

RK11/RK05

DYNAMIC TEST
MD-11-DZRKL-E

EP-DZRKL-E-DL-A
COPYRIGHT © 75-77
FICHE 1 OF 1

AUG 1977
digital
MADE IN USA

The microfiche card contains 100 frames of data. The first 99 frames are organized into a 10x10 grid. Each frame contains a different set of data, including text, tables, and graphs. The 100th frame is blank.

Frame	Content
1	Table with 2 columns: Name, Address
2	Table with 2 columns: Name, Address
3	Table with 2 columns: Name, Address
4	Table with 2 columns: Name, Address
5	Table with 2 columns: Name, Address
6	Table with 2 columns: Name, Address
7	Table with 2 columns: Name, Address
8	Table with 2 columns: Name, Address
9	Table with 2 columns: Name, Address
10	Table with 2 columns: Name, Address
11	Table with 2 columns: Name, Address
12	Table with 2 columns: Name, Address
13	Table with 2 columns: Name, Address
14	Table with 2 columns: Name, Address
15	Table with 2 columns: Name, Address
16	Table with 2 columns: Name, Address
17	Table with 2 columns: Name, Address
18	Table with 2 columns: Name, Address
19	Table with 2 columns: Name, Address
20	Table with 2 columns: Name, Address
21	Table with 2 columns: Name, Address
22	Table with 2 columns: Name, Address
23	Table with 2 columns: Name, Address
24	Table with 2 columns: Name, Address
25	Table with 2 columns: Name, Address
26	Table with 2 columns: Name, Address
27	Table with 2 columns: Name, Address
28	Table with 2 columns: Name, Address
29	Table with 2 columns: Name, Address
30	Table with 2 columns: Name, Address
31	Table with 2 columns: Name, Address
32	Table with 2 columns: Name, Address
33	Table with 2 columns: Name, Address
34	Table with 2 columns: Name, Address
35	Table with 2 columns: Name, Address
36	Table with 2 columns: Name, Address
37	Table with 2 columns: Name, Address
38	Table with 2 columns: Name, Address
39	Table with 2 columns: Name, Address
40	Table with 2 columns: Name, Address
41	Table with 2 columns: Name, Address
42	Table with 2 columns: Name, Address
43	Table with 2 columns: Name, Address
44	Table with 2 columns: Name, Address
45	Table with 2 columns: Name, Address
46	Table with 2 columns: Name, Address
47	Table with 2 columns: Name, Address
48	Table with 2 columns: Name, Address
49	Table with 2 columns: Name, Address
50	Table with 2 columns: Name, Address
51	Table with 2 columns: Name, Address
52	Table with 2 columns: Name, Address
53	Table with 2 columns: Name, Address
54	Table with 2 columns: Name, Address
55	Table with 2 columns: Name, Address
56	Table with 2 columns: Name, Address
57	Table with 2 columns: Name, Address
58	Table with 2 columns: Name, Address
59	Table with 2 columns: Name, Address
60	Table with 2 columns: Name, Address
61	Table with 2 columns: Name, Address
62	Table with 2 columns: Name, Address
63	Table with 2 columns: Name, Address
64	Table with 2 columns: Name, Address
65	Table with 2 columns: Name, Address
66	Table with 2 columns: Name, Address
67	Table with 2 columns: Name, Address
68	Table with 2 columns: Name, Address
69	Table with 2 columns: Name, Address
70	Table with 2 columns: Name, Address
71	Table with 2 columns: Name, Address
72	Table with 2 columns: Name, Address
73	Table with 2 columns: Name, Address
74	Table with 2 columns: Name, Address
75	Table with 2 columns: Name, Address
76	Table with 2 columns: Name, Address
77	Table with 2 columns: Name, Address
78	Table with 2 columns: Name, Address
79	Table with 2 columns: Name, Address
80	Table with 2 columns: Name, Address
81	Table with 2 columns: Name, Address
82	Table with 2 columns: Name, Address
83	Table with 2 columns: Name, Address
84	Table with 2 columns: Name, Address
85	Table with 2 columns: Name, Address
86	Table with 2 columns: Name, Address
87	Table with 2 columns: Name, Address
88	Table with 2 columns: Name, Address
89	Table with 2 columns: Name, Address
90	Table with 2 columns: Name, Address
91	Table with 2 columns: Name, Address
92	Table with 2 columns: Name, Address
93	Table with 2 columns: Name, Address
94	Table with 2 columns: Name, Address
95	Table with 2 columns: Name, Address
96	Table with 2 columns: Name, Address
97	Table with 2 columns: Name, Address
98	Table with 2 columns: Name, Address
99	Table with 2 columns: Name, Address
100	Blank frame

B01

EOFINIRI: SEQ

00010000

770715

PDP10 411

-AMDRIDZKLESEG

00010000

770715

.REM :

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DZRKL-E-D
PRODUCT NAME:	RK11/RK05 DYNAMIC TEST
DATE CREATED:	APRIL, 1977
MAINTAINER:	DIAGNOSTIC GROUP
AUTHOR:	JIM KAPADIA
REVISED BY:	PERVEZ ZAKI TOM SAWYER CHUCK HESS

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS MANUAL.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1975, 1977 BY DIGITAL EQUIPMENT CORPORATION

001

TABLE OF CONTENTS

1.0	ABSTRACT
2.0	REQUIREMENTS
2.1	EQUIPMENT
2.2	PRELIMINARY PROGRAMS
2.3	EXECUTION TIME
3.0	STARTING ADDRESSES
4.0	PROGRAM CONTROL MODES AND OPERATOR ACTION
4.1	PAPER TAPE
4.2	RKDP DUMP MODE
4.3	RKDP CHAIN MODE
4.4	ACT11
5.0	DRIVE SELECTION
6.0	SWITCH OPTIONS
7.0	PROGRAM DESCRIPTION
7.1	PERMISSIBLE USER PROGRAM MODIFICATIONS
8.0	SEEK TIMER AND GRAPHS
9.0	FUNCTION SELECTION PROGRAM
10.0	ERROR INFORMATION
11.0	UNEXPECTED TIMEOUTS
12.0	COMMONLY USED SUBROUTINES
13.0	SAMPLE GRAPH AND TIMER OUTPUTS

EO1

1.0 ABSTRACT

THE RK11/RK05 DYNAMIC TEST AIMS AT

1. DEMONSTRATING THE ELECTROMECHANICAL INTEGRITY OF THE DRIVE.
2. CHECKING THE LINEAR POSITIONER CONTROL AND SPEED CONTROL
3. VERIFYING THE INTEGRITY OF THE READ/WRITE LOGIC
4. PROVIDING A TIMER FOR THE SEEK FUNCTION.

THIS IS A TEST ONE LEVEL HIGHER THAN THE BASIC RK11 LOGIC TESTS.

2.0 REQUIREMENTS

2.1 EQUIPMENT

- A. PDP11 WITH CONSOLE TELETYPE.
- B. 8K OF MEMORY
- C. RK11 OR RKV11 CONTROLLER
- D. 1-8 RK05 OR RK05F DRIVES (DRIVE TYPES MAY BE MIXED)

2.2 PRELIMINARY PROGRAMS

RK11 LOGIC TEST I (MAINDEC-11-DZRKJ)
RK11 LOGIC TEST II (MAINDEC-11-DZRKK)

2.3 EXECUTION TIME

ERROR FREE FIRST PASS ON PDP11/20 WITH CORE MEMORY TAKES APPROXIMATELY 5 MINUTES (WITHOUT THE SEEK TIMER AND GRAPH, ADDITIONAL 3.5 MINUTES FOR THESE). LESS FOR FASTER MACHINES OR MEMORIES.

3.0 STARTING ADDRESS

200 FOR ANY NORMAL MODE OF OPERATION. ALL SWITCHES DOWN

210 FOR FUNCTION SELECTING PROGRAM (CONVERSATIONAL MODE).

4.0 PROGRAM CONTROL MODES & OPERATOR ACTION

PAPER TAPE LOADING
RKDP DUMP MODE
RKDP CHAIN MODE
ACT11

FO1

4.1 PAPER TAPE LOADING

4.1.1 LOAD PROGRAM INTO MEMORY USING STANDARD PROCEDURE FOR ABSOLUTE TAPES.

4.1.2 MAKE SURE THAT THE DRIVES TO BE CHECKED ARE LOADED WITH DISKS AND ARE IN 'RUN'. 'WRT ENABLE' THEM. CHECK THAT 'WRT PROT' LIGHT ON THESE DRIVES IS OFF. PUT DRIVES THAT ARE NOT TO BE TESTED ON 'LOAD'.

4.1.3 LOAD ADDRESS 200

4.1.4 SET SWITCHES IF DESIRED (SEE SEC 6.0)

PRESS START.

4.1.5 THE PROGRAM IDENTIFIES ITSELF

RK11 DYNAMIC TEST
MAINDEC-11-DZRKL-E

THEN IT PROCEEDS TO FIND WHICH DRIVES ARE PRESENT AND PRINTS OUT THE DRIVES FOUND. IF AN RK-05F IS DETECTED, AN F IS APPENDED TO THE DRIVE NUMBER:

DRIVES PRESENT

0
1

AFTER TYPING OUT THE DRIVE NUMBER THAT IS GOING TO BE TESTED, EXECUTION OF THE TESTS START.

AFTER ALL THE TESTS HAVE BEEN EXECUTED ON ONE DRIVE THEY ARE EXECUTED ON THE NEXT DRIVE, IF PRESENT. THIS IS REPEATED TILL ALL DRIVES ARE TESTED.

AT THE END OF A PASS THE FOLLOWING IS TYPED OUT:

END PASS X X=0,1,2.....

CONTROL IS TRANSFERRED BACK TO THE BEGINNING OF THE PROGRAM AND RE-EXECUTION BEGINS.

4.2 RKDP DUMP MODE

4.2.1 THE PROGRAM IS LOADED BY THE RKDP MONITOR.

4.2.2 SET SA=200. SELECT ANY SWITCHES YOU WANT AND PRESS START.

4.2.3 THE PROGRAM IDENTIFIES ITSELF AND PRINTS OUT:

'TO TEST' DRIVE 'N' HALT PROGRAM, REMOVE RKDP PACK AND REPLACE IT WITH A WORK PACK, CLEAR LOCATION 40, AND RESTART PROGRAM'

4.3 RKDP CHAIN MODE

THE PROGRAM IS CHAIN LOADED FROM RKDP PACK ON DRIVE 'N'. AFTER IDENTIFYING ITSELF, THE FOLLOWING MESSAGE APPEARS:

'DRIVE 'N' NOT TESTED'

DRIVE 'N' WILL NOT BE TESTED SINCE THE RKDP PACK IS ON THAT DRIVE.

4.4 ACT11 MODE

THE PROGRAM IS LOADED BY THE ACT11 MONITOR. AFTER IDENTIFYING ITSELF, ASCERTAINS THE NUMBER OF DRIVES PRESENT AND PROCEEDS TO TEST EACH OF THEM AS BEFORE.

5.0 DRIVE SELECTION

IF ANY PARTICULAR DRIVE IS TO BE SELECTED FOR TESTING, PUT THAT DRIVE ON 'RUN', 'WRITE ENABLE'. PUT THE REST OF THE DRIVES ON 'LOAD', 'WRITE LOCK'. THEN START AS USUAL.

6.0 SWITCH OPTIONS

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E. AN 11/34), THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE 'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176 (8). THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' WHENEVER THE PROGRAM ENTERS THE SCOPE ROUTINE OR BEGINS A NEW TEST. THE 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

'SWR = NNNNNN NEW ='

EACH TIME SWITCH SETTING ARE ENTERED, THE ENTIRE SWITCH REGISTER IMAGE MUST BE ENTERED. LEADING ZEROS ARE NOT REQUIRED. 'RUBOUT' AND 'CONTROL U' FUNCTIONS MAY BE USED TO CORRECT TYPING ERRORS DURING SWITCH ENTRY.

ON PROCESSORS WITH HARDWARE SWITCH REGISTERS, THE 'SOFTWARE' SWITCH REGISTER MAY BE USED. IF THE PROGRAM FINDS ALL 16 SWITCHES IN THE 'JP' POSITION, ALL SWITCH REGISTER REFERENCES WILL BE TO THE

'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST BE FOLLOWED.

SW<15>=1 HALT ON ERROR
 SW<14>=1 LOOP ON TEST
 SW<13>=1 INHIBIT ERROR PRINTOUTS
 SW<12>=1 CYCLE ON ERROR TO THE PREVIOUS
 'SCOPE' STATEMENT
 SW<11>=1 DUMP ALL RK11 REGISTERS ON ERROR
 SW<10>=1 RING BELL ON ERROR
 SW<09>=1 LOOP ON SPECIFIC ERROR
 SW<08>=1 LOOP ON TEST INDICATED BY USER (SEE
 SEC. 6.8)
 SW<06>=1 TYPE SEEK TIMER
 SW<05>=1 TYPE THE GRAPHS
 SW<04>=1 PRINT THE COMPLETE GRAPH

SW<03>=1 TERMINATE FUNCTION SELECTED BY USER
 SW<02>=1 DROP THE DRIVE AFTER MAXIMUM
 ALLOWABLE NUMBER OF ERRORS OCCUR
 SW<00>=1 ASK FOR PATTERN TO BE WRITTEN OR
 WRITE CHECKED (FUNCTION SELECTION
 PROGRAM)

- 6.1 SW<15>
 THE PROGRAM HALTS ON ENCOUNTERING AN ERROR. AFTER TYPING OUT THE ERROR MESSAGE AND PERTINENT INFORMATION, PRESSING "CONTINUE" RESTORES NORMAL OPERATION OF THE PROGRAM.
- 6.2 SW<14>
 THE PROGRAM LOOPS ON THE SUBTEST THAT IS BEING EXECUTED WHEN THE SWITCH IS PUT ON. THIS SWITCH IS USED NORMALLY ALONG WITH SW 15.
- 6.3 SW<13>
 THIS SWITCH INHIBITS ALL ERROR MESSAGES. NORMALLY USED WHEN LOOPING ON TEST (SW 14) OR LOOPING ON ERROR (SW 9).
- 6.4 SW<12>
 THIS SWITCH ALLOWS THE PROGRAM TO CYCLE FROM THE POINT OF ERROR TO THE PREVIOUS SCOPE STATEMENT. NOTE THAT IN DOING SO ANY INITIALIZATION BEING DONE

AT THE BEGINNING OF THE SUBTEST WILL BE DONE AGAIN AND AGAIN. SEE SEC. 6.7 FOR A DIFFERENT KIND OF SCOPE LOOP.

6.5 SW<11>

THIS SWITCH ALLOWS DUMPING OF ALL RK11 REGISTERS ON ENCOUNTERING AN ERROR.

6.6 SW<10>

RINGS A BELL ON ERROR. USEFUL WHEN ERROR TIMEOUT IS INHIBITED.

6.7 SW<09>

THIS SWITCH PROVIDES THE TIGHTEST POSSIBLE SCOPE LOOP. NOTE THAT UNLIKE SW12 THE INITIALIZATION OF PARAMETERS AT THE BEGINNING OF THE SUBTEST MAY NOT BE DONE IN THIS CASE. THIS SWITCH IS HELPFUL WHEN A PARTICULAR PART OF A SUBTEST IS BEING REPEATED USING DIFFERENT PARAMETERS AND YOU WANT TO SCOPE ON THE PARAMETER IN ERROR. (EXAMPLE: RKDA IS BEING WRITTEN AND READ BACK WITH COUNT PATTERNS FROM 1 TO 177777. PATTERN 561 IS GIVING ERROR. YOU MIGHT NOT WANT TO GO THROUGH THE 560 PATTERNS BEFORE HITTING ERROR ON THE 561TH PATTERN. IN THIS CASE SW 9 WILL GIVE YOU A SCOPE LOOP ON THE 561TH PATTERN ONLY.)

6.8 SW<08>

THIS SWITCH IS USED TO SELECT A PARTICULAR TEST FOR EXECUTION. WHEN THE PROGRAM IS STARTED (200) WITH THIS SWITCH SET, THE FOLLOWING MESSAGE APPEARS:

OCTAL TEST#?

THE USER SHOULD REPLY WITH THE TEST NUMBER (OCTAL) HE WANTS TO SELECT, FOLLOWED BY CARRIAGE RETURN.

THE SELECTED TEST IS EXECUTED AGAIN AND AGAIN. TO GET OUT OF THIS LOOP, PUT SW 8 BACK TO 0. THIS WILL RESUME NORMAL OPERATION OF THE PROGRAM. NOTE THAT BEFORE TEST 4 CAN BE EXECUTED TEST 2 SHOULD HAVE BEEN DONE AND TEST 6 SHOULD HAVE BEEN DONE BEFORE TEST 7.

6.9 SW<06>

THIS SWITCH WHEN SET MAKES THE PROGRAM TYPE THE SEEK
TIMER. THIS SWITCH CAN BE SET OR RESET BEFORE OR DURING THE
SEEK TIMER EXECUTION, AND EVEN WHILE THE TYPEOUT IS
OCCURING.

6.10 SW<05>

THIS SWITCH MAKES THE PROGRAM TYPE THE GRAPHS. IF
RESET BEFORE THE GRAPH-PLOTTING ROUTINE IS ENTERED,
THE GRAPHS WILL BE SKIPPED ENTIRELY. IT CAN BE RESET
EVEN AFTER SOME OF THE POINTS HAVE BEEN PLOTTED, TO
SKIP PLOTTING REST OF THE POINTS.

