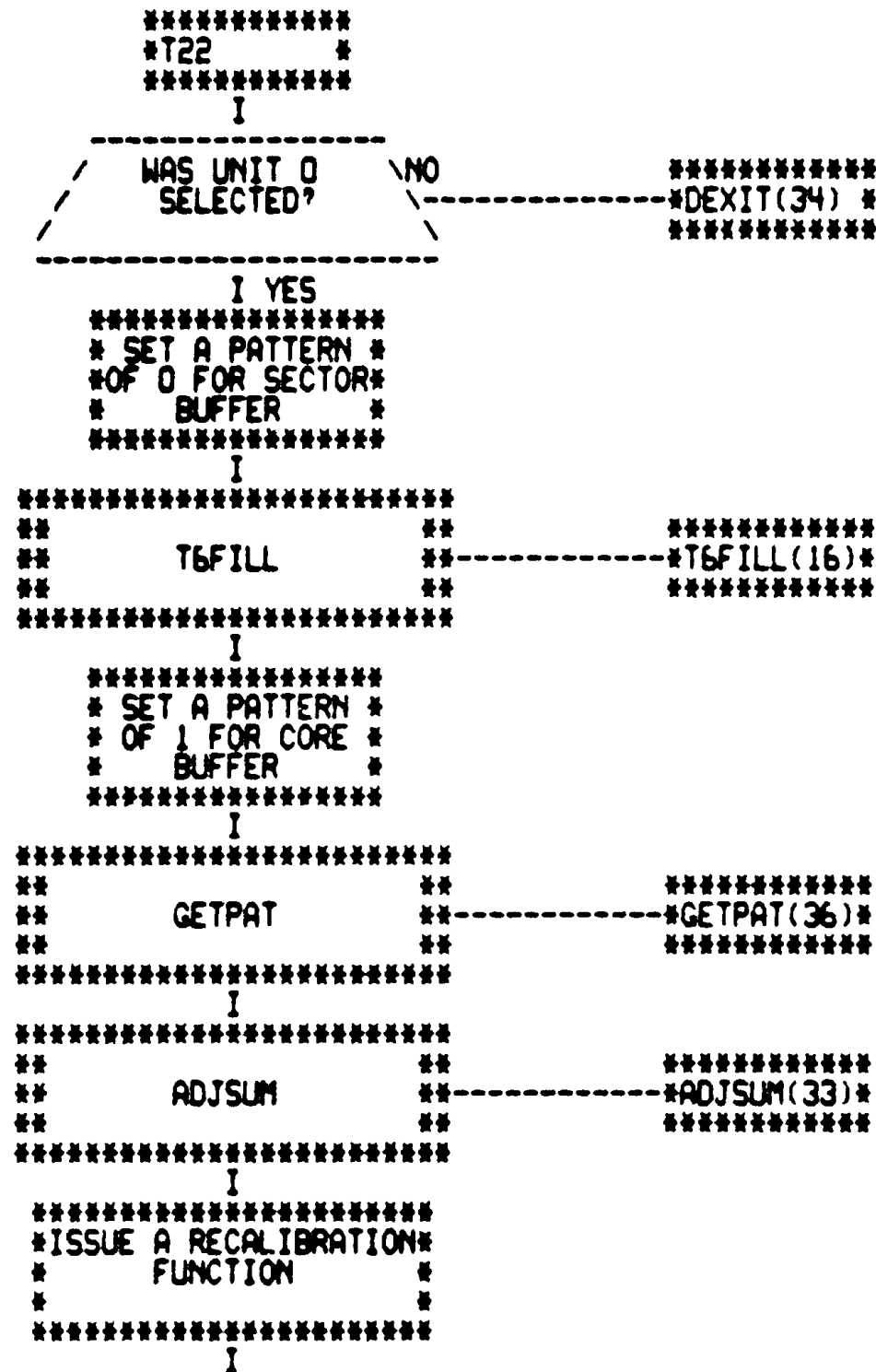
```

*****
*T20      *
*****
I
*****
*SET UP VECTORS 264 & *
* 266 FOR INTERRUPT *
*                   *
*****
I
*****
**          **
**      RONLY      **-----**RDONLY(39)**
**          **
*****
I
*****
*DEXIT(34) *
*****

```

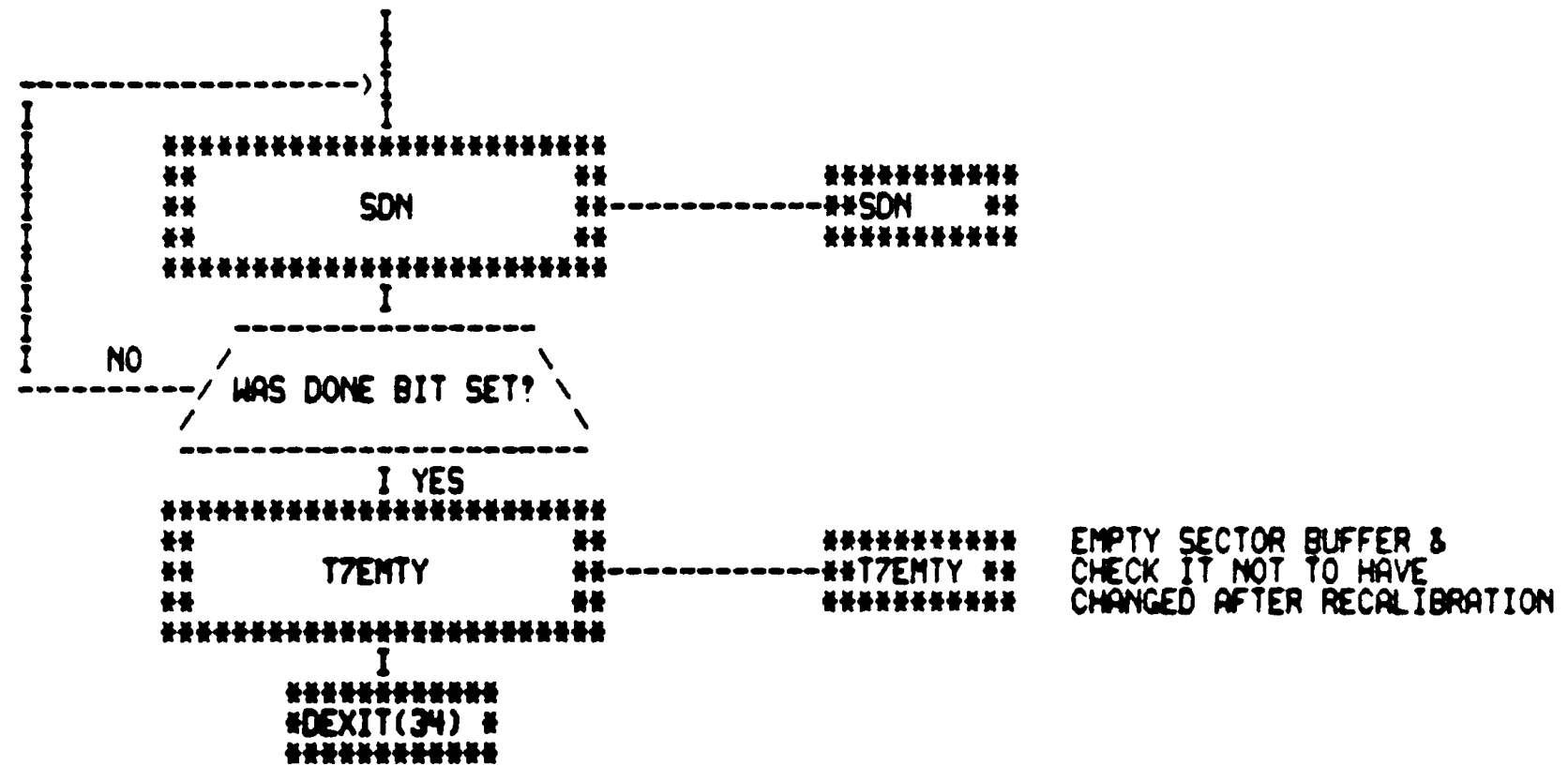






NOTE: THIS PROCESS WILL LOAD
THE SECTOR BUFFER WITH
ALL 0'S

NOTE: THIS PROCESS WILL LOAD
THE CORE BUFFER WITH
ALL 1'S



```

*****
*T23
*****
I
*****
**          **          *****
**      T6FILL      **-----**T6FILL **  LOADS SECTOR WITH
**          **          *****          ALL ZEROS
**          **
*****
I
*****
**          **          *****
**      GUNIT      **-----**GUNIT **
**          **          *****
**          **
*****
I
*****
**          **          *****
**      GETPAT      **-----**GETPAT **  LOADS CORE BUFFER WITH
**          **          *****          ALL 1'S
**          **
*****
I
*****
**          **          *****
**      ADJSUM      **-----**ADJSUM **
**          **          *****
**          **
*****
I
*****
**ISSUE A READ FUNCTION*
**      & 'GO'      *
**          **
*****
I
*****
**          **          *****
**      T21X      **-----**T21X(27) *  WAIT FOR DONE & CHECK
**          **          *****          SECTOR BUFFER TO HAVE
**          **          *****          BEEN LOADED BY READ
**          **          *****          FUNCTION
*****
I
*****
**DEXIT(34) **
*****

```

```

*****
#T24      *
*****
      I
*****
#SET UP A DATA PATTERN#
#  FOR FLOATING 0'S   #
#                     #
*****
#T24X      *----->I
*****
*****
#SET UP VECTORS 264 & #
# 266 FOR INTERRUPT  #
#                     #
*****
      I
*****
**                     **
**      DRVSWP          **-----DRVSWP(40)*
**                     **
*****
      I
*****
#DEXIT(34) #
*****

*****
#T25      *
*****
      I
*****
#SET UP A DATA PATTERN#
#  FOR FLOATING 1'S   #
#                     #
*****
      I
*****
#T24X      *
*****

```

THIS ROUTINE WILL ALTERNATE
BETWEEN DRIVES ON THE
SELECTED TRACKS

```

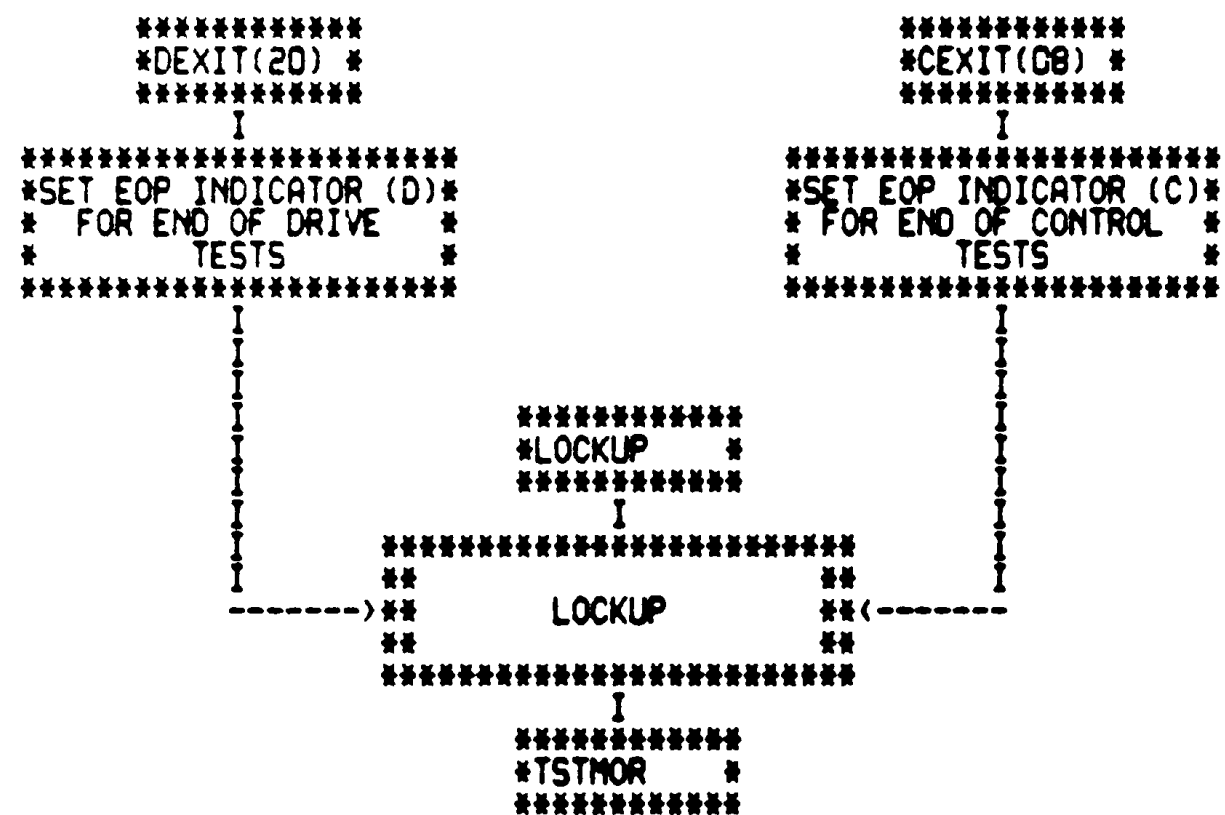
*****
*T26 *
*****
I
*****
* DECREMENT SEQUENCE *
*SETUP & SELECT RANDOM*
* DATA PATTERN *
*****
I
*****
*SET UP VECTORS 264 & *
* 266 FOR INTERRUPT *
* *
*****
I
*****
** WTRDCK **-----**WTRDCK(42)**
** *
** *
*****
I
*****
* INHIBIT *
* DECREMENT *
* SEQUENCE *
*****
I
*****
*DEXIT(34) *
*****

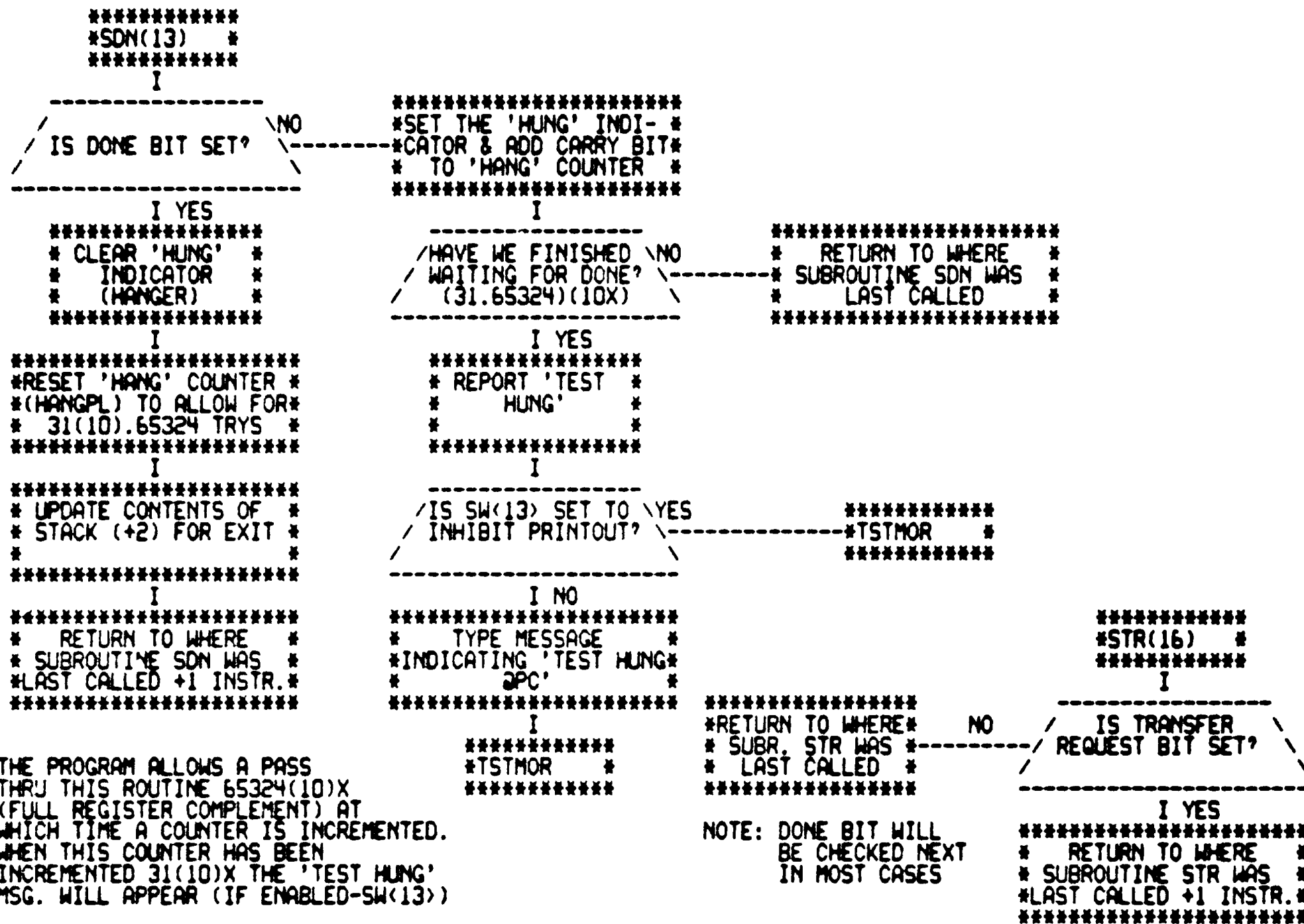
```

```

*****
*ADJSUM(26)*
*****
I
*****
*FILL 1ST WORD OF CORE*
* BUFFER WITH TRACK & *
*SECTOR (LOW & HGH BY)*
*****
I
*****
*STORE THE PATTERN SUM*
*   IN 'CKSUM'   *
*               *
*****
I
*****
* ADD TRACK & SECTOR *
* ADDRESS VALUES TO *
*   CHECKSUM   *
*****
I
*****
*INSERT CHECKSUM INTO *
*LOW BYTE OF LAST WORD*
*   OF CORE BUFFER *
*****
I
*****
* GENERATE *
* NEGATIVE *
* CHECKSUM *
*****
I
*****
* INSERT NEG. CKSUM *
* INTO HIGH BYTE OF *
* LAST WORD OF BUFFER *
*****
I
*****
* INIT. START *
*ADDRESS OF CORE*-----*SUBROUTINE ADJSUM WAS*
*BUFFER INTO RD *      *   LAST CALLED   *
*****
*****

```





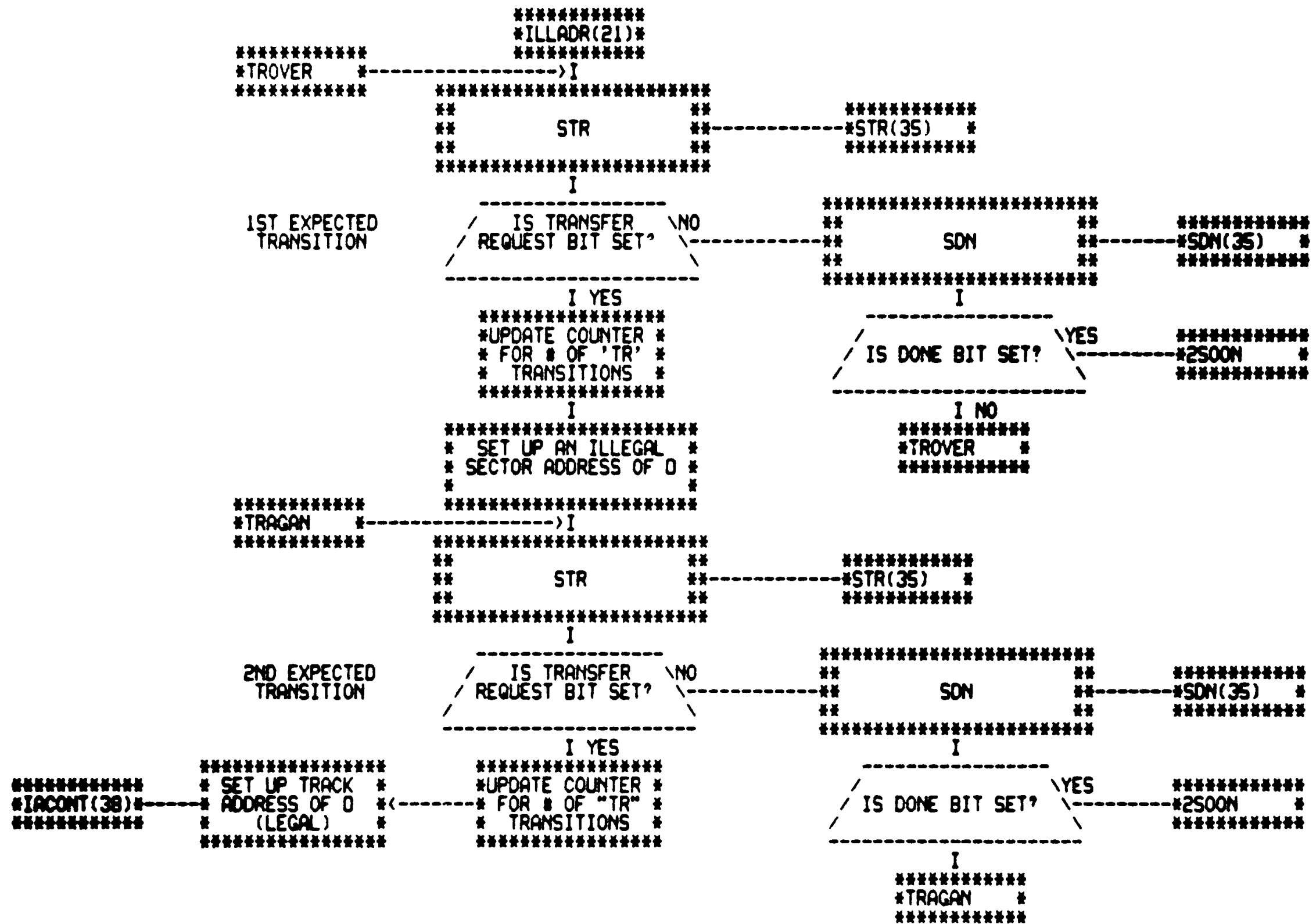
NOTE: PATTERN SELECTIONS ARE
AS FOLLOWS:

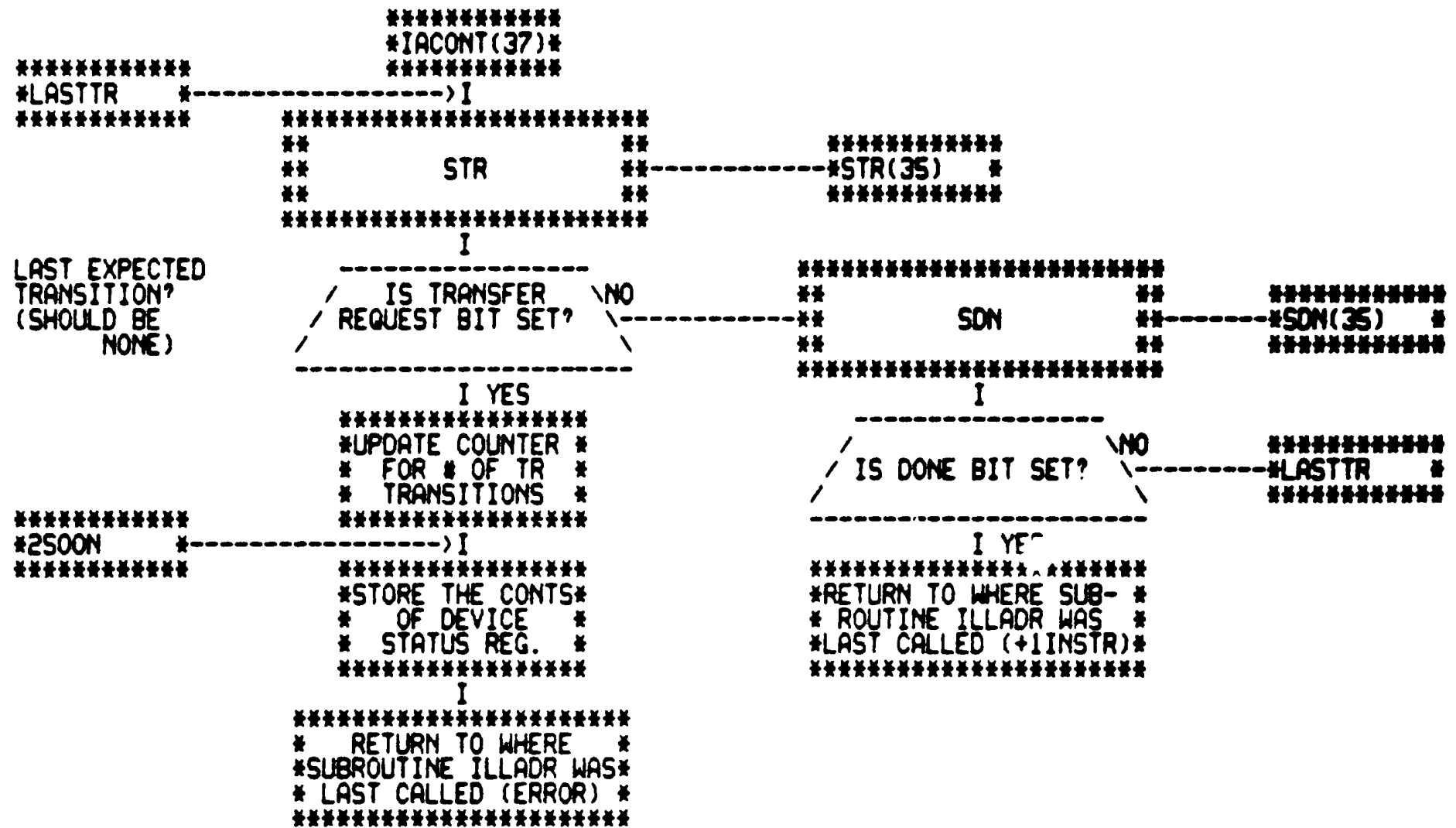
- A) ZEROS
- B) ONES
- C) FLOATING 0
- D) FLOATING 1
- E) 125052
- F) 314063
- G) COUNT UP
- H) RANDOM

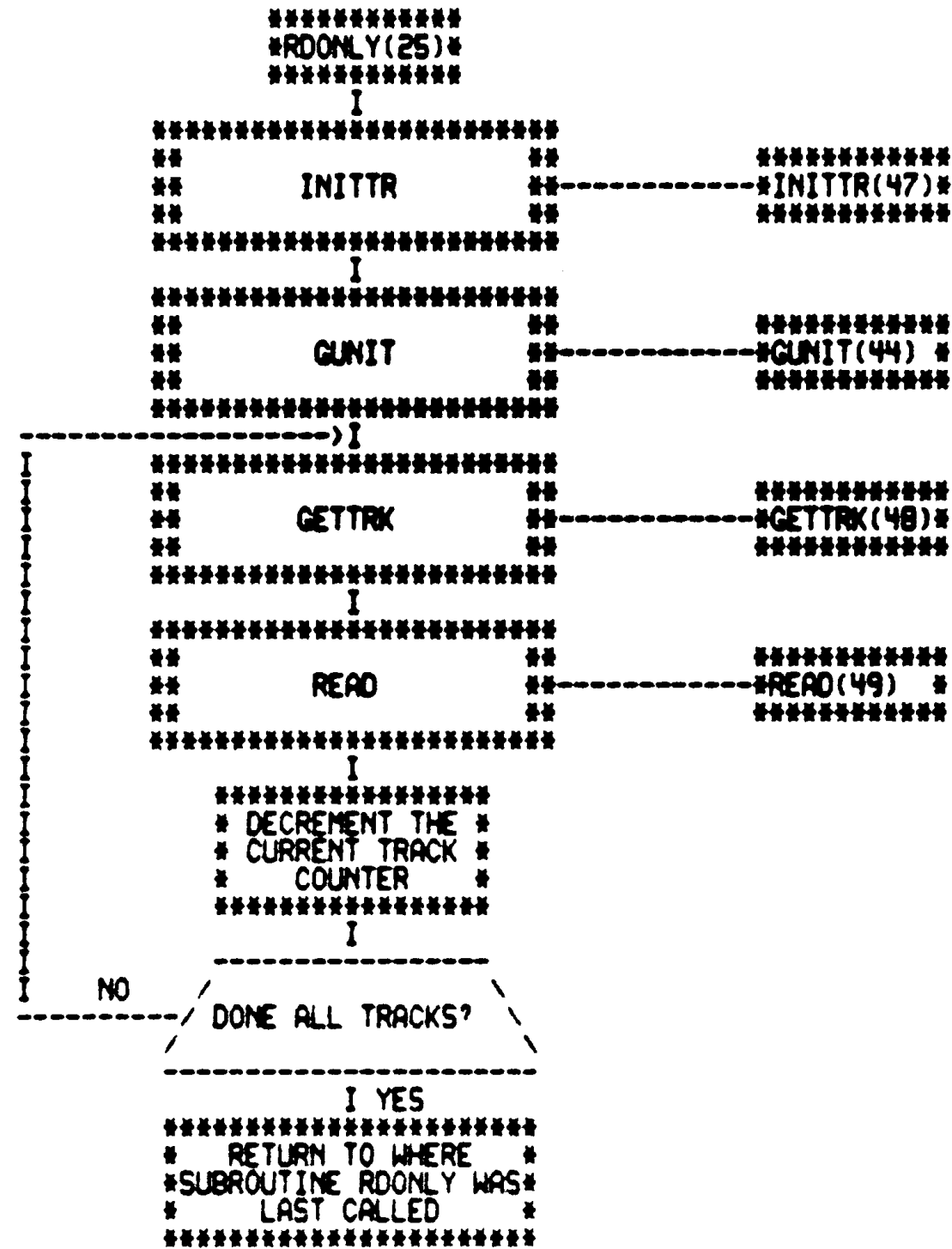
```

*****
*GETPAT(16)*
*****
I
*****
* SET ADDRESS OF 1ST *
*DATA BYTE TO 1ST ADD.*
* OF CORE BUFFER *
*****
I
*****
* SET UP FOR *
* CALCULATION OF *
* PATTERN CHECKSUM *
*****
I
*****
* GET PATTERN BITS *
*(ENTERED BY USER THRU*
* CONSOLE SWR) *
*****
I
*****
*SHIFT PATTERN BITS TO*
* OBTAIN OFFSET FOR *
* PATTERN TABLE *
*****
I
*****
* USE OFFSET TO *
*GET ADDRESS OF *
*DESIRED PATTERN*
*****
I
*****
* JUMP TO ROUTINE TO *
* SET UP DESIRED *
* PATTERN *
*****

