

# RX11

SYSTEM RELIABILITY TEST  
MD-11-DZRXA-E

EP-DZRXA-E-DL-A  
COPYRIGHT © 1978  
FIGHE 1 OF 1

NOV 1978  
digital  
MADE IN USA

The main body of the document consists of a 10x10 grid of small, illegible data tables or charts. Each cell in the grid contains a small table with multiple columns and rows of text and numbers, typical of a reliability test report. The text is too small to be read, but the layout suggests a comprehensive set of test results.



IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZRXA-E-D  
PRODUCT NAME: RX11 SYSTEM RELIABILITY TEST  
DATE: APRIL 1976  
MAINTAINER: DIAGNOSTIC ENGINEERING  
AUTHOR: DAVID L. ADAMS

COPYRIGHT (C) 1975, 1976  
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

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

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

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



TABLE OF CONTENTS

- 1.0 GENERAL PROGRAM INFORMATION
  - 1.1 ABSTRACT
  - 1.2 SYSTEM REQUIREMENTS
    - 1.2.1 HARDWARE
    - 1.2.2 SOFTWARE
- 2.0 OPERATING INSTRUCTIONS
  - 2.0.1 OUTLINE OF OPERATING PROCEDURE
  - 2.1 LOADING PROCEDURE
  - 2.2 STARTING ADDRESSES
  - 2.3 OPERATOR ACTION BEFORE STARTING PROGRAM
    - 2.3.1 DEVICE ADDRESS SELECTION
    - 2.3.2 NON-STANDARD DISKETTE ADDRESS SELECTION
    - 2.3.3 SOFTWARE SWITCH REGISTER (LOC. 176)
    - 2.3.4 TEST PARAMETER SELECTION ("DTESTP" LOC. 1212)
  - 2.4 OPERATOR ACTION TO RUN THE PROGRAM
    - 2.4.1 STARTING THE PROGRAM
    - 2.4.2 OPERATING CONDITIONS
    - 2.4.3 ACT11 & XXDP HOOKS
  - 2.5 PROGRAM OPTIONS
    - 2.5.1 PSEUDO SCOPE LOOP
    - 2.5.2 DISKETTE COMPATABILITY
    - 2.5.3 RX11 TO RX8 COMPATABILITY
  - 2.6 RUN TIME
- 3.0 ERROR DETECTION
  - 3.1 ERROR DEFINITIONS
  - 3.2 DEFINITIVE ERROR CODES
  - 3.3 UNEXPECTED OR MISSING ERROR CONDITIONS
  - 3.4 POWER FAILURE
  - 3.5 PROGRAM HUNG
- 4.0 ERROR REPORTING
- 5.0 HALTS



## 1.0 GENERAL PROGRAM INFORMATION

## 1.1 ABSTRACT

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

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

- A. COMPLETED PASSES OF THE PROGRAM
- B. NUMBER OF RESTARTS
- C. NUMBER OF SECTORS WRITTEN/READ
- D. RECOVERABLE AND UNRECOVERABLE FAULTS AS FOLLOWS:
  - 1. PARITY ERRORS
  - 2. ERROR FLAG ERRORS
  - 3. INTERRUPT ERRORS
  - 4. SEEK ERRORS
  - 5. DATA CRC ERRORS
  - 6. CRC NO DATA ERRORS
  - 7. DATA NO CRC ERRORS
  - 8. DELETED DATA MARK ERRORS
  - 9. WRITE ERRORS
  - 10. READ ERRORS
- E. TOTAL OF EACH ERROR CODE DETECTED
- F. TOTAL OF TIMES EACH TRACK WAS ACCESSED
- G. TOTAL OF TIMES HEAD MOVED TO A TRACK.
- H. TOTAL OF ERRORS DETECTED PER TRACK PER DRIVE.

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

## 1.2 SYSTEM REQUIREMENTS

## 1.2.1 HARDWARE REQUIREMENTS

THE FOLLOWING EQUIPMENT IS REQUIRED.

- A. PDP-11 SERIES OF COMPUTER WITH MIN 8K MEMORY
- B. RX11 FLOPPY DISK SYSTEM INCLUDING DUAL OR SINGLE DRIVE RX01 AND PDP-11 INTERFACE. (SEE SECTION 2.3 FOR SELECTION OF REGISTER ADDRESSES AND VECTOR ADDRESS)  
NOTE: A DISKETTE MUST BE INCLUDED WITH EACH DRIVE TESTED.
- C. CONSOLE TELEPRINTER

## 1.2.1 SOFTWARE REQUIREMENTS

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

## 2.0 OPERATING INSTRUCTIONS

## 2.0.1 OUTLINE OF OPERATING PROCEDURE



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

- A. LOAD THE PROGRAM INTO MEMORY
  - 1. IF IT'S BEING LOADED FROM A DISKETTE, REPLACE THE "LIBRARY DISKETTE" WITH A "SCRATCH DISKETTE"
- B. START THE PROGRAM RUNNING AT LOCATION 200
- C. THE PROGRAM WILL TYPE OUT THE FOLLOWING:
  - 1. PROGRAM NAME AND REVISION
  - 2. RX11 REGISTER AND VECTOR ADDRESSES
  - 3. UNITS BEING TESTED
  - 4. "P"ATTERN, "T"EST, AND "S"EQUENCEIT THEN STARTS RUNNING UNDER THOSE CONDITIONS
- D. IF THERE ARE NO ERRORS, AT THE END OF THE COMPLETED PASS A "D" AND "BELL" IS TYPED, AND THE PROGRAM WILL CONTINUE ON FOR ANOTHER PASS.
- E. HALT THE PROGRAM AS FOLLOWS:
  - 1. IF THERE IS A HARDWARE SWITCH REGISTER, PUT SW 14 UP TO HALT AT THE END OF PASS.
  - 2. IF THERE IS NO SWITCH REGISTER, OR TO HALT AT ANY TIME, HALT THE PROCESSOR.

## 2.1 LOADING PROCEDURE

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

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

## 2.2 STARTING ADDRESSES

THE PROGRAM HAS THREE (3) STARTING LOCATIONS.

### 2.2.1 INITIAL START (LOC. 200)

THIS STARTING LOCATION INITIALIZES THE PROGRAM AS FOLLOWS:

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



## 2.2.2 RESTART (LOC. 202)

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

## 2.2.3 ERROR DUMP (LOC. 204)

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

## 2.3 OPERATOR ACTION BEFORE STARTING PROGRAM

## 2.3.1 DEVICE ADDRESS SELECTION

LIKE MOST OPTIONS ON THE PDP11 THE RX11 INTERFACE CARD HAS JUMPERABLE REGISTER AND VECTOR ADDRESSES. THIS ALLOWS FOR DEVICES WITH THE SAME STANDARD ADDRESSES TO BE JUMPERED TO AN OTHER ADDRESS SO THEY WILL RUN WITHOUT CONFLICT.

THE PROGRAM MUST KNOW WHAT ADDRESSES ARE BEING USED, AS IT IS THROUGH THESE REGISTERS AND VECTOR THAT ALL COMMUNICATION BETWEEN THE PDP11 AND RX11 IS HANDLED.

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

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



## 2.3.2 NON-STANDARD DISKETTE ADDRESS SELECTION

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

## A. DEFINITIONS:

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

## B. LOCATIONS:

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

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

## C. RESTRICTIONS:

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

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



## 2.3.3 SOFTWARE SWITCH REGISTER (LOC. 176)

FOR THE PDP 11 PROCESSORS THAT DO NOT HAVE A HARDWARE SWITCH REGISTER, OR IF THE OPERATOR WISHES TO SELECT THE SOFTWARE SWITCH REGISTER, BY PUTTING ALL THE SWITCHES UP TO A "1". (THIS MUST BE DONE EACH TIME THE PROGRAM IS STARTED AT LOCATION 200, OTHERWISE THE PROGRAM WILL USE THE HARDWARE SWR.) 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.

TO CHANGE THE SOFTWARE SWR. WHILE THE PROGRAM IS RUNNING, TYPE "CONTROL G". EACH TIME THE SWR. IS TO BE TESTED THE PROGRAM WILL CHECK TO SEE IF THE SOFTWARE SWR IS SELECTED, AND THE PROGRAM IS NOT RUNNING IN AUTO MODE OF RXDP/ACT11. IF BOTH CONDITIONS EXIST THEN THE PROGRAM CHECKS FOR THE CTRL G IN THE KEYBOARD BUFFER. IF THE CTRL G IS THERE THE CONTENTS OF THE SOFTWARE SWR. ARE PRINTED AND A "NEW =" IS ASKED FOR. THE OPERATOR MAY NOW TYPE IN THE NEW SWITCH REGISTER CONTENTS, TERMINATED BY A CARRIAGE RETURN (CR), OR IF HE DOESN'T WANT TO CHANGE THE SWR. JUST TERMINATE WITH THE (CR). NOTE SEE THE CHARACTER RESTRICTIONS BELOW.

WHEN THE PROGRAM DETECTS THE (CR) IT WILL REPLACE THE CONTENTS OF THE SOFTWARE SWR. IF A NEW ONE HAS BEEN TYPED IN, AND RETURN TO THE FLOW OF THE PROGRAM.

NOTE: CHARACTER RESTRICTIONS FOR CHANGING THE SOFTWARE SWR.

1. ONLY OCTAL NUMBERS 0 - 7 ARE ACCEPTED. ANY OTHER CHARACTER TYPED WILL BE PRINTED AS A ? AND THE WHOLE SWR MUST BE RETYPED.
2. TO WIPE OUT A "NEW" CONTENTS JUST TYPED IN, TYPE CTRL U. NOW A NEW CONTENTS CAN BE RETYPED.
3. ONLY 6 OCTAL CHARACTERS WILL BE PUT INTO THE SWR. IF MORE THAN 6 CHARACTERS ARE TYPED IN ONLY THE LAST 6 WILL BE PUT INTO THE SWR.



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

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

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

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

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

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

THE SELECTIONS ARE DEFINED AS FOLLOWS:

P = DATA PATTERN SELECTION

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

T = FUNCTIONAL TESTS

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

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



# J01

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

S = TRACK SEQUENCING

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

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

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

## 2.4 OPERATOR ACTION TO RUN THE PROGRAM

### 2.4.1 STARTING THE PROGRAM

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

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

- SA200 - THE PROGRAM WILL TYPE DRIVE AND TEST PARAMETERS, THE TWO REGISTER ADDRESSES, AND VECTOR ADDRESS. IT WILL THEN BEGIN FUNCTIONAL TESTING, AND INFORMATION COLLECTION.
- SA202 - THE PROGRAM WILL CONFIGURE TO CONDITIONS SET IN PREVIOUS "INITIAL START", PRINT OUT THESE CONDITIONS AND CONTINUE FUNCTIONAL TESTING AND DATA COLLECTING. THE ONLY OPERATOR ACTION REQUIRED IS THE DYNAMIC SELECTION OF OPERATING CONDITIONS IN THE SWITCH REGISTER AS REQUIRED.
- SA204 - THIS PROGRAM WILL REPORT VIA THE TELEPRINTER ALL DATA COLLECTED. NO OTHER OPERATOR ACTION IS REQUIRED.



## 2.4.2 OPERATING CONDITIONS

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

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

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

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

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

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

## 2.4.3 ACT11 &amp; XXDP HOOKS

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

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

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



## 2.5 PROGRAM OPTIONS

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

## 2.5.1 PSEUDO - SCOPE LOOP

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

## 2.5.2 DISKETTE COMPATABILITY

TO CHECK FOR DATA TRANSFER COMPATABILITY BETWEEN DRIVE UNITS USE THE FOLLOWING PROCEDURE. (IT IS ASSUMED THE SYSTEM HAS DUAL DRIVE RX01, IF NOT YOU MAY TEST COMPATABILITY BETWEEN DIFFERENT RX11 SYSTEMS BY RUNNING THE SAME TESTS ON BOTH.) THIS TEST INSURES PROPER HEAD ALIGNMENT.

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

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

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

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



## 2.5.3 RX11 TO RX8 COMPATABILITY

TO WRITE A DISKETTE ON THE RX11 AND READ/VERIFY THE SAME DATA ON A RX8 REQUIRES THE FOLLOWING:

A. SET UP LOCATION "DTESTP" (1212) WITH ONE OF THE FOLLOWING DATA PATTERNS

P=1	(0'S)	
P=2	(1'S)	
P=5	(125)	*
P=6	(314)	*

\* NOTE: IF ONE OF THESE PATTERNS IS SELECTED A MODIFICATION TO THE PROGRAM MUST BE MADE TO ALLOW THIS DATA TO BE READ ON A RX8 SYSTEM. THE MODIFICATION IS IN THE "PAT125" ROUTINE, THE TWO (2) LOCATIONS CONTAINING THE "COMB DATABYTE" INSTRUCTION MUST BE CHANGED TO TWO (2) NOP'S (OCTAL 000240)

B. SET IN "DTESTP" TEST 3 WHICH WILL WRITE AND READ CHECK THE DATA THEREBY VERIFYING THE QUALITY OF THE DATA PRIOR TO THE DISKETTE SWAP.

SET THE SEQUENCE TO 1 TO INSURE THAT ALL TRACKS HAVE BEEN WRITTEN ON.

C. HALT THE PROGRAM AT THE COMPLETION OF ONE PASS.

D. THE DISKETTE IS NOW READY TO BE READ ON THE RX8 SYSTEM. SEE MD-8-DIRXB-\* FOR THE PROCEDURE TO READ THIS DISKETTE.

TO READ A DISKETTE THAT WAS WRITTEN ON A RX8 SYSTEM SET UP AS FOLLOWS:

E. SET IN "DTESTP" THE DATA PATTERN THAT WAS USED BY THE RX8 SYSTEM. (SEE NOTE UNDER SECTION A.)

F. SET "DTESTP" FOR TEST 4 (READ CHECK ONLY) AND SEQUENCE 1 TO CHECK ALL TRACKS. START THE PROGRAM, IT SHOULD RUN WITHOUT DATA ERRORS.



## 2.6 RUN TIME

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

EXAMPLES OF RUN TIMES FOLLOW.

(THESE TIMES ARE FOR 1 DRIVE AND COMPLETE DISKETTE, MAXIMUM ID/OD, FIRST/LAST LIMITS. IF OPERATING ON 2 DRIVES DOUBLE THE TIME.)

THESE TIMES ARE FOR A PDP 11/05 PROCESSOR AND MAY CHANGE SLIGHTLY FOR A FASTER PROCESSOR.

TEST SELECTION	P	T	S	TIME/PASS
	7	6	5	3 MIN.30 SEC.
	7	7	4	2 MIN.47 SEC.
	7	7	7	2 MIN.23 SEC.
	6	3	4	1 MIN.46 SEC.
	7	3	1	1 MIN.30 SEC.
	7	1	1	51 SEC.

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

## 3.0 ERROR DETECTION

## 3.1 PROGRAM DEFINITIONS

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

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

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

THE PROGRAM HAS DEFINED THE FOLLOWING AS ERRORS.

## 3.1.1 WRITE ERROR

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

## 3.1.2 READ (CRC) ERROR

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



## 3.1.3 CRC AND DATA ERROR

## 3.1.4 NO CRC ERROR BUT DATA ERROR

## 3.1.5 CRC ERROR BUT NO DATA ERROR

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

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

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

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

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

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

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

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

## 3.1.6 SEEK ERROR

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

## 3.1.7 PARITY ERROR

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



### 3.2 DEFINITIVE ERROR CODES

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

A DEFINITIVE ERROR CODE REPRESENTS [WHERE] WITHIN A FUNCTION AN ERROR WAS DETECTED.

THE FOLLOWING ARE THE DEFINITIVE ERROR CODES AND MEANINGS:

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

### 3.3 UNEXPECTED OR MISSING ERROR CONDITIONS

#### 3.3.1 MISSING DD MARK

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

#### 3.3.2 UNEXPECTED DD MARK

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



## 3.3.3 NO INTERRUPT ON DONE

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

## 3.3.4 UNKNOWN INTERRUPT

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

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

## 3.4 POWER FAILURE

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

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

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

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

## 3.5 PROGRAM HUNG

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

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

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



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

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

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

#### 4.C ERROR REPORTING

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

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

#### 5.0 HALTS

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

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

MAINDEC-11-DZRXA-E  
DZRXA.E.P11

MACY11 27(732) 20-SEP-76 15:46 PAGE 2

1  
2  
3  
4  
5  
6  
7  
8

.NLIST CND,MD,MC  
.LIST ME  
.ENABL ABS,AMA  
.MCALL .HEADER,.EQUAT,.SETUP,.STYPE,.STYPOCT,.SREAD  
.MCALL .\$POWER,STARS,.\$DB2D,.\$SB2D,.\$SAVE,.\$TRAP,GETSWR



F02

```

.TITLE MAINDEC-11-DZRXA-E
;*COPYRIGHT (C) MARCH 21,1976
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*PROGRAM BY DAVID L ADAMS
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-CO),MAR 21, 1976.
;*
$TN=1
$SWR=160000      ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT

```

```

000001
160000

```

```

;COPYRIGHT (C) 1975,1976
;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 OWNERSHIP OF THE
;SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.

;THE INFOMATION 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.

```

.SBTTL BASIC DEFINITIONS

```

;*INITIAL ADDRESS OF THE STACK POINTER *** 1200 ***
STACK= 1200
.EQUIV EMT,EPROR      ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE     ;;BASIC DEFINITION OF SCOPE CALL

```

```

;*MISCELLANEOUS DEFINITIONS
HT= 11                ;;CODE FOR HORIZONTAL TAB
LF= 12                ;;CODE FOR LINE FEED
CR= 15                ;;CODE FOR CARRIAGE RETURN
CRLF= 200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776           ;;PROCESSOR STATUS WORD
.EQUIV PS,PSW

```

```

000011
000012
000015
000200
177776

```

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	177774	STKLMT= 177774	:: STACK LIMIT REGISTER
58	177772	PIRQ= 177772	:: PROGRAM INTERRUPT REQUEST REGISTER
59	177570	DSWR= 177570	:: HARDWARE SWITCH REGISTER
60	177570	DDISP= 177570	:: HARDWARE DISPLAY REGISTER

::\*GENERAL PURPOSE REGISTER DEFINITIONS

62		R0= %0	:: GENERAL REGISTER
63	000000	R1= %1	:: GENERAL REGISTER
64	000001	R2= %2	:: GENERAL REGISTER
65	000002	R3= %3	:: GENERAL REGISTER
66	000003	R4= %4	:: GENERAL REGISTER
67	000004	R5= %5	:: GENERAL REGISTER
68	000005	R6= %6	:: GENERAL REGISTER
69	000006	R7= %7	:: GENERAL REGISTER
70	000007		
71		.EQUIV R6,SP	:: STACK POINTER
72		.EQUIV R7,PC	:: PROGRAM COUNTER

::\*PRIORITY LEVEL DEFINITIONS

74		PR0= 0	:: PRIORITY LEVEL 0
75	000000	PR1= 40	:: PRIORITY LEVEL 1
76	000040	PR2= 100	:: PRIORITY LEVEL 2
77	000100	PR3= 140	:: PRIORITY LEVEL 3
78	000140	PR4= 200	:: PRIORITY LEVEL 4
79	000200	PR5= 240	:: PRIORITY LEVEL 5
80	000240	PR6= 300	:: PRIORITY LEVEL 6
81	000300	PR7= 340	:: PRIORITY LEVEL 7
82	000340		

::\*"SWITCH REGISTER" SWITCH DEFINITIONS

83		SW15= 100000	
84		SW14= 40000	
85	100000	SW13= 20000	
86	040000	SW12= 10000	
87	020000	SW11= 4000	
88	010000	SW10= 2000	
89	004000	SW09= 1000	
90	002000	SW08= 400	
91	001000	SW07= 200	
92	000400	SW06= 100	
93	000200	SW05= 40	
94	000100	SW04= 20	
95	000040	SW03= 10	
96	000020	SW02= 4	
97	000010	SW01= 2	
98	000004	SW00= 1	
99	000002		
100	000001		
101		.EQUIV SW09,SW9	
102		.EQUIV SW08,SW8	
103		.EQUIV SW07,SW7	
104		.EQUIV SW06,SW6	
105		.EQUIV SW05,SW5	
106		.EQUIV SW04,SW4	
107		.EQUIV SW03,SW3	
108		.EQUIV SW02,SW2	
109		.EQUIV SW01,SW1	
110		.EQUIV SW00,SW0	

::\*DATA BIT DEFINITIONS (BIT00 TO BIT15)



113	100000	BIT15=	100000
114	040000	BIT14=	40000
115	020000	BIT13=	20000
116	010000	BIT12=	10000
117	004000	BIT11=	4000
118	002000	BIT10=	2000
119	001000	BIT09=	1000
120	000400	BIT08=	400
121	000200	BIT07=	200
122	000100	BIT06=	100
123	000040	BIT05=	40
124	000020	BIT04=	20
125	000010	BIT03=	10
126	000004	BIT02=	4
127	000002	BIT01=	2
128	000001	BIT00=	1
129		.EQUIV	BIT09,BIT9
130		.EQUIV	BIT08,BIT8
131		.EQUIV	BIT07,BIT7
132		.EQUIV	BIT06,BIT6
133		.EQUIV	BIT05,BIT5
134		.EQUIV	BIT04,BIT4
135		.EQUIV	BIT03,BIT3
136		.EQUIV	BIT02,BIT2
137		.EQUIV	BIT01,BIT1
138		.EQUIV	BIT00,BIT0
139			
140		.*BASIC "CPU" TRAP VECTOR ADDRESSES	
141	000004	ERRVEC= 4	:: TIME OUT AND OTHER ERRORS
142	000010	RESVEC= 10	:: RESERVED AND ILLEGAL INSTRUCTIONS
143	000014	TBITVEC=14	:: "T" BIT
144	000014	TRTVEC= 14	:: TRACE TRAP
145	000014	BPTVEC= 14	:: BREAKPOINT TRAP (BPT)
146	000020	IOTVEC= 20	:: INPUT/OUTPUT TRAP (IOT) **SCOPE**
147	000024	PWRVEC= 24	:: POWER FAIL
148	000030	EMTVEC= 30	:: EMULATOR TRAP (EMT) **ERROR**
149	000034	TRAPVEC=34	:: "TRAP" TRAP
150	000060	TKVEC= 60	:: TTY KEYBOARD VECTOR
151	000064	TPVEC= 64	:: TTY PRINTER VECTOR
152	000240	PIRQVEC=240	:: PROGRAM INTERRUPT REQUEST VECTOR
153			
154			
155		;SPECIAL EQUATES	
156			
157			
158	000013	RDOSTAT =13	
159	000033	RD1STAT =33	
160	000017	RDER =17	
161	000040	DONEBIT =40	
162	000101	FBIE =101	
163	000103	EBIE =103	
164	000105	WRTIE =105	
165	000107	RDIE =107	
166	000115	WTDDIE =115	
167	040001	RECAL =40001	
168	000200	PR4 =200	



```

169      000340
170      000000
171
172      000000
173 000000 000000 000000
174
175      000024
176 000024 013764
177 000026 000340
178
179      000034
180 000034 013716
181 000036 000340
182
183      000046
184 000046 011130
185
186      000052
187 000052 000000
188
189      000174
190 000174 000000
191 000176 000000

```

```

PR7      =340
OPEN     =0
.=0      .WORD 0,0
.=24     $PWRDN
          340
.=34     $TRAP           ;ADDRESS OF TRAP SERVICE
          340
.=46     LOGICAL        ;ACT11 EOP HOOK
.=52     .WORD 0
.=174    DISPRG:        0
          SWREG:        0

```

;;\*\*\*\*\*

```

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

```

```

209      000200
210 000200 000402
211 000202 000403
212 000204 000404
213 000206 000137 001220
214 000212 000137 002700
215 000216 000137 021062

```

```

.=200    BR 1$
          BR 2$
          BR 3$
1$:      JMP SA200           ; OPERATOR SELECTED CONDITIONS
2$:      JMP RESTART        ; RESTART PROGRAM WITH PREVIOUS CONDITIONS
3$:      JMP ERDUMP         ; STATISTICAL ERROR PRINT OUT

```

;;\*\*\*\*\*

```

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

```

1. DEFINITATIONS:

223  
224



225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
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

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

2. LOCATIONS:  
TRACKS LOC. 1200 BITS 14-----8 6-----0  
  ID          OD  
SECTORS LOC. 1202 BITS 12-----8 4-----0  
  LAST        FIRST

3. RESTRICTIONS:  
THE CONTENTS OF "OD" MUST BE <= THE CONTENTS OF "ID"  
THE CONTENTS OF FIRST MUST BE <= THE CONTENTS OF "LAST"

001200 001200  
000000  
001201  
001202 000000  
001203

  .=1200  
OD:  0  
ID=OD+1  
FIRST:  0  
LAST=FIRST+1

;;\*\*\*\*\*

: THE NEXT WORD IS THE LOCATION OF THE "INTERRUPT VECTOR" ADDRESS  
: IF THE HARDWARE IS JUMPERED FOR OTHER THAN STANDARD (264) VECTOR  
: ADDRESS, THEN LOAD INTO THIS LOCATION THE NEW VECTOR ADDRESS.

: IF THIS ADDRESS IS INCORRECT (DOES NOT MATCH THE HARDWARE) THE  
: PROGRAM WILL HALT AT THE VECTOR ADDRESS THE HARDWARE IS JUMPERED  
: TO, AS ALL OTHER VECTOR ADDRESSES WILL HAVE HALTS IN THEM.

\*\*\*\*\*NOTE\*\*\*\*\*

: THE INTERRUPT VECTOR ADDRESS CAN NOT BE SET FOR ADDRESSES 200  
: THRU 243, AS THESE ADDRESS ARE USED BY THE PROGRAM

\*\*\*\*\*

001204 000264

INTVEC: 264

;;\*\*\*\*\*

: THE FOLLOWING TWO WORDS CONTAIN THE DEVICE CODES FOR  
: THE RX11 INTERFACE REGISTERS  
: RXCS = COMMAND STATUS REGISTER  
: RXDB = DATA BUFFER REGISTER (USED AT VARIOUS TIMES FOR:)  
:          (RXTA = TRACK ADDRESS)  
:          (RXSA = SECTOR ADDRESS)  
:          (RXES = ERROR STATUS)

: IF THE RX11 SYSTEM UNDER TEST IS JUMPERED FOR REGISTER  
: ADDRESSES OTHER THAN STANDARD: RXCS = 177170  
:  RXDB = 177172



281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336

PLACE IN THESE TWO WORDS THE CORRESPONDING NEW ADDRESSES FOR THE REGISTERS. THE PROGRAM WILL TYPE OUT THE REGISTER ADDRESSES FOR VERIFICATION BY THE OPERATOR. IF THE ADDRESSES ARE INCORRECT (I.E. - DON'T MATCH THE HARDWARE) AND THE PROGRAM IS RUN WITH THE INCORRECT ADDRESSES, A 'HUNG' CONDITION WILL BE REPORTED AND THE PROGRAM WILL 'HALT' AS THE RX11 WILL NOT BE ABLE TO RESPOND TO COMMANDS.

\*\*\*\*\*

001206 177170  
001210 177172

RXCS: 177170  
RXDB: 177172

BIT ASSIGNMENT IN THE RXCS REGISTER.

KEY: R - READ ONLY BIT  
W - WRITE ONLY BIT

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

FUNCTION CODES:

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

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

- 15 - NOT USED
- 14 - NOT USED
- 13 - NOT USED
- 12 - NOT USED
- 11 - NOT USED
- 10 - NOT USED
- 9 - NOT USED
- 8 - NOT USED
- 7 - SELECTED DRIVE READY \*
- 6 - DELETED DATA
- 5 - NOT USED
- 4 - NOT USED
- 3 - WRITE PROTECT ERROR (WHEN AVAILABLE)
- 2 - INITIALIZE DONE \*\*



337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392

001212 000000  
001214 177570  
001216 177570

DTESTP: .WORD 0 ;TEST SELECTION WORD  
SWR: .WORD DSWR ;ADDRESS OF SWITCH REGISTER  
DISPLAY: .WORD DDISP ; ADDRESS OF DISPLAY REGISTER

1 - PARITY ERROR  
0 - CRC ERROR  
\* VISIBLE ONLY IF THE FUNCTION WAS A #12 "READ STATUS A"  
\*\* VISIBLE ONLY AFTER AN INITIALIZE [KEY] OR [PROGRAMMED] WAS ISSUED.