6.11 SW<04>

THIS SWITCH IS USED TO SELECT THE COMPLETE GRAPH
OUTPUT (SEEK TIMES OF ALL CYLINDERS ARE PLOTTED)
NORMALLY WHEN THIS SWITCH IS NOT SET, THE SMALL
GRAPH (ONLY SELECTED CYLINDERS PLOTTED) IS PRINTED
OUT.

6.12 SW<03>

THIS SWITCH WHEN SET TERMINATES THE EXECUTION OF THE
FUNCTION SELECTED BY THE USER (SA=210). A NEW
FUNCTION MAY BE INITIATED NOW. IF YOU WANT TO KEEP
ON LOOPING ON THE SAME FUNCTION, PUT SW 3 DOWN. SEE
SEC. 9.0.

6.13 SW<02>

THIS SWITCH ALLOWS THE PROGRAM TO DROP A DRIVE FROM
THE SELECTION LIST AND TESTING, AFTER MAXIMUM
ALLOWABLE ERROR COUNT (TOTAL NUMBER OF ERRORS) ON
THAT DRIVE IS EXCEEDED. THE MAXIMUM ALLOWABLE ERROR
COUNT IS 6, AFTER 6 ERRORS HAVE OCCURED THE DRIVE IS
DROPPED AND A MESSAGE (DRIVE # XXXXX DROPPED) IS
PRINTED.

6.14 SW<00>

THIS SWITCH IS TO BE USED WITH THE FUNCTION
SELECTION PROGRAM (SA=210). IF A WRITE OR A WRITE
CHECK FUNCTION IS SELECTED WITH THIS SW SET, THE
PROGRAM WILL ASK FOR THE PATTERN TO BE WRITTEN OR
WRITE CHECKED (PATRN?). THE USER SHOULD TYPE IN THE
(OCTAL) PATTERN. THIS PATTERN WILL BE WRITTEN (OR
WRITE CHECKED) ON THE DISK. FOR FURTHER INFORMATION
REFER TO SEC. 9.0.

PROGRAM DESCRIPTION

THE FIRST TEST IS AIMED AT DETECTING IMPEPENDING ELECTRO-MECHANICAL FAILURES IN THE DRIVE AND INNER/OUTER LIMIT SWITCHES.

IN THE NEXT TWO TESTS, THE DISK IS FORMATTED AND CHECKED FOR CORRECT FORMATTING. IF THE DISK IS AN RK-DSF, THE ENTIRE DISK IS FORMATTED EACH TIME THE EVEN DRIVE IS TESTED. NO FORMATTING IS DONE WHEN THE ODD DRIVE IS TESTED. THE DISK IS CHECKED EACH TIME FOR PROPER FORMAT, HOWEVER.

IN NEXT TWO TESTS THE SEEK LOGIC, POSITIONER, ETC ARE CHECKED OUT BY DOING IMPLIED SEEK, USING TWO DIFFERENT SEEKING PATTERNS. THE FIRST ONE IS A DECREASING SAW-TOOTH PATTERN (0-312-0-311-0-310....), THE SECOND ONE IS A CONVERGING-DIVERGING PATTERN (0-312-1-311-2-310....). ON GETTING AN ERROR, FURTHER ANALYSIS IS DONE TO FIND OUT MORE ABOUT THE NATURE OF ERROR. MANY TIMES ADDITIONAL INFORMATION IS GIVEN FOR THE CONVIENCE OF THE USER. RETRIES ARE DONE WHENEVER AN ERROR OCCURS.

IN THE SUBSEQUENT TESTS EXTENSIVE WRITING IS DONE USING MORE THAN 2000 DIFFERENT PATTERNS. THE DATA IS READ, (SOFTWARE) COMPARED, AND WRITE CHECKED.

EVERYTIME AN ERROR OCCURS RETRIES ARE DONE, TO CHECK IF IT WAS A RECOVERABLE ERROR OR NOT. THE USER CAN CHANGE THE PATTERNS TO BE WRITTEN ON THE DISK. THE DATA TRANSFER BUFFERS CAN BE RE-LOCATED BY THE USER TO DIFFERENT PARTS OF MEMORY. REFER TO LOCATIONS 'PBUF0', 'PBUF1', 'PAT1', 'PTRND1' IN THE LISTINGS FOR MORE DETAILS. SEE SEC 7.1.

THE SHUNT CURRENT CHANGE TEST WRITES, READS AND CHECKS FOR ERRORS ON CYLINDERS 127 AND 128. THIS REGION HAS CRITICAL "PACKING DENSITY" TO "WRITE CURRENT" RATIOS.

THE SEEK TIMER PROVIDES SEEK TIMES AND GRAPHS AS EXPLAINED IN SEC 8.0

A FUNCTION SELECTION SUB-PROGRAM IS PROVIDED FOR USER SELECTION OF FUNCTIONS. SEE SEC 9.0

EVERY TEST IN THE PROGRAM IS PRECEDED BY AN EXPLANATION OF THAT TEST. THE USER IS ADVISED TO REFER TO THAT, IF MORE INFORMATION IS NEEDED.

7.1 PERMISSIBLE USER PROGRAM MODIFICATIONS

THE USER CAN MAKE MINOR CHANGES IN POINTERS, TABLES, ETC. TO TAKE CARE OF HIS SPECIAL NEEDS. IT IS ADVISABLE TO MAKE CHANGES IF ANY, RIGHT AT THE BEGINNING.

- 7.1.1 SEEK TIMING CAN BE DONE BETWEEN ANY TWO CYLINDERS, BY MAKING CHANGES DESCRIBED IN THE CYLINDER ADDRESS TABLE AT LOCATIONS 'SOAD' AND 'SIAD' IN THE LISTINGS.
- 7.1.2 IN CASE YOU HAVE A LINE PRINTER AND WANT YOUR OUTPUT ON THE LINE PRINTER, CHANGE LOCATION 'STPS' TO 177514 AND LOCATION 'STPB' TO 177516 (LINE PRINTER VECTORS).
- 7.1.3 INPUT/OUTPUT DATA BUFFERS (FROM WHERE DATA TRANSFERS WILL BE DONE TO AND FROM THE DISK) CAN BE RELOCATED TO ANYWHERE IN THE 28K OF MEMORY (DO NOT OVERLAY THE PROGRAM). THIS CAN BE DONE BY CHANGING THE CONTENTS OF LOCATIONS 'PBUFO' AND 'PBUF1' TO THE STARTING ADDRESSES OF THE TWO USER SELECTED BUFFERS. IT SHOULD BE NOTED THAT EACH OF THE TWO BUFFERS SHOULD BE 768 (DECIMAL) WORD LONG.
- 7.1.4 FOUR DIFFERENT PATTERN GENERATOR ROUTINES HAVE BEEN USED IN THIS PROGRAM: A. PTGEN0 B. PTGEN1 C. PTGEN2 D. PTGEN3. THEY HAVE BEEN DESCRIBED IN DETAIL AT CORRESPONDING LOCATIONS IN THE LISTING. THE ORDER IN WHICH THEY ARE CALLED IS DESCRIBED AT THE BEGINNING OF TEST 6. THIS CALLING ORDER CAN BE CHANGED BY MAKING CHANGES IN THE FOUR POINTERS A. 'PAT0' B. 'PAT1' C. 'PAT2' D. 'PAT3'. THESE 4 POINTERS CONTAIN THE STARTING ADDRESS OF EACH ROUTINE.
- 7.1.5 AS A SPECIAL CASE OF THE ABOVE, YOU CAN WRITE THE SAME TWO (OR ONE) PATTERN/S ON THE ENTIRE DISK USING 'PTGEN0' ROUTINE. TO WRITE THE SAME ONE PATTERN:
CHANGE LOCATION 'PAT1' TO 'PTGEN0' (STARTING ADDRESS OF PTGEN0)
CHANGE LOCATION 'PAT2' TO 'PTGEN0' (STARTING ADDRESS OF PTGEN0)
CHANGE LOCATION 'PAT3' TO 'PTGEN0' (STARTING ADDRESS OF PTGEN0)
FILL LOCATIONS 'PTRNO1' AND 'PTRNO2' WITH THE PATTERN YOU WANT.
TO WRITE 2 DIFFERENT PATTERNS (IN ALTERNATING SECTORS):
CHANGE 'PAT1', 'PAT2' AND 'PAT3' AS DESCRIBED ABOVE.
FILL 'PTRNO1' AND 'PTRNO2' WITH THE TWO PATTERNS YOU WANT.
- 7.1.6 IN TEST 10, IF YOU WANT TO WRITE AND CHECK CYLINDERS 127 AND 128 WITH PATTERNS OTHER THAN THE 12 USED, CHANGE ANY OR ALL OF THE 12 POINTERS 'SPI' THROUGH

'SP12' TO CONTAIN PATTERNS YOU WANT.

8.0 SEEK TIMER & GRAPHS

THE LAST TEST IN THIS PROGRAM IS THE SEEK TIMER. IN ORDER TO TIME THE SEEKS, THE SECTOR COUNTER HAS BEEN USED AS A TIME BASE. THUS THE ACCURACY OF THE TIMES RECORDED IS AS GOOD AS THE ACCURACY OF THE SECTOR COUNTER (WHICH IN TURN DEPENDS ON THE ROTATION SPEED OF THE DISK).

IN THE FIRST PART OF THIS TIMER, SOME CRITICAL SEEKS HAVE BEEN TIMED (CYLINDERS 0-1, 179-181, 0-3, 0-16, 0-32, 0-202, 0=100) EACH SEEK IS DONE 100 TIMES, TIMES ARE RECORDED, THEN THE TIMES ARE SORTED OUT AND A PRINTOUT IS GIVEN SHOWING HOW MANY TIMES A PARTICULAR SEEK TIME WAS OBTAINED. EXAMPLE: SEEK BETWEEN 0 AND LAST CYLINDER WAS DONE 100 TIMES. 99 TIMES A SEEK TIME OF 95 MS WAS OBTAINED, ONCE IT GAVE 100 MS. THIS GIVES THE USER AN IDEA OF HOW CONSISTENT ARE THE SEEK TIMES.

IF YOU WANT TO TIME SEEK BETWEEN ANY OTHER SET OF CYLINDERS, YOU CAN DO BY FOLLOWING THE INSTRUCTIONS AT LOCATION 'SOAD' IN LISTINGS. SEE SEC 7.1

IN THE SECOND PART, A GRAPH OF THE 'CYLINDER SEEKED FROM 0' IS PLOTTED AGAINST 'SEEK TIME'. TWO GRAPHS ARE AVAILABLE, NORMALLY THE SMALL GRAPH IS PRINTED OUT. THE SMALL GRAPH PLOTS THE SEEK TIMES FOR SELECTED CYLINDERS (ABOUT 49) COVERING THE RANGE FROM CYLINDER 0 TO 202. IT GIVES THE USER A QUICK SEEK CHARACTERISTICS OF A DRIVE.

THE OPTIONAL COMPLETE GRAPH (SW 4) GIVES A GRAPH SIMILAR TO THE ABOVE ONE, BUT PLOTS ALL THE CYLINDERS (203).

THE GRAPH SHOWN ON LAST PAGE IS A SAMPLE OUTPUT. IT SHOULD BE REALIZED THAT DIFFERENT DRIVES MAY HAVE A SLIGHTLY DIFFERENT CHARACTERISTIC.

9.0 FUNCTION SELECTION PROGRAM

THIS PROGRAM GIVES THE USER A CAPABILITY TO SELECT A FUNCTION AND EXECUTE IT, FROM THE CONSOLE TELETYPE.

STARTING ADDRESS=210

ON STARTING THE PROGRAM AT 210, THE FOLLOWING QUESTION APPEARS:

FUNCTION?

NO1

THE REPLY SHOULD BE:

WR	FOR WRITE
WC	FOR WRITE CHECK
RD	FOR READ
RC	FOR READ CHECK
CR	FOR CONTROL RESET
DR	FOR DRIVE RESET
SK	FOR SEEK DR

ALL COMMANDS SHOULD BE TERMINATED BY A CARRIAGE RETURN. DEPENDING ON WHICH FUNCTION IS GIVEN THE

FOLLOWING QUESTIONS APPEAR:

RKBA? TYPE IN THE BUS ADDRESS (OCTAL) FOLLOWED BY A C.R.

RKDA? TYPE IN THE DISK ADDRESS (OCTAL) FOLLOWED BY C.R.

IF A NON-EXISTENT CYLINDER OR SECTOR IS SELECTED, THE QUESTION IS REPEATED AGAIN.

#WORDS? TYPE IN THE NUMBER OF WORDS YOU WANT TO TRANSFER. IT SHOULD BE IN OCTAL. THUS IF YOU WANT TO READ A SECTOR TYPE IN 400 FOLLOWED BY C.R. ANY NUMBER OF WORDS CAN BE TRANSFERRED DEPENDING ON THE BUFFER SIZE AVAILABLE.

FOR A WRITE FUNCTION: IF SW0 IS SET TO 1 THE PROGRAM WILL ASK FOR THE DATA PATTERN TO BE WRITTEN:
PATRN? THE USER SHOULD TYPE IN THE DATA PATTERN (OCTAL) TO BE WRITTEN, FOLLOWED BY <CR>. THE PATTERN WILL BE WRITTEN ON THE DISK. NOTE THE NUMBER OF WORDS TO BE WRITTEN AND THE DISK ADDRESS SHOULD BE SPECIFIED.

FOR A WRITE CHECK FUNCTION: IF SW 0 IS SET TO 1, THE USER IS ASKED FOR THE PATTERN TO BE WRITE CHECKED:
PATRN? THE USER SHOULD TYPE IN THE (OCTAL) PATTERN.

FOR A SEEK FUNCTION: CYL1? CYL2? IN REPLY TO THESE, TYPE IN THE CYLINDER NUMBERS (OCTAL) BETWEEN WHICH THE SEEK IS TO BE DONE. IF A NON EXISTENT CYLINDER IS TYPED IN THE QUESTION IS REPEATED AGAIN.

THE FUNCTION IS EXECUTED AGAIN AND AGAIN. TO GET OUT OF THIS LOOP SW 3 SHOULD BE SET, AT THIS POINT THE QUESTION (FUNCTION?) IS ASKED AGAIN.

IF UPON EXECUTION OF A FUNCTION AN ERROR OCCURS IT IS REPORTED. ALL SWITCH OPTIONS WHICH APPLY TO ANY OTHER ERROR, ALSO APPLY TO THIS ERROR.

IF ON INPUTTING A NUMBER OR COMMAND A MISTAKE IS MADE, THE INPUT STRING CAN BE DELETED BY HITTING 'RUBOUT' KEY, THE NEW STRING CAN BE TYPED IN AGAIN.

10.0 ERROR INFORMATION

WHENEVER AN ERROR MESSAGE IS PRINTED OUT, ALL REGISTERS AND OTHER DATA PERTAINING TO THE ERROR ARE ALSO GIVEN. RKDS, RKER...RKBA INDICATE THE CONTENTS OF THE CORRESPONDING REGISTERS AT THE TIME OF ERROR.

EVERY ERROR MESSAGE CONTAINS A PC. THIS PC INDICATES THE POSITION IN PROGRAM WHERE THE ERROR CALL IS LOCATED. THE ERROR MESSAGE, BECAUSE OF PRACTICAL CONSIDERATIONS IS MADE SHORT AND MEANINGFUL. THE USER IS ADVISED TO LOOK UP THE PC IN THE PROGRAM LISTING, WHERE HE WILL FIND MORE INFORMATION ABOUT THE ERROR. IN MANY INSTANCES, A SINGLE FAULT WILL GIVE RISE TO MORE THAN ONE ERROR REPORT. A LITTLE DELIBERATION AND CAREFUL EXAMINATION OF THE DATA GIVEN WILL BE CERTAINLY VERY HELPFUL. A BRIEF EXPLANATION OF WHAT IS BEING CHECKED IN THE SUBTEST IS GIVEN AT THE BEGINNING OF EVERY SUBTEST. ALL THE NUMBERS GIVEN WITH ERROR MESSAGES ARE IN OCTAL.

AT TIMES WHEN AN ERROR OCCURS BESIDES THE ERROR PRINTOUT MORE PRINTOUTS OCCUR. THEY ARE GIVEN TO HELP THE USER UNDERSTAND THE PROBLEM.

11.0 UNEXPECTED TIMEOUTS AND RK11 INTERRUPTS

WHEN AN UNEXPECTED TIMEOUT OCCURS, THE PC AT WHICH THE OUT OCCURRED IS TYPED OUT AND THE PROGRAM HALTS. IF IT IS INTACT, IT CAN BE RESTARTED BY PRESSING CONTINUE.

IF AN UNEXPECTED RK11 INTERRUPT OCCURS THE PROGRAM TYPES OUT THE PC AT WHICH THE INTERRUPT CAME IN AND THEN HALTS. PRESSING CONTINUE WOULD RESTART THE PROGRAM FROM BEGINNING.

12.0 COMMONLY USED SUBROUTINES

A BRIEF EXPLANATION OF EVERY SUBROUTINE IS GIVEN IN THE LISTINGS (JUST BEFORE THE CODE FOR THAT SUBROUTINE). ALL SUB-ROUTINES ARE LISTED IN THE 'TABLE OF CONTENTS' FOUND AT THE BEGINNING OF LISTINGS. THESE ARE TWO WAYS IN WHICH ROUTINES ARE CALLED, 1. JSR PC ROUTINE 2. THROUGH AN ENCODED TRAP INSTRUCTION. THE LOWER BYTE OF THE 'TRAP' INSTRUCTION IS USED TO INDEX THROUGH THE TRAP TABLE

(STRPAD) FOR THE STARTING ADDRESS OF THE DESIRED ROUTINE.