```



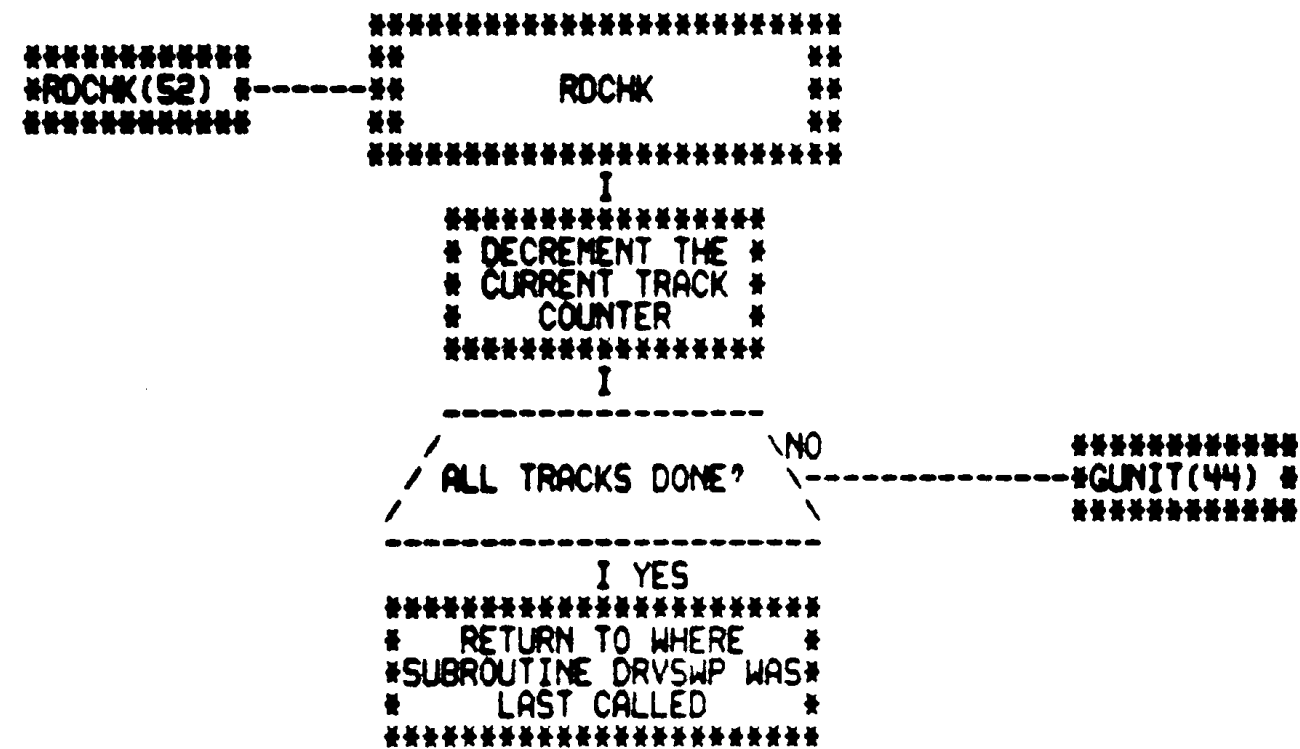




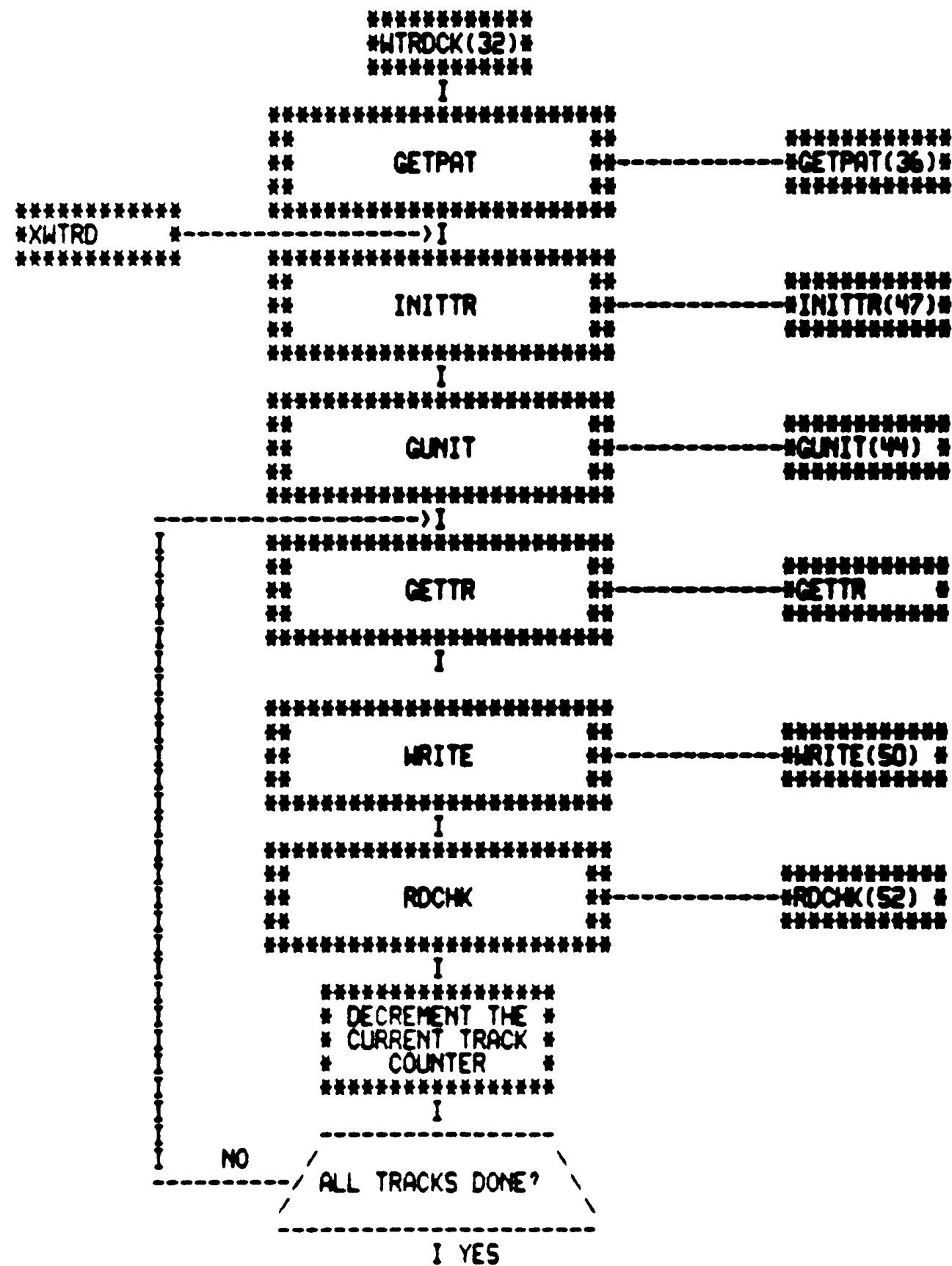
```

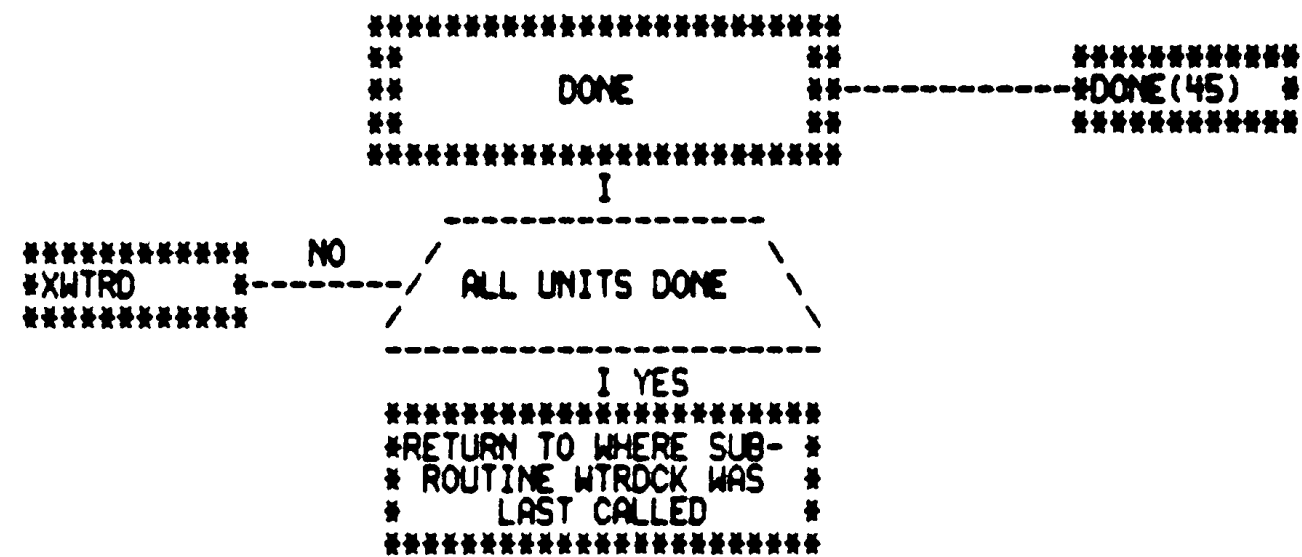
*****
*DRVSWP(31)*
*****
      I
*****
**          **          *****
**      GETPAT      **-----*GETPAT(36)*
**          **          *****
*****
      I
*****
**          **          *****
**      INITTR      **-----*INITTR(47)*
**          **          *****
*****
      I
*****
**          **          *****
**      GUNIT      **-----*GUNIT(44) *
**          **          *****
*****
      I
*****
**          **          *****
**      GETTR      **-----*GETTR      *
**          **          *****
*****
      I
*****
**          **          *****
**      WRITE      **-----*WRITE(50) *
**          **          *****
*****
      I
*****
**          **          *****
**      RDCHK      **-----*RDCHK(52) *
**          **          *****
*****
      I
*****
**          **          *****
**      GUNIT      **-----*GUNIT(44) *
**          **          *****
*****
      I
*****
**          **          *****
**      WRITE      **-----*WRITE(50) *
**          **          *****
*****
      I

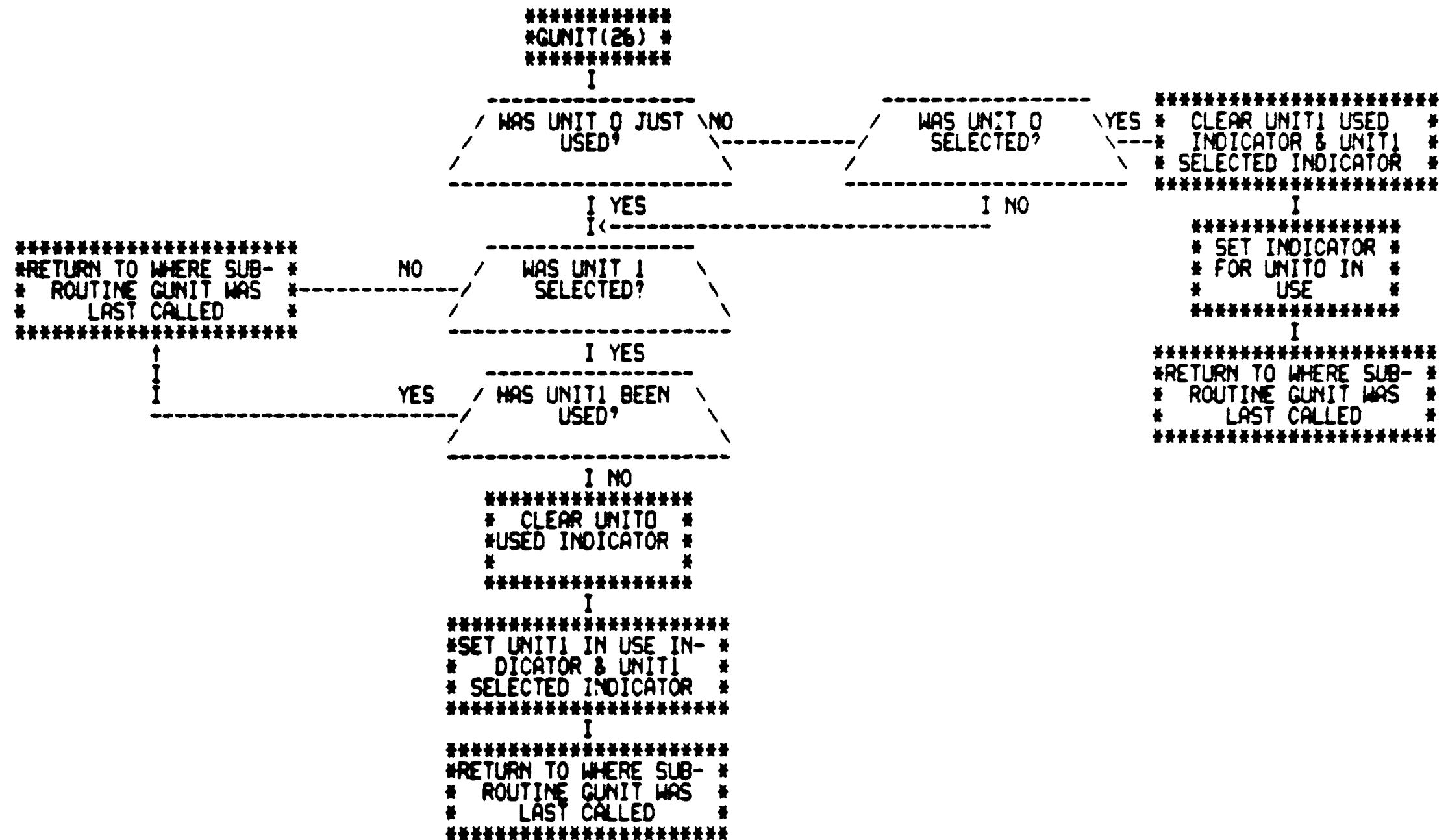
```

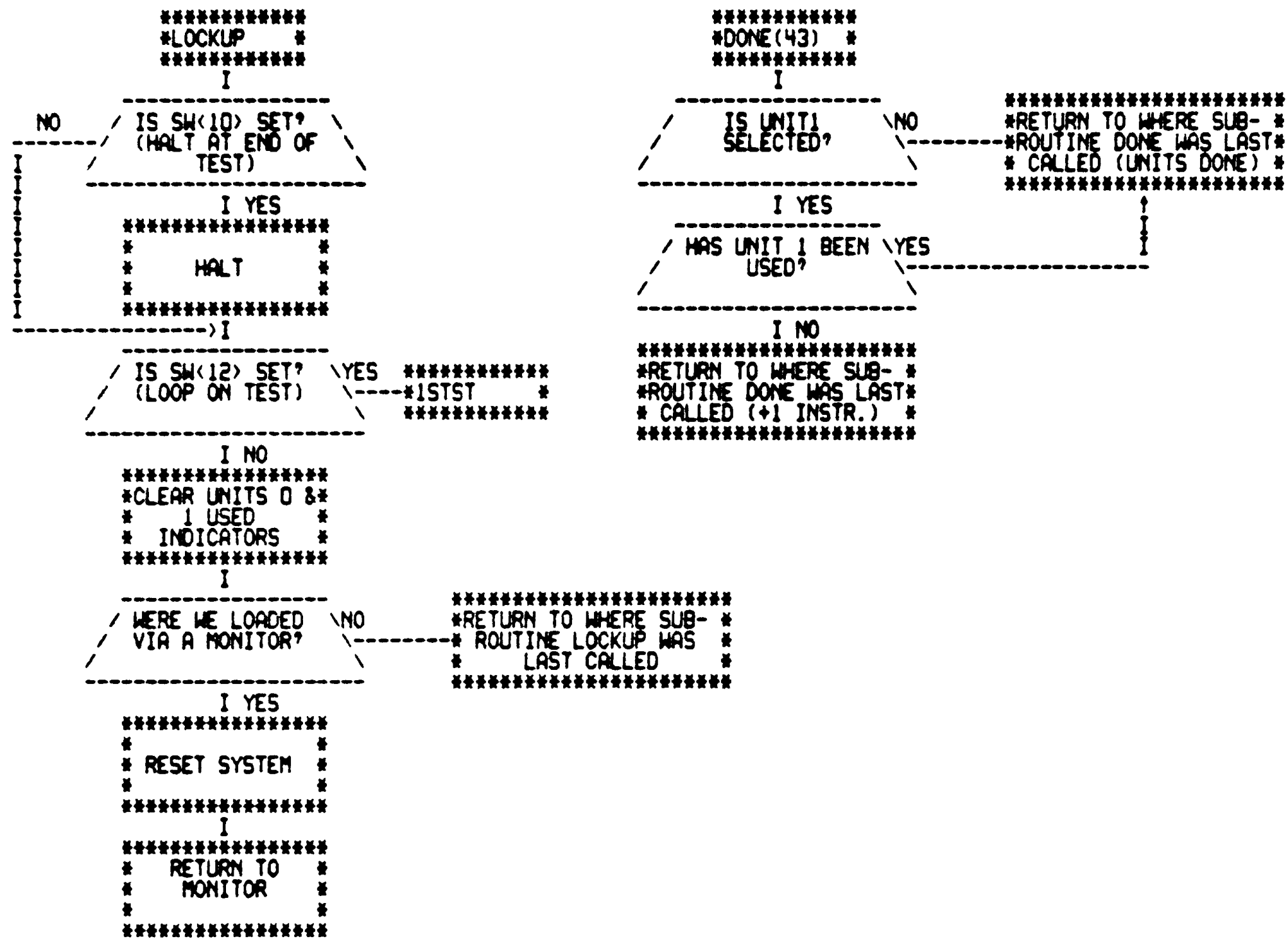


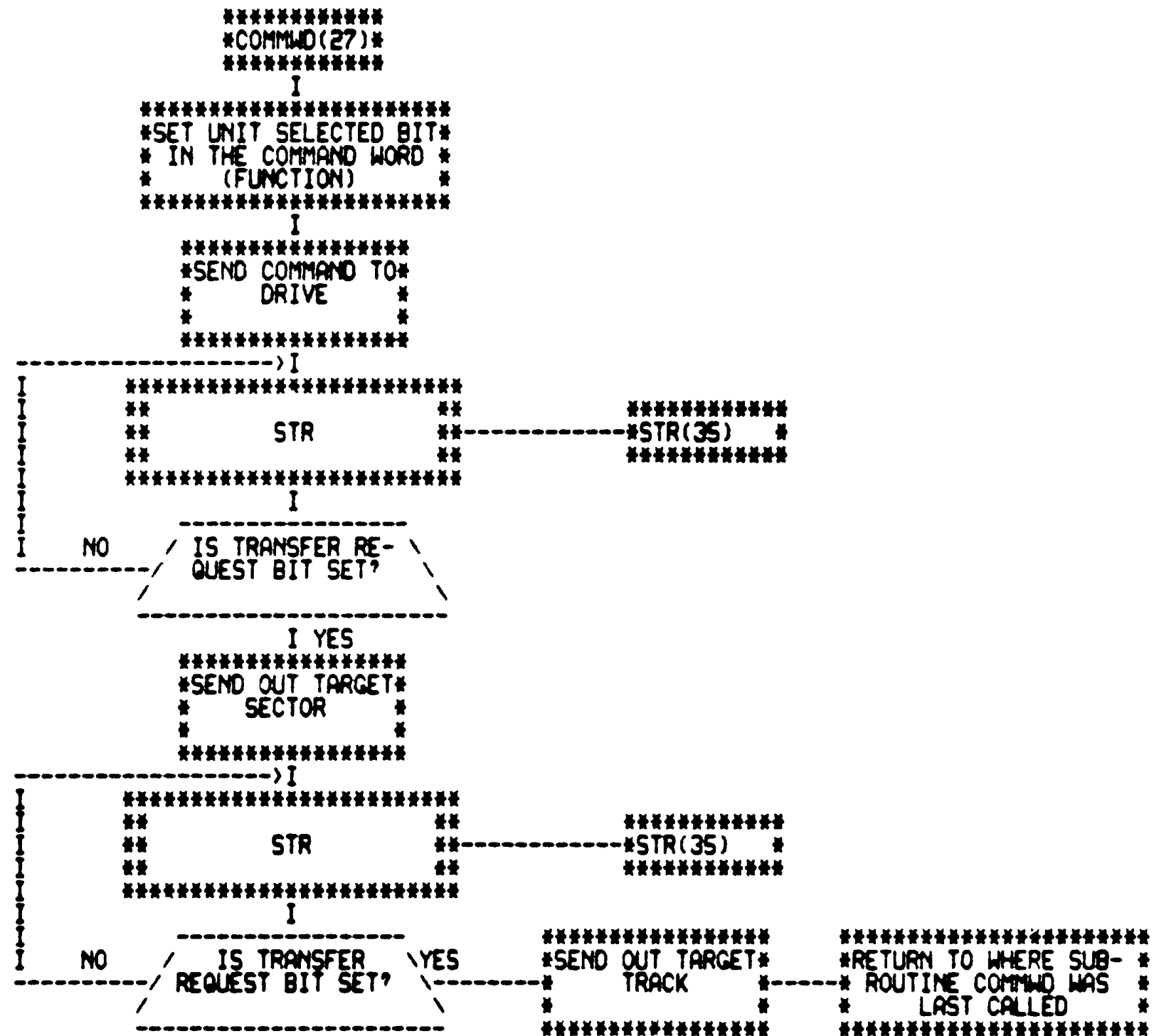
FLOW CHART FOR RX11 DIAGNOSTIC INTERFACE
SUBROUTINE TO VERIFY DATA WRITTEN

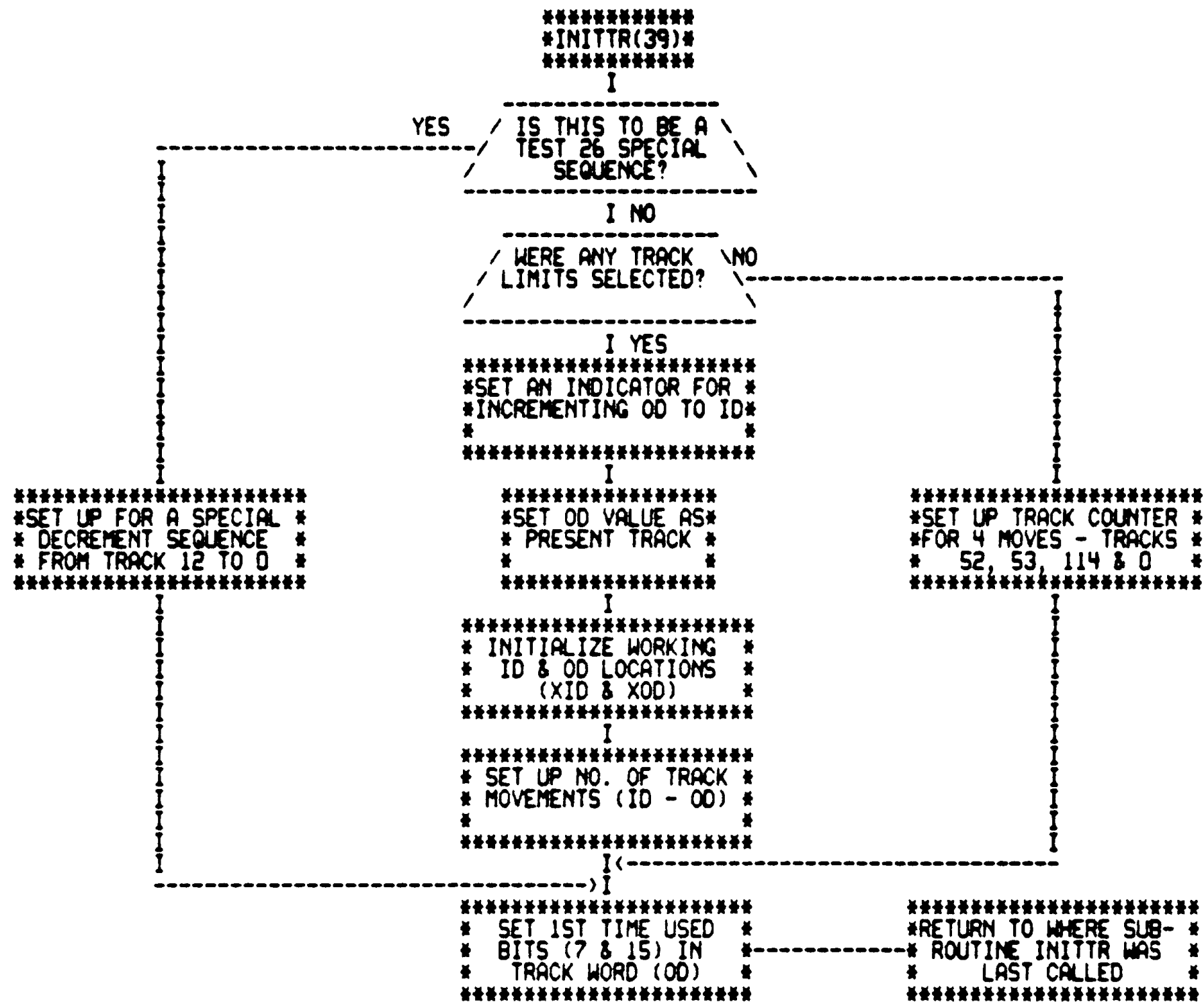


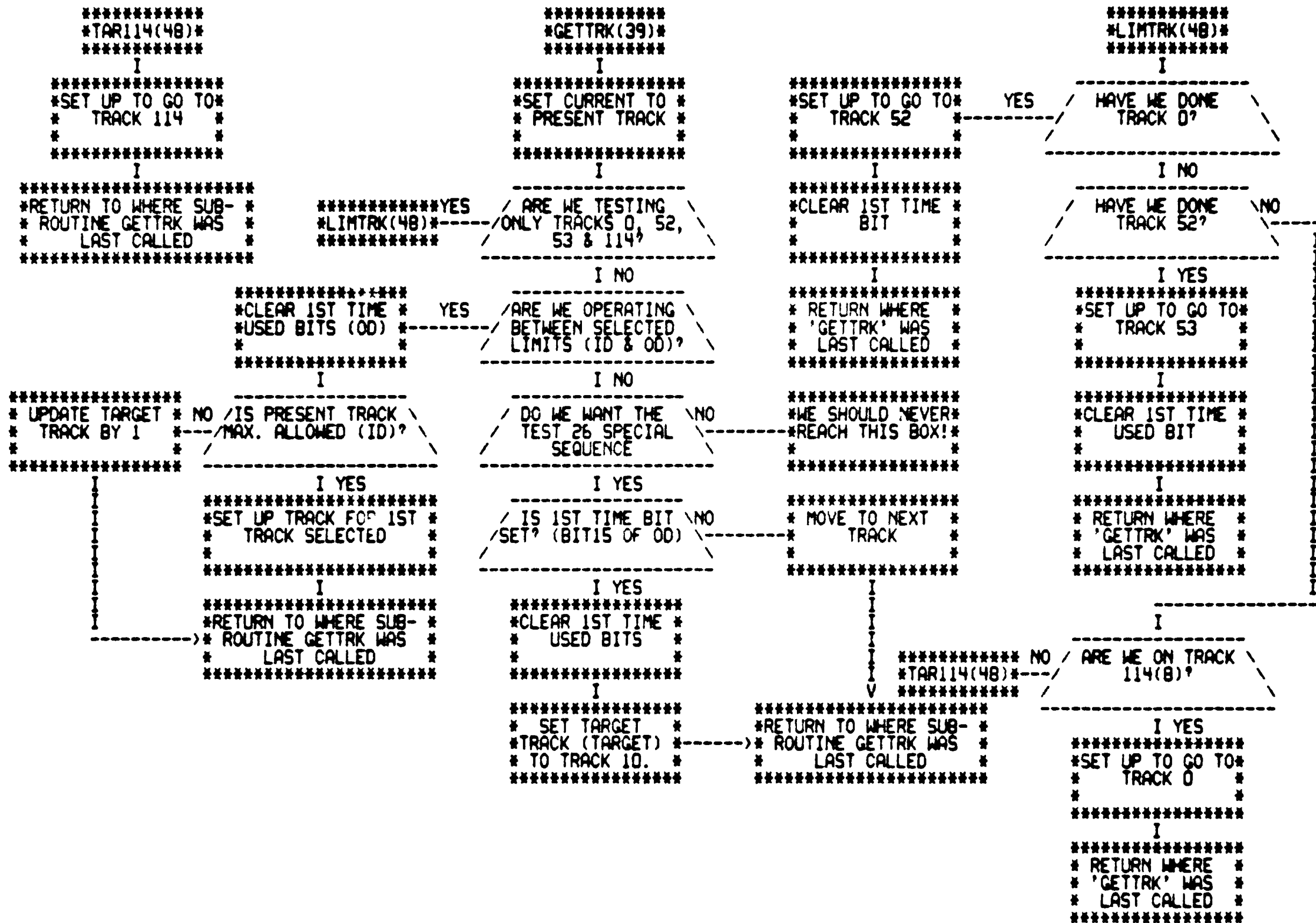


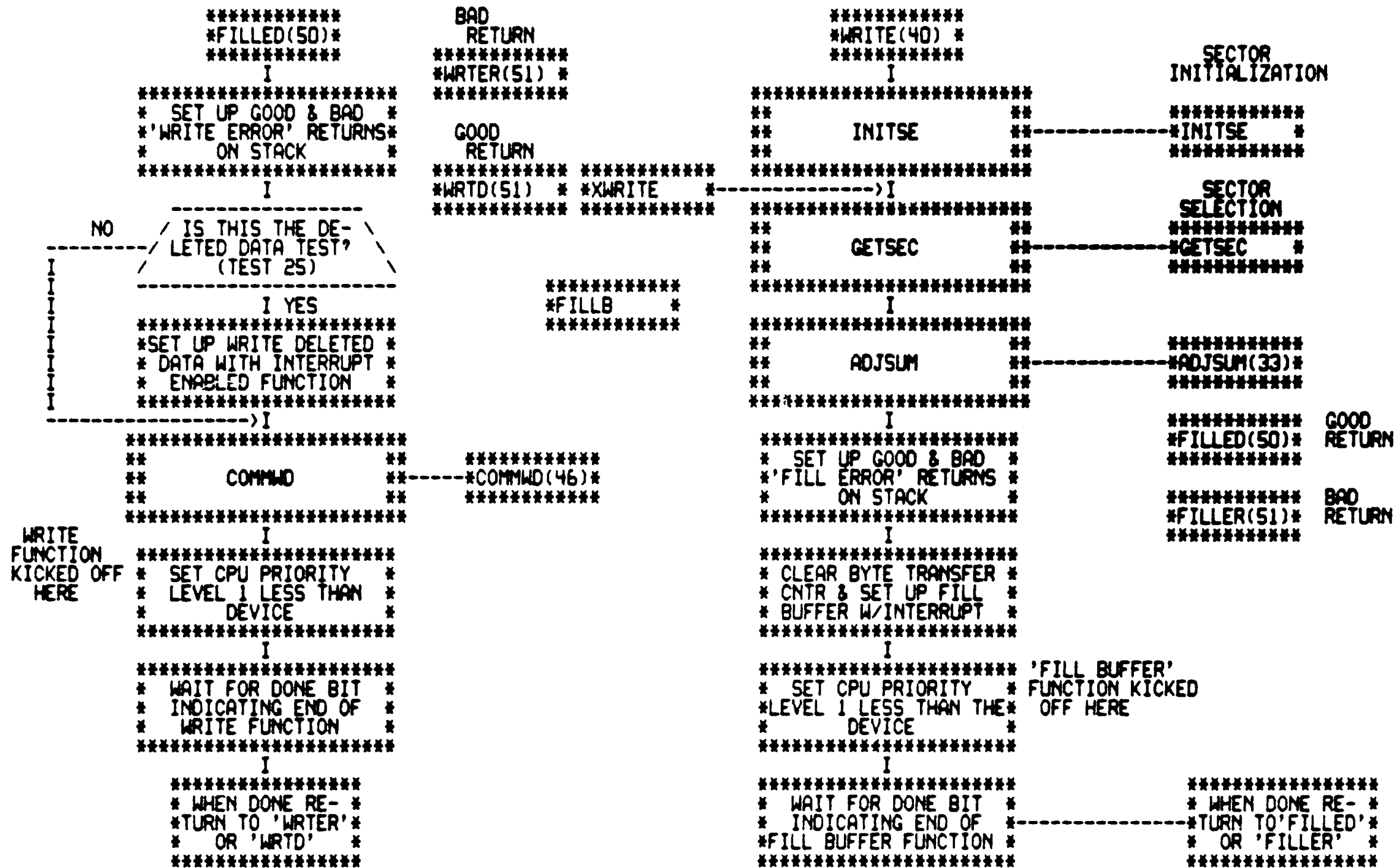













```
*****
*RDCHK(40) *
*****
I
*****
* READ A SECTOR *
*      &      *
* EMPTY SECTOR BUFFER *
*****
I
*****
* VERIFY DATA READ *
*AGAINST DATA IN CORE *
*      BUFFER      *
*****
I
*****
*EVENTUALLY RETURN TO *
*WHERE SUBR. RDCHK WAS*
*      LAST CALLED   *
*****
```

[illegible]

FLOW CHART FOR RX11 DIAGNOSTIC INTERFACE
 FLOW CHART CROSS REFERENCE LIST

T14	21					
T14R08	21	21				
T15	22					
T15R08	22	22				
T16	23					
T16R08	23	23				
T17	24					
T17R08	24	24				
T2	25					
T20	25					
T21	26					
T21WR	26	26				
T21X	27	27	30			
T22	28					
T23	30					
T24	31					
T24X	31	31				
T25	31					
T26	32					
T3	11					
T4	12					
T5	13					
T6	16					
T6FILL	16	16	18	19	28	30
T7	17					
T7EMPTY	17	17	18	19	27	29
TAR114	48	48				
TRAGAN	37	37				
TROVER	37	37				
TSTMOR	06	34	35	35		
WRITE	40	40	42	50		
WRID	50	51	51			
WATER	50	51				
WTROCK	32	42				
XREAD	49	49				
XSA202	01	02	03			
XWRITE	50	51	51			
XWTRD	42	43				
1STST	45					
1STTST	06					
2500N	37	37	38			

85	BASIC DEFINITIONS
243	TEST SELECTION VIA SWITCH REGISTER
263	OPERATIONAL SWITCH REGISTER POSITIONS
289	RXCS (RX COMMAND STATUS REGISTER)
340	RXDB (RX DATA BUFFER REGISTER)
392	START AND RESTART ADDRESSES
504	PRETEST - INITIALIZE [KEY] PART I
851	TEST 1 - RXCS TEST PART I / INTERRUPT TEST PART I
1013	TEST 2 - INTERRUPT TEST PART II / VECTOR ADDRESS VERIFICATION
1251	TEST 3 - INTERRUPT TEST PART III / PRIORITY LEVEL VERIFICATION PART I
1309	TEST 4 - INTERRUPT TEST PART IV / PRIORITY VERIFICATION PART II
1369	TEST 5 - INIT (PROGRAMMED) / RST
1555	TEST 6 - FILL BUFFER TRANSFER LENGTH VERIFICATION
1656	TEST 10 - EMPTY BUFFER XFER LENGTH AND CONTENT VERIFICATION PART II
1664	TEST 7 - EMPTY BUFFER XFER LENGTH AND CONTENT VERIFICATION PART I
1745	TEST 12 - FILL/EMPTY BUFFER ALL 1'S
1752	TEST 11 - FILL/EMPTY BUFFER ALL 0'S
1764	TEST 13 DRIVE READY VERIFICATION
1837	TEST 14 - ERROR FLAG AND B-CODE VERIFICATION PART I
1972	TEST 15 - ERROR FLAG AND B-CODE VERIFICATION PART II
2025	TEST 16 - ERROR FLAG AND B-CODE VERIFICATION PART III
2108	TEST 17 - ILLEGAL TRACK ERROR VERIFICATION
2213	TEST 20 - SEEK VERIFICATION VIA READ FUNCTION
2251	TEST 21 - WRITE TEST
2320	TEST 22 - INITIALIZE IMPLIED READ
2342	TEST 23 - READ TEST
2357	TEST 24 - DATA TRANSFER AND VERIFICATION
2371	TEST 25 - DATA TRANSFER AND VERIFICATION VIA DELETED DATA MODE
2379	TEST 26 - HEAD "HOME" TEST
2447	" ERROR " TRAP SERVICE ROUTINE
2519	" SCOPE " TRAP SERVICE ROUTINE
2631	DRIVE TEST SELECTION
2678	WRITE FUNCTION
2909	READ DATA FROM THE DISKETTE
2933	READ AND VERIFY DATA
3067	INTERRUPT SERVICE
3173	PATTERN GENERATOR
3316	UNIT SELECTION
3359	TRACK SEQUENCE SELECTION
3451	SECTOR SELECTION
3486	TYPE ROUTINE
3578	BINARY TO OCTAL (ASCII) AND TYPE
3656	SAVE AND RESTORE R0-R5 ROUTINES
3702	TRAP DECODER
3718	TRAP TABLE
3735	POWER DOWN AND UP ROUTINES
3777	SINGLE LENGTH BINARY TO DECIMAL ASCII ROUTINE
3795	DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
3888	MESSAGES

```

.NLIST CND,MD,MC
.LIST ME
.ENABL ABS,AMA
.MCALL .HEADER,.EQUAT,.STYPE,.STYPOCT,.STRAP,TYPOCS
.MCALL SETPRI,.SPOWER,STARS,.SSB2D,.S0B2D,.SSAVE,COMMENT
.MCALL ENDCOMMENT

```

```

.TITLE MAINDEC-11-DZRXB-D
*COPYRIGHT (C) DEC 21, 1975
*DIGITAL EQUIPMENT CORP.
*MAYNARD, MASS. 01754
*
*PROGRAM BY D. ADAMS/B. BURGESS
*
*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
*PACKAGE (MAINDEC-11-DZQAC-B1), AUG 29, 1975.
*

```

```

000001
160000

```

```

$TN=1
$SWR=160000 ;;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT

```

```

;THIS SOFTWARE IS FURNISHED UNDER LICENCE 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 LICENCE TERMS. TITLE TO 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.

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41

```

42
43
44
45
46
47
48
49
50
51
52
53
54
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

:MODIFIED TO REV. D BY B. BURGESS NOV. 10, 1975 AS FOLLOWS:

- :A) ADDED CAPABILITY OF VARIABLE DEVICE 'BR' LEVEL. ALL RELEVANT TESTS
CALCULATE 'CPU' LEVEL BASED ON CURRENT CONTENTS OF LOCATION 'BRLEV:'.
DEFAULT 'BR' LEVEL, FOR THE DEVICE, SET BY THE PROGRAM IS 5. ANY OTHER
'BR' LEVEL (E.G. 6) WOULD HAVE TO BE PATCHED INTO LOCATION 'BRLEV:'
BEFORE RUNNING THE PROGRAM.
- :B) ADDED TWO (2) ROUTINES TO HANDLE 'UNEXPECTED' BUS TIMEOUT AND
RESERVED INSTRUCTION TRAPS (TRAPS TO VECTORS 4 & 10, RESPECTIVELY).
BOTH ROUTINES WILL INDICATE WHICH TRAP OCCURRED, THE 'PC' LOCATION
OF WHERE THE TRAP OCCURRED, AND ATTEMPT TO RESTART THE PROGRAM.
- :C) ADDED CODE TO FAILSAFE UNIT 0 UNDERGOING TESTING IF PROGRAM WAS
LOADED VIA UNIT 0 USING 'RXDP' MONITOR AND USER STARTED RUNNING
THE PROGRAM WITHOUT HAVING REPLACED HIS LOAD MEDIUM WITH A 'SCRATCH'
DISKETTE.
- :D) ADDED MESSAGES TO INDICATE TO USER WHEN HE HAS SELECTED TRACK AND/OR
SECTOR LIMITS 'OUT OF RANGE' AND CORRESPONDING DEFAULT LIMITS WHEN
THIS CONDITION ARISES
- :E) MODIFIED TESTS 1 THRU 4 TO CORRECTLY PRINT OUT THE CONTENTS OF
'KRXVEC' (LOCATION HOLDING THE DEVICE VECTOR) AS 264 INSTEAD OF 270.
- :F) MODIFIED TEST 2 TO HANDLE A 'LOCKED IN INTERRUPT STATE' CONDITION
ARISING WHEN 'INTERRUPT ENABLE' AND 'DONE' ARE BOTH QUALIFIED AND
THE 'REQUEST INTERRUPT' FLOP NEVER GETS CLEARED.
- :G) ADDED EXTENSIVE MAINTENANCE INFORMATION BASED ON FAULT INSERTION
RESULTS. INFORMATION IS KEYED TO THE 'ERROR' REPORT WITHIN A
TEST. INFORMATION PROVIDED SHOULD BE SELF-EXPLANATORY BUT SHOULD
NOT BE MISCONSTRUED AS BEING ALL ENCOMPASSING DUE TO HUMAN ERRORS
IN STATISTICS GATHERING, INABILITY TO FAULT INSERT SOME CHIPS, AS
WELL AS ONLY TWO (2) MODULES ABLE TO BE FAULT INSERTED I.E. -
M7846 (UNIBUS INTERFACE) AND M7727 (READ/WRITE CONTROL).
- :H) ADDED FLOW CHARTS

```

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
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135