;;\*\*\*\*\*

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

;SWITCH REGISTER BITS

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

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

D = DELETED DATA FUNCTIONS

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

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

T = FUNCTIONAL TESTS  
0 DO TEST 7 (WRITE/READ/READ CHECK)  
1 WRITE ONLY  
2 WRITE/READ  
3 WRITE/READ CHECK  
4 READ CHECK ONLY  
5 READ ONLY  
6 WRITE/READ CHECK ON ALTERNATE DRIVES  
7 WRITE/READ/READ CHECK \*

; \* NOTE: THE FIRST HALF OF TEST 7 FORCES THE TRACK SEQUENCE



393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448

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

S = TRACK SEQUENCING  
0 DO SEQUENCE 7 (RANDOM)  
1 INCREMENT  
2 DECREMENT  
3 INCREMENT/DECREMENT  
4 BOUNCE  
5 DECREASING BOUNCE  
6 STROBE  
7 RANDOM

;;\*\*\*\*\*

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

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

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

;;\*\*\*\*\*

SA200:  
001220 000005  
001222 012737 177570 001214  
001230 012706 001200  
001234 012746 000340  
001240 012746 001246  
001244 000002  
2\$:  
001246 104400 016554  
001252 012737 001272 000004  
001260 022777 177777 177726  
001266 001402  
001270 000423  
3\$:  
001272 022626  
447 001274 012737 000176 001214  
448 001302 012737 000174 001216

RESET ;INITIALIZE THE RX01  
MOV #177570,SWR ;RESET TO HARDWARE SWR.  
MOV #STACK,SP  
MOV #PR7,-(SP)  
MOV #2\$,-(SP)  
RTI  
TYPE MREV ;PRINT NAME AND REVISION  
MOV #3\$,4 ;SET TIME OUT VECTOR  
CMP #177777,SWR ;IS SOFTWARE SWR SELECTED  
BEQ 4\$ ;YES, INSERT IT'S ADDRESS  
BR 5\$ ;BRANCH IF NO TIMEOUT TRAP OCCURS  
CMP (SP)+,(SP)+ ;RESTORE STACK AFTER TRAP  
MOV #SWREG,SWR ;POINT TO SOFTWARE SWITCH REG.  
MOV #DISPREG,DISPLAY ;POINT TO SOFTWARE DISP. REG.



```

449 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
450 001310 005737 000042 TST J#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
451 001314 001006 BNE 64$ ;;BRANCH IF YES
452 001316 023727 001214 000176 CMP SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
453 001324 001005 BNE 65$ ;;BRANCH IF NO
454 001326 104404 GTSWR ;;GET SOFT-SWR SETTINGS
455 001330 000403 BR 65$
456 001332 112737 000001 013714 64$: MOVB #1,$AUTOB ;;SET AUTO-MODE INDICATOR
457 001340 65$:
458 001340 012737 000006 000004 5$: MOV #6,4 ;RESTORE ERROR VECTOR
459 001346 013703 001204 MOV INTVEC,R3 ;SET INTERRUPT VECTOR LOCATIONS
460 001352 012723 007250 MOV #INTSERV,(R3)+
461 001356 012713 000340 MOV #340,(R3)
462 001362 012737 001234 012210 MOV #001234,RAN1 ;INITIALIZE CONSTANTS IN RANDOM NUMBER GENERATOR
463 001370 012737 000765 012212 MOV #000765,RAN2
464 001376 012705 017024 MOV #PARLOG,R5 ;SET UP TO CLEAR ALL COUNTERS
465 001402 005025 1$: CLR (R5)+ ;OF ERRORS,ACCESSES,AND PASSES
466 001404 022705 021044 CMP #PASCNTR+2,R5 ;HAVE ALL COUNTERS BEEN CLEARED
467 001410 001374 BNE 1$
468 001412 012705 017004 START1: MOV #RDRETRY,R5 ;SET ALL RETRY COUNTERS TO 10
469 001416 012725 000012 3$: MOV #10,(R5)+
470 001422 022705 017024 CMP #SRETRY+2,R5
471 001426 001373 BNE 3$
472 001430 104400 015167 TYPE ,MRXCS ;TYPE OUT DEVICE REGISTER ADDRESSES
473 001434 013746 001206 MOV RXCS,-(SP) ;;SAVE RXCS FOR TYPEOUT
474 001440 104402 TYPOS ;;GO TYPE--OCTAL ASCII
475 001442 006 .BYTE 6 ;;TYPE 6 DIGITS
476 001443 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
477 001444 104400 015177 TYPE ,MRXDB
478 001450 013746 001210 MOV RXDB,-(SP) ;;SAVE RXDB FOR TYPEOUT
479 001454 104402 TYPOS ;;GO TYPE--OCTAL ASCII
480 001456 006 .BYTE 6 ;;TYPE 6 DIGITS
481 001457 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
482 001460 104400 015150 TYPE ,MINTVEC ;AND VECTOR ADDRESS
483 001464 013746 001204 MOV INTVEC,-(SP) ;;SAVE INTVEC FOR TYPEOUT
484 001470 104402 TYPOS ;;GO TYPE--OCTAL ASCII
485 001472 003 .BYTE 3 ;;TYPE 3 DIGIT(S)
486 001473 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
487 001474 004737 007122 JSR PC,DONECK ;WAIT FOR DONE BIT AFTER RECAL
488 001500 032777 000004 177502 BIT #BIT2,$RXDB ;IS INIT DONE SET
489 001506 001002 BNE 1$ ;YES SKIP NEXT INSTRUCTION
490 001510 104400 015535 TYPE ,INIT1 ;NO,PRINT NO INIT DONE ERROR
491 001514 005777 177466 1$: TST $RXCS ;WAS THERE AN ERROR ON START RECAL
492 001520 100002 BPL TESTDR ;IF NOT CONTINUE
493 001522 004737 002550 JSR PC,HOME ;YES,DO RECAL AGAIN
494 001526 012737 000012 017014 TESTDR: MOV #10,$PRETRY ;SET THE PARITY RETRY COUNTER
495 001534 005037 010704 CLR UNITSEL ;CLEAR UNIT SELECTION WORD
496 001540 105777 177444 TSTB $RXDB ;IS DRY BIT SET FOR UNIT 0 AFTER RECAL
497 001544 100003 BPL 2$ ;IF NOT SKIP NEXT INSTRUCTION
498 001546 052737 000200 010704 BIS #BIT7,UNITSEL ;YES,SET UNIT 0 READY BIT
499 001554 032737 140000 001212 2$: BIT #140000,DTESTP ;WERE ANY DRIVES SPECIFIED
500 001562 001445 BEQ NOSEL ;NO,SEE IF UNIT 1 IS READY
501 001564 005737 001212 TST DTESTP ;WAS UNIT 1 SELECTED
502 001570 100044 BPL OPCOND ;NO,IT MUST BE UNIT 0
503 001572 004737 001614 JSR PC,DRY1 ;SEE IF UNIT 1 IS READY
504 001576 032737 040000 001212 BIT #40000,DTESTP ;WAS UNIT 0 SELECTED

```



```

505 001604 001036          BNE OPCOND          ;YES,CONTINUE
506 001606 105037 010704  CLRB UNITSEL      ;NO,CLEAR UNIT 0 READY BIT
507 001612 000433          BR OPCOND
508
509 001614 012777 000033 177364 DRY1:  MOV #RD1STAT,DRXCS ;READ STATUS OF UNIT 1
510 001622 004737 007122          JSR PC,DONECK      ;WAIT FOR DONE BIT
511 001626 132777 000002 177354  BITB #2,DRXDB     ;IS PARITY ERROR BIT SET
512 001634 001411          BEQ 2$            ;NO,CONTINUE
513 001636 004737 002046          JSR PC,STATER     ;YES,GO HANDLE PARITY ERROR
514 001642 000764          BR DRY1          ;REISSUE FUNCTION
515 001644 032777 000004 177336  BIT #BIT2,DRXDB  ;INIT DONE SHOULD NOT BE SET
516 001652 001402          BEQ 2$
517 001654 104400 015567          TYPE ,MINIT2     ;INIT DONE WAS SET PRINT ERROR
518 001660 105777 177324          TSTB DRXDB       ;IS UNIT 1 DRY BIT SET
519 001664 100003          BPL 3$           ;IF NOT SKIP NEXT INSTRUCTION
520 001666 052737 100000 010704  BIS #BIT15,UNITSEL ;YES,SET UNIT 1 READY BIT
521 001674 000207          RTS PC
522
523 001676 004737 001614          JSR PC,DRY1      ;TEST UNIT 1 FOR DRY SET
524 001702 105737 010704          TSTB UNITSEL    ;IS UNIT 0 SELECTED
525 001706 100016          BPL 1$          ;NO
526
527          ;TEST ACT1! LOAD MEDIA INDICATOR
528
529 001710 123727 000041 000010  CMPB #41,#10     ;DOES LOCATION 41 CONTAIN THE NUMBER 10?
530 001716 001012          BNE 1$          ;BRANCH IF NOT
531 001720 005737 000042          TST #42         ;CHECK FOR RXDP OPERATION
532 001724 001404          BEQ 2$
533 001726 042737 000200 010704  BIC #200,UNITSEL ;IN CHAIN MODE, DESELECT UNIT 0
534 001734 000403          BR 1$          ;AND DO NOT HALT
535 001736 104400 014643          TYPE, DOLOAD    ;INFORM USER TO REMOVE LOAD MEDIUM
536          ;FROM UNIT 0 AND REPLACE WITH
537          ;A 'SCRATCH' DISKETTE IF HE
538 001742 000000          HALT           ;WISHES TO TEST UNIT 0
539 001744 000241          CLC           ;CLEAR C BIT FOR ROR'S THAT FOLLOW
540 001746 013737 001212 011370 1$:  MOV DTESTP,SEQUEN ;PUT INITIAL CONDITIONS IN SEQUENCE
541 001754 042737 177770 011370  BIC #177770,SEQUEN ;CLEAR ALL BUT SEQUENCE BITS
542 001762 013737 001212 002140  MOV DTESTP,TEST  ;PUT INITIAL CONDITIONS IN TEST
543 001770 042737 177707 002140  BIC #177707,TEST ;CLEAR ALL BUT TEST BITS
544 001776 006037 002140          ROR TEST       ;RIGHT JUSTIFY TEST BITS
545 002002 006037 002140          ROR TEST
546 002006 006037 002140          ROR TEST
547 002012 013737 001212 010420  MOV DTESTP,PAT  ;PUT INITIAL CONDITIONS IN PATTERN
548 002020 006137 010420          ROL PAT        ;RIGHT JUSTIFY PATTERN BITS
549 002024 006137 010420          ROL PAT
550 002030 000337 010420          SWAB PAT
551 002034 042737 177770 010420  BIC #177770,PAT ;CLEAR ALL BUT PATTERN BITS
552 002042 000137 002722          JMP XSTART
553
554 002046 005237 017024          STATER: INC PARLOG ;INC PARITY ERROR COUNTER
555 002052 104400 015167          TYPE ,MRXCS    ;PRINT THE RXCS CONTENTS
556 002056 017746 177124          MOV DRXCS,-(SP) ;SAVE DRXCS FOR TYPEOUT
557 002062 104402          TYPOS         ;GO TYPE--OCTAL ASCII
558 002064 003          .BYTE 3       ;TYPE 3 DIGIT(S)
559 002065 000          .BYTE 0       ;SUPPRESS LEADING ZEROS
560 002066 005337 017014          DEC P2RETRY    ;HAVE 10 ERRORS OCCURED

```











```

673 002656 104400 015713          TYPE ,MHALT3          ;YES,CAN'T CONTINUE AS RX11 WILL NOT RECAL
674 002662 000000          HLT3:          HALT
675 002664 000137 001220          JMP SA200          ;IF CONT. SWITCH IS PRESSED GO TO START
676 002670 012737 000001 011362 XHOME:          MOV #1,PRESTRK      ;RESET THE PRESENT TRACK TO TRACK 1
677 002676 000207          RTN:          RTS PC          ;RETURN

```

;;\*\*\*\*\*

```

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

```

BITS			TESTS
5	4	3	
0	0	0	NO TEST SELECTED (DEFAULT TO TEST 7)
0	0	1	WRITE ONLY
0	1	0	WRITE FOLLOWED BY READ
0	1	1	WRITE FOLLOWED BY READ AND VERIFY
1	0	0	READ AND VERIFY ONLY
1	0	1	READ ONLY (CRC CHECK)
1	1	0	READ AND VERIFY DELETED DATA
1	1	1	WRITE FOLLOWED BY READ FOLLOWED BY ANOTHER READ AND VERIFY

;;\*\*\*\*\*

```

700 002700 012706 001200          RESTART:          MOV #STACK,SP      ;RESET THE STACK POINTER
701 002704 012746 000340          MOV #PR7,-(SP)    ;AND INTERRUPT LEVEL
702 002710 012746 002716          MOV #1$,-(SP)
703 002714 000002          RTI
704 002716 005237 021040          1$:          INC RESTCNTR      ;INC RESTART COUNTER
705 002722 004737 002452          XSTART:          JSR PC,ICOND      ;TYPE OUT INITIAL CONDITIONS
706 002726 005037 011142          TESTSEL:          CLR CHARCT        ;CLEAR EOP CHARACTER COUNTER
707 002732 012737 000012 017004          MOV #10.,RDRETRY ;SET UP READ AND WRITE RETRY COUNTERS
708 002740 012737 000012 017006          MOV #10.,WTRTRY
709 002746 005037 004124          CLR ERWRT        ;CLEAR WRITE CAUSED BY ERROR FLAG
710 002752 005037 004122          CLR RDAFTWT      ;CLEAR READ AFTER WRITE FLAG
711 002756 142737 000377 003014          BICB #377,2#BRONTEST ;CLEAR OUT BRANCH OFFSET
712 002764 005737 002140          TST TEST         ;IF NO TEST SPECIFIED FORCE TEST 7
713 002770 001003          BNE 1$
714 002772 012737 000007 002140          1$:          MOV #7,TEST
715 003000 013704 002140          MOV TEST,R4
716 003004 005304          DEC R4           ;ADJUST TEST BITS FOR CORRECT
717 003006 006304          ASL R4           ;BRANCH OFFSET
718 003010 150437 003014          BRONTEST:          BISB R4,2#BRONTEST ;INSERT OFFSET TO BRANCH INST.
719 003014 000777          BR              ;BRANCH BY TEST OFFSET
720 003016 000137 003052          JMP WRONLY       ;WRITE ONLY FUNCTION
721 003022 000137 004130          JMP WRTRD        ;WRITE THEN READ FUNCTION
722 003026 000137 005456          JMP WTRDCK       ;WRITE THEN READ VERIFY
723 003032 000137 006444          JMP RDCHK        ;READ VERIFY
724 003036 000137 006504          JMP RDONLY       ;READ ONLY FUNCTION
725 003042 000137 006544          JMP DRVSWP       ;WRITE,AND READ VERIFY ON ALTERNATING DRIVES
726 003046 000137 006630          JMP TEST7        ;WRITE,READ,FOLLOWED BY READ VERIFY
727
728

```



```

729
730 ;*****
731 ;
732 ; THIS IS A WRITE ONLY FUNCTION USING DATA PATTERN AND
733 ; TRACK SEQUENCE SPECIFIED BY INITIAL SWR SETTINGS
734 ; T = 1
735
736 003052 004737 010312 WRONLY: JSR PC,GETPATTERN ;SET UP SOFTWARE DATA BUFFER
737 003056 004737 011144 XWRONLY: JSR PC,INITTRACKS ;SET UP ID,OD,AND TRACK COUNTER
738 003062 004737 010614 JSR PC,GETUNIT ;SET UP DRIVE UNIT SELECTION
739 003066 004737 011254 1$: JSR PC,GETTRACK ;PICK UP NEXT TRACK
740 003072 004737 003112 JSR PC,WRITE ;DO THE WRITE FUNCTION
741 003076 005337 011356 DEC TRKCNTR ;TEST TRACK COUNTER
742 003102 001371 BNE 1$
743 003104 004737 010732 JSR PC,STOP ;CHECK FOR LAST DRIVE AND EOP
744 003110 000762 BR XWRONLY ;NEXT PASS
745
746 003112 004737 012216 WRITE: JSR PC,INITSECTOR ;SET UP FIRST, LAST AND SECTOR COUNTER
747 003116 004737 012316 XWRITE: JSR PC,GETSECTOR ;PICK UP NEXT SECTOR
748 003122 012737 000012 017022 MOV #10.,SRETRY
749 003130 012737 000012 017016 ONEWRT: MOV #10.,PRETRY ;SET RETRY COUNTER
750 003136 004737 005656 FILLBUF: JSR PC,ADJSUM ;ADJUST DATA BUFFER AND CHECK SUM FOR ADDRESSES
751 003142 012746 003372 MOV #FILLDONE,-(SP) ;PUT GOOD RETURN ON STACK
752 003146 012746 003214 MOV #FILLER,-(SP) ;PUT ERROR RETURN ON STACK
753 003152 005037 006434 CLR BYTECNTR
754 003156 012746 000200 MOV #PR4,-(SP)
755 003162 012746 003170 MOV #1$,-(SP)
756 003166 000002 RTI
757 003170 012777 000101 176010 1$: MOV #FBIE,DRXCS ;EXECUTE FILLBUFFER COMMAND
758 003176 004737 007014 FILLFLAG: JSR PC,TRCK ;TEST FOR TRANSFER REQUEST FLAG
759 003202 112077 176002 XFRBYTE: MOVB (RD)+,DRXDB ;TRANSFER DATA BYTE
760 003206 005237 006434 INC BYTECNTR
761 003212 000771 BR FILLFLAG ;WAIT FOR NEXT TR FLAG
762
763 003214 005726 FILLER: TST (SP)+ ;REMOVE THE DONE RETURN FROM THE STACK
764 003216 012737 015366 003264 MOV #MFIL,PTYP1+2 ;PUT ADDR OF FILLBUF MESSAGE IN PAR ERR TYP0UT 1
765 003224 012737 015366 003354 MOV #MFIL,PTYP2+2 ;PUT ADDR OF FILLBUF MESSAGE IN PAR ERR TYP0UT 2
766 003232 012737 003136 003334 MOV #FILLBUF,PCONT+2 ;IF NOT HARD ERR RETURN THROUGH PCONT TO FILLBUF
767 003240 000137 003244 JMP PARTEST ;PRINT OUT PAR ERR AND TEST CONDITIONS FOR RETRY
768
769
770 003244 005237 017024 PARTEST: INC PARLOG ;INCREMENT PARITY ERROR
771 003250 104405 CKSWR
772 003252 032777 020000 175734 BIT #SW13,DSWR ;TEST DON'T PRINT ERROR SWITCH
773 003260 001006 BNE CONT4
774 003262 104400 000000 PTYP1: TYPE ,OFEN ;PRINT THE PARITY ERROR MESSAGE
775 003266 104400 016017 TYPE ,MPAR
776 003272 104400 015052 TYPE ,MCRLF
777 003276 104405 CONT4: CKSWR
778 003300 005777 175710 TST DSWR ;TEST HALT ON ERROR SWITCH
779 003304 100001 BPL CONT13
780 003306 000000 HLT6: HALT ;HALT ON ERROR
781 003310 032777 004000 175676 CONT13: BIT #SW11,DSWR ;TEST NO RETRY SWITCH
782 003316 001007 BNE CONT5 ;IF SET LOG HARD ERROR
783 003320 005337 017016 DEC PRETRY ;DECREMENT RETRY COUNTER
784 003324 005737 017016 TST PRETRY ;HAVE 10 ERRORS OCCURED
    
```



```

785 003330 001402          BEQ CONTS          ;IF CLEARED LOG HARD ERROR
786 003332 000137 003136 PCONT:          JMP FILLBUF      ;RETURN TO RETRY TEST THROUGH HERE
787 003336 005237 017026 CONTS:          INC HPARLOG      ;INC.HARD PARITY ERROR COUNTER
788 003342 104400 015240          TYPE ,DBLIF
789 003346 104400 015755          TYPE ,MUNREC     ;TYPE HARD PARITY ERROR
790 003352 104400 000000 PTP2:          TYPE ,OPEN
791 003356 104400 016017          TYPE ,MPAR
792 003362 104400 015052          TYPE ,MCRLF
793 003366 000137 010074          JMP DELUNIT      ;GO DELETE THE UNIT,CAN NOT CONTINUE
794
795          ;SWITCH 9 OF INITIAL TEST CONDITIONS SWITCH SETTINGS IS THE DELETED DATA
796          ;FUNCTION INDICATOR. WHEN THIS BIT IS SET ALL WRITE / READ FUNCTIONS WILL
797          ;SET AND CHECK THE DELETED DATA BIT ON THE DISKETTE.
798          ;*****
799
800 003372 012737 000012 017016 FILLDONE:      MOV #10,PRETRY   ;SET UP RETRY COUNTER
801 003400 012746 003462 REWRITE:      MOV #WRTDONE,-(SP) ;SET GOOD RETURN ON STACK
802 003404 012746 003510          MOV #WRTERR,-(SP) ;SET ERROR RETURN ON STACK
803 003410 112737 000105 004120          MOVB #WRTIE,FUNCTION ;SET FUNCTION WORD TO WRITE
804 003416 032737 001000 001212          BIT #BIT9,DTESTP ;TEST FOR WRITE DELETED DATA
805 003424 001403          BEQ IS
806 003426 112737 000115 004120          MOVB #WRTDIE,FUNCTION
807 003434 062737 000001 021030 1$:          ADD #1,WTCNTR    ;INC TOTAL WRITE FUNCTIONS COUNTER
808 003442 005537 021032          ADC WTCNTR+2    ;DOUBLE PRECISION COUNTER
809 003446 004737 004022          JSR PC,COMMWORD ;TRANSFER COMMAND TO DRIVE
810 003452 004737 007122          JSR PC,DONECK   ;TEST FOR DONE FLAG
811 003456 000137 007204          JMP NOINTER     ;NO INTERRUPT ERROR
812
813 003462 005737 004124          WRTDONE:       TST ERWRT       ;IS THIS A REWRITE FROM A DATA ERROR
814 003466 001004          BNE IS         ;YES GO REREAD THIS SECTOR
815 003470 005337 012306          DEC SECCNTR    ;NO,TEST SECTOR COUNTER
816 003474 001003          BNE 2$       ;NOT LAST SECTOR GO TO NEXT ONE
817 003476 000207          RTS PC
818 003500 000137 004246          1$:          JMP REREAD     ;REREAD THE SECTOR
819 003504 000137 003116          2$:          JMP XWRITE
820
821 003510 005726          WRTERR:       TST (SP)+       ;REMOVE THE DONE RETURN FROM THE STACK
822 003512 032737 000002 010070          BIT #BIT1,ASTAT ;IS THIS A PARITY ERROR
823 003520 001413          BEQ WRTSEK    ;NO, IT MUST BE A SEEK ERROR
824          ;PARITY ERROR DURING A WRITE FUNCTION
825 003522 012737 016010 003264          MOV #MWRITE,PTYP1+2 ;PUT ADDR OF WRITE MESSAGE IN PAR ER TYP0UT 1
826 003530 012737 016010 003354          MOV #MWRITE,PTYP2+2 ;PUT ADDR OF WRITE MESSAGE IN PAR ER TYP0UT 2
827 003536 012737 003400 003334          MOV #REWRITE,PCONT+2 ;IF NOT HARD ERR RETURN THROUGH PCONT TO REWRITE
828 003544 000137 003244          JMP PARTEST    ;GO INC LOG AND TEST FOR RETRY
829
830          ;SEEK ERROR DURING A WRITE FUNCTION
831 003550 012737 003130 003674 WRTSEK:      MOV #ONEWRT,SEKRTY+2 ;SETUP FOR WRT RETRY ON SEEK ERROR
832          ;(AFTER A RECAL. THE CONTENTS OF SECTOR 1,
833          ;TRACK 1 ARE LOADED INTO THE SECTOR BUFFER.
834          ;TO REWRITE THE CORRECT DATA THE PROGRAM
835          ;MUST REFILL THE SECTOR BUFFER.
836 003556 012737 016010 003630          MOV #MWRITE,STYP1+2 ;PUT ADDR OF WRITE MESSAGE IN SEEK ER TYPEOUT 1
837 003564 012737 016010 003722          MOV #MWRITE,STYP2+2 ;PUT ADDR OF WRITE MESSAGE IN SEEK ER TYPEOUT 2
838 003572 004737 003576          JSR PC,SEEKER  ;RECORD SEEK ERROR
839
840 003576 012703 017036          SEEKER:       MOV #ZSEKLOG,R3  ;SETUP INC INSTRUCTION FOR UNIT 0 LOG

```



841	003602	004737	021044		JSR PC,U1LOG	;TEST FOR UNIT 1 LOG
842	003606	005213			INC (R3)	;INCREMENT SEEK ERROR LOG
843	003610	004737	003770		JSR PC,TRKERR	;INC ERROR PER TRACK COUNTERS
844	003614	104405			CKSWR	
845	003616	032777	020000	175370	BIT #SW13,@SWR	;CHECK DON'T PRINT ERROR SWITCH
846	003624	001004			BNE SWHLT1	
847	003626	104400	016010	STYP1:	TYPE ,MWRITE	;PRINT WRITE (READ) SEEK ERROR
848	003632	004737	003736		JSR PC,SEKTYP	
849	003636	104405		SWHLT1:	CKSWR	
850	003640	005777	175350		TST @SWR	;TEST THE HALT ON ERROR SWITCH
851	003644	100001			BPL CONT14	
852	003646	000000		HLT7:	HALT	;HALT ON THE ERROR
853	003650	032777	004000	175336	CONT14:	;CHECK THE NO RETRY SWITCH
854	003656	001007			BIT #SW11,@SWR	;IF SET LOG HARD SEEK ERROR
855	003660	005337	017022		BNE HARDSK	;HAVE 10 ERRORS BEEN LOGED
856	003664	001404			DEC SRETRY	;YES LOG HARD ERROR
857	003666	004737	002550		BEQ HARDSK	;RECALIBRATE DRIVES ON SEEK ERROR
858	003672	000137	003130	SEKRTY:	JSR PC,HOME	;NO RETRY WRITE COMMAND (READ COMAND)
859	003676	012703	017076	HARDSK:	JMP ONEWRT	;SET INC. INST.FOR UNIT 0 ERR LOG
860	003702	004737	021044		MOV #ZHSEKLOG,R3	;TEST FOR UNIT 1 ERROR LOG
861	003706	005213			JSR PC,U1LOG	;HARD SEEK ERROR
862	003710	104400	015240		INC (R3)	
863	003714	104400	015755		TYPE ,DBLLF	;TYPE UNRECOVERABLE SEEK ERROR
864	003720	104400	016010	STYP2:	TYPE ,MUNREC	;ON WRITE (READ) COMMAND
865	003724	004737	003736		TYPE ,MWRITE	
866	003730	062706	000002		JSR PC,SEKTYP	
867	003734	000207			ADD #2,SP	;REMOVE SEEK ERROR FROM STACK POINTER
868					RTS PC	;RETURN TO NEXT SECTOR BY DONE RETURN ON STACK
869	003736	104400	015775	SEKTYP:	TYPE ,MSEEK	;TYPE SEEK ERROR
870	003742	104400	014567		TYPE ,MPRES	;TYPE ADDRESS OF TRACK MOVED FROM
871	003746	013737	011362	003760	MOV PRESTRK,1\$	
872	003754	004537	014446		JSR R5,SGLDEC	
873	003760	000000		1\$:	OPEN	
874	003762	104400	015052		TYPE ,MCRLF	
875	003766	000207			RTS PC	
876						
877	003770	013705	011360	TRKERR:	MOV TARGET,R5	;SET UP TO INC ERROR PER TRACK COUNTER
878	003774	006305			ASL R5	;ADJUST FOR EVEN ADDRESS
879	003776	062705	020344		ADD #UOTRK,R5	;ADDIN ADDRESS OF UNIT 0 LOG
880	004002	032737	000020	010704	BIT #BIT4,UNITSEL	;CHECK THE UNIT SELECTION BIT
881	004010	001402			BEQ 1\$	;IF CLEARED UNIT 0 IS ACTIVE
882	004012	062705	020576		ADD #U1TRK,R5	;ADJUST FOR UNIT 1 LOG
883	004016	005215		1\$:	INC (R5)	;INC THE CORRECT COUNTER
884	004020	000207			RTS PC	
885						
886	004022	013705	011360	COMMWORD:	MOV TARGET,R5	;GET TRACK NUMBER
887	004026	006305			ASL R5	;MULTIPLY BY 4 TO DOUBLE PRECISION
888	004030	006305			ASL R5	;INTERLEAVE COUNTER LOCATIONS
889	004032	062705	017174		ADD #TKACC,R5	;ADD ON ADDRESS OF TRACK ACCESS COUNTER
890	004036	062715	000001		ADD #1,(R5)	;INCREMENT THE COUNTER
891	004042	005565	000002		ADC 2(R5)	;ADD CARRY TO HIGH ORDER WORD
892	004046	153737	010704	004120	BISB UNITSEL,FUNCTION	;SET UNIT SELECTION BIT IN COMMAND WORD
893	004054	013777	004120	175124	MOV FUNCTION,@RXCS	;SEND OUT COMMAND TO DRIVE
894	004062	004737	007014		JSR PC,TRCK	;WAIT FOR TR FLAG
895	004066	113777	012310	175114	MOVB TSECTOR,@RXDB	;SEND OUT TARGET SECTOR
896	004074	004737	007014		JSR PC,TRCK	;WAIT FOR TR FLAG



```

897 004100 113777 011360 175102          MOVB TARGET,DRXDB          ;SEND OUT TARGET TEACK
898 004106 005046                      CLR -(SP)                  ;LOWER INTERUPT LEVEL TO ALLOW AN INTERUPT
899 004110 012746 004116          MOV #1$,-(SP)
900 004114 000002                      RTI
901 004116 000207          IS:          RTS PC
902
903 004120 000000          FUNCTION:          0
904 004122 000000          RDAFTWT:          0          ;READ AFTER WRITE FUNCTION FLAG
905 004124 000000          ERWRT:          0          ;WRITE CAUSED BY DATA ERROR FLAG
906 004126 000000          DATAACK:        0          ;DATA CHECK ON CRC ERROR FLAG
907
908          ;;*****
909
910          ;THIS IS A WRITE ALL SECTORS FOLLOWED BY A READ ALL SECTORS
911          ;WITH DATA PATTERNS AND HEAD SEQUENCING SET BY INITIAL SWR
912          ; T = 2
913
914 004130 005137 004122          WRTRD:          COM RDAFTWT          ;SET READ AFTER WRITE FLAG
915 004134 004737 010312          JSR PC,GETPATTERN
916 004140 004737 011144          XWRTRD:        JSR PC,INITTRACKS
917 004144 004737 010614          JSR PC,GETUNIT
918 004150 004737 011254          IS:          JSR PC,GETTRACK
919 004154 004737 003112          JSR PC,WRITE
920 004160 004737 004200          JSR PC,READ
921 004164 005337 011356          DEC TRKNTR
922 004170 001367          BNE IS
923 004172 004737 010732          JSR PC,STOP
924 004176 000760          BR XWRTRD
925
926          ;;*****
927
928          ;READ DATA FROM THE DISKETTE
929
930
931 004200 004737 012216          READ:          JSR PC,INITSECTOR
932 004204 004737 012316          XREAD:        JSR PC,GETSECTOR
933 004210 012737 000012 017010          MOV #10.,DDRETRY          ;SET UP RETRY COUNTERS FOR DELETED DATA
934 004216 012737 000012 017012          MOV #10.,DATARETRY          ;DATA ERROR
935 004224 012737 000012 017016          MOV #10.,PRETRY          ;PARITY
936 004232 012737 000012 017022          MOV #10.,SRETRY          ;SEEK
937 004240 012737 000012 017020          MOV #10.,CRETRY          ;AND CRC ERRORS
938 004246 005037 004126          REREAD:        CLR DATAACK          ;CLEAR CRC DATA CHECK FLAG
939 004252 012746 004316          MOV #RDDONE,-(SP)          ;SET GOOD RETURN ON STACK
940 004256 012746 004460          MOV #RDERR,-(SP)          ;SET READ ERROR RETURN ON STACK
941 004262 112737 000107 004120          MOVB #RDIE,FUNCTION
942 004270 062737 000001 021034          ADD #1,RDCNTR          ;INC TOTAL READ FUNCTIONS COUNTER
943 004276 005537 021036          ADC RDCNTR+2          ;DOUBLE PRECISION COUNTER
944 004302 004737 004022          JSR PC,COMMWORD
945 004306 004737 007122          JSR PC,DONECK          ;TEST FOR DONE
946 004312 000137 007204          JMP NOINTER          ;NO INTERUPT ON DONE
947
948
949
950 004316 022737 000012 017004  RDDONE:  CMP #10.,RDRETRY          ;IS READ RETRY EQUAL TO 10
951 004324 001420          BEQ CONT1          ;YES,NO ERRORS OCCURED
952 004326 012703 017052          MOV #ZRDL0G,R3          ;SET INC. INST. FOR UNIT 0 ERROR LOG
  