13.0 SAMPLE GRAPH AND SEEK TIMER OUTPUTS

'# OF SEEKS' INDICATES THE NUMBER OF TIMES A PARTICULAR 'SEEK TIME' WAS OBTAINED. NOTE THAT TIMES ARE RECORDED FOR BOTH FORWARD AND REVERSE SEEKS, BETWEEN A SET OF CYLINDERS.

SEEK TIME SCALE FACTOR=0.01 MILLI SECS

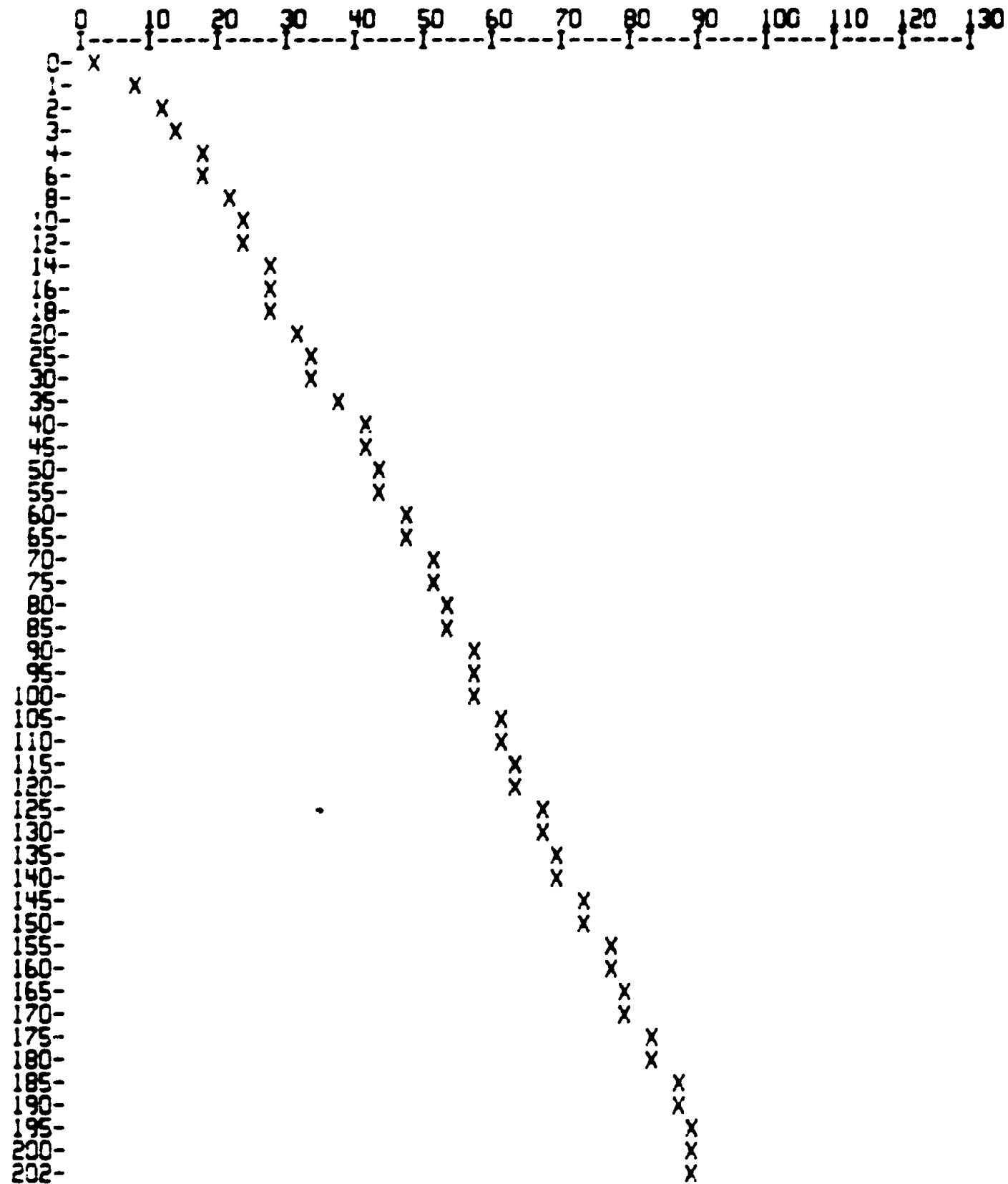
# OF SEEKS	SEEK TIME	# OF SEEKS	SEEK TIME
CYLS:0-202			
		REVRSE	
100	9075	100	9075
CYLS:0-1			
		REVRSE	
100	825	100	1155
CYLS:179-181			
		REVRSE	
100	1155	100	1155
CYLS:0-3			
		REVRSE	
100	1485	100	1485
CYLS:0-16			
		REVRSE	
100	3135	100	3135
CYLS:0-32			
		REVRSE	
100	3795	100	3795
CYLS:0-100			
		REVRSE	
100	5775	100	5775

002

MC-11-DZRAL-E, R11-RK05 DYNAMIC TEST MACY11 30(1046) 14-JUL-77 08:03 PAGE 16
DZRAL.E.P11 26-APR-77 12:27

X AXIS - SEEK TIME - MILI SECS
Y AXIS - CYLINDER SEEKED FROM 0

SAMPLE OUTPUT



E02

MC-11-CZRAL-E, RK11-RK05 DYNAMIC TEST MACY11 30(1046) 14-JUL-77 08:03 PAGE 17
CZRAL.E.P11 26-APR-77 12:27

.

.

825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867

```

.TITLE MD-11-DZRKL-E, RK11-RK05 DYNAMIC TEST
.*COPYRIGHT (C) 1974, 1977
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754
.*
.*PROGRAM BY JIM KAPADIA
.*
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
.*
.*JANUARY 1975
.*
.*REVISED MARCH 1976 BY TOM SAWYER
.*REVISED BY CHUCK HESS, AUGUST, 1976

.SBTTL OPERATIONAL SWITCH SETTINGS
.*
.*      SWITCH      USE
.*      -----
.*      15      HALT ON ERROR
.*      14      LOOP ON TEST
.*      13      INHIBIT ERROR TYPEOUTS
.*      12      CYCLE ON ERROR TO PREVIOUS 'SCOPE'
.*      10      BELL ON ERROR
.*      9       LOOP ON ERROR
.*      8       SELECT TEST TYPED IN BY USER
.*      6       EXECUTE THE SEEK TIMER (TEST 11)
.*      5       TYPE THE SEEK TIMER GRAPHS (TEST 11)
.*      4       TYPE THE COMPLETE GRAPH (ALL SEEK TIMES
.*              NOTE, OTHERWISE YOU GET SMALL GRAPH
.*      3       TERMINATE FUNCTION SELECTED BY USER
.*              (FOR FUNCTION SELECTION PROGRAM SA=210
.*      2       DROP THE DRIVE AFTER MAXIMUM ALLOWABLE
.*              NUMBER OF ERRORS HAVE OCCURED
.*      0       ASK FOR PATTERN TO BE WRITTEN (OR WRITE
.*              CHECKED), IN FUNCTION SELECTION PROGRAM
.*      11      DUMP OUT ALL RK11 REGISTERS ON ERROR

.*
.*      YOU ARE ADVISED TO READ THE DOCUMENT FOR THIS PROGRAM.
.*      FUNCTION SELECTION PROGRAM STARTS AT 210.

```

.SBTTL BASIC DEFINITIONS

.*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***

STACK= 1100

.EQUIV EMT,ERROR

::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

STKLMT= 177774

::STACK LIMIT REGISTER

PIRQ= 177772

::PROGRAM INTERRUPT REQUEST REGISTER

DSWR= 177570

::HARDWARE SWITCH REGISTER

DDISP= 177570

::HARDWARE DISPLAY REGISTER

.*GENERAL PURPOSE REGISTER DEFINITIONS

R0= %0

::GENERAL REGISTER

R1= %1

::GENERAL REGISTER

R2= %2

::GENERAL REGISTER

R3= %3

::GENERAL REGISTER

R4= %4

::GENERAL REGISTER

R5= %5

::GENERAL REGISTER

R6= %6

::GENERAL REGISTER

R7= %7

::GENERAL REGISTER

SP= %6

::STACK POINTER

PC= %7

::PROGRAM COUNTER

.*PRIORITY LEVEL DEFINITIONS

PR0= 0

::PRIORITY LEVEL 0

PR1= 40

::PRIORITY LEVEL 1

PR2= 100

::PRIORITY LEVEL 2

PR3= 140

::PRIORITY LEVEL 3

PR4= 200

::PRIORITY LEVEL 4

PR5= 240

::PRIORITY LEVEL 5

PR6= 300

::PRIORITY LEVEL 6

PR7= 340

::PRIORITY LEVEL 7

.*"SWITCH REGISTER" SWITCH DEFINITIONS

SW15= 100000

SW14= 40000

SW13= 20000

SW12= 10000

SW11= 4000

SW10= 2000

SW09= 1000

SW08= 400

SW07= 200

SW06= 100

SW05= 40

SW04= 20

SW03= 10

SW02= 4

924 000002
 925 000001
 926
 927
 928
 929
 930
 931
 932
 933
 934
 935
 936
 937
 938 100000
 939 040000
 940 020000
 941 010000
 942 004000
 943 002000
 944 001000
 945 000400
 946 000200
 947 000100
 948 000040
 949 000020
 950 000010
 951 000004
 952 000002
 953 000001
 954
 955
 956
 957
 958
 959
 960
 961
 962
 963
 964
 965
 966 000004
 967 000010
 968 000014
 969 000014
 970 000014
 971 000020
 972 000024
 973 000030
 974 000034
 975 000060
 976 000064
 977 000240
 978
 979

SW01= 2
 SW00= 1
 .EQUIV SW09,SW9
 .EQUIV SW08,SW8
 .EQUIV SW07,SW7
 .EQUIV SW06,SW6
 .EQUIV SW05,SW5
 .EQUIV SW04,SW4
 .EQUIV SW03,SW3
 .EQUIV SW02,SW2
 .EQUIV SW01,SW1
 .EQUIV SW00,SW0
 ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
 BIT15= 100000
 BIT14= 40000
 BIT13= 20000
 BIT12= 10000
 BIT11= 4000
 BIT10= 2000
 BIT09= 1000
 BIT08= 400
 BIT07= 200
 BIT06= 100
 BIT05= 40
 BIT04= 20
 BIT03= 10
 BIT02= 4
 BIT01= 2
 BIT00= 1
 .EQUIV BIT09,BIT9
 .EQUIV BIT08,BIT8
 .EQUIV BIT07,BIT7
 .EQUIV BIT06,BIT6
 .EQUIV BIT05,BIT5
 .EQUIV BIT04,BIT4
 .EQUIV BIT03,BIT3
 .EQUIV BIT02,BIT2
 .EQUIV BIT01,BIT1
 .EQUIV BIT00,BIT0
 ;*BASIC "CPU" TRAP VECTOR ADDRESSES
 ERRVEC= 4 ; TIME OUT AND OTHER ERRORS
 RESVEC= 10 ; RESERVED AND ILLEGAL INSTRUCTIONS
 TRITVEC= 14 ; "T" BIT
 TRIVVEC= 14 ; TRACE TRAP
 BPTVEC= 14 ; BREAKPOINT TRAP (BPT)
 IOTVEC= 20 ; INPUT/OUTPUT TRAP (IOT) **SCOPE**
 PWRVEC= 24 ; POWER FAIL
 EMTVEC= 30 ; EMULATOR TRAP (EMT) **ERROR**
 TRAPVEC= 34 ; "TRAP" TRAP
 TKVEC= 60 ; TTY KEYBOARD VECTOR
 TPVEC= 64 ; TTY PRINTER VECTOR
 PIRQVEC= 240 ; PROGRAM INTERRUPT REQUEST VECTOR
 .SBTTL TRAP CATCHER

```

980
981      000000      . = 0
982      ; *ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
983      ; *SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
984      ; *LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
985      000174      000174      . = 174
986      000174      000000      DISPREG: .WORD 0      ;; SOFTWARE DISPLAY REGISTER
987      000176      000000      SWREG: .WORD 0      ;; SOFTWARE SWITCH REGISTER
988      000200      000137      002462      .SBTTL STARTING ADDRESS(E5)
989      000200      000137      002462      JMP 2*START ;; JUMP TO STARTING ADDRESS OF PROGRAM
990
991      000210      000210      . = 210
992      000210      105237      001216      INCB FFUNC      ; SET FLAG INDICATING SELECTION OF
993      000214      000137      002462      JMP 2*START      ; FUNCTION PROGRAM.
994      .SBTTL ACT11 HOOKS
995
996      ; *****
997      ; HOOKS REQUIRED BY ACT11
998      000220      000046      $SVPC=      ; SAVE PC
999      000046      015330      . = 46
1000      000046      015330      SENDAD      ;; 1) SET LOC.46 TO ADDRESS OF SENDAD IN .SECP
1001      000052      000052      . = 52
1002      000052      000000      .WORD 0      ;; 2) SET LOC.52 TO ZERO
1003      000220      000220      . = $SVPC      ;; RESTORE PC
1004

```

.SBTTL COMMON TAGS

 : THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
 : USED IN THE PROGRAM.

1005
 1006
 1007
 1008
 1009
 1010
 1011 001100
 1012 001100 000000
 1013 001100 000000
 1014 001102 000
 1015 001103 000
 1016 001104 000000
 1017 001106 000000
 1018 001110 000000
 1019 001112 000000
 1020 001114 000
 1021 001115 001
 1022 001116 000000
 1023 001120 000000
 1024 001122 000000
 1025 001124 000000
 1026 001126 000000
 1027 001130 000000
 1028 001132 000000
 1029 001134 000
 1030 001135 000
 1031 001136 000000
 1032 001140 177570
 1033 001142 177570
 1034 001144 177560
 1035 001146 177562
 1036 001150 177564
 1037 001152 177566
 1038 001154 000
 1039 001155 002
 1040 001156 012
 1041 001157 000
 1042 001160 000000
 1043
 1044 001162 000000
 1045 001164 000000
 1046 001166 000000
 1047 001170 000000
 1048 001172 000000
 1049 001174 000000
 1050 001176 000000
 1051 001200 000000
 1052 001202 000000
 1053 001204 000000
 1054 001206 177607 000377
 1055 001212 077
 1056 001213 015
 1057 001214 000012
 1058
 1059
 1060

SCMTAG: . =1100
 \$PASS: .WORD 0
 \$STNM: .BYTE 0
 \$ERFLG: .BYTE 0
 \$ICNT: .WORD 0
 \$LPADR: .WORD 0
 \$LPERR: .WORD 0
 \$ERTTL: .WORD 0
 \$ITEMB: .BYTE 0
 \$ERMAX: .BYTE 1
 \$ERRPC: .WORD 0
 \$GDADR: .WORD 0
 \$BDADR: .WORD 0
 \$GDDAT: .WORD 0
 \$BDDAT: .WORD 0
 .WORD 0
 \$AUTOB: .BYTE 0
 \$INTAG: .BYTE 0
 .WORD 0
 \$SWR: .WORD DSWR
 \$DISPLAY: .WORD DDISP
 \$TKS: 177560
 \$TKB: 177562
 \$TPS: 177564
 \$TPB: 177566
 \$NULL: .BYTE 0
 \$FILLS: .BYTE 2
 \$FILLC: .BYTE 12
 \$TPFLG: .BYTE 0
 \$REGAD: .WORD 0
 \$REG0: .WORD 0
 \$REG1: .WORD 0
 \$REG2: .WORD 0
 \$REG3: .WORD 0
 \$REG4: .WORD 0
 \$REG5: .WORD 0
 \$REG6: .WORD 0
 \$REG7: .WORD 0
 \$REG10: .WORD 0
 \$ESCAPE: 0
 \$BELL: .ASCIIZ <207><377><377>
 \$QUES: .ASCII /?
 \$CRLF: .ASCII <15>
 \$LF: .ASCIIZ <12>

;; START OF COMMON TAGS
 ;; CONTAINS PASS COUNT
 ;; CONTAINS THE TEST NUMBER
 ;; CONTAINS ERROR FLAG
 ;; CONTAINS SUBTEST ITERATION COUNT
 ;; CONTAINS SCOPE LOOP ADDRESS
 ;; CONTAINS SCOPE RETURN FOR ERRORS
 ;; CONTAINS TOTAL ERRORS DETECTED
 ;; CONTAINS ITEM CONTROL BYTE
 ;; CONTAINS MAX. ERRORS PER TEST
 ;; CONTAINS PC OF LAST ERROR INSTRUCTION
 ;; CONTAINS ADDRESS OF 'GOOD' DATA
 ;; CONTAINS ADDRESS OF 'BAD' DATA
 ;; CONTAINS 'GOOD' DATA
 ;; CONTAINS 'BAD' DATA
 ;; RESERVED--NOT TO BE USED
 ;; AUTOMATIC MODE INDICATOR
 ;; INTERRUPT MODE INDICATOR
 ;; ADDRESS OF SWITCH REGISTER
 ;; ADDRESS OF DISPLAY REGISTER
 ;; TTY KBD STATUS
 ;; TTY KBD BUFFER
 ;; TTY PRINTER STATUS REG. ADDRESS
 ;; TTY PRINTER BUFFER REG. ADDRESS
 ;; CONTAINS NULL CHARACTER FOR FILLS
 ;; CONTAINS # OF FILLER CHARACTERS REQUIRED
 ;; INSERT FILL CHARS. AFTER A "LINE FEED"
 ;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES,
 ;; CONTAINS THE ADDRESS FROM
 ;; WHICH (\$REG0) WAS OBTAINED
 ;; CONTAINS ((\$REGAD)+0)
 ;; CONTAINS ((\$REGAD)+2)
 ;; CONTAINS ((\$REGAD)+4)
 ;; CONTAINS ((\$REGAD)+6)
 ;; CONTAINS ((\$REGAD)+10)
 ;; CONTAINS ((\$REGAD)+12)
 ;; CONTAINS ((\$REGAD)+14)
 ;; CONTAINS ((\$REGAD)+16)
 ;; CONTAINS ((\$REGAD)+20)
 ;; ESCAPE ON ERROR ADDRESS
 ;; CODE FOR BELL
 ;; QUESTION MARK
 ;; CARRIAGE RETURN
 ;; LINE FEED