```

.SBTTL BASIC DEFINITIONS

;;*INITIAL ADDRESS OF THE STACK POINTER *** 1200 ***

STACK= 1200

;;EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL

;;EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL

PS= 177776 ;;PROCESSOR STATUS WORD

;;EQUIV PS,PSW

STKLM= 177774 ;;STACK LIMIT REGISTER

PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER

DSWR= 177570 ;;HARDWARE SWITCH REGISTER

DDISP= 177570 ;;HARDWARE DISPLAY REGISTER

;;*GENERAL PURPOSE REGISTER DEFINITIONS

R0= %0 ;;GENERAL REGISTER

R1= %1 ;;GENERAL REGISTER

R2= %2 ;;GENERAL REGISTER

R3= %3 ;;GENERAL REGISTER

R4= %4 ;;GENERAL REGISTER

R5= %5 ;;GENERAL REGISTER

R6= %6 ;;GENERAL REGISTER

R7= %7 ;;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

```

136      .EQUIV SW09,SW9
137      .EQUIV SW08,SW8
138      .EQUIV SW07,SW7
139      .EQUIV SW06,SW6
140      .EQUIV SW05,SW5
141      .EQUIV SW04,SW4
142      .EQUIV SW03,SW3
143      .EQUIV SW02,SW2
144      .EQUIV SW01,SW1
145      .EQUIV SW00,SW0
146
147      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
148      100000      BIT15= 100000
149      040000      BIT14= 40000
150      020000      BIT13= 20000
151      010000      BIT12= 10000
152      004000      BIT11= 4000
153      002000      BIT10= 2000
154      001000      BIT09= 1000
155      000400      BIT08= 400
156      000200      BIT07= 200
157      000100      BIT06= 100
158      000040      BIT05= 40
159      000020      BIT04= 20
160      000010      BIT03= 10
161      000004      BIT02= 4
162      000002      BIT01= 2
163      000001      BIT00= 1
164      .EQUIV BIT09,BIT9
165      .EQUIV BIT08,BIT8
166      .EQUIV BIT07,BIT7
167      .EQUIV BIT06,BIT6
168      .EQUIV BIT05,BIT5
169      .EQUIV BIT04,BIT4
170      .EQUIV BIT03,BIT3
171      .EQUIV BIT02,BIT2
172      .EQUIV BIT01,BIT1
173      .EQUIV BIT00,BIT0
174
175      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
176      000004      ERRVEC= 4      ;: TIME OUT AND OTHER ERRORS
177      000010      RESVEC= 10     ;: RESERVED AND ILLEGAL INSTRUCTIONS
178      000014      TBITVEC=14     ;: "T" BIT
179      000014      TRIVEC= 14     ;: TRACE TRAP
180      000014      BPTVEC= 14     ;: BREAKPOINT TRAP (BPT)
181      000020      IOTVEC= 20     ;: INPUT/OUTPUT TRAP (IOT) **SCOPE**
182      000024      PWRVEC= 24     ;: POWER FAIL
183      000030      EMTVEC= 30     ;: EMULATOR TRAP (EMT) **ERROR**
184      000034      TRAPVEC=34     ;: "TRAP" TRAP
185      000060      TKVEC= 60      ;: TTY KEYBOARD VECTOR
186      000064      TPVEC= 64      ;: TTY PRINTER VECTOR
187      000240      PIRQVEC=240    ;: PROGRAM INTERRUPT REQUEST VECTOR
188
189

```

```

190                                     ;SPECIAL EQUATES
191
192
193
194         000017         RDER      =17           ; READ B CODE
195         000040         DONEBIT  =40
196         000101         FBIE     =101          ; IE+FILL BUFFER
197         000103         EBIE     =103          ; IE+EMPTY BUFFER
198         000105         WRTIE    =105          ; IE+WRITE SECTOR
199         000107         RDIE     =107          ; IE+READ SECTOR
200         000115         WTDIE    =115          ; IE+WRITE DD SECTOR
201         040001         RECAL    =40001
202         000000         OPEN     =0
203
204         000000         . =0
205 000000 000000 000000         .WORD 0,0
206
207         000004         . =4
208 000004 005756         .WORD  BUSERR          ; UNEXPECTED TIMEOUT TRAP PC
209 000006 000340         .WORD  PR7            ; UNEXPECTED TIMEOUT TRAP PS
210 000010 006002         .WORD  RESERR         ; UNEXPECTED RESERVED INSTRUCTION TRAP PC
211 000012 000340         .WORD  PR7            ; UNEXPECTED RESERVED INSTRUCTION TRAP PS
212
213         000020         . =20
214 000020 006316         XSCOPE
215 000022 000340         PR7
216 000024 014044         $PWDRN
217 000026 000340         PR7
218 000030 006060         XERROR
219 000032 000340         PR7
220 000034 014004         $TRAP
221 000036 000340         PR7          ; ADDRESS OF TRAP SERVICE
222
223         000046         . =46
224 000046 002434         LOGICAL          ; ACT 11 EOP HOOKS
225
226
227         000052         . =52
228 000052 000000         .WORD 0
229
230         000174         . =174
231 000174 000000         DISPREG:          0
232 000176 000000         SWREG:          0
233
234         000200         . =200
235 000200 000401         BR 1$
236 000202 000402         BR 2$
237 000204 000137 001232 1$: JMP SA200
238 000210 000137 001222 2$: JMP SA202          ; OPERATOR SELECTED CONDITIONS
                                           ; RESTART PROGRAM WITH PREVIOUS PARAMETERS

```

K07

MAINDEC-11-DZRXB-D MACY11 27(663) 6-NOV-75 08:50 PAGE 6
DZRXBD.P11 BASIC DEFINITIONS

SEQ 0087

239
240
241

.SBTTL TEST SELECTION VIA SWITCH REGISTER

```

*****
*****
*****
; SET TEST AND DRIVE SELECTION IN " DTESTP " LOCATION 1212
; BIT 15 = 1 - UNIT 1 SELECTED
; BIT 14 = 1 - UNIT 0 SELECTED
; BIT 15 & BIT 14 = 0 - BOTH DRIVES MUST BE READY
; BIT 4 - BIT 0 = OCTAL NUMBER OF DESIRED STARTING TEST
; BIT 4 - BIT 0 = 0 -ALL TESTS WILL BE SEQUENCED THROUGH
*****
*****
*****

```

.SBTTL OPERATIONAL SWITCH REGISTER POSITIONS

```

*****
*****
*****
; SET OPERATING CONDITIONS IN THE SWITCH REGISTER (HARDWARE)
; OR SOFTWARE SWITCH REGISTER LOCATION 176
; 15 = 1 - HALT ON ERROR
; 14 = 1 - HALT AT END OF PASS
; 13 = 1 - INHIBIT ERROR TYPEOUT
; 12 = 1 - LOOP ON TEST
; 11 = 1 - LOCK ON ERROR
; 10 = 1 - HALT AT END OF TEST
; 9 = 1 - PRINT ALL DATA ERRORS
; 9 = 0 - PRINT ONLY FIRST 10 DATA ERRORS PER SECTOR
; 8 = 1 - INHIBIT RECALIBRATION ON SEEK ERRORS.
; 0 = 1 - INHIBIT <BELL> AT ERROR
; 15-0 = 1 - SELECT SOFTWARE SWITCH REGISTER
*****
*****
*****

```

242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287

```

288 .SBTTL RXCS (RX COMMAND STATUS REGISTER)
289
290 001200 . =STACK
291
292 001200 000000 00: 0 ;00/ID = 0 UNLESS SPECIFIC TRACKS SELECTED.
293 001201 ID=00+1
294 001202 015001 FIRST: 015001 ; FIRST = 1, LAST = 32
295 001203 LAST=FIRST+1
296
297 001204 000264 KRXVEC: 264
298
299 001206 177170 RXCS: 177170
300
301 ; RXCS: STANDARD DEVICE ADDRESS = 177170
302
303 ; TOGGLE INTO PROGRAM LOCATION " RXCS " THE RX11 DEVICE ADDRESS IF NOT = 177170
304
305 ; KEY: R - READ ONLY BIT
306 ; W - WRITE ONLY BIT
307
308 :
309 15 - R - ERROR
310 14 - W - INITIALIZE
311 13 -
312 12 -
313 11 - (BITS 13-8)
314 10 - (NOT USED)
315 9 -
316 8 -
317 7 - R - TRANSFER REQUEST
318 6 - R/W - INTERRUPT ENABLE
319 5 - R - DONE
320 4 - W - UNIT SELECT
321 3 - W - FUNCTION
322 2 - W - FUNCTION
323 1 - W - FUNCTION
324 0 - W - GO !
325
326 ; FUNCTION
327 ; 3 2 1 0
328
329 ; - - - GO
330
331 : 0 + GO - FILL BUFFER
332 : 2 + GO - EMPTY BUFFER
333 : 4 + GO - WRITE SECTOR
334 : 6 + GO - READ SECTOR
335 :
336 : 12 + GO - READ STATUS " A "
337 : 14 + GO - WRITE DELETED DATA
338 : 16 + GO - READ STATUS " B " (CODES)

```

```

339 .SBTTL RXDB (RX DATA BUFFER REGISTER)
340
341 001210 177172 RXDB: 177172
342
343 ; RXDB: STANDARD DEVICE ADDRESS = 177172
344
345 ; THE FOLLOWING BIT IDENTIFICATION REPRESENTS THE STATUS AT THE END OF A FUNCTION
346 ; (BUT NOT FUNCTION # 16 TO READ STATUS " B ") DISPLAYED WITHIN THE RX-DATA BUFFER.
347
348 ; (A) 7 - SELECTED DRIVE READY
349 ; 6 - DELETED DATA
350 ; 5 -
351 ; 4 -
352 ; (B) 3 - WRITE PROTECT ERROR
353 ; 2 - INITIALIZE DONE
354 ; 1 - PARITY ERROR
355 ; 0 - CRC ERROR
356
357 ; (A) - VISIBLE ONLY IF THE FUNCTION WAS # 12 READ STATUS " A "
358
359 ; (B) - INIT DONE VISIBLE IF AN INITIZLAE (KEY) OR (PROGRAMMED) WAS ISSUED
360
361 001212 000000 DTESTP: 0
362
363 001214 000005 BRLEV: 5
364
365 ;BRLEV: STANDARD PRIORITY INTERRUPT LEVEL = 5
366
367 ;TOGGLE INTO PROGRAM LOCATION "BRLEV" THE RX11 INTERRUPT PRIORITY
368 ;LEVEL IF NOT = 5
369
370 001216 177570 SWR: .WORD DSWR
371 001220 177570 DISPLAY: .WORD DDISP
372
373 ; R0 - GOOD /EXPECTED RESULT OF TEST
374 ; R1 - EAC /ACTUAL RESULT OF TEST
375 ; R2 - BLANK /
376 ; R3 - TEST Q
377
378 ;*****
379
380 ;WORD "UNITSEL" HAS THE FOLLOWING BIT DEFFINITIONS
381
382 ;BIT15 = 1 - UNIT 1 SELECTED FOR USE
383 ;BIT14 = 1 - UNIT 1 IN USE
384 ;BIT8 = 1 - THIS PASS HAD AN ERROR
385 ;BIT7 = 1 - UNIT 0 SELECTED FOR USE
386 ;BIT6 = 1 - UNIT 0 IN USE
387 ;BIT4 = UNIT SELECTION BIT
388
389 ;*****
390

```

.SBTTL START AND RESTART ADDRESSES

; THE STARTING ADDRESS WAS 202

SA202: INC R0
MOV #STACK, SP
BR RESTART

; THE STARTING ADDRESS WAS 200

SA200: CLR R0
MOV #STACK, SP
MOV 4, -(SP)
MOV #15, 4
CMP #177777, JSWR
BEQ 2\$
BR 3\$
1\$: CMP (SP)+, (SP)+
2\$: MOV #SWREG, SWR
MOV #DISPREG, DISPLAY
3\$: MOV (SP)+, 4
RESTART: MOV #PR7, -(SP)
MOV #4\$, -(SP); SAVE 'BUSERR' TIMEOUT 'PC'
; SET UP TIMEOUT VECTOR
; IS SOFTWARE SWR SELECTED
; YES, INSERT IT'S ADDRESS
; BR IF NO TIMEOUT TRAP OCCURS
; RESTORE THE STACK
; POINT TO SOFTWARE SWITCH REGISTER
; POINT TO SOFTWARE DISPLAY REG.
; RESET TIMEOUT VECTOR TO 'BUSERR'RTI
4\$: MOV RXCS, RXDB
ADD #2, RXDB
MOV #001234, RAN1
MOV #000765, RAN2
CLR CCOUNT
CLR PASS
CLR HANGER
TST R0
BNE XSA202
TYPE, MREV
CLR UNITSEL
BIT #140000, DTESTP
BNE 1\$
BIS #100200, UNITSEL
BR XSA202
1\$: BIT #BIT14, DTESTP
BEQ 3\$
CMPB 41, #10; LOAD THE PSW
; GET ADDRESS OF RXCS
; SET UP ADDRESS OF RXDB
; INITIALIZE CONSTANTS OF
; RANDOM NUMBER GENERATOR; STARTING ADDRESS WAS 202
; PRESENT MAINDEC REVISION; WERE ANY DRIVES SELECTED
; YES GO SET THEIR BITS
; NO, BOTH UNITS MUST BE READY; WAS UNIT 0 SELECTED
; NO, MUST BE UNIT 1
; WAS PROGRAM LOADED IN DUMP MODE
; VIA XXDP?; BRANCH IF NOT
; INFORM USER TO REMOVE LOAD MEDIUM
; FROM UNIT 0 AND REPLACE WITH
; A 'SCRATCH' DISKETTE IF HE; WISHES TO TEST UNIT 0
; WAIT FOR USER RESPONSE
; YES, SET SELECTED BIT
; WAS UNIT 1 SELECTED; NO
; SET SELECTED BIT
; CLEAR 1ST TIME BITS FOR BOTH DRIVES391
392
393
394
395 001222 005200
396 001224 012706 001200
397 001230 000426
398
399
400
401 001232 005000
402 001234 012706 001200
403 001240 013746 000004
404 001244 012737 001264 000004
405 001252 022777 177777 177736
406 001260 001402
407 001262 000407
408 001264 022626
409 001266 012737 000176 001216
410 001274 012737 000174 001220
411 001302 012637 000004
412 001306 012746 000340
413 001312 012746 001320
414 001316 000002
415 001320 013737 001206 001210
416 001326 062737 000002 001210
417 001334 012737 001234 012350
418 001342 012737 000765 012352
419 001350 005037 002336
420 001354 005037 002340
421 001360 005037 006560
422 001364 005700
423 001366 001040
424 001370 104400 015605
425 001374 005037 012446
426 001400 032737 140000 001212
427 001406 001004
428 001410 052737 100200 012446
429 001416 000424
430 001420 032737 040000 001212
431 001426 001415
432 001430 123727 000041 000010
433
434 001436 001003
435 001440 104400 016125
436
437
438
439 001444 000000
440 001446 052737 000200 012446
441 001454 005737 001212
442 001460 100003
443 001462 052737 100000 012446
444 001470 042737 100200 001200
2\$: BIS #200, UNITSEL
TST DTESTP
BPL XSA202
3\$: BIS #BIT15, UNITSEL
XSA202: BIC #100200, 00

008

MAINDEC-11-DZRXB-D MACY11 27(663) 6-NOV-75 08:50 PAGE 12
DZRXBD.P11 START AND RESTART ADDRESSES

SEQ 0093

499
500
501 001716 012737 015001 001202 MOV #15001,FIRST
502 001724 000755 BR 35

;A 1ST SECTOR VALUE OF 1 AND
;A LAST SECTOR VALUE OF 32
;SET FIRST TO 1 LAST TO 32

```

503 .SBTTL PRETEST - INITIALIZE (KEY) PART I
504
505 "KEY" INITIALIZE SHOULD HAVE (SET) THE DONE FLAG BECAUSE
506 ANY (INIT) OF THE RX01 MICROCONTROLLER IS AN IMPLIED (READ SECTOR)
507 OF TRACK 1 SECTOR 1 (FOR SYSTEMS PROGRAMMING BOOTSTRAP APPLICATIONS).
508
509 THEREFORE, ANY ERROR (EXCEPT PARITY) THAT MAY OCCUR FROM A NORMAL
510 "READ SECTOR" COMMAND MAY OCCUR HERE CAUSING THE ERROR FLAG TO SET, AND
511 DISPLAYING THE ERROR STATUS WITHIN THE TRANSFER REGISTER AT "DONE".
512
513 THE TRANSFER REQUEST FLAG SHOULD BE CLEARED.
514
515 ; NOTE: SCOPE LOOPING IS NOT OFFERED BECAUSE THE "INIT" FUNCTION
516 ; WHICH PRODUCED THE ERROR HAS NOT YET BEEN VERIFIED.
517
518 001726 042737 000400 012446 PRETEST: BIC #BIT8,UNITSEL
519 001734 005037 006312 CLR ERRORS
520 001740 000004 SCOPE ; REFRESHES FAST FOR ERROR TYPEOUT
521 001742 013737 006374 002342 MOV PCSCOPE,FAST ; FOR ERROR TYPEOUT INFORMATION
522 001750 012700 000040 MOV #40, R0 ; PROGRAM EXPECTS DONE FLAG
523 001754 017701 177226 MOV 2 RXCS, R1 ; ACTUAL CONTENTS OF RXCS
524 001760 020100 CMP R1, R0
525 001762 001401 BEQ 1$ ; OK
526
527 ; (R0) = 40 ; (R1) = ACTUAL RXCS ; (R2) = N/A
528
529 001764 104000 ERROR ; RXCS NOT = 40 (DONE)
530

```


Address	Hex	Dec	Label	Instruction	Comment
706	002030	000522		BR	REBEGIN
707	002032	004737	002366	MORETESTS:	JSR PC, LOCKUP
708	002036	005723		TONOTHERE:	TST (R3)+
709	002040	016337	002064	FIRSTTEST:	MOV TESTS(R3), PCSCOPE
710	002046	013737	006374		MOV PCSCOPE, FAST
711	002054	012706	002342		MOV #STACK, SP
712	002060	000173	002064		JMP 2TESTS(R3)
713	002064	002036	002446	TESTS:	TONOTHERE, T1, T2, T3, T4, T5, T6, T7
714	002072	003110	003246		
715	002100	003636	003770		
716	002104	003766	004054		T10, T11, T12, T13, T14, T15, T16, T17
717	002112	004120	004244		
718	002120	004576	005044		

```

719 002124 005310 005334 005520          T20, T21, T22, T23, T24, T25, T26, NOMORETESTS
720 002132 005576 005644 005676
721 002140 005710 002144
722
723      ; TEST 1 - RXCS TEST PART I / INTERRUPT TEST PART I
724
725      ; TEST 2 - INTERRUPT TEST PART II / VECTOR ADDRESS VERIFICATION
726
727      ; * TEST 3 - INTERRUPT TEST PART III / PRIORITY LEVEL VERIFICATION PART I
728
729      ; * TEST 4 - INTERRUPT TEST PART IV / PRIORITY VERIFICATION PART II
730
731      ; TEST 5 - INIT (PROGRAMMED) / RST
732
733      ; TEST 6 - FILL BUFFER TRANSFER LENGTH VERIFICATION
734
735      ; TEST 7 - EMPTY BUFFER TRANSFER LENGTH AND CONTENT VERIFICATION PART I
736
737      ; TEST 10 - EMPTY BUFFER TRANSFER LENGTH AND CONTENT VERIFICATION PART II
738
739      ; TEST 11 - FILL/EMPTY BUFFER ALL 0'S
740
741      ; TEST 12 - FILL/EMPTY BUFFER ALL 1'S
742
743      ; TEST 13 - DRIVE READY VERIFICATION FOR SELECTED DRIVES
744
745      ; TEST 14 - ERROR FLAG AND B - CODE VERIFICATION PART I
746
747      ; TEST 15 - ERROR FLAG AND B - CODE VERIFICATION PART II
748      ; /DELETED DATA BIT SETS
749
750      ; TEST 16 - ERROR FLAG AND B - CODE VERIFICATION PART III
751      ; /DELETED DATA BIT CLEARS
752
753      ; TEST 17 - ILLEGAL TRACK ERROR AND B - CODE VERIFICATION
754
755      ; TEST 20 - SEEK VERIFICATION VIA READ FUNCTION
756
757      ; TEST 21 - WRITE TEST
758
759      ; TEST 22 - INITIALIZE IMPLIED READ
760
761      ; TEST 23 - READ TEST
762
763      ; TEST 24 - DATA TRANSFER & VERIFICATION
764
765      ; TEST 25 - DATA TRANSFER & VERIFICATION VIA DELETED DATA MODE
766
767      ; TEST 26 - HEAD "HOME" TEST
768
769      ; THERE ARE NO MORE TESTS
770
771      ; * NOTE: ON PROCESSORS WITHOUT HARDWARE PROCESSOR STATUS WORDS (PSW)
772      ; THESE TEST WILL NOT BE RUN.

```

```

773 ;PRINT AN END OF PASS INDICATOR
774
775 ; C - RX11/RX01 TEST PASS OK
776 ; D - RX11/RX01 AND DRIVE TESTING OK
777 ; - - AN ERROR OCCURRED (DURING C OR D)
778
779
780 ; NOTE: IF BIT 8 OF UNITSEL IS A 1
781 ; THEN AN ERROR HAS OCCURRED FOR THIS PASS
782
783 002144 042777 000100 177034 NOMORETESTS: BIC #BIT6,DRXCS ;CLEAR 'IE' BIT BEFORE NEXT PASS
784 002152 032737 000400 012446 BIT #BIT8,UNITSEL ;"C" OR "D" MEANS ERRORLESS PASS.
785 002160 001403 BEQ 1$
786 002162 012737 000055 002344 MOV #'-,MX ; " - " MEANS UN-ERRORLESS PASS
787 002170 005737 002336 1$: TST CCOUNT
788 002174 001002 BNE 3$
789 002176 104400 015030 TYPE,MCRLF
790 002202 005237 002336 3$: INC CCOUNT
791 002206 022737 000110 002336 CMP #72.,CCOUNT
792 002214 001002 BNE 4$
793 002216 005037 002336 CLR CCOUNT
794 002222 104400 002344 4$: TYPE,MX
795 002226 104400 006310 TYPE,MABELL
796 002232 005237 002340 2$: INC PASS
797 002236 102775 BVS 2$
798 002240 032777 040000 176750 BIT #SW14,DSWR ; AC SW 14 = 1 TO HALT AT END OF PASS
799 002246 001413 BEQ REBEGIN
800 002250 104400 015030 TYPE,MCRLF
801 002254 104400 006527 TYPE,MPASS
802 002260 013737 002340 002272 MOV PASS,5$
803 002266 004537 014504 JSR RS,SGLDEC
804 002272 000000 5$: OPEN
805 002274 000000 HALT
806 002276 042737 000400 012446 REBEGIN: BIC #BIT8,UNITSEL ;CLEAR HARD ERROR INDICATOR
807 002304 013703 001212 MOV DTESTP,R3 ; R3 CONTAINS TEST # 0 TO 26
808 002310 042703 177740 BIC #177740,R3
809 002314 020327 000027 CMP R3,#27
810 002320 103002 BHS 1$
811 002322 006303 ASL R3
812 002324 000645 BR FIRSTTEST
813
814 002326 104400 002346 1$: TYPE,MILTST
815 002332 000137 001232 JMP SA200
816
817 002336 000000 CCOUNT: 0
818 002340 000000 PASS: 0
819 002342 000000 FAST: 0
820
821 002344 000103 MX: .ASCIZ "C"
822
823 002346 046111 042514 040507 MILTST: .ASCIZ "ILLEGAL TEST"(15)(12)
824 002354 020114 042524 052123
825 002362 005015 000
826

```

K08

MAINDEC-11-DZRXB-D MACY11 27(663) 6-NOV-75 08:50 PAGE 19
DZRXBD.P11 PRETEST - INITIALIZE [KEY] PART I

SEQ 0100

827

002366

.EVEN

```

828
829                ; DATA SW 10 = 1 TO HALT AT END OF TEST
830
831 002366 032777 002000 176622 LOCKUP: BIT #SW10, @SWR
832 002374 001401          BEQ 1$
833 002376 000000          HALT
834
835                ; DATA SW 12 = 1 TO LOCK SCOPE LOOP ON TEST OK OR NOT
836
837 002400 032777 010000 176610 1$: BIT #SW12, @SWR                ; IS LOOP ON TEST SWITCH SET
838 002406 001403          BEQ 2$                ; IF NOT SET GO ON TO NEXT TEST
839 002410 062716 000002          ADD #2, @SP                ; IF SET RETURN TO FIRSTTEST
840 002414 000207          RTS PC
841 002416 042737 040100 012446 2$: BIC #40100, UNITSEL        ; CLEAR UNIT USED BITS
842 002424 013705 000042          MOV @#42, R5                ; ACT 11 END OF PASS HOOKS
843 002430 001405          BEQ HERE
844 002432 000005          RESET
845 002434 004715          LOGICAL: JSR PC, (R5)
846 002436 000240          NOP
847 002440 000240          NOP
848 002442 000240          NOP
849 002444 000207          HERE: RTS PC

```


904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926

```

////////////////////////////////////
RXDB AT "DONE"
  7      6      -      -      3      2      1      0      /
SEL      DRIVE  DD      WRITE  INIT  PAR      /
RDY      PROTECT [DONE] (N/A)  CRC      /
////////////////////////////////////

```

```

002472 005000          CLR R0
002474 017701 176510  MOV 2 RXDB, R1
002500 020100          CMP R1, R0
002502 001401          BEQ 3$

; (R0) = 0 ; (R1) = ACTUAL RXDB ; (R2) = N/A

002504 104000          ERROR ; RXDB NOT = 0

```

;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
;THIS ERROR REPORT WAS GENERATED. THE INFORMATION SHOWN IS
;BASED ON FAULT INSERTION RESULTS, AND SHOULD PROVIDE LOGICAL
;AREAS TO CHECK FOR THE RELEVANT FAULT/S.
;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
;ANALYZE THE FOLLOWING AREA/S:

M7846 (UNIBUS INTERFACE)		
SIGNAL NAME	REASON	POSSIBLE CHIPS
BUS D01 -> D03 LOAD ----- -----	STUCK HIGH	E7
	STUCK LOW	E17,E18
	NO SACK	E12
IN RX BUSY	CLOCK STUCK HIGH/LOW	E12
	STUCK LOW/HIGH	E21
OUT	STUCK HIGH	E22
BUS D00/D02/D03	STUCK LOW	E22
BUS -> RXCS	STUCK HIGH	E4
REG SELECT	STUCK LOW	E40
B DONE	STUCK HIGH	E23
-----	STUCK HIGH	E19
	'B2' INPUT STUCK HIGH	E1

;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
;THIS ERROR REPORT WAS GENERATED. THE INFORMATION SHOWN IS
;BASED ON FAULT INSERTION RESULTS, AND SHOULD PROVIDE LOGICAL
;AREAS TO CHECK FOR THE RELEVANT FAULT/S.
;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
;ANALYZE THE FOLLOWING AREA/S:


```

1012 .SBTTL TEST 2 - INTERRUPT TEST PART II / VECTOR ADDRESS VERIFICATION
1013
1014 ; THE PURPOSE OF THIS TEST IS TO VERIFY THAT WRITING THE RXCS INTERRUPT ENABLE BIT
1015 ; (BIT 6) = 1 DOES INDEED WRITE IT TO A 1, THEREFORE BECAUSE DONE = 1
1016 ; AN INTERRUPT SHOULD OCCUR (THE PDP PRIORITY IS 0)
1017
1018 002566 013702 001204 T2: MOV KRXVEC, R2
1019 002572 010246 MOV R2, -(SP) ; SAVE INTERRUPT VECTOR FOR
1020 ; ERROR REPORT
1021 002574 012722 002644 MOV #15, (R2)+ ; RX01 VECTOR ADDRESS
1022 002600 012722 000340 MOV #PR7, (R2)+
1023 002604 012602 MOV (SP)+, R2 ; RESTORE INTERRUPT VECTOR FOR
1024 ; ERROR REPORT
1025 002606 005046 CLR -(SP) ; PDP PRIORITY <ON>
1026 002610 012746 002616 MOV #65, -(SP)
1027 002614 000002 RTI
1028 002616 052777 000100 176362 65: BIS #BIT6, @ RXCS ; SET RX01 INTERRUPT ENABLE
1029 002624 000240 NOP
1030

```

```

;*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:
;*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:
;*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:*/:

```

THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
 THIS ERROR REPORT WAS GENERATED. THE INFORMATION SHOWN IS
 BASED ON FAULT INSERTION RESULTS, AND SHOULD PROVIDE LOGICAL
 AREAS TO CHECK FOR THE RELEVANT FAULT/S.

IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
 ANALYZE THE FOLLOWING AREA/S:

M7846 (UNIBUS INTERFACE)

SIGNAL NAME	REASON	POSSIBLE CHIPS
-----	SACK FLOP LOCKED CLEAR	E13, E31, E36
-----	BBSY FLOP CLOCK LOCKED HIGH	E13, E31, E36, E33
BUS REQUEST	STUCK HIGH	E31
BUS INTR	STUCK HIGH	E31, E35, E8, E38
-----	SACK FLOP CLOCK LOCKED HIGH	E31, E21, E9, E12
-----	SACK FLOP CLOCK LOCKED LOW	E31, E21
-----	BBSY FLOP CLOCK LOCKED LOW	E31, E33, E9
BUS D02/D04/D05	STUCK HIGH	E38
BUS SACK	STUCK HIGH	E32
BG OUT	STUCK HIGH	E32, E25, E28
BUS REQUEST	STUCK LOW	E39
-----	JUMPER (N1) NOT IN	-----
BUS D07	STUCK HIGH	E35, E8
-----	GRANT FLOP CAN'T BE PRESET	E25, E9, E28
B SER DATA	STUCK LOW	E9
-----	GRANT FLOP 'Q' OUTPUT	E28
-----	STUCK HIGH	


```

1174 002712 104406          2$:  SUBSCOPE
1175
1176
1177          ; TEST 2 - CONT'D
1178
1179          ; THE PURPOSE OF THIS TEST IS TO VERIFY THAT BIT 6 OF THE RXCS (INTERRUPT ENABLE)
1180          ; CAN BE CLEARED AFTER IT WAS KNOWN TO BE SET
1181
1182 002714 013702 001204      MOV KRXVEC, R2
1183 002720 010246            MOV     R2, -(SP)          ; SAVE INTERRUPT VECTOR FOR
1184                                     ; ERROR REPORT
1185 002722 012722 003012      MOV #4$, (R2)+          ; RX11 VECTOR ADDRESS
1186 002726 012722 000340      MOV #PR7, (R2)+
1187 002732 012602            MOV     (SP)+, R2          ; RESTORE INTERRUPT VECTOR FOR
1188                                     ; ERROR REPORT
1189 002734 042777 000100 176244 BIC #BIT6, @ RXCS      ; CLEAR THE RX11 INTERRUPT ENABLE BIT
1190
1191          ; THE RXCS SHOULD = 40 (DONE)
1192
1193 002742 012700 000040      MOV #40, R0
1194 002746 020077 176234      CMP R0, @ RXCS
1195 002752 001403            BEQ 3$
1196 002754 017701 176226      MOV @ RXCS, R1
1197
1198          ; (R0) = 40 ; (R1) = ACTUAL RXCS ; (R2) = N/A
1199
1200 002760 104000            ERROR                      ; RXCS NOT = 40
1201 002762 104406          3$:  SUBSCOPE
1202          ; NO RX11 INTERRUPTS SHOULD OCCUR (YET)
1203
1204 002764 005046            CLR -(SP)                  ; PDP PRIORITY <ON>
1205 002766 012746 002774      MOV #10$, -(SP)
1206 002772 000002            RTI
1207 002774 000240          10$:  NOP
1208 002776 000240            NOP
1209 003000 012746 000340      MOV #PR7, -(SP)          ; PDP PRIORITY <OFF> 7
1210 003004 012746 003014      MOV #11$, -(SP)
1211 003010 000002            RTI
1212
1213          ; RETURN TO HERE WITH THE PDP PRIORITY = 7 IF AN UNEXPECTED RX11 INTERRUPT
1214          ; WHILE CLEARING THE RX11 INTERRUPT ENABLE BIT 6
1215
1216          ; (R0) = N/A ; (R1) = N/A ; (R2) = N/A
1217
1218 003012 104000          4$:  ERROR                      ; UNEXPECTED RX11 INTERRUPT
1219 003014 104406          11$:  SUBSCOPE
1220
1221          ; AN RX11 INTERRUPT SHOULD OCCUR (NOW)
1222
1223 003016 013702 001204      MOV KRXVEC, R2
1224 003022 010246            MOV     R2, -(SP)          ; SAVE INTERRUPT VECTOR FOR
1225                                     ; ERROR REPORT
1226 003024 012722 003074      MOV #5$, (R2)+          ; RX11 VECTOR ADDRESS
1227 003030 012722 000340      MOV #PR7, (R2)+

```

H09

MAINDEC-11-DZRXB-D MACY11 27(663) 6-NOV-75 08:50 PAGE 29
 DZRXB0.P11 TEST 2 - INTERRUPT TEST PART II / VECTOR ADDRESS VERIFICATION

SEQ 0110

```

1228 003034 012602      MOV      (SP)+,R2      ;RESTORE INTERRUPT VECTOR FOR
1229                                     ;ERROR REPORT
1230 003036 005046      CLR      -(SP)      ; PDP PRIORITY <ON>
1231 003040 012746 003046  MOV      #12$,-(SP)
1232 003044 000002      RTI
1233 003046 052777 000100 176132 12$:  BIS      #BIT6, 2 RXCS      ; SET RX11 INTERRUPT ENABLE BIT
1234 003054 000240      NOP
1235 003056 000240      NOP
1236 003060 012746 000340  MOV      #PR7,-(SP)
1237 003064 012746 003072  MOV      #13$,-(SP)
1238 003070 000002      RTI
1239
1240                                     ; (R0) = N/A ; (R1) = N/A ; (R2) = N/A
1241
1242 003072 104000      13$:  ERROR      ; NO RX11 INTERRUPT OCCURRED
1243
1244                                     ; RETURN TO HERE WITH THE PDP PRIORITY = 7 IF AN RX01 INTERRUPT
1245
1246 003074 000004      5$:  SCOPE
1247 003076 042777 000100 176102  BIC      #BIT6, 2 RXCS      ; CLEAR THE RX11 INTERRUPT ENABLE
1248
1249 003104 000137 004074  JMP      CEXIT      ;END OF TEST 2