```



953	004332	004737	021044			JSR PC,U1LOG	;TEST FOR UNIT 1 ERROR LOG
954	004336	005213				INC (R3)	;INC RECOVERABLE READ LOG
955	004340	012737	000012	017004		MOV #10.,RDRETRY	;RESET READ RETRY COUNTER
956	004346	104400	015240			TYPE ,DBLLF	
957	004352	104400	015627			TYPE ,MREC	;TYPE RECOVERABLE READ ERROR
958	004356	104400	015674			TYPE ,MREAD	
959	004362	104400	015052			TYPE ,MCRLF	
960	004366	022737	000012	017006	CONT1:	CMP #10.,WTRETRY	;IS WRITE RETRY EQUAL TO 10
961	004374	001420				BEQ CONT2	;YES,NO ERRORS OCCURED
962	004376	012703	017056			MOV #ZWRLOG,R3	;SET INC.INST.FOR UNIT 0 ERROR LOG
963	004402	004737	021044			JSR PC,U1LOG	;TEST FOR UNIT 1 LOG
964	004406	005213				INC (R3)	;INC RECOVERABLE WRITE LOG
965	004410	012737	000012	017006		MOV #10.,WTRETRY	;RESET THE WRITE RETRY COUNTER
966	004416	104400	015240			TYPE ,DBLLF	
967	004422	104400	015627			TYPE ,MREC	;TYPE RECOVERABLE WRITE ERROR
968	004426	104400	016010			TYPE ,MWRITE	
969	004432	104400	015052			TYPE ,MCRLF	
970	004436	004737	005204		CONT2:	JSR PC,DDCHK	;CHECK FOR DELETED DATA INDICATOR
971	004442	005701				TST R1	;BIT 15 OF R1 IS READ 1 SECTOR FLAG
972	004444	100001				BPL NEXTRD	
973	004446	000207				RTS PC	;IF SET,GO VERIFY DATA JUST READ
974	004450	005337	012306		NEXTRD:	DEC SECCNTR	
975	004454	001253				BNE XREAD	
976	004456	000207				RTS PC	;READ FUNCTION IS DONE
977							
978	004460	005726			RDERR:	TST (SP)+	;REMOVE THE DONE RETURN FROM THE STACK
979	004462	032737	000002	010070		BIT #BIT1,ASTAT	;IS THIS A PARITY ERROR
980	004470	001413				BEQ 1\$	;NO,SEE IF ITS A CRC ERROR
981						;PARITY ERROR DURING A READ FUNCTION	
982	004472	012737	015674	003264		MOV #MREAD,PTYP1+2	;PUT ADDR OF READ MESSAGE IN PAR ERR TYPEOUT 1
983	004500	012737	015674	003354		MOV #MREAD,PTYP2+2	;PUT ADDR OF READ MESSAGE IN PAR ER TYPEOUT 2
984	004506	012737	004246	003334		MOV #REREAD,PCONT+2	;IF HARD ERR RETURN THROUGH PCONT TO REREAD
985	004514	000137	003244			JMP PARTEST	;RECORD PARITY ERROR AND RETRY FUNCTION
986	004520	032737	000001	010070	1\$:	BIT #BIT0,ASTAT	;IS THIS A CRC ERROR
987	004526	001014				BNE CRCER	;YES GO TEST AND LOG IT
988						;SEEK ERROR DURING A READ FUNCTION	
989	004530	012737	004246	003674		MOV #REREAD,SEKRTY+2	;SET SEEK CONTINUE FOR READ RETRY
990	004536	012737	015674	003630		MOV #MREAD,STYP1+2	;SET ADDR OF READ MESSAGE IN SEEK ER TYPEOUT 1
991	004544	012737	015674	003722		MOV #MREAD,STYP2+2	;SET ADDR OF READ MESSAGE IN SEEK ER TYPEOUT 2
992	004552	004737	003576			JSR PC,SEEKER	;RECORD SEEK ERROR
993	004556	000734				BR NEXTRD	;GO TO NEXT SECTOR,CAN'T READ THIS ONE
994						;CRC ERROR DETECTED WHILE READING	
995	004560	004737	003770		CRCER:	JSR PC,TRKERR	;INC ERROR PER TRACK COUNTER
996	004564	005701				TST R1	;IF READ ONLY, REPORT DATA CRC ERROR
997	004566	100061				BPL DATACRC	
998	004570	005237	004126			INC DATACK	;SET DATA CHECK FLAG
999	004574	004737	005536			JSR PC,EMPBUFF	;CHECK FOR A DATA ERROR
1000	004600	005737	006442			TST ERCNTR	;WAS THERE A DATA ERROR
1001	004604	001052				BNE DATACRC	;YES,REREAD AND/OR REWRITE THE DATA
1002	004606	012703	017046			MOV #ZCRCBAD,R3	;SET INC.INST.FOR UNIT 0 ERROR LOG
1003	004612	004737	021044			JSR PC,U1LOG	;TEST FOR UNIT 1 ERROR LOGS
1004	004616	005213				INC (R3)	;NO,INC BAD CRC GENERATOR ERROR
1005	004620	104405				CKSWR	
1006	004622	032777	020000	174364		BIT #SW13,JSWR	;TEST DON'T SWITCH
1007	004630	001004				BNE 2\$	
1008	004632	104400	015644			TYPE ,MBADCRC	;TYPE CRC GENERATOR ERROR



1009	004636	104400	015052		TYPE ,MCRLF		
1010	004642	104405		2\$:	CKSWR		
1011	004644	005777	174344		TST @SWR	;TEST HALT ON ERROR SWITCH	
1012	004650	100001			BPL CONT15		
1013	004652	000000		HLT10:	HALT	;HALT ON ERROR	
1014	004654	032777	004000	174332	CONT15:	BIT #SW11,@SWR	;CHECK NO RETRY SWITCH
1015	004662	001005			BNE 3\$	;IF SET LOG HARD ERROR	
1016	004664	005337	017020		DEC CRETRY	;HAVE 10 ERRORS BEEN LOGED	
1017	004670	001402			BEQ 3\$	;YES,LOG HARD ERROR	
1018	004672	000137	004246		JMP REREAD	;NO,GO REREAD DATA	
1019	004676	012703	017106	3\$:	MOV #ZHCRCBAD,R3	;SET INC.INST.FOR UNIT 0 ERROR LOG	
1020	004702	004737	021044		JSR PC,U1LOG	;TEST FOR UNIT 1 ERROR LOGS	
1021	004706	005213			INC (R3)		
1022	004710	104400	015240		TYPE ,DBLLF		
1023	004714	104400	015644		TYPE ,MBADCRC	;TYPE HARD CRC GENERATOR ERROR	
1024	004720	104400	015052		TYPE ,MCRLF		
1025	004724	104400	015724		TYPE ,MHALT11	;AND HALT AS THE CRC GENERATOR DOESN'T WORK	
1026	004730	000000		HLT11:	HALT		
1027						;DATA CRC ERROR REREAD AND/OR REWRITE TO GET GOOD DATA	
1028	004732	012703	017042		DATA CRC:	MOV #ZCRCLOG,R3	;SET INC.INST.FOR UNIT 0 ERROR LOG
1029	004736	004737	021044			JSR PC,U1LOG	;TEST FOR UNIT 1 ERROR LOGS
1030	004742	005213				INC (R3)	;TRUE DATA CRC ERROR
1031	004744	104405			CKSWR		
1032	004746	032777	020000	174240	BIT #SW13,@SWR	;TEST DON'T PRINT ERROR SWITCH	
1033	004754	001004			BNE 4\$		
1034	004756	104400	015736		TYPE ,MCRC	;TYPE DATA CRC ERROR	
1035	004762	104400	015052		TYPE ,MCRLF		
1036	004766	104405		4\$:	CKSWR		
1037	004770	005777	174220		TST @SWR	;TEST HALT ON ERROR SWITCH	
1038	004774	100001			BPL CONT16		
1039	004776	000000		HLT12:	HALT	;HALT ON ERROR	
1040	005000	032777	004000	174206	CONT16:	BIT #SW11,@SWR	;TEST NO RETRY SWITCH
1041	005006	001005			BNE 5\$	;IF SET LOG HARD ERROR	
1042	005010	005337	017004		DEC RDRETRY	;HAVE 10 ERRORS OCCURED	
1043	005014	001402			BEQ 5\$	;YES LOG HARD ERROR OR REWRITE DATA	
1044	005016	000137	004246		JMP REREAD	;NO,GO REREAD THIS SECTOR	
1045	005022	012703	017102	5\$:	MOV #ZHCRCLOG,R3	;SET INC.INST.FOR UNIT 0 ERROR LOG	
1046	005026	004737	021044		JSR PC,U1LOG	;TEST FOR UNY 1 ERROR LOGS	
1047	005032	005213			INC (R3)	;INC HARD CRC ERROR LOG	
1048	005034	012737	000012	017004	MOV #10.,RDRETRY	;RESET READ RETRY COUNTER	
1049	005042	005737	004122		TST RDAFTWT	;IS THIS A READ AFTER WRITE FUNCTION	
1050	005046	001021			BNE CONT6	;YES,GO REWRITE IT	
1051	005050	012703	017112		MOV #ZHRDLOG,R3	;NO,SET INC.INST.FOR UNIT 0 ERROR LOG	
1052	005054	004737	021044		JSR PC,U1LOG	;TEST FOR UNIT 1 ERROR LOGS	
1053	005060	005213			INC (R3)	;READ ONLY, HARD READ ERROR	
1054	005062	104400	015240		TYPE ,DBLLF		
1055	005066	104400	015755		TYPE ,MUNREC	;TYPE UNRECOVERABLE READ ERROR	
1056	005072	104400	015674		TYPE ,MREAD		
1057	005076	104400	015052		TYPE ,MCRLF		
1058	005102	062706	000002		CONT7:	ADD #2,SP	;REMOVE READ DONE ADDRESS FROM STACK
1059	005106	000137	004450		JMP NEXTRD	;READ NEXT SECTOR CAN'T READ THIS ONE	
1060	005112	104405			CKSWR		
1061	005114	032777	004000	174072	CONT6:	BIT #SW11,@SWR	;TEST NO RETRY SWITCH
1062	005122	001011			BNE 7\$	;IF SET LOG HARD ERROR	
1063	005124	005337	017006		DEC WRETRY	;HAVE 10 REWRITES BEEN DONE	
1064	005130	001406			BEQ 7\$	;YES,LOG HARD WRITE ERROR	



1065	005132	005137	004124			COM ERWRT	;NO, SET THE WRITE CAUSED BY ERROR FLAG
1066	005136	062706	000002			ADD#2, SP	;REMOVE RDDONE ADDRESS FROM STACK
1067	005142	000137	003130			JMP ONEWRT	;REWRITE SECTOR THEN REREAD TO CHECK FOR ERROR
1068	005146	012703	017116			MOV #ZHWRTLOG, R3	;SET INC. INST. FOR UNIT 0 ERROR LOG
1069	005152	004737	021044			JSR PC, U1LOG	;TEST FOR UNIT 1 ERROR LOGS
1070	005156	005213				INC (R3)	;HARD WRITE ERROR
1071	005160	012737	000012	017006		MOV #10, WTRTRY	;RESET WRITE RETRY COUNTER
1072	005166	104400	015240			TYPE ,DBLLF	
1073	005172	104400	015755			TYPE ,MUNREC	;TYPE UNRECOVERABLE WRITE ERROR
1074	005176	104400	016010			TYPE ,MWRITE	
1075	005202	000735				BR CONT7	
1076							
1077							
1078							
1079	005204	032737	001000	001212	DDCHK:	BIT #BIT9, DTESTP	;TEST BIT 9 AS TO DELETED DATA TRANSFER
1080	005212	001472				BEQ CONT10	
1081	005214	132737	000100	010070		BITB #BIT6, ASTAT	;THIS IS A DELETED DATA FUNCTION
1082	005222	001114				BNE RETURN	;DD BIT SHOULD BE SET
1083	005224	012703	017066			MOV #ZDDMIS, R3	;SET INC INST FOR UNIT 0 ERROR LOG
1084	005230	004737	021044			JSR PC, U1LOG	;TEST FOR UNIT 1 ERROR LOGS
1085	005234	005213				INC (R3)	;INC MISSING DELETED DATA LOG
1086	005236	004737	003770			JSR PC, TRKERR	;INC ERROR PER TRACK COUNTER
1087	005242	104405				CKSWR	
1088	005244	032777	020000	173742		BIT #SW13, QSWR	;TEST DON'T PRINT ERROR SWITCH
1089	005252	001011				BNE CONT11	
1090	005254	104400	014506			TYPE ,MDDMIS	;TYPE MISSING DELETED DATA BIT
1091	005260	052737	000400	010704	DDERR:	BIS #BIT8, UNITSEL	;SET HAD ERROR FLAG
1092	005266	004737	007716			JSR PC, TYPADR	;TYPE ADDRESS OF ERROR
1093	005272	104400	015052			TYPE ,MCRLF	
1094	005276	104405			CONT11:	CKSWR	
1095	005300	005777	173710			TST QSWR	;TEST HALT ON ERROR SWITCH
1096	005304	100001				BPL CONT17	
1097	005306	000000			HLT13:	HALT	;HALT ON DELETED DATA ERROR
1098	005310	032777	004000	173676	CONT17:	BIT #SW11, QSWR	;TEST NO RETRY SWITCH
1099	005316	001005				BNE 4\$	;IF SET LOG HARD ERROR
1100	005320	005337	017010			DEC DDRETRY	;HAVE 10 ERRORS BEEN LOGED
1101	005324	001402				BEQ 4\$	;YES LOG HARD ERROR
1102	005326	000137	004246			JMP REREAD	;NO, REREAD SECTOR
1103	005332	012703	017126			MOV #ZHDDLOG, R3	;SET INC INST. FOR UNIT 0 ERROR LOG
1104	005336	004737	021044		4\$:	JSR PC, U1LOG	;TEST FOR UNIT 1 ERROR LOGS
1105	005342	005213				INC (R3)	
1106	005344	104400	015240			TYPE ,DBLLF	
1107	005350	104400	015755			TYPE ,MUNREC	;TYPE UNRECOVERABLE DELETED DATA ERROR
1108	005354	104400	015615			TYPE ,MDDER	
1109	005360	104400	015052			TYPE ,MCRLF	
1110	005364	004737	007716			JSR PC, TYPADR	
1111	005370	104400	015052			TYPE ,MCRLF	
1112	005374	000137	004450			JMP NEXTRD	;READ NEXT SECTOR
1113	005400	032737	000100	010070	CONT10:	BIT #BIT6, ASTAT	;THIS IS NOT A DELETED DATA TRANSFER
1114	005406	001422				BEQ RETURN	
1115	005410	012703	017072			MOV #ZUNXDD, R3	;SET INC INST. FOR UNIT 0 ERROR LOG
1116	005414	004737	021044			JSR PC, U1LOG	;TEST FOR UNIT 1 ERROR LOGS
1117	005420	005213				INC (R3)	;UNEXPECTED DD BIT SET
1118	005422	052737	000400	010704		BIS #BIT8, UNITSEL	;SET HAD ERROR FLAG
1119	005430	004737	003770			JSR PC, TRKERR	;INC ERROR PER TRACK COUNTER
1120	005434	104405				CKSWR	



```

1121 005436 032777 020000 173550          BIT #SW13,DSWR          ;TEST DON'T PRINT ERROR SWITCH
1122 005444 001314                                BNE CONT11
1123 005446 104400 014462          TYPE ,MUNXDD          ;TYPE UNEXPECTED DELETED DATA BIT
1124 005452 000702                                BR DDERR
1125 005454 000207          RETURN:          RTS PC
1126
1127
1128
1129          ;;*****
1130
1131          ;WRITE ALL SECTORS,READ AND VERIFY ALL SECTORS
1132          ; T = 3
1133
1134 005456 005137 004122          WTRDCK:          COM RDAFTWT          ;SET READ AFTER WRITE FLAG
1135 005462 004737 010312                                JSR PC,GETPATTERN
1136 005466 004737 011144          XWTRDCK:          JSR PC,INITTRACKS
1137 005472 004737 010614                                JSR PC,GETUNIT
1138 005476 004737 011254          1$:          JSR PC,GETTRACK
1139 005502 004737 003112                                JSR PC,WRITE
1140 005506 004737 005526                                JSR PC,READCHK
1141 005512 005337 011356                                DEC TRKCNTR
1142 005516 001367                                BNE 1$
1143 005520 004737 010732                                JSR PC,STOP
1144 005524 000760                                BR XWTRDCK
1145
1146
1147          ;;*****
1148
1149          ;READ A SECTOR,EMPTY THE SECTOR BUFFER AND VERIFY
1150          ;THE DATA READ AGAINST CORE DATA BUFFER
1151
1152 005526 052701 100000          READCHK:          BIS #BIT15,R1          ;SET READ ONE SECTOR FLAG
1153 005532 004737 004200                                JSR PC,READ          ;GO READ ONE SECTOR
1154 005536 005737 012306          EMPBUFF:          TST SECCNTR          ;IF CLEARED NO SECTOR WAS FOUND
1155 005542 001002                                BNE 2$
1156 005544 000137 006432                                JMP EXIT          ;GO TO NEXT TRACK CAN'T READ THIS ONE
1157 005550 005037 006434          2$:          CLR BYTECNTR          ;CLEAR THE BYTE AND ERROR COUNTERS
1158 005554 005037 006442                                CLR ERCNTR
1159 005560 052701 000200          BIS #BIT7,R1          ;R1 BIT 7 IS USED AS FIRST ERROR FLAG
1160 005564 004737 005656          JSR PC,ADJSUM          ;ADJUST DATA AND CK SUM FOR ADDRESSES
1161 005570 005037 005746          CLR CKSUM          ;SET UP FOR CHECK SUM ACCUMULATION
1162 005574 012746 006212          MOV #EMPDONE,-(SP)          ;SET UP RETURN ADDRESSES
1163 005600 012746 005750          MOV #EMPER,-(SP)
1164 005604 005046          CLR -(SP)          ;LOWER INTERRUPT LEVEL
1165 005606 012746 005614          MOV #1$,-(SP)
1166 005612 000002          RTI
1167 005614 012777 000103 173364 1$:          MOV #EBIE,DRXCS          ;LOAD EMPTY BUFFER FUNCTION
1168 005622 004737 007014          EMPFLAG:          JSR PC,TRCK          ;TEST FOR TR FLAG
1169
1170 005626 117737 173356 006436          CKBYTE:          MOV B DRXDB,BADBYTE          ;SAVE BYTE FROM DISKETTE
1171 005634 063737 006436 005746          ADD BADBYTE,CKSUM          ;ACCUMULATE CHECK SUM
1172 005642 123720 006436          CMPB BADBYTE,(R0)+          ;COMPARE AGAINST GOOD BYTE
1173 005646 001054          BNE DATAER          ;IF NOT EQUAL GO TO DATAER
1174 005650 005237 006434          INC BYTECNTR
1175 005654 000762          BR EMPFLAG          ;GET NEXT BYTE
1176

```



1177	005656	113737	011360	016604	ADJSUM:	MOVB TARGET, BUFADR	;SET FIRST AND SECOND BYTES WITH ADDRESS'S
1178	005664	113737	012310	016605		MOVB TSECTOR, BUFADR+1	
1179	005672	013737	010612	005746		MOV SUM, CKSUM	;GET THE PATTERN SUM
1180	005700	063737	011360	005746		ADD TARGET, CKSUM	;ADD TRACK ADDRESS TO CHECK SUM
1181	005706	063737	012310	005746		ADD TSECTOR, CKSUM	;ADD SECTOR ADDRESS TO CHECK SUM
1182	005714	113737	005746	017002		MOVB CKSUM, BUFADR+176	;INSERT CHECK SUM TO DATA BUFFER
1183	005722	106337	005746			ASLB CKSUM	;GENERATE NEGITIVE CHECK SUM
1184	005726	105437	005746			NEGB CKSUM	
1185	005732	113737	005746	017003		MOVB CKSUM, BUFADR+177	;INSERT NEG. SUM INTO DATA BUFFER
1186	005740	012700	016604			MOV #BUFADR, RO	;SET ADDRESS OF BYTE IN RO
1187	005744	000207				RTS PC	;RETURN
1188							
1189	005746	000000			CKSUM:	0	
1190							
1191	005750	005726			EMPER:	TST (SP)+	;REMOVE THE DONE RETURN FROM THE STACK
1192	005752	012737	015402	003264		MOV #MEMPTY, PTYP1+2	;PUT ADDR OF EMPTYBUF MESSAGE IN PAR ER TYPOUT 1
1193	005760	012737	015402	003354		MOV #MEMPTY, PTYP2+2	;PUT ADDR OF EMPTYBUF MESSAGE IN PAR ER TYPOUT 2
1194	005766	012737	005536	003334		MOV #EMPBUFF, PCONT+2	;NOT HARD ERR RETURN THROUGH PCONT TO EMPTYBUF
1195	005774	000137	003244			JMP PARTEST	;REPORT PARITY ERROR
1196							
1197	006000	052737	000400	010704	DATAER:	BIS #BIT8, UNITSEL	;SET THE HAD ERROR FLAG
1198	006006	005237	006442			INC ERCNTR	;INC THE BYTE ERROR COUNTER
1199	006012	104405				CKSWR	
1200	006014	032777	020000	173172		BIT #SW13, @SWR	;TEST PRINT ERROR SW IN SWR
1201	006022	001052				BNE NOERTYP	;DON'T PRINT THE ERROR
1202	006024	032777	010000	173162		BIT #SW12, @SWR	;TEST PRINT 10 ERRORS SWITCH
1203	006032	001404				BEG 1\$	;PRINT ALL ERRORS
1204	006034	023727	006442	000012		CMP ERCNTR, #10.	;HAVE 10 ERRORS BEEN TYPED
1205	006042	003052				BGT NOERTYP	;YES, DON'T PRINT ANY MORE
1206	006044	005737	004126		1\$:	TST DATAK	;WAS THIS A READ CHECK DUE TO CRC ERROR
1207	006050	001403				BEG 2\$	;NO, REPORT DATA ERRORS
1208	006052	105701				TSTB R1	;YES, TEST FIRST ERROR FLAG
1209	006054	106717				BPL TYPERR	;TYPE THE DATA CRC ERROR
1210	006056	000412				BR 3\$	
1211	006060	105701			2\$:	TSTB R1	;TEST FIRST ERROR FLAG
1212	006062	100014				BPL TYPERR	
1213	006064	104400	015052			TYPE ,MCRLF	
1214	006070	104400	014527			TYPE ,MDERHDR	;PRINT ERROR HEADER
1215	006074	104400	015052			TYPE ,MCRLF	
1216	006100	004737	007716			JSR PC, TYPADR	;PRINT TRACK AND SECTOR LOCATIONS
1217	006104	104400	014616		3\$:	TYPE ,MCOLMUN	;SET UP COLMUN HEADINGS
1218	006110	042701	000200			BIC #BIT7, R1	;CLEAR FIRST ERROR FLAG
1219	006114	013737	006434	006126	TYPERR:	MOV BYTECNTR, 1\$	;PRINT BYTE NUMBER
1220	006122	004537	014446			JSR R5, SGLDEC	
1221	006126	000000			1\$:	OPEN	
1222	006130	104400	015047			TYPE ,DBLSP	
1223	006134	013746	006436			MOV BADBYTE, -(SP)	;PRINT BYTE READ FROM DISKETTE
1224	006140	104402				TYPOS	
1225	006142	000003				.WORD 3	
1226	006144	104400	015047			TYPE ,DBLSP	
1227	006150	114037	006440			MOVB -(RO), GOODBYTE	;GET GOOD BYTE
1228	006154	005200				INC RO	;RETURN RO TO NEXT BYTE IN BUFFER
1229	006156	013746	006440			MOV GOODBYTE, -(SP)	
1230	006162	104403				TYPON	;PRINT GOOD DATA
1231	006164	104400	015052			TYPE ,MCRLF	
1232	006170	104405			NOERTYP:	CKSWR	



```

1233 006172 005777 173016          TST 2SWR          ;TEST HALT ON ERROR SWITCH
1234 006176 100001          BPL CONT20
1235 006200 000000          HLT14:          HALT
1236 006202 005237 006434          CONT20:        INC BYTECNTR
1237 006206 000137 005622          JMP EMPFLAG
1238
1239 006212 005737 004126          EMPDONE:       TST DATAK      ;WAS THIS READCHECK CAUSED BY CRC ERROR
1240 006216 001401          BEQ 1$        ;NO CONTINUE
1241 006220 000207          RTS PC        ;YES, RETURN TO CRC ERROR HANDLER
1242 006222 005737 006442          1$:           TST ERCNTR     ;WAS THERE ERRORS
1243 006226 001471          BEQ CONT3    ;NO ERRORS
1244 006230 104405          CKSWR
1245 006232 032777 020000 172754        BIT 2SW13,2SWR ;YES, TEST DON'T PRINT SWITCH
1246 006240 001024          BNE 2$        ;DON'T PRINT THE ERROR
1247 006242 104400 015333          TYPE ,MERC    ;PRINT THE TOTAL DATA ERROR COUNT
1248 006246 013737 006442 006260        MOV ERCNTR,3$
1249 006254 004537 014446          JSR R5,SGLDEC
1250 006260 000000          3$:          OPEN
1251 006262 104400 016327          TYPE ,MSUM    ;INDICATE IF CHECK SUM WAS GOOD OR HAD ERRORS
1252 006266 105737 005746          TSTB CKSUM
1253 006272 001403          BEQ 4$
1254 006274 104400 016067          TYPE ,MERS
1255 006300 000402          BR 5$
1256 006302 104400 016322          4$:          TYPE ,MGOOD
1257 006306 104400 015052          5$:          TYPE ,MCRLF
1258 006312 012703 017062          2$:          MOV 2$DATALOG,R3 ;SET INC.INST.FOR UNIT 0 ERROR LOG
1259 006316 004737 021044          JSR PC,U1LOG  ;TEST FOR UNIT 1 ERROR LOGS
1260 006322 005213          INC (R3)      ;INC DATA ERROR LOG
1261 006324 004737 003770          JSR PC,TRKERR ;INC ERROR PER TRACK COUNTER
1262 006330 104405          CKSWR
1263 006332 032777 004000 172654        BIT 2SW11,2SWR ;TEST NO RETRY SWITCH
1264 006340 001007          BNE 6$        ;IF SET LOG HARD ERROR
1265 006342 005337 017012          DEC DATARETRY ;HAVE 10 ERRORS BEEN LOGED
1266 006346 001404          BEQ 6$
1267 006350 004737 004246          JSR PC,REREAD ;NO, GO REREAD THE DATA
1268 006354 000137 005536          JMP EMPBUFF   ;GO RECHECK THE DATA
1269 006360 012703 017122          6$:          MOV 6$DATALOG,R3 ;YES, SET INC.INST.FOR UNIT 0 ERROR LOG
1270 006364 004737 021044          JSR PC,U1LOG  ;TEST FOR UNIT 1 ERROR LOGS
1271 006370 005213          INC (R3)      ;INC HARD DATA NO STATUS ERROR
1272 006372 104400 015240          TYPE ,DBLLF
1273 006376 104400 015755          TYPE ,MUNREC ;TYPE UNRECOVERABLE DATA NO
1274 006402 104400 014527          TYPE ,MDERHOR ;STATUS ERROR
1275 006406 104400 015052          TYPE ,MCRLF
1276 006412 005337 012306          CONT3:       DEC SECCNTR
1277 006416 001405          BEQ EXIT
1278 006420 004737 004204          JSR PC,XREAD  ;READ THE NEXT SECTOR
1279 006424 000137 005536          JMP EMPBUFF
1280 006430 005001          3$:          CLR R1        ;CLEAR THE ONE READ FLAG
1281 006432 000207          EXIT:        RTS PC
1282
1283 006434 000000          BYTECNTR:    0
1284 006436 000000          BADBYTE:     0
1285 006440 000000          GOODBYTE:   0
1286 006442 000000          ERCNTR:      0
1287
1288 ;*****