K02

```

1061 ;IN CASE YOU WANT THE OUTPUT TO COME OUT ON LINE PRINTER, (IF YOU HAVE
1062 ;ONE), MAKE THE FOLLOWING CHANGES ABOVE:
1063
1064 ;CHANGE CONTENTS OF 'STPS' TO 177514 (LPT VECTOR)
1065 ;CHANGE CONTENTS OF 'STPB' TO 177516 ( " ")
1066
1067 ;TAGS AND GENERAL DATA AREA
1068
1069 001216 000000 FFUNC: .WORD 0 ;FLAG SET, TO INDICATE ENTRY INTO FUNCTION PROGRAM
1070 001220 000000 XXDPMO: .WORD 0 ;IF PROGRAM LOADED BY XXDP, THE
1071 ;LOWER BYTE HAS THE DRIVE NUMBER
1072 ;AND THE UPPER BYTE CONTAINS THE RK05 'XXDP' CODE
1073 001222 000 LUPSW: .BYTE 0 ;FLAG SET TO INDICATE THAT A
1074 ;PARTICULAR TEST WAS SELECTED BY USER 'SW 8'
1075
1076
1077
1078 001223 000 DRVDON: .BYTE 0 ;CONTAINS NUMBER OF DRIVES THAT HAVE
1079 ;BEEN ALREADY CHECKED
1080 001224 000 DRIVS: .BYTE 0 ;CONTAINS TOTAL # OF DRIVES PRESENT
1081
1082 001226 .EVEN
1083
1084
1085 001226 000000 DRVPTR: 0 ;CONTAINS POINTER TO INDICATOR STARTING
1086 ;WHICH CHECKING SHOULD BE DONE FOR NEXT
1087 ;AVAILABLE DRIVE
1088 001230 000000 DRIVAD: 0 ;CONTAINS THE ADDRESS OF THE DRIVE
1089 ;BEING TESTED
1090
1091 001232 000000 DRIVO: 000000 ;THESE ARE FLAGS TO INDICATE
1092 001234 020000 DRIV1: 020000 ;THAT A PARTICULAR DRIVE IS
1093 001236 040000 DRIV2: 040000 ;PRESENT. BIT 0 IS SET TO
1094 001240 060000 DRIV3: 060000 ;INDICATE THAT. BITS 13, 14, 15
1095 001242 100000 DRIV4: 100000 ;CONTAIN THE LOGICAL DRIVE
1096 001244 120000 DRIV5: 120000 ;ADDRESS
1097 001246 140000 DRIV6: 140000
1098 001250 160000 DRIV7: 160000
1099
1100
1101
1102 001252 000000 RETRY1: 0 ;GENERAL REGISTERS
1103 001254 000000 RETRY2: 0
1104 001256 000000 RETRY3: 0
1105
1106
1107 001260 000000 INADR: 0 ;CONTAINS INNER ADDRESS
1108 001262 000000 OUTADR: 0 ;CONTAINS OUTER ADDRESS
1109 001264 000000 TIMER: 0
1110
1111
1112
1113 001266 00C015 BUFR: .BLKW 13. ;GENERAL BUFFERS
1114 001320 00C015 BUFR1: .BLKW 13.
1115 001352 00C015 BUFR2: .BLKW 13.
1116

```

L02

```

1117
1118      ;IN CASE, YOU WANT TO USE BUFFERS STARTING AT SOME OTHER MEMORY
1119      ;ADDRESS YOU CAN DO SO BY CHANGING THE FOLLOWING POINTERS
1120      ;BOTH THE BUFFERS SHOULD BE 768 (DECIMAL) WORDS LONG.
1121
1122      001404 026446      PBUF0:  IOBUF0      ; POINTER TO THE STARTING ADDRESS OF THE
1123      ; BUFFER USED TO READ INTO FROM DISK.
1124      001406 031446      PBUF1:  IOBUF1      ; POINTER TO STARTING ADDRESS OF BUFFER
1125      ; IN WHICH PATTERNS ARE GENERATED. (WRITING
1126      ; IS DONE FROM THIS BUFFER)
1127      001410 000000      BUFLG0: .WORD 0      ; FLAG FOR 'IOBUF0'
1128      001412 000000      BUFLG1: .WORD 0      ; FLAG FOR 'IOBUF1'
1129
1130
1131      001414 010106      PAT0:   PTGEN0      ; ADRES OF 'PATRN GENERATOR 0'
1132      ; ROUTINE
1133      001416 010170      PAT1:   PTGEN1      ; ADRES OF 'PATRN GENERATOR 1'
1134
1135      001420 010272      PAT2:   PTGEN2      ; ADRES OF 'PATRN GENRATOR 2'
1136
1137      001422 010334      PAT3:   PTGEN3      ; ADRES OF 'PATRN GENRATOR 3'
1138
1139      001424 000000      PRSPAT: .WORD 0      ; CONTAINS THE POINTER TO THE
1140      ; ADRES OF 1 OF THE 3 'PATRN
1141      ; GENERATOR' ROUTINES
1142      001426 000000      NXTPAT: .WORD 0      ; SAME AS ABOVE
1143
1144      001430 000000      PGSUBR: .WORD 0
1145
1146      001432 000000      DSKADR: .WORD 0      ; CONTAINS DISK ADRES (DA)
1147
1148      001434 000000      BUSADR: .WORD 0      ; CONTAINS BUS ADRES (BA)
1149
1150      001436 000000      WRDCNT: .WORD 0      ; CONTAINS WORD COUNT
1151
1152      001440 000000      WDSKAD: .WORD 0      ; CONTAINS DISK ADRES
1153
1154      001442 000000      WBUSAD: .WORD 0      ; CONTAINS BUS ADRES
1155
1156      001444 000000      WWRDCN: .WORD 0      ; CONTAINS WORD COUNT
1157
1158      001446 000000      BUFNO:  .WORD 0      ; CONTAINS STARTING ADRES
1159
1160      001450 000000      ADRES:  .WORD 0      ; OF A BUFFER
1161
1162      ;RK11 REGISTERS
1163      ;IF FOR ANY REASON THE REGISTER ADDRESSES ARE DIFFERENT FROM
1164      ;THESE (BELOW) THE CONTENTS OF THE APPROPRIATE POINTERS SHOULD
1165      ;BE MODIFIED SO THAT THE CORRECT REGISTER ADDRESS IS USED.
1166
1167
1168      001452 177400      RKDS:   177400
1169      001454 177402      RKER:   177402
1170      001456 177404      RKCS:   177404
1171      001460 177406      RKWC:   177406
1172      001462 177410      RKBA:   177410

```

1173 001464 177412
1174 001466 177416
1175
1176 001470 000200
1177
1178
1179
1180
1181
1182
1183
1184 001472 000220
1185
1186
1187
1188
1189
1190
1191
1192 001474 000000
1193 001476 000000
1194 001500 000000
1195 001502 000000
1196
1197 001504 000000
1198 001506 000000
1199 001510 000000
1200 001512 000000
1201 001514 000000
1202 001516 000000
1203 001520 000000
1204 001522 000000
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214 001524 000000
1215 001526 000000
1216 001530 013140
1217 001532 000000
1218 001534 000000
1219 001536 000000
1220 001540 000000
1221
1222
1223 001542 014500
1224 001544 000040
1225 001546 013240
1226 001550 000140
1227 001552 001000
1228 001554 002000

RKDA: 177412
RKDB: 177416

RKPRI: 200

RKVEC: 220

INDX1: 0
INDX2: 0
INDX3: 0
INDX4: 0

ERCNT1: 0
ERCNT2: 0
ERCNT3: 0
ERCNT4: 0
ERCNT5: 0
ERCNT6: 0
ERCNT7: 0
ERCNT8: 0

;CONTAINS THE CPU LEVEL (4) AT WHICH
;RK11 NORMALLY INTERRUPTS. THIS WORD
;SHOULD BE CHANGED IF RK11 IS DESIGNATED
;A BR LEVEL OTHER THAN S.EXP: IF IT
;IS CHANGED TO 6, THE CPU LEVEL WOULD
;BE 1 LESS (5) & HENCE THIS WORD
;SHOULD BE 240 (BIT POSITIONS ARE
;IDENTICAL TO THE PRIORITY BITS IN PSW)
;CONTAINS THE NORMAL VECTOR ADDRESS
;TO WHICH THE RK11 INTERRUPTS. IF THE
;VECTOR ADDRESS HAS BEEN CHANGED, MODIFY
;THIS WORD.

;GENERAL INDEX REGISTERS

;GENERAL REGISTERS
;GENERAL REGISTERS
;GENERAL REGISTERS

;*THE FOLLOWING TABLE CONTAINS THE CYLINDERS BETWEEN WHICH THE SEEKS WILL BE
;*TIMED. THEY HAVE BEEN SELECTED TO GIVE SOME TYPICAL SEEKS TIMES FOR THE
;*3 SEEK SPEEDS. IF FOR ANY REASON YOU WANT TO TIME SEEKS BETWEEN ANY
;*OTHER SET OF CYLINDERS, MAKE CHANGES IN THE CORRESPONDING SEEK CYLINDER
;*ADDRESSES.

;*OUTER CYLINDER ADDRESS, FROM WHERE SEEK WILL BE DONE

SOAD:	0	;CYLINDER 0
	0	" 0
	13140	" 179
	0	" 0
	0	" 0
	0	" 0
	0	" 0

;*INNER ADDRESS, TO WHICH SEEK WILL BE DONE

SIAD:	14500	;CYLINDER 202, LAST
	40	" 1
	13240	" 181
	140	" 3
	1000	" 16
	2000	" 32

1229 001556 006200

6200

; " 100

1230

1231

1232

1233

1234

1235

1236

1237

1238

1239

1240

1241

1242

1243

1244

1245

1246

1247

1248

1249

1250

1251

1252

1253

1254

1255

1256

1257

1258

1259

1260

1261

1262

1263

1264

1265

1266

1267

1268

1269

1270

1271

1272

1273

1274

1275

1276

1277

1278

1279

1280

1281

1282

1283

1284

001560 004262

001562 004626

001564 005124

001566 005614

001570 006622

001572 007360

001574 010440

001576 012176

001600 012740

; FOLLOWING POINTERS ARE USED TO TRANSFER CONTROL TO THE
; TEST SELECTED BY USING SW 8. IF ANY MORE TESTS ARE
; ADDED TO THIS PROGRAM ADDITIONAL POINTERS SHOULD BE INSERTED.

PT1: TST1+2

PT2: TST2+2

PT3: TST3+2

PT4: TST4+2

PT5: TST5+2

PT6: TST6+2

PT7: TST7+2

PT10: TST10+2

PT11: TST11+2

; MESSAGES & ASCII STRINGS

MSG1: .ASCIIZ <15><12>/SIN/

MSG2: .ASCIIZ <15><12>/SKE/

MSG3: .ASCIIZ <15><12>/TEST # ABORTED:/

MSG4: .ASCIIZ <15><12>/PROG ABORTED/

MSG5: .ASCIIZ <15><12>/READ HDRS OK FROM CYLB ABOVE/

MSG6: .ASCIIZ /EXPCTD HDR= /

MSG7: .ASCIIZ / PC= /

MSG10: .ASCIIZ <15><12>/CNTRL RDY DIDN'T SET/

MSG11: .ASCIIZ /SECTR EXPC P-HDR RECV P-HDR/

B03

MD-11-DZRL-E, RK11-RK05 DYNAMIC TEST
DZRL-E.P11 26-APR-77 12:27

MACY11 30(1046) 14-JUL-77 08:03 PAGE 27
COMMON TAGS

1285					
1286	002027	015	051012	053457	MSG12: .ASCIZ <15><12>"R/W/S RDY NOT SET"
1287	002034	051457	051040	054504	
1288	002042	047040	052117	051440	
1289	002050	052105	000		
1290					
1291	002053	040	052040	054522	MSG13: .ASCIZ / TRY #: /
1292	002060	021440	000072		
1293					
1294					
1295	002064	005015	051104	053111	MSG14: .ASCIZ <15><12>/DRIVE /
1296	002072	020105	000		
1297					
1298	002075	040	020040		BLNK13: .ASCII / / /
1299	002100	040			BLNK10: .ASCII / /
1300	002101	040			BLNK59: .ASCII / /
1301	002102	040			BLNK58: .ASCII / /
1302	002103	040			BLNK57: .ASCII / /
1303	002104	040			BLNK56: .ASCII / /
1304	002105	040			BLNK55: .ASCII / /
1305	002106	040			BLNK54: .ASCII / /
1306	002107	040			BLNK53: .ASCII / /
1307	002110	040			BLNK52: .ASCII / /
1308	002111	040	000		BLNK51: .ASCIZ / /
1309					
1310		002114			.EVEN
1311	002114	000000			FDRIVE: 0
1312	002116	000000			FDRIVE1: 0
1313	002120	000000			DRHOLD: 0

.SBTTL ERROR POINTER TABLE

;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
 ;*LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
 ;*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).
 ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

;* EM ::POINTS TO THE ERROR MESSAGE
 ;* DH ::POINTS TO THE DATA HEADER
 ;* DT ::POINTS TO THE DATA
 ;* DF ::POINTS TO THE DATA FORMAT

\$ERRTB:

:ERROR ITEMS TABLE

:ITEM

1

EM1 ;CNTRL RDY DIDN'T SET AFTER SEEK
 DH1 ;PC RKCS RKER RKDS RKDA
 DT1 ;\$ERRPC \$REGO \$REG1 \$REG2 \$REG3
 0

:ITEM

2

EM2 ;SIN ON SEEK
 DH1 ;PC RKCS RKER RKDS RKDA
 DT1 ;\$ERRPC \$REGO \$REG1 \$REG2 \$REG3
 0

:ITEM

3

EM3 ;DRE ON SEEK
 DH1 ;PC RKCS RKER RKDS RKDA
 DT1 ;\$ERRPC \$REGO \$REG1 \$REG2 \$REG3
 0

:ITEM

4

EM4 ;'ERR' ON SEEK
 DH1 ;PC RKCS RKER RKDS RKDA
 DT1 ;\$ERRPC \$REGO \$REG1 \$REG2 \$REG3
 0

:ITEM

5

EM5 ;'DRU' ON SEEK; PUT DRIVE ON 'LOAD' BACK TO 'RUN'
 DH1 ;PC RKCS RKER RKDS RKDA
 DT1 ;\$ERRPC \$REGO \$REG1 \$REG2 \$REG3
 0

1314
 1315
 1316
 1317
 1318
 1319
 1320
 1321
 1322
 1323
 1324
 1325
 1326
 1327
 1328 002122
 1329
 1330
 1331
 1332
 1333
 1334
 1335
 1336
 1337 002122 024334
 1338 002124 025500
 1339 002126 026334
 1340 002130 000000
 1341
 1342
 1343
 1344 002132 024373
 1345 002134 025500
 1346 002136 026334
 1347 002140 000000
 1348
 1349
 1350
 1351 002142 024407
 1352 002144 025500
 1353 002146 026334
 1354 002150 000000
 1355
 1356
 1357
 1358 002152 024423
 1359 002154 025500
 1360 002156 026334
 1361 002160 000000
 1362
 1363
 1364
 1365 002162 024441
 1366 002164 025500
 1367 002166 026334
 1368 002170 000000
 1369