```

.SBTTL TEST 3 - INTERRUPT TEST PART III / PRIORITY LEVEL VERIFICATION PART I

THE PURPOSE OF THIS TEST IS TO VERIFY THE PRIORITY OF THE RX11 INTERRUPT REQUEST LINE
 THE PROGRAM SETS THE PDP PRIORITY TO 1 LESS THAN THE DEVICE LEVEL
 (DEVICE LEVEL SPECIFIED BY CONTENTS OF LOCATION 'BRLEV:' -- NORMALLY 5)
 AN RX01 INTERRUPT SHOULD OCCUR
 IF NO INTERRUPT OCCURS THEN THE PRIORITY LEVEL OF THE RX11 IS [NOT] = NORMAL
 DEVICE LEVEL OF 5 OR THE DEVICE LEVEL AS SPECIFIED BY THE CONTENTS OF
 LOCATION 'BRLEV:' WHICH MAY HAVE BEEN CHANGED BY THE USER BEFORE PROGRAM
 EXECUTION, BUT MAYBE SOME VALUE LESS THAN THE CONTENTS OF LOCATION 'BRLEV:'.
 NOTE: IF THERE IS NO HARDWARE "PSW" THIS TEST WILL BE SKIPPED.

```

1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262 003110 005001      T3:   CLR      R1                ; INDICATOR TO CPU PRIORITY
1263                                     ; ROUTINE TO DROP CPU PRIORITY
1264                                     ; TO 1 LESS THAN DEVICE LEVEL
1265 003112 013702 001204      MOV KRXVEC, R2
1266 003116 010246          MOV      R2, -(SP)          ; SAVE INTERRUPT VECTOR FOR
1267                                     ; ERROR REPORT
1268 003120 012722 003232      MOV #1$, (R2)+          ; RX01 VECTOR ADDRESS
1269 003124 012722 000340      MOV #PR7, (R2)+
1270 003130 012602          MOV      (SP)+, R2          ; RESTORE INTERRUPT VECTOR FOR
1271                                     ; ERROR REPORT
1272 003132 013746 000004      MOV      4, -(SP)          ; SAVE 'BUSERR' TIMEOUT 'PC'
1273 003136 012737 003154 000004      MOV      #2$, 4          ; SET TIMEOUT VECTOR
1274 003144 012737 000200 177776      MOV      #PR4, PSW          ; SET LEVEL TO 4 IF 'PSW' EXISTS
1275 003152 000404          BR      3$                  ; GO TO RESET VECTOR 4 & DO TEST
1276 003154 022626          2$:  CMP      (SP)+, (SP)+      ; CORRECT STACK FROM BUS TIMEOUT
1277 003156 012637 000004      MOV      (SP)+, 4          ; RESTORE TIMEOUT VECTOR TO 'BUSERR'
1278 003162 000427          BR      4$                  ; NO HARDWARE PSW - SKIP THIS TEST
1279 003164 012637 000004      3$:  MOV      (SP)+, 4          ; RESET TIMEOUT VECTOR TO 'BUSERR'
1280 003170 004737 006026      JSR      PC, CPUPRI          ; CALCULATE PRIORITY LEVEL OF CPU
1281                                     ; BASED ON CURRENT DEVICE PRIORITY
1282                                     ; LEVEL RESIDING IN LOC. 'BRLEV'
1283 003174 010046          MOV      R0, -(SP)          ;; PUT NEW PS ON STACK
1284 003176 012746 003204      MOV      #64$, -(SP)          ;; PUT NEW PC ON STACK
1285 003202 000002          RTI                          ;; POP NEW PC AND PS
1286 003204          64$:  BIS #BIT6, 2 RXCS                ; SET THE RX01 INTERRUPT ENABLE
1287 003204 052777 000100 175774      NOP
1288 003212 000240          NOP
1289 003214 000240          NOP
1290 003216 013746 000340      MOV      PR7, -(SP)          ;; PUT NEW PS ON STACK
1291 003222 012746 003230      MOV      #65$, -(SP)          ;; PUT NEW PC ON STACK
1292 003226 000002          RTI                          ;; POP NEW PC AND PS
1293 003230          65$:
1294
1295          ; (R0) = N/A ; (R1) = N/A ; (R2) = N/A
1296
1297 003230 104000          ERROR                ; PRIORITY LEVEL IS NOT = CONTENTS
1298                                     ; OF 'BRLEV:' (NORMALLY 5) BUT
1299                                     ; MAYBE SOME VALUE LESS THAN THE
1300                                     ; THE CURRENT CONTENTS OF 'BRLEV:'
1301
1302          ; RETURN TO HERE WITH THE PDP PRIORITY = 7 IF AN RX01 INTERRUPT
1303

```

```

1304 003232 000004 1S: SCOPE
1305 003234 042777 000100 175744 BIC #BIT6, 2 RXCS ; CLEAR THE RX11 INTERRUPT ENABLE
1306 003242 000137 004074 4S: JMP CEXIT ; END OF TEST 3
1307
1308 .SBTTL TEST 4 - INTERRUPT TEST PART IV / PRIORITY VERIFICATION PART II
1309
1310 ; THE PURPOSE OF THIS TEST IS TO VERIFY THE PRIORITY OF THE RX11 INTERRUPT REQUEST LINE
1311 ; THE PROGRAM SETS THE PDP PRIORITY = THE DEVICE LEVEL, (NORMALLY 5 OR THE CONTENTS OF L
1312 ; NO RX01 INTERRUPTS SHOULD OCCUR
1313 ; IF AN INTERRUPT DOES OCCUR THEN THE PRIORITY LEVEL OF THE RX11 IS (NOT)
1314 ; = THE NORMAL DEVICE LEVEL OF 5, OR WHATEVER IS THE VALUE IN LOCATION 'BRLEV:'
1315 ; BUT MAYBE SOME VALUE GREATER THAN THE CONTENTS OF LOC. 'BRLEV:'
1316 ; NOTE: IF THERE IS NO HARDWARE "PSW" THIS TEST WILL BE SKIPPED.
1317
1318 003246 005001 T4: CLR R1 ; INDICATOR TO CPU PRIORITY ROUTINE
1319 ; TO DROP CPU PRIORITY 1 LEVEL
1320 ; LESS THAN THE DEVICE LEVEL
1321 003250 013702 001204 MOV KRXVEC, R2
1322 003254 010246 MOV R2, -(SP) ; SAVE INTERRUPT VECTOR FOR
1323 ; ERROR REPORT
1324 003256 012722 003374 MOV #1$, (R2)+ ; RX01 VECTOR ADDRESS
1325 003262 012722 000340 MOV #PR7, (R2)+
1326 003266 012602 MOV (SP)+, R2 ; RESTORE INTERRUPT VECTOR FOR
1327 ; ERROR REPORT
1328 003270 052701 000200 BIS #BIT7, R1 ; SET INDICATOR TO CPU PRIORITY
1329 ; ROUTINE TO SET CPU PRIORITY LEVEL
1330 ; TO THE SAME LEVEL AS THE DEVICE
1331 003274 013746 000004 MOV 4, -(SP) ; SAVE 'BUSERR' TIMEOUT 'PC'
1332 003300 012737 003316 000004 MOV #3$, 4 ; SET TIMEOUT VECTOR
1333 003306 012737 000240 177776 MOV #PR5, PSW ; SET LEVEL TO 5 IF 'PSW' EXISTS
1334 003314 000404 BR 4S ; GO ON TO RESET VECTOR & DO TEST
1335 003316 022626 3S: CMP (SP)+, (SP)+ ; CORRECT STACK FROM BUS TIMEOUT
1336 003320 012637 000004 MOV (SP)+, 4 ; RESTORE TIMEOUT VECTOR TO 'BUSERR'
1337 003324 000430 BR 5S ; NO HARDWARE PSW - SKIP THIS TEST
1338 003326 012637 000004 4S: MOV (SP)+, 4 ; RESET TIMEOUT VECTOR TO BUSERR
1339 003332 004737 006026 JSR PC, CPU PRI ; CALCULATE CPU PRIORITY LEVEL TO
1340 ; BE THE SAME AS THE DEVICE LEVEL
1341 ; I.E. - SAME AS CONTENTS OF LOC.
1342 ; 'BRLEV'
1343 003336 010046 MOV R0, -(SP) ; PUT NEW PS ON STACK
1344 003340 012746 003346 MOV #6$, -(SP) ; PUT NEW PC ON STACK
1345 003344 000002 RTI ; POP NEW PC AND PS
1346
1347 003346 052777 000100 175632 64S: BIS #BIT6, 2 RXCS ; SET RX01 INTERRUPT ENABLE
1348 003354 000240 NOP
1349 003356 000240 NOP
1350 003360 013746 000340 MOV PR7, -(SP) ; PUT NEW PS ON STACK
1351 003364 012746 003372 MOV #65$, -(SP) ; PUT NEW PC ON STACK
1352 003370 000002 RTI ; POP NEW PC AND PS
1353
1354 003372 000401 65S: BR 2S
1355
1356 ; RETURN TO HERE WITH THE PDP PRIORITY = 7 IF AN RX01 INTERRUPT
1357

```

K09

MAINDEC-11-DZRXB-D MACY11 27(663) 6-NOV-75 08:50 PAGE 32
DZRXBO.P11 TEST 4 - INTERRUPT TEST PART IV / PRIORITY VERIFICATION PART II

SEQ 0113

```

1358                                     ; (R0) = N/A ; (R1) = N/A ; (R2) = N/A
1359
1360 003374 104000                      1$:      ERROR                      ;PRIORITY LEVEL NOT = TO CONTENTS
1361                                     ;OF LOCATION 'BRLEV;' (NORMALLY 5)
1362                                     ;BUT MAYBE SOME VALUE GREATER THAN
1363                                     ;THE CONTENTS SPECIFIED BY LOC.
1364                                     ;'BRLEV:'
1365 003376 00G004                      2$:      SCOPE
1366 003400 042777 000100 175600        BIC #BIT6, 2 RXCS                ; CLEAR THE RX01 INTERRUPT ENABLE
1367 003406 000137 004074              5$:      JMP CEXIT                ;END OF TEST 4

```

```

1368 .SBTTL TEST 5 - INIT [PROGRAMMED] / RST
1369
1370 ; THE PURPOSE OF THIS TEST IS TO VERIFY THAT SETTING THE RXCS BIT 14
1371 ; CAUSES AN RXD1 PROGRAMMED SUBSYSTEM INITIALIZE
1372
1373 ; THE RXCS SHOULD = 40 (DONE)
1374
1375 ; THE RXDB SHOULD = 4, OR 104, OR 204, OR 304
1376
1377 003412 052777 040000 175566 T5: BIS #BIT14, 2 RXCS ; RXD1 PROGRAMMED INITIALIZE
1378 003420 004737 006412 15: JSR PC, SDN ; WAIT FOR THE DONE BIT
1379 003424 000775 BR 15
1380 003426 012700 000040 MOV #40, R0 ; PROGRAM EXPECTS RXCS = 40 (DONE)
1381 003432 017701 175550 MOV 2 RXCS, R1 ; ACTUAL RXCS
1382 003436 020100 CMP R1, R0
1383 003440 001401 BEQ 25
1384
1385 ; (R0) = 40 ; (R1) = ACTUAL RXCS ; (R2) = N/A
1386
1387 003442 104000 ERROR ; RXCS NOT = 40
1388

```

```

; /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*:
; /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*:
; /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*:

```

; IF THE FAULT CANNOT BE FOUND ON THE UNIBUS INTERFACE MODULE
; AND/OR THE FAULT IS NOT INHERENT TO THE UNIBUS INTERFACE MODULE
; M7846 THERE IS A POSSIBILITY OF ITS EXISTENCE ON THE READ/WRITE
; MODULE M7727.

NOTE: ONLY APPROX. 30% OF THIS MODULE LENT ITSELF CONDUCTIVE TO
THE FAULT INSERTION PROCESS; ERGO, THE RESOLUTION FOR FAULT
ANALYSIS OBTAINABLE BY THIS MODULE IS NOT VERY HIGH.
HOWEVER, ANALYSIS OF THE FOLLOWING AREA/S, IF THIS ERROR
REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS, SHOULD AT
LEAST PLACE YOU WITHIN THE RELEVANT AREA ON THE MODULE.

M7727 (READ/WRITE CONTROL)

SIGNAL NAME	REASON	POSSIBLE CHIPS
-----	'J' INPUT LOCKED LOW	E16

```

; /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*:
; /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*:
; /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*: /*:

```

```

1416 003444 104406 25: SUBSCOPE
1417
1418 ; //////////////////////////////////////
1419 ;
1420 ; 7 6 - - 3 2 1 0 /
1421 ;

```


N09

MAINDEC-11-DZRXB-D MACY11 27(663) 6-NOV-75 08:50 PAGE 35
DZRXB0.P11 TEST 5 - INIT (PROGRAMMED) / RST

SEQ 0116

[illegible]

1478 003510 104406

45: SUBSCOPE


```

1533      ; THE RXDB SHOULD = 200 (IF DRIVE 0 IS READY), OR 0 IF UNIT 0 IS NOT SELECTED
1534      ; MAYBE 300 (IF DRIVE 0 IS READY AND SECTOR 1 WAS WRITTEN WITH DELETED DATA)
1535
1536      003570 017702 175414      MOV 3 RXDB, R2      ; ACTUAL RXDB
1537      003574 010201      MOV R2, R1
1538      003576 042701 000100      BIC #BIT6, R1      ; CLEAR N/A DELETED DATA BIT
1539      003602 012700 000200      MOV #200, R0      ; EXPECT UNIT 0 RDY SET
1540      003606 105737 012446      TSTB UNITSEL
1541      003612 100403      BMI 11$
1542      003614 042701 000200      BIC #BIT7, R1      ; UNIT 0 NOT SELECTED
1543      003620 005000      CLR R0      ; DISREGARD RDY BITS
1544      003622 020100      11$: CMP R1, R0
1545      003624 001401      BEQ 12$
1546
1547      ; (R0) = 0 OR 200
1548      ; (R1) = ACTUAL RXDB MINUS DELETED DATA BIT#6
1549      ; (R2) = ACTUAL RXDB
1550
1551      003626 104000      12$: ERROR      ; RXDB NOT = 200, OR NOT = 0
1552      003630 000004      SCOPE
1553      003632 000137 004074      JMP CEXIT      ; END OF TEST 5

```

•

```

1652 003762 000004      25:  SCOPE
1653
1654 003764 000207      RTS PC                      : EXIT SUBROUTINE FBEB

```

;
;
;

SIGNAL NAME	REASON	POSSIBLE CHIPS
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		
27		
28		
29		
30		
31		
32		
33		
34		
35		
36		
37		
38		
39		
40		
41		
42		
43		
44		
45		
46		
47		
48		
49		
50		
51		
52		
53		
54		
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		

RX RUN	STUCK LOW	E24
LOAD	STUCK LOW	E18
-----	BINARY COUNTER 'RESET'	E34
	LOCKED HIGH	
RX DATA	STUCK HIGH	E3
BUS 004/005	STUCK HIGH/LOW	E1,E4
BUS 002	STUCK HIGH	E7

IF THE FAULT CANNOT BE FOUND ON THE UNIBUS INTERFACE MODULE
 AND/OR THE FAULT IS NOT INHERENT TO THE UNIBUS INTERFACE MODULE
 M7846 THERE IS A POSSIBILITY OF ITS EXISTENCE ON THE READ/WRITE
 MODULE M7727.

NOTE: ONLY APPROX. 30% OF THIS MODULE LENT ITSELF CONDUCTIVE TO
 THE FAULT INSERTION PROCESS; ERGO, THE RESOLUTION FOR FAULT
 ANALYSIS OBTAINABLE BY THIS MODULE IS NOT VERY HIGH.
 HOWEVER, ANALYSIS OF THE FOLLOWING AREA/S, IF THIS ERROR
 REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS, SHOULD AT
 LEAST PLACE YOU WITHIN THE RELEVANT AREA ON THE MODULE.

M7727 (READ/WRITE CONTROL)

SIGNAL NAME	REASON	POSSIBLE CHIPS
-----	REPLACE E5	-----
:/\/		
:/\/		
:/\/		

1741	004040	000004	15:	SCOPE
1742	004042	000764		BR 25
1743				
1744				.SBTTL TEST 12 - FILL/EMPTY BUFFER ALL 1'S
1745				
1746				; PROGRAMMING NOTE: THE FOLLOWING " MOV " INSTEAD OF " INC " FOR LOOPS
1747				
1748	004044	012737	000001	012046 T12: MOV #1, PAT
1749	004052	000402		BR .+6
1750				
1751				.SBTTL TEST 11 - FILL/EMPTY BUFFER ALL 0'S
1752				
1753	004054	005037	012046	T11: CLR PAT
1754	004060	004737	003654	JSR PC, T6FILL
1755	004064	004737	004000	JSR PC, T7EMPTY
1756	004070	000137	004074	JMP CEXIT

;END OF TEST 11 OR 12

MAINDEC-11-DZRXB-D MACY11 27(663) 6-NOV-75 08:50 PAGE 42
DZRXBD.P11 TEST 11 - FILL/EMPTY BUFFER ALL 0'S

MAINDEC-11-DZRXB-D
DZRXBD.P11 TE

TEST 11 - FILL/EMPTY BUFFER ALL 0'S

;
;
;

```

;IF THE FAULT CANNOT BE FOUND ON THE UNIBUS INTERFACE MODULE
;AND/OR THE FAULT IS NOT INHERENT TO THE UNIBUS INTERFACE MODULE

```

NOTE: ONLY APPROX. 30% OF THIS MODULE LENT ITSELF CONDUCTIVE TO THE FAULT INSERTION PROCESS; ERGO, THE RESOLUTION FOR FAULT ANALYSIS OBTAINABLE BY THIS MODULE IS NOT VERY HIGH. HOWEVER, ANALYSIS OF THE FOLLOWING AREA/S, IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS, SHOULD AT LEAST PLACE YOU WITHIN THE RELEVANT AREA ON THE MODULE.

SIGNAL NAME	REASON	POSSIBLE CHIPS
SEL DK1	STUCK LOW	E13,E14
DK1 INDX	STUCK HIGH/LOW	E15
SEL INDX	STUCK HIGH/LOW	E15
-----	'A2' INPUT STUCK LOW	E15

[illegible]

1834 004236 000004
1835 004240 000137 004106

```

35:      SCOPE
        JMP DEXIT

```

;END OF TEST 13

:TI4 CONT. - THIS SECTION TESTS THAT NO PARITY OR CRC ERROR OCCURRED

THE FAULT INSERTION PROCESS; ERGO, THE RESOLUTION FOR FAULT ANALYSIS OBTAINABLE BY THIS MODULE IS NOT VERY HIGH. HOWEVER, ANALYSIS OF THE FOLLOWING AREA/S, IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS, SHOULD AT LEAST PLACE YOU WITHIN THE RELEVANT AREA ON THE MODULE.

M7727 (READ/WRITE CONTROL)

SIGNAL NAME	REASON	POSSIBLE CHIPS
INIT	STUCK HIGH	E16
SEL TRK 0	STUCK HIGH	E15
DK1 TRK 0	STUCK HIGH	E15
SEL DK0	STUCK LOW	E15, E14, E13
WT GATE	STUCK HIGH	E5
SEL WT PROT	STUCK LOW	E5
RAW DATA	STUCK HIGH/LOW	E14
STEP	STUCK LOW	E14
-----	REPLACE E4	-----

```

/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/
/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/
/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/

```

1968	004430	000004		65:	SCOPE	
1969	004432	000137	004106		JMP DEXIT	;END OF TEST 14
1970						

```

1971                                     .SBTTL TEST 15 - ERROR FLAG AND B-CODE VERIFICATION PART II
1972
1973                                     ; THIS TEST VERIFIES THAT TRYING TO WRITE, USING DELETED DATA MODE, ON A
1974                                     ; NON-EXISTANT SECTOR WILL PRODUCE AN ERROR AND THE CORRECT B-CODE IS PRODUCED
1975                                     ; THIS SECTION SENDS OUT AN ILLEGAL SECTOR ADDRESS AND EXPECTS AN ERROR
1976                                     ; NOTE TEST 14 MUST BE RUN BEFORE THIS TEST
1977
1978 004436 005000      T15:  CLR R0
1979 004440 005002      CLR R2
1980 004442 105737 012446      TSTB UNITSEL                      ; WAS UNIT 0 SELECTED
1981 004446 100004      BPL 10$
1982 004450 012777 000015 174530      MOV #15, 2RXCS              ; SET WRITE DELETED DATA FUNCTION
1983 004456 000403      BR 11$
1984 004460 012777 000035 174520 10$:  MOV #35, 2RXCS              ; SEND WTR DD FUNCTION TO UNIT 1
1985 004466 004737 004734 11$:  JSR PC, ILLADR                  ; SEND THE ILLEGAL SECTOR ADDRESS
1986 004472 000406      BR 1$                                       ; PREMATURE ERROR CONDITION
1987 004474 012700 100040      MOV #100040, R0                      ; EXPECT ERROR AND DONE FLAGS
1988 004500 017701 174502      MOV 2RXCS, R1
1989 004504 020001      CMP R0, R1
1990 004506 001401      BEQ 2$
1991
1992                                     ; R0 = 100040 ; R1 = ACTUAL RXCS ; R2 = # OF TR FLAGS
1993
1994 004510 104000      1$:  ERROR                      ; RXCS NOT = 100040
1995 004512 104406      2$:  SUBSCOPE
1996
1997                                     ; T15 CONT. - THIS SECTION TESTS THAT THERE IS NO PARITY, CRC ERROR
1998                                     ; AND THAT THE DELETED DATA BIT IS SET.
1999
2000 004514 005002      CLR R2
2001 004516 012700 000100      MOV #100, R0                      ; EXPECT DELETED DATA BIT TO BE SET
2002 004522 017701 174462      MOV 2RXDB, R1
2003 004526 020001      CMP R0, R1
2004 004530 001401      BEQ 3$
2005
2006                                     ; R0 = 100 ; R1 = ACTUAL RXDB ; R2 = N/A
2007 004532 104000      3$:  ERROR                      ; DELETED DATA NOT SET OR OTHER ERRORS
2008 004534 104406      SUBSCOPE
2009
2010                                     ; T15 CONT. - THIS SECTION TESTS FOR THE B-CODE FOR ILLEGAL SECTOR.
2011
2012 004536 012777 000017 174442      MOV #17, 2RXCS              ; SET READ STATUS B FUNCTION
2013 004544 004737 006412      4$:  JSR PC, SON                  ; WAIT FOR DONE FLAG
2014 004550 000775      BR 4$
2015 004552 012700 000070      MOV #70, R0
2016 004556 017701 174426      MOV 2RXDB, R1
2017 004562 020001      CMP R0, R1
2018 004564 001401      BEQ 5$
2019
2020                                     ; R0 = 70 ; R1 = ACTUAL B-CODE ; R2 = N/A
2021 004566 104000      5$:  ERROR                      ; RXDB NOT = 70
2022 004570 000004      SCOPE
2023 004572 000137 004106      JMP DEXIT                      ; END OF TEST 15

```

```

2024 .SBTTL TEST 16 - ERROR FLAG AND B-CODE VERIFICATION PART III
2025
2026 ;THIS TEST VERIFIES THAT A WRITE FUNCTION TO A NON-EXISTANT SECTOR WILL
2027 ;PRODUCE AN ERROR AND A B-CODE OF 70. THE DELETED DATA BIT SHOULD ALSO BE CLEARED
2028 ;THIS SECTION TRANSFERS AN ILLEGAL SECTOR ADDRESS FOR A WRITE FUNCTION
2029 ; NOTE TEST 14 MUST BE RUN BEFORE THIS TEST
2030
2031 004576 005000 T16: CLR R0
2032 004600 005002 CLR R2
2033 004602 105737 012446 TSTB UNITSEL ;WAS UNIT 0 SELECTED
2034 004606 100004 BPL 10$
2035 004610 012777 000005 174370 MOV #5,ARXCS ;SET THE WRITE FUNCTION
2036 004616 000403 BR 11$
2037 004620 012777 000025 174360 10$: MOV #25,ARXCS ;SEND WRITE FUNCTION TO UNIT 1
2038 004626 004737 004734 11$: JSR PC,ILLADR ;SEND THE ILLEGAL ADDRESS
2039 004632 000406 BR 1$ ;PREMATURE ERROR CONDITION
2040 004634 012700 100040 MOV #100040,R0 ;EXPECT ERROR AND DONE BITS SET
2041 004640 017701 174342 MOV ARXCS,R1
2042 004644 020001 CMP R0,R1
2043 004646 001401 BEQ 2$
2044
2045 ; R0 = 100040 ; R1 = ACTUAL RXCS ; R2 = # OF TR FLAGS
2046
2047 004650 104000 1$: ERROR
2048 004652 104406 2$: SUBSCOPE
2049
2050 ;T16 CONT. - TESTS FOR NO PARITY, CRC ERRORS, AND NO DELETED DATA BIT
2051
2052 004654 005002 CLR R2
2053 004656 005000 CLR R0 ;NO BITS SHOULD BE SET IN THE RXDB
2054 004660 017701 174324 MOV ARXDB,R1
2055 004664 020001 CMP R0,R1
2056 004666 001401 BEQ 3$
2057
2058 ; R0 = 0 ; R1 = ACTUAL RXDB ; R2 = N/A
2059 004670 104000 ERROR ;SOME BIT IS SET IN THE RXDB
2060 004672 104406 3$: SUBSCOPE
2061
2062 ;T16 CONT. - TEST FOR CORRECT B-CODE FOR ILLEGAL SECTOR ADDRESS
2063
2064 004674 012777 000017 174304 4$: MOV #17,ARXCS ;SET READ STATUS B FUNCTION
2065 004702 004737 006412 JSR PC,SDN ;WAIT FOR DONE FLAG
2066 004706 000775 BR 4$
2067 004710 012700 000070 MOV #70,R0
2068 004714 017701 174270 MOV ARXDB,R1
2069 004720 020001 CMP R0,R1 ;IS B-CODE = 70
2070 004722 001401 BEQ 5$ ;YES, CONTINUE
2071
2072 ; R0 = 70 ; R1 = ACTUAL RXDB ; R2 = N/A
2073 004724 104000 ERROR
2074 004726 000004 5$: SCOPE
2075 004730 000137 004106 JMP DEXIT ;END OF TEST 16

```

2076					
2077					
2078					;GENERATE AN ILLEGAL SECTOR ADDRESS
2079	004734	004737	006376	ILLADR:	JSR PC,STR
2080	004740	000402			BR 2\$
2081	004742	005202			INC R2
2082	004744	000404			BR 3\$
2083	004746	004737	006412	2\$:	JSR PC,SDN
2084	004752	000770			BR ILLADR
2085	004754	000430			BR 1\$
2086	004756	005077	174226	3\$:	CLR JRXDB
2087	004762	004737	006376	7\$:	JSR PC,STR
2088	004766	000402			BR 5\$
2089	004770	005202			INC R2
2090	004772	000404			BR 6\$
2091	004774	004737	006412	5\$:	JSR PC,SDN
2092	005000	000770			BR 7\$
2093	005002	000415			BR 1\$
2094	005004	005077	174200	6\$:	CLR JRXDB
2095	005010	004737	006376	11\$:	JSR PC,STR
2096	005014	000402			BR 10\$
2097	005016	005202			INC R2
2098	005020	000406			BR 1\$
2099					
2100	005022	004737	006412	10\$:	JSR PC,SDN
2101	005026	000770			BR 11\$
2102	005030	062716	000002		ADD #2,2SP
2103	005034	000207		4\$:	RTS PC
2104	005036	017701	174144	1\$:	MOV JRXCS,R1
2105	005042	000774			BR 4\$

```

;LOOK FOR A TR FLAG
;NO TR FLAG, IS DONE SET
;TR FLAG COUNTER

;LOOK FOR DONE FLAG
;NOT DONE RECHECK TR FLAG
;DONE IS SET TOO EARLY GO TO ERROR
;0 SECTOR ADDRESS (ILLEGAL)
;LOOK FOR SECOND TR FLAG
;NOT TR, IS IT DONE

;TR FLAG SEND TRACK ADDRESS
;LOOK FOR DONE FLAG
;NOT DONE, RECHECK TR FLAG
;DONE TOO SOON GO TO ERROR
;SEND TRACK ADDRESS OF 0
;ARE THERE ANY MORE TR FLAGS
;NO, LOOK FOR DONE
;YES
;TOO MANY TR FLAGS OR MICROCONTROLLER
;DID NOT DETECT THE ERROR
;LOOK FOR DONE FLAG
;NOT DONE RETEST TR FLAG

```

!M7846 (UNIBUS INTERFACE)

	SIGNAL NAME	REASON	POSSIBLE CHIPS
	-----	INT ENB FLOP CLOCK LOCKED HIGH	E40
:	:~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~:::		
:	:~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~:::		
:	:~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~::~~:::		
2169	005176	104406	11\$: SUBSCOPE
2170			
2171			;T17 CONT. - TEST THAT THERE WERE NO OTHER ERRORS THAN THE ILLEGAL TRACK ERROR EXPECTED
2172			;AND THAT THE DELETED DATA BIT WAS CLEARED BY TEST 16.
2173			
2174	005200	005000	CLR R0 ;DD BIT CLEARED IN TEST 16 SO RXDB = 0
2175	005202	005002	CLR R2
2176	005204	017701	MOV @RXDB,R1
2177	005210	020001	CMP R0,R1
2178	005212	001401	BEG 12\$
2179			
2180			; R0 = 0 ; R1 = ACTUAL RXDB ; R2 = N/A
2181			
2182	005214	104000	ERROR
2183	005216	104406	12\$: SUBSCOPE
2184			
2185			;T17 CONT. - VERIFIES THAT READ STATUS B HAS NO ERRORS
2186			
2187	005220	012777	MOV #17,@RXCS ;SET READ STATUS B FUNCTION
2188	005226	004737	JSR PC,\$DN ;WAIT FOR DONE FLAG
2189	005232	000775	BR 13\$
2190	005234	005777	TST @RXCS ;WAS THERE AN ERROR ON THIS FUNCTION
2191	005240	100007	BPL 14\$;NO
2192	005242	012700	MOV #40,R0
2193	005246	017701	MOV @RXCS,R1
2194	005252	017702	MOV @RXDB,R2
2195			
2196			; R0 = 40 (DONE) ; R1 = ACTUAL RXCS ; R2 = ACTUAL RXDB
2197			
2198	005256	104000	ERROR ;THERE WAS AN ERROR NO READ STATUS B
2199	005260	104406	14\$: SUBSCOPE
2200	005262	005002	CLR R2
2201	005264	012700	MOV #40,R0 ;B-CODE FOR ILLEGAL TRACK ADDRESS
2202	005270	017701	MOV @RXDB,R1
2203	005274	020001	CMP R0,R1
2204	005276	001401	BEG 15\$;B-CODE IS CORRECT
2205			
2206			; R0 = 40 ; R1 = ACTUAL B-CODE ; R2 = N/A
2207			
2208	005300	104000	ERROR
2209	005302	000004	15\$: SCOPE
2210	005304	000137	JMP DEXIT ;END OF TEST 17

ADDRESS	HEX	ASCII	OPERATION	COMMENT
2255	005334	012737	000001	012650 T21: MOV #1,TARGET
2256	005342	012737	000001	013140 MOV #1,TSECTOR
2257	005350	004737	012356	JSR PC,GETUNIT
2258	005354	012737	000001	012046 MOV #1,PAT
2259	005362	004737	012002	JSR PC,GETPATTERN
2260	005366	004737	010532	JSR PC,ADJSUM ;SET CHECK SUM VALUES
2261	005372	005002		CLR R2
2262	005374	012777	000001	173604 MOV #1,IRXCS ;SET FILL BUFFER FUNCTION
2263	005402	004737	003712	18: JSR PC,FBEB
2264	005406	000404		BR 25

F11

MAINDEC-11-DZRXB-0 MACY11 27(663) 6-NOV-75 08:50 PAGE 53
DZRXBD.P11 TEST 21 - WRITE TEST

SEQ 0134

2265	005410	112077	173574		MOVB (R0)+, DZRXDB
2266	005414	005202			INC R2
2267	005416	000771			BR 15
2268	005420	012737	000005	007472	25: MOV #5, FUNCTION
2269	005426	004737	005436		JSR PC, T21X
2270	005432	000137	004106		JMP DEXIT

;SET WRITE FUNCTION
;GO ISSUE THE COMMAND
;END OF TEST 21


```

2319 .SBTTL TEST 22 - INITIALIZE IMPLIED READ
2320 ; AFTER PREVIOUSLY WRITING A PATTERN ON SECTOR 1 TRACK 1, THIS TEST
2321 ; CHANGES THE CONTENTS OF THE SECTOR BUFFER AND DOES A PROGRAMMED INITIALIZE.
2322 ; AFTER WHICH THE SECTOR BUFFER MUST AGAIN CONTAIN THE DATA PREVIOUSLY
2323 ; WRITTEN ON THAT SECTOR AND TRACK.
2324 ; NOTE: THIS TEST WILL ONLY WORK IF UNIT 0 IS SELECTED AND ON LINE.
2325
2326
2327 005520 105737 012446 T22: TSTB UNITSEL ; IF UNIT 0 IS NOT SELECTED SKIP THIS TEST
2328 005524 100022 BPL 2S
2329 005526 005037 012046 CLR PAT
2330 005532 004737 003654 JSR PC,T6FILL ; LOAD THE SECTOR BUFFER WITH 0
2331 005536 005237 012046 INC PAT ; RELOAD CORE BUFFER WITH 1'S
2332 005542 004737 012002 JSR PC,GETPATTERN
2333 005546 004737 010532 JSR PC,ADJSUM
2334 005552 052777 040001 173426 BIS #RECAL,DRXCS ; SET THE INIT. BIT
2335 005560 004737 006412 1S: JSR PC,SDN
2336 005564 000775 BR 1S
2337 005566 004737 004000 JSR PC,T7EMPTY ; EMPTY THE SECTOR BUFFER AND CHECK IT.
2338 005572 000137 004106 2S: JMP DEXIT ; END OF TEST 22
2339
2340

```

.SBTTL TEST 23 - READ TEST

```

2341 ; THIS TEST VERIFIES THAT A READ FUNCTION DOES IN FACT LOAD THE SECTOR
2342 ; BUFFER WITH DATA READ FROM THE SELECTED ADDRESS.
2343
2344

```

```

2345
2346 005576 005037 012046 T23: CLR PAT
2347 005602 004737 003654 JSR PC,T6FILL ; LOAD SECTOR BUFFER WITH 0'S
2348 005606 005237 012046 INC PAT
2349 005612 004737 012356 JSR PC,GETUNIT
2350 005616 004737 012002 JSR PC,GETPATTERN ; RELOAD CORE BUFFER WITH 1'S
2351 005622 004737 010532 JSR PC,ADJSUM ; SET UP FOR CHECK SUM
2352 005626 012737 000007 007472 MOV #7,FUNCTION ; SET READ FUNCTION AND GO
2353 005634 004737 005436 JSR PC,T21X ; ISSUE COMMAND, WAIT FOR DONE, & TEST DATA
2354 005640 000137 004106 JMP DEXIT ; END OF TEST 23

```

2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393

.SBTTL TEST 24 - DATA TRANSFER AND VERIFICATION

;THE PURPOSE OF THIS TEST IS TO WRITE THEN READ AND VERIFY DATA
;ON ALL SECTORS OF THE SELECTED TRACKS. THE TEST ALTERNATES BETWEEN
;DRIVES ON THE SELECTED TRACKS. DATA PATTERN IS A FLOATING 0.

005644	012737	000002	012046	T24:	MOV #2,PAT	;SET DATA PATTERN TO FLOATING 0
005652	013702	001204		T24X:	MOV KRXVEC,R2	;SET INTERRUPT ADDRESSES
005656	012722	011256			MOV #INTSERV,(R2)+	
005662	012712	000340			MOV #PR7,(R2)	
005666	004737	006614			JSR PC,DRVSWP	;GO TRANSFER THE DATA
005672	000137	004106			JMP DEXIT	;END OF TEST 24 OR 25

.SBTTL TEST 25 - DATA TRANSFER AND VERIFICATION VIA DELETED DATA MODE

;THIS TEST TRANSFERES DATA JUST LIKE TEST 24 EXCEPT IT USES THE
;DELETED DATA FORMAT AND A DATA PATTERN OF FLOATING 1.

005676	012737	000003	012046	T25:	MOV #3,PAT	;SET DATA PATTERN TO FLOATING 1
005704	000137	005652			JMP T24X	;GO TRANSFER THE DATA

.SBTTL TEST 26 - HEAD "HOME" TEST

;THIS TEST MOVES THE HEAD OUT TO TRACK 12 (OCTAL) AND THEN WRITES/READ CHECKS
;ALL SECTORS (RANDOM DATA) ON EACH TRACK. THE TRACK SEQUENCE
;IS DECREMENT;ED BACK TO TRACK 0 (HOME). AFTER COMPLETING
;DRIVE 0 IT SWITCHES OVER TO DRIVE 1 DOING THE SAME TEST.