```







```

1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358 006630 004737 010312 TEST7: JSR PC,GETPATTERN
1359 006634 013737 011370 007012 MOV SEQUEN,SEQSAV ;SAVE THE SELECTED SEQUENCE
1360 006642 012737 000001 011370 TEST7X: MOV #1,SEQUEN ;FORCE INCREMENT SEQUENCE
1361 006650 005137 004122 COM RDAFTWT ;SET READ AFTER WRITE FLAG FOR ERROR TESTING
1362 006654 004737 011144 JSR PC,INITTRACKS
1363 006660 004737 010614 JSR PC,GETUNIT
1364 006664 004737 011254 1$: JSR PC,GETTRACK
1365 006670 004737 003112 JSR PC,WRITE
1366 006674 004737 005526 JSR PC,READCHK
1367 006700 005337 011356 DEC TRKCNTR ;IS THE FIRST HALF OF THE PASS DONE
1368 006704 001367 BNE 1$ ;NO,CONTINUE WRITING AND READING
1369 006706 005037 004122 CLR RDAFTWT ;YES,NOW DO A READ ONLY
1370 006712 013737 007012 011370 MOV SEQSAV,SEQUEN ;USING SELECTED SEQUENCE
1371 006720 004737 011144 JSR PC,INITTRACKS
1372 006724 004737 011254 2$: JSR PC,GETTRACK
1373 006730 004737 005526 JSR PC,READCHK
1374 006734 005337 011356 DEC TRKCNTR ;IS THIS HALF OF THE PASS DONE
1375 006740 001371 BNE 2$ ;NO,CONTINUE READ CHECKING
1376 006742 004737 010732 JSR PC,STOP ;YES,TEST FOR END OF PASS CONDITIONS
1377 006746 032737 040100 010704 BIT #40100,UNITSEL ;HAVE ALL SELECTED UNITS BEEN USED
1378 006754 001332 BNE TEST7X ;IF A USED BIT IS STILL SET GO TO NEXT DRIVE
1379 006756 012706 001200 MOV #1200,SP ;RESET STACK ADDRESS FOR ACT11 EOP
1380 006762 032737 001000 001212 BIT #BIT9,DTESTP ;NEXT PASS,COMPLEMENT D D BIT
1381 006770 001404 BEQ 3$
1382 006772 042737 001000 001212 BIC #BIT9,DTESTP ;IT WAS ON CLEAR IT
1383 007000 000720 BR TEST7X
1384 007002 052737 001000 001212 3$: BIS #BIT9,DTESTP ;IT WAS OFF SET IT
1385 007010 000714 BR TEST7X
1386
1387 007012 000000 SEQSAV: 0
1388 ;;*****
1389
1390 ;TEST FOR TRANSFER FLAG AND PROGRAM NOT HUNG
1391
1392 007014 005037 007200 TRCK: CLR HCNTR1 ;CLEAR HUNG COUNTER
1393 007020 012746 000340 MOV #PR7,-(SP) ;RAISE INTERRUPT LEVEL TO INHIBIT INTERUPTS
1394 007024 012746 007032 MOV #3$,-(SP)
1395 007030 000002 RTI
1396 007032 105777 172150 3$: TSTB DRXCS ;TEST FOR TR FLAG
1397 007036 100001 BPL 1$ ;SKIP NEXT INST. IF NOT SET
1398 007040 000207 RTS PC ;RETURN TO TRANSFER DATA
1399 007042 023727 006434 000200 1$: CMP BYTECNTR,#128. ;HAVE 128 BYTES BEEN TRANSFERED
1400 007050 001403 BEQ 2$ ;ALLOW AN INTERUPR IF EQUAL

```



```

1401 007052 005237 007200          INC HCNTR1          ; IF THE HUNG COUNTER OVERFLOWES EITHER
1402                                     ; THERE WAS NO TR FLAG OR NO INTERUPT OCCURED
1403                                     ; AFTER 128 BYTES TRANSFERED
1404 007056 001365          BNE 3$             ; NO OVERFLOW RETEST FOR TR FLAG
1405 007060 132777 000040 172120 2$: BITB #DONEBIT,ARXCS ; TEST FOR A DONE FLAG WHICH WOULD INDICATE A
1406                                     ; POSSABLE ERROR CONDITION EXISTS WHILE
1407                                     ; TRANSFERING COMMANDS OR DATA
1408 007066 001407          BEQ 4$             ; IF DONE NOT SET,TYPE HUNG
1409 007070 005037 006434          CLR BYTECNTR       ; CLEAR BYTE COUNTER SO IT WON'T INTERFERE
1410                                     ; WITH TESTING FOR DONE.
1411 007074 005726          TST (SP)+          ; REMOVE THIS RETURN ADDRESS FROM STACK
1412 007076 005046          CLR -(SP)          ; LOWER INTERUPT LEVEL TO ACCEPT AN INTERUPT
1413 007100 012746 007106          MOV #4$,-(SP)     ; THE INTERUPT WILL RETURN THROUGH PREVIOUSLY
1414 007104 000002          RTI              ; SET ADDRESSES
1415 007106 005237 007200          4$: INC HCNTR1       ; ALLOW SOME TIME FOR FLAG TO SET
1416 007112 001362          BNE 2$
1417 007114 104400 015432          TYPE ,MHUNG
1418 007120 000000          HLT21:           ; PROGRAM HUNG, SP (177706) CONTAINS RETURN
1419                                     ; ADDRESS OF WAITING LOOP
1420
1421                                     ; TEST FOR DONE FLAG AND PROGRAM HUNG
1422 007122 012603          DONECK: MOV (SP)+,R3 ; TAKE THE RETURN ADDRESS OF THE STACK INCASE
1423                                     ; THE PROGRAM IS EXPECTING AN INTERUPT AT DONE
1424 007124 005037 007200          CLR HCNTR1        ; CLEAR THE LOWER HUNG COUNTER
1425 007130 012737 177740 007202          MOV #177740,HCNTR2 ; SET UPER COUNTER FOR 31 FULL LOOPS
1426 007136 032777 000040 172042 2$: BIT #DONEBIT,ARXCS ; TEST FOR DONE FLAG
1427 007144 001402          BEQ 1$            ; IF NOT SET SKIP RETURN
1428 007146 010346          MOV R3,-(SP)     ; PUT RETURN ADDRESS BACK ON THE STACK
1429 007150 000207          RTS PC           ; RETURN
1430 007152 062737 000001 007200 1$: ADD #1,HCNTR1     ; WHEN THE DOUBLE WORD COUNTER OVERFLOWES
1431 007160 005537 007202          ADC HCNTR2       ; THE PROGRAM MUST BE HUNG
1432 007164 005737 007202          TST HCNTR2
1433 007170 001362          BNE 2$
1434 007172 104400 015432          TYPE ,MHUNG
1435 007176 000000          HLT20:           ; PROGRAM HUNG, CONTENTS OF R3 (177703) HOLDES
1436                                     ; ADDRESS OF WAITING TEST.
1437
1438 007200 000000          HCNTR1:          0
1439 007202 177740          HCNTR2:          177740
1440
1441                                     ; AN INTERUPT DID NOT OCCURE AT A FUNCTION DONE FLAG.
1442
1443 007204 005237 017034          NOINTER: INC INTER ; INC INTERUPT ERROR COUNTER
1444 007210 104405          CKSWR
1445 007212 032777 020000 171774          BIT #SW13,ASWR ; TEST DON'T PRINT ERROR SWITCH
1446 007220 001004          BNE 1$
1447 007222 104400 015260          TYPE ,MINTER ; TYPE NO INTERUPT ON DONE ERROR
1448 007226 104400 015052          TYPE ,MCRLF
1449 007232 104405          1$: CKSWR
1450 007234 005777 171754          TST ASWR ; TEST HALT ON ERROR SWITCH
1451 007240 100001          BPL CONT21
1452 007242 000000          HLT15:          HALT ; HALT ON ERROR
1453 007244 004737 007250          CONT21: JSR PC,INTSERV ; JSR TO INTSERV AS IF IT WAS AN INTERUPT
1454
1455 ; ; *****
1456

```



```

; INTERRUPT SERVICE
1457
1458
1459 007250 117737 171734 010070 INTSERV:      MOVB 2RXDB,ASTAT      ;SAVE THE ERROR AND STATUS WORD
1460 007256 005777 171724                    TST 2RXCS             ;TEST THE ERROR FLAG
1461 007262 100450                    BMI  RXERROR         ;THERE WAS AN ERROR GO REPORT IT
1462 007264 032737 000004 010070            BIT  #BIT2,ASTAT     ;IS INIT DONE SET
1463 007272 001402                    BEQ  2$              ;NO CONTINUE
1464 007274 000137 007702                    JMP  RXPWR           ;YES, REPORT POWER FAILED AND RESTART
1465 007300 032737 000003 010070 2$:      BIT  #3,ASTAT        ;ARE PAR OR CRC BITS SET
1466 007306 001023                    BNE  1$              ;YES GO LOG ERROR
1467 007310 132777 000040 171670            BITB #DONEBIT,2RXCS ;IS DONE SET
1468 007316 001014                    BNE  3$              ;IF SET RETURN TO TEST
1469 007320 005237 017032                    INC  UKNINT         ;INC UNKNOWN INTERRUPT ERROR LOG
1470 007324 104405                    CKSWR
1471 007326 032777 020000 171660            BIT  #SW13,2SWR     ;TEST DON'T PRINT ERROR SWITCH
1472 007334 001004                    BNE  4$              ;DON'T PRINT
1473 007336 104400 015312                    TYPE ,MUKNINT       ;TYPE UNKNOWN INTERRUPT
1474 007342 104400 015052                    TYPE ,MCRLF
1475 007346 000002                    RTI                  ;RETURN FROM THE INTERRUPT
1476 007350 062706 000006                    4$: ADD  #6,SP         ;BYPASS INTERRUPT POINTERS ON STACK
1477 007354 000207                    3$: RTS PC           ;RETURN TO PROGRAM
1478 007356 005237 017030                    1$: INC  NOERLOG     ;NO STATUS ERROR FLAG ERROR
1479 007362 104405                    CKSWR
1480 007364 032777 020000 171622            BIT  #SW13,2SWR     ;TEST DON'T PRINT ERROR SWITCH
1481 007372 001004                    BNE  RXERROR        ;TYPE NO STATUS ERROR ERROR
1482 007374 104400 016034                    TYPE ,MNOFLAG
1483 007400 104400 015052                    TYPE ,MCRLF
1484
1485 007404 052737 000400 010704 RXERROR:    BIS  #BIT8,UNITSEL  ;SET HAD ERROR FLAG
1486 007412 012737 000012 017014            MOV  #10,P2RETRY    ;SET RETRY COUNTER
1487 007420 012777 000017 171560 2$:      MOV  #RDER,2RXCS   ;GET THE ERROR CODE
1488 007426 004737 007122                    JSR  PC,DONECK      ;TEST FOR DONE FLAG
1489 007432 032777 000002 171550            BIT  #2,2RXDB       ;WAS THERE A PARITY ERROR
1490 007440 001403                    BEQ  1$              ;NO CONTINUE
1491 007442 004737 002046                    JSR  PC,STATERR     ;YES, GO REPORT THE PARITY ERROR
1492 007446 000764                    BR   2$              ;REISSUE THE FUNCTION
1493 007450 117737 171534 010072 1$:      MOVB 2RXDB,BSTAT   ;SAVE THE ERROR CODE IN B STATUS
1494 007456 005737 010072                    TST  BSTAT          ;IS THERE A DEFINITE CODE
1495 007462 001407                    BEQ  NOPRNT         ;NO CONTINUE
1496 007464 013705 010072                    MOV  BSTAT,R5       ;ADJUST ERROR CODE TO PRODUCE AN EVEN ADDR
1497 007470 006005                    ROR  R5              ;OF THE CORRESPONDING COUNTER
1498 007472 006005                    ROR  R5
1499 007474 062705 017130                    ADD  #ERCODE-2,R5  ;ADD ON ADDR OF ERROR LOG -2, AS THERE IS NO 0
1500                                ;ERROR CODE. THE CONTENTS OF R5 WILL READJUST
1501                                ;ADDRESS TO CORRECT LOG.
1502 007500 005215                    INC  (R5)           ;INC THE CORRECT ERROR CODE COUNTER
1503 007502 104405                    CKSWR
1504 007504 032777 020000 171502 NOPRNT:    BIT  #SW13,2SWR     ;TEST PRINT ERROR SWITCH IN SWR
1505 007512 001032                    BNE  2$              ;
1506 007514 104400 015052                    TYPE ,MCRLF
1507 007520 104400 015063                    TYPE ,MERHEADER    ;TYPE ERROR HEADER
1508 007524 104400 016142                    TYPE ,MPASS         ;TYPE PASSES COMPLETED AT ERROR
1509 007530 013737 021042 007542            MOV  PASCNTR,1$
1510 007536 004537 014446                    JSR  R5,SGLDEC
1511 007542 000000                    OPEN
1512 007544 104400 015052                    TYPE ,MCRLF
    
```



1513	007550	104400	015167			MOV	TYPE ,MRXCS		;TYPE COMMAND STATUS REGISTER
1514	007554	013746	004120			TYPOS	FUNCTION,-(SP)	:::SAVE FUNCTION FOR TYPEOUT	
1515	007560	104402				.BYTE	3	:::GO TYPE--OCTAL ASCII	
1516	007562	003				.BYTE	0	:::TYPE 3 DIGIT(S)	
1517	007563	000						:::SUPPRESS LEADING ZEROS	
1518	007564	004737	007716				JSR PC,TYPADR		;TYPE ADDRESSES AND RUN CONDITIONS
1519	007570	104400	015052				TYPE ,MCRLF		
1520	007574	004737	010250				JSR PC,TYPDEC		;PRINT THE STATUS REGISTERS
1521	007600	032737	000020	004120	2\$:		BIT #BIT4,FUNCTION		;WHAT DRIVE IS BEING USED
1522	007606	001006					BNE 6\$		
1523	007610	012777	000013	171370			MOV #RDOSTAT,ARXCS		;DRIVE 0 BEING USED
1524	007616	004737	007122		5\$:		JSR PC,DONECK		;TEST FOR DONE FLAG
1525	007622	000404					BR 7\$		
1526	007624	012777	000033	171354	6\$:		MOV #RD1STAT,ARXCS		;DRIVE 1 BEING USED
1527	007632	000771					BR 5\$		
1528	007634	032777	000002	171346	7\$:		BIT #2,ARXDB		;WAS THERE A PARITY ERROR
1529	007642	001403					BEQ 3\$		;NO CONTINUE
1530	007644	004737	002046				JSR PC,STATER		;YES,REPORT THE ERROR
1531	007650	000753					BR 2\$		;REISSUE THE COMMAND
1532	007652	105777	171332		3\$:		TSTB ARXDB		;WAS DRIVE READY SET
1533	007656	100406					BMI 4\$		;YES RETURN
1534	007660	104400	015016				TYPE ,MNODRY		
1535	007664	104400	015052				TYPE ,MCRLF		
1536	007670	004737	010074				JSR PC,DELUNIT		;NO GO DELETE THAT UNIT
1537	007674	062706	000004		4\$:		ADD #4,SP		;MOVE ERROR RETURN TO TOP OF STACK
1538	007700	000207					RTS PC		
1539									
1540	007702	104400	016344		RXPWR:		TYPE ,MRX11		;ONLY THE RX11 POWER HAS FAILED
1541	007706	104400	014136				TYPE ,SPOWER		;PRINT POWER FAILED
1542	007712	000137	002700				JMP RESTART		;GO TO RESTART
1543									
1544									
1545	007716	104400	014555		TYPADR:		TYPE ,MTRK		;TYPE TRACK ADDRESS
1546	007722	013737	011360	007734			MOV TARGET,3\$		
1547	007730	004537	014446				JSR R5,SGLDEC		
1548	007734	000000			3\$:		OPEN		
1549	007736	104400	014604				TYPE ,MSECT		;TYPE SECTOR ADDRESS
1550	007742	013737	012310	007762			MOV TSECTOR,2\$		
1551	007750	042737	177740	007762			BIC #177740,2\$		;CLEAR ALL BUT SECTOR ADDRESS
1552	007756	004537	014446				JSR R5,SGLDEC		
1553	007762	000000			2\$:		OPEN		
1554	007764	104400	015047				TYPE ,DBLSP		
1555	007770	032737	000020	010704			BIT #BIT4,UNITSEL		;WHITCH DRIVE IS BEING USED
1556	007776	001003					BNE 1\$		
1557	010000	104400	015106				TYPE ,MUNIT0		;TYPE UNIT 0
1558	010004	000402					BR TYPSEL		
1559	010006	104400	015116		1\$:		TYPE ,MUNIT1		;TYPE UNIT 1
1560	010012	104400	015244		TYPSEL:		TYPE ,MPAT		;TYPE PATTERN CODE
1561	010016	013746	010420			MOV	PAT,-(SP)	:::SAVE PAT FOR TYPEOUT	
1562	010022	104402				TYPOS		:::GO TYPE--OCTAL ASCII	
1563	010024	001				.BYTE	1	:::TYPE 1 DIGIT(S)	
1564	010025	000				.BYTE	0	:::SUPPRESS LEADING ZEROS	
1565	010026	104400	015047				TYPE ,DBLSP		
1566	010032	104400	015250				TYPE ,MTEST		;TYPE TEST CODE
1567	010036	013746	002140			MOV	TEST,-(SP)	:::SAVE TEST FOR TYPEOUT	
1568	010042	104402				TYPOS		:::GO TYPE--OCTAL ASCII	



```

1569 010044 001 .BYTE 1 ;;TYPE 1 DIGIT(S)
1570 010045 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
1571 010046 104400 015047 TYPE ,DBLSP
1572 010052 104400 015254 TYPE ,MSEQ ;TYPE SEQUENCE CODE
1573 010056 013746 011370 MOV SEQUEN,-(SP) ;;SAVE SEQUEN FOR TYPEOUT
1574 010062 104402 TYPOS ;;GO TYPE--OCTAL ASCII
1575 010064 001 .BYTE 1 ;;TYPE 1 DIGIT(S)
1576 010065 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
1577 010066 000207 RTS PC
1578
1579 010070 000000 ASTAT: 0
1580 010072 000000 BSTAT: 0
1581
1582 010074 132737 000020 004120 DELUNIT: BITB #BIT4,FUNCTION ;TEST FOR PRESENTLY USED UNIT
1583 010102 001016 BNE 3$
1584 010104 104400 015106 TYPE ,MUNITO ;UNIT 0 HAS BEEN DELETED FROM USE
1585 010110 104400 015126 TYPE ,MDELET
1586 010114 104400 015240 TYPE ,DBLLF
1587 010120 042737 000200 010704 BIC #BIT7,UNITSEL ;CLEAR UNIT 0 SELECTION BIT
1588 010126 005737 010704 TST UNITSEL ;WAS UNIT 1 SELECTED FOR USE
1589 010132 100020 BPL 1$ ;UNIT 1 NOT SELECTED GO DUMP ERROR REPORT
1590 010134 000137 002726 2$: JMP TESTSEL ;CONTINUE ON OTHER UNIT AT BEGINING OF TEST
1591 010140 104400 015116 3$: TYPE ,MUNIT1 ;UNIT 1 HAS BEEN DELETED FROM USE
1592 010144 104400 015126 TYPE ,MDELET
1593 010150 104400 015240 TYPE ,DBLLF
1594 010154 042737 100000 010704 BIC #BIT15,UNITSEL ;CLEAR UNIT 1 SELECTION BIT
1595 010162 105737 010704 TSTB UNITSEL ;WAS UNIT 0 SELECTED FOR USE
1596 010166 100762 BMI 2$ ;YES CONTINUE ON UNIT 0
1597 010170 005137 022514 COM SHORTRPT ;SET SHORT REPORT FLAG
1598 010174 000137 021062 1$: JMP ERDUMP ;CAN'T CONTINUE TYPE OUT STATISTICAL REPORT
1599
1600
1601 010200 117737 171004 010070 RDCODE: MOVB #RDXB,ASTAT ;SAVE THE A STATUS
1602 010206 012777 000017 170772 2$: MOV #RDER,#RXC ;READ THE B STATUS REGISTER
1603 010214 004737 007122 JSR PC,DONECK ;WAIT FOR DONE FLAG
1604 010220 032777 000002 170762 BIT #2,#RDXB ;WAS THERE A PARITY ERROR
1605 010226 001403 BEQ 1$ ;NO,CONTINUE
1606 010230 004737 002046 JSR PC,STATER ;YES,REPORT THE PARITY ERROR
1607 010234 000764 BR 2$ ;RETRY READING STATUS B
1608 010236 117737 170746 010072 1$: MOVB #RDXB,BSTAT ;SAVE THE B STATUS CODES
1609 010244 104400 015052 TYPE ,MCRLF
1610 010250 104400 015210 TYPCODE: TYPE ,MASTAT ;TYPE THE CONTENTS OF THE TWO STATUS REGISTERS
1611 010254 013746 010070 MOV ASTAT,-(SP) ;;SAVE ASTAT FOR TYPEOUT
1612 010260 104402 TYPOS ;;GO TYPE--OCTAL ASCII
1613 010262 003 .BYTE 3 ;;TYPE 3 DIGIT(S)
1614 010263 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
1615 010264 104400 015040 TYPE ,TAB
1616 010270 104400 015224 TYPE ,MBSTAT
1617 010274 013746 010072 MOV BSTAT,-(SP) ;;SAVE BSTAT FOR TYPEOUT
1618 010300 104402 TYPOS ;;GO TYPE--OCTAL ASCII
1619 010302 003 .BYTE 3 ;;TYPE 3 DIGIT(S)
1620 010303 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
1621 010304 104400 015052 TYPE ,MCRLF
1622 010310 000207 RTS PC
1623
1624

```



1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680

;;\*\*\*\*\*

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

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

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

;;\*\*\*\*\*

```

1651 010312 142737 000377 010360 GETPATTERN: BICB #377,2#BRONPAT ;CLEAR BRANCH OFFSET
1652 010320 005037 010612 CLR SUM ;SET UP FOR ACCUMULATION OF CHECK SUM
1653 010324 005737 010420 TST PAT ;IF NO PATTERN SPECIFIED FORCE PATTERN 7
1654 010330 001003 BNE 1$
1655 010332 012737 000007 010420 MOV #7,PAT
1656 010340 013704 010420 1$: MOV PAT,R4 ;GET PATTERN BITS
1657 010344 005304 DEC R4 ;ADJUST FOR CORRECT OFFSET
1658 010346 006304 ASL R4
1659 010350 150437 010360 BISB R4,2#BRONPAT ;INSERT OFFSET
1660 010354 012704 016606 MOV #BUFADR+2,R4 ;SET UP ADDRESS OF FIRST BYTE
1661 010360 000777 BRONPAT: BR ;BRANCH BY OFFSET SELECTED
1662 010362 000137 010422 JMP DATA0 ;000 DATA BYTE
1663 010366 000137 010434 JMP DATA1 ;377 DATA BYTE
1664 010372 000137 010444 JMP FLOAT0 ;FLOAT A 0 THROUGH ALL 1'S
1665 010376 000137 010506 JMP FLOAT1 ;FLOAT A 1 THROUGH ALL 0'S
1666 010402 000137 010514 JMP PAT125 ;125/052 DATA WORD
1667 010406 000137 010534 JMP PAT314 ;314/063 DATA WORD
1668 010412 000137 010544 JMP RANDATA ;RANDOM DATA BYTE

```

DATABYTE: 0  
PAT: 0

;;\*\*\*\*\*

;LOAD SOFTWARE BUFFER WITH ALL ZEROS  
; P = 1

```
1680 010422 005037 010416 DATA0: CLR DATABYTE
```



```

1681 010426 004737 010564 PATGEN: JSR PC,LOAD ;GO LOAD THE DATA BUFFER
1682 010432 000775 BR PATGEN
1683
1684 ;;*****
1685
1686 ;LOAD SOFTWARE BUFFER WITH ALL ONES
1687 ; P = 2
1688
1689
1690 010434 112737 000377 010416 DATA1: MOVB #377,DATABYTE
1691 010442 000771 BR PATGEN
1692
1693 ;;*****
1694
1695 ;FLOAT A 0 THROUGH ONES IN SOFTWARE BUFFER
1696 ; P = 3
1697
1698
1699 010444 112737 000376 010416 FLOAT0: MOVB #376,DATABYTE ;SET UP A ONES FIELD
1700 010452 000261 XPATGEN: SEC ;SET THE C BIT TO ROTATE THROUGH THE DATA
1701 010454 012702 000000 1$: MOV #0,R2 ;CLR R2 (CAN'T USE "CLR" AS IT CLEARS "C" BIT)
1702 010460 103001 BCC 2$ ;BR IF THE "C" BIT IS CLEARED
1703 010462 005202 INC R2 ;SET R2 IF NOT
1704 010464 004737 010564 2$: JSR PC,LOAD ;GO LOAD THE DATA BUFFER
1705 010470 000241 CLC
1706 010472 005702 TST R2 ;IS R2 NONZERO
1707 010474 001401 BEQ 3$
1708 010476 000261 SEC ;YES, SET THE "C" BIT
1709 010500 106137 010416 3$: ROLB DATABYTE
1710 010504 000763 BR 1$
1711
1712 ;;*****
1713
1714 ;FLOAT A 1 THROUGH ALL ZEROS IN SOFTWARE BUFFER
1715 ; P = 4
1716
1717 010506 005037 010416 FLOAT1: CLR DATABYTE
1718 010512 000757 BR XPATGEN
1719
1720 ;;*****
1721
1722 ;LOAD SOFTWARE BUFFER WITH ALTERNATING 1 AND 0 FOR
1723 ;ONE BYTE AND THE COMPLIMENT INTO THE NEXT
1724 ; P = 5
1725
1726 010514 112737 000125 010416 PAT125: MOVB #125,DATABYTE
1727 010522 004737 010564 XXPATGEN: JSR PC,LOAD
1728 010526 105137 010416 COMB DATABYTE ; * (SEE NOTE BELOW )
1729 010532 000773 BR XXPATGEN
1730
1731 ; * NOTE: TO MAKE PATTERN 5 AND 6 COMPATABLE WITH THE RX8 PATTERNS
1732 ;NOP THIS INSTRUCTION. (CHANGE THESE 2 LOCATIONS TO 000240)
1733 ;THIS CHANGE IS FOR INTERPROCESSOR RX** COMPATABILITY TESTING ONLY.
1734
1735 ;;*****
1736