1370			:ITEM 6						
1371				EM6	:R/W/S RDY NOT SET AFTER SEEK				
1372	002172	024510		DH1	:PC RKCS RKER RKDS RKDA				
1373	002174	025500		DT1	:SERRPC \$REG0 \$REG1 \$REG2 \$REG3				
1374	002176	026334		0					
1375	002200	000000							
1376			:ITEM 7						
1377				EM7	:SIN ON WRITE FMT				
1378				DH7	:PC RKCS RKER RKDS RKDA CYLINDER				
1379	002202	024545		DT7	:SERRPC \$REG0 \$REG1 \$REG2 \$REG3 \$REG4				
1380	002204	025576		0					
1381	002206	026350							
1382	002210	000000							
1383			:ITEM 10						
1384				EM10	: 'ERR' ON DOING WRITE FMT				
1385				DH7	:PC RKCS RKER RKDS RKDA CYLINDER				
1386	002212	024564		DT7	:SERRPC \$REG0 \$REG1 \$REG2 \$REG3 \$REG4				
1387	002214	025576		0					
1388	002216	026350							
1389	002220	000000							
1390			:ITEM 11						
1391				EM11	:SIN ON READ FMT				
1392				DH11	:PC RKCS RKER RKDS RKDA:				
1393	002222	024615		DT11	:DRV# CYL SUR SEC				
1394	002224	025653		0	:SERRPC \$REG0 \$REG1 \$REG2 \$REG3 \$REG4 \$REG5 \$REG6 \$REG7				
1395									
1396	002226	026366							
1397									
1398	002230	000000							
1399			:ITEM 12						
1400				EM12	: 'ERR' ON READ FMT				
1401				DH7	:PC RKCS RKER RKDS RKDA CYLINDER				
1402	002232	024633		DT7	:SERRPC \$REG0 \$REG1 REG2 \$REG3 \$REG4				
1403	002234	025576		0					
1404	002236	026350							
1405	002240	000000							
1406			:ITEM 13						
1407				EM13	:WRONG HEADERS FROM 'SEC #				
1408				DH13	:SECTOR # HEADER RECVD				
1409	002242	024655		0					
1410	002244	025750		ESR13	:USE THIS SUBROUTINE FOR TYPING OUT ERROR DATA				
1411	002246	000000							
1412	002250	015456							
1413			:ITEM 14						
1414				EM14	:ERROR ON IMPLIED SEEK FROM CYLA TO CYLB				
1415				DH14	:PC CYLA CYLB RKER RKDS TRY#				
1416	002252	024704		DT7	:SERRPC \$REG0 \$REG1 \$REG2 \$REG3 \$REG4				
1417	002254	025767		0					
1418	002256	026350							
1419	002260	000000							
1420			:ITEM 15						
1421				MS15	:READ WRONG HDRS FROM CYLB ABOVE				
1422				DH13	:SEC# HEADER RECVD				
1423	002262	024757		0					
1424	002264	025750							
1425	002266	000000							

1426	002270	015364	ESR15	;GO TO "ESR15" FOR TYPING OUT					
1427									
1428			:ITEM	16					
1429									
1430	002272	025021	EM16	;READ WRONG FIRST WORD FROM SECTOR 0, 'CYLB' ON IMPLIED SEEK FROM CYLA					
1431	002274	026046	DH16	;PC	CYLA	CYLB	EXPCT	RECVD	TRY#
1432	002276	026350	DT7	;SERRPC	\$REG0	\$REG1	\$REG2	\$REG3	\$REG4
1433	002300	000000	0						
1434									
1435			:ITEM	17					
1436									
1437	002302	025126	MS17	;READ FIRST WORD FROM SECTOR 1, 'CYLB' ABOVE					
1438	002304	026123	DH17	;PC	CYLB	EXPCT	RECVD		
1439	002306	026410	DT17	;SERRPC	\$REG0	\$REG1	\$REG2		
1440	002310	000000	0						
1441									
1442			:ITEM	20					
1443									
1444	002312	025174	EM20	;READ WRONG HEADER ON IMPLIED SEEK FROM 'CYLA' TO 'CYLB'					
1445	002314	025750	DH13	;SECTOR # HEADER RECVD					
1446	002316	000000	0						
1447	002320	015532	ESR20	;USE THIS SUBROUTINE FOR TYPING OUT ERROR DATA					
1448									
1449									
1450			:ITEM	21					
1451									
1452	002322	025262	EM21	;ERROR ON DOING WRITE ON DSK					
1453	002324	025500	DH1	;PC	RKCS	RKER	RKDS	RKDA	
1454	002326	026334	DT1	;SERRPC	\$REG0	\$REG1	\$REG2	\$REG3	
1455	002330	000000	0						
1456									
1457			:ITEM	22					
1458									
1459	002332	025316	EM22	;SIN ON DOING WRITE					
1460	002334	025500	DH1	;PC	RKCS	RKER	RKDS	RKDA	
1461	002336	026334	DT1	;SERRPC	\$REG0	\$REG1	\$REG2	\$REG3	
1462	002340	000000	0						
1463									
1464			:ITEM	23					
1465									
1466	002342	025341	EM23	;HE ON DOING READ					
1467	002344	025653	DH11	;PC	RKCS	RKER	RKDS	RKDA:	
1468				;DRV#	CYL	SUR	SEC		
1469	002346	026366	DT11	;SERRPC	\$REG0	\$REG1	\$REG2		
1470				;SREG4	\$REG5	\$REG6	\$REG7		
1471	002350	000000	0						
1472									
1473			:ITEM	24					
1474									
1475	002352	025362	EM24	;CSE ON READ					
1476	002354	026161	DH24	;PC	TRY#	RKCS	RKER	RKDS	RKDA:
1477				;DRV#	CYL	SUR	SEC		
1478	002356	026422	DT24	;SERRPC	\$REG10	\$REG0	\$REG1	\$REG2	
1479				;SREG4	\$REG5	\$REG6	\$REG7		
1480	002360	000000	0						

[illegible]

:THIS IS THE HANDLER FOR UNEXPECTED TIME OUT. PRESSING CONTINUE WILL
:RESTART THE PROGRAM.

1519									
1520									
1521									
1522									
1523									
1524	002422	011600				BADTMO:	MOV	(SP),R0	:SAVE PC WHERE TIME OUT OCCURED
1525	002424	005740					TST	-(R0)	
1526	002426	022626					CMP	(SP)+,(SP)+	:RESTORE STACK POINTER
1527	002430	104401	002436				TYPE	65\$	:TYPE ASCIZ STRING
1528	002434	000407					BR	64\$	:GET OVER THE ASCIZ
1529						65\$:	ASCIZ	<15><12>/TIMOUT:PC=	
1530	002454					64\$:			
1531	002454	010046					MOV	R0,-(SP)	:SET UP FOR TYPING OUT PC
1532	002456	104402					TYPOC		:GO TYPE OUT OCTAL PC
1533	002460	000000					HALT		
1534									
1535	002462	000005				START:	RESET		:CLEAR THE BUS
1536						::GIVE	DRIVES	TIME TO RELOAD HEADS IN CASE OF AN APT START	
1537	002464	023737	000042	000046			CMP	2042,2046	:ARE WE IN ACT11 AUTO MODE?
1538	002472	001016					BNE	STARTA	:NO, SKIP DELAY
1539	002474	005077	176764				CLR	20KDA	:SELECT UNIT 0
1540	002500	012700	000250				MOV	250,R0	:WAIT FOR..
1541	002504	032777	000200	176740	20\$:		BIT	200,20KDS	:DRIVE READY..
1542	002512	001006					BNE	STARTA	:IN CASE..
1543	002514	005001					CLR	R1	:OF APT..
1544	002516	005301					DEC	R1	:START, BUT..
1545	002520	001376					BNE	-2	:DON'T WAIT..
1546	002522	005300					DEC	R0	:FOREVER.
1547	002524	001367					BNE	20\$	
1548	002526	000000					HALT		:RKDS BIT 7 (DIRVE READY) NEVER SET
1549	002530					STARTA:			
1550						.SBTTL	INITIALIZE THE COMMON TAGS		
1551						::CLEAR	THE COMMON TAGS (\$CMTAG) AREA		
1552	002530	012706	001100				MOV	2\$CMTAG,R6	:FIRST LOCATION TO BE CLEARED
1553	002534	005026					CLR	(R6)+	:CLEAR MEMORY LOCATION
1554	002536	022706	001140				CMP	2\$SWR,R6	:; DONE?
1555	002542	001374					BNE	-6	:; LOOP BACK IF NO
1556	002544	012706	001100				MOV	2\$STACK,SP	:; SETUP THE STACK POINTER
1557						::INITIALIZE	A FEW VECTORS		
1558	002550	012737	017006	000020			MOV	2\$SCOPE,2\$IOTVEC	:; IOT VECTOR FOR SCOPE ROUTINE
1559	002556	012737	000340	000022			MOV	2340,2\$IOTVEC+2	:; LEVEL 7
1560	002564	012737	017162	000030			MOV	2\$ERRROR,2\$EMTVEC	:; EMT VECTOR FOR ERROR ROUTINE
1561	002572	012737	000340	000032			MOV	2340,2\$EMTVEC+2	:; LEVEL 7
1562	002600	012737	022702	000034			MOV	2\$TRAP,2\$TRAPVEC	:; TRAP VECTOR FOR TRAP CALLS
1563	002606	012737	000340	000036			MOV	2340,2\$TRAPVEC+2	:; LEVEL 7
1564	002614	012737	023010	000024			MOV	2\$SPWRDN,2\$PWAVEC	:; POWER FAILURE VECTOR
1565	002622	012737	000340	000026			MOV	2340,2\$PWAVEC+2	:; LEVEL 7
1566	002630	005037	001204				CLR	2\$ESCAPE	:; CLEAR THE ESCAPE ON ERROR ADDRESS
1567	002634	112737	000001	001115			MOVB	21,2\$ERRMAX	:; ALLOW ONE ERROR PER TEST
1568	002642	012737	002642	001106			MOV	2\$,2\$SLPADR	:; INITIALIZE THE LOOP ADDRESS FOR SCOPE
1569	002650	012737	002650	001110			MOV	2\$,2\$SLPERR	:; SETUP THE ERROR LOOP ADDRESS
1570						::SIZE	FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR I* IS		
1571						::EQUAL	TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.		
1572	002656	013746	000004				MOV	2\$ERRVEC,-(SP)	:; SAVE ERROR VECTOR
1573	002662	012737	002716	000004			MOV	264\$,2\$ERRVEC	:; SET UP ERROR VECTOR
1574	002670	012737	177570	001140			MOV	2\$DSWR,2\$SWR	:; SETUP FOR A HARDWARE SWICH REGISTER

H03

MD-11-DZRL-E, RK11-RK05 DYNAMIC TEST
DZRL-E.P11 26-APR-77 12:27MACY11 30(1046) 14-JUL-77 08:03 PAGE 33
INITIALIZE THE COMMON TAGS

```

1575 002676 012737 177570 001142      MOV      #DDISP,DISPLAY    ;;AND A HARDWARE DISPLAY REGISTER
1576 002704 022777 177777 176226      CMP      #-1,SWR        ;;TRY TO REFERENCE HARDWARE SWR
1577 002712 001012                BNE      66$          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1578                                ;;AND THE HARDWARE SWR IS NOT = -1
1579 002714 000403                BR       65$          ;;BRANCH IF NO TIMEOUT
1580 002716 012716 002724      64$: MOV      #65$, (SP)      ;;SET UP FOR TRAP RETURN
1581 002722 000002                RTI                     ;;
1582 002724 012737 000176 001140      65$: MOV      #SWREG,SWR    ;;POINT TO SOFTWARE SWR
1583 002732 012737 000174 001142      MOV      #DISPREG,DISPLAY
1584 002740 012637 000004      66$: MOV      (SP)+,SWERRVEC    ;;RESTORE ERROR VECTOR
1585
1586 002744 004737 020622                JSR      PC,STKINT      ;;INITIALIZE THE TTY HANDLER
1587 002750 023737 000042 000046      CMP      #42,SWR        ;;ARE WE IN ACT11 AUTO MODE?
1588 002756 001416                BEQ      69$          ;;YES, SKIP TITLE
1589
1590                                .SBTTL   TYPE PROGRAM NAME
1591                                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1592 002760 005227 177777                INC      #-1          ;;FIRST TIME?
1593 002764 001043                BNE      67$          ;;BRANCH IF NO
1594 002766 104401 003024                TYPE     68$          ;;TYPE ASCIZ STRING
1595                                .SBTTL   GET VALUE FOR SOFTWARE SWITCH REGISTER
1596 002772 005737 000042                TST      SWR        ;;ARE WE RUNNING UNDER XXDP/ACT?
1597 002776 001006                BNE      69$          ;;BRANCH IF YES
1598 003000 023727 001140 000176      CMP      SWR,#SWREG    ;;SOFTWARE SWITCH REG SELECTED?
1599 003006 001005                BNE      70$          ;;BRANCH IF NO
1600 003010 104406                GTSWR          ;;GET SOFT-SWR SETTINGS
1601 003012 000403                BR       70$
1602 003014 112737 000001 001134      69$: MOV      #1,SAUTOB    ;;SET AUTO-MODE INDICATOR
1603 003022 000424                70$: BR       67$
1604                                ;;68$: .ASCIZ <CRLF>/RK11 DYNAMIC TEST/1512/MAINDEC-11-DZRL-E/1CRLF
1605 003074                                67$: START1: TSTB    FFUNC      ;;FUNCTION PROGRAM SELECTED?
1606 003074 105737 001216                BEQ      7$          ;;NO
1607 003100 001404                CLRB     FFUNC      ;;YES, CLEAR THE FLAG
1608 003102 105037 001216                JMP      #FUNBEG    ;;GO TO 'FUNCTION SELECTION PROGRAM'
1609 003106 000137 023172                MOV      #XXDPMO,RO    ;;CLEAR FLAGS FROM
1610 003112 012700 001220      7$: CLRB      (RO)+      ;;'XXDPMO' TO 'DRIVAD'
1611 003116 105020                CMP      RO,#DRIVAD+2
1612 003120 020027 001232                BNE      5$
1613 003124 001374                MOV      #-10,R1
1614 003126 012701 177770      6$: BIC      #3,(RO)+    ;;CLEAR BIT 0'S IN 'DRIVE
1615 003132 042720 000003                INC      R1          ;;PRESENT' FLAGS.
1616 003136 005201                BNE      6$
1617 003140 001374
1618
1619                                ;THE FOLLOWING CODE FINDS OUT THE PROGRAM CONTROL MODE:
1620                                ;PAPER TAPE (MANUAL), ACT11, RKDP CHAIN OR DUMP
1621
1622 003142 122737 000002 000041      CMPB     #2,41      ;;LOADED FROM AN RK05 ?
1623 003150 001160                BNE      ST2          ;;BR IF NOT
1624 003152 013737 000040 001220      MOV      40,XXDPMO    ;;GET DEVICE INDICATOR AND DRIVE ADDRESS OF
1625                                ;;LOADING RK05
1626 003160 122737 000010 001220      CMPB     #10,XXDPMO    ;;DRIVE ADDRESS 7 OR LESS ?
1627 003166 101002                BHI      2$          ;;BR IF YES
1628 003170 105037 001220                CLRB     XXDPMO    ;;DRIVE ZERO LOADED THE PROGRAM
1629 003174 005737 000042      2$: TST      42          ;;CHAIN MODE OR ACT11 AUTO ACCEPT ?
1630 003200 001424                BEQ      3$          ;;BR IF NEITHER

```

```

1631 003202 104401 003210
1632 003206 000413
1633
1634 003236
1635 003236 005046
1636 003240 113716 001220
1637 003244 104403
1638 003246 001
1639 003247 000
1640 003250 000520
1641 003252 005227 177777
1642 003256 001115
1643 003260 104401 003266
1644 003264 000411
1645
1646 003310
1647 003310 005046
1648 003312 113716 001220
1649 003316 104403
1650 003320 001
1651 003321 000
1652 003322 104401 003330
1653 003326 000431
1654
1655 003412
1656 003412 104401 003420
1657 003416 000435
1658
1659 003512
1660
1661
1662 003512 012737 002422 000004 ST2: MOV #BADTMO,ERRVEC ;SET TIMEOUT VECTOR FOR
1663 ;UNEXPECTED TIME OUT
1664
1665 ;THIS CODE FINDS WHICH DRIVES ARE PRESENT & PRINTS OUT THE DRIVE
1666 ;DRIVE NUMBERS THAT WERE FOUND ON LINE.
1667
1668 003520 104401 003526
1669 003524 000411
1670
1671 003550
1672 003550 105037 001224
1673 003554 005001
1674 003556 012702 001232
1675 003562 005003
1676 003564 005737 001220
1677 003570 001403
1678 003572 120337 001220
1679 003576 001411
1680 003600 010177 175660
1681 003604 105777 175642
1682 003610 100004
1683 003612 105237 001224
1684 003616 052712 000001
1685 003622 005722
1686 003624 005203

        TYPE 65$          ;;TYPE ASCIZ STRING
        BR 64$          ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <15><12>/NOT TESTING DRIVE /
64$:
        CLR -(SP)        ;CLEAR WORD ON STACK
        MOVB XXDPMO,(SP) ;GET DRIVE ADDRESS
        TYP0S           ;TYPE THE ADDRESS
        .BYTE 1          ;ONLY 1 CHARACTER
        .BYTE 0          ;SUPPRESS LEADING ZEROS
        BR ST2           ;GET NUMBER OF DRIVES
3$: INC #1              ;FIRST TIME THROUGH HERE
        BNE ST2          ;BR IF NOT FIRST TIME
        TYPE 67$        ;;TYPE ASCIZ STRING
        BR 66$          ;;GET OVER THE ASCIZ
;;67$: .ASCIZ <15><12>/TO TEST DRIVE /
66$:
        CLR -(SP)        ;CLEAR WORD ON THE STACK
        MOVB XXDPMO,(SP) ;GET DRIVE ADDRESS
        TYP0S           ;TYPE THE DRIVE ADDRESS
        .BYTE 1          ;ONLY 1 CHARACTER
        .BYTE 0          ;SUPPRESS LEADING ZEROS
        TYPE 69$        ;;TYPE ASCIZ STRING
        BR 68$          ;;GET OVER THE ASCIZ
;;69$: .ASCIZ / HALT PROGRAM, REMOVE RKDP PACK AND REPLACE IT /<15> 12/
68$:
        TYPE 71$        ;;TYPE ASCIZ STRING
        BR 70$          ;;GET OVER THE ASCIZ
;;71$: .ASCIZ /WITH A WORK PACK, CLEAR LOCATION 40, AND RESTART PROGRAM
70$:

ST2: MOV #BADTMO,ERRVEC ;SET TIMEOUT VECTOR FOR
        ;UNEXPECTED TIME OUT

;THIS CODE FINDS WHICH DRIVES ARE PRESENT & PRINTS OUT THE DRIVE
;DRIVE NUMBERS THAT WERE FOUND ON LINE.

        TYPE 65$          ;;TYPE ASCIZ STRING
        BR 64$          ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <15><12>/DRIVES PRESENT/
64$:
        CLPA DRVS        ;INITIALIZE NO. OF DRVS PRESENT
        CLR R1
        MOV #DRIVO,R2
        CLR R3
1$: TST XXDPMO           ;INITIALIZE COUNT TO 0
        BEQ 6$          ;LOADED FROM AN RK05 ?
        BEQ 6$          ;BR IF NOT
        CMPB R3,XXDPMO   ;CHECKING THE LOAD DRIVE ?
        BEQ 2$          ;BR IF YES
6$: MOV R1,2RKDA         ;ADRES A DRIVE
        TSTB 2RKDS       ;IS IT PRESENT?
        BPL 2$          ;NO, BRANCH
        INCB DRVS        ;INCREMENT TOTAL # OF DRVS
        BIS #1,(R2)      ;SET FLAG INDICATING THIS DRV PRSNT
2$: TST (R2)+
        INC R3           ;INCREMENT COUNT