005710	052737	000200	012660	T26:	BIS #BIT7,SEQUEN	;SPECIAL DECREMENT SEQUENCE
005716	012737	000007	012046		MOV #7,PAT	;SELECT RANDOM DATA
005724	013702	001204			MOV KRXVEC,R2	
005730	012722	011256			MOV #INTSERV,(R2)+	
005734	012712	000340			MOV #PR7,(R2)	
005740	004737	006670			JSR PC,WTROCK	
005744	042737	000200	012660		BIC #BIT7,SEQUEN	
005752	000137	004106			JMP DEXIT	;END OF TEST 26

```

2394
2395
2396
2397
2398 005756 104400 015633 BUSERR: TYPE, LOC4M
2399
2400 005762 012646 MOV (SP)+, -(SP)
2401
2402 005764 104402 TYPOS
2403 005766 006 .BYTE 6
2404 005767 000 .BYTE 0
2405 005770 104400 015746 TYPE, PCM
2406 005774 012716 002276 MOV #REBEGIN, (SP)
2407
2408 006000 000002 RTI
2409
2410
2411
2412
2413
2414
2415 006002 104400 015700 RESERR: TYPE, LOC10M
2416
2417 006006 012646 MOV (SP)+, -(SP)
2418
2419 006010 104402 TYPOS
2420 006012 006 .BYTE 6
2421 006013 000 .BYTE 0
2422 006014 104400 015746 TYPE, PCM
2423 006020 012716 002276 MOV #REBEGIN, (SP)
2424
2425 006024 000002 RTI
2426
2427
2428
2429
2430 006026 013700 001214 CPUPRI: MOV BRLEV, R0
2431
2432 006032 105701 TSTB R1
2433
2434 006034 100401 BMI 15
2435 006036 005300 DEC R0
2436
2437 006040 006300 15: ASL R0
2438 006042 006300 ASL R0
2439 006044 006300 ASL R0
2440 006046 006300 ASL R0
2441 006050 006300 ASL R0
2442 006052 042700 000037 BIC #37, R0
2443
2444 006056 000207 RTS PC
2445

```

;THE FOLLOWING SECTION OF CODE WILL ALLOW PROVIDING INFORMATION
 ;TO THE USER WHEN AN 'UNEXPECTED' BUS TIMEOUT TO LOCATION 4 OCCURS
 ;TYPE MESSAGE INDICATING AN
 ;UNEXPECTED BUS TIMEOUT OCCURRED
 ;SAVE (SP)+ FOR TYPEOUT
 ;SETUP TO TYPE PC WHERE TIMEOUT OCCURRED
 ;GO TYPE--OCTAL ASCII
 ;TYPE 6 DIGITS
 ;SUPPRESS LEADING ZEROS
 ;TYPE MESSAGE '=PC'
 ;SET RETURN 'PC' TO START THE
 ;PROGRAM OVER AGAIN
 ;RETURN TO BEGINNING OF PROGRAM

;THE FOLLOWING SECTION OF CODE WILL ALLOW PROVIDING INFORMATION
 ;TO THE USER WHEN AN 'UNEXPECTED' RESERVED INSTRUCTION TRAP TO LOCATION
 ;10 OCCURS
 ;TYPE MESSAGE INDICATING AN
 ;UNEXPECTED RESERVED INSTRUCTION
 ;TRAP OCCURRED
 ;SAVE (SP)+ FOR TYPEOUT
 ;SETUP TO TYPE PC WHERE RESERVED TRAP OCCURRED
 ;GO TYPE--OCTAL ASCII
 ;TYPE 6 DIGITS
 ;SUPPRESS LEADING ZEROS
 ;TYPE MESSAGE '=PC'
 ;SET RETURN 'PC' TO START THE
 ;PROGRAM OVER AGAIN
 ;RETURN TO BEGINNING OF PROGRAM

;THIS ROUTINE WILL CALCULATE THE PRIORITY LEVEL FOR THE PROCESSOR
 ;BASED ON THE CURRENT PRIORITY LEVEL OF THE DEVICE (CONTENTS OF 'BRLEV:')
 ;GET THE PROPOSED RX11 DEVICE
 ;INTERRUPT PRIORITY LEVEL VALUE
 ;IS CPU LEVEL TO BE THE SAME AS
 ;THE DEVICE LEVEL OR 1 LESS?
 ;BRANCH IF SAME AS!
 ;DROP DEVICE LEVEL PRIORITY
 ;BY 1 LEVEL FOR PSW
 ;FORM BITS <7-5> FOR PSW
 ;ENSURE THAT T, N, Z, V, & C BITS
 ;FOR THE PROCESSOR ARE CLEAR
 ;RETURN TO MAINLINE CODE

.SBTTL " ERROR " TRAP SERVICE ROUTINE

;*****
;*****

; " ERROR "

;*****
;*****XERROR: MOV @ SP, EPCSCOPE ; RETURN ADDRESS FROM " ERROR"
ADD #2, EPCSCOPE ; NOW (EPCSCOPE) = SUBSCOPE+2, OR SCOPE+2
INCERRORS: INC ERRORS
BEQ INCERRORS

; DATA SW 13 = 0 TO PRINT APPROPRIATE ERROR MESSAGE

BIT #SW13, @SWR
BNE NOPRINT
CLR CCOUNT
BIT #BIT8, UNITSEL ; WAS PREVIOUS ERROR REPORTED ON THIS PASS
BNE 1\$
TYPE, MXEHEADER1\$: TYPE, MCRLF
MOV @ SP, R4 ; ERADR
SUB #2, R4
MOV R4, -(SP)
TYPON
.BYTE 6
.BYTE 1
TYPE, SPACE
MOV FAST, -(SP) ; FAST (FIRST ADDRESS OF SELECTED TEST)
TYPON
TYPE, SPACE
MOV PCSCOPE, -(SP) ; FAPT (FIRST ADDRESS OF PRESENT TEST)
TYPON
TYPE, SPACE
MOV R2, -(SP) ; BLANK
TYPON
TYPE, SPACE
MOV R0, -(SP) ; EXPECTED (GOOD) RESULT OF TEST
TYPON
TYPE, SPACE
MOV R1, -(SP) ; ACTUAL (BAD) RESULT OF TEST
TYPON2\$: TYPE, SPACE
MOV PASS, 2\$
JSR R5, SGLDEC
OPEN2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456 006060 011637 006314
2457 006064 062737 000002 006314
2458 006072 005237 006312
2459 006076 001775
2460
2461
2462
2463 006100 032777 020000 173110
2464 006106 001056
2465 006110 005037 002336
2466 006114 032737 000400 012446
2467 006122 001002
2468 006124 104400 014546
2469
2470 006130 104400 015030
2471 006134 011604
2472 006136 162704 000002
2473 006142 010446
2474 006144 104402
2475 006146 006
2476 006147 001
2477 006150 104400 015542
2478 006154 013746 002342
2479 006160 104403
2480 006162 104400 015542
2481 006166 013746 006374
2482 006172 104403
2483 006174 104400 015542
2484 006200 010246
2485 006202 104403
2486 006204 104400 015542
2487 006210 010046
2488 006212 104403
2489 006214 104400 015542
2490 006220 010146
2491 006222 104403
2492 006224 104400 015542
2493 006230 013737 002340 006242
2494 006236 004537 014504
2495 006242 000000

L11

MAINDEC-11-DZRXB-D MACY11 27(663) 6-NOV-75 08:50 PAGE 59
DZRXBD.P11 " ERROR " TRAP SERVICE ROUTINE

SEQ 0140

```
2496 ; DATA SW 0 = 0 TO RING BELL AT ERROR
2497
2498 006244 052737 000400 012446 NOPRINT: BIS #BIT8,UNITSEL ;SET HARD ERROR FLAG
2499 006252 004737 006272 JSR PC,DING
2500
2501 ; DATA SW 15 = 1 TO HALT AT ERROR
2502
2503 006256 032777 100000 172732 1$: BIT #SW15,2SWR
2504 006264 001401 BEQ 2$
2505 006266 000000 HALT
2506 006270 000002 2$: RTI
2507
2508 006272 032777 000001 172716 DING: BIT #SW0,2SWR
2509 006300 001002 BNE 1$
2510 006302 104400 006310 TYPE MABELL
2511 006306 000207 1$: RTS PC
2512
2513 006310 000007 MABELL: .ASCIZ <07> ; DING - A - LING
2514 .EVEN
2515
2516 006312 000000 ERRORS: 0
2517 006314 000000 EPCSCOPE: 0
```

```

2518 .SBTTL " SCOPE " TRAP SERVICE ROUTINE
2519
2520 ; " SCOPE "
2521
2522 006316 005737 006312 XSCOPE: TST ERRORS
2523 006322 001015 BNE SCOPING
2524
2525 ; NO ERRORS HAVE BEEN DETECTED
2526
2527 ; JUST SET (PCSCOPE) = FIRST ADDRESS OF THE SCOPE LOOP
2528
2529 ; (IN CASE ERRORS ARE DETECTED LATER)
2530
2531 006324 005037 006312 NOSCOPE: CLR ERRORS
2532 006330 011637 006374 MOV @ SP, PCSCOPE
2533 006334 000002 RTI
2534
2535 ; " SUBSCOPE "
2536
2537 006336 005737 006312 XSUBSCOPE: TST ERRORS
2538 006342 001001 BNE IS
2539 006344 000002 RTI ; NO ERRORS EXIST
2540
2541 ; ERRORS DO EXIST
2542
2543 ; IF THIS ERROR ADDRESS IS THE SAME ADDRESS WITHIN PROGRAM LOCATION " EPCSCOPE"
2544
2545 ; THEN THIS IS A SCOPING LOOP
2546
2547 ; IF NOT - THEN EXIT
2548
2549 006346 021637 006314 IS: CMP @ SP, EPCSCOPE
2550 006352 001401 BEQ SCOPING
2551 006354 000002 RTI
2552
2553 ; SW 11 = 1 TO LOCK ON SCOPING LOOP
2554
2555 ; THIS IS A SCOPING LOOP
2556
2557 006356 032777 004000 172632 SCOPING: BIT #SW11, @SWR
2558 006364 001757 BEQ NOSCOPE ;DO NOT LOOP ON ERROR
2559 006366 013716 006374 MOV PCSCOPE, @ SP
2560 006372 000002 RTI ; LOCK FOR SCOPE LOOP
2561 006374 000000 PCSCOPE: 0

```


MAINDEC-11-DZRXB-D MACY11 27(663) 6-NOV-75 08:50 PAGE 62
DZRXB0.P11 " SCOPE " TRAP SERVICE ROUTINE

SEQ 0143

2616	006504	042503	052040	051505	
2617	006512	020124	052510	043516	
2618	006520	040040	050040	020103	
2619	006526	000			
2620	006527	040	040520	051523	MPASS: .ASCIZ " PASS ="
2621	006534	036440	000		
2622		006540			.EVEN
2623					
2624	006540	005037	006560		XSON: CLR HANGER
2625	006544	012737	177740	006562	MOV #177740, HANGPL
2626	006552	062716	000002		ADD #2, 2 SP
2627	006556	000207			RTS PC
2628	006560	000000			HANGER: 0
2629	006562	177740			HANGPL: 177740

; UPDATE FOR EXIT

.SBTTL DRIVE TEST SELECTION

;DO A READ ONLY FUNCTION ON ALL SECTORS.
;THIS DOES NOT VERIFY THE DATA, ONLY TESTS FOR CRC ERRORS.RDONLY: JSR PC,INITTRACKS
JSR PC,GETUNIT
IS: JSR PC,GETTRACK
JSR PC,READ
DEC TRKNTR
BNE IS
RTS PC

;*****

;WRITE AND READ DATA ON SPECIFIED TRACK AND ALTERNATE
;DRIVES BEFORE GOING TO THE NEXT TRACK.DRVSWP: JSR PC,GETPATTERN
JSR PC,INITTRACKS
IS: JSR PC,GETUNIT
JSR PC,GETTRACK
JSR PC,WRITE
JSR PC,READCHK
JSR PC,GETUNIT
JSR PC,WRITE
JSR PC,READCHK
DEC TRKNTR
BNE IS
RTS PC

;*****

;WRITE ALL SECTORS AND READ/VERIFY ALL SECTORS

WTRDCK: JSR PC,GETPATTERN
XWTRDCK: JSR PC,INITTRACKS
JSR PC,GETUNIT
IS: JSR PC,GETTRACK
JSR PC,WRITE
JSR PC,READCHK
DEC TRKNTR
BNE IS
JSR PC,DONE
RTS PC
BR XWTRDCK;HAVE BOTH DRIVES BEEN TESTED
;YES
;NO, GO TO OTHER UNIT2630
2631
2632
2633
2634
2635 006564 004737 012474
2636 00-570 004737 012356
2637 00-574 004737 012626
2638 006600 004737 007576
2639 006604 005337 012646
2640 006610 001371
2641 006612 000207
2642
2643
2644
2645
2646
2647
2648 006614 004737 012002
2649 006620 004737 012474
2650 006624 004737 012356
2651 006630 004737 012626
2652 006634 004737 006736
2653 006640 004737 010400
2654 006644 004737 012356
2655 006650 004737 006736
2656 006654 004737 010400
2657 006660 005337 012646
2658 006664 001357
2659 006666 000207
2660
2661
2662
2663
2664
2665
2666 006670 004737 012002
2667 006674 004737 012474
2668 006700 004737 012356
2669 006704 004737 012626
2670 006710 004737 006736
2671 006714 004737 010400
2672 006720 005337 012646
2673 006724 001367
2674 006726 004737 012450
2675 006732 000207
2676 006734 000757

```

2677          .SBTTL  WRITE FUNCTION
2678
2679 006736 004737 013046      WRITE:      JSR PC,INITSECTOR      ;SET UP FIRST, LAST, AND SECTOR COUNTER
2680 006742 004737 013144      XWRITE:     JSR PC,GETSECTOR      ;PICK UP NEXT SECTOR
2681 006746 004737 010532      FILLBUF:    JSR PC,ADJSUM        ;ADJUST DATA BUFFER AND CHECK SUM FOR ADDRESSES
2682 006752 012746 007130      MOV #FILLDONE, -(SP)      ;PUT GOOD RETURN ON STACK
2683 006756 012746 007024      MOV #FILLER, -(SP)        ;PUT ERROR RETURN ON STACK
2684 006762 005037 011206      CLR BYTECNTR
2685 006766 005046      CLR                ;LOWER 'CPU' LEVEL
2686 006770 012746 006776      MOV          #1$, -(SP)      ;SET RETURN 'PC'
2687 006774 000002      RTI                          ;GET 'CPU' LEVEL INTO 'PSW'
2688 006776 012777 000101 172202 1$:      MOV #FBIE, ARXCS    ;EXECUTE FILLBUFER COMMAND
2689 007004 105777 172176      FILLFLAG:   TSTB ARXCS        ;TEST FOR TRANSFER REQUEST FLAG
2690 007010 100375      BPL FILLFLAG
2691 007012 112077 172172      XFRBYTE:    MOVB (RO)+, ARXDB  ;TRANSFER DATA BYTE
2692 007016 005237 011206      INC BYTECNTR
2693 007022 000770      BR FILLFLAG          ;WAIT FOR NEXT TR FLAG
2694
2695 007024 005726      FILLER:      TST (SP)+      ;REMOVE THE DONE RETURN FROM THE STACK
2696 007026 012737 015267 007066      MOV #MFIL, PTYPE1+2    ;PUT ADDR OF FILLBUF MESSAGE IN PAR ERR TYP0UT 1
2697 007034 012737 006742 007126      MOV #XWRITE, PCONT+2  ;IF NO LOOP ON ERROR GO TO NEXT SECTOR
2698 007042 012737 006746 007122      MOV #FILLBUF, PLOOP+2 ;IF LOOP ON ERROR RETURN THROUGH PLOOP
2699 007050 000137 007054      JMP PARTEST      ;PRINT OUT PAR ERR AND TEST CONDITIONS FOR RETRY
2700
2701 007054 032777 020000 172134 PARTEST:  BIT #SW13, ASWR    ;TEST DON'T PRINT ERROR SWITCH
2702 007062 001006      BNE CONT4
2703 007064 104400 000000      PTYPE1:     TYPE ,OPEN        ;PRINT THE PARITY ERROR MESSAGE
2704 007070 104400 015500      TYPE ,MPAR
2705 007074 104400 015030      TYPE ,MCRLF
2706 007100 005777 172112      CONT4:      TST ASWR          ;TEST HALT ON ERROR SWITCH
2707 007104 100001      BPL CONT13
2708 007106 000000      HLT6:      HALT
2709 007110 032777 004000 172100 CONT13:   BIT #SW11, ASWR    ;HALT ON ERROR
2710 007116 001402      BEQ PCONT        ;TEST LOOP ON ERROR SWITCH
2711 007120 000137 006746      PLOOP:     JMP FILLBUF        ;IF NOT SET GO TO NEXT SECTOR
2712 007124 000137 007714      PCONT:     JMP NEXTRD        ;RETURN TO LOOP ON TEST THROUGH HERE
2713
2714 007130 005037 006560      FILLDONE:   CLR HANGER        ;GO TO NEXT SECTOR THROUGH HERE
2715 007134 012746 007226      MOV #WRTDONE, -(SP)      ;SET GOOD RETURN ON STACK
2716 007140 012746 007242      MOV #WATER, -(SP)        ;SET ERROR RETURN ON STACK
2717 007144 112737 000105 007472      MOVB #WRTIE, FUNCTION ;SET FUNCTION WORD TO WRITE
2718 007152 022737 005676 006374      CMP #T25, PCSCOPE    ;IS THIS THE DELETED DATA TEST
2719 007160 001003      BNE 1$
2720 007162 112737 000115 007472      MOVB #WTD0IE, FUNCTION
2721 007170 004737 007424      1$:      JSR PC, COMMAND      ;TRANSFER COMMAND TO DRIVE
2722 007174 005046      CLR                ;LOWER 'CPU' LEVEL
2723 007176 012746 007204      MOV          #2$, -(SP)      ;SET RETURN 'PC'
2724 007202 000002      RTI                          ;GET 'CPU' LEVEL INTO 'PSW'
2725 007204 032777 000040 171774 2$:      BIT #DONEBIT, ARXCS ;WAIT FOR DONE
2726 007212 001774      BEQ 2$
2727 007214 005237 006560      3$:      INC HANGER        ;WAIT FOR INTERRUPT
2728 007220 001375      BNE 3$
2729 007222 000137 011216      JMP NOINTER      ;NO INTERRUPT ERROR

```

```

2730 007226 005337 013136      WRTDONE:      DEC SECCNTR      ;TEST SECTOR COUNTER
2731 007232 001001              BNE 25          ;NOT LAST SECTOR GO TO NEXT ONE
2732 007234 000207              RTS PC
2733 007236 000137 006742      25:             JMP XWRITE
2734
2735 007242 005726              WRTER:          TST (SP)+      ;REMOVE THE DONE RETURN FROM THE STACK
2736 007244 032737 000002 011670      BIT #BIT1,ASTAT    ;IS THIS A PARITY ERROR
2737 007252 001413              BEQ WRTSEK      ;NO, IT MUST BE A SEEK ERROR
2738                                ;PARITY ERROR DURING A WRITE FUNCTION
2739 007254 012737 015471 007066      MOV #MWRITE,PTYP1+2  ;PUT ADDR OF WRITE MESSAGE IN PAR ER TYP0UT 1
2740 007262 012737 007226 007126      MOV #WRTDONE,PCONT+2  ;IF NO LOOP GO TO NEXT SECTOR
2741 007270 012737 007130 007122      MOV #FILLDONE,PLOOP+2 ;IF LOOP RETURN THROUGH PLOOP TO REWRITE
2742 007276 000137 007054              JMP PARTEST    ;GO INC LOG AND TEST FOR RETRY
2743
2744                                ;SEEK ERROR DURING A WRITE FUNCTION
2745 007302 012737 006746 007374      WRTSEK:      MOV #FILLBUF,SEKRTY+2 ;SETUP FOR WRT RETRY ON SEEK ERROR
2746                                ;(AFTER A RECAL. THE CONTENTS OF SECTOR 1
2747                                ;TRACK 1 ARE LOADED INTO THE SECTOR BUFFER.
2748                                ;TO REWRITE THE CORRECT DATA THE PROGRAM
2749                                ;MUST REFILL THE SECTOR BUFFER.
2750 007310 012737 015471 007336      MOV #MWRITE,STYP1+2  ;PUT ADDR OF WRITE MESSAGE IN SEEK ER TYPEOUT 1
2751 007316 004737 007324              JSR PC,SEEKER    ;RECORD SEEK ERROR
2752 007322 000741              BR WRTDONE      ;GO TO NEXT SECTOR CAN'T FIND THIS ONE
2753
2754 007324 032777 020000 171664      SEEKER:      BIT #SW13,JSWR    ;CHECK DON'T PRINT ERROR SWITCH
2755 007332 001004              BNE SWHLT1
2756 007334 104400 015471      STYP1:          TYPE MWRITE    ;PRINT WRITE (READ) SEEK ERROR
2757 007340 004737 007376              JSR PC,SEKTYP
2758 007344 005777 171646      SWHLT1:         TST JSWR      ;TEST THE HALT ON ERROR SWITCH
2759 007350 100001              BPL CONT14
2760 007352 000000      HLT7:             HALT          ;HALT ON THE ERROR
2761 007354 004737 007476      CONT14:         JSR PC,HOMER    ;RECALIBRATE ON SEEK ERRORS
2762 007360 032777 004000 171630      BIT #SW11,JSWR    ;CHECK THE LOOP ON ERROR SWITCH
2763 007366 001001              BNE SEKRTY      ;IF SET LOOP ON THE ERROR THROUGH SEEK RETRY.
2764 007370 000207              RTS PC
2765 007372 000137 006746      SEKRTY:         JMP FILLBUF    ;RETRY WRITE COMMAND (READ COMAND)
2766
2767 007376 104400 015456      SEKTYP:         TYPE MFEK      ;TYPE SEEK ERROR
2768 007402 104400 014742      TYPE MFEK      ;TYPE ADDRESS OF TRACK MOVED FROM
2769 007406 013746 012652      PRESTRK,-(SP)  ;SAVE PRESTRK FOR TYPEOUT
2770 007412 104402              MOV TYP05      ;GO TYPE--OCTAL ASCII
2771 007414 003              .BYTE 3          ;TYPE 3 DIGIT(S)
2772 007415 000              .BYTE 0          ;SUPPRESS LEADING ZEROS
2773 007416 104400 015030      TYPE MCRLF
2774 007422 000207              RTS PC
2775

```

2776	007424	153737	012446	007472	COMMWORD:	BISB UNITSEL,FUNCTION	;SET UNIT SELECTION BIT IN COMMAND WORD
2777	007432	013777	007472	171546		MOV FUNCTION,2RXCS	;SEND OUT COMMAND TO DRIVE
2778	007440	004737	006376		1\$:	JSR PC,STR	;WAIT FOR TR FLAG
2779	007444	000775				BR 1\$	
2780	007446	113777	013140	171534		MOVB TSECTOR,2RXDB	;SEND OUT TARGET SECTOR
2781	007454	004737	006376		2\$:	JSR PC,STR	;WAIT FOR TR FLAG
2782	007460	000775				BR 2\$	
2783	007462	113777	012650	171520		MOVB TARGET,2RXDB	;SEND OUT TARGET TRACK
2784	007470	000207				RTS PC	
2785							
2786	007472	000000			FUNCTION:	0	
2787	007474	000000			DATAK:	0	;DATA CHECK ON CRC ERROR FLAG
2788							
2789	007476	032777	000400	171512	HOME:	BIT #SW8,2SWR	;TEST NO RECAL SWITCH
2790	007504	001033				BNE RTN	
2791	007506	012777	040001	171472		MOV #RECAL,2RXCS	;ISSUE RECAL FUNCTION
2792	007514	004737	006412		2\$:	JSR PC,SDN	
2793	007520	000775				BR 2\$	
2794	007522	005777	171460			TST 2RXCS	;WAS THERE AN ERROR
2795	007526	100017				BPL XHOME	;NO
2796	007530	032777	020000	171460		BIT #BIT13,2SWR	;YES, SHOULD IT BE PRINTED
2797	007536	001002				BNE 1\$	;NO
2798	007540	004737	011674			JSR PC,ROCODE	
2799	007544	005777	171446		1\$:	TST 2SWR	;TEST HALT ON ERROR SWITCH
2800	007550	100001				BPL 3\$	
2801	007552	000000				HALT	
2802	007554	032777	004000	171434	3\$:	BIT #SW11,2SWR	;TEST LOOP ON ERROR SWITCH
2803	007562	001345				BNE HOME	
2804	007564	000207				RTS PC	
2805	007566	012737	000001	012652	XHOME:	MOV #1,PRESTRK	;SET THE PRESENT TRACK TO TRACK 1
2806	007574	000207			RTN:	RTS PC	
2807							

```

2808 .SBTTL READ DATA FROM THE DISKETTE
2809
2810
2811 007576 004737 013046 READ: JSR PC,INITSECTOR
2812 007602 004737 013144 XREAD: JSR PC,GETSECTOR
2813 007606 005037 007474 REREAD: CLR DATAACK ;CLEAR CRC DATA CHECK FLAG
2814 007612 005037 006560 CLR HANGER
2815 007616 012746 007672 MOV #RDDONE,-(SP) ;SET GOOD RETURN ON STACK
2816 007622 012746 007724 MOV #RDERR,-(SP) ;SET READ ERROR RETURN ON STACK
2817 007626 112737 000107 007472 MOV #RDIE,FUNCTION
2818 007634 004737 007424 JSR PC,COMWORD
2819 007640 005046 CLR -(SP) ;LOWER 'CPU' LEVEL
2820 007642 012746 007650 MOV #15,-(SP) ;SET RETURN 'PC'
2821 007646 000002 RTI ;GET 'CPU' LEVEL INTO 'PSW'
2822 007650 032777 000040 171330 15: BIT #DONEBIT,DRXCS ;WAIT FOR DONE BIT
2823 007656 001774 BEQ 15
2824 007660 005237 006560 25: INC HANGER ;WAIT FOR INTERRUPT
2825 007664 001375 BNE 25
2826 007666 000137 011216 JMP NOINTER ;NO INTERRUPT ON DONE
2827
2828 007672 022737 005310 006374 RDDONE: CMP #T20,PCSCOPE ;IS THIS THE READ ONLY TEST (T20)
2829 007700 001405 BEQ NEXTFD ;YES,DON'T CHECK FOR DELETED DATA
2830 007702 004737 010162 JSR PC,DDCHK ;CHECK FOR DELETED DATA INDICATOR
2831 007706 005701 TST R1 ;BIT 15 OF R1 IS READ 1 SECTOR FLAG
2832 007710 100001 BPL NEXTRD
2833 007712 006207 RTS PC ;IF SET,GO VERIFY DATA JUST READ
2834 007714 005337 013136 NEXTRD: DEC SECCNTR
2835 007720 001330 BNE XREAD
2836 007722 000207 RTS PC ;READ FUNCTION IS DONE
2837
2838 007724 005726 RDERR: TST (SP)+ ;REMOVE THE DONE RETURN FROM THE STACK
2839 007726 032737 000002 011670 BIT #BIT1,ASTAT ;IS THIS A PARITY ERROR
2840 007734 001413 BEQ 15 ;NO, SEE IF ITS A CRC ERROR
2841 ;PARITY ERROR DURING A READ FUNCTION
2842 007736 012737 015431 007066 MOV #MREAD,PTYP1+2 ;PUT ADDR OF READ MESSAGE IN PAR ERR TYPEOUT 1
2843 007744 012737 007606 007122 MOV #REREAD,PLOOP+2 ;IF LOOP ON ERROR LOOP THROUGH PLOOP
2844 007752 012737 007714 007126 MOV #NEXTRD,PCONT+2 ;IF NO LOOP GO TO NEXT READ
2845 007760 000137 007054 JMP PARTEST ;RECORD PARITY ERROR AND RETRY FUNCTION
2846 007764 032737 000001 011670 15: BIT #BIT0,ASTAT ;IS THIS A CRC ERROR
2847 007772 001011 BNE CACER ;YES GO TEST AND LOG IT
2848 ;SEEK ERROR DURING A READ FUNCTION
2849 007774 012737 007606 007374 MOV #FREAD,SEKRTY+2 ;SET SEEK CONTINUE FOR READ RETRY
2850 010002 012737 015431 007336 MOV #MREAD,STYP1+2 ;SET ADDR OF READ MESSAGE IN SEEK ER TYPEOUT 1
2851 010010 004737 007324 JSR PC,SEEKER ;RECORD SEEK ERROR
2852 010014 000737 BR NEXTD ;GO TO NEXT SECTOR,CAN'T READ THIS ONE

```

2853						;CRC ERROR DETECTED WHILE READING	
2854							
2855	010016	005701			CRCER:	TST R1	;IF READ ONLY, REPORT DATA CRC ERROR
2856	010020	100032				BPL DATACRC	
2857	010022	005237	007474			INC DATAACK	;SET DATA CHECK FLAG
2858	010026	004737	010410			JSR PC,EMPBUFF	;CHECK FOR A DATA ERROR
2859	010032	005737	011214			TST ERCONTR	;WAS THERE A DATA ERROR
2860	010036	001023				BNE DATACRC	;YES, REPORT IT
2861	010040	032777	020000	171150		BIT #SW13,@SWR	;TEST DON'T PRINT SWITCH
2862	010046	001004				BNE 2\$	
2863	010050	104400	015401			TYPE ,MBADCRC	;TYPE CRC GENERATOR ERROR
2864	010054	104400	015030			TYPE ,MCRLF	
2865	010060	005777	171132		2\$:	TST @SWR	;TEST HALT ON ERROR SWITCH
2866	010064	100001				BPL CONT15	
2867	010066	000000			HLT10:	HALT	;HALT ON ERROR
2868	010070	032777	004000	171120	CONT15:	BIT #SW11,@SWR	;CHECK LOOP ON ERROR SWITCH
2869	010076	001001				BNE 3\$	
2870	010100	000705				BR NEXTRD	;DON'T LOOP GO TO NEXT SECTOR
2871	010102	000137	007606		3\$:	JMP REREAD	;LOOP ON TEST.
2872							
2873							
2874						;DATA CRC ERROR	
2875	010106	032777	020000	171102	DATACRC:	BIT #SW13,@SWR	;TEST DON'T PRINT ERROR SWITCH
2876	010114	001004				BNE 4\$	
2877	010116	104400	015437			TYPE ,MCRC	;TYPE DATA CRC ERROR
2878	010122	104400	015030			TYPE ,MCRLF	
2879	010126	005777	171064		4\$:	TST @SWR	;TEST HALT ON ERROR SWITCH
2880	010132	100001				BPL CONT16	
2881	010134	000000			HLT12:	HALT	;HALT ON ERROR
2882	010136	032777	004000	171052	CONT16:	BIT #SW11,@SWR	;TEST LOOP ON ERROR
2883	010144	001004				BNE 5\$	;IF SET LOOP ON THE TEST
2884	010146	002706	000002			ADD #2,SP	;REMOVE READ DONE ADDRESS FROM STACK
2885	010152	000137	007714			JMP NEXTRD	;READ NEXT SECTOR CAN'T READ THIS ONE
2886	010156	000137	007606		5\$:	JMP REREAD	;NO,GO REREAD THIS SECTOR
2887							
2888							

```

2899 010162 022737 005676 006374 DOCHK:    CMP #T25,PCSCOPE      ;IS THIS TEST 25
2890 010170 001037                                BNE CONT10
2891 010172 132737 000100 011670            BITB #BIT6,ASTAT      ;THIS IS TEST 25
2892 010200 001053                                BNE RETURN          ;DD BIT SHOULD BE SET
2893 010202 032777 020000 171006            BIT #SW13,DSWR       ;TEST DON'T PRINT ERROR SWITCH
2894 010210 001013                                BNE CONT11
2895 010212 004737 010332                    JSR PC,ERMSG
2896 010216 104400 014657                    TYPE ,MDDMIS          ;TYPE MISSING DELETED DATA BIT
2897 010222 052737 000400 012446 DOERR:      BIS #BIT8,UNITSEL    ;SET HARD ERROR FLAG
2898 010230 004737 011572                    JSR PC,TYPAL 3      ;TYPE ADDRESS OF ERROR
2899 010234 104400 015030                    TYPE ,MCRLF
2900 010240 005777 170752                    TST DSWR          ;TEST HALT ON ERROR SWITCH
2901 010244 100001                                BPL CONT17
2902 010246 000000                                HALT                ;HALT ON DELETED DATA ERROR
2903 010250 032777 004000 170740 CONT17:      BIT #SW11,DSWR    ;TEST LOOP ON ERROR
2904 010256 001402                                BEQ 4$
2905 010260 000137 007606                    JMP REREAD          ;LOOP ON TEST
2906 010264 000137 007714                    JMP NEXTRD          ;READ NEXT SECTOR
2907 010270 032737 000100 011670 CONT10:      BIT #BIT6,ASTAT      ;THIS IS NOT A DELETED DATA TRANSFER
2908 010276 001414                                BEQ RETURN
2909 010300 052737 000400 012446            BIS #BIT8,UNITSEL    ;SET HARD ERROR FLAG
2910 010306 032777 020000 170702            BIT #SW13,DSWR       ;TEST DON'T PRINT ERROR SWITCH
2911 010314 001351                                BNE CONT11
2912 010316 004737 010332                    JSR PC,ERMSG
2913 010322 104400 014631                    TYPE ,MUNXDO          ;TYPE UNEXPECTED DELETED DATA BIT
2914 010326 000735                                BR DOERR
2915 010330 000207                                RTS PC
2916
2917
2918 010332 104400 015033                    ERMSG:    TYPE ,MERHEADER
2919 010336 013746 006374                    MOV PCSCOPE,-(SP)    ;SAVE PCSCOPE FOR TYPEOUT
2920 010342 104402                                TYP0S              ;GO TYPE--OCTAL ASCII
2921 010344 006                                .BYTE 6            ;TYPE 6 DIGITS
2922 010345 000                                .BYTE 0            ;SUPPRESS LEADING ZEROS
2923 010346 104400 006527                    TYPE ,MPASS
2924 010352 013737 002340 010364            MOV PASS,1$
2925 010360 004537 014504                    JSR RS,SGLDEC
2926 010354 000000                                OPEN
2927 010356 104400 015030                    TYPE ,MCRLF
2928 010372 004737 006272                    JSR PC,DING
2929 010376 000207                                RTS PC
2930
2931

```

```

2932 .SBTTL READ AND VERIFY DATA
2933
2934 ;READ A SECTOR,EMPTY THE SECTOR BUFFER AND VERIFY
2935 ;THE DATA READ AGAINST CORE DATA BUFFER
2936
2937 010400 052701 100000 READCHK: BIS #BIT15,R1 ;SET READ ONE SECTOR FLAG
2938 010404 004737 007576 JSR PC,READ ;GO READ ONE SECTOR
2939 010410 005737 013136 EMPBUFF: TST SECCNTR ;IF CLEARED NO SECTORS WERE FOUND
2940 010414 001002 BNE 1$
2941 010416 000137 011202 JMP EXIT ;GO TO NEXT TRACK
2942 010422 005037 011206 1$: CLR BYTECNTR ;CLEAR THE BYTE AND ERROR COUNTERS
2943 010426 005037 011214 CLR ERNCTR
2944 010432 052701 000200 BIS #BIT7,R1 ;R1 BIT 7 IS USED AS FIRST ERROR FLAG
2945 010436 004737 010532 JSR PC,ADJSUM ;ADJUST DATA AND CK SUM FOR ADDRESSES
2946 010442 005037 010622 CLR CKSUM ;SET UP FOR CHECK SUM ACCUMULATION
2947 010446 012746 011046 MOV #EMPDONE,-(SP) ;SET UP RETURN ADDRESSES
2948 010452 012746 010624 MOV #EMPER,-(SP)
2949 010456 005046 CLR ;LOWER 'CPU' LEVEL
2950 010460 012746 010466 MOV #2$,-(SP) ;SET RETURN 'PC'
2951 010464 000002 RTI ;GET 'CPU' LEVEL INTO 'PSW'
2952 010466 012777 000103 170512 2$: MOV #EBIE,DRXCS ;LOAD EMPTY BUFFER FUNCTION
2953 010474 105777 170506 EMPFLAG: TSTB DRXCS ;TEST FOR TR FLAG
2954 010500 100375 BPL EMPFLAG
2955
2956 010502 117737 170502 011210 CKBYTE: MOVB DRXDB,BADBYTE ;SAVE BYTE FROM DISKETTE
2957 010510 063737 011210 010622 ADD BADBYTE,CKSUM ;ACCUMULATE CHECK SUM
2958 010516 123720 011210 CMPB BADBYTE,(R0)+ ;COMPARE AGAINST GOOD BYTE
2959 010522 001054 BNE DATAER ;IF NOT EQUAL GO TO DATAER
2960 010524 005237 011206 INC BYTECNTR
2961 010530 000761 BR EMPFLAG ;GET NEXT BYTE
2962
2963 010532 113737 012650 016300 ADJSUM: MOVB TARGET,BUFADR ;SET FIRST AND SECOND BYTES WITH ADDRESSES
2964 010540 113737 013140 016301 MOVB TSECTOR,BUFADR+1
2965 010546 013737 012260 010622 MOV SUM,CKSUM ;GET THE PATTERN SUM
2966 010554 063737 012650 010622 ADD TARGET,CKSUM ;ADD TRACK ADDRESS TO CHECK SUM
2967 010562 063737 013140 010622 ADD TSECTOR,CKSUM ;ADD SECTOR ADDRESS TO CHECK SUM
2968 010570 113737 010622 016476 MOVB CKSUM,BUFADR+176 ;INSERT CHECK SUM TO DATA BUFFER
2969 010576 106337 010622 ASLB CKSUM ;GENERATE NEGATIVE CHECK SUM
2970 010602 105437 010622 NEGB CKSUM
2971 010606 113737 010622 016477 MOVB CKSUM,BUFADR+177 ;INSERT NEG SUM INTO DATA BUFFER
2972 010614 012700 016300 MOV #BUFADR,R0 ;SET ADDRESS OF BYTE IN R0
2973 010620 000207 RTS PC ;RETURN
2974
2975 010622 000000 CKSUM: 0
2976
2977 010624 005726 EMPER: TST (SP)+ ;REMOVE THE DONE RETURN FROM THE STACK
2978 010626 012737 015303 007066 MOV #EMPTY,PTYP1+2 ;PUT ADDR OF EMPTYBUF MESSAGE IN PAR ER TYP0UT 1
2979 010634 012737 010410 007122 MOV #EMPBUFF,PLOOP+2 ;RETURN THROUGH HERE TO LOOP ON ERROR
2980 010642 012737 011164 007126 MOV #NXREAD,PCONT+2 ;IF NO LOOP ON ERROR GO TO NEXT SECTOR
2981 010650 000137 007054 JMP PARTEST ;REPORT PARITY ERROR
2982