```



# K04

```

1737                                     ;LOAD SOFTWARE BUFFER WITH ALTERNATING PAIRS OF 1 AND 0 AND
1738                                     ;COMPLIMENT INTO THE NEXT
1739                                     ; P = 6
1740
1741 010534 112737 000314 010416 PAT314:      MOVB #314,DATABYTE
1742 010542 000767                                     BR XXPATGEN
1743
1744                                     ;*****
1745
1746                                     ;LOAD SOFTWARE BUFFER WITH RANDOM DATA PATTERN
1747                                     ; P = 0 OR 7
1748
1749 010544 004737 012122 RANDATA:          JSR PC,RANGEN          ;GET RANDOM NUMBER
1750 010550 113737 012214 010416          MOVB RANUM,DATABYTE
1751 010556 004737 010564          JSR PC,LOAD
1752 010562 000770          BR RANDATA
1753
1754 010564 063737 010416 010612 LOAD:      ADD DATABYTE,SUM      ;ACCUMULATE THE PATTERN CHECK SUM
1755 010572 113724 010416          MOVB DATABYTE,(R4)+   ;LOAD THE DATA BUFFER
1756 010576 022704 017002          CMP #BUFADR+176,R4   ;HAVE 124 BYTES BEEN GENERATED
1757 010602 001401          BEQ 1$              ;IF YES, RETURN TO TEST
1758 010604 000207          RTS PC              ;IF NO, RETURN TO PATTERN GENERATOR
1759 010606 005726          1$: TST (SP)+          ;TAKE PATTERN RETURN ADDRESS OF STACK
1760 010610 000207          RTS PC              ;RETURN TO TEST
1761
1762 010612 000000          SUM:          0
1763
1764                                     ;*****
1765
1766                                     ;TEST FOR SELECTED UNITS,DRIVE READY,AND USED CONDITIONS
1767                                     ;ALSO CONTAINS A "HAD ERROR" FLAG TO BE TESTED AT EOP.
1768                                     ;THE BITS IN UNITSEL ARE USED AS FOLLOWS
1769
1770                                     ;BIT15 =UNIT 1 SELECTED VIA SWR
1771                                     ;BIT14 =UNIT 1 USED BIT
1772                                     ;BIT8  =THIS PASS HAD AN ERROR
1773                                     ;BIT7  =UNIT 0 SELECTED VIA SWR
1774                                     ;BIT6  =UNIT 0 USED BIT
1775                                     ;BIT4  =UNIT SELECTION FOR FUNCTION WORD
1776
1777                                     ;*****
1778
1779 010614 032737 000100 010704 GETUNIT:   BIT #BIT6,UNITSEL   ;WAS UNIT 0 JUST USED
1780 010622 001012          BNE 1$              ;UNIT 0 USED CHECK UNIT 1
1781 010624 105737 010704          TSTB UNITSEL       ;WAS UNIT 0 SELECTED
1782 010630 100007          BPL 1$              ;NO GO TO UNIT 1
1783 010632 042737 040020 010704          BIC #40020,UNITSEL ;CLEAR UNIT 1 USED BIT AND FUNCTION UNIT BIT
1784 010640 052737 000100 010704          BIS #BIT6,UNITSEL ;SET UNIT 0 USED BIT
1785 010646 000207          RTS PC
1786 010650 005737 010704          1$: TST UNITSEL      ;WAS UNIT 1 SELECTED
1787 010654 100012          BPL 2$              ;NO RETURN
1788 010656 032737 040000 010704          BIT #BIT14,UNITSEL ;HAS UNIT 1 BEEN USED
1789 010664 001006          BNE 2$              ;YES RETURN
1790 010666 042737 000100 010704          BIC #BIT6,UNITSEL ;CLEAR UNIT 0 USED BIT
1791 010674 052737 040020 010704          BIS #40020,UNITSEL ;SET UNIT 1 USED BIT AND FUNCTION UNIT BIT
1792 010702 000207          2$: RTS PC

```



1793							
1794							
1795							
1796	010704	000000		UNITSEL:	0		
1797							
1798							
1799							
1800	010706	005737	010704	DONE:	TST UNITSEL		; IS UNIT 1 SELECTED
1801	010712	100006			BPL 1\$		; NO RETURN
1802	010714	032737	040000 010704		BIT #BIT14,UNITSEL		; YES HAS IT BEEN USED
1803	010722	001002			BNE 1\$		; YES RETURN
1804	010724	062706	000002		ADD #2,SP		; BYPASS NOT DONE RETURE ON STACK
1805	010730	000207		1\$:	RTS PC		
1806							
1807							
1808	010732	004737	010706	STOP:	JSR PC,DONE		; HAVE ALL DRIVES BEEN USED
1809	010736	005037	007200		CLR HCNTR1		
1810	010742	005237	021042		INC PASCNTR		; KEEP TOTAL NUMBER OF PASSES
1811	010746	005737	011142		TST CHARCT		; HAVE 72 EOP INDICATORS BEEN TYPED
1812	010752	001005			BNE 3\$		
1813	010754	104400	015052		TYPE ,MCRLF		; YES DO A CRLF
1814	010760	012737	177670 011142		MOV #-72, CHARCT		; RESET CHARACTER COUNTER
1815	010766	005237	011142	3\$:	INC CHARCT		
1816	010772	032737	000400 010704		BIT #BIT8,UNITSEL		; TEST HAD ERROR FLAG
1817	011000	001403			BEQ 2\$		; NO ERROR PRINT 0
1818	011002	104400	015057		TYPE ,MEREOP		; HAD ERROR PRINT ?
1819	011006	000404			BR 1\$		
1820	011010	104400	015055	2\$:	TYPE ,MEOP		; PRINT EOP INDICATOR
1821	011014	104400	015061		TYPE ,MABELL		
1822	011020	104405		1\$:	CKSWR		
1823	011022	032777	040000 170164		BIT #SW14,SWR		; TEST EOP HALT SWITCH
1824	011030	001417			BEQ CONT22		
1825	011032	104400	015052		TYPE ,MCRLF		
1826	011036	104400	016142		TYPE ,MPASS		; AT HALT ON PASS PRINT NUMBER OF PASSES COMPLETE
1827	011042	013737	021042 011054		MOV PASCNTR,6\$		
1828	011050	004537	014446		JSR RS,SGLDEC		
1829	011054	000000		6\$:	OPEN		
1830	011056	104400	015052		TYPE ,MCRLF		
1831	011062	000000		HLT16:	HALT		
1832	011064	005037	011142		CLR CHARCT		; THIS CAUSES A CRLF AFTER HALT AT E.O.P.
1833	011070	042737	040500 010704	CONT22:	BIC #40500,UNITSEL		; CLEAR USED BITS, AND "HAD ERROR" BIT
1834	011076	022737	000007 010420		CMP #7,PAT		; IF RANDOM DATA GET NEW DATA FOR BUFFER
1835	011104	001002			BNE 2\$		
1836	011106	004737	010312		JSR PC,GETPATTERN		
1837	011112	005237	007200	2\$:	INC HCNTR1		; WAIT FOR THE EOP INDICATOR TO BE PRINTED
1838	011116	001375			BNE 2\$		
1839	011120	013705	000042		MOV #42,R5		; ACT11 END OF PASS HOOKS
1840	011124	001405			BEQ HERE		
1841	011126	000005			RESET		
1842	011130	004715		LOGICAL:	JSR PC,(R5)		
1843	011132	000240			NOP		
1844	011134	000240			NOF		
1845	011136	000240			NOP		
1846	011140	000207		HERE:	RTS PC		
1847							
1848	011142	000000		CHARCT:	0		



```

1849
1850 ;;*****
1851
1852 ;INITIALIZE TRACK SEQUENCE
1853
1854
1855 011144 042737 100200 001200 INITTRACK: BIC #100200,OD ;CLEAR FIRST USED BITS
1856 011152 005737 001200 TST OD ;TEST CONTENTS OF ID,OD FOR 0
1857 011156 001005 BNE 1$ ;ID,OD SPECIFIED USE THEM
1858 011160 112737 000114 001201 MOVB #114,ID ;NONE SPECIFIED SET ID TO MAXIMUM
1859 011166 105037 001200 CLRB OD ;SET OD TO MINIMUM
1860 011172 113737 001200 011360 1$: MOVB OD,TARGET ;INIT OD AS PRESENT TRACK
1861 011200 005037 011366 CLR XID ;INIT WORKING ID AND OD LOCATIONS
1862 011204 113737 001201 011366 MOVB ID,XID
1863 011212 005037 011364 CLR XOD
1864 011216 113737 001200 011364 MOVB OD,XOD
1865 011224 013737 011366 011356 MOV XID,TRKCNT ;SET UP NUMBER OF TRACK MOVEMENTS
1866 011232 163737 011364 011356 SUB XOD,TRKCNT
1867 011240 005237 011356 INC TRKCNT
1868 011244 052737 100200 001200 BIS #100200,OD ;SET FIRST TIME BITS IN ID,OD
1869 011252 000207 RTS PC
1870
1871
1872 ;;*****
1873
1874 ;TEST FOR HEAD SEQUENCE SELECTED BY INITIAL SWR SETTING
1875 ;BITS 0,1,AND 2 OF THE INITIAL SWR SELECTED THE TRACK SEQUENCING
1876 ;TO BE FOLLOWED AS INITICATED BELOW
1877
1878 ;
1879 ;
1880 ;
1881 ;
1882 ;
1883 ;
1884 ;
1885 ;
1886 ;
1887 ;
1888 ;
1889 ;
1890 ;;*****
1891
1892 011254 113737 011360 011362 GETTRACK: MOVB TARGET,PRESTRK ;RESET TO PRESENT TRACK
1893 011262 142737 000377 011320 BICB #377,2#BRONTRK ;CLEAR OUT BRANCH OFFSET
1894 011270 005737 011370 TST SEQUEN ;IF NO SEQUENCE SPECIFIED FORCE SEQUENCE 7
1895 011274 001003 BNE 1$
1896 011276 012737 000007 011370 1$: MOV #7,SEQUEN
1897 011304 013704 011370 MOV SEQUEN,R4 ;GET SEQUENCE BITS
1898 011310 005304 DEC R4 ;ADJUST FOR CORRECT OFFSET
1899 011312 006304 ASL R4
1900 011314 150437 011320 BISB R4,2#BRONTRK ;THIS BR INST. IS MODIFIED BY THE TEST CONDITIONS
1901 ;SELECTED ,TO BRANCH TO ONE OF THE SEQ. BELOW
1902
1903 011320 000777 BRONTRK: BR ;BRANCH BY OFFSET SELECTED
1904 011322 000137 011372 JMP SEQ1 ;INCREMENT
    
```







1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016

::\*\*\*\*\*

;BOUNCE BETWEEN ID AND OD ONLY  
; S = 4

011524	042737	100200	001200	SEQ4:	BIC #100200,OD	;CLEAR THE FIRST TIME BITS
011532	123737	011362	001200		CMPB PRESTRK,OD	;DID IT JUST DO OD
011540	001404				BEG IDNEXT	;YES
011542	113737	001200	011360		MOVB OD,TARGET	;NO GO TO OD TRACK
011550	000404				BR NEWTRK	
011552	113737	001201	011360	IDNEXT:	MOVB ID,TARGET	;GO DO ID TRACK
011560	000400				BR NEWTRK	

::\*\*\*\*\*

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

011562	013705	011360	NEWTRK:	MOV TARGET,R5	;GET TRACK ADDRESS
011566	006305			ASL R5	;MULTIPLY BY 4 TO DOUBLE PRECISION
011570	006305			ASL R5	;INTERLEAVE COUNTER ADDRESSES
011572	062705	017660		ADD #HDMOVE,R5	;ADD ON ADDRESS OF HEAD MOVE COUNTER
011576	062715	000001		ADD #1,(R5)	;INC COUNTER
011602	005565	000002		ADC 2(R5)	;ADD CARRY TO HIGH ORDER WORD
011606	000207			RTS PC	;RETURN TO ACCESS NEXT TRACK

::\*\*\*\*\*

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

011610	105737	001201	SEQ5:	TSTB ID	;TEST FIRST TIME BIT OF ID
011614	100011			BPL 1S	
011616	005237	011356		INC TRKCNTR	;ADJUST FOR ONE EXTRA MOVEMENT TO TRK OD
011622	142737	000200	001201	BICB #200,ID	;CLEAR FIRST TIME BIT OF ID
011630	113737	011366	011360	MOVB XID,TARGET	;MOVE TO ID
011636	000751			BR NEWTRK	
011640	105737	001200	1S:	TSTB OD	;TEST FIRST TIME BIT OF OD
011644	100007			BPL 2S	
011646	142737	000200	001200	BICB #200,OD	;CLEAR FIRST TIME BIT OF OD
011654	113737	011364	011360	MOVB XOD,TARGET	;MOVE TO OD
011662	000737			BR NEWTRK	
011664	132737	000001	011356	2S:	BITB #1,TRKCNTR
011672	001006			BNE 3S	;TEST COUNTER FOR ODD OR EVEN
011674	005237	011364		INC XOD	;ODD MOVE TO NEW ID
011700	113737	011364	011360	MOVB XOD,TARGET	;EVEN ADD 1 TO OD
011706	000405			BR LASTTRK	
011710	005337	011366		3S:	DEC XID
011714	113737	011366	011360	MOVB XID,TARGET	;SEE IF LAST TRACK IS BEING ACCESSED
011722	123727	011356	000001	LASTTRK:	CMPB TRKCNTR,#1
011730	001003			BNE XRETURN	;SUBTRACT 1 FROM ID
011732	113737	001200	011360	XRETURN:	MOVB OD,TARGET
011740	000710			BR NEWTRK	;THIS IS LAST TRACK RETURN TO OD



```

2017
2018
2019
2020
2021
2022
2023 011742 105737 001200 SEQ6: TSTB OD ;CHECK FIRST TIME BIT
2024 011746 100007 BPL 1$ ;NOT FIRST TIME
2025 011750 042737 100200 001200 BIC #100200,OD ;WAS FIRST TIME,CLEAR THE BITS
2026 011756 005337 011356 DEC TRKCNTR ;RESET COUNTER TO DOUBLE THE NUMBER OF TRACKS
2027 011762 006337 011356 ASL TRKCNTR
2028 011766 123737 011362 001200 1$: CMPB PRESTRK,OD ;WAS OD JUST ACCESSED
2029 011774 001006 BNE ODNEXT ;NO GO TO OD
2030 011776 113737 011366 011360 MOVB XID,TARGET ;DECREASING ID IS NEXT TRACK
2031 012004 005337 011366 DEC XID
2032 012010 000664 BR NEWTRK
2033 012012 113737 001200 011360 ODNEXT: MOVB OD,TARGET ;OD IS NEXT TRACK
2034 012020 000660 BR NEWTRK
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044 012022 042737 100200 001200 SEQ7: BIC #100200,OD ;CLEAR THE FIRST TIME BITS
2045 012030 004737 012122 JSR PC,RANGEN ;GET A RANDOM NUMBER
2046 012034 042737 177600 012214 BIC #177600,RANUM ;CLEAR ALL BUT LOW 7 BITS
2047 012042 123737 012214 011366 IDCOMP: CMPB RANUM,XID ;IS RANUM LARGER THAN ID ADDRESS
2048 012050 003404 BLE ODCOMP ;NO,RANUM IS LESS OR EQUAL TO ID
2049 012052 163737 011366 012214 SUB XID,RANUM ;YES,SUBTRACT ID FROM IT
2050 012060 000770 BR IDCOMP ;SEE IF RANUM IS NOW OK
2051 012062 123737 012214 011364 ODCOMP: CMPB RANUM,XOD ;IS RANUM SMALLER THAN OD ADDRESS
2052 012070 002004 BGE PRESCHK ;NO,RANUM IS GREATER OR EQUAL TO OD
2053 012072 063737 011364 012214 ADD XOD,RANUM ;YES,ADD OD TO RANUM
2054 012100 000770 BR ODCOMP ;SEE IF RANUM IS NOW OK
2055 012102 123737 012214 011362 PRESCHK: CMPB RANUM,PRESTRK ;IF RANUM EQUALS PRESENT TRACK
2056 012110 001744 BEQ SEQ7 ;GET ANOTHER RANDOM NUMBER
2057 012112 013737 012214 011360 MOV RANUM,TARGET ;RANUM OK PUT IT IN TARGET TRACK
2058 012120 000620 BR NEWTRK
2059
2060 012122 012700 000001 RANGEN: MOV #1,RO
2061 012126 063700 012210 ADD RAN1,RO
2062 012132 063700 012212 ADD RAN2,RO
2063 012136 042700 170000 BIC #170000,RO
2064 012142 000241 CLC
2065 012144 006100 ROL RO
2066 012146 006100 ROL RO
2067 012150 010037 012210 MOV RO,RAN1
2068 012154 005000 CLR RO
2069 012156 013700 012212 MOV RAN2,RO
2070 012162 006000 ROR RO
2071 012164 006000 ROR RO
2072 012166 063700 012210 ADD RAN1,RO
    
```



```

2073 012172 042700 170000          BIC #170000,RO
2074 012176 010037 012212          MOV RO,RAN2
2075 012202 010037 012214          MOV RO,RANUM
2076 012206 000207                    RTS PC
2077
2078 012210 000000          RAN1:          0
2079 012212 000000          RAN2:          0
2080 012214 000000          RANUM:         0
2081
2082          ;;*****
2083
2084          ;SECTOR INITIALIZATION AND SELECTION
2085
2086 012216 005737 001202          INITSECTOR:  TST FIRST          ;TEST FIRST AND LAST FOR 0
2087 012222 001005                    BNE 1$          ;SECTORS SPECIFIED USE THEM
2088 012224 005237 001202          INC FIRST          ;NONE SPECIFIED SET FIRST TO 1
2089 012230 112737 000032 001203  MOVB #32, LAST          ;SET LAST TO MAXIMUM
2090 012236 113737 001203 012306 1$:  MOVB LAST, SECCNTR          ;SET UP SECTOR COUNTER
2091 012244 163737 001202 012306  SUB FIRST, SECCNTR
2092 012252 005237 012306          INC SECCNTR
2093 012256 105037 012307          CLRB SECCNTR+1
2094 012262 113737 001202 012310  MOVB FIRST, TSECTOR          ;PUT FIRST SECTOR IN TARGET SECTOR
2095 012270 163737 012314 012310  SUB THREE, TSECTOR          ;SUB 3 FROM TSECTOR AS FIRST TIME THROUGH
2096          ;IT GETS ADDED BACK ON.
2097 012276 012737 000001 012312  MOV #1, INTLEAV          ;SET INTERLEAVE OFFSET
2098 012304 000207                    RTS PC
2099
2100 012306 000000          SECCNTR:       0
2101 012310 000000          TSECTOR:       0
2102 012312 000000          INTLEAV:       0
2103 012314 000003          THREE:        3
2104
2105 012316 063737 012314 012310  GETSECTOR:    ADD THREE, TSECTOR          ;ADD 3 FOR INTERLEAVING
2106 012324 123737 001203 012310  CMPB LAST, TSECTOR
2107 012332 002010                    BGE 1$          ;NEW SECTOR IS WITHIN LIMITS
2108 012334 113737 001202 012310  MOVB FIRST, TSECTOR          ;RESET TARGET SECTOR TO INTERLEAVE
2109 012342 063737 012312 012310  ADD INTLEAV, TSECTOR          ;ADD ON INTERLEAVE OFFSET VALUE
2110 012350 005237 012312          INC INTLEAV          ;UP DATE THE OFFSET VALUE
2111 012354 000207                    RTS PC
2112
2113
2114          .SBTTL TYPE ROUTINE
2115
2116          ;;*****
2117          ;ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
2118          ;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
2119          ;*NOTE1:          $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
2120          ;*NOTE2:          $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
2121          ;*NOTE3:          $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
2122          ;*
2123          ;*CALL:
2124          ;*1) USING A TRAP INSTRUCTION
2125          ;*          TYPE          ,MESADR          ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
2126          ;*OR
2127          ;*          TYPE
2128          ;*          MESADR

```



```

2129 ;*
2130
2131 012356 105737 012605 $TYPE: TSTB $TFPLG ;; IS THERE A TERMINAL?
2132 012362 100002 BPL 1$ ;; BR IF YES
2133 012364 000000 HALT ;; HALT HERE IF NO TERMINAL
2134 012366 000407 BR 3$ ;; LEAVE
2135 012370 010046 1$: MOV RO, -(SP) ;; SAVE RO
2136 012372 017600 000002 MOV 2(SP), RO ;; GET ADDRESS OF ASCIZ STRING
2137 012376 112046 2$: MOVB (RO)+, -(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
2138 012400 001005 BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
2139 012402 005726 TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
2140 012404 012600 60$: MOV (SP)+, RO ;; RESTORE RO
2141 012406 062716 3$: ADD #2, (SP) ;; ADJUST RETURN PC
2142 012412 000002 RTI ;; RETURN
2143 012414 122716 4$: CMPB #HT, (SP) ;; BRANCH IF <HT>
2144 012420 001430 BEQ 8$
2145 012422 122716 000200 CMPB #CRLF, (SP) ;; BRANCH IF NOT <CRLF>
2146 012426 001006 BNE 5$
2147 012430 005726 TST (SP)+ ;; POP <CR><LF> EQUIV
2148 012432 104400 TYPE ;; TYPE A CR AND LF
2149 012434 012607 SCRLF
2150 012436 105037 012572 CLRB $CHARCNT ;; CLEAR CHARACTER COUNT
2151 012442 000755 BR 2$ ;; GET NEXT CHARACTER
2152 012444 004737 012526 5$: JSR PC, $TYPEC ;; GO TYPE THIS CHARACTER
2153 012450 123726 012604 6$: CMPB $FILLC, (SP)+ ;; IS IT TIME FOR FILLER CHARS.?
2154 012454 001350 BNE 2$ ;; IF NO GO GET NEXT CHAR.
2155 012456 013746 012602 MOV $NULL, -(SP) ;; GET # OF FILLER CHARS. NEEDED
2156 AND THE NULL CHAR.
2157 012462 105366 000001 7$: DECB 1(SP) ;; DOES A NULL NEED TO BE TYPED?
2158 012466 002770 BLT 6$ ;; BR IF NO--GO POP THE NULL OFF OF STACK
2159 012470 004737 012526 JSR PC, $TYPEC ;; GO TYPE A NULL
2160 012474 105337 012572 DECB $CHARCNT ;; DO NOT COUNT AS A COUNT
2161 012500 000770 BR 7$ ;; LOOP
2162
2163 ;HORIZONTAL TAB PROCESSOR
2164
2165 012502 112716 000040 8$: MOVB #' (SP) ;; REPLACE TAB WITH SPACE
2166 012506 004737 012526 9$: JSR PC, $TYPEC ;; TYPE A SPACE
2167 012512 132737 000007 012572 BITB #7, $CHARCNT ;; BRANCH IF NOT AT
2168 012520 001372 BNE 9$ ;; TAB STOP
2169 012522 005726 TST (SP)+ ;; POP SPACE OFF STACK
2170 012524 000724 BR 2$ ;; GET NEXT CHARACTER
2171 012526 105777 000044 $TYPEC: TSTB 2$TPS ;; WAIT UNTIL PRINTER IS READY
2172 012532 100375 BPL $TYPEC
2173 012534 116677 000002 000036 MOVB 2(SP), 2$TPB ;; LOAD CHAR TO BE TYPED INTO DATA REG.
2174 012542 122766 000015 000002 CMPB #CR, 2(SP) ;; IS CHARACTER A CARRIAGE RETURN?
2175 012550 001003 BNE 1$ ;; BRANCH IF NO
2176 012552 105037 012572 CLRB $CHARCNT ;; YES--CLEAR CHARACTER COUNT
2177 012556 000406 BR $TYPEX ;; EXIT
2178 012560 122766 000012 000002 1$: CMPB #LF, 2(SP) ;; IS CHARACTER A LINE FEED?
2179 012566 001402 BEQ $TYPEX ;; BRANCH IF YES
2180 012570 105227 INCB (PC)+ ;; COUNT THE CHARACTER
2181 012572 000000 $CHARCNT: .WORD 0 ;; CHARACTER COUNT STORAGE
2182 012574 000207 $TYPEX: RTS PC
2183
2184 012576 177564 $TPS: .WORD 177564 ;; TTY PRINTER STATUS REG. ADDRESS
    
```



2185	012600	177566		\$TPB:	.WORD	177566		::	TTY PRINTER BUFFER REG. ADDRESS
2186	012602	000		\$NULL:	.BYTE	0		::	CONTAINS NULL CHARACTER FOR FILLS
2187	012603	002		\$FILLS:	.BYTE	2		::	CONTAINS # OF FILLER CHARACTERS REQUIRED
2188	012604	012		\$FILLC:	.BYTE	12		::	INSERT FILL CHARS. AFTER A "LINE FEED"
2189	012605	000		\$TPFLG:	.BYTE	0		::	"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
2190	012606	077		\$QUES:	.ASCII	"?"		::	QUESTION MARK
2191	012607	015		\$CRLF:	.ASCII	<15>		::	CARRIAGE RETURN
2192	012610	000012		\$LF:	.ASCIZ	<12>		::	LINEFEED
2193				.SBTTL				::	BINARY TO OCTAL (ASCII) AND TYPE
2194								::	*****
2195								::	*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
2196								::	*OCTAL (ASCII) NUMBER AND TYPE IT.
2197								::	*\$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
2198								::	*CALL:
2199				*	MOV	NUM,-(SP)		::	NUMBER TO BE TYPED
2200				*	TYPOS			::	CALL FOR TYPEOUT
2201				*	.BYTE	N		::	N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2202				*	.BYTE	M		::	M=1 OR 0
2203				*				::	1=TYPE LEADING ZEROS
2204				*				::	0=SUPPRESS LEADING ZEROS
2205								::	*\$STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2206								::	*\$TYPOS OR \$TYPOC
2207								::	*CALL:
2208				*	MOV	NUM,-(SP)		::	NUMBER TO BE TYPED
2209				*	TYPON			::	CALL FOR TYPEOUT
2210								::	*\$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2211								::	*CALL:
2212				*	MOV	NUM,-(SP)		::	NUMBER TO BE TYPED
2213				*	TYPOC			::	CALL FOR TYPEOUT
2214								::	*\$TYPOS: MOV
2215	012612	017646	000000		2(SP),-(SP)			::	PICKUP THE MODE
2216	012616	116637	000001	013035	MOV	1(SP), \$OFILL		::	LOAD ZERO FILL SWITCH
2217	012624	112637	013037		MOV	(SP)+, \$OMODE+1		::	NUMBER OF DIGITS TO TYPE
2218	012630	062716	000002		ADD	#2, (SP)		::	ADJUST RETURN ADDRESS
2219	012634	000406			BR	\$TYPON		::	
2220	012636	112737	000001	013035	\$TYPOC: MOV	#1, \$OFILL		::	SET THE ZERO FILL SWITCH
2221	012644	112737	000006	013037	MOV	#6, \$OMODE+1		::	SET FOR SIX(6) DIGITS
2222	012652	112737	000005	013034	\$TYPON: MOV	#5, \$OCNT		::	SET THE ITERATION COUNT
2223	012660	010346			MOV	R3,-(SP)		::	SAVE R3
2224	012662	010446			MOV	R4,-(SP)		::	SAVE R4
2225	012664	010546			MOV	R5,-(SP)		::	SAVE R5
2226	012666	113704	013037		MOV	\$OMODE+1, R4		::	GET THE NUMBER OF DIGITS TO TYPE
2227	012672	005404			NEG	R4		::	
2228	012674	062704	000006		ADD	#6, R4		::	SUBTRACT IT FOR MAX. ALLOWED
2229	012700	110437	013036		MOV	R4, \$OMODE		::	SAVE IT FOR USE
2230	012704	113704	013035		MOV	\$OFILL, R4		::	GET THE ZERO FILL SWITCH
2231	012710	016605	000012		MOV	12(SP), R5		::	PICKUP THE INPUT NUMBER
2232	012714	005003			CLR	R3		::	CLEAR THE OUTPUT WORD
2233	012716	006105		15:	ROL	R5		::	ROTATE MSB INTO "C"
2234	012720	000404			BR	\$S		::	GO DO MSB
2235	012722	006105		25:	ROL	R5		::	FORM THIS DIGIT
2236	012724	006105			ROL	R5		::	
2237	012726	006105			ROL	R5		::	



```

2241 012730 010503          MOV      R5,R3
2242 012732 006103          3$:    ROL      R3          ;; GET LSB OF THIS DIGIT
2243 012734 105337 013036  DEC8     $OMODE        ;; TYPE THIS DIGIT?
2244 012740 100016          BPL      7$          ;; BR IF NO
2245 012742 042703 177770  BIC      #177770,R3    ;; GET RID OF JUNK
2246 012746 001002          BNE      4$          ;; TEST FOR 0
2247 012750 005704          TST      R4          ;; SUPPRESS THIS 0?
2248 012752 001403          BEQ      5$          ;; BR IF YES
2249 012754 005204          4$:    INC      R4          ;; DON'T SUPPRESS ANYMORE 0'S
2250 012756 052703 000060  BIS      #'0,R3       ;; MAKE THIS DIGIT ASCII
2251 012762 052703 000040  5$:    BIS      #' ,R3    ;; MAKE ASCII IF NOT ALREADY
2252 012766 110337 013032  MOVB     R3,8$        ;; SAVE FOR TYPING
2253 012772 104400 013032  TYPE     ,8$          ;; GO TYPE THIS DIGIT
2254 012776 105337 013034  7$:    DEC8     $OCNT        ;; COUNT BY 1
2255 013002 003347          BGT      2$          ;; BR IF MORE TO DO
2256 013004 002402          BLT      6$          ;; BR IF DONE
2257 013006 005204          INC      R4          ;; INSURE LAST DIGIT ISN'T A BLANK
2258 013010 000744          BR       2$          ;; GO DO THE LAST DIGIT
2259 013012 012605          6$:    MOV      (SP)+,R5  ;; RESTORE R5
2260 013014 012604          MOV      (SP)+,R4    ;; RESTORE R4
2261 013016 012603          MOV      (SP)+,R3    ;; RESTORE R3
2262 013020 016666 000002 000004  MOV      2(SP),4(SP) ;; SET THE STACK FOR RETURNING
2263 013026 012616          MOV      (SP)+,(SP)
2264 013030 000002          RTI          ;; RETURN
2265 013032 000          8$:    .BYTE    0          ;; STORAGE FOR ASCII DIGIT
2266 013033 000          .BYTE    0          ;; TERMINATOR FOR TYPE ROUTINE
2267 013034 000          $OCNT:   .BYTE    0          ;; OCTAL DIGIT COUNTER
2268 013035 000          $DFILL:  .BYTE    0          ;; ZERO FILL SWITCH
2269 013036 000000          $OMODE:  .WORD    0          ;; NUMBER OF DIGITS TO TYPE
2270          .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
2271
2272          ;; *****
2273          ;; *SAVE R0-R5
2274          ;; *CALL:
2275          ;; * SAVREG
2276          ;; *UPON RETURN FROM $$SAVREG THE STACK WILL LOOK LIKE:
2277          ;; *
2278          ;; *TOP---(+16)
2279          ;; * +2---(+18)
2280          ;; * +4---R5
2281          ;; * +6---R4
2282          ;; * +8---R3
2283          ;; *+10---R2
2284          ;; *+12---R1
2285          ;; *+14---R0
2286
2287          ;; $SAVREG:
2288          013040 010046          MOV      R0,-(SP)    ;; PUSH R0 ON STACK
2289          013042 010146          MOV      R1,-(SP)    ;; PUSH R1 ON STACK
2290          013044 010246          MOV      R2,-(SP)    ;; PUSH R2 ON STACK
2291          013046 010346          MOV      R3,-(SP)    ;; PUSH R3 ON STACK
2292          013050 010446          MOV      R4,-(SP)    ;; PUSH R4 ON STACK
2293          013052 010546          MOV      R5,-(SP)    ;; PUSH R5 ON STACK
2294          013054 016646 000022  MOV      22(SP),-(SP) ;; SAVE PS OF MAIN FLOW
2295          013060 016646 000022  MOV      22(SP),-(SP) ;; SAVE PC OF MAIN FLOW
2296          013064 016646 000022  MOV      22(SP),-(SP) ;; SAVE PS OF CALL