```

```

1687 003626 062701 020000      ADD      #20000,R1      ;ADRES THE NXT DRV
1688                                ;CHKD ALL 8 DRIVES?
1689 003632 001354      BNE      1$      ;IF NOT, GO CHK IF NEXT DRV PRSN*
1690                                ;FIND WHICH ARE FS
1691 003634 004737 024250      JSR      PC,SIZEF      ;WERE ANY DRIVES FOUND?
1692 003640 105737 001224      TSTB     DRVS      ;YES, BRANCH
1693 003644 001010      BNE      3$      ;TYPE ASCIZ STRING
1694 003646 104401 003654      TYPE     67$      ;GET OVER THE ASCIZ
1695 003652 000403      BR        66$
1696                                ;:67$: .ASCIZ / NONE/
1697 003662                                66$:
1698 003662 000137 015246      JMP      $EOP      ;IF NONE WERE FOUND, GO
1699                                ;TO THE END OF PROGRAM
1700 003666 005002      3$:      CLR      R2      ;DRIVE NUMBER
1701 003670 012700 001232      MOV      #DRIVO,R0      ;TABLE OF AVAIL DRIVES
1702 003674 105710      5$:      TSTB     (R0)      ;DRIVE HERE?
1703 003676 001414      BEQ      4$      ;NO
1704 003700 104401      TYPE     $CRLF
1705 003702 001213      MOV      R2,-(SP)      ;PUSH NO ON THE STACK
1706 003704 010246      TYPOS      ;TO TYPE OCTAL NO.
1707 003706 104403      .BYTE     1      ;TYPE 1 DIGIT, SUPRESS LDG 0'S
1708 003710 001      .BYTE     0
1709 003711 000      BIT      #2,(R0)      ;IS IT RK05F?
1710 003712 032710 000002      BEQ      4$      ;NO
1711 003716 001404      TYPE     69$      ;TYPE ASCIZ STRING
1712 003720 104401 003726      BR        68$      ;GET OVER THE ASCIZ
1713 003724 000401      69$:      .ASCIZ  /F/
1714                                68$:
1715 003730      4$:      INC      R2      ;POINT TO NEXT DRIVE #
1716 003730 005202      TST      (R0)+      ;NEXT DRIVE IN TABLE
1717 003732 005720      CMP      R0,#DRIV7+1      ;ALL DONE?
1718 003734 020027 001251      BLT      5$      ;NO, CHECK REST
1719 003740 002755
1720
1721      ;FIND OUT THE FIRST (FROM 0-7) DRIVE THAT IS PRESENT. PUT THE ADDRESS
1722      ;OF THAT DRIVE IN 'DRIVAD'. INDICATE THAT DRIVE # WILL
1723      ;BE TESTED.
1724
1725 003742 012737 001232 001226 ST3:  MOV      #DRIVO,DRVPTR
1726 003750 105037 001223      CLRB     DRVON
1727 003754 005037 001230      CLR      DRIVAD
1728 003760 105037 001102      NXTDRV: CLRB     $STNM      ;RESET TEST NUMBER TO 1
1729 003764 005037 001112      CLR      $ERTL      ;CLEAR ERROR COUNT FOR THIS DRIVE
1730 003770 013701 001226      MOV      DRVPTR,R1
1731 003774 032721 000001      1$:      BIT      #1,(R1)+      ;IS THIS DRIVE PRESENT?
1732 004000 001005      BNE      2$      ;YES, BRANCH
1733 004002 020127 001252      4$:      CMP      R1,#DRIV7+2      ;CHECKED THE WHOLE LIST?
1734 004006 001372      BNE      1$      ;NO
1735 004010 000137 015246      JMP      $EOP      ;YES, EXIT
1736 004014 010137 001226      2$:      MOV      R1,DRVPTR      ;NO, GO AHEAD
1737 004020 014104      MOV      -(R1),R4      ;GET DRIVE NO. TO BE TESTED
1738 004022 005037 002116      CLR      FDRVE1
1739 004026 005037 002114      CLR      FDRIVE
1740 004032 032704 000002      BIT      #2,R4      ;SHOWS F IF -1
1741 004036 001410      BEQ      7$      ;RK-05F?
1742 004040 005237 002116      INC      FDRVE1      ;NO
;SHOWS F

```

K03

MC-11-DZRL-E, RK11-RK05 DYNAMIC TEST
DZRL-E.P11 26-APR-77 12:27MACY11 30(1046) 14-JUL-77 08:03 PAGE 36
GET VALUE FOR SOFTWARE SWITCH REGISTER

1743	004044	032704	020000		BIT	#20000,R4	;EVEN DRIVE?
1744	004050	001003			BNE	75	;NO
1745	004052	012737	177777	002114	MOV	#-1,FDRIVE	;RK05F AND EVEN
1746	004060	042704	000003		BIC	#3,R4	
1747	004064	010437	001230		MOV	R4,DRIVAD	;SET UP DRIVE ADRES
1748	004070	104401	002064		TYPE	,MSG14	
1749	004074	000241			CLC		
1750	004076	006104			ROL	R4	;TYPE OUT THE DRIVE NO.
1751	004100	006104			ROL	R4	
1752	004102	006104			ROL	R4	
1753	004104	006104			ROL	R4	
1754	004106	010446			MOV	R4,-(SP)	
1755	004110	104403			TYP05		
1756	004112	001			.BYTE	1	
1757	004113	000			.BYTE	0	
1758							
1759	004114	005737	002116		TST	FDRVE1	;RK-05F?
1760	004120	001404			BEG	65	;NO
1761	004122	104401	004130		TYPE	655	;TYPE ASCIZ STRING
1762	004126	000401			BR	645	;GET OVER THE ASCIZ
1763					;;655:	.ASCIZ /F/	
1764	004132				645:		
1765	004132				65:		
1766					;;IF SW 8 IS SET THEN FIND OUT WHICH TEST NUMBER IS		
1767					;;SELECTED AND JUMP TO THAT TEST.		
1768							
1769	004132	105037	001222		CLRB	LUPSW	;CLEAR FLAG INDICATING SW8 SET
1770	004136	032777	000400	174774	BIT	#SW8,2SWP	;SW 8 SET?
1771	004144	001445			BEG	TST1	;NO. BRANCH
1772	004146				55:		
1773	004146	104401	004154		TYPE	675	;TYPE ASCIZ STRING
1774	004152	000410			BR	665	;GET OVER THE ASCIZ
1775					;;675:	.ASCIZ <15><12>/OCTAL TEST#?/	
1776	004174				665:		
1777	004174	104412			RDOCT		
1778	004176	012600			MOV	(SP)+,R0	
1779	004200	001762			BFG	55	
1780	004202	020027	000011		CMP	R0,#11	;CHECK TYPED IN TEST #
1781	004206	003357			BGT	55	;IS LEGAL. IF NOT ASK
1782	004210	110037	001102		MOVB	R0,\$TSTNM	
1783	004214	005300			DEC	R0	;FORM POINTERS FOR THE TEST #
1784	004216	006300			ASL	R0	
1785	004220	016037	001560	001106	MOV	PT1(R0),\$LPADR	;ADJUST POINTERS FOR SCOPE
1786	004226	013737	001106	001110	MOV	\$LPADR,\$LPERR	;LOOP, ETC.
1787	004234	105237	001222		INCB	LUPSW	;SET FLAG INDICATING TEST #
1788							;SELECTED
1789	004240	000177	174642		JMP	2\$LPADR	;GO TO THE TEST SELECTED
1790							
1791							
1792							
1793							
1794							
1795							
1796							
1797	004244	005000			PWRFL:	CLR R0	
1798	004246	005001				CLR R1	

;ON RECOVERY FROM POWER FALIURE RETURN HERE

1799	004250	005201			15:	INC	R1	
1800	004252	001376				BNE	15	
1801	004254	105200				INCB	R0	
1802	004256	001374				BNE	15	
1803								
1804								
1805								
1806								
1807								
1808								
1809								
1810								
1811								
1812								
1813								
1814	004260	000004						
1815	004262	005000						
1816	004264	005001						
1817	004266	005002						
1818	004270	012737	004322	001110				
1819								
1820	004276	012703	001266					
1821	004302	012704	177767					
1822	004306	012705	177770					
1823	004312	000402						
1824								
1825	004314	005703			15:	TST	R3	
1826	004316	001403				BEQ	35	
1827	004320	005003			25:	CLR	R3	
1828	004322	104415			205:	CON.RESET		
1829								
1830	004324	104416				DRV.RESET		
1831								
1832	004326	013777	001230	175130	35:	MOV	DRIVAD, DRKDA	
1833	004334	050277	175124			BIS	R2, DRKDA	
1834	004340	105777	175106			TSTB	DRKDS	
1835	004344	100406				BMI	215	
1836	004346	004737	016162			JSR	PC, GT4RG	
1837	004352	104030				ERROR	30	
1838								
1839	004354	005203				INC	R3	
1840	004356	005205				INC	R5	
1841	004360	001515				BEQ	185	
1842								
1843	004362	012777	000011	175066	215:	MOV	#11, DRKCS	
1844	004370	005200			45:	INC	R0	
1845	004372	001007				BNE	55	
1846								
1847	004374	004737	016162			JSR	PC, GT4RG	
1848	004400	104001				ERROR	1	
1849								
1850								
1851	004402	005203				INC	R3	
1852	004404	005204				INC	R4	
1853	004406	001502				BEQ	185	
1854	004410	000403				BR	65	

 :TEST 1 CHECK INNER LIMIT SWITCH & ELECTROMECHANICAL INTEGRITY
 :THIS TEST PERFORMS 200(8) PURE SEEKS FROM CYLINDER 0
 :TO CYLINDER 312 AND BACK TO 0. IT IS SUPPOSED TO
 :CHECK THE INNER LIMIT SWITCH AND THE MECHANICAL
 :INTEGRITY OF THE DRIVE. NOTE THAT A VISUAL CHECK FOR
 :ANY MECHANICAL FAILURES WOULD BE VERY HELPFUL.

 :ST1: SCOPE
 :CLR R0 :INITIALIZE COUNT
 :CLR R1 :INITIALIZE COUNT FOR # OF SEEKS
 :CLR R2 :CONTAINS SEEK ADRES
 :MOV #205, \$LPERR :SET RETURN ADRES FOR LUPING
 :ON ERROR
 :MOV #BUFR, R3 :INITIALIZE POINTER TO THE TABLE
 :MOV #-11, R4 :ALLOW ONLY 9 CNTRL-RDY, SIN, OR R/W/S RDY ERRORS
 :MOV #-10, R5 :ALLOW ONLY 8 DRU+DRE+ERR+DRY ERRORS
 :BR 25
 :
 :15: TST R3 :WAS THERE ANY ERROR?
 :BEQ 35 :NO, BRANCH
 :25: CLR R3 :CLR EROR FLAG
 :205: CON.RESET :GO DO CNTRL RESET. SUB ROUTINE
 : :AT 'CNT.RST'
 : :GO TO 'DRV.RST' & DO DRV RESET
 :
 :35: MOV DRIVAD, DRKDA :ADRES THE DRIVE
 :BIS R2, DRKDA :SET SEEK ADRES
 :TSTB DRKDS :DRIVE RDY?
 :BMI 215 :YES
 :JSR PC, GT4RG :NO, GET RKCS, ER, DS, DA
 :ERROR 30 :DRIVE RDY BIT IS NOT SET
 : :IN RKDS
 : :SET ERROR FLAG
 : :ALLOW ONLY 5 ERRORS, IF MORE
 : :ABORT
 :
 :215: MOV #11, DRKCS :GO, SEEK
 :45: INC R0 :WAIT FOR CNTRL RDY
 :BNE 55 :WAITED LONG?
 : :IF YES, ERROR
 : :GO, GET RKCS, ER, DS, DA
 : :CNTRL RDY DIDN'T SET AFTER
 : :SEEK WAS DONE TO CYLINDER
 : :SHOWN IN RKDA. GO TO 'RK11 LOGIC TESTS'
 : :SET ERROR FLAG
 : :EXIT THIS TEST IF THERE R 5 OR MORE ERRORS

MD-11-DZRL-E, RK11-RK05 DYNAMIC TEST
DZRLKLE.P11 26-APR-77 12:27

MACY11 30(1046) 14-JUL-77 08:03 PAGE 38
T1 CHECK INNER LIMIT SWITCH & ELECTROMECHANICAL INTEGRITY

1855	004412	105777	175040	5\$:	TSTB	2RKCS	:DID CNTRL RDY SET?
1856	004416	100364			BPL	4\$	:IF NOT WAIT FOR IT
1857							
1858	004420	005000		6\$:	CLP	R0	:INITIALIZE COUNT
1859	004422	032777	000100 175022		BIT	#100,2RKDS	:R/W/S RDY SET?
1860	004430	001010			BNE	7\$	:YES
1861	004432	005200			INC	R0	:WAIT FOR R/W/S RDY
1862	004434	001372			BNE	6\$+2	
1863	004436	004737	016162		JSR	PC,GT4RG	:GET RKCS, ER, DS, DA
1864	004442	104006			ERROR	6	:R/W/S RDY DID NOT SET WHEN SEEK
1865							:WAS DONE TO CYLINDER INDICATED IN RKDA
1866	004444	005203			INC	R3	:SET ERROR FLAG
1867	004446	005204			INC	R4	:IF MAXM ERROR COUNT, ABORT
1868	004450	001461			BEQ	18\$	
1869	004452	032777	001000 174772	7\$:	BIT	#1000,2RKDS	:SIN ERROR?
1870	004460	001406			BEQ	8\$	:NO, BRANCH
1871	004462	004737	016162		JSR	PC,GT4RG	:GO, GET RKCS, ER, DS, DA
1872	004466	104002			ERROR	2	:SIN ERROR, ON DOING SEEK TO
1873							:CYL AS SHOWN IN RKDA
1874	004470	005203			INC	R3	:SET ERROR FLAG
1875	004472	005204			INC	R4	:IF MAXM ERROR COUNT REACHED,
1876	004474	001447			BEQ	18\$	:ABORT THE TEST
1877	004476	005777	174752	8\$:	TST	2RKER	:DRE ERROR?
1878	004502	100006			BPL	10\$	:NO, BRANCH
1879							
1880	004504	004737	016162		JSR	PC,GT4RG	:GO, GET RKCS, ER, DS, DA
1881	004510	104003			ERROR	3	:DRE ON DOING SEEK TO CYLINDER
1882							:AS SHOWN IN RKDA
1883	004512	005203			INC	R3	:SET ERROR FLAG
1884	004514	005205			INC	R5	:IF MAXM ERROR COUNT REACHED,
1885	004516	001767			BEQ	8\$	:ABORT THE TEST
1886							
1887	004520	005777	174732	10\$:	TST	2RKCS	:ERR' BIT IN RKCS SET?
1888	004524	100006			BPL	12\$	:NO, BRANCH
1889	004526	004737	016162		JSR	PC,GT4RG	:GO, GET RKCS, ER, DS, DA
1890	004532	104004			ERROR	4	:ERR' IN RKCS SET, ON DOING SEEK
1891							:TO CYL AS SHOWN IN RKDA. NOTE
1892							:WHICH BIT IN RKER SET?
1893	004534	005203			INC	R3	:SET ERROR FLAG
1894	004536	005205			INC	R5	:IF MAXM ERROR COUNT REACHED,
1895	004540	001425			BEQ	18\$	:ABORT THE TEST
1896							
1897	004542	032777	002000 174702	12\$:	BIT	#2000,2RKDS	:DRU SET?
1898	004550	001406			BEQ	15\$	:NO, BRANCH
1899	004552	004737	016162		JSR	PC,GT4RG	:GO, GET RKCS, ER, DS, DA
1900	004556	104005			ERROR	5	:DRU SET, THIS IS AN IRRECOVERABLE
1901							:ERROR. HENCE PUT THE DRIVE ON
1902							:LOAD, BACK TO RUN. DRU ERROR
1903							:SHOULD BE CLEARED, IF IT IS NOT
1904							:1) THE HEAD POSITION TRANSDUCER LAMP
1905							:IS INOPERATIVE
1906							:2) OR ERASE OR WRT CURRENT PRESENT
1907							:WITHOUT 'WRT GATE'
1908	004560	005203			INC	R3	:SET ERROR FLAG
1909	004562	005205			INC	R5	:ALLOW ONLY 5 ERRORS
1910	004564	001413			BEQ	18\$	:IF MORE THAN 5