```

2983	010654	052737	000400	012446	DATAER:	BIS #BIT8,UNITSEL	;SET THE HAD ERROR FLAG
2984	010662	005237	011214			INC ERCNTR	;INC THE BYTE ERROR COUNTER
2985	010666	032777	020000	170322		BIT #SW13,2SWR	;TEST PRINT ERROR SW IN SWR
2986	010674	001054				BNE NOERTYP	;DON'T PRINT THE ERROR
2987	010676	032777	001000	170312		BIT #SW9,2SWR	;TEST PRINT 10 ERRORS SWITCH
2988	010704	001004				BNE 15	;IF SET PRINT ALL ERRORS
2989	010706	023727	011214	000012		CMP ERCNTR,#10.	;HAVE 10 ERRORS BEEN TYPED
2990	010714	003044				BGT NOERTYP	;YES,DON'T PRINT ANY MORE
2991	010716	105701			15:	TSTB R1	;TEST FIRST ERROR FLAG
2992	010720	100014				BPL TYPERR	
2993	010722	004737	010332			JSR PC,ERMSG	;PRINT ADDRESS OF TEST
2994	010726	104400	014702			TYPE ,MDERHOR	;FIRST ERROR, PRINT ERROR HEADER
2995	010732	104400	015030			TYPE ,MCRLF	
2996	010736	004737	011572			JSR PC,TYPADR	;PRINT TRACK AND SECTOR LOCATIONS
2997	010742	104400	014771			TYPE ,MCLMUN	;SET UP COLMUN HEADINGS
2998	010746	042701	000200			BIC #BIT7,R1	;CLEAR FIRST ERROR FLAG
2999	010752	013737	011206	010764	TYPERR:	MOV BYTECNTR,15	;PRINT BYTE NUMBER
3000	010760	004537	014504			JSR R5,SGLDEC	
3001	010764	000000			15:	OPEN	
3002	010766	104400	015025			TYPE ,DBLSP	
3003	010772	013746	011210			MOV BADBYTE,-(SP)	;PRINT BYTE READ FROM DISKETTE
3004	010776	104402				TYP05	
3005	011000	000003				.WORD 3	
3006	011002	104400	015025			TYPE ,DBLSP	
3007	011006	114037	011212			MOVB -(R0),GOODBYTE	;GET GOOD BYTE
3008	011012	005200				INC R0	;RETURN R0 TO NEXT BYTE IN BUFFER
3009	011014	013746	011212			MOV GOODBYTE,-(SP)	
3010	011020	104403				TYP0N	;PRINT GOOD DATA
3011	011022	104400	015030			TYPE ,MCRLF	
3012	011026	005777	170164		NOERTYP:	TST 2SW	;TEST HALT ON ERROR SWITCH
3013	011032	100001				BPL CONT20	
3014	011034	000000			HLT14:	HALT	
3015	011036	005237	011206		CONT20:	INC BYTECNTR	
3016	011042	000137	010474			JMP EMPFLAG	
3017							

```

3018 011046 005737 007474 EMPDONE: TST DATAK ; WAS THIS READ CHECK CAUSED BY A CRC ERROR
3019 011052 001401 BEQ 1$ ; NO
3020 011054 000207 RTS PC ; YES, RETURN TO CRC HANDLER
3021 011056 005737 011214 1$: TST ER CNTR ; WAS THERE ERRORS
3022 011062 001440 BEQ NXREAD ; NO ERRORS
3023 011064 032777 020000 170124 BIT #SW13, 2SWR ; YES, TEST DON'T PRINT SWITCH
3024 011072 001024 BNE 2$ ; DON'T PRINT THE ERROR
3025 011074 104400 015234 TYPE ,MERC ; PRINT THE TOTAL DATA ERROR COUNT
3026 011100 013737 011214 011112 MOV ER CNTR, 3$
3027 011106 004537 014504 JSR R5, SGLDEC
3028 011112 000000 3$: OPEN
3029 011114 104400 015551 TYPE ,MSUM ; INDICATE IF CHECK SUM WAS GOOD OR HAD ERRORS
3030 011120 105737 010622 TSTB CKSUM
3031 011124 001403 BEQ 4$
3032 011126 104400 015536 TYPE ,MBAD
3033 011132 000402 BR 5$
3034 011134 104400 015544 TYPE ,MG00D
3035 011140 104400 015030 TYPE ,MCRLF
3036 011144 032777 004000 170044 2$: BIT #SW11, 2SWR ; TEST LOOP ON ERROR SWITCH
3037 011152 001404 BEQ NXREAD ; IF NOT SET GO TO NEXT SECTOR
3038 011154 004737 007606 JSR PC, REREAD ; YES, GO REREAD THE DATA
3039 011160 000137 010410 JMP EMPBUFF ; GO RECHECK THE DATA
3040 011164 005337 013136 NXREAD: DEC SECCNTR
3041 011170 001404 BEQ EXIT
3042 011172 004737 007602 JSR PC, XREAD ; READ THE NEXT SECTOR
3043 011176 000137 010410 JMP EMPBUFF
3044 011202 005001 EXIT: CLR R1 ; CLEAR THE ONE READ FLAG
3045 011204 000207 RTS PC
3046
3047 011206 000000 BYTE CNTR: 0
3048 011210 000000 BAD BYTE: 0
3049 011212 000000 GOOD BYTE: 0
3050 011214 000000 ER CNTR: 0
3051
3052 ;*****
3053
3054 ;AN INTERRUPT DID NOT OCCURE AT A FUNCTION DONE FLAG.
3055
3056 011216 032777 020000 167772 NOINTER: BIT #SW13, 2SWR ; TEST DON'T PRINT ERROR SWITCH
3057 011224 001006 BNE 1$
3058 011226 004737 010332 JSR PC, ERMSG
3059 011232 104400 015157 TYPE ,MINTER ; TYPE NO INTERRUPT ON DONE ERROR
3060 011236 104400 015030 TYPE ,MCRLF
3061 011242 005777 167750 1$: TST 2SWR ; TEST HALT ON ERROR SWITCH
3062 011246 100001 BPL CONT21
3063 011250 000000 HLT15: HALT ; HALT ON ERROR
3064 011252 004737 011256 CONT21: JSR PC, INTSERV ; JSR TO INTSERV AS IF IT WAS AN INTERRUPT
3065

```

```

3066 .SBTTL INTERRUPT SERVICE
3067
3068 011256 117737 167726 011670 INTSERV: MOV B DRXDB,ASTAT ;SAVE THE ERROR AND STATUS WORD
3069 011264 005777 167716 TST DRXCS ;TEST THE ERROR FLAG
3070 011270 100442 BMI RXERROR ;THERE WAS AN ERROR GO REPORT IT
3071 011272 032737 000004 011670 BIT #BIT2,ASTAT ;IS INIT DONE SET
3072 011300 001402 BEQ 2$ ;NO CONTINUE
3073 011302 000137 011530 JMP RXPWR ;YES,REPORT POWER FAILED AND RESTART
3074 011306 032737 000003 011670 2$: BIT #3,ASTAT ;ARE PAR OR CRC BITS SET
3075 011314 001020 BNE 1$ ;YES GO REPORT ERROR
3076 011316 132777 000040 167662 BITB #DONEBIT,DRXCS ;IS DONE SET
3077 011324 001011 BNE 3$ ;IF SET RETURN TO TEST
3078 011326 032777 020000 167662 BIT #SW13,DSWR ;TEST DON'T PRINT ERROR SWITCH
3079 011334 001004 BNE 4$ ;DON'T PRINT
3080 011336 104400 015212 TYPE ,MUKNINT ;TYPE UNKNOWN INTERRUPT
3081 011342 104400 015030 TYPE ,MCRLF
3082 011346 000002 4$: RTI ;RETURN FROM THE INTERRUPT
3083 011350 062706 000006 3$: ADD #6,SP ;BYPASS INTERRUPT POINTERS ON STACK
3084 011354 000207 RTS PC ;RETURN TO PROGRAM
3085 011356 032777 020000 167632 1$: BIT #SW13,DSWR ;TEST DON'T PRINT ERROR SWITCH
3086 011364 001004 BNE RXERROR ;TYPE NO STATUS ERROR ERROR
3087 011366 104400 015515 TYPE ,MNOFLAG ;TYPE NO STATUS ERROR ERROR
3088 011372 104400 015030 TYPE ,MCRLF ;AN ERROR INDICATOR
3089 011376 005237 006312 RXERROR: INC ERRORS ;AN ERROR INDICATOR
3090 011402 001775 BEQ RXERROR ;SET HARD ERROR FLAG
3091 011404 052737 000400 012446 BIS #BIT8,UNITSEL ;SET HARD ERROR FLAG
3092 011412 012777 000017 167566 2$: MOV #RDER,DRXCS ;GET THE ERROR CODE
3093 011420 004737 006412 3$: JSR PC,SDN ;TEST FOR DONE FLAG
3094 011424 000775 BR 3$
3095 011426 032777 000002 167554 BIT #2,DRXDB ;WAS THERE A PARITY ERROR
3096 011434 001403 BEQ 1$ ;NO CONTINUE
3097 011436 004737 011544 JSR PC,PARTYP ;YES GO REPORT THE PARITY ERROR
3098 011442 000763 BR 2$ ;REISSUE THE FUNCTION
3099 011444 117737 167540 011672 1$: MOV B DRXDB,BSTAT ;SAVE THE ERROR CODE IN B STATUS
3100 011452 032777 020000 167536 NOPRNT: BIT #SW13,DSWR ;TEST PRINT ERROR SWITCH IN SWR
3101 011460 001020 BNE 2$
3102 011462 104400 015030 TYPE ,MCRLF
3103 011466 004737 010332 JSR PC,ERMSG ;TYPE ERROR AND MESSAGES
3104 011472 104400 015113 TYPE ,MRXCS ;TYPE COMMAND STATUS REGISTER
3105 011476 013746 007472 MOV FUNCTION,-(SP) ;SAVE FUNCTION FOR TYPEOUT
3106 011502 104402 TYPPOS ;GO TYPE--OCTAL ASCII
3107 011504 006 .BYTE 6 ;TYPE 6 DIGIT(S)
3108 011505 000 .BYTE 0 ;SUPPRESS LEADING ZEROS
3109 011506 004737 011572 JSR PC,TYPADR ;TYPE ADDRESSES AND RUN CONDITIONS
3110 011512 104400 015030 TYPE ,MCRLF
3111 011516 004737 011742 JSR PC,TYP CODE ;PRINT THE STATUS REGISTERS
3112 011522 062706 000004 2$: ADD #4,SP ;MOVE ERROR RETURN TO TOP OF STACK
3113 011526 000207 RTS PC
3114
3115 011530 104400 015566 RXPWR: TYPE ,MRX11 ;ONLY THE RX11 POWER HAS FAILED
3116 011534 104400 014200 TYPE ,SPOWER ;PRINT POWER FAILED
3117 011540 000137 001306 JMP RESTART ;GO TO RESTART

```

```

3118 011544 104400 015113      PARTYP:      TYPE ,MRXCS
3119 011550 017746 167432      MOV          2RXCS,-(SP)      ;;SAVE 2RXCS FOR TYPEOUT
3120 011554 104402      TYPPOS      GO TYPE--OCTAL ASCII
3121 011556      .BYTE      6      ;;TYPE 6 DIGIT(S)
3122 011557      .BYTE      0      ;;SUPPRESS LEADING ZEROS
3123 011560 104400 015500      TYPE ,MPAR
3124 011564 104400 015030      TYPE ,MCRLF
3125 011570 000207      RTS PC
3126
3127 011572 104400 014730      TYPADR:      TYPE ,MTRK      ;TYPE TRACK ADDRESS
3128 011576 013746 012650      MOV          TARGET,-(SP)      ;;SAVE TARGET FOR TYPEOUT
3129 011602 104402      TYPPOS      GO TYPE--OCTAL ASCII
3130 011604      .BYTE      3      ;;TYPE 3 DIGIT(S)
3131 011605      .BYTE      0      ;;SUPPRESS LEADING ZEROS
3132 011606 104400 014757      TYPE ,MSECT      ;TYPE SECTOR ADDRESS
3133 011612 013737 013140 011666      MOV TSECTOR,2$
3134 011620 042737 177740 011666      BIC #177740,2$      ;CLEAR ALL BUT SECTOR ADDRESS
3135 011626 013746 011666      MOV          2$,-(SP)      ;;SAVE 2$ FOR TYPEOUT
3136 011632 104402      TYPPOS      GO TYPE--OCTAL ASCII
3137 011634      .BYTE      2      ;;TYPE 2 DIGIT(S)
3138 011635      .BYTE      0      ;;SUPPRESS LEADING ZEROS
3139 011636 104400 015025      TYPE ,DBLSP
3140 011642 032737 000020 012446      BIT #BIT4,UNITSEL      ;WHICH DRIVE IS BEING USED
3141 011650 001003      BNE 1$
3142 011652 104400 015073      TYPE ,MUNIT0      ;TYPE UNIT 0
3143 011656 000402      BR 4$
3144 011660 104400 015103      TYPE ,MUNIT1      ;TYPE UNIT 1
3145 011664 000207      RTS PC
3146 011666 000000      OPEN
3147
3148 011670 000000      ASTAT:      0
3149 011672 000000      BSTAT:      0
3150
3151
3152 011674 117737 167310 011670      RDCODE:      MOV 2RXDB,ASTAT      ;SAVE THE A STATUS
3153 011702 012777 000017 167276      2$:      MOV #RDR,2RXCS      ;READ THE B STATUS REGISTER
3154 011710 004737 006412      3$:      JSR PC,SON      ;WAIT FOR DONE FLAG
3155 011714 000775      BR 3$
3156 011716 032777 000002 167264      BIT #2,2RXDB      ;WAS THERE A PARITY ERROR
3157 011724 001403      BEQ 1$      ;NO,CONTINUE
3158 011726 004737 011544      JSR PC,PARTYP      ;YES,REPORT THE PARITY ERROR
3159 011732 000763      BR 2$      ;RETRY READING STATUS B
3160 011734 117737 167250 011672      1$:      MOV 2RXDB,BSTAT      ;SAVE THE B STATUS CODES
3161 011742 104400 015123      TYPCODE:      TYPE ,MASTAT      ;TYPE THE CONTENTS OF THE TWO STATUS REGISTERS
3162 011746 013746 011670      MOV          ASTAT,-(SP)      ;;SAVE ASTAT FOR TYPEOUT
3163 011752 104402      TYPPOS      GO TYPE--OCTAL ASCII
3164 011754      .BYTE      3      ;;TYPE 3 DIGIT(S)
3165 011755      .BYTE      0      ;;SUPPRESS LEADING ZEROS
3166 011756 104400 015016      TYPE ,TAB
3167 011762 104400 015137      TYPE ,MBSTAT
3168 011766 013746 011672      MOV BSTAT,-(SP)
3169 011772 104403      TYPON
3170 011774 104400 015030      TYPE ,MCRLF
3171 012000 000207      RTS PC

```

3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219

.SBTTL PATTERN GENERATOR

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 MOST
 SIGNIFICANT BIT OF THIS SECOND BYTE INDICATES THE UNIT. UNIT 0
 IF "0" UNIT 1 IF "1". THE LAST TWO BYTES CONTAIN THE CHECK SUM.
 ;*****

```

GETPATTERN:  MOV #BUFADR,R4      ;SET ADDRESS OF FIRST DATA BYTE
              CLR SUM            ;SET UP FOR ACCUMULATION OF CHECK SUM
              MOV PAT,R5        ;GET PATTERN BITS
              ASL R5
              JMP @PATTERNS(R5)

PATTERNS:    DATA0             ;000 DATA BYTE
              DATA1            ;377 DATA BYTE
              FLOAT0            ;FLOAT A 0 THROUGH ALL 1'S
              FLOAT1            ;FLOAT A 1 THROUGH ALL 0'S
              PAT125            ;125/052 DATA WORD
              PAT314            ;314/063 DATA WORD
              COUNT             ;INCREMENT DATA PATTERN
              RANDATA           ;RANDOM DATA BYTE

DATABYTE:    0
PAT:         0

```

;*****

```

;LOAD SOFTWARE BUFFER WITH ALL ZEROS
; PAT = 0

```

```

DATA0:       CLR DATABYTE
PATGEN:      JSR PC_LOAD      ;GO LOAD THE DATA BUFFER
              BR PATGEN

```

;*****

```

;LOAD SOFTWARE BUFFER WITH ALL ONES
; PAT = 1

```

```

012062 112737 000377 012044 DATA1:  MOVB #377,DATABYTE
012070 000771                BR PATGEN

```

```

3220                                     ;FLOAT A 0 THROUGH ONES IN SOFTWARE BUFFER
3221                                     ; PAT = 2
3222
3223 012072 112737 000376 012044 FLOAT0:      MOVB #376,DATABYTE      ;SET UP A ONES FIELD
3224 012100 000261 XPATGEN:      SEC                      ;SET THE C BIT TO ROTATE THROUGH THE DATA
3225 012102 012702 000000 15:      MOV #0,R2              ;CLR R2 (CAN'T USE "CLR" IT CLEARS "C" BIT)
3226 012106 103001 BCC 25          ;BR IF "C" BIT IS CLEARED
3227 012110 005202 INC R2          ;SET R2 IF "C" BIT IS SET
3228 012112 004737 012232 25:      JSR PC,LOAD            ;GO LOAD THE DATA BUFFER
3229 012116 000241 CLC                      ;CLEAR THE "C" BIT
3230 012120 005702 TST R2              ;IS R2 NONZERO
3231 012122 001401 BEQ 35
3232 012124 000261 SEC                      ;YES, SET THE "C" BIT
3233 012126 106137 012044 35:      ROLB DATABYTE
3234 012132 000763 BR 15
3235
3236 ;*****
3237
3238                                     ;FLOAT A 1 THROUGH ALL ZEROS IN SOFTWARE BUFFER
3239                                     ; PAT = 3
3240
3241 012134 005037 012044 FLOAT1:      CLR DATABYTE
3242 012140 000757 BR XPATGEN
3243
3244 ;*****
3245
3246                                     ;LOAD SOFTWARE BUFFER WITH ALTERNATING 1 AND 0 FOR
3247                                     ;ONE BYTE AND THE COMPLIMENT INTO THE NEXT
3248                                     ; PAT = 4
3249
3250 012142 112737 000125 012044 PAT125:  MOVB #125,DATABYTE
3251 012150 004737 012232 XXPATGEN:  JSR PC,LOAD
3252 012154 105137 012044 COMB DATABYTE
3253 012160 000773 BR XXPATGEN
3254
3255 ;*****
3256
3257                                     ;LOAD SOFTWARE BUFFER WITH ALTERNATING PAIRS OF 1 AND 0 AND
3258                                     ;COMPLIMENT INTO THE NEXT
3259                                     ; PAT = 5
3260
3261 012162 112737 000314 012044 PAT314:  MOVB #314,DATABYTE
3262 012170 000767 BR XXPATGEN
3263
3264 ;*****
3265
3266                                     ;LOAD SOFTWARE BUFFER WITH COUNT PATTERN
3267                                     ; PAT = 6
3268
3269 012172 012737 000377 012044 COUNT:  MOV #377,DATABYTE
3270 012200 005237 012044 15:      INC DATABYTE
3271 012204 004737 012232 JSR PC,LOAD
3272 012210 000773 BR 15

```

```

3273 ;*****
3274
3275 ;LOAD SOFTWARE BUFFER WITH RANDOM DATA PATTERN
3276 ; PAT = 7
3277
3278 012212 004737 012262 RANDATA: JSR PC,RANGEN ;GET RANDOM NUMBER
3279 012216 113737 012354 012044 MOV B,RANUM,DATABYTE
3280 012224 004737 012232 JSR PC,LOAD
3281 012230 000770 BR RANDATA
3282
3283 012232 063737 012044 012260 LOAD: ADD DATABYTE,SUM ;ACCUMULATE THE PATTERN CHECK SUM
3284 012240 113724 012044 MOV B,DATABYTE,(R4)+ ;LOAD THE DATA BUFFER
3285 012244 022704 016500 CMP #BUFADR+200,R4 ;HAVE 128 BYTES BEEN GENERATED
3286 012250 001401 BEQ IS ;IF YES RETURN TO TEST
3287 012252 000207 RTS PC ;IF NO RETURN TO PATTERN GENERATOR
3288 012254 005726 IS: TST (SP)+ ;TAKE PATTERN RETURN ADDRESS OF STACK
3289 012256 000207 RTS PC ;RETURN TO TEST
3290
3291 012260 000000 SUM: 0
3292
3293 012262 012700 000001 RANGEN: MOV #1,R0
3294 012266 063700 012350 ADD RAN1,R0
3295 012272 063700 012352 ADD RAN2,R0
3296 012276 042700 170000 BIC #170000,R0
3297 012302 000241 CLC
3298 012304 005100 ROL R0
3299 012306 006100 ROL R0
3300 012310 010037 012350 MOV R0,RAN1
3301 012314 005000 CLR R0
3302 012316 013700 012352 MOV RAN2,R0
3303 012322 006000 ROR R0
3304 012324 006000 ROR R0
3305 012326 063700 012350 ADD RAN1,R0
3306 012332 042700 170000 BIC #170000,R0
3307 012336 010037 012352 MOV R0,RAN2
3308 012342 010037 012354 MOV R0,RANUM
3309 012346 000207 RTS PC
3310
3311 012350 001234 RAN1: 001234
3312 012352 000765 RAN2: 000765
3313 012354 000000 RANUM: 0
3314

```

```

3315 .SBTTL UNIT SELECTION
3316
3317 ;TEST FOR SELECTED UNITS, DRIVE READY, AND USED CONDITIONS
3318 ;ALSO CONTAINS A "HAD ERROR" FLAG TO BE TESTED AT EOP.
3319 ;THE BITS IN UNITSEL ARE USED AS FOLLOWS
3320
3321 ;BIT15 =UNIT 1 SELECTED VIA SWR
3322 ;BIT14 =UNIT 1 USED BIT
3323 ;BIT8 =THIS PASS HAD AN ERROR
3324 ;BIT7 =UNIT 0 SELECTED VIA SWR
3325 ;BIT6 =UNIT 0 USED BIT
3326 ;BIT4 =UNIT SELECTION FOR FUNCTION WORD
3327
3328 ;*****
3329
3330 012356 032737 000100 012446 GETUNIT: BIT #BIT6,UNITSEL ;WAS UNIT 0 JUST USED
3331 012364 001012 BNE 1$ ;UNIT 0 USED CHECK UNIT 1
3332 012366 105737 012446 TSTB UNITSEL ;WAS UNIT 0 SELECTED
3333 012372 100007 BPL 1$ ;NO GO TO UNIT 1
3334 012374 042737 040020 012446 BIC #40020,UNITSEL ;CLEAR UNIT 1 USED BIT AND FUNCTION UNIT BIT
3335 012402 052737 000100 012446 BIS #BIT6,UNITSEL ;SET UNIT 0 USED BIT
3336 012410 000207 RTS PC
3337 012412 005737 012446 1$: TST UNITSEL ;WAS UNIT 1 SELECTED
3338 012416 100012 BPL 2$ ;NO RETURN
3339 012420 032737 040000 012446 BIT #BIT14,UNITSEL ;HAS UNIT 1 BEEN USED
3340 012426 001006 BNE 2$ ;YES RETURN
3341 012430 042737 000100 012446 BIC #BIT6,UNITSEL ;CLEAR UNIT 0 USED BIT
3342 012436 052737 040020 012446 BIS #40020,UNITSEL ;SET UNIT 1 USED BIT AND FUNCTION UNIT BIT
3343 012444 000207 2$: RTS PC
3344
3345
3346
3347 012446 000000 UNITSEL: 0
3348
3349 ;TEST THAT ALL UNITS HAVE BEEN ACCESSED
3350
3351 012450 005737 012446 DONE: TST UNITSEL ;IS UNIT 1 SELECTED
3352 012454 100006 BPL 1$ ;NO RETURN
3353 012456 032737 040000 012446 BIT #BIT14,UNITSEL ;YES HAS IT BEEN USED
3354 012464 001002 BNE 1$ ;YES RETURN
3355 012466 062716 000002 ADD #2,PC ;BYPASS NOT DONE RETURE ON STACK
3356 012472 000207 1$: RTS PC
3357

```

.SBTTL TRACK SEQUENCE SELECTION

;INITIALIZE TRACK SEQUENCE

;NOTE: IF WORD SEQUEN IS CLEARED THEN TRACK SEQUENCE IS FROM 0-52-53-114 ONLY
 ;IF BIT 15 OF SEQUEN IS "1" THEN TRACK SELECTION IS INC. BETWEEN SELECTED 00/ID LIMITS.
 ;IF BIT 7 IS "1" THEN TEST 25 DECREMENT SEQUENCE IS REQUIRED.

3358											
3359											
3360											
3361											
3362											
3363											
3364											
3365											
3366											
3367											
3368	012474	105737	012660		INITTRACK:	TSTB SEQUEN					
3369	012500	100442				BMI 25					
3370	012502	042737	100200	001200		BIC #100200,00					
3371	012510	005737	001200			TST 00					
3372	012514	001440				BEQ 35					
3373	012516	052737	100000	012660		BIS #BIT15,SEQUEN					
3374	012524	113737	001200	012650		MOVB 00,TARGET					
3375	012532	005037	012656			CLR XID					
3376	012536	113737	001201	012656		MOVB ID,XID					
3377	012544	005037	012654			CLR X00					
3378	012550	113737	001200	012654		MOVB 00,X00					
3379	012556	013737	012656	012646		MOV XID,TRKCNTR					
3380	012564	163737	012654	012646		SUB X00,TRKCNTR					
3381	012572	005237	012646			INC TRKCNTR					
3382	012576	052737	100200	001200	15:	BIS #100200,00					
3383	012604	000207				RTS PC					
3384	012606	012737	000013	012646	25:	MOV #13,TRKCNTR					
3385	012614	000770				BR 15					
3386	012616	012737	000004	012646	35:	MOV #4,TRKCNTR					
3387	012624	000764				BR 15					
3388											
3389											
3390											
3391											
3392	012626	113737	012650	012652	GETTRACK:	MOVB TARGET,PRESTRK					
3393	012634	005737	012660			TST SEQUEN					
3394	012640	001410				BEQ LIMTRK					
3395	012642	100446				BMI SEQ1					
3396	012644	000463				BR SEQ2					
3397											
3398	012646	000000			TRKCNTR:	0					
3399	012650	000000			TARGET:	0					
3400	012652	000000			PRESTRK:	0					
3401	012654	000000			X00:	0					
3402	012656	000000			XID:	0					
3403	012660	000000			SEQUEN:	0					
3404											

;*****

;IS THIS TEST 26 SPECIAL SEQUENCE
 ;YES, DEC FROM TRACK 12 TO 0
 ;CLEAR FIRST USED BITS
 ;TEST CONTENTS OF ID 00 FOR 0
 ;SEQUENCE WILL BE FROM "HOME"-52-53-114-0
 ;LIMITS WERE SELECTED, INC FROM 00 TO ID.
 ;INIT 00 AS PRESENT TRACK
 ;INIT WORKING ID AND 00 LOCATIONS

;SET UP NUMBER OF TRACK MOVEMENTS

;SET FIRST TIME BITS IN ID,00

;SET TRACK COUNTER

;SET THE TRACK COUNTER

;RESET TO PRESENT TRACK
 ;IS THIS THE LIMITED SEQUENCE
 ;YES, DOING ONLY 0-52-53-114
 ;NO, SEQUENCE IS BETWEEN SELECTED LIMITS
 ;NO, THIS IS TEST 26 DEC SEQUENCE

```

3405 ;*****
3406
3407 ;LIMITED SEQUENCE, ACCESS TRACKS 52 TO 53 TO 114 BACK TO 0
3408 LIMTRK: TST 00 ;TEST HIGH ORDER FIRST TIME BIT
3409 BPL 1$ ;NOT SET, ON TRACK 52
3410 MOV #52,TARGET ;GO TO TRACK 52
3411 BIC #BIT15,00 ;CLEAR FIRST TIME BIT
3412 RTS PC
3413 1$: TSTB 00 ;TEST LOW ORDER FIRST TIME BIT
3414 BPL 2$ ;NOT SET, ON TRACK 53
3415 MOV #53,TARGET ;GO TO TRACK 53
3416 BIC #BIT7,00
3417 RTS PC
3418 2$: CMP TARGET,#114 ;IS IT ON TRACK 114
3419 BEQ 3$ ;YES,GO TO TRACK 0
3420 MOV #114,TARGET ;NO, GO TO TRACK 114
3421 RTS PC
3422 3$: CLR TARGET ;GO TO TRACK 0
3423 RTS PC
3424 ;*****
3425
3426 ; INCREMENT FROM 00+1 TO 10 AND RETURN TO 00
3427 ; USED WHEN TRACK LIMITS ARE SELECTED
3428
3429 SEQ1: BIC #100200,00 ;CLEAR FIRST TIME BITS
3430 CMPB X10,PRESTRK ;PRESENT TRACK EQUAL TO 10
3431 BNE 1$ ;NO GET NEW TRACK
3432 MOVB 00,TARGET ;YES RETURN TO 00
3433 RTS PC
3434 1$: INC TARGET ;ADD 1 TO TARGET TRACK
3435 RTS PC
3436 ;*****
3437
3438 ;DECREMENT FROM 10 = 12 TO 00 = 0
3439 ;USED IN TEST 26 ONLY
3440
3441 SEQ2: TST 00 ;FIRST TIME BIT SET
3442 BPL 1$ ;NO GET NEXT TRACK
3443 BIC #100200,00 ;YES CLEAR FIRST TIME BITS
3444 MOV #12,TARGET ;MOVE OUT 10 TRACKS
3445 RTS PC
3446 1$: DEC TARGET ;MOVE TO NEXT TRACK
3447 RTS PC
3448
3449

```

.SBTTL SECTOR SELECTION

;SECTOR INITIALIZATION AND SELECTION

3450											
3451											
3452											
3453											
3454	013046	005737	001202			INITSECTOR:	TST FIRST			;TEST FIRST AND LAST FOR 0	
3455	013052	001005					BNE 1\$			;SECTORS SPECIFIED USE THEM	
3456	013054	005237	001202				INC FIRST			;NONE SPECIFIED SET FIRST TO 1	
3457	013060	112737	000032	001203			MOVB #32, LAST			;SET LAST TO MAXIMUM	
3458	013066	113737	001203	013136		1\$:	MOVB LAST, SECCNTR			;SET UP SECTOR COUNTER	
3459	013074	163737	001202	013136			SUB FIRST, SECCNTR				
3460	013102	005237	013136				INC SECCNTR				
3461	013106	105037	013137				CLRB SECCNTR+1				
3462	013112	113737	001202	013140			MOVB FIRST, TSECTOR			;PUT FIRST SECTOR IN TARGET SECTOR	
3463	013120	162737	000003	013140			SUB #3, TSECTOR			;SUB 3 FROM TSECTOR AS FIRST TIME THROUGH	
3464										;IT GETS ADDED BACK ON.	
3465	013126	012737	000001	013142			MOV #1, INTLEAV			;SET INTERLEAVE OFFSET	
3466	013134	000207					RTS PC				
3467											
3468	013136	000000				SECCNTR:	0				
3469	013140	000000				TSECTOR:	0				
3470	013142	000000				INTLEAV:	0				
3471											
3472	013144	042737	000200	013140		GETSECTOR:	BIC #200, TSECTOR			;CLEAR THE UNIT BIT BEFORE TESTING	
3473	013152	062737	000003	013140			ADD #3, TSECTOR			;ADD 3 FOR INTERLEAVING	
3474	013160	123737	001203	013140			CMPB LAST, TSECTOR				
3475	013166	002010					BGE 1\$			;NEW SECTOR IS WITHIN LIMITS	
3476	013170	113737	001202	013140			MOVB FIRST, TSECTOR			;RESET TARGET SECTOR TO INTERLEAVE	
3477	013176	063737	013142	013140			ADD INTLEAV, TSECTOR			;ADD ON INTERLEAVE OFFSET VALUE	
3478	013204	005237	013142				INC INTLEAV			;UP DATE THE OFFSET VALUE	
3479	013210	032737	000020	012446		1\$:	BIT #BIT4, UNITSEL			;IS THIS UNIT 0	
3480	013216	001403					BEQ 2\$				
3481	013220	052737	000200	013140			BIS #BIT7, TSECTOR			;NO, SET UNIT IDENTIFIER IN TARGET SECTOR	
3482	013226	000207				2\$:	RTS PC				

```

3493 ;*****
3494
3495 .SBTTL TYPE ROUTINE
3496
3497 ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
3498 ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
3499 ;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
3500 ;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
3501 ;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
3502
3503 ;*CALL:
3504 ;*1) USING A TRAP INSTRUCTION
3505 ;*      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
3506 ;*OR
3507 ;*      TYPE
3508 ;*      MESADR
3509 ;*
3510
3511 $TYPE:  TSTB      $TPFLG      ;; IS THERE A TERMINAL?
3512         BPL       1$          ;; BR IF YES
3513         HALT      1$          ;; HALT HERE IF NO TERMINAL
3514         BR        3$          ;; LEAVE
3515 1$:     MOV       R0,-(SP)      ;; SAVE R0
3516         MOV       22(SP),R0    ;; GET ADDRESS OF ASCIZ STRING
3517 2$:     MOVB      (R0)+,-(SP)   ;; PUSH CHARACTER TO BE TYPED ONTO STACK
3518         BNE       4$          ;; BR IF IT ISN'T THE TERMINATOR
3519         TST       (SP)+        ;; IF TERMINATOR POP IT OFF THE STACK
3520 60$:    MOV       (SP)+,R0      ;; RESTORE R0
3521 3$:     ADD       #2,(SP)       ;; ADJUST RETURN PC
3522         RTI                    ;; RETURN
3523 4$:     CMPB      #THT,(SP)     ;; BRANCH IF <HT>
3524         BEQ       8$          ;;
3525         CMPB      #TCRLF,(SP)  ;; BRANCH IF NOT <CRLF>
3526         BNE       5$          ;;
3527         TST       (SP)+        ;; POP <CR><LF> EQUIV
3528         TYPE      $CRLF        ;; TYPE A CR AND LF
3529         BR        2$          ;; GET NEXT CHARACTER
3530 5$:     JSR       PC,$TYPEPC    ;; GO TYPE THIS CHARACTER
3531 6$:     CMPB      $FILLC,(SP)+  ;; IS IT TIME FOR FILLER CHARS.?
3532         BNE       2$          ;; IF NO GO GET NEXT CHAR.
3533         MOV       $NULL,-(SP)  ;; GET # OF FILLER CHARS. NEEDED
3534         AND       $NULL,PC      ;; AND THE NULL CHAR.
3535         DOES      A NULL NEED TO BE TYPED?
3536         BR IF NO--GO POP THE NULL OFF OF STACK
3537         GO TYPE A NULL
3538         DO NOT COUNT AS A COUNT
3539         LOOP
3540
3541 ;HORIZONTAL TAB PROCESSOR
3542
3543 8$:     MOVB      #40,(SP)      ;; REPLACE TAB WITH SPACE
3544 9$:     JSR       PC,$TYPEPC    ;; TYPE A SPACE
3545         BITB      #7,$CHARCNT  ;; BRANCH IF NOT AT
3546
3547 013230 105737 013453
3548 013234 100002
3549 013236 000000
3550 013240 000407
3551 013242 010046
3552 013244 017600 000002
3553 013250 112046
3554 013252 001005
3555 013254 005726
3556 013256 012600
3557 013260 062716 000002
3558 013264 000002
3559 013266 122716 000011
3560 013272 001426
3561 013274 122716 000200
3562 013300 001004
3563 013302 005726
3564 013304 104400
3565 013306 013455
3566 013310 000757
3567 013312 004737 013374
3568 013316 123726 013452
3569 013322 001352
3570 013324 013746 013450
3571
3572 013330 105366 000001
3573 013334 002770
3574 013336 004737 013374
3575 013342 105337 013440
3576 013346 000770
3577
3578 013350 112716 000040
3579 013354 004737 013374
3580 013360 132737 000007 013440