```



```

2297 013070 016646 000022          MOV      22(SP),-(SP)    ;;SAVE PC OF CALL
2298 013074 000002          RTI
2299
2300          ;*RESTORE RO-R5
2301          ;*CALL:
2302          ;* RESREG
2303          $RESREG:
2304 013076 012666 000022          MOV      (SP)+,22(SP)    ;;RESTORE PC OF CALL
2305 013102 012666 000022          MOV      (SP)+,22(SP)    ;;RESTORE PS OF CALL
2306 013106 012666 000022          MOV      (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
2307 013112 012666 000022          MOV      (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
2308 013116 012605          MOV      (SP)+,R5        ;;POP STACK INTO R5
2309 013120 012604          MOV      (SP)+,R4        ;;POP STACK INTO R4
2310 013122 012603          MOV      (SP)+,R3        ;;POP STACK INTO R3
2311 013124 012602          MOV      (SP)+,R2        ;;POP STACK INTO R2
2312 013126 012601          MOV      (SP)+,R1        ;;POP STACK INTO R1
2313 013130 012600          MOV      (SP)+,R0        ;;POP STACK INTO R0
2314 013132 000002          RTI
2315          .SBTTL  TTY INPUT ROUTINE
2316
2317          ;*****
2318 013134 177560          $TKS:   .WORD 177560      ;;TTY KBD STATUS
2319 013136 177562          $TKB:   .WORD 177562      ;;TTY KBD BUFFER
2320          .ENABL  LSB
2321
2322          ;*****
2323          ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2324          ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2325          ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2326          ;*WHEN OPERATING IN TTY FLAG MODE.
2327 013140 022737 000176 001214  $CKSWR:  CMP      #SWREG,SWR    ;;IS THE SOFT-SWR SELECTED?
2328 013146 001074          BNE     15$              ;;BRANCH IF NO
2329 013150 105777 177760          TSTB   @TKS             ;;CHAR THERE?
2330 013154 100071          BPL     15$              ;;IF NO, DON'T WAIT AROUND
2331 013156 117746 177754          MOVB   @TKB, -(SP)      ;;SAVE THE CHAR
2332 013162 042716 177600          BIC    #1C177, (SP)     ;;STRIP-OFF THE ASCII
2333 013166 022726 000007          CMP    #7, (SP)+        ;;IS IT A CONTROL G?
2334 013172 001062          BNE     15$              ;;NO, RETURN TO USER
2335 013174 123727 013714 000001  CMPB   $AUTOB, #1       ;;ARE WE RUNNING IN AUTO-MODE?
2336 013202 001456          BEQ    15$              ;;BRANCH IF YES
2337
2338 013204 104400 013665          SGTSWR: TYPE ,SCNTLG     ;;ECHO THE CONTROL-G (↑G)
2339 013210 104400 013672          TYPE ,SMSWR            ;;TYPE CURRENT CONTENTS
2340 013214 013746 000176          MOV    SWREG, -(SP)     ;;SAVE SWREG FOR TYPEOUT
2341 013220 104401          TYPOC                    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2342 013222 104400 013703          TYPE ,SMNEW            ;;PROMPT FOR NEW SWR
2343 013226 005046          19$:  CLR    -(SP)        ;;CLEAR COUNTER
2344 013230 005046          CLR    -(SP)           ;;THE NEW SWR
2345 013232 105777 177676          7$:   TSTB   @TKS        ;;CHAR THERE?
2346 013236 100375          BPL    7$              ;;IF NOT TRY AGAIN
2347
2348 013240 117746 177672          MOVB   @TKB, -(SP)     ;;PICK UP CHAR
2349 013244 042716 177600          BIC    #1C177, (SP)    ;;MAKE IT 7-BIT ASCII
2350
2351
2352

```



```

2353 013250 021627 000025 9$: CMP (SP),#25 ;: IS IT A CONTROL-U?
2354 013254 001005 ;: BNE 10$ ;: BRANCH IF NOT
2355 013256 104400 013660 ;: TYPE ,SCNTLU ;: YES, ECHO CONTROL-U (1U)
2356 013262 062706 000006 20$: ADD #6,SP ;: IGNORE PREVIOUS INPUT
2357 013266 000757 BR 19$ ;: LET'S TRY IT AGAIN
2358
2359
2360 013270 021627 000015 10$: CMP (SP),#15 ;: IS IT A <CR>?
2361 013274 001022 ;: BNE 16$ ;: BRANCH IF NO
2362 013276 005766 000004 ;: TST 4(SP) ;: YES, IS IT THE FIRST CHAR?
2363 013302 001403 ;: BEQ 11$ ;: BRANCH IF YES
2364 013304 016677 000002 165702 ;: MOV 2(SP),@SWR ;: SAVE NEW SWR
2365 013312 062706 000006 11$: ADD #6,SP ;: CLEAR UP STACK
2366 013316 104400 012607 14$: TYPE ,$CRLF ;: ECHO <CR> AND <LF>
2367 013322 123727 013715 000001 ;: CMPB $INTAG,#1 ;: RE-ENABLE TTY KBD INTERRUPTS?
2368 013330 001003 ;: BNE 15$ ;: BRANCH IF NOT
2369 013332 012777 000100 177574 ;: MOV #100,@$TKS ;: RE-ENABLE TTY KBD INTERRUPTS
2370 013340 000002 15$: RTI ;: RETURN
2371 013342 004737 012526 16$: JSR PC,$TYPEC ;: ECHO CHAR
2372 013346 021627 000060 ;: CMP (SP),#60 ;: CHAR < 0?
2373 013352 002420 ;: BLT 18$ ;: BRANCH IF YES
2374 013354 021627 000067 ;: CMP (SP),#67 ;: CHAR > 7?
2375 013360 003015 ;: BGT 18$ ;: BRANCH IF YES
2376 013362 042726 000060 ;: BIC #60,(SP)+ ;: STRIP-OFF ASCII
2377 013366 005766 000002 ;: TST 2(SP) ;: IS THIS THE FIRST CHAR
2378 013372 001403 ;: BEQ 17$ ;: BRANCH IF YES
2379 013374 006316 ;: ASL (SP) ;: NO, SHIFT PRESENT
2380 013376 006316 ;: ASL (SP) ;: CHAR OVER TO MAKE
2381 013400 006316 ;: ASL (SP) ;: ROOM FOR NEW ONE.
2382 013402 005266 000002 17$: INC 2(SP) ;: KEEP COUNT OF CHAR
2383 013406 056616 177776 ;: BIS -2(SP),(SP) ;: SET IN NEW CHAR
2384 013412 000707 ;: BR 7$ ;: GET THE NEXT ONE
2385 013414 104400 012606 18$: TYPE ,SQUES ;: TYPE ?<CR><LF>
2386 013420 000720 BR 20$ ;: SIMULATE CONTROL-U
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398 013422 011646 ;: SRDCHR: MOV (SP),-(SP) ;: PUSH DOWN THE PC
2399 013424 016666 000004 000002 ;: MOV 4(SP),2(SP) ;: SAVE THE PS
2400 013432 105777 177476 1$: TSTB @$TKS ;: WAIT FOR
2401 013436 100375 ;: BPL 1$ ;: A CHARACTER
2402 013440 117766 177472 000004 ;: MOVB @$TKB,4(SP) ;: READ THE TTY
2403 013446 042766 177600 000004 ;: BIC #1C<177>,4(SP) ;: GET RID OF JUNK IF ANY
2404 013454 026627 000004 000023 ;: CMP 4(SP),#23 ;: IS IT A CONTROL-S?
2405 013462 001013 ;: BNE 3$ ;: BRANCH IF NO
2406 013464 105777 177444 2$: TSTB @$TKS ;: WAIT FOR A CHARACTER
2407 013470 100375 ;: BPL 2$ ;: LOOP UNTIL ITS THERE
2408 013472 117746 177440 ;: MOVB @$TKB,-(SP) ;: GET CHARACTER

```

\*\*\*\*\*

THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY

\*CALL:

\* RDCHR  
\* RETURN HERE

;; INPUT A SINGLE CHARACTER FROM THE TTY  
;; CHARACTER IS ON THE STACK  
;; WITH PARITY BIT STRIPPED OFF

;







2465  
2466 013716 010046  
2467 013720 016600 000002  
2468 013724 005740  
2469 013726 111000  
2470 013730 006300  
2471 013732 016000 013740  
2472 013736 000200

```
$TRAP: MOV RO, -(SP) ;;SAVE RO
        MOV 2(SP), RO ;;GET TRAP ADDRESS
        TST -(RO) ;;BACKUP BY 2
        MOVB (RO), RO ;;GET RIGHT BYTE OF TRAP
        ASL RO ;;POSITION FOR INDEXING
        MOV $TRPAD(RO), RO ;;INDEX TO TABLE
        RTS RO ;;GO TO ROUTINE
```

.SBTTL TRAP TABLE

;\*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED  
;\*BY THE "TRAP" INSTRUCTION.

ROUTINE  
-----

2481 013740  
2482 013740 012356  
2483 013742 012636  
2484 013744 012612  
2485 013746 012652  
2486  
2487 013750 013210  
2488  
2489 013752 013140  
2490 013754 013422  
2491 013756 013542  
2492 013760 013040  
2493 013762 013076

```
$TRPAD: $TYPE ;;CALL=TYPE TRAP+0(104400) TTY TYPEOUT ROUTINE
        $TYPOC ;;CALL=TYPOC TRAP+1(104401) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
        $TYPOS ;;CALL=TYPOS TRAP+2(104402) TYPE OCTAL NUMBER (NO LEADING ZEROS)
        $TYPON ;;CALL=TYPON TRAP+3(104403) TYPE OCTAL NUMBER (AS PER LAST CALL)

        $GTSWR ;;CALL=GTSWR TRAP+4(104404) GET SOFT-SWR SETTING

        $CKSWR ;;CALL=CKSWR TRAP+5(104405) TEST FOR CHANGE IN SOFT-SWR
        $RDCHR ;;CALL=RDCHR TRAP+6(104406) TTY TYPEIN CHARACTER ROUTINE
        $RDLIN ;;CALL=RDLIN TRAP+7(104407) TTY TYPEIN STRING ROUTINE
        $SAVREG ;;CALL=SAVREG TRAP+10(104410) SAVE RO-R5 ROUTINE
        $RESREG ;;CALL=RESREG TRAP+11(104411) RESTORE RO-R5 ROUTINE
```

.SBTTL POWER DOWN AND UP ROUTINES

\*\*\*\*\*  
:POWER DOWN ROUTINE

2497  
2498 013764 012737 014130 000024  
2499 013772 012737 000340 000026  
2500 014000 010046  
2501 014002 010146  
2502 014004 010246  
2503 014006 010346  
2504 014010 010446  
2505 014012 010546  
2506 014014 017746 165174  
2507 014020 010637 014134  
2508 014024 012737 014036 000024  
2509 014032 000000  
2510 014034 000776

```
$PWRDN: MOV $SILLUP, @#PWRVEC ;;SET FOR FAST UP
        MOV #340, @#PWRVEC+2 ;;PRIO:7
        MOV RO, -(SP) ;;PUSH RO ON STACK
        MOV R1, -(SP) ;;PUSH R1 ON STACK
        MOV R2, -(SP) ;;PUSH R2 ON STACK
        MOV R3, -(SP) ;;PUSH R3 ON STACK
        MOV R4, -(SP) ;;PUSH R4 ON STACK
        MOV R5, -(SP) ;;PUSH R5 ON STACK
        MOV @SWR, -(SP) ;;PUSH @SWR ON STACK
        MOV SP, $SAVR6 ;;SAVE SP
        MOV $PWRUP, @#PWRVEC ;;SET UP VECTOR
        HALT
        BR .-2 ;;HANG UP
```

\*\*\*\*\*  
:POWER UP ROUTINE

2513  
2514 014036 012737 014130 000024  
2515 014044 013706 014134  
2516 014050 005037 014134  
2517 014054 005237 014134  
2518 014060 001375  
2519 014062 012677 165126  
2520 014066 012605

```
$PWRUP: MOV $SILLUP, @#PWRVEC ;;SET FOR FAST DOWN
        MOV $SAVR6, SP ;;GET SP
        CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
        1$: INC $SAVR6 ;;WAIT FOR THE INC
        BNE 1$ ;;OF WORD
        MOV (SP)+, @SWR ;;POP STACK INTO @SWR
        MOV (SP)+, R5 ;;POP STACK INTO R5
```



```

2521 014070 012604      MOV      (SP)+,R4      ;; POP STACK INTO R4
2522 014072 012603      MOV      (SP)+,R3      ;; POP STACK INTO R3
2523 014074 012602      MOV      (SP)+,R2      ;; POP STACK INTO R2
2524 014076 012601      MOV      (SP)+,R1      ;; POP STACK INTO R1
2525 014100 012600      MOV      (SP)+,R0      ;; POP STACK INTO R0
2526 014102 012737 013764 000024      MOV      #SPWRDN, @#PWRVEC ;; SET UP THE POWER DOWN VECTOR
2527 014110 012737 000340 000026      MOV      #340, @#PWRVEC+2 ;; PRIO:7
2528 014116 104400      TYPE      ;; REPORT THE POWER FAILURE
2529 014120 014136      $PWRMG: .WORD $POWER ;; POWER FAIL MESSAGE POINTER
2530 014122 012716      MOV      (PC)+, (SP) ;; RESTART AT RESTART
2531 014124 002700      $PWRAD: .WORD RESTART ;; RESTART ADDRESS
2532 014126 000002      RTI      ;;
2533 014130 000000      $ILLUP: HALT ;; THE POWER UP SEQUENCE WAS STARTED
2534 014132 000776      BR      -2 ;; BEFORE THE POWER DOWN WAS COMPLETE
2535 014134 000000      $$SAVR6: 0 ;; PUT THE SP HERE
2536 014136 005015 047520 042527      $POWER: .ASCIZ <15><12>"POWER"
2537 014144 000122
2538
2539      .EVEN
2540      .SBTTL SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE
2541
2542      ;; *****
2543      ;; *THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
2544      ;; *UNSIGNED DECIMAL ASCIZ NUMBER.
2545      ;; *CALL
2546      ;; * MOV NUMBER, -(SP) ;; PUT BINARY NUMBER ON THE STACK
2547      ;; * JSR PC, @#$SB2D ;; CALL
2548      ;; * RETURN ;; ADDRESS OF THE 1ST ASCIZ CHAR. IS ON THE STACK
2549
2550 014146 016637 000002 014176      $SB2D: MOV      2(SP), 1$ ;; SAVE BINARY NUMBER
2551 014154 012746 014176      MOV      #1$, -(SP) ;; SET POINTER
2552 014160 004737 014202      JSR      PC, @#$DB2D ;; CALL DOUBLE LENGTH CONVERT
2553 014164 062716 000005      ADD      #5, (SP) ;; ONLY ALLOW FIVE CHARACTERS
2554 014170 012666 000002      MOV      (SP)+, 2(SP) ;; PICKUP POINTER
2555 014174 000207      RTS      PC ;; RETURN
2556 014176 000000 000000      1$: .WORD 0,0
2557      .SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
2558
2559      ;; *****
2560      ;; *THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
2561      ;; *DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
2562      ;; *POSITIVE.
2563      ;; *CALL
2564      ;; * MOV #PNTR, -(SP) ;; POINTER TO LOW WORD OF BINARY NUMBER
2565      ;; * JSR PC, @#$DB2D
2566      ;; * RETURN ;; THE FIRST ADDRESS OF ASCIZ
2567      ;; ;; IS ON THE STACK
2568
2569
2570 014202 104410      $DB2D: SAVREG ;; SAVE REGISTERS
2571 014204 016602 000002      MOV      2(SP), R2 ;; PICKUP THE DATA POINTER
2572 014210 012700 014362      MOV      #$DECVL, R0 ;; GET ADDRESS OF "$DECVL" STRING
2573 014214 010066 000002      MOV      R0, 2(SP) ;; PUT ADDRESS OF ASCIZ STRING ON STACK
2574 014220 012201      MOV      (R2)+, R1 ;; PICKUP THE BINARY NUMBER
2575 014222 012202      MOV      (R2)+, R2
2576 014224 012737 000012 014300      MOV      #10., 4$ ;; SET UP TO DO 10 CONVERSIONS

```



# M05

MAINDEC-11-DZRXA-E MACY11 27(732) 20-SEP-76 15:46 PAGE 48  
 DZRXA.E.P11 DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

```

2577 014232 012704 014312      MOV      # $TNPWR,R4      ;; ADDRESS OF TEN POWER
2578 014236 012705 014314      MOV      # $TNPWR+2,R5
2579 014242 005003      1$: CLR      R3      ;; CLEAR PARTIAL
2580 014244 161401      2$: SUB      (R4),R1      ;; SUBTRACT TEN POWER
2581 014246 005602      SBC      R2
2582 014250 161502      SUB      (R5),R2
2583 014252 002402      BLT      3$      ;; BR IF TEN POWER TOO LARGE
2584 014254 005203      INC      R3      ;; ADD 1 TO PARTIAL
2585 014256 000772      BR      2$      ;; LOOP
2586 014260 062401      3$: ADD      (R4)+,R1      ;; RESTORE SUBTRACTED VALUE
2587 014262 005502      ADC      R2
2588 014264 062402      ADD      (R4)+,R2
2589 014266 022525      CMP      (R5)+,(R5)+      ;; MOVE TO NEXT TEN POWER
2590 014270 052703 000060      BIS      #'0,R3      ;; CHANGE PARTIAL TO ASCII
2591 014274 110320      MOV      R3,(R0)+      ;; SAVE IT
2592 014276 005327      DEC      (PC)+      ;; DONE?
2593 014300 000000      4$: .WORD    0
2594 014302 001357      BNE      1$      ;; BR IF NO
2595 014304 105020      CLRB     (R0)+      ;; TERMINATOR
2596 014306 104411      RESREG
2597 014310 000207      RTS      PC      ;; RESTORE REGISTERS
2598 014312 145000      $TNPWR: 145000      ;; RETURN
2599 014314 035632      35632      ;; 1.0E09
2600 014316 160400      160400      ;; 1.0E08
2601 014320 002765      2765      ;; 1.0E07
2602 014322 113200      113200      ;; 1.0E06
2603 014324 000230      230      ;; 1.0E05
2604 014326 041100      041100      ;; 1.0E04
2605 014330 000017      17      ;; 1.0E03
2606 014332 103240      103240      ;; 1.0E02
2607 014334 000001      1      ;; 1.0E01
2608 014336 023420      23420      ;; 1.0E00
2609 014340 000000      0
2610 014342 001750      1750
2611 014344 000000      0
2612 014346 000144      144
2613 014350 000000      0
2614 014352 000012      12
2615 014354 000000      0
2616 014356 000001      1
2617 014360 000000      0
2618 014362 000014      $DECVL: .BLKB 12.      ;; RESERVE STORAGE FOR ASCII STRING
2619
2620      ;; *****
2621
2622      ; TYPE NUMERICAL ASCII STRING, RIGHT JUSTIFIED
2623      ; REPLACING LEADING ZEROS WITH SPACES.
2624      ;
2625      ; FIRST ADDRESS OF ASCII STRING MUST BE ON TOP OF THE STACK
2626
2627 014376 010046      RTJUST: MOV      R0,-(SP)      ; SAVE R0
2628 014400 016600 000004      MOV      4(SP),R0      ; PICK UP ADDRESS OF ASCII STRING
2629 014404 010037 014436      MOV      R0,3$      ; SAVE ADDRESS FOR TYPE OUT
2630 014410 105710      1$: TSTB     (R0)      ; IS THIS THE TERMINATOR
2631 014412 001406      BEQ      2$      ; IF YES TYPE IT OUT
2632 014414 122710 000060      CMPB     #'0,(R0)      ; IS IT A ZERO

```



2633	014420	001005				BNE 4\$	: IF NO GO PRINT IT
2634	014422	112720	000040			MOVB #' , (RO)+	: IF YES REPLACE IT WITH A SPACE
2635	014426	000770				BR 1\$	: TEST NEXT CHAR.
2636	014430	112740	000060		2\$:	MOVB #'0, -(RO)	: STRING OFF ALL ZEROS, PUT BACK THE LAST ONE
2637	014434	104400			4\$:	TYPE	: TYPE THE STRING
2638	014436	000000			3\$:	OPEN	
2639	014440	012600				MOV (SP)+, RO	: RESTORE RO
2640	014442	012616				MOV (SP)+, (SP)	: RESTORE THE STACK
2641	014444	000207				RTS PC	: RETURN
2642							
2643							
2644							
2645	014446	012546					
2646	014450	004737	014146		SGLDEC:	MOV (R5)+, -(SP)	: PUT NUMBER TO BE TYPED ON STACK
2647	014454	004737	014376			JSR PC, @#\$\$SB2D	: CONVERT NUMBER TO DECIMAL
2648	014460	000205				JSR PC, RTJUST	: TYPE THE DECIMAL NUMBER
2649						RTS R5	
2650							
2651	014462	047125	054105	042520	MUNXDD:	.ASCIZ "UNEXPECTED D D MARK"	
2652	014470	052103	042105	042040			
2653	014476	042040	046440	051101			
2654	014504	000113					
2655							
2656	014506	020104	020104	040515	MDDMIS:	.ASCIZ "D D MARK MISSING"	
2657	014514	045522	046440	051511			
2658	014522	044523	043516	000			
2659							
2660							
2661	014527	104	052101	026101	MDERHDR:	.ASCIZ "DATA, NO STATUS ERROR"	
2662	014534	047040	020117	052123			
2663	014542	052101	051525	042440			
2664	014550	051122	051117	000			
2665							
2666	014555	040	047117	052040	MTRK:	.ASCIZ " ON TRACK"	
2667	014562	040522	045503	000			
2668							
2669							
2670	014567	040	043040	047522	MPRES:	.ASCIZ " FROM TRACK"	
2671	014574	020115	051124	041501			
2672	014602	000113					
2673							
2674	014604	027440	051440	041505	MSECT:	.ASCIZ " / SECTOR"	
2675	014612	047524	000122				
2676							
2677	014616	005015	041040	052131	MCOLMUN:	.ASCIZ <15><12>" BYTE BAD GOOD"<15><12>	
2678	014624	020105	041040	042101			
2679	014632	020040	047507	042117			
2680	014640	005015	000				
2681							
2682	014643	015	041412	052501	DOLOAD:	.ASCII <15><12>"CAUTION - IF YOU DESIRE TO TEST UNIT 0"	
2683	014650	044524	047117	026440			
2684	014656	044440	020106	047531			
2685	014664	020125	042504	044523			
2686	014672	042522	052040	020117			
2687	014700	042524	052123	052440			
2688	014706	044516	020124	060			



2689	014713	015	051012	050105		.ASCII <15><12>"REPLACE LOAD MEDIUM WITH A SCRATCH DISKETTE"
2690	014720	040514	042503	046040		
2691	014726	040517	020104	042515		
2692	014734	044504	046525	053440		
2693	014742	052111	020110	020101		
2694	014750	041523	040522	041524		
2695	014756	020110	044504	045523		
2696	014764	052105	042524			
2697	014770	005015	044124	047105		.ASCIZ <15><12>"THEN PRESS CONTINUE"
2698	014776	050040	042522	051523		
2699	015004	041440	047117	044524		
2700	015012	052516	000105			
2701						
2702	015016	047516	042040	044522	MNODRY:	.ASCIZ "NO DRIVES READY"<15><12>
2703	015024	042526	020123	042522		
2704	015032	042101	006531	000012		
2705						
2706	015040	020040	020040	020040	TAB:	.ASCIZ <40><40><40><40><40><40>
2707	015046	000				
2708						
2709	015047	040	000040		DBLSP:	.ASCIZ <40><40>
2710						
2711	015052	005015	000		MCRLF:	.ASCIZ <15><12>
2712						
2713	015055	104	000		MEOP:	.ASCIZ "D"
2714						
2715	015057	055	000		MEREOP:	.ASCIZ "--"
2716						
2717	015061	007	000		MABELL:	.ASCIZ <07>
2718						
2719	015063	105	051122	051117	MERHEADER:	.ASCIZ "ERROR CONDITIONS "
2720	015070	041440	047117	044504		
2721	015076	044524	047117	020123		
2722	015104	000040				
2723						
2724	015106	047125	052111	030040	MUNIT0:	.ASCIZ "UNIT 0 "
2725	015114	000040				
2726						
2727	015116	047125	052111	030440	MUNIT1:	.ASCIZ "UNIT 1 "
2728	015124	000040				
2729						
2730	015126	040510	020123	042502	MDELET:	.ASCIZ "HAS BEEN DELETED."
2731	015134	047105	042040	046105		
2732	015142	052105	042105	000056		
2733						
2734	015150	044440	052116	053040	MINTVEC:	.ASCIZ " INT VECTOR = "
2735	015156	041505	047524	020122		
2736	015164	020075	000			
2737						
2738	015167	122	041530	020123	MRXCS:	.ASCIZ "RXCS = "
2739	015174	020075	000			
2740						
2741	015177	040	054122	041104	MRXDB:	.ASCIZ " RXDB = "
2742	015204	036440	000040			
2743						
2744	015210	052123	052101	051525	MASTAT:	.ASCIZ "STATUS A = "