MD-11-DZRKL-E, RK11-RK05 DYNAMIC TEST
DZRKLE.P11 26-APR-77 12:27

MACY11 30(1046) 14-JUL-77 08:03 PAGE 39
T1 CHECK INNER LIMIT SWITCH & ELECTROMECHANICAL INTEGRITY

```

1911                                     ;GO TO THE END OF THE PROGRAM
1912
1913 004566 005702          15$:  TST      R2          ;WAS SEEKING TO 0 OR 312?
1914 004570 001402          BEQ      16$          ;TO 0, BRANCH
1915                                     ;TO 312,
1916 004572 005002          CLR      R2          ;SEEK NXT TIME TO 0
1917 004574 000647          BR       1$          ;GO BAK & SK TO 0
1918
1919 004576 012702 014500    16$:  MOV      #14500,R2      ;SEEK NXT TIME TO 312
1920
1921 004602 005201          INC      R1          ;DONE SEEKS 200 TIMES?
1922 004604 022701 000200    CMP      #200,R1
1923
1924 004610 001241          BNE      1$          ;IF NOT, GO BAK
1925 004612 000404          BR       TST2        ;;EXIT
1926
1927
1928 004614 104401 001640    18$:  TYPE     MSG4
1929 004620 000137 015224    JMP      TST12
1930
1931 ;*****
1932 ;*TEST 2          FORMAT THE DISK
1933 ;*THIS PROGRAM ASSUMES AN UNFORMATTED DISK AND ITS
1934 ;*FORMATTING IS DONE IN THIS TEST. A SECTOR IS FORMATTED
1935 ;*AT A TIME. THE FIRST WORD OF EVERY SECTOR IS WRITTEN
1936 ;*TO BE A PSEUDO-HEADER CONTAINING THE DRIVE #, CYLINDER
1937 ;*#, SURFACE AND SECTOR #. THE FOLLOWING IS CHECKED
1938 ;*1. 'SIN' IF 'SIN' OCCURS, A DRIVE RESET IS DONE
1939 ;*AND THE SAME SECTOR IS FORMATTED AGAIN. THREE
1940 ;*RETRIES ARE DONE BEFORE AN ERROR MESSAGE IS PRINTED.
1941 ;*2. 'ERR' ON FINDING THAT THE 'ERR' BIT SET, RKER
1942 ;*SCANNED TO FIND OUT WHAT CAUSED IT AND THE
1943 ;*ERROR IS REPORTED.
1944 ;*****
1945 004624 000004          TST2:  SCOPE
1946 004626 013737 001230 002120    MOV     DRIVAD,DRHOLD      ;SAVE DRIVE NUMBER
1947 004634 005737 002114          TST     FDRIVE          ;SEE IF EVEN RK-05F DRIVE
1948 004640 001003          BNE      11$          ;YES
1949 004642 005737 002116          TST     FDRVE1         ;ODD RK-05F?
1950 004646 001125          BNE      TST3          ;DO NOT FORMAT IF ODD RK-05F
1951 004650
1952 004650 012702 177152    11$:  MOV      #-626,R2      ;203 CYLINDERS, (406 TRAKS)
1953 004654 012703 177764    MOV      #-14,R3          ;12 SECTORS
1954 004660 012701 177773    MOV      #-5,R1          ;ALLOW ONLY 5 'SIN' ERRORS
1955 004664 012705 177773    MOV      #-5,R5          ;ALLOW ONLY 5 'ERR'S
1956 004670 013704 001230    MOV     DRIVAD,R4          ;STORE ADRES OF DRIVE.
1957 004674 104415          4$:  CON.RESET
1958 004676 104416          DRV.RESET      ;GO TO 'DR-RST' & DO DRIVE RESET
1959 004700 005000          1$:  CLR      R0          ;KEEP COUNT OF 'SIN' ERORS
1960                                     ;ALLOW 3 RETRIES ON SIN
1961 004702 005777 174550    TST      DRKCS          ;ERR?
1962 004706 100001          BPL      3$          ;NO
1963
1964 004710 104415          CON.RESET      ;GO TO 'CN-RST' & DO CONTROL RESET
1965
1966 004712 005046          3$:  CLR      -(SP)

```

1967	004714	012746	004722		MOV	#12\$, -(SP)	
1968	004720	000002			RTI		; SET PRIORITY TO ZERO
1969	004722	010437	026446	12\$:	MOV	R4, OUTBUF	; WRITE THIS WORD
1970	004726	012777	026446	174526	MOV	OUTBUF, ARKBA	; FROM THIS ADRES
1971	004734	010477	174524		MOV	R4, ARKDA	; ON THIS DISK SECTOR
1972	004740	012777	177777	174512	MOV	#-1, ARKWC	; WRITE 1 WORD
1973	004746	012737	004674	001110	MOV	#4\$, SLPERR	; SET RETURN ADDRESS FOR
1974							; LUPING ON ERROR
1975							
1976	004754	012777	002003	174474	MOV	#2003, ARKCS	; GO WRT FMT
1977							
1978	004762	104421			CON.RDY		; WAIT FOR CONTROL READY
1979	004764	032777	001000	174460	BIT	#1000, ARKDS	; WAS THERE A SIN?
1980	004772	001413		5\$:	BEG	6\$	; NO, SKIP DOING DRV RESET
1981	004774	004737	016136		JSR	PC, GTSRG	; GO, GET RKCS, ER, DS, DA, CYLINDER
1982	005000	104007			ERROR	7	; SIN ERROR ON TRYING TO
1983							; WRT FMT ON CYLINDER AS
1984							; INDICATED IN RKDA. 3 RETRIES
1985							; ARE DONE
1986							; NOTE THAT BEFORE
1987	005002	104415			CON.RESET		; RETRYING A DRIVE RESET WAS DONE
1988	005004	104416			DRV.RESET		; GO TO 'DR-RST' & DO DRV RESET
1989	005006	005200			INC	R0	; INCRMT SIN COUNT
1990	005010	022700	000003		CMP	#3, R0	; ALLOW 3 RETRIES WERE THERE 3?
1991	005014	001332			BNE	1\$+2	; IF NOT, GO & RETRY
1992							
1993	005016	005201			INC	R1	; ALLOW ONLY 12 SIN ERRORS
1994							; IF MORE THAN 5 EXIT THIS TEST
1995	005020	001436			BEG	9\$	; IF MORE THAN 5 EXIT THIS TEST
1996	005022	005777	174430	6\$:	TST	ARKCS	; DID 'ERR' BIT SET IN RKCS?
1997	005026	100005			BPL	7\$	; NO, BRANCH
1998	005030	004737	016136		JSR	PC, GTSRG	; GO, GET RKCS, ER, DS, DA, CYL
1999	005034	104010			ERROR	10	; 'ERR' OCCURED WHILE DOING
2000							; WRT FMT ON SECTOR, CYLINDER
2001							; AS INDICATED IN RKDA.
2002	005036	005205			INC	R5	; ALLOW ONLY 5 'ERR'S. IF
2003	005040	001426			BEG	9\$	; MORE THAN 5 EXIT THIS TEST
2004	005042	005204		7\$:	INC	R4	; INCRMT DISK ADRES TO NXT SCTR
2005	005044	005203			INC	R3	; ALL 12 SECTORS DONE?
2006	005046	001314			BNE	1\$	; IF NOT, GO BAK & FMT NXT SCTR
2007							; IF YES
2008	005050	012703	177764		MOV	#-14, R3	; RESET COUNT FOR 12 SECTORS
2009	005054	042704	000017		BIC	#17, R4	; CLR OUT SEC BITS
2010							
2011	005060	062704	000020	8\$:	ADD	#20, R4	; ADRES THE NXT TRAK TO B FMTED
2012	005064	005202			INC	R2	; ALL TRAKS FMTED?
2013	005066	001304			BNE	1\$	; IF NOT GO BAK & FMT NXT CYL, SUR 0
2014	005070	005237	002114		INC	DRIVE	; EVEN RKDSF?
2015	005074	001004			BNE	10\$	; NO
2016	005076	062737	020000	001230	ADD	#20000, DRIVAD	; FORMAT ODD DRIVE OF F
2017	005104	000661			BR	11\$	
2018	005106	013737	002120	001230	MOV	DRHOLD, DRIVAD	; RESTORE DRIVE ADDR
2019	005114	000402		10\$:	BR	TST3	; EXIT
2020							
2021	005116	004737	016772	9\$:	JSR	PC, ABRT	
2022							

C04

2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078

005122	000004			
005124	012737	177773	001504	
005132	012737	177766	001506	
005140	012737	177773	001510	
005146	013705	001230		
005152	012737	177152	001476	
005160	104415			
005162	104416			
005164	005037	001254		
005170	005037	001252		
005174	012777	026446	174260	
005202	010577	174256		
005206	012777	177764	174244	
005214	012737	005160	001110	
005222	012777	002005	174226	
005230	104421			
005232	032777	001000	174212	
005240	001420			
005242	004737	016214		
005246	104011			

```

*****
:TEST 3 READ FORMAT OF THE DISK
:
: * IN THIS TEST, THE HEADERS FROM ALL THE SECTORS ARE READ
: * & CHECKED IF THEY ARE CORRECT. THE FOLLOWING IS THE
: * TEST SEQUENCE.
: * 1. READ 12 SECTORS (HORS ONLY) AT A TIME
: * 2. IF THERE IS A 'SIN' ERROR RETRY ONCE MORE, IF SAW AGAIN
: * REPORT ERROR & READ HEADER FROM NEXT CYLINDER
: * 3. IF THERE IS 'ERR' IN RKCS, DO A CONTROL RESET, REPORT
: * ERROR & READ HEADER FROM NEXT CYLINDER. IF THERE ARE
: * MORE THAN 5 ERRORS OF THIS KIND, THIS TEST WILL BE EXITED
: * 4. THE 12 HEADERS ARE CHECKED. IF THEY ARE CORRECT THE
: * NEXT CYLINDER IS READ.
: * IF THEY ARE NOT CORRECT, A RETRY IS DONE. IF AGAIN CORRECT
: * HEADERS ARE NOT RECIEVED, AN ERROR IS REPORTED. THE
: * SECTOR #'S GIVING THE BAD HEADERS, & THE BAD HEADERS ARE
: * STORED.
: * 5. IF INHIBIT TYPEOUT' SWITCH IS NOT SET, THE FIRST WORDS OF
: * THE 12 SECTORS (PSUEDO-HEADERS) ARE READ. IN A PREVIOUS
: * TEST THE FIRST WORD OF EVERY SECTOR WAS WRITTEN
: * AS A SOFTWARE HEADER (CONSISTING OF DRIVE #, CYL#, SUR SEC#,
: * THEN THE SECTOR # GIVING BAD HEADER, EXPECTED PSUEDO-HEADER,
: * & THE PS-HEADER RECIEVED ARE TYPED OUT. THIS WOULD
: * WRONG, HEADER WAS READ WRONG, ETC.
: * 6. THE NEXT CYLINDER IN LINE IS READ. ORDER OF READING IS
: * CYLO,SURO CYLO,SURI CYL312,SURI
*****
TST3: SCOPE
MOV #5,ERCNT1 ;ALLOW ONLY 5 ERRORS (OF BAD HEADER
;KIND FROM 5 CYLINDERS)
MOV #12,ERCNT2 ;ALLOW ONLY 12 'ERR'S
MOV #5,ERCNT3 ;ALLOW ONLY 5 ERRORS
MOV DRIVAD,RS ;SET DRIVE #,CYL ADRES=0
MOV #626,INDX2 ;313 CYLS (626 TRAKS) TO B REAC
4$: CON.RESET ;GO DO CONTROL RESET
DRV.RESET ;GO DO DRIVE RESET

1$: CLR RETRY2 ;ALLOW 2 RETRIES IF HORS READ WRONG
2$: CLR RETRY1 ;ALLOW 2 RETRIES FOR 'SINS'

3$: MOV #OUTBUF,ARKBA ;RD HORS INTO LOC STARTING AT THIS
MOV RS,ARKDA ;FROM THIS DSK ADRES
MOV #14,ARKWC ;12 HORS TO BE READ
MOV #45,$LPERR ;SET RETURN ADRES FOR LUPING ON ERROR

MOV #2005,ARKCS ;GO, RD FMT OF THIS CYLINDER

CON.RDY ;WAIT FOR CNTRL RDY TO SET

5$: BIT #1000,ARKDS ;'SIN' ERROR?
BEQ 6$ ;NO, BRANCH
JSR PC,GETINF

ERROR 11 ;'SIN' OCCURED WHEN DOING RD FMT
;FROM CYL SHOWN IN RKDA. IT

```

004

2079	005250	104415			CON. RESET		:DO CNTRL RESET
2080	005252	104416			DRV. RESET		:GO, DO DRIVE RESET
2081	005254	005237	001252		INC	RETRY1	:ALLOW ONLY 2 RETRIES FOR THIS ERROR
2082	005260	022737	000002	001252	CMP	#2,RETRY1	:IF TRIED 2 TIMES REPORT
2083	005266	001342			BNE	3\$	:ERROR, OTHERWISE GO BAK & RETRY
2084							:WAS TRIED TWICE, BUT 'SIN'.
2085	005270	005237	001510		INC	ERCNT3	:ALLOW 5 ERRORS AT MOST
2086	005274	001002			BNE	6\$	
2087	005276	000137	005606		JMP	16\$	
2088							
2089	005302	005777	174150		TST	DRKCS	: 'ERR' IN RKCS?
2090	005306	100010			BPL	7\$	:NO, BRANCH
2091	005310	004737	016136		JSR	PC,GT5RG	:GO, GET RKCS, ER,DS,DA,CYL,NDR
2092	005314	104012			ERROR	12	: 'ERR' SET WHILE DOING RD FMT
2093							:FROM CYL SHOWN IN RKDA
2094	005316	104415			CON. RESET		:GO DO CNTRL RESET
2095							:ALLOW ONLY 12 ERRORS OF THIS
2096	005320	005237	001506		INC	ERCNT2	:KIND, IF MORE THAN FIVE ERRORS
2097							:SKIP THIS TEST
2098	005324	001532			BEQ	TST4	:EXIT
2099	005326	000520			BR	14\$	:GO SET UP TO RD FMT FROM NXT
2100							:CYL IN LINE
2101							:CHECK THAT CORRECT HEADERS WERE RECVD.
2102							:SECTR # HAVING BAD HDR IS STORED ALONG
2103							:WITH BAD HDR
2104							
2105	005330	004737	007260		JSR	PC,CHKHDRS	:GO CHECK IF CORRECT HEADERS WERE READ
2106							
2107	005334	005737	001500		TST	INDX3	:WAS THERE A MISCOMPARISON?
2108	005340	001513			BEQ	14\$	:IF NOT, GO SET UP TO RD FMT
2109							:NXT CYL IN LINE
2110	005342	012737	005170	001110	MOV	#2\$,SLPERR	
2111	005350	104013			ERROR	13	:CORRECT HDRS WERE NOT RECVD
2112							:FROM SECTRS AS TYPED OUT.
2113							:THE SAME CYLINDER WAS READ TWICE
2114	005352	005237	001254		INC	RETRY2	:RETRY RD FMT ON SAME CYL AGAIN
2115	005356	022737	000002	001254	CMP	#2,RETRY2	:TRIED RDING SAME CYL TWICE
2116	005364	001301			BNE	2\$	:IF NOT, GO RD AGAIN
2117							:YES, REPORT ERROR
2118	005366	005237	001504		INC	ERCNT1	:ALLOW ONLY 5 ERRORS OF THE
2119							:ABOVE TYPE. IF MORE THAN 12
2120							:EXIT THIS TEST
2121	005372	001505			BEQ	16\$	
2122							
2123	005374						:THE PSEUDO-HEADERS (FIRST WORD OF EVERY
2124							:SECTOR) FROM THIS CYLINDER (ABOVE,
2125							:THE CYLINDER THAT GAVE WRONG HEADERS)
2126							:WILL BE READ, NOW. FOLLOWING WILL B TYPD OUT:
2127							:SEC#, EXPCD PSEUDO-HDR, RECVD PHDR.
2128							:IF "INHIBIT TIMEOUT" SW IS SET THIS ENTIRE
2129							:READING & TYPING WILL BE SKIPPED
2130	005374	032777	020000	173536	BIT	#20000,JSWR	:INHIBIT TIMEOUT?
2131	005402	001072			BNE	14\$	:YES, SKIP THE FOLLOWING & GO
2132							:SET UP TO RD FMT NXT CYL IN LINE
2133							
2134	005404	012701	177764		MOV	#-14,R1	:READ FROM 12 SECTRS