```

3537	013366	001372			BNE	9\$	;; TAB STOP
3538	013370	005726			TST	(SP)+	;; POP SPACE OFF STACK
3539	013372	000726			BR	2\$	;; GET NEXT CHARACTER
3540	013374	105777	000044		STPEC: TSTB	2\$TPS	;; WAIT UNTIL PRINTER IS READY
3541	013400	100375			BPL	STPEC	
3542	013402	116677	000002	000036	NOVB	2(SP), 2\$TPB	;; LOAD CHAR TO BE TYPED INTO DATA REG.
3543	013410	122766	000015	000002	CMPB	#15, 2(SP)	;; BRANCH IF
3544	013416	001003			BNE	1\$	;; NOT <CR>
3545	013420	105037	013440		CLRB	\$CHARCNT	
3546	013424	000406			BR	STPEX	;; EXIT
3547	013426	2766	000012	000002	1\$: CMPB	#12, 2(SP)	;; BRANCH IF
3548	013434	002002			BGE	STPEX	;; <LF>
3549	013436	105227			INCB	(PC)+	;; INC SPACE
3550	013440	000000			\$CHARCNT: .WORD	0	;; COUNT
3551	013442	000207			STPEX: RTS	PC	
3552					;; EQUATES		
3553		000011			TAT=11		
3554		000200			TCRLF=200		
3555							
3556	013444	177564			STPS: .WORD	177564	;; TTY PRINTER STATUS REG. ADDRESS
3557	013446	177566			STPB: .WORD	177566	;; TTY PRINTER BUFFER REG. ADDRESS
3558	013450	000			\$NULL: .BYTE	0	;; CONTAINS NULL CHARACTER FOR FILLS
3559	013451	002			\$FILLS: .BYTE	2	;; CONTAINS # OF FILLER CHARACTERS REQUIRED
3560	013452	012			\$FILLC: .BYTE	12	;; INSERT FILL CHARS. AFTER A "LINE FEED"
3561	013453	000			\$TPFLG: .BYTE	0	;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
3562	013454	077			\$QUES: .ASCII	"?"	;; QUESTION MARK
3563	013455	015	000		\$CRLF: .ASCII	<15>	;; CARRIAGE RETURN
3564	013457	012	000		\$LF: .ASCII	<12>	;; LINEFEED
3565		013462			.EVEN		

```

3566 ;*****
3567
3568 .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
3569
3570 *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
3571 *OCTAL (ASCII) NUMBER AND TYPE IT.
3572 *STYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
3573 *CALL:
3574 *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
3575 *      TYPOS      CALL FOR TYPEOUT
3576 *      .BYTE      N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
3577 *      .BYTE      M      ;;M=1 OR 0
3578 *                               ;;1=TYPE LEADING ZEROS
3579 *                               ;;0=SUPPRESS LEADING ZEROS
3580
3581 *STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
3582 *STYPOS OR STYPOC
3583 *CALL:
3584 *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
3585 *      TYPON      CALL FOR TYPEOUT
3586
3587 *STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
3588 *CALL:
3589 *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
3590 *      TYPOC      CALL FOR TYPEOUT
3591
3592 013462 017646 000000 013705 STYPOS: MOV      2(SP),-(SP)      ;;PICKUP THE MODE
3593 013466 116637 000001 013705 MOV      1(SP),SOFILL      ;;LOAD ZERO FILL SWITCH
3594 013474 112637 013707 013705 MOV      (SP)+,SOMODE+1      ;;NUMBER OF DIGITS TO TYPE
3595 013500 062716 000002 013705 ADD      #2,(SP)      ;;ADJUST RETURN ADDRESS
3596 013504 000406 013705 BR      STYPON
3597 013506 112737 000001 013705 STYPOC: MOV      #1,SOFILL      ;;SET THE ZERO FILL SWITCH
3598 013514 112737 000006 013707 MOV      #6,SOMODE+1      ;;SET FOR SIX(6) DIGITS
3599 013522 112737 000005 013704 STYPON: MOV      #5,SOCNT      ;;SET THE ITERATION COUNT
3600 013530 010346 013705 MOV      R3,-(SP)      ;;SAVE R3
3601 013532 010446 013705 MOV      R4,-(SP)      ;;SAVE R4
3602 013534 010546 013705 MOV      R5,-(SP)      ;;SAVE R5
3603 013536 113704 013707 MOV      SOMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
3604 013542 005404 013707 NEG      R4
3605 013544 062704 000006 013706 ADD      #6,R4      ;;SUBTRACT IT FOR MAX. ALLOWED
3606 013550 110437 013706 MOV      R4,SOMODE      ;;SAVE IT FOR USE
3607 013554 113704 013705 MOV      SOFILL,R4      ;;GET THE ZERO FILL SWITCH
3608 013560 016605 000012 013705 MOV      12(SP),R5      ;;PICKUP THE INPUT NUMBER
3609 013564 005003 013705 CLR      R3      ;;CLEAR THE OUTPUT WORD
3610 013566 006105 013705 1$: ROL      R5      ;;ROTATE MSB INTO "C"
3611 013570 000404 013705 BR      3$      ;;GO DO MSB
3612 013572 006105 013705 2$: ROL      R5      ;;FORM THIS DIGIT
3613 013574 006105 013705 ROL      R5
3614 013576 006105 013705 ROL      R5
3615 013600 010503 013705 MOV      R5,R3
3616 013602 006103 013706 3$: ROL      R3      ;;GET LSB OF THIS DIGIT
3617 013604 105337 013706 DECB      SOMODE      ;;TYPE THIS DIGIT?
3618 013610 100016 013706 BPL      7$      ;;BR IF NO
3619 013612 042703 177770 013706 BIC      #177770,R3      ;;GET RID OF JUNK

```

3620	013616	001002		BNE	4\$	::TEST FOR 0
3621	013620	005704		TST	R4	::SUPPRESS THIS 0?
3622	013622	001403		BEQ	5\$	::BR IF YES
3623	013624	005204		INC	R4	::DON'T SUPPRESS ANYMORE 0'S
3624	013626	052703	000060	BIS	#'0,R3	::MAKE THIS DIGIT ASCII
3625	013632	052703	000040	BIS	#',R3	::MAKE ASCII IF NOT ALREADY
3626	013636	110337	013702	MOVB	R3,8\$	::SAVE FOR TYPING
3627	013642	104400	013702	TYPE	8\$	::GO TYPE THIS DIGIT
3628	013646	105337	013704	DECB	\$OCNT	::COUNT BY 1
3629	013652	003347		BGT	2\$	::BR IF MORE TO DO
3630	013654	002402		BLT	6\$	::BR IF DONE
3631	013656	005204		INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
3632	013660	000744		BR	2\$	::GO DO THE LAST DIGIT
3633	013662	012605		MOV	(SP)+,R5	::RESTORE R5
3634	013664	012604		MOV	(SP)+,R4	::RESTORE R4
3635	013666	012603		MOV	(SP)+,R3	::RESTORE R3
3636	013670	016666	000002 000004	MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
3637	013676	012616		MOV	(SP)+,(SP)	
3638	013700	000002		RTI		::RETURN
3639	013702	000		.BYTE	0	::STORAGE FOR ASCII DIGIT
3640	013703	000		.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
3641	013704	000		.BYTE	0	::OCTAL DIGIT COUNTER
3642	013705	000		.BYTE	0	::ZERO FILL SWITCH
3643	013706	000000		.WORD	0	::NUMBER OF DIGITS TO TYPE

```

3644 ;*****
3645
3646 .SBTTL SAVE AND RESTORE RO-R5 ROUTINES
3647
3648 ;*SAVE RO-R5
3649 ;*CALL:
3650 ;* SAVREG
3651 ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
3652 ;*
3653 ;*TOP---(+16)
3654 ;* +2---(+18)
3655 ;* +4---R5
3656 ;* +6---R4
3657 ;* +8---R3
3658 ;*+10---R2
3659 ;*+12---R1
3660 ;*+14---R0
3661
3662 $SAVREG:
3663 013710 MOV R0,-(SP) ;: PUSH R0 ON STACK
3664 013712 010046 MOV R1,-(SP) ;: PUSH R1 ON STACK
3665 013714 010246 MOV R2,-(SP) ;: PUSH R2 ON STACK
3666 013716 010346 MOV R3,-(SP) ;: PUSH R3 ON STACK
3667 013720 010446 MOV R4,-(SP) ;: PUSH R4 ON STACK
3668 013722 010546 MOV R5,-(SP) ;: PUSH R5 ON STACK
3669 013724 016646 000022 MOV 22(SP),-(SP) ;: SAVE PS OF MAIN FLOW
3670 013730 016646 000022 MOV 22(SP),-(SP) ;: SAVE PC OF MAIN FLOW
3671 013734 016646 000022 MOV 22(SP),-(SP) ;: SAVE PS OF CALL
3672 013740 016646 000022 MOV 22(SP),-(SP) ;: SAVE PC OF CALL
3673 013744 000002 RTI
3674
3675 ;*RESTORE RO-R5
3676 ;*CALL:
3677 ;* RESREG
3678 $RESREG:
3679 013746 012666 000022 MOV (SP)+,22(SP) ;: RESTORE PC OF CALL
3680 013752 012666 000022 MOV (SP)+,22(SP) ;: RESTORE PS OF CALL
3681 013756 012666 000022 MOV (SP)+,22(SP) ;: RESTORE PC OF MAIN FLOW
3682 013762 012666 000022 MOV (SP)+,22(SP) ;: RESTORE PS OF MAIN FLOW
3683 013766 012605 MOV (SP)+,R5 ;: POP STACK INTO R5
3684 013770 012604 MOV (SP)+,R4 ;: POP STACK INTO R4
3685 013772 012603 MOV (SP)+,R3 ;: POP STACK INTO R3
3686 013774 012602 MOV (SP)+,R2 ;: POP STACK INTO R2
3687 013776 012601 MOV (SP)+,R1 ;: POP STACK INTO R1
3688 014000 012600 MOV (SP)+,R0 ;: POP STACK INTO R0
3689 014002 000002 RTI

```

```

3690 ;*****
3691
3692 .SBTTL TRAP DECODER
3693
3694 ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
3695 ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3696 ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3697 ;*GO TO THAT ROUTINE.
3698
3699 014004 010046          STRAP:  MOV      RO, -(SP)          ;; SAVE RO
3700 014006 016600 000002  MOV      2(SP), RO          ;; GET TRAP ADDRESS
3701 014012 005740          TST      -(RO)          ;; BACKUP BY 2
3702 014014 111000          MOVB     (RO), RO          ;; GET RIGHT BYTE OF TRAP
3703 014016 006300          ASL      RO          ;; POSITION FOR INDEXING
3704 014020 016000 014026  MOV      $TRAPD(RO), RO      ;; INDEX TO TABLE
3705 014024 000200          RTS      RO          ;; GO TO ROUTINE
3706
3707
3708 .SBTTL TRAP TABLE
3709
3710 ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3711 ;*BY THE "TRAP" INSTRUCTION.
3712
3713 ; ROUTINE
3714 ; -----
3715 014026          $TRPAD:  $TYPE      ;; CALL=TYPE      TRAP+0(104400)  TTY TYPEOUT ROUTINE
3716 014026 013230  $TYPOC      ;; CALL=TYPOC      TRAP+1(104401)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3717 014030 013506  $TYPOS      ;; CALL=TYPOS      TRAP+2(104402)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
3718 014032 013462  $TYPON      ;; CALL=TYPON      TRAP+3(104403)  TYPE OCTAL NUMBER (AS PER LAST CALL)
3719 014034 013522  $SAVREG     ;; CALL=SAVREG      TRAP+4(104404)  SAVE RO-R5 ROUTINE
3720 014036 013710  $RESREG     ;; CALL=RESREG      TRAP+5(104405)  RESTORE RO-R5 ROUTINE
3721 014040 013746  XSUBSCOPE   ;; CALL=SUBSCOPE TRAP+6(104406)
3722 014042 006336

```

```

3723 ;*****
3724
3725 .SBTTL POWER DOWN AND UP ROUTINES
3726
3727 :POWER DOWN ROUTINE
3728 014044 012737 014172 000024 $PWRDN: MOV $SILLUP,2#PWRVEC ;;SET FOR FAST UP
3729 014052 012737 000340 000026 MOV #340,2#PWRVEC+2 ;;PRIO:7
3730 014060 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
3731 014062 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
3732 014064 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
3733 014066 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
3734 014070 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
3735 014072 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
3736 014074 010637 014176 MOV SP,$SAVR6 ;;SAVE SP
3737 014100 012737 014112 000024 MOV $PWRUP,2#PWRVEC ;;SET UP VECTOR
3738 014106 000000 HALT
3739 014110 000776 BR .-2 ;;HANG UP
3740
3741 :POWER UP ROUTINE
3742 014112 013706 014176 $PWRUP: MOV $SAVR6,SP ;;GET SP
3743 014116 005037 014176 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
3744 014122 005237 014176 IS: INC $SAVR6 ;;WAIT FOR THE INC
3745 014126 001375 BNE IS OF WORD
3746 014130 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
3747 014132 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
3748 014134 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
3749 014136 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
3750 014140 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
3751 014142 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
3752 014144 012737 014044 000024 MOV $PWRDN,2#PWRVEC ;;SET UP THE POWER DOWN VECTOR
3753 014152 012737 000340 000026 MOV #340,2#PWRVEC+2 ;;PRIO:7
3754 014160 104400 TYPE ;;REPORT THE POWER FAILURE
3755 014162 014200 $PWRMG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
3756 014164 012716 MOV (PC)+,(SP) ;;RESTART AT RESTART
3757 014166 001306 $PWRAD: .WORD RESTART ;;RESTART ADDRESS
3758 014170 000002 RTI
3759 014172 000000 $SILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
3760 014174 000776 BR .-2 ;;BEFORE THE POWER DOWN WAS COMPLETE
3761 014176 000000 $SAVR6: 0 ;;PUT THE SP HERE
3762 014200 005015 047520 042527 $POWER: .ASCIZ (15)(12)"POWER"
3763 014206 000122
3764 .EVEN

```

```

3765 ;*****
3766
3767 .SBTTL SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE
3768
3769 ;*THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
3770 ;*UNSIGNED DECIMAL ASCIZ NUMBER.
3771 ;*CALL
3772 ;*      MOV      NUMBER, -(SP)      ;; PUT BINARY NUMBER ON THE STACK
3773 ;*      JSR      PC, 2*SSB20      ;; CALL
3774 ;*      RETURN    ;; ADDRESS OF THE 1ST ASCIZ CHAR. IS ON THE STACK
3775
3776
3777 014210 016637 000002 014234 $SB20: MOV      2(SP), 1$      ;; SAVE BINARY NUMBER
3778 014216 012746 014234      MOV      1$, -(SP)      ;; SET POINTER
3779 014222 004737 014240      JSR      PC, 2*$DB20      ;; CALL DOUBLE LENGTH CONVERT
3780 014226 012666 000002      MOV      (SP)+, 2(SP)      ;; PICKUP POINTER
3781 014232 000207      RTS      PC      ;; RETURN
3782 014234 000000 000000 1$: .WORD 0,0
3783 ;*****
3784
3785 .SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
3786
3787 ;*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
3788 ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
3789 ;*POSITIVE.
3790 ;*CALL
3791 ;*      MOV      $PNTR, -(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
3792 ;*      JSR      PC, 2*$DB20
3793 ;*      RETURN    ;; THE FIRST ADDRESS OF ASCII
3794 ;*                  ;; IS ON THE STACK
3795
3796
3797 014240 104404 $DB20: SAVREG      ;; SAVE REGISTERS
3798 014242 016602      MOV      2(SP), R2      ;; PICKUP THE DATA POINTER
3799 014246 012700 014420      MOV      $SDECVL, R0      ;; GET ADDRESS OF "SDECVL" STRING
3800 014252 010066 000002      MOV      R0, 2(SP)      ;; PUT ADDRESS OF ASCII STRING ON STACK
3801 014256 012201      MOV      (R2)+, R1      ;; PICKUP THE BINARY NUMBER
3802 014260 012202      MOV      (R2)+, R2
3803 014262 012737 000012 014336      MOV      #10, 4$      ;; SET UP TO DO 10 CONVERSIONS
3804 014270 012704 014350      MOV      $STNPNR, R4      ;; ADDRESS OF TEN POWER
3805 014274 012705 014352      MOV      $STNPNR+2, R5
3806 014300 005003 1$: CLR      R3      ;; CLEAR PARTIAL
3807 014302 161401 2$: SUB      (R4), R1      ;; SUBTRACT TEN POWER
3808 014304 005602      SBC      R2
3809 014306 161502      SUB      (R5), R2
3810 014310 002402      BLT      3$      ;; BR IF TEN POWER TOO LARGE
3811 014312 005203      INC      R3      ;; ADD 1 TO PARTIAL
3812 014314 000772      BR      2$      ;; LOOP
3813 014316 062401 3$: ADD      (R4)+, R1      ;; RESTORE SUBTRACTED VALUE
3814 014320 005502      ADC      R2
3815 014322 062402      ADD      (R4)+, R2
3816 014324 022525      CMP      (R5)+, (R5)+      ;; MOVE TO NEXT TEN POWER
3817 014326 052703 000060      BIS      #0, R3      ;; CHANGE PARTIAL TO ASCII
3818 014332 110320      MOVB     R3, (R0)+      ;; SAVE IT

```

MAINDEC-11-DZRXB-D MACY11 27(663) 6-NOV-75 08:50 PAGE 90
 DZRXBD.P11 DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

SEQ 0171

3819	014334	005327	DEC	(PC)+	;;DONE?
3820	014336	000000	4S: .WORD	0	
3821	014340	001357	BNE	1S	;;BR IF NO
3822	014342	105020	CLRB	(RD)+	;;TERMINATOR
3823	014344	104405	RESREG		;;RESTORE REGISTERS
3824	014346	000207	RTS	PC	;;RETURN
3825	014350	145000	\$TNPWR: 145000		;;1.0E09
3826	014352	035632	35632		
3827	014354	160400	160400		;;1.0E08
3828	014356	002765	2765		
3829	014360	113200	113200		;;1.0E07
3830	014362	000230	230		
3831	014364	041100	041100		;;1.0E06
3832	014366	000017	17		
3833	014370	103240	103240		;;1.0E05
3834	014372	000001	1		
3835	014374	023420	23420		;;1.0E04
3836	014376	000000	0		
3837	014400	001750	1750		;;1.0E03
3838	014402	000000	0		
3839	014404	000144	144		;;1.0E02
3840	014406	000000	0		
3841	014410	000012	12		;;1.0E01
3842	014412	000000	0		
3843	014414	000001	1		;;1.0E00
3844	014416	000000	0		
3845	014420	000014	\$DECVL: .BLKB 12.		;;RESERVE STORAGE FOR ASCII STRING
3846					

```

3847 ;*****
3848
3849 ;TYPE NUMERICAL ASCII STRING,RIGHT JUSTIFIED
3850 ;REPLACING LEADING ZEROS WITH SPACES.
3851 ;FIRST ADDRESS OF ASCII STRING MUST BE ON TOP OF THE STACK
3852
3853
3854 014434 010046 RTJUST: MOV RO,-(SP) ;SAVE RO
3855 014436 016600 000004 MOV 4(SP),RO ;PICK UP ADDRESS OF ASCII STRING
3856 014442 010037 014474 MOV RO,3$ ;SAVE ADDRESS FOR TYPE OUT
3857 014446 105710 1$: TSTB (RO) ;IS THIS THE TERMINATOR
3858 014450 001406 BEQ 2$ ;IF YES TYPE IT OUT
3859 014452 122710 000060 CMPB #'0,(RO) ;IS IT A ZERO
3860 014456 001005 BNE 4$ ;IF NO GO PRINT IT
3861 014460 112720 000040 MOVB #'',(RO)+ ;IF YES REPLACE IT WITH A SPACE
3862 014464 000770 BR 1$ ;TEST NEXT CHAR.
3863 014466 112740 000060 2$: MOVB #'0,-(RO) ;STRING OFF ALL ZEROS,PUT BACK THE LAST ONE
3864 014472 104400 4$: TYPE ;TYPE THE STRING
3865 014474 000000 3$: OPEN
3866 014476 012600 MOV (SP)+,RO ;RESTORE RO
3867 014500 012616 MOV (SP)+,(SP) ;RESTORE THE STACK
3868 014502 000207 RTS PC ;RETURN
3869
3870 ;TYPES 16 BIT WORD IN DECIMAL
3871
3872 014504 012546 SGLDEC: MOV (RS)+,-(SP) ;PUT NUMBER TO BE TYPED ON STACK
3873 014506 004737 014210 JSR PC,@#5$B2D ;CONVERT NUMBER TO DECIMAL
3874 014512 062716 000005 ADD #5,(SP) ;MOVE ADDRESS OF ASCII STRING OVER BY 5 BYTES
3875 ;TO TYPE SINGLE DECIMAL NUMBER
3876 014516 004737 014434 JSR PC,RTJUST ;TYPE THE DECIMAL NUMBER
3877 014522 000205 RTS RS

```

MAINDEC-11-DZRXB-D
DZRXBD.P11 ME

MACY11 27(663)

6-NOV-75 08:50 PAGE 92

MESSAGES

.SBTTL MESSAGES

3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931

014524	042524	052123	050040
014532	051101	046501	052105
014540	051105	035123	000040
014546	005015	042412	040522
014554	051104	020040	040506
014562	052123	020040	043040
014570	050101	020124	020040
014576	020040	020040	020040
014604	043440	047517	020104
014612	020040	040502	020104
014620	020040	020040	040520
014626	051523	000	
014631	125	042516	050130
014636	041505	042524	020104
014644	020104	020104	040515
014652	045522	005015	000
014657	104	042040	046440
014664	051101	020113	044515
014672	051523	047111	006507
014700	000012		
014702	040504	040524	020054
014710	047516	051440	040524
014716	052524	020123	051105
014724	047522	000122	
014730	047440	020116	051124
014736	041501	000113	
014742	020040	051106	046517
014750	052040	040522	045503
014756	000		
014757	040	020057	042523
014764	052103	051117	000
014771	015	020012	054502
014776	042524	020040	040502
015004	020104	043440	047517
015012	006504	000012	
015016	020040	020040	020040
015024	000		
015025	040	000040	
015030	005015	000	

```

MDTESTP:      .ASCIZ "TEST PARAMETERS: "

MXEHEADER:    .ASCIZ <15><12><12> "ERADR  FAST  FAPT

MUNXDO:      .ASCIZ "UNEXPECTED D D MARK"<15><12>

MDDMIS:      .ASCIZ "D D MARK MISSING"<15><12>

MDERHDR:     .ASCIZ "DATA, NO STATUS ERROR"

MTRK:        .ASCIZ " ON TRACK"

MPRES:       .ASCIZ "  FROM TRACK"

MSECT:       .ASCIZ " / SECTOR"

MCOLMUN:     .ASCIZ <15><12>" BYTE  BAD  GOOD"<15><12>

TAB:         .ASCIZ <40><40><40><40><40><40>

DBLSP:       .ASCIZ <40><40>

MCRLF:       .ASCIZ <15><12>

```

G14

 MAINDEC-11-DZRXB-D MACY11 27(663) 6-NOV-75 08:50 PAGE 93
 DZRXBD.P11 MESSAGES

SEQ 0174

3932	015033	015	042412	051122	MERHEADER:	.ASCIZ <15><12>"ERROR CONDITIONS: TEST PC = "
3933	015040	051117	041440	047117		
3934	015046	044504	044524	047117		
3935	015054	035123	020040	042524		
3936	015062	052123	050040	020103		
3937	015070	020075	000			
3938						
3939	015073	125	044516	020124	MUNIT0:	.ASCIZ "UNIT 0 "
3940	015100	020060	000			
3941						
3942	015103	125	044516	020124	MUNIT1:	.ASCIZ "UNIT 1 "
3943	015110	020061	000			
3944						
3945	015113	122	041530	020123	MRXCS:	.ASCIZ "RXCS = "
3946	015120	020075	000			
3947						
3948	015123	123	040524	052524	MASTAT:	.ASCIZ "STATUS A = "
3949	015130	020123	020101	020075		
3950	015136	000				
3951						
3952	015137	123	040524	052524	MBSTAT:	.ASCIZ "STATUS B = "
3953	015144	020123	020102	020075		
3954	015152	000				
3955						
3956	015153	015	005012	000	DBLLF:	.ASCIZ <15><12><12>
3957						
3958	015157	116	020117	047111	MINTER:	.ASCIZ "NO INTERRUPT AT DONE ERROR"
3959	015164	042524	051122	050125		
3960	015172	020124	052101	042040		
3961	015200	047117	020105	051105		
3962	015206	047522	000122			
3963						
3964	015212	047125	047113	053517	MUKNINT:	.ASCIZ "UNKNOWN INTERRUPT"
3965	015220	020116	047111	042524		
3966	015226	051122	050125	000124		
3967						
3968	015234	047524	040524	020114	MERCT:	.ASCIZ "TOTAL READ CHECK ERRORS = "
3969	015242	042522	042101	041440		
3970	015250	042510	045503	042440		
3971	015256	051122	051117	020123		
3972	015264	020075	000			
3973						
3974	015267	106	046111	041114	MFIL:	.ASCIZ "FILLBUFFER "
3975	015274	043125	042506	020122		
3976	015302	000				
3977						
3978	015303	105	050115	054524	MEMPTY:	.ASCIZ "EMPTYBUFFER "
3979	015310	052502	043106	051105		
3980	015316	000040				
3981						
3982	015320	052040	040522	045503	MLIMTRK:	.ASCIZ " TRACKS 52,53,114,0 "
3983	015326	020123	031065	032454		
3984	015334	026063	030461	026064		
3985	015342	020060	000040			

H14

 MAINDEC-11-DZRXB-D
 DZRXBD.P11

 MACY11 27(663)
 MESSAGES

6-NOV-75 08:50 PAGE 94

SEQ 0175

3986						
3987	015346	042117	000075		MOD:	.ASCIZ "00="
3988						
3989	015352	020040	042111	000075	MID:	.ASCIZ " ID="
3990						
3991	015360	020040	044506	051522	MFIRST:	.ASCIZ " FIRST="
3992	015366	036524	000			
3993						
3994	015371	040	046040	051501	MLAST:	.ASCIZ " LAST="
3995	015376	036524	000			
3996						
3997	015401	103	041522	042440	MBADCRC:	.ASCIZ "CRC ERROR NO DATA ERROR"
3998	015406	051122	051117	047040		
3999	015414	020117	040504	040524		
4000	015422	042440	051122	051117		
4001	015430	000				
4002						
4003	015431	122	040505	020104	MREAD:	.ASCIZ "READ "
4004	015436	000				
4005						
4006	015437	104	052101	020101	MCRC:	.ASCIZ "DATA CRC ERROR"
4007	015444	051103	020103	051105		
4008	015452	047522	000122			
4009						
4010	015456	042523	045505	042440	MSEEK:	.ASCIZ "SEEK ERROR"
4011	015464	051122	051117	000		
4012						
4013	015471	127	044522	042524	MWRITE:	.ASCIZ "WRITE "
4014	015476	000040				
4015						
4016	015500	040520	044522	054524	MPAR:	.ASCIZ "PARITY ERROR"
4017	015506	042440	051122	051117		
4018	015514	000				
4019						
4020	015515	105	051122	051117	MNOFLAG:	.ASCIZ "ERROR FLAG ERROR"
4021	015522	043040	040514	020107		
4022	015530	051105	047522	000122		
4023						
4024	015536	040502	000104		MBAD:	.ASCIZ "BAD"
4025						
4026	015542	000040			SPACE:	.ASCIZ "<40>
4027						
4028	015544	047507	042117	000	MGOOD:	.ASCIZ "GOOD"
4029						
4030	015551	040	041440	042510	MSUM:	.ASCIZ " CHECK SUM "
4031	015556	045503	051440	046525		
4032	015564	000040				
4033						
4034	015566	005015	054122	030461	MRX11:	.ASCIZ "<15><12>"RX11 / RXV11"
4035	015574	027440	051040	053130		
4036	015602	030461	000			
4037						
4038	015605	015	005012	040515	MREV:	.ASCIZ "<15><12><12> "MAINDEC-11-DZRXB-D"
4039	015612	047111	042504	026503		

4040	015620	030461	042055	051132	
4041	015626	041130	042055	000	
4042					
4043	015633	015	052412	042516	LOC4M: .ASCIZ <15><12>"UNEXPECTED TRAP TO LOC. 4 OCCURRED"
4044	015640	050130	041505	042524	
4045	015646	020104	051124	050101	
4046	015654	052040	020117	047514	
4047	015662	027103	032040	047440	
4048	015670	041503	051125	042522	
4049	015676	000104			
4050					
4051	015700	005015	047125	054105	LOC10M: .ASCIZ <15><12>"UNEXPECTED TRAP TO LOC. 10 OCCURRED"
4052	015706	042520	052103	042105	
4053	015714	052040	040522	020120	
4054	015722	047524	046040	041517	
4055	015730	020056	030061	047440	
4056	015736	041503	051125	042522	
4057	015744	000104			
4058					
4059	015746	050075	000103		PCM: .ASCIZ "=PC"
4060					
4061	015752	005015	051124	041501	0028IG: .ASCII <15><12>"TRACK LIMITS SELECTED OUT OF RANGE"
4062	015760	020113	044514	044515	
4063	015766	051524	051440	046105	
4064	015774	041505	042524	020104	
4065	016002	052517	020124	073117	
4066	016010	051040	047101	042507	
4067	016016	005015	042504	040506	.ASCIZ <15><12>"DEFAULTING TO "
4068	016024	046125	044524	043516	
4069	016032	052040	020117	000	
4070					
4071	016037	015	051412	041505	S28IG: .ASCII <15><12>"SECTOR LIMITS SELECTED OUT OF RANGE"
4072	016044	047524	020122	044514	
4073	016052	044515	051524	051440	
4074	016060	046105	041505	042524	
4075	016066	020104	052517	020124	
4076	016074	043117	051040	047101	
4077	016102	042507			
4078	016104	005015	042504	040506	.ASCIZ <15><12>"DEFAULTING TO "
4079	016112	046125	044524	043516	
4080	016120	052040	020117	000	
4081					
4082	016125	015	041412	052501	D0LOAD: .ASCII <15><12>"CAUTION - IF YOU DESIRE TO TEST UNIT 0"
4083	016132	044524	047117	026440	
4084	016140	044440	020106	047531	
4085	016146	020125	042504	044523	
4086	016154	042522	052040	020117	
4087	016162	042524	052123	052440	
4088	016170	044516	020124	060	
4089	016175	015	051012	050105	.ASCII <15><12>"REPLACE LOAD MEDIUM WITH A SCRATCH DISKETTE"
4090	016202	040514	042503	046040	
4091	016210	040517	020104	042515	
4092	016216	044504	046525	053440	
4093	016224	052111	020110	020101	

J14

MAINDEC-11-DZRXB-D MACY11 27(663) 6-NOV-75 08:50 PAGE 96
DZRXB0.P11 MESSAGES

SEQ 0177

4094 016232 041523 040522 041524
4095 016240 020110 044504 045523
4096 016246 052105 042524
4097 016252 005015 044124 047105
4098 016260 050040 042522 051523
4099 016266 041440 047117 044524
4100 016274 052516 000105
4101
4102
4103
4104
4105
4106
4107
4108
4109 016300 000200
4110
4111 000001

.ASCIZ <15><12>"THEN PRESS CONTINUE"

.EVEN

;*****

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

BUFADR: .BLKB 200

.END

MAINDEC-11-DZRXB-D MACY11 27(663) 6-NOV-75 08:50 PAGE 97
DZRXBD.P11 CROSS REFERENCE TABLE

AOJSUM	010532	2260	2333	2351	2681	2945	2963#												
ASTAT	011670	2736	2839	2846	2891	2907	3068#	3071	3074	3148#	3152#	3162							
BA08YT	011210	2956*	2957	2958	3003	3048#													
BIT0 =	000001	173#	2846																
BIT00 =	000001	163#	173																
BIT01 =	000002	162#	172																
BIT02 =	000004	161#	171																
BIT03 =	000010	160#	170																
BIT04 =	000020	159#	169																
BIT05 =	000040	158#	168																
BIT06 =	000100	157#	167																
BIT07 =	000200	156#	166																
BIT08 =	000400	155#	165																
BIT09 =	001000	154#	164																
BIT1 =	000002	172#	2736	2839															
BIT10 =	002000	153#																	
BIT11 =	004000	152#																	
BIT12 =	010000	151#																	
BIT13 =	020000	150#	2796																
BIT14 =	040000	149#	430	1377	3339	3353													
BIT15 =	100000	148#	443	2937	3373	3411													
BIT2 =	000004	171#	3071																
BIT3 =	000010	170#																	
BIT4 =	000020	169#	3140	3479															
BIT5 =	000040	168#	1485	2571															
BIT6 =	000100	167#	629	783	1028	1189	1233	1247	1287	1305	1347	1366	1434	1538					
		1777	1796	1895	2891	2907	3330	3335	3341										
BIT7 =	000200	166#	632	633	1328	1437	1438	1542	2386	2392	2944	2998	3416	3481					
BIT8 =	000400	165#	518	784	806	2466	2498	2897	2909	2983	3091								
BIT9 =	001000	164#																	
BPTVEC=	000014	180#																	
BRLEV	001214	363#	2430																
BSTAT	011672	3099*	3149#	3160*	3168														
BUFA0R	016300	1570	1673	2963*	2964*	2969*	2971*	2972	3183	3285	4109#								
BUSERR	005756	208	2398#																
BYTECH	011206	2684*	2692*	2942*	2960*	2999	3015*	3047#											