MAINDEC-11-DZRXA-E MACY11 27(732) 20-SEP-76 15:46 PAGE 51  
 DZRXA.E.P11 DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

2745	015216	040440	036440	000040		
2746						
2747	015224	052123	052101	051525	MBSTAT:	.ASCIZ "STATUS B = "
2748	015232	041040	036440	000040		
2749						
2750	015240	005015	000012		DBLLF:	.ASCIZ <15><12><12>
2751						
2752	015244	036520	000040		MPAT:	.ASCIZ "P= "
2753						
2754	015250	036524	000040		MTEST:	.ASCIZ "T= "
2755						
2756	015254	036523	000040		MSEQ:	.ASCIZ "S= "
2757						
2758	015260	047516	044440	052116	MINTER:	.ASCIZ "NO INTERRUPT AT DONE ERROR"
2759	015266	051105	050125	020124		
2760	015274	052101	042040	047117		
2761	015302	020105	051105	047522		
2762	015310	000122				
2763						
2764	015312	047125	047113	053517	MUKNINT:	.ASCIZ "UNKNOWN INTERRUPT"
2765	015320	020116	047111	042524		
2766	015326	052522	052120	000		
2767						
2768	015333	124	052117	046101	MERCT:	.ASCIZ "TOTAL READ CHECK ERRORS = "
2769	015340	051040	040505	020104		
2770	015346	044103	041505	020113		
2771	015354	051105	047522	051522		
2772	015362	036440	000040			
2773						
2774	015366	044506	046114	052502	MFIL:	.ASCIZ "FILLBUFFER "
2775	015374	043106	051105	000040		
2776						
2777	015402	046505	052120	041131	MEMPTY:	.ASCIZ "EMPTYBUFFER "
2778	015410	043125	042506	020122		
2779	015416	000				
2780						
2781	015417	104	044522	042526	MICON:	.ASCIZ "DRIVE(S) "
2782	015424	051450	020051	000040		
2783						
2784	015432	042524	052123	044040	MHUNG:	.ASCIZ "TEST HUNG"<15><12>
2785	015440	047125	006507	000012		
2786						
2787	015446	047516	051516	040524	MNONSTD:	.ASCIZ "NONSTANDARD TRACK / SECTOR LIMITS"
2788	015454	042116	051101	020104		
2789	015462	051124	041501	020113		
2790	015470	020057	042523	052103		
2791	015476	051117	046040	046511		
2792	015504	052111	000123			
2793						
2794	015510	042117	000075		MOD:	.ASCIZ "OD="
2795						
2796	015514	042111	000075		MID:	.ASCIZ "ID="
2797						
2798	015520	044506	051522	036524	MFIRST:	.ASCIZ "FIRST="
2799	015526	000				
2800						



MAINDEC-11-DZRXA-E MACY11 27(732) 20-SEP-76 15:46 PAGE 52  
 DZRXA.E.P11 DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

2801	015527	114	051501	036524	MLAST:	.ASCIZ "LAST="
2802	015534	000				
2803						
2804	015535	111	044516	027124	MINIT1:	.ASCIZ "INIT.DONE NOT SET ERROR" <15><12>
2805	015542	047504	042516	047040		
2806	015550	052117	051440	052105		
2807	015556	042440	051122	051117		
2808	015564	005015	000			
2809						
2810	015567	111	044516	027124	MINIT2:	.ASCIZ "INIT.DONE SET ERROR" <15><12>
2811	015574	047504	042516	051440		
2812	015602	052105	042440	051122		
2813	015610	051117	005015	000		
2814						
2815	015615	104	042040	042440	MDDER:	.ASCIZ "D D ERROR"
2816	015622	051122	051117	000		
2817						
2818	015627	122	041505	053117	MREC:	.ASCIZ "RECOVERABLE "
2819	015634	051105	041101	042514		
2820	015642	000040				
2821						
2822	015644	051103	020103	051105	MBADCR:	.ASCIZ "CRC ERROR NO DATA ERROR"
2823	015652	047522	020122	047516		
2824	015660	042040	052101	020101		
2825	015666	051105	047522	000122		
2826						
2827	015674	042522	042101	000040	MREAD:	.ASCIZ "READ "
2828						
2829	015702	040510	052114	032040	MHALT4:	.ASCIZ "HALT 4" <15><12>
2830	015710	005015	000			
2831						
2832	015713	110	046101	020124	MHALT3:	.ASCIZ "HALT 3" <15><12>
2833	015720	006463	000012			
2834						
2835	015724	040510	052114	030440	MHALT11:	.ASCIZ "HALT 11" <15><12>
2836	015732	006461	000012			
2837						
2838	015736	040504	040524	041440	MCRC:	.ASCIZ "DATA CRC ERROR"
2839	015744	041522	042440	051122		
2840	015752	051117	000			
2841						
2842						
2843	015755	040	047125	042522	MUNREC:	.ASCIZ " UNRECOVERABLE "
2844	015762	047503	042526	040522		
2845	015770	046102	020105	000		
2846						
2847	015775	123	042505	020113	MSEEK:	.ASCIZ "SEEK ERROR"
2848	016002	051105	047522	000122		
2849						
2850	016010	051127	052111	020105	MWRITE:	.ASCIZ "WRITE "
2851	016016	000				
2852						
2853	016017	120	051101	052111	MPAR:	.ASCIZ "PARITY ERROR"
2854	016024	020131	051105	047522		
2855	016032	000122				
2856						



## E06

MAINDEC-11-DZRXA-E MACY11 27(732) 20-SEP-76 15:46 PAGE 53  
 DZRXA.E.P11 DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

2857	016034	051105	047522	020122	MNOFLAG:	.ASCIZ "ERROR FLAG ERROR"
2858	016042	046106	043501	042440		
2859	016050	051122	051117	000		
2860						
2861	016055	122	030530	020061	MDUMP:	.ASCIZ "RX11 DUMP"
2862	016062	052504	050115	000		
2863						
2864	016067	105	051122	051117	MERS:	.ASCIZ "ERRORS"
2865	016074	000123				
2866						
2867	016076	044123	051117	020124	MSHORT:	.ASCIZ "SHORT DUMP"
2868	016104	052504	050115	000		
2869						
2870	016111	122	051505	040524	MRESTART:	.ASCIZ "RESTARTS = "
2871	016116	052122	020123	020075		
2872	016124	000				
2873						
2874	016125	120	051501	020123	MNOPAS:	.ASCIZ "PASS ABORTED"
2875	016132	041101	051117	042524		
2876	016140	000104				
2877						
2878	016142	040520	051523	051505	MPASS:	.ASCIZ "PASSES = "
2879	016150	036440	000040			
2880						
2881	016154	051127	052111	042524	MWRTEEN:	.ASCIZ "WRITTEN "
2882	016162	020116	000			
2883						
2884	016165	057	000040		MSLASH:	.ASCIZ "/"
2885						
2886	016170	000040			SPACE:	.ASCIZ <40>
2887						
2888	016172	051105	047522	051522	MCODES:	.ASCIZ "ERRORS PER ERROR CODES"<15><12>
2889	016200	050040	051105	042440		
2890	016206	051122	051117	041440		
2891	016214	042117	051505	005015		
2892	016222	000				
2893						
2894	016223	124	040522	045503	MTKLOG:	.ASCIZ "TRACK ACCESSED MOVED TO UNIT 0 UNIT 1 ERRORS"<15><12>
2895	016230	020040	041501	042503		
2896	016236	051523	042105	020040		
2897	016244	046440	053117	042105		
2898	016252	052040	020117	020040		
2899	016260	052440	044516	020124		
2900	016266	020060	052440	044516		
2901	016274	020124	020061	051105		
2902	016302	047522	051522	005015		
2903	016310	000				
2904						
2905	016311	050	047516	042516	MNONE:	.ASCIZ "(NONE)"<15><12>
2906	016316	006451	000012			
2907						
2908	016322	047507	042117	000	MGOOD:	.ASCIZ "GOOD"
2909						
2910	016327	040	041440	042510	MSUM:	.ASCIZ " CHECK SUM "
2911	016334	045503	051440	046525		
2912	016342	000040				



2913  
2914  
2915  
2916  
2917  
2918  
2919  
2920  
2921  
2922  
2923  
2924  
2925  
2926  
2927  
2928  
2929  
2930  
2931  
2932  
2933  
2934  
2935  
2936  
2937  
2938  
2939  
2940  
2941  
2942  
2943  
2944  
2945  
2946  
2947  
2948  
2949  
2950  
2951  
2952  
2953  
2954  
2955  
2956  
2957  
2958  
2959  
2960  
2961  
2962  
2963  
2964  
2965  
2966  
2967  
2968

016344 005015 054122 030461  
016352 027440 051040 053130  
016360 030461 000  
016363 015 052012 040522  
016370 045503 046040 046511  
016376 052111 020123 042523  
016404 042514 052103 042105  
016412 047440 052125 047440  
016420 020106 040522 043516  
016426 026105  
016430 051525 047111 020107  
016436 042117 030075 020054  
016444 042111 030475 032061  
016452 005015 000  
016455 015 051412 041505  
016462 047524 020122 044514  
016470 044515 051524 051440  
016476 046105 041505 042524  
016504 020104 052517 020124  
016512 043117 051040 047101  
016520 042507 054  
016523 125 044523 043516  
016530 043040 051111 052123  
016536 030475 020054 040514  
016544 052123 031475 006462  
016552 000012  
016554 005015 040515 047111  
016562 042504 026503 030461  
016570 042055 051132 040530  
016576 042455 005015 000  
016604  
016604 000200

MRX11: .ASCIZ <15><12>"RX11 / RXV11"  
  
OD2BIG: .ASCII <15><12> "TRACK LIMITS SELECTED OUT OF RANGE,"  
  
.ASCIZ "USING OD=0, ID=114"<15><12>  
  
S2BIG: .ASCII <15><12> "SECTOR LIMITS SELECTED OUT OF RANGE,"  
  
.ASCIZ "USING FIRST=1, LAST=32"<15><12>  
  
MREV: .ASCIZ <15><12> "MAINDEC-11-DZRXA-E" <15><12>  
  
.EVEN  
;\*\*\*\*\*  
;THE FOLLOWING LOCATIONS ARE USED FOR DATA STORAGE,RETRY COUNTERS  
;ACCESS COUNTERS ETC.  
BUFADR: .BLKB 200  
;RETRY COUNTERS  
RDRETRY: 10. ;FOR SETUP PURPOSE RDRETRY MUST BE FIRST ON RETRY LIST  
WTRETRY: 10.  
DDRETRY: 10.  
DATARETRY: 10.  
P2RETRY: 10.  
PRETRY: 10.  
CRETRY: 10.  
SRETRY: 10. ;FOR SETUP PURPOSE SRETRY MUST BE LAST ON RETRY LIST  
  
;GENERAL ERROR COUNTERS



;FOR SETUP PURPOSE PARLOG MUST BE FIRST ON COUNTERS LIST

2969 017024 000000  
2970 017026 000000  
2971 017030 000000  
2972 017032 000000  
2973 017034 000000  
2974  
2975 017036 000000  
2976 017040 000000  
2977 017042 000000  
2978 017044 000000  
2979 017046 000000  
2980 017050 000000  
2981 017052 000000  
2982 017054 000000  
2983 017056 000000  
2984 017060 000000  
2985 017062 000000  
2986 017064 000000  
2987 017066 000000  
2988 017070 000000  
2989 017072 000000  
2990 017074 000000  
2991 017076 000000  
2992 017100 000000  
2993 017102 000000  
2994 017104 000000  
2995 017106 000000  
2996 017110 000000  
2997 017112 000000  
2998 017114 000000  
2999 017116 000000  
3000 017120 000000  
3001 017122 000000  
3002 017124 000000  
3003 017126 000000  
3004 017130 000000  
3005

PARLOG: 0  
HPARLOG: 0  
NOERLOG: 0  
UKNINT: 0  
INTER: 0  
ZSEKLOG: 0  
SEKLOG: 0  
ZCRCLOG: 0  
CRCLOG: 0  
ZCRCBAD: 0  
CRCBAD: 0  
ZRDLOG: 0  
RDLOG: 0  
ZWRTLOG: 0  
WRTLOG: 0  
ZDATALOG: 0  
DATALOG: 0  
ZDDMIS: 0  
DDMIS: 0  
ZUNXDD: 0  
UNXDD: 0  
ZHSEKLOG: 0  
HSEKLOG: 0  
ZHRCRCLOG: 0  
HRCRCLOG: 0  
ZHRCRCBAD: 0  
HRCRCBAD: 0  
ZHRDLOG: 0  
HRDLOG: 0  
ZHVRTLOG: 0  
HWRTLOG: 0  
ZHATALOG: 0  
HATALOG: 0  
ZHDDLOG: 0  
HDDLOG: 0

;DRIVE RELATED ERROR COUNTERS

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

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

3006  
3007 017132 000021  
3008  
3009  
3010  
3011  
3012  
3013  
3014  
3015  
3016  
3017  
3018  
3019  
3020 017174 000232  
3021  
3022  
3023 017660 000232  
3024

ERCODE: ;ERROR CODES  
.BLKW 17.

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

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

TKACC: ;TOTAL ACCESS / TRACK  
.BLKW 232

HDMOVE: ;TOTAL HEAD MOVEMENT TO TRACK  
.BLKW 232



```

3025                                     ;THE FOLLOWING 2 BLOCKS OF COUNTERS ARE SINGLE PRECISION
3026
3027                                     ;ERROR PER TRACK ON UNIT 0
3028 020344 000115      UOTRK:          .BLKW 115
3029
3030                                     ;ERROR PER TRACK ON UNIT 1
3031 020576 000115      UITRK:          .BLKW 115
3032
3033                                     ;TOTAL WRITE AND READ FUNCTIONS DOUBLE PRECISION COUNTERS
3034 021030 000000 000000      WTCNTR:      .WORD 0,0
3035 021034 000000 000000      RDCNTR:      .WORD 0,0
3036
3037                                     ;RESTART AND PASSES COMPLETED COUNTERS
3038 021040 000000      RESTCNTR:      0
3039 021042 000000      PASCNTR:      0          ;FOR SETUP PURPOSE PASCNTR MUST BE PLACED LAST
3040
3041 021044 032737 000020 010704  UILOG:      BIT #BIT4,UNITSEL      ;TEST FOR UNIT IN OPERATION
3042 021052 001402          BEQ 1$          ;IF BIT 4 IS 0 RETURN AND INC. UNIT 0 COUNTERS
3043 021054 062703 000002          ADD #2,R3          ;ADJUST THE CONTENTS OF R3 FOR
3044                                     ;THE ADDRESS OF UNIT 1 ERROR LOG COUNTERS.
3045 021060 000207      1$:          RTS PC
3046
3047                                     ;STATISTICAL ERROR REPORT ROUTEEN
3048                                     ;PRINTS ALL ERROR LOGS AND OPERATING CONDITIONS
3049
3050                                     ;*****
3051
3052 021062 012706 001200      ERDUMP:      MOV #STACK,SP
3053 021066 104400 015240          TYPE ,DBLLF
3054 021072 005737 022514          TST SHORTRPT          ;IS SHORT REPORT REQUESTED
3055 021076 001405          BEQ 1$          ;NO,PRINT LONG REPORT
3056 021100 104400 015040          TYPE ,TAB
3057 021104 104400 016076          TYPE ,MSHORT          ;YES,TYPE SHORT REPOR HEADER
3058 021110 000404          BR 2$
3059 021112 104400 015040      1$:      TYPE ,TAB
3060 021116 104400 016055          TYPE ,MDUMP          ;TYPE LONG REPORT HEADER
3061 021122 104400 016554      2$:      TYPE ,MREV          ;PRINT NAME AND REVISION OF PROGRAM
3062 021126 004737 002452          JSR PC,ICOND          ;TYPE OUT SELECTED DRIVES AND TESTS
3063 021132 005737 021040          TST RESTCNTR          ;HAVE THERE BEEN ANY RESTARTS
3064 021136 001412          BEQ 3$          ;NO
3065 021140 104400 016111          TYPE ,MRESTART          ;YES,PRINT OUT HOW MANY
3066 021144 013737 021040 021156      MOV RESTCNTR,12$
3067 021152 004537 014446          JSR R5,SGLDEC
3068 021156 000000      12$:      OPEN
3069 021160 104400 015240          TYPE ,DBLLF
3070 021164 005737 021042      3$:      TST PASCNTR          ;HAVE ANY PASSES BEEN COMPLETED
3071 021170 001005          BNE 4$          ;YES,PRINT OUT HOW MANY
3072 021172 104400 016125          TYPE ,MNOPAS          ;NO,TYPE PASS ABORTED
3073 021176 104400 015240          TYPE ,DBLLF
3074 021202 000412          BR 5$
3075 021204 104400 016142      4$:      TYPE ,MPASS          ;TYPE OUT NUMBER OF PASSES
3076 021210 013737 021042 021222      MOV PASCNTR,7$
3077 021216 004537 014446          JSR R5,SGLDEC
3078 021222 000000      7$:      OPEN
3079 021224 104400 015052          TYPE ,MCRLF
3080 021230 005737 022514      5$:      TST SHORTRPT          ;IS THIS A SHORT REPORT

```



3081	021234	001030			BNE AERRS		; YES, DON'T PRINT WRITE/READ TOTALS
3082							
3083							; PRINTS TOTAL NUMBER OF SECTORS WRITTEN AND/OR
3084							; READ FOR THE DURATION OF RUN TIME.
3085	021236	104400	016154		TYPE ,MWRTE		; PRINT WRT/RD MESSAGE
3086	021242	104400	016165		TYPE ,MSLASH		
3087	021246	104400	015674		TYPE ,MREAD		
3088	021252	104400	015047		TYPE ,DBLSP		
3089	021256	012746	021030		MOV #WTCNTR, -(SP)		; ADDRESS OF TOTAL WRITES ON STACK
3090	021262	004737	014202		JSR PC, @#\$DB2D		; TYPE TOTAL WRITES
3091	021266	004737	014376		JSR PC, RTJUST		
3092	021272	104400	016165		TYPE ,MSLASH		
3093	021276	012746	021034		MOV #RDCNTR, -(SP)		; ADDRESS OF READS ON STACK
3094	021302	004737	014202		JSR PC, @#\$DB2D		; TYPE TOTAL READS
3095	021306	004737	014376		JSR PC, RTJUST		
3096	021312	104400	015240		TYPE ,DBLLF		
3097							
3098							; TYPE OUT MICRO CPU RELATED ERRORS
3099	021316	104400	015052		AERRS: TYPE ,MCRLF		
3100	021322	005737	017024		TST PARLOG		; WERE THERE ANY PARITY ERRORS
3101	021326	001435			BEQ 1\$		
3102	021330	013737	017024	021342	MOV PARLOG, 20\$		; YES, TYPE OUT THE NUMBER
3103	021336	004537	014446		JSR R5, SGLDEC		
3104	021342	000000			20\$: OPEN		
3105	021344	104400	015047		TYPE ,DBLSP		
3106	021350	104400	016017		TYPE ,MPAR		
3107	021354	104400	015052		TYPE ,MCRLF		
3108	021360	005737	017026		TST HPARLOG		; WERE THERE ANY HARD PARITY ERRORS
3109	021364	001416			BEQ 1\$		
3110	021366	013737	017026	021400	MOV HPARLOG, 21\$		; YES, TYPE OUT THE COUNT
3111	021374	004537	014446		JSR R5, SGLDEC		
3112	021400	000000			21\$: OPEN		
3113	021402	104400	015047		TYPE ,DBLSP		
3114	021406	104400	015755		TYPE ,MUNREC		
3115	021412	104400	016017		TYPE ,MPAR		
3116	021416	104400	015052		TYPE ,MCRLF		
3117	021422	005737	017030		1\$: TST NOERLOG		; ANY STATUS ERROR FLAG ERRORS
3118	021426	001414			BEQ 3\$		; IF NONE CHECK NEXT ERROR
3119	021430	013737	017030	021442	MOV NOERLOG, 24\$		; YES, TYPE OUT COUNT
3120	021436	004537	014446		JSR R5, SGLDEC		
3121	021442	000000			24\$: OPEN		
3122	021444	104400	015047		TYPE ,DBLSP		
3123	021450	104400	016034		TYPE ,MNOFLAG		
3124	021454	104400	015052		TYPE ,MCRLF		
3125	021460	005737	017034		3\$: TST INTER		; ANY NO INTERRUPT ON DONE ERRORS
3126	021464	001414			BEQ 4\$		; IF NONE CHECK NEXT ERROR GROUP
3127	021466	013737	017034	021500	MOV INTER, 25\$		; YES, PRINT OUT NUMBER
3128	021474	004537	014446		JSR R5, SGLDEC		
3129	021500	000000			25\$: OPEN		
3130	021502	104400	015047		TYPE ,DBLSP		
3131	021506	104400	015260		TYPE ,MINTER		
3132	021512	104400	015052		TYPE ,MCRLF		
3133							; TYPES DRIVE RELATED ERRORS
3134	021516	104400	015240		4\$: TYPE ,DBLLF		
3135	021522	104400	015106		TYPE ,MUNIT0		; TYPE OUT COLUMN HEADINGS
3136	021526	104400	015116		TYPE ,MUNIT1		



3137	021532	104400	016170	TYPE ,SPACE	
3138	021536	104400	016067	TYPE ,MERS	
3139	021542	104400	015052	TYPE ,MCRLF	
3140	021546	005002		CLR R2	;R2 USED AS AN ERROR PRINTED FLAG
3141	021550	013746	022516	MOV ZERO,-(SP)	;PUT ADDR OF ZERO MARK ON STACK
3142	021554	012746	014462	MOV #MUNXDD,-(SP)	;PUT ADDRESSES OF ALL RECOVERABLE
3143	021560	012746	014506	MOV #MDDMIS,-(SP)	;ERROR MESSAGES ON THE STACK
3144	021564	012746	014527	MOV #MDERHDR,-(SP)	
3145	021570	012746	016010	MOV #MWRITE,-(SP)	
3146	021574	012746	015674	MOV #MREAD,-(SP)	
3147	021600	012746	015644	MOV #MBADCRC,-(SP)	
3148	021604	012746	015736	MOV #MCRC,-(SP)	
3149	021610	012746	015775	MOV #MSEEK,-(SP)	
3150	021614	012701	017036	MOV #ZSEKLOG,R1	;PUT FIRST ERROR CUONTER IN REGISTER
3151	021620	005716		TST (SP)	;GET ADDRESS OF MESSAGE
3152	021622	001436		BEQ 5\$	;IF 0 FINISHED SOFT ERRS DO HARD ERRS
3153	021624	005721		TST (R1)+	;IS DRIVE 0 COUNTER CLEAR
3154	021626	001005		BNE 6\$	;NO,GO PRINT ERROR COUNTER
3155	021630	005711		TST (R1)	;IS DRIVE 1 COUNTER CLEAR
3156	021632	001003		BNE 6\$	;NO,GO PRINT ERROR COUNTER
3157	021634	005721		TST (R1)+	;ADJUST R1 FOR NEXT UNIT 0 COUNTER
3158	021636	005726		TST (SP)+	;ADJUST STACK FOR NEXT ADDRESS
3159	021640	000767		BR 7\$	;GO TEST NEXT PAIR OF COUNTERS
3160	021642	005202		INC R2	;MAKE ERROR PRINTED FLAG NON ZERO
3161	021644	014137	021654	MOV -(R1),30\$	;TYPE CONTENTS OF UNIT 0 COUNTER
3162	021650	004537	014446	JSR R5,SGLDEC	
3163	021654	000000		OPEN	
3164	021656	104400	015047	TYPE ,DBLSP	
3165	021662	005721		TST (R1)+	;ADJUST R1 FOR UNIT 1 COUNTER
3166	021664	012137	021674	MOV (R1)+,31\$	;TYPE COUNTER,GET ADDR OF NEXT IN R1
3167	021670	004537	014446	JSR R5,SGLDEC	
3168	021674	000000		OPEN	
3169	021676	104400	015047	TYPE ,DBLSP	
3170	021702	012637	021710	MOV (SP)+,13\$	
3171	021706	104400		TYPE	;TYPE ERROR MESSAGE FROM STACK
3172	021710	000000		OPEN	
3173	021712	104400	015052	TYPE ,MCRLF	
3174	021716	000740		BR 7\$	;GET NEXT COUNTER
3175	021720	104400	015052	TYPE ,MCRLF	
3176	021724	013746	022516	MOV ZERO,-(SP)	;PUT ADDR OF ZERO MARK ON THE STACK
3177	021730	012746	015615	MOV #MDDER,-(SP)	;PUT ADDRESS OF ALL HARD ERROR
3178	021734	012746	014527	MOV #MDERHDR,-(SP)	;MESSAGES ON THE STACK
3179	021740	012746	016010	MOV #MWRITE,-(SP)	
3180	021744	012746	015674	MOV #MREAD,-(SP)	
3181	021750	012746	015644	MOV #MBADCRC,-(SP)	
3182	021754	012746	015736	MOV #MCRC,-(SP)	
3183	021760	012746	015775	MOV #MSEEK,-(SP)	
3184	021764	012701	017076	MOV #ZHSEKLOG,R1	;PUT FIRST HARD ERROR COUNTER IN R1
3185	021770	005716		TST (SP)	;GET ADDRESS OF MESSAGE FROM STACK
3186	021772	001437		BEQ BERRS	;IF 0 FINISHED ERROR LOGS
3187	021774	005721		TST (R1)+	;IS DRIVE 0 COUNTER CLEAR
3188	021776	001005		BNE 11\$	;NO,PRINT ERROR COUNTER AND MESSAGE
3189	022000	005711		TST (R1)	;IS DRIVE 1 COUNTER CLEAR
3190	022002	001003		BNE 11\$	;NO,GO PRINT THE ERROR
3191	022004	005721		TST (R1)+	;ADJUST R1 FOR NEXT UNIT 0 LOG
3192	022006	005726		TST (SP)+	;ADJUST STACK FOR NEXT ADDRESS



```

3193 022010 000767 BR 12$
3194 022012 014137 022022 11$: MOV -(R1),32$ ;TYPE CONTENTS OF UNIT 0 COUNTER
3195 022016 004537 014446 JSR R5,SGLDEC
3196 022022 000000 32$: OPEN
3197 022024 104400 015047 TYPE ,DBLSP
3198 022030 005721 TST (R1)+ ;ADJUST R1 FOR UNIT 1 COUNTER
3199 022032 012137 022042 MOV (R1)+,33$ ;TYPE UNIT 1 LOG
3200 022036 004537 014446 JSR R5,SGLDEC
3201 022042 000000 33$: OPEN
3202 022044 104400 015047 TYPE ,DBLSP
3203 022050 104400 015755 TYPE ,MUNREC ;TYPE UNRECOVERABLE
3204 022054 012637 022062 MOV (SP)+,14$
3205 022060 104400 TYPE ;ERROR MESSAGE FROM STACK
3206 022062 000000 14$: OPEN
3207 022064 104400 015052 TYPE ,MCRLF
3208 022070 000737 BR 12$ ;GET NEXT LOG
3209 ;PRINTS ERRORS PER ERROR CODES
3210 022072 005702 BERRS: TST R2 ;WAS AN ERROR PRINTED IN THE LAST GROUP
3211 022074 001004 BNE 5$ ;IF R2 IS NONZERO BRANCH
3212 022076 104400 015040 TYPE ,TAB
3213 022102 104400 016311 TYPE ,MNONE ;TYPE NONE
3214 022106 005002 5$: CLR R2 ;CLEAR ERROR PRINTED FLAG AGAIN
3215 022110 104400 015240 TYPE ,DBLLF
3216 022114 104400 016172 TYPE ,MCODES ;TYPE ERROR PER ERROR CODE
3217 022120 012700 000001 MOV #1,R0 ;SET UP CODE COUNTER
3218 022124 012701 017132 MOV #ERCODE,R1 ;GET ADDR OF FIRST ERROR CODE
3219 022130 020027 000022 2$: CMP R0,#22 ;IS THE LAST CODE PRINTED
3220 022134 001001 BNE 3$ ;NO TEST THE NEXT ONE
3221 022136 000427 BR TYPTRK ;YES,GO PRINT TRACK INFOMATION
3222 022140 005721 3$: TST (R1)+ ;TEST FOR 0 WORD,GET NEXT ADDRESS
3223 022142 001002 BNE 1$ ;IF NOT 0 TYPE COUNT AND CODE NUMBER
3224 022144 005200 INC R0 ;INC CODE COUNTER
3225 022146 000770 BR 2$
3226 022150 014137 022160 1$: MOV -(R1),4$ ;PUT CONTENTS OF COUNTER IN OPEN
3227 022154 004537 014446 JSR R5,SGLDEC ;FOR TYPE OUT
3228 022160 000000 4$: OPEN
3229 022162 104400 015040 TYPE ,TAB
3230
3231 022166 010002 MOV R0,R2 ;PUT CODE NUMBER IN REGISTER TO
3232 022170 006302 ASL R2 ;TYPE IT OUT IN THE FROM OF ERROR CODE
3233 022172 006302 ASL R2
3234 022174 006302 ASL R2
3235 022176 010246 MOV R2,-(SP) ;;SAVE R2 FOR TYPEOUT
3236 022200 104402 TYPOS ;;GO TYPE--OCTAL ASCII
3237 022202 004 .BYTE 4 ;;TYPE 4 DIGIT(S)
3238 022203 001 .BYTE 1 ;;TYPE LEADING ZEROS
3239 022204 104400 015052 TYPE ,MCRLF
3240 022210 005200 INC R0 ;SET UP FOR NEXT COUNTER
3241 022212 005721 TST (R1)+ ;GET NEXT ERROR COUNTER
3242 022214 000745 BR 2$
3243 ;PRINTS TRACK ACCESS,HEAD MOVEMENT,AND ERRORS PER TRACK
3244 022216 005702 TYPTRK: TST R2 ;WAS AN ERROR PRINTED IN LAST GROUP
3245 022220 001004 BNE 5$ ;IF PRINTED ERROR FLAG NONZERO BRANCH
3246 022222 104400 015040 TYPE ,TAB
3247 022226 104400 016311 TYPE ,MNONE ;TYPE NONE
3248 022232 104400 015240 5$: TYPE ,DBLLF