DZRXBO.P11 CROSS REFERENCE TABLE

CRCR	010016	2847	2855#											
DATABY	012044	3198#	3207*	3217*	3223*	3233*	3241*	3250*	3252*	3261*	3269*	3270*	3279*	3283
		3284												
DATACK	007474	2787#	2813*	2857*	3018									
DATACR	010106	2856	2860	2875#										
DATACR	010654	2959	2983#											
DATAD	012050	3188	3207#											
DATAD	012062	3189	3217#											
DBLLF	015153	3956#												
DBLSP	015025	3002	3006	3139	3928#									
DOCHK	010162	2830	2889#											
DDERR	010222	2897#	2914											
DDISP =	177570	95#	371											
DEATH	002662	1120	1130#											
DEXIT	004106	1760#	1835	1969	2023	2075	2210	2247	2270	2338	2354	2367	2393	
DING	006272	2439	2508#	2928										
DISPLA	001220	371#	410*											
DISPRE	000174	231#	410											
DONE	012450	2674	3351#											
DONEBI=	000040	195#	2725	2822	3076									
DRVSUP	006614	2366	2648#											
DSWR =	177570	94#	370											
DTESTP	001212	361#	426	430	441	447	807							
DLOAD	016125	435	4082#											
EBIE =	000103	197#	2952											
EMPBUF	010410	2858	2939#	2979	3039	3043								
EMPDON	011046	2947	3018#											
EMPER	010624	2948	2977#											
EMPFLA	010474	2953#	2954	2961	3016									
EMTVEC=	000030	183#												
EPCSCO	006314	2456#	2457*	2517#	2549									
ERCNTR	011214	2859	2943#	2984*	2989	3021	3026	3050#						
ERMSG	010332	2895	2912	2918#	2993	3058	3103							
ERRORS	006312	519*	2458*	2516#	2522	2531*	2537	3089*						
ERRVEC=	000004	176#												
EXIT	011202	2941	3041	3044#										
FAST	002342	521*	710*	819#	2478									
FBEB	003712	1571	1580#	1586	1674	2263								
FBIE =	000101	196#	2688											
FILLBU	006746	2681#	2698	2711	2745	2765								
FILLDO	007130	2682	2714#	2741										
FILLER	007024	2683	2695#											
FILLFL	007004	2689#	2690	2693										
FIRST	001202	294#	295	481	483*	486	489	501*	3454	3456*	3459	3462	3476	
FIRSTT	002040	709#	812											
FLOATO	012072	3190	3223#											
FLOATI	012134	3191	3241#											
FUNCTI	007472	2268*	2352*	2717*	2720*	2776*	2777	2786#	2817*	3105				
GETPAT	012032	1569	2259	2332	2350	2648	2666	3183#						
GETSEC	013144	2680	2812	3472#										
GETTRA	012626	2637	2651	2669	3392#									
GETUNI	012356	2257	2349	2636	2650	2654	2668	3330#						
GNS =	***** U	3716	3717	3718	3719	3720	3721	3722						
GOODBY	011212	3007*	3009	3049#										

MAINDEC-11-DZRXB-D MACY11 27(663) 6-NOV-75 08:50 PAGE 99
DZRXB0.P11 CROSS REFERENCE TABLE

HANGER	006560	421*	2573*	2624*	2628*	2714*	2727*	2814*	2824*										
HANGPL	006562	2574*	2625*	2629*															
HERE	002444	843	849*																
HLT10	010066	2867*																	
HLT12	010134	2881*																	
HLT13	010246	2902*																	
HLT14	011034	3014*																	
HLT15	011250	3063*																	
HLT6	007106	2708*																	
HLT7	007352	2760*																	
HOME	007476	2761	2789*	2803															
HUNGUP	006440	2575	2582*																
ID =	001201	293*	460	462	470	3376													
ILLADR	004734	1852	1985	2038	2079*	2084													
INCERR	006072	2458*	2459																
INITSE	013046	2679	2811	3454*															
INITTR	012474	2635	2649	2667	3368*														
INTLEA	013142	3465*	3470*	3477	3478*														
INTSER	011256	2244	2364	2389	3064	3068*													
IOTVEC=	000020	181*																	
KRXVEC	001204	297*	964	1018	1182	1223	1265	1321	2243	2363	2388								
LAST =	001203	295*	484	486	492	3457*	3458	3474											
LIMITS	001522	452*	477																
LIMTRK	012662	3394	3408*																
LOAD	012232	3208	3228	3251	3271	3280	3283*												
LOCKUP	002366	707	831*																
LOC10M	015700	2414	4051*																
LOC4M	015633	2398	4043*																
LOGICA	002434	224	845*																
MABELL	006310	795	2510	2513*															
MASTAT	015123	3161	3948*																
MBAD	015536	3032	4024*																
MEADCR	015401	2863	3997*																
MBSTAT	015137	3167	3952*																
MCOLMU	014771	2997	3920*																
MCRC	015437	2877	4006*																
MCRLF	015030	445	451	494	789	800	2470	2705	2773	2864	2878	2899	2927	2995					
		3011	3035	3060	3081	3													

N14

 MAINDEC-11-DZRXB-D MACY11 27(663) 6-NOV-75 08:50 PAGE 100
 DZRXB0.P11 CROSS REFERENCE TABLE

SEQ 0181

MOD	015346	464	3987*															
MORETE	002032	707*	1758	1761	2604	2613												
MPAR	015500	2704	3123	4016*														
MPASS	006527	801	2620*	2923														
MPRES	014742	2768	3913*															
MREAD	015431	2842	2850	4003*														
MREV	015605	424	4038*															
MRXCS	015113	3104	3118	3945*														
MRX11	015566	3115	4034*															
MSECT	014757	3132	3917*															
MSEEK	015456	2767	4010*															
MSJM	015551	3029	4030*															
MTRK	014730	3127	3910*															
MUKNIN	015212	3080	3964*															
MUNITO	015073	3142	3939*															
MUNIT1	015103	3144	3942*															
MUNYCD	014631	2913	3894*															
MWRITE	015471	2739	2750	2756	4013*													
MX	002344	786*	794	821*	1757*	1760*												
MXEHEA	014546	2468	3885*															
EXTRO	007714	2712	2829	2832	2834*	2844	2852	2870	2885	2906								
NOERTY	011026	2986	2990	3012*														
NOINTE	011216	2729	2826	3056*														
NOMORE	002144	719	783*															
NOPRIN	006244	2464	2498*															
NOPRNT	011452	3100*																
NOSCOF	006324	2531*	2558															
NXREAD	011164	2980	3022	3037	3040*													
OD	001200	292*	293	444*	452	462	465	476*	3370*	3371	3374	3378	3382*	3408				
		3411*	3413	3416*	3430*	3433	3443	3445*										
0028IG	015752	473	4061*															
OK2GO	002664	1123	1135*															
OPEN =	000000	202*	804	2495	2703	2926	3001	3028	3146	3865								
PARTES	007054	2699	2701*	2742	2845	2981												
PARTYP	011544	3097	3118*	3158														
PASS	002340	420*	796*	802	818*	2493	2924											
PAT	012046	1561*	1748*	1753*	2258*	2329*	2331*	2346*	2348*	2362*	2375*	2387*	3185	3199*				
PATGEN	012054	3208*	3209	3218														
PATTER	012024	3187	3188*															
PAT125	012142	3192	3250*															
PAT314	012162	3193	3261*															
PC	=000007	107*	707*	840*	845*	849*	1280*	1339*	1378*	1494*	1562*	1569*	1571*	1572*				
		1580*	1583*	1585*	1654*	1669*	1674*	1675*	1754*	1755*	1772*	1791*	1852*	1908*				
		1985*	2013*	2038*	2065*	2079*	2083*	2087*	2091*	2095*	2100*	2103*	2115*	2118*				
		2124*	2127*	2133*	2188*	2246*	2257*	2259*	2260*	2263*	2269*	2271*	2272*	2278*				
		2317*	2318*	2330*	2332*	2333*	2335*	2337*	2347*	2349*	2350*	2351*	2353*	2366*				
		2391*	2444*	2499*	2511*	2567*	2576*	2627*	2635*	2636*	2637*	2638*	2641*	2648*				
		2649*	2650*	2651*	2652*	2653*	2654*	2655*	2656*	2659*	2666*	2667*	2668*	2669*				
		2670*	2671*	2674*	2675*	2679*	2680*	2681*	2721*	2732*	2751*	2757*	2761*	2764*				
		2774*	2778*	2781*	2784*	2792*	2798*	2804*	2806*	2811*	2812*	2818*	2830*	2833*				
		2836*	2851*	2858*	2895*	2898*	2912*	2915*	2928*	2929*	2938*	2945*	2973*	2993*				
		2996*	3020*	3038*	3042*	3045*	3058*	3064*	3084*	3093*	3097*	3103*	3109*	3111*				
		3113*	3125*	3145*	3154*	3158*	3171*	3208*	3228*	3251*	3271*	3278*	3280*	3287*				
		3289*	3309*	3336*	3343*	3356*	3383*	3412*	3417*	3421*	3423*	3434*	3436*	3447*				

		3449*	3466*	3482*	3521*	3528*	3535*	3549*	3551*	3756	3779*	3781*	3819*	3824*
PCM	015746	3868*	3873*	3876*										
PCONT	007124	2405	2422	4059*										
PCSCOP	006374	2697*	2710	2712*	2740*	2844*	2980*							
PIRQ =	177772	521	709*	710	2481	2532*	2559	2561*	2718	2828	2889	2919		
PIRQVE =	000240	93*												
PLOOP	007120	187*												
PRESTR	012652	2698*	2711*	2741*	2843*	2979*								
PRETES	001726	2769	2805*	3392*	3400*	3431								
PRO =	000000	495	518*											
PR1 =	000040	110*												
PR2 =	000100	111*												
PR3 =	000140	112*												
PR4 =	000200	113*												
PR5 =	000240	114*	1274											
PR6 =	000300	115*	1333											
PR7 =	000340	116*												
		117*	209	211	215	217	219	221	412	968	976	1022	1071	1186
PS =	177776	1209	1227	1236	1269	1290	1325	1350	2245	2365	2390			
PSM =	177776	90*	91											
PTYP1	007064	91*	1274*	1333*										
PURVEC =	000024	2696*	2703*	2739*	2842*	2978*								
RANDAT	0122.2	182*	3728*	3729*	3737*	3752*	3753*							
RANGEN	012262	3195	3278*	3281										
RANUM	012354	3278	3293*											
RAN1	012350	3279	3308*	3313*										
RAN2	012352	417*	3294	3300*	3305	3311*								
ROCODE	011674	418*	3295	3302	3307*	3312*								
RODOONE	007672	2798	3152*											
ROER =	000017	2815	2828*											
ROERR	007724	194*	3092	3153										
ROIE =	000107	2816	2838*											
ROOMLY	006564	199*	2817											
READ	007576	2246	2635*											
READCH	010400	2638	2811*	2938										
REBEG1	002276	2653	2656	2671	2937*									
RECAL =	040001	635	706	799	806*	2406	2423							
REREAD	007606	201*	2334	2791										
RESERR	006002	2813*	2843	2849	2871	2886	2905	3038						
RESREG =	104405	210	2414*											
RESTAR	001306	3721*	3823											
RESVEC =	000010	397	412*	3117	3757									
RETURN	010330	177*												
RTJUST	014434	2892	2908	2915*										
RTN	007574	3854*	3876											
RTSPC	006410	2790	2806*											
RXCS	001206	2565	2567*											
		299*	415	523	783*	863*	865	1028*	1142	1144	1189*	1194	1196	1233*
		1247*	1287*	1305*	1347*	1366*	1377*	1381	1484*	1485	1487	1497	1568*	1587
		1672*	1771*	1790*	1849*	1851*	1855	1907*	1910	1913	1982*	1984*	1988	2012*
		2035*	2037*	2041	2064*	2104	2114*	2120	2129	2136	2187*	2190	2193	2262*
		2274	2277*	2334*	2564	2571	2688*	2689	2725	2777*	2791*	2794	2822	2952*
		2953	3069	3076	3092*	3119	3153*							
RXD2	001210	341*	415*	416*	627	919	1432	1536	1574*	1589	1680	1775	1794	1893

		1914	1924	2002	2016	2054	2068	2086*	2094*	2121	2123*	2130	2132*	2176
		2194	2202	2265*	2276	2280	2691*	2780*	2783*	2956	3068	3095	3099	3152
		3156	3160											
RXERR0	011376	3070	3086	3089*	3090									
RXPWR	011530	3073	3115*											
RO	=X000000	98*	395*	401*	422	522*	524	626*	633*	634	864*	866	918*	920
		1141*	1142	1193*	1194	1283	1343	1380*	1382	1431*	1438*	1439	1488*	1496*
		1498	1539*	1543*	1544	1679*	1682	1774*	1778	1793*	1797	1846*	1854*	1856
		1892*	1896	1912*	1923*	1926	1978*	1987*	1989	2001*	2003	2015*	2017	2031*
		2040*	2042	2053*	2055	2067*	2069	2113*	2135*	2137	2174*	2177	2192*	2201*
		2203	2265	2315*	2430*	2435*	2437*	2438*	2439*	2440*	2441*	2442*	2487	2691
		2958	2972*	3007	3008*	3293*	3294*	3295*	3296*	3298*	3299*	3300	3301*	3302*
		3303*	3304*	3305*	3306*	3307	3308	3505	3506*	3507	3510*	3663	3688*	3699
		3700*	3701	3702*	3703*	3704*	3705*	3730	3751*	3799*	3800	3818*	3822*	3854
		3855*	3856	3857	3859	3861*	3863*	3866*						
R1	=X000001	99*	523*	524	628*	629*	632*	634	865*	866	919*	920	1144*	1196*
		1262*	1318*	1328*	1381*	1382	1433*	1434*	1437*	1439	1487*	1497*	1498	1537*
		1538*	1542*	1544	1589*	1680*	1682	1776*	1777*	1778	1795*	1796*	1797	1855*
		1856	1894*	1895*	1896	1913*	1924*	1926	1988*	1989	2002*	2003	2016*	2017
		2041*	2042	2054*	2055	2068*	2069	2104*	2120*	2129*	2136*	2137	2176*	2177
		2193*	2202*	2203	2276*	2316*	2432	2490	2831	2855	2937*	2944*	2991	2998*
		3044*	3664	3687*	3731	3750*	3801*	3807*	3813*					
R2	=X000002	100*	627*	628	964*	965	967*	968*	969*	1018*	1019	1021*	1022*	1023*
		1182*	1183	1185*	1186*	1187*	1223*	1224	1226*	1227*	1228*	1265*	1266	1268*
		1269*	1270*	1321*	1322	1324*	1325*	1326*	1432*	1433	1536*	1537	1567*	1575*
		1622	1671*	1681*	1775*	1776	1794*	1795	1845*	1893*	1894	1914*	1925*	1979*
		2000*	2032*	2052*	2081*	2099*	2097*	2112*	2121*	2130*	2175*	2194*	2200*	2243*
		2244*	2245*	2261*	2266*	2290*	2363*	2364*	2365*	2388*	2389*	2390*	2484	3225*
		3227*	3230	3665	3686*	3732	3749*	3798*	3801	3802*	3808*	3809*	3814*	3815*
R3	=X000003	101*	708	709	712	807*	808*	809	811*	3600	3609*	3615*	3616*	3619*
		3624*	3625*	3626	3635*	3666	3685*	3733	3748*	3806*	3811*	3817*	3818	
R4	=X000004	102*	1570*	1574	1673*	1679	2471*	2472*	2473	3183*	3284*	3285	3601	3603*
		3604*	3605*	3606	3607*	3621	3623*	3631*	3634*	3667	3684*	3734	3747*	3804*
		3807	3813	3815										
R5	=X000005	103*	803*	842*	845	2494*	2925*	3000*	3027*	3185*	3186*	3187	3602	3608*
		3610*	3612*	3613*	3614*	3615	3633*	3668	3683*	3735	3746*	3805*	3809	3816
		3872	3877*											
R6	=X000006	104*	106											
R7	=X000007	105*	107											
SAVREG=	104404	3720*	3797	815										
SA200	001232	237	401*											
SA202	001222	238	395*											
SCOPIN	006356	2523	2550	2557*										
SDN	006412	1378	1494	1585	1772	1791	1908	2013	2065	2083	2091	2100	2118	2127
		2133	2188	2272	2278	2335	2571*	2792	3093	3154				
SECCNT	013136	2730*	2834*	2939	3040*	343*	3459*	3460*	3461*	3468*				
SECLMT	001624	456	472	481*										
SEEKER	007324	2751	2754*	2851										
SEKRTY	007372	2745*	2763	2765*	2849*									
SEKTYP	007376	2757	2767*											
SEQUEN	012660	454*	2386*	2392*	3368	3373*	3393	3403*						
SEQ1	012760	3395	3430*											
SEQ2	013014	3396	3443*											
SQLEDC	014504	803	2494	2925	3000	3027	3872*							

6-NOV-75 08:50 PAGE 103

106*	396*	402*	403*	408	411	412*	413*	447*	465*	470*	489*	492*
711*	839*	965*	969	971*	972*	976*	977*	1019*	1023	1025*	1026*	1071*
1072*	1183*	1187	1204*	1205*	1209*	1210*	1224*	1228	1230*	1231*	1236*	1237*
1266*	1270	1272*	1276	1277	1279	1283*	1284*	1290*	1291*	1322*	1326	1331*
1335	1336	1338	1343*	1344*	1350*	1351*	1582*	2102*	2400*	2406*	2417*	2423*
2456	2471	2473*	2478*	2481*	2484*	2487*	2490*	2532	2549	2559*	2566*	2609*
2626*	2682*	2683*	2685*	2686*	2695	2715*	2716*	2722*	2723*	2735	2769*	2815*
2816*	2819*	2820*	2838	2884*	2919*	2947*	2948*	2949*	2950*	2977	3003*	3009*
3083*	3105*	3112*	3119*	3128*	3135*	3162*	3168*	3288	3355*	3505*	3506	3507*
3509	3510	3511*	3513	3515	3517	3522	3524*	3526*	3534*	3538	3542	3543
3547	3592*	3593	3594	3595*	3600*	3601*	3602*	3608	3633	3634	3635	3636*
3637*	3663*	3664*	3665*	3666*	3667*	3668*	3669*	3670*	3671*	3672*	3679*	3680*
3681*	3682*	3683	3684	3685	3686	3687	3688	3699*	3700	3730*	3731*	3732*
3733*	3734*	3735*	3736	3742*	3746	3747	3748	3749	3750	3751	3756*	3777
3778*	3780*	3798	3800*	3854*	3855	3866	3867*	3872*	3874*			
2477	2480	2483	2486	2489	2492	4026*						
87*	290	396	402	711								

2477	2480	2483	2486	2489	2492	4026#						
87#	290	396	402	711								
92#												
1580	2079	2087	2095	2115	2124	2564#	2778	2781				
2750*	2756#	2850*										
903	961	1135	1174	1201	1219	1416	1478	1493	1531	1618	1784	1887
1902	1919	1995	2008	2048	2060	2169	2183	2199	2314	3722#		
2965	3184*	3283*	3291#									
2755	2758#											
370#	405	409*	798	831	837	2463	2503	2508	2557	2602	2701	2706
2709	2754	2758	2762	2789	2796	2799	2802	2861	2865	2868	2875	2879
2882	2893	2900	2903	2910	2985	2987	3012	3023	3036	3056	3061	3078

2965	3184*	3283*	3291*									
2755	2758*											
370*	405	409*	798	831	837	2463	2503	2508	2557	2602	2701	2706
2709	2754	2758	2762	2789	2796	2799	2802	2861	2865	2868	2875	2879
2882	2893	2900	2903	2910	2985	2987	3012	3023	3036	3056	3061	3078

[illegible]

5089	3100
121#	798
120#	2503
143#	
142#	
141#	
140#	
139#	
138#	

SW8	=	000400	137#	2789															
SW9	=	001000	136#	2987															
S2BIG		016037	496	4071#															
TAB		015016	3166	3925#															
ARGET		012650	2255	2783	2963	2966	3128	3374*	3392	3399#	3410*	3415*	3418	3420*	3422*				
			3433*	3435*	3446*	3448*													
TBITVE=		000014	178#																
TCRLF =		000200	3515	3554#															
TESTS		002064	709	712	713#														
THT	=	000011	3513	3553#															
TKVEC	=	000060	185#																
TPVEC	=	000064	186#																
TRAPVE=		000034	184#																
TRKCNT		012646	2639*	2657*	2672*	3379*	3380*	3381*	3384*	3386*	3389*								
TRKLMT		001542	453	460#															
TRTVEC=		000014	179#																
TSECTO		013140	2256*	2780	2964	2967	3133	3462*	3463*	3469#	3472*	3473*	3474	3476*	3477*				
			3481*																
TYPADR		011572	2898	2996	3109	3127#													
TYPESOD		011742	3111	3161#															
TYPE	=	104400	424	435	445	446	451	455	464	469	473	488	491	494	496				
			789	794	795	800	801	814	2398	2405	2414	2422	2468	2470	2477				
			2480	2483	2486	2489	2492	2510	2605	2703	2704	2705	2756	2767	2768				
			2773	2863	2864	2877	2878	2896	2899	2913	2918	2923	2927	2994	2995				
			2997	3002	3006	3011	3025	3029	3032	3034	3035	3059	3060	3080	3081				
			3087	3088	3102	3104	3110	3115	3116	3118	3123	3124	3127	3132	3139				
			3142	3144	3161	3166	3167	3170	3518	3627	3716#	3754	3864						
TYPERR		010752	2992	2999#															
TYPORC	=	104401	3717#																
TYPON	=	104403	471	490	493	2479	2482	2485	2488	2491	3010	3169	3719#						
TYPDS	=	104402	448	466	2402	2419	2474	2610	2770	2920	3504	3106	3120	3129	3136				
			3163	3718#															
TOMOTH		002036	708#	713															
T1		002446	713	863#															
T10		003766	716	1661#															

T5	003412	713	1377#																
T6	003636	713	1561#																
T6FILL	003654	1562	1567#	1754	2330	2347													
T7	003770	713	1669#																
T7EMPT	004000	1669	1671#	1755	2317	2337													
UNITSE	012446	425*	428*	440*	443*	518*	630	784	806*	841*	1435	1540	1769	1788					
		1847	1980	2033	2327	2466	2498*	2776	2897*	2909*	2983*	3091*	3140	3330					
		3332	3334*	3335*	3337	3339	3341*	3342*	3347*	3351	3353	3479							
WRITE	006736	2652	2655	2670	2679#														
WRIDON	007226	2715	2730#	2740	2752														
WITER	007242	2716	2735#																
WRTIE =	000105	198#	2717																
WRTSEK	007302	2737	2745#																
WTDOTE=	000115	200#	2720																
WTRDCK	006670	2391	2666#																
XERROR	006060	218	2456#																
XFRBYT	007012	2691#																	
XHOME	007566	2795	2805#																
XID	012656	3375*	3376*	3379	3402#	3431													
XOD	012654	3377*	3378*	3380	3401#														
XPATGE	012100	3224#	3242																
XREAD	007602	2812#	2835	3042															
XSA202	001470	423	429	442	444#														
XSCOPE	006316	214	2522#																
XSDN	006540	2572	2624#																
XSUBSC	006336	2537#	3722																
XWRITE	006742	2680#	2697	2733															
XWTRDC	006674	2667#	2676																
XXPATG	012150	3251#	3253	3262															
\$CHARC	013440	3529*	3536	3545*	3550#														
\$CRLF	013455	3519	3563#																
\$OB20	014240	3779	3797#																
\$DECVL	014420	3799	3845#																
\$FILLC	013452	3522	3560#																
\$FILLS	013451	3559#																	
\$HD =	000003	21	22																
\$ILLUP	014172	3728	3759#																
\$LF	013457	3564#																	
\$MAIL =	***** U	3507																	
\$NULL	013450	3524	3558#																
\$OCNT	013704	3599*	3628*	3641#															
\$OMODE	013706	3594*	3638*	3603	3606*	3617*	3643#												
\$POWER	014200	3116	3755	3762#															
\$PWAD	014166	3757#																	
\$PWADN	014044	216	3728#	3752															
\$PWARMG	014162	3755#																	
\$PWRRUP	014112	3737	3742#																
\$QUES	013454	3562#																	
\$RDCHR=	***** U	3720																	
\$RDDEC=	***** U	3720																	
\$RDLIN=	***** U	3720																	
\$RDOCT=	***** U	3720																	
\$RESRE	013746	3678#	3721																
\$R2A =	***** U	3722																	

SSAVRE	013710	3662#	3720																
SSAVR6	014176	3736#	3742	3743#	3744#	3761#													
SSB20	014210	3777#	3873																
SSWR =	160000	21	22#	3758															
STN =	000001	21#																	
STNPLR	014350	3804	3805	3825#															
STP8	013446	3542#	3557#																
STPFLG	013453	3501	3561#																
STPS	013444	3540	3556#																
STRAP	014004	220	3699#																
STRP =	000007	3707#	3717#	3718#	3719#	3720#	3721#	3722#	3723#										
STRPAD	014026	3704	3715#																
STYPBN=	*****	3720																	
STYPOS=	*****	3720																	
STYPE	013230	3501#	3707	3716															
STYPEC	013374	3521	3528	3535	3540#	3541													
STYPEX	013442	3546	3548	3551#															
STYPOC	013506	3597#	3717																
STYPON	013522	3596	3599#	3719															
STYPOS	013462	3592#	3718																
SOFILL	013705	3593#	3597#	3607	3642#														
.	= 016500	204#	206	207#	213#	223#	227#	230#	234#	290#	827#	1749	2622#	3556					
		3557	3558	3559	3560	3561	3562	3563	3564	3565#	3739	3760	3845#	4109#					

[illegible]

ADC	2574	3814													
ADD	416	839	1582	2102	2457	2566	2573	2626	2884	2957	2966	2967	3083	3112	3283
	3294	3295	3305	3355	3473	3477	3511	3595	3605	3813	3815	3874			
ASL	811	2437	2438	2439	2440	2441	3186	3703							
ASLB	2969														
BCC	3226														
BEQ	406	431	525	635	785	799	832	838	843	867	921	1143	1195	1383	1440
	1486	1499	1545	1623	1773	1779	1798	1857	1897	1927	1990	2004	2018	2043	2056
	2070	2138	2178	2204	2474	2504	2550	2558	2575	2603	2710	2726	2737	2823	2829
	2840	2904	2908	3019	3022	3031	3037	3041	3072	3090	3096	3157	3231	3286	3372
	3394	3419	3480	3514	3622	3858									
BGE	3475	3548													
BGT	2990	3629													
BHI	461	463	485	487											
BHIS	810														
BIC	444	518	629	632	633	783	806	808	841	1189	1247	1305	1366	1434	1437
	1438	1538	1542	1777	1796	1895	2392	2442	2998	3134	3296	3306	3334	3341	3370
	3411	3416	3430	3445	3472	3619									
BIS	428	440	443	1028	1233	1287	1328	1347	1377	2334	2386	2498	2897	2909	2937
	2944	2983	3091	3335	3342	3373	3382	3481	3624	3625	3817				
BISB	2776														
BIT	426	430	784	798	831	837	1485	2463	2466	2503	2508	2557	2571	2602	2701
	2709	2725	2736	2754	2762	2789	2796	2802	2822	2839	2846	2861	2868	2875	2882
	2893	2903	2907	2910	2985	2987	3023	3036	3056	3071	3074	3078	3085	3095	3100
	3140	3156	3330	3339	3353	3479									
BITB	2891	3076	3536												
BLT	3527	3630	3810												
BMI	631	1436	1541	2434	3070	3369	3395								
BNE	423	427	434	453	482	788	792	1120	2464	2467	2509	2523	2538	2572	2640
	2658	2673	2702	2719	2728	2731	2755	2763	2790	2797	2803	2825	2835	2847	2860
	2862	2869	2876	2883	2890	2892	2894	2911	2940	2959	2936	2938	3024	3057	3075
	3077	3079	3086	3101	3141	3331	3340	3354	3432	3455	3508	3516	3523	3537	3544
	3620	3745	3821	3860											
BPL	442	1588	1770	1789	1848	1911	1981	2034	2191	2275	2328	2565	2690	2707	2759
	2795	2800	2832	2856	2866	2880	2901	2954	2992	3013	3062	3333	3338	3352	3409
	3414	3444	3502	3541	3618										
BR	235	236	397	407	429	456	472	477	495	502	706	812	1123	1275	1278
	1334	1337	1354	1379	1495	1576	1581	1586	1742	1749	1773	1792	1850	1853	1909
	1983	1986	2014	2036	2039	2066	2080	2082	2084	2085	2088	2090	2092	2093	2096
	2098	2101	2105	2116	2117	2119	2122	2125	2126	2128	2131	2134	2189	2264	2267
	2273	2279	2336	2676	2693	2752	2779	2782	2793	2852	2870	2914	2961	3033	3094
	3098	3143	3155	3159	3209	3218	3234	3242	3253	3262	3272	3281	3385	3387	3396
	3504	3520	3530	3539	3546	3596	3611	3632	3739	3760	3812	3862			
BVS	797														
CLC	3229	3297													
CLR	401	419	420	421	425	454	476	519	793	863	918	971	1025	1136	1204
	1230	1262	1318	1488	1543	1567	1671	1753	1845	1846	1892	1925	1978	1979	2000
	2031	2032	2052	2053	2086	2094	2112	2113	2174	2175	2200	2261	2315	2316	2329
	2346	2465	2531	2624	2684	2685	2714	2722	2813	2814	2819	2942	2943	2946	2949
	3044	3184	3207	3241	3301	3375	3377	3422	3609	3743	3806				
CLRB	3461	3545	3822												
CMP	405	408	524	634	791	809	866	920	1142	1194	1276	1335	1382	1439	1498
	1544	1622	1682	1778	1797	1856	1896	1926	1989	2003	2017	2042	2055	2069	2137
	2177	2203	2549	2718	2828	2889	2989	3285	3418	3816					

NEGB	2970														
NOP	846	847	848	974	975	1029	1070	1207	1208	1234	1235	1288	1289	1348	1349
	1661														
RESET	844														
ROL	3298	3299	3610	3612	3613	3614	3616								
ROLB	3233														
ROR	3303	3304													
RTI	414	973	978	1027	1073	1206	1211	1232	1238	1285	1292	1345	1352	2408	2425
	2506	2533	2539	2551	2560	2687	2724	2821	2951	3082	3512	3638	3673	3689	3758
RTS	840	849	1572	1583	1654	1675	2103	2318	2444	2511	2567	2576	2627	2641	2659
	2675	2732	2764	2774	2784	2804	2806	2833	2836	2915	2929	2973	3020	3045	3084
	3113	3125	3145	3171	3287	3289	3309	3336	3343	3356	3383	3412	3417	3421	3423
	3434	3436	3447	3449	3466	3482	3551	3705	3781	3824	3868	3877			
SBC	3808														
SEC	3224	3232													
SUB	2472	2609	3380	3459	3463	3807	3809								
TRAP	3707	3717	3718	3719	3720	3721	3722								
TST	422	441	452	708	787	1116	1587	1788	1910	2190	2274	2522	2537	2695	2706
	2735	2758	2794	2799	2831	2838	2855	2859	2865	2879	2900	2939	2977	3012	3018
	3021	3061	3069	3230	3288	3337	3351	3371	3393	3408	3443	3454	3509	3517	3538
	3621	3701													
TSTB	481	630	1435	1540	1769	1847	1980	2033	2327	2432	2564	2689	2953	2991	3030
	3332	3368	3413	3501	3540	3857									
.ASCII	3562	4061	4071	4082	4089										
.ASCIZ	621	823	2513	2615	2620	3563	3564	3762	3891	3895	3894	3899	3905	3910	3913
	3917	3920	3925	3928	3930	3932	3939	3942	3945	3948	3952	3956	3958	3964	3968
	3974	3978	3982	3987	3989	3991	3994	3997	4003	4006	4010	4013	4016	4020	4024
	4026	4028	4030	4034	4038	4043	4051	4059	4067	4078	4097				
.BLKB	3845	4109													
.BYTE	449	450	467	468	2403	2404	2420	2421	2475	2476	2611	2612	2771	2772	2921
	2922	3107	3108	3121	3122	3130	3131	3137	3138	3164	3165	3558	3559	3560	3561
	3639	3640	3641	3642											
.ENABL	4														
.END	4111														
.ENDC	16	88	174	188	245	246	247	258	259	260	265	266	267	286	287
	288	379	390	450	451	468	469	535	612	643	705	877	902	931	960
	991	1009	1035	1069	1083	1112	1154	1173	1393	1415	1452	1477	1509	1530	1599
	1617	1633	1651	1697	1740	1808	1833	1867	1886	1937	1967	2150	2168	2226	2242
	2290	2313	2404	2405	2421	2422	2449	2450	2454	2455	2587	2601	2644	2662	2772
	2773	2922	2923	3053	3108	3109	3122	3123	3131	3132	3138	3139	3165	3166	3180
	3203	3212	3237	3245	3256	3265	3274	3329	3391	3406	3426	3439	3484	3507	3567
	3645	3691	3700	3703	3716	3717	3718	3719	3720	3721	3722	3724	3736	3746	3756
	3758	3765	3766	3784	3848	4105									
.EQUIV	88	89	91	106	107	136	137	138	139	140	141	142	143	144	145
	164	165	166	167	168	169	170	171	172	173					
.EVEN	827	2514	2622	3565	3764	4102									
.IF	12	86	146	174	244	245	246	257	258	259	264	265	266	285	286
	287	378	389	449	450	467	468	532	609	645	702	874	899	928	957
	988	1006	1032	1066	1080	1109	1151	1170	1390	1412	1449	1474	1506	1527	1596
	1614	1630	1648	1694	1737	1805	1830	1864	1883	1934	1964	2147	2165	2223	2239
	2287	2310	2403	2404	2420	2421	2448	2449	2453	2454	2584	2598	2643	2661	2771
	2772	2921	2922	3052	3107	3108	3121	3122	3130	3131	3137	3138	3164	3165	3179
	3202	3211	3236	3244	3255	3264	3273	3328	3390	3405	3425	3438	3483	3507	3566
	3644	3690	3699	3703	3707	3717	3718	3719	3720	3721	3722	3723	3736	3746	3754

.IFF	3756 88 450 3053 3391	3758 245 468 3108 3406	3762 246 2403 3122 3426	3765 247 2404 3131 3439	3783 258 2420 3138 3484	3847 259 2421 3165 3567	4104 260 2449 3180 3645	265 266 2450 3203 3691	267 266 2455 3212 3700	286 267 2644 3237 3724	287 286 2662 3245 3756	288 2772 2665 3256 3766	379 2921 3265 3274 3784	390 2922 3329 4105 3163
.IIF	11 3556 3721 3663	16 3557 3722 3683	21 3558 3730 3746	22 3559 3746 531	448 3560 613 644	466 3561 706 873	2401 3562 873 903	2418 3563 903 927	2770 3564 927 961	2920 3565 961 987	3106 3716 987 1010	3120 3717 1010 1031	3129 3718 1031 1070	3136 3719 1070 1693
.IRP														
.LIST	1079 1741 3707	1113 1804 3716	1150 1834 3717	1174 1863 3718	1389 1887 3719	1416 1933 3720	1448 1968 3721	1478 2146 3722	1505 2169 3723	1531 2222 3723	1595 2243 3723	1618 2286 3723	1629 2314 3723	1652 2583 3723
.MACRO	239	241	3707											
.MCALL	5	6	7	188										
.MLIST	2 1079 1741 3707	188 1113 1804 3716	206 1150 1834 3717	531 1174 1863 3718	613 1389 1887 3719	644 1416 1933 3720	706 1448 1968 3721	873 1478 2146 3722	903 1505 2169 3723	927 1531 2222 3723	961 1595 2243 3723	987 1618 2286 3723	1010 1629 2314 3723	1031 1652 2583 3723
.PAGE	42	82	2446											
.REPT	206 1151 1830	532 1170 1864	609 1390 1883	645 1412 1934	702 1449 1964	874 1474 2147	899 1506 2165	928 1527 2223	957 1596 2239	988 1614 2287	1006 1630 2310	1032 1648 2584	1066 1694 2598	1109 1805 2446
.SBTTL	84 1744 2518 3725	242 1751 2630 3767	262 1763 2677 3785	288 1836 2808 3878	339 1971 2932	391 2024 3066	503 2107 3172	850 2212 3315	1012 2250 3358	1250 2319 3450	1308 2341 3485	1368 2356 3568	1554 2370 3646	1663 2446 3708
.TITLE	11													
.WORD	205 3757	208 3782	209 3820	210	211	228	370	371	1124	3005	3550	3556	3557	3755

ERRORS DETECTED: 0

*DZRXBD, DZRXBD/CRF/SOL=DZRXBD
 RUN-TIME: 36 29 5 SECONDS
 CORE USED: 12K