```



3249	022236	005737	022514		TST SHORTRPT	; IF SHORT REPORT DON'T TYPE TRACK ACCESS OR ERRO
3250	022242	001121			BNE 6\$	
3251	022244	104400	016223		TYPE ,MTKLOG	; TYPE FORMAT OF ERROR PRINT OUT
3252	022250	005037	022330		CLR 2\$	; TRACK ADDRESS COUNTER
3253	022254	012704	017170		MOV #TKACC-4,R4	; SET UP ADDRESS OF TRACK ACCESSED
3254	022260	012703	017654		MOV #HDMOVE-4,R3	; ADDRESS OF HEAD MOVEMENT COUNTER
3255	022264	012702	020344		MOV #UOTRK,R2	; UNIT 0 ERROR PER TRACK COUNTER
3256	022270	012701	020576		MOV #UITRK,R1	; UNIT 1 ERROR PER TRACK COUNTER
3257	022274	062704	000004	1\$:	ADD #4,R4	; ADJUST FOR ADDRESS OF NEXT COUNTER
3258	022300	005714			TST (R4)	; HAS THIS TRACK BEEN ACCESSED
3259	022302	001010			BNE 10\$	; YES, PRINT OUT THE COUNTERS
3260	022304	005764	000002		TST 2(R4)	; TEST UPPER WORD OF ACCESS COUNTER
3261	022310	001005			BNE 10\$	
3262	022312	062703	000004		ADD #4,R3	; THIS TRACK HAS NOT BEEN USED. ADJUST
3263	022316	005722			TST(R2)+	; REGISTERS TO POINT TO NEXT TRACK COUNTERS
3264	022320	005721			TST (R1)+	
3265	022322	000463			BR 12\$	; TEST FOR LAST TRACK
3266	022324	004537	014446	10\$:	JSR R5, SGLDEC	; TYPE TRACK NUMBER
3267	022330	000000		2\$:	OPEN	; THIS LOCATION USED AS TRACK ADDR COUNTER
3268	022332	104400	016170		TYPE ,SPACE	
3269	022336	010446			MOV R4, -(SP)	; TYPE TRACK ACCESSED COUNT
3270	022340	004737	014202		JSR PC, @#\$DB2D	
3271	022344	004737	014376		JSR PC, RTJUST	
3272	022350	104400	016170		TYPE ,SPACE	
3273	022354	062703	000004		ADD #4,R3	
3274	022360	010346			MOV R3, -(SP)	; TYPE HEAD MOVED TO COUNT
3275	022362	004737	014202		JSR PC, @#\$DB2D	
3276	022366	004737	014376		JSR PC, RTJUST	
3277	022372	104400	015047		TYPE ,DBLSP	
3278	022376	104400	016170		TYPE ,SPACE	
3279	022402	005712			TST (R2)	; ARE THERE ANY UNIT 0 ERRORS
3280	022404	001421			BEQ 20\$	; NO, CHECK UNIT 1
3281	022406	012237	022416		MOV (R2)+, 3\$	; TYPE UNIT 0 ERRORS
3282	022412	004537	014446		JSR R5, SGLDEC	
3283	022416	000000		3\$:	OPEN	
3284	022420	005711			TST (R1)	; ARE THERE ANY UNIT 1 ERRORS
3285	022422	001420			BEQ 7\$	; NO, GO ADJUST POINTERS
3286	022424	104400	016170		TYPE ,SPACE	
3287	022430	104400	015047	22\$:	TYPE ,DBLSP	
3288	022434	012137	022444		MOV (R1)+, 4\$	; TYPE UNIT 1 ERRORS
3289	022440	004537	014446		JSR R5, SGLDEC	
3290	022444	000000		4\$:	OPEN	
3291	022446	000407			BR 11\$	; GO TO NEXT TRACK
3292						
3293	022450	005722		20\$:	TST (R2)+	; NO ERRORS, ADJUST REG. FOR NEXT COUNTER
3294	022452	005711			TST (R1)	; ARE THERE ANY UNIT 1 ERRORS
3295	022454	001403			BEQ 7\$	; NO, GO TO NEXT TRACK
3296	022456	104400	015040		TYPE ,TAB	; YES, TYPE TAB FOR PLACEMENT
3297	022462	000762			BR 22\$	; GO TYPE THE NUMBER
3298	022464	005721		7\$:	TST (R1)+	; NO ERRORS, ADJUST REG. FOR NEXT COUNTER
3299	022466	104400	015052	11\$:	TYPE ,MCRLF	
3300	022472	005237	022330	12\$:	INC 2\$	
3301	022476	023727	022330	000115	CMP 2\$, #115	; HAVE ALL TRACK ERRORS BEEN TYPED
3302	022504	001273			BNE 1\$	; NO, TYPE THE NEXT ONE
3303	022506	005037	022514	6\$:	CLR SHORTRPT	; SET UP FOR LONG REPORT
3304	022512	000000		HLT17:	HALT	; YES DONE



M06

MAINDEC-11-DZRXA-E MACY11 27(732) 20-SEP-76 15:46 PAGE 61  
DZRXA.E.P11 DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

3305				
3306	022514	000000	SHORTRPT:	0
3307	022516	000000	ZERO:	0
3308				
3309		000001	.END	































	2296*	2297*	2304*	2305*	2306*	2307*	2308	2309	2310	2311	2312	2313	2331*	
	2332*	2333	2340*	2343*	2344*	2348*	2349*	2353	2356*	2360	2362	2364	2365*	
	2372	2374	2376*	2377	2379*	2380*	2381*	2382*	2383*	2398*	2399*	2402*	2403*	
	2404	2408*	2409*	2410	2413	2415	2417*	2426*	2431	2442	2443*	2444*	2445*	
	2466*	2467	2500*	2501*	2502*	2503*	2504*	2505*	2506*	2507	2515*	2519	2520	
	2521	2522	2523	2524	2525	2530*	2550	2551*	2553*	2554*	2571	2573*	2627*	
	2628	2639	2640*	2645*	3052*	3089*	3093*	3141*	3142*	3143*	3144*	3145*	3146*	
	3147*	3148*	3149*	3151	3158	3170	3176*	3177*	3178*	3179*	3180*	3181*	3182*	
	3183*	3185	3192	3204	3235*	3269*	3274*							
	2886#	3137	3268	3272	3278	3286								
SPACE	016170													
SRETRY	017022	470	748*	855*	936*	2965#								
STACK =	001200	46#	437	700	3052									
START1	001412	468#												
STATER	002046	513	554#	663	1491	1530	1606							
STKLMT=	177774	57#												
STOP	010732	743	923	1143	1302	1319	1342	1376	1808#					
STYP1	003626	836*	847#	990*										
STYP2	003720	837*	864#	991*										
SUM	010612	1179	1652*	1754*	1762#									
SWHLT1	003636	846	849#											
SWR	001214	345#	436*	443	447*	452	655	772	778	781	845	850	853	1006
		1011	1014	1032	1037	1040	1061	1088	1095	1098	1121	1200	1202	1233
		1245	1263	1445	1450	1471	1480	1504	1823	2327	2364*	2506	2519*	
SWREG	000176	191#	447	452	2327	2340								
SW0	= 000001	110#												
SW00	= 000001	100#	110											
SW01	= 000002	99#	109											
SW02	= 000004	98#	108											
SW03	= 000010	97#	107											
SW04	= 000020	96#	106											
SW05	= 000040	95#	105											
SW06	= 000100	94#	104											
SW07	= 000200	93#	103											
SW08	= 000400	92#	102											
SW09	= 001000	91#	101											
SW1	= 000002	109#												
SW10	= 002000	90#												
SW11	= 004000	89#	781	853	1014	1040	1061	1098	1263					
SW12	= 010000	88#	1202											
SW13	= 020000	87#	772	845	1006	1032	1088	1121	1200	1245	1445	1471	1490	1504
SW14	= 040000	86#	1823											
SW15	= 100000	85#												
SW2	= 000004	108#												
SW3	= 000010	107#												
SW4	= 000020	106#												
SW5	= 000040	105#												
SW6	= 000100	104#												
SW7	= 000200	103#												
SW8	= 000400	102#	655											
SW9	= 001000	101#												
S2BIG	016455	624	2930#											
TAB	015040	643	1615	2706#	3056	3059	3212	3229	3246	3296				
TARGET	011360	877	886	897	1177	1180	1546	1860*	1892	1913#	1927*	1930*	1941*	1943*
		1971*	1973*	1981	1999*	2004*	2009*	2012*	2015*	2030*	2033*	2057*		
TBITVE=	000014	143#												
TEST	002140	542*	543*	544*	545*	546*	572#	712	714*	715	1567			











SMNEW	013703	2342	2454#																	
SMSWR	013672	2339	2452#																	
SNULL	012602	2155	2186#																	
SOCNT	013034	2225*	2254*	2267#																
SOMODE	013036	2220*	2224*	2229	2232*	2243*	2269#													
SPOWER	014136	1541	2529	2536#																
SPWRAD	014124	2531#																		
SPWRDN	013764	176	2498#	2526																
SPWRNG	014120	2529#																		
SPWRUP	014036	2508	2514#																	
SQUES	012606	2190#	2385	2434	2450															
SRDCHR	013422	2398#	2490																	
SRDDEC=	***** U	2492																		
SRDLIN	013542	2426#	2491																	
SRDOCT=	***** U	2492																		
SRDSZ =	000010	2419#																		
SRESRE	013076	2303#	2493																	
SR2A =	***** U	2494																		
SSAVRE	013040	2287#	2492																	
SSAVR6	014134	2507*	2515	2516*	2517*	2535#														
SSB2D	014146	2550#	2646																	
SSETUP=	000114	208#	449	2322	2456															
SSTUP =	177777	208#																		
SSWR =	160000	20	21#	2532																
STKB	013136	2319#	2331	2348	2402	2408														
STKS	013134	2318#	2329	2345	2369*	2400	2406													
STN =	000001	20#																		
STNPWR	014312	2577	2578	2598#																
STPB	012600	2173*	2185#																	
STPFLG	012605	2131	2189#																	
STPS	012576	2171	2184#																	
STRAP	013716	180	2466#																	
STRP =	000012	2474#	2483#	2484#	2485#	2486#	2487	2488#	2489	2490#	2491#	2492#	2493#	2494#						
STRPAD	013740	2471	2481#																	
STTYIN	013650	2427	2428	2445	2449#															
STYPBN=	***** U	2486																		
STYPDS=	***** U	2486																		
STYPE	012356	2131#	2474	2482																
STYPEC	012526	2152	2159	2166	2171#	2172	2371													
STYPEX	012574	2177	2179	2182#																
STYPOC	012636	2223#	2483																	
STYPON	012652	2222	2225#	2485																
STYPOS	012612	2218#	2484																	
SOFILL	013035	2219#	2223*	2233	2268#															
.	= 022520	172#	174	175#	179#	183#	186#	189#	209#	243#	719	1661	1903	2184						
		2185	2186	2187	2188	2189	2190	2191	2192	2193	2318	2319	2449#	2450						
		2456	2457	2458	2510	2534	2618#	2948#	2955#	3007#	3020#	3023#	3028#	3031#						







.\$RAND	1#		
.\$RDDE	1#		
.\$RDOC	1#		
.\$READ	1#	5#	2315
.\$R2AZ	1#		
.\$SAVE	1#	6#	2270
.\$SB20	1#	6#	2539
.\$SB20	1#		
.\$SCOP	1#		
.\$SIZE	1#		
.\$SUPR	1#		
.\$STRAP	1#	6#	2458
.\$STYPB	1#		
.\$STYPD	1#		
.\$STYPE	1#	5#	2114
.\$STYPO	1#	5#	2193
.\$4OCA	1#		
.\$1170	1#		



ADC	808	891	943	1431	1986	2587									
ADD	807	866	879	882	889	890	942	1058	1066	1171	1180	1181	1430	1476	1499
	1537	1754	1804	1984	1985	2053	2061	2062	2072	2105	2109	2141	2221	2231	2356
ASL	2365	2553	2586	2588	3043	3257	3262	3273							
	717	878	887	888	1658	1899	1955	1982	1983	2027	2379	2380	2381	2470	3232
ASLB	3233	3234													
BCC	1183														
BEQ	1702														
	444	500	512	516	532	561	576	579	596	599	601	662	785	805	823
	856	891	951	961	980	1017	1043	1064	1080	1101	1114	1203	1207	1240	1243
	1253	1266	1277	1381	1400	1408	1427	1463	1490	1495	1529	1605	1707	1757	1817
	1824	1840	1970	2056	2144	2179	2248	2336	2363	2378	2631	3042	3055	3064	3101
	3109	3118	3126	3152	3186	3280	3285	3295							
BGE	670	1959	2052	2107											
BGT	1205	2255	2375	2416											
BHI	578	581	598	603											
BIC	533	541	543	551	1218	1382	1551	1587	1594	1783	1790	1833	1855	1924	1940
	1953	1968	2025	2044	2046	2063	2073	2245	2332	2349	2376	2403	2409	2417	
BICB	711	1651	1893	1998	2003										
BIS	498	520	1091	1118	1152	1159	1197	1384	1485	1784	1791	1868	2250	2251	2383
	2590														
BISB	718	892	1659	1900											
BIT	488	499	504	515	655	661	772	781	804	822	845	853	880	979	986
	1006	1014	1032	1040	1061	1079	1088	1098	1113	1121	1200	1202	1245	1263	1377
	1380	1426	1445	1462	1465	1471	1480	1489	1504	1521	1528	1555	1604	1779	1788
	1802	1816	1823	3041											
BITB	511	1081	1405	1467	1582	2006	2167								
BLE	2048														
BLOS	2429														
BLT	2158	2256	2373	2414	2583										
BMI	1461	1533	1596												
BNE	451	453	467	471	489	505	530	594	618	629	647	656	672	713	742
	773	782	814	816	846	854	922	975	987	1001	1007	1015	1033	1041	1050
	1062	1082	1089	1099	1122	1142	1155	1173	1201	1246	1264	1301	1318	1341	1368
	1375	1378	1404	1416	1433	1446	1466	1468	1472	1481	1505	1522	1556	1583	1654
	1780	1789	1803	1812	1835	1838	1857	1895	1926	2007	2014	2029	2087	2138	2146
	2154	2168	2175	2246	2328	2334	2354	2361	2368	2405	2411	2433	2439	2518	2594
	2633	3071	3081	3154	3156	3188	3190	3211	3220	3223	3245	3250	3259	3261	3302
BPL	492	497	502	519	525	638	641	666	779	851	972	997	1012	1038	1096
	1209	1212	1234	1397	1451	1589	1782	1787	1801	1939	1952	1996	2002	2024	2132
	2172	2244	2330	2346	2401	2407									
BR	210	211	212	445	455	507	514	534	592	616	623	626	664	719	744
	761	924	993	1075	1124	1144	1175	1210	1255	1303	1320	1343	1383	1385	1492
	1525	1527	1531	1558	1607	1661	1682	1691	1710	1718	1729	1742	1752	1819	1903
	1928	1931	1942	1944	1960	1972	1974	2000	2005	2010	2016	2032	2034	2050	2054
	2058	2134	2151	2161	2170	2177	2222	2237	2258	2357	2384	2386	2412	2435	2510
	2534	2585	2635	3058	3074	3159	3174	3193	3208	3221	3225	3242	3265	3291	3297
CLC	539	1705	2064												
CLR	465	495	574	622	625	706	709	710	753	898	938	1157	1158	1161	1164
	1280	1369	1392	1409	1412	1424	1652	1680	1717	1809	1832	1861	1863	2068	2235
	2343	2344	2516	2579	3140	3214	3252	3303							
CLRB	506	1859	2093	2150	2176	2440	2595								
CMP	443	446	452	466	470	950	960	1204	1399	1756	1834	2327	2333	2353	2360
	2372	2374	2404	2410	2413	2415	2428	2589	3219	3301					
CMPB	529	577	580	597	602	617	668	1172	1925	1969	2013	2028	2047	2051	2055
	2106	2143	2145	2153	2174	2178	2335	2367	2432	2438	2632				



COM	914	1065	1134	1330	1361	1597									
COMB	1728														
DEC	560	671	716	741	783	815	855	921	974	1016	1042	1063	1100	1141	1265
DECB	1276	1300	1317	1340	1367	1374	1657	1898	1943	1954	2011	2026	2031	2592	
EMT	2157	2160	2243	2254											
HALT	47														
INC	174	538	570	649	674	780	852	1013	1026	1039	1097	1235	1418	1435	1452
INCB	1831	2133	2509	2533	3304										
IOB	554	565	632	704	760	770	787	842	861	883	954	964	998	1004	1021
JMP	1030	1047	1053	1070	1085	1105	1117	1174	1198	1228	1236	1260	1271	1401	1415
JSR	1443	1469	1478	1502	1703	1810	1815	1837	1867	1930	1997	2008	2088	2092	2110
MOV	2249	2257	2382	2517	2584	3160	3224	3240	3300						
MOV8	2180														
NEG	48														
NEGB	213	214	215	552	571	650	675	720	721	722	723	724	725	726	767
	786	793	811	818	819	828	858	946	985	1018	1044	1059	1067	1102	1112
	1156	1195	1237	1268	1279	1464	1542	1590	1598	1662	1663	1664	1665	1666	1667
	1668	1904	1905	1906	1907	1908	1909	1910							
	487	493	503	510	513	523	582	585	590	604	608	613	644	651	660
	663	667	705	736	737	738	739	740	743	746	747	750	758	809	810
	838	841	843	848	857	860	865	872	894	896	915	916	917	918	919
	920	923	931	932	944	945	953	963	970	992	995	999	1003	1020	1029
	1046	1052	1069	1084	1086	1092	1104	1110	1116	1119	1135	1136	1137	1138	1139
	1140	1143	1153	1160	1168	1216	1220	1249	1259	1261	1267	1270	1278	1295	1296
	1297	1298	1299	1302	1312	1313	1314	1315	1316	1319	1331	1332	1333	1334	1335
	1336	1337	1338	1339	1342	1358	1362	1363	1364	1365	1366	1371	1372	1373	1376
	1453	1488	1491	1510	1518	1520	1524	1530	1536	1547	1552	1603	1606	1681	1704
	1727	1749	1751	1808	1828	1836	1842	2045	2152	2159	2166	2371	2552	2646	2647
	3062	3067	3077	3090	3091	3094	3095	3103	3111	3120	3128	3162	3167	3195	3200
	3227	3266	3270	3271	3275	3276	3282	3289							
	436	437	438	439	442	447	448	458	459	460	461	462	463	464	468
	469	473	478	483	494	509	540	542	547	556	657	658	659	676	700
	701	702	707	708	714	715	748	749	751	752	754	755	757	764	765
	766	800	801	802	825	826	827	831	836	837	840	859	871	877	886
	893	899	933	934	935	936	937	939	940	952	955	962	965	982	983
	984	989	990	991	1002	1019	1028	1045	1048	1051	1068	1071	1083	1103	1115
	1162	1163	1165	1167	1179	1186	1192	1193	1194	1219	1223	1229	1248	1258	1269
	1359	1360	1370	1379	1393	1394	1413	1422	1425	1428	1486	1487	1496	1509	1514
	1523	1526	1546	1550	1561	1567	1573	1602	1611	1617	1655	1656	1660	1701	1814
	1827	1839	1865	1896	1897	1956	1981	2057	2060	2067	2069	2074	2075	2097	2135
	2136	2140	2155	2218	2226	2227	2228	2234	2241	2259	2260	2261	2262	2263	2288
	2289	2290	2291	2292	2293	2294	2295	2296	2297	2304	2305	2306	2307	2308	2309
	2310	2311	2312	2313	2340	2364	2369	2398	2399	2426	2427	2442	2443	2444	2445
	2466	2467	2471	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2514
	2515	2519	2520	2521	2522	2523	2524	2525	2526	2527	2530	2550	2551	2554	2571
	2572	2573	2574	2575	2576	2577	2578	2627	2628	2629	2639	2640	2645	3052	3066
	3076	3089	3093	3102	3110	3119	3127	3141	3142	3143	3144	3145	3146	3147	3148
	3149	3150	3161	3166	3170	3176	3177	3178	3179	3180	3181	3182	3183	3184	3194
	3199	3204	3217	3218	3226	3231	3235	3253	3254	3255	3256	3269	3274	3281	3288
	456	584	589	607	612	759	803	806	895	897	941	1170	1177	1178	1182
	1185	1227	1459	1493	1601	1608	1690	1699	1726	1741	1750	1755	1858	1860	1862
	1864	1892	1927	1941	1971	1973	1999	2004	2009	2012	2015	2030	2033	2089	2090
	2094	2108	2137	2165	2173	2219	2220	2223	2224	2225	2229	2232	2233	2252	2331
	2348	2402	2408	2431	2436	2469	2591	2634	2636						



NOP	1843	1844	1845												
RESET	435	1841													
ROL	548	549	2065	2066	2236	2238	2239	2240	2242						
ROLB	1709														
ROR	544	545	546	1497	1498	2070	2071								
RTI	440	703	756	900	1166	1395	1414	1475	2142	2264	2298	2314	2370	2418	2446
	2532														
RTS	521	564	619	633	652	677	817	867	875	884	901	973	976	1125	1187
	1241	1281	1398	1429	1477	1538	1577	1622	1758	1760	1785	1792	1805	1846	1869
	1987	2076	2098	2111	2182	2472	2555	2597	2641	2648	3045				
SBC	2581														
SEC	1700	1708													
SUB	1866	1957	1958	2049	2091	2095	2580	2582							
SWAB	550														
TRAP	2474	2483	2484	2485	2487	2489	2490	2491	2492	2493					
TST	450	491	501	531	575	595	628	640	646	665	712	763	778	784	813
	821	850	971	978	996	1000	1011	1037	1049	1095	1154	1191	1206	1233	1239
	1242	1411	1432	1450	1460	1494	1588	1653	1706	1759	1786	1800	1811	1856	1894
	2086	2139	2147	2169	2247	2362	2377	2468	3054	3063	3070	3080	3100	3108	3117
	3125	3151	3153	3155	3157	3158	3165	3185	3187	3189	3191	3192	3198	3108	3117
	3241	3244	3249	3258	3260	3263	3264	3279	3284	3293	3294	3298	3198	3210	3222
TSTB	496	518	524	593	600	637	1208	1211	1252	1396	1532	1595	1781	1938	1951
	1995	2001	2023	2131	2171	2329	2345	2400	2406	2630					
.ASCII	2190	2191	2682	2689	2918	2930									
.ASCIZ	2192	2450	2451	2452	2454	2536	2651	2656	2661	2666	2670	2674	2677	2697	2702
	2706	2709	2711	2713	2715	2717	2719	2724	2727	2730	2734	2738	2741	2744	2747
	2750	2752	2754	2756	2758	2764	2768	2774	2777	2781	2784	2787	2794	2796	2798
	2801	2804	2810	2815	2818	2822	2827	2829	2832	2835	2838	2843	2847	2850	2853
	2857	2861	2864	2867	2870	2874	2878	2881	2884	2886	2888	2894	2905	2908	2910
	2914	2925	2937	2943											
.BLKB	2449	2618	2955												
.BLKW	3007	3020	3023	3028	3031										
.BYTE	475	476	480	481	485	486	558	559	1516	1517	1563	1564	1569	1570	1575
	1576	1613	1614	1619	1620	2186	2187	2188	2189	2265	2266	2267	2268	2447	2448
	2456	2457	3237	3238											
.DSABL	2387														
.ENABL	1	4	2320												
.END	3309														
.ENDC	15	47	139	153	195	208	219	251	271	289	349	414	432	452	458
	476	477	481	482	486	487	559	560	680	699	731	799	909	927	1130
	1148	1289	1307	1325	1347	1389	1456	1517	1518	1564	1565	1570	1571	1576	1577
	1614	1615	1620	1621	1626	1648	1676	1685	1694	1713	1721	1736	1745	1765	1778
	1851	1874	1891	1920	1934	1947	1964	1977	1991	2019	2039	2083	2117	2137	2196
	2273	2318	2321	2323	2351	2387	2391	2419	2420	2427	2429	2432	2434	2450	2458
	2461	2467	2470	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493
	2494	2497	2506	2507	2513	2519	2520	2530	2532	2539	2542	2560	2621	2951	3051
	3238	3239													
.EQUIV	47	48	56	71	72	101	102	103	104	105	106	107	108	109	110
	129	130	131	132	133	134	135	136	137	138					
.EVEN	2538	2948													
.IF	11	45	111	139	194	208	218	250	270	288	348	413	431	449	452
	475	476	480	481	485	486	558	559	679	698	730	798	908	926	1129
	1147	1288	1306	1324	1346	1388	1455	1516	1517	1563	1564	1569	1570	1575	1576
	1613	1614	1619	1620	1625	1647	1675	1684	1693	1712	1720	1735	1744	1764	1777
	1850	1873	1890	1919	1933	1946	1963	1976	1990	2018	2038	2082	2116	2137	2195
	2272	2317	2321	2322	2323	2351	2390	2391	2419	2427	2428	2432	2433	2449	2450



	2456	2460	2466	2470	2474	2483	2484	2485	2486	2487	2489	2490	2491	2492	2493
	2494	2496	2506	2507	2512	2519	2520	2528	2530	2532	2536	2541	2559	2620	2950
.IFF	3050	3237	3238												
	47	195	219	251	271	289	349	414	432	475	476	480	481	486	559
	620	699	731	799	909	927	1130	1148	1289	1307	1325	1347	1389	1456	1517
	1564	1570	1576	1614	1620	1626	1648	1676	1685	1694	1713	1721	1736	1745	1765
	1778	1851	1874	1891	1920	1934	1947	1964	1977	1991	2019	2039	2083	2117	2196
	2273	2318	2323	2391	2393	2398	2419	2420	2429	2433	2450	2461	2467	2497	2513
	2530	2542	2560	2621	2951	3051	3238	3239							
.IFT	2393	2398													
.IFTF	2338	2391	2394												
.IIF	10	15	20	21	474	479	484	557	1515	1562	1568	1574	1612	1618	2184
	2185	2186	2187	2188	2189	2190	2191	2192	2193	2318	2319	2341	2442	2450	2456
	2457	2458	2482	2483	2484	2485	2487	2489	2490	2491	2492	2493	3236		
.IRP	208	2288	2308	2500	2506	2519	2520								
.LIST	1	3	153	174	208	2419	2474	2482	2483	2484	2485	2486	2487	2488	2489
	2490	2491	2492	2493	2494										
.MACRO	1	2474													
.MCALL	5	6	153												
.NLIST	1	2	153	174	208	2419	2474	2482	2483	2484	2485	2486	2487	2488	2489
	2490	2491	2492	2493	2494										
.REPT	174														
.SBTTL	43	449	2114	2193	2270	2315	2458	2474	2494	2539	2557				
.TITLE	10														
.WORD	173	187	344	345	346	1225	2181	2184	2185	2269	2318	2319	2529	2531	2556
	2593	3034	3035												

ERRORS DETECTED: 0  
 DEFAULT GLOBALS GENERATED: 0

\*.DZRXA.E.SEQ/SOL/CRF=SYSMAC.SML,DZRXA.E.P11  
 RUN-TIME: 42 44 6 SECONDS  
 RUN-TIME RATIO: 375/93=4.0  
 CORE USED: 34K (67 PAGES)