E04

2135	005410	010577	174050		MOV	R5,ARKDA	:FROM THIS DSK-ADRES
2136	005414	012777	026446	174040	MOV	ROUTBUF,ARKBA	:INTO THIS BUS-ADRES
2137	005422	012777	177777	174030	10S: MOV	#-1,ARKWC	:RD 1 WRD
2138							
2139	005430	012777	000005	174020	MOV	#5,ARKCS	:GO, RD
2140	005436	104421			CON.RDY		:WAIT FOR CNTRL RDY
2141	005440	005777	174012		TST	ARKCS	:ANY ERROR?
2142	005444	100002			BPL	15S	:NO, PROCEED
2143	005446	104415			CON.RESET		:CLEAR THE ERROR
2144	005450	000447			BR	14S	:ERROR, SO COULDN'T READ PSUEDO-HDRS
2145							
2146	005452	005201			15S: INC	R1	:READ FROM ALL 12 SECS
2147	005454	001362			BNE	10S	:IF NOT GO RD THE NXT ONE
2148							
2149							:TYPE OUT SUEDO-HDRS CORRESPONDING TO
2150							:THE SECTORS WHICH GAVE BAD HEADERS
2151							:TYPE: SEC #, EXPC P-HDR, RECVD P-HDR
2152							
2153	005456	104401			TYPE		:TYPE OUT
2154	005460	001771			MSG11		
2155	005462	012701	001266		MOV	#BUFR,R1	:SEC #'S ARE STORED HERE
2156							
2157	005466	104401			11S: TYPE		:TYPE CR, LF
2158	005470	001213			SCRLF		
2159							
2160	005472	011102			MOV	(R1),R2	
2161	005474	012703	026446		MOV	ROUTBUF,R3	:PSUEDO-HEADERS WHICH WERE
2162							:READ ARE STORED HERE
2163							
2164	005500	005702			12S: TST	R2	:IS THIS SEC # CORRESPONDING TO THE
2165	005502	001403			BEG	13S	:ONE IN ERROR
2166	005504	005302			DEC	R2	:R2 CONTAINS THE SEC #
2167	005506	005723			TST	(R3)+	
2168	005510	000773			BR	12S	
2169							
2170	005512	011146			13S: MOV	(R1),-(SP)	:GO TYPEOUT SEC # GIVING
2171	005514	104403			TYPOS		:MISCOMPARISON OF HEADERS
2172	005516	002			.BYTE	2	
2173	005517	000			.BYTE	0	:SUPRES LDG 0'S
2174							
2175	005520	104401			TYPE		:TYPE 2 BLANKS
2176	005522	002105			BLNK55		
2177							
2178	005524	010546			MOV	R5,-(SP)	:GO TYPE EXPCD PSUEDO HEADER
2179	005526	051116			BIS	(R1),(SP)	
2180	005530	104402			TYPOC		
2181							
2182	005532	104401			TYPE		:TYPE 2 BLNKS
2183	005534	002103			BLNK57		
2184							
2185	005536	011346			MOV	(R3),-(SP)	:GO TYPE PSUEDO-HEADER RECVD
2186	005540	104402			TYPOC		
2187							
2188	005542	005721			TST	(R1)+	:TYPED OUT ALL SEC #'S IN ERROR.
2189	005544	021127	177777		CMP	(R1),#177777	
2190	005550	001346			BNE	11S	:IF NOT GO BAK & TYPE NXT

F04

```

2191
2192 005552 104401 001733      TYPE      ,MSG7      ;TYPE OUT PC
2193 005556 012746 005374      MOV      #20$ -(SP)
2194 005562 104402      TYPQC
2195 005564 104401 001213      TYPE      ,SCRLF
2196                                     ;TYPE ROUTINE ENDS HERE
2197
2198                                     ;FIND OUT NXT TRAK TO B READ
2199                                     ;FORMATTED
2200
2201 005570 062705 000020      14$:  ADD      #20,R5      ;SET ADRES FOR SUR 0, NXT CYL IN LINE
2202 005574 005237 001476      INC      INDX2      ;READ ALL 313 CYLINDERS (626 TRAKS)?
2203 005600 001404      BEQ      TST4      ;EXIT
2204 005602 000137 005164      JMP      IS      ;IF NOT, GO BAK & READ NXT
2205
2206 005606 004737 016772      16$:  JSR      PC,ABRT
2207
2208 ;*****
2209 ;*TEST 4      SEEK PATTERNS: 0-312-0-311-...,USING IMPLIED SEEK
2210
2211 ;****TEST 2 (WRITING PSUEDO-HEADERS) SHOULD HAVE BEEN DONE BEFORE****
2212 ;**** DOING THIS TEST****
2213 ;*THIS TEST PERFORMS SEEKS (IMPLIED SEEKS USING 'READS') IN THE
2214 ;*FOLLOWING PATTERN.
2215 ;*0-312-0-311-0-310-.....0-1-0-0
2216
2217 ;*THE FIRST WORD OF EVERY SECTOR IS A PSEUDO-HEADER (WRITTEN IN
2218 ;*A PREVIOUS TEST) CONSISTING OF DRIVE NO, CYLINDER NO., SURFACE
2219 ;*AND SECTOR NO. AN IMPLIED SEEK IS DONE BY ISSUING A 'READ' FOR
2220 ;*THE PSEUDO-HEADER OF SECTOR 0, SURFACE 0.
2221
2222 ;*IF A 'SIN' OCCURS TWO TRIES ARE DONE BEFORE ABORTING. IF A 'SKE'
2223 ;*OCCURS IT COULD MEAN THAT 1) EITHER THE HEADERS WAS READ WRONG
2224 ;*OR 2) THE HEADS GOT POSITIONED ON THE WRONG CYLINDER. IN
2225 ;*ORDER TO PROVIDE A FURTHER INSIGHT INTO THE PROBLEM, THE FOLLOWING
2226 ;*IS DONE:
2227 ;*THE HEADERS ARE READ FROM THE CYLINDER THAT GAVE 'SKE'. IF THE HEADERS
2228 ;*ARE CORRECT IT IS SO REPORTED. IF THE HEADERS ARE INCORECT, THEN THE
2229 ;*EXPECTED HEADERS AND THE RECEIVED ONES ARE REPORTED. ONE MORE TRY IS
2230 ;*DONE (THE IMPLIED SEEK IS TRIED AGAIN BETWEEN THE CYLINDERS THAT GAVE RISE
2231 ;*TO 'SKE')
2232
2233 ;*THE FOLLOWING ACTION IS TAKEN WHEN THERE IS NO 'SKE' OR 'SIN' BUT STILL THE
2234 ;*PSEUDO-HEADER IS READ WRONG:
2235 ;*FIRST THE HEADERS ARE READ FROM THAT CYLINDER AND CHECKED. IF THEY ARE
2236 ;**CORRECT, IT IS SO REPORTED. IF THEY ARE WRONG THEN THE EXPECTED AND RECEIVED
2237 ;*HEADERS ARE REPORTED. THEN THE PSEUDO-HEADER FROM SECTOR 1 IS READ AND REPORT
2238 ;*ONE MORE TRY IS DONE BY REPEATING THE WHOLE PROCESS. (IMPLIED SEEK
2239 ;*BETWEEN THE TWO CYLINDERS AND READING PSEUDO-HEADER FROM SECTOR 0 OF THE DES*!
2240 ;*CYLINDER).
2241
2242 ;*UP TO 12 ERRORS OF EACH KIND (SIN, SKE, BAD PSEUDO-HEADER) ARE ALLOWED.
2243 ;*IF ANY ERROR OCCURS MORE THAN 12 TIMES THE TEST IS ABORTED.
2244 ;*****
2245 005612 000004      15$:  SCOPE
2246 005614 104415      CON.RESET      ;GO DO CONTROL RESET

```

2247	005616	104416				DRV.RESET	;GO DO DRIVE RESET
2248							
2249	005620	005004				CLR R4	;FLAG, CLR IF DOING IMPLIED ;SEEK IN FROM 0 TO 'INADR'
2250							;=1, IF GOING FROM 'INADR'
2251							;OUT TC CYL 0
2252							;313 SEEK PATTERNS
2253	005622	012737	177465	001476		MOV #313,INDX2	
2254	005630	012700	177764			MOV #14,R0	
2255	005634	010037	001504			MOV R0,ERCNT1	;ALLOW ONLY 12 ERRORS
2256	005640	010037	001506			MOV R0,ERCNT2	;OF THESE KINDS
2257	005644	010037	001510			MOV R0,ERCNT3	
2258							
2259	005650	012737	014500	001260		MOV #14500,INADR	; 'INADR' CONTAINS THE INNER ;CYL TO WHICH IMPLIED SEEK WILL ;BE DONE
2260							
2261							
2262							
2263	005656	005704			15:	TST R4	;GOING IN OR OUT?
2264	005660	001005				BNE 25	;GOING OUT, BRANCH
2265	005662	013705	001260			MOV INADR,R5	;SET CYL ADRES BITS FOR GOING IN
2266	005666	053705	001230			BIS DRIVAD,R5	;FORM DISK ADRES FOR INNER
2267	005672	000402				BR 35	;CYLINDER
2268	005674	013705	001230		25:	MOV DRIVAD,R5	;FORM DISK ADRES FOR OUTER
2269							;CYLINDER - 0
2270							;ALLOW 2 TRIES WHEN ;ERRORS OCCUR
2271	005700	012737	177776	001254	35:	MOV #-2,RETRY2	
2272	005706	012737	177776	001252	135:	MOV #-2,RETRY1	
2273	005714	012737	177777	001256	45:	MOV #-1,RETRY3	
2274	005722	000404				BR 55	
2275	005724	104415			65:	CON.RESET	
2276	005726	104416				DRV.RESET	
2277	005730	004737	006526			JSR PC,SBR1	;REPOSITION HEADS TO PRE-ERROR CYL
2278							
2279	005734	012777	177777	173516	55:	MOV #-1,ARKWC	;READ 1 WORD
2280	005742	010577	173516			MOV RS,ARKDA	;FROM THIS CYLINDER, SEC 0
2281	005746	012777	026446	173506		MOV #OUTBUF,ARKBA	;INTO THIS BUS ADRES
2282	005754	012737	005724	001110		MOV #65,SLPERR	;SET RETURN ADRES FOR LUPING ;ON 'ERROR'
2283							
2284							
2285	005762	012777	000005	173466		MOV #5,ARKCS	;GO, READ
2286							
2287	005770	104421				CON.RDY	;WAIT FOR CNTRL RDY
2288							
2289	005772	032777	001000	173452		BIT #1000,ARKDS	;SIN?
2290	006000	001434				BEQ 85	;NO, BRANCH
2291							;YES, THERE WAS A SIN
2292	006002	004737	016376			JSR PC,ERR1	;GO GET, CYLS BETW'N WHICH SK WAS TRIED
2293	006006	017737	173440	001170		MOV ARKDS,\$REG3	
2294	006014	017737	173434	001166		MOV ARKER,\$REG2	
2295	006022	104420	001602			TYPMSG MSG1	
2296	006026	013737	001256	001172		MOV RETRY3,\$REG4	;SAVE TRY # ON 'SIN'
2297	006034	062737	000002	001172		ADC #2,\$REG4	
2298	006042	104C14				ERROR 14	;AN IMPLIED SEEK WAS TRIED ;FROM 'CYLA' TO 'CYLB' (INDICATED ;IN ERROR MESSAGE), 'SIN' OCCURRED. ;2 TRIES ARE DONE BEFORE ;ABORTING
2299							
2300							
2301							
2302							

2303	006044	005737	001256		TST	RETRY3	: DONE RETRIES
2304	006050	001403			BEG	7\$	: YES, BRANCH
2305	006052	005237	001256		INC	RETRY3	: GO DO 2ND TRY
2306	006056	000722			BR	6\$	
2307							
2308	006060	005237	001504	7\$:	INC	ERCNT1	: ALLOW LESS THAN 12 ERRORS OF THIS TYPE
2309	006064	001103			BNE	19\$	: IF MORE SKIP THIS TEST
2310	006066	000137	006614		JMP	EXT4	: EXIT THIS TEST
2311							
2312	006072	032777	010000	173354	9\$:	BIT	: #10000, JAKER
2313	006100	001506			BEG	20\$	: SKE?
2314	006102	004737	016376		JSR	PC,ERR1	: GO GET 2 CYL NOS. BETWEEN WHICH
2315	006106	017737	173342	001166	15\$:	JAKER, \$REG2	: IMPLIED SEEK WAS DONE
2316	006114	017737	173332	001170	MOV	JRKDS, \$REG3	
2317	006122	013737	001252	001172	MOV	RETRY1, \$REG4	: GET TRY # ON 'SKE'
2318	006130	062737	000003	001172	ADD	#3, \$REG4	
2319	006136	104420	001610		TYPMSG	MSG2	: GO PRINT 'SKE'
2320	006142	104014			ERROR	14	: IMPLIED SEEK WAS TRIED FROM
2321							: 'CYLA' TO 'CYLB' (INDICATED
2322							: IN ERROR MESSAGE); 'SKE' OCCURRED.
2323							: 2 TRIES ARE DONE.
2324	006144	104415			CON.RESE*		: DO CONTROL RESET
2325							
2326	006146	004737	006552		9\$:	JSR	PC, SBR2
2327							: GO READ 12 HEADERS FROM
2328							: THIS CYLINDER & COMPARE
2329							: THEM. NOTE RS CONTAINS THE
2330							: DISK ADRES THAT WILL BE USED.
2331	006152	012777	000015	173276	MOV	#15, JRKCS	: GO DO DRIVE RESET
2332							: WHILE THE DRIVE IS DOING RESET
2333							: THE HDRS THAT WERE READ
2334							: ABOVE ARE CHECKED, PRINTED
2335							
2336	006160	005737	001500		TST	INDX3	: WAS THERE A MISCOMPARISON
2337							: IN ANY HEADER?
2338	006164	001006			BNE	10\$	: IF INDX3>0, THERE WAS.
2339							: NO, THERE WASN'T. HDRS OK
2340	006166	005237	001252		INC	RETRY1	: ONLY 2 TRIES FOR SKE
2341	006172	001414			BEG	12\$	: BRANCH IF THIS WAS A 2ND TRY
2342	006174	104420	001657		TYPMSG	MSG5	: TYPE OUT THAT HDRS WERE READ
2343							: CORRECTLY. THIS WAS TRY # 1
2344							
2345	006200	000405			BR	11\$	
2346							
2347							: HDRS WERE READ INCORRECT.
2348	006202	005237	001252	10\$:	INC	RETRY1	: ALLOW 2 TRIES FOR SKE
2349	006206	001411			BEG	14\$	: BRANCH, IF THIS WAS 2ND TRY
2350							
2351							: THERE WAS SKE ON DOING IMPLIED
2352	006210	104417	000015		MESSAGE	.15	: SEEK TO 'CYL B'. THEN HDRS WERE
2353							: READ FROM CYL B, WRONG HDRS
2354							: RECVD
2355							
2356	006214	104423		11\$:	RESDON		: WAIT FOR PREVIOUS DRIVE RESET
2357							: TO BE DONE
2358	006216	004737	006526		JSR	PC, SBR1	: GO, REPOSITION HEADS

Address	Op Code	Op 2	Op 3	Op 4	Op 5	Op 6	Op 7	Op 8	Op 9	Op 10	Op 11	Op 12	Op 13	Op 14	Op 15	Op 16	Op 17	Op 18	Op 19	Op 20	Op 21	Op 22	Op 23	Op 24	Op 25	Op 26	Op 27	Op 28	Op 29	Op 30	Op 31	Op 32	Op 33	Op 34	Op 35	Op 36	Op 37	Op 38	Op 39	Op 40	Op 41	Op 42	Op 43	Op 44	Op 45	Op 46	Op 47	Op 48	Op 49	Op 50	Op 51	Op 52	Op 53	Op 54	Op 55	Op 56	Op 57	Op 58	Op 59	Op 60	Op 61	Op 62	Op 63	Op 64	Op 65	Op 66	Op 67	Op 68	Op 69	Op 70	Op 71	Op 72	Op 73	Op 74	Op 75	Op 76	Op 77	Op 78	Op 79	Op 80	Op 81	Op 82	Op 83	Op 84	Op 85	Op 86	Op 87	Op 88	Op 89	Op 90	Op 91	Op 92	Op 93	Op 94	Op 95	Op 96	Op 97	Op 98	Op 99	Op 100	Op 101	Op 102	Op 103	Op 104	Op 105	Op 106	Op 107	Op 108	Op 109	Op 110	Op 111	Op 112	Op 113	Op 114	Op 115	Op 116	Op 117	Op 118	Op 119	Op 120	Op 121	Op 122	Op 123	Op 124	Op 125	Op 126	Op 127	Op 128	Op 129	Op 130	Op 131	Op 132	Op 133	Op 134	Op 135	Op 136	Op 137	Op 138	Op 139	Op 140	Op 141	Op 142	Op 143	Op 144	Op 145	Op 146	Op 147	Op 148	Op 149	Op 150	Op 151	Op 152	Op 153	Op 154	Op 155	Op 156	Op 157	Op 158	Op 159	Op 160	Op 161	Op 162	Op 163	Op 164	Op 165	Op 166	Op 167	Op 168	Op 169	Op 170	Op 171	Op 172	Op 173	Op 174	Op 175	Op 176	Op 177	Op 178	Op 179	Op 180	Op 181	Op 182	Op 183	Op 184	Op 185	Op 186	Op 187	Op 188	Op 189	Op 190	Op 191	Op 192	Op 193	Op 194	Op 195	Op 196	Op 197	Op 198	Op 199	Op 200	Op 201	Op 202	Op 203	Op 204	Op 205	Op 206	Op 207	Op 208	Op 209	Op 210	Op 211	Op 212	Op 213	Op 214	Op 215	Op 216	Op 217	Op 218	Op 219	Op 220	Op 221	Op 222	Op 223	Op 224	Op 225	Op 226	Op 227	Op 228	Op 229	Op 230	Op 231	Op 232	Op 233	Op 234	Op 235	Op 236	Op 237	Op 238	Op 239	Op 240	Op 241	Op 242	Op 243	Op 244	Op 245	Op 246	Op 247	Op 248	Op 249	Op 250	Op 251	Op 252	Op 253	Op 254	Op 255	Op 256	Op 257	Op 258	Op 259	Op 260	Op 261	Op 262	Op 263	Op 264	Op 265	Op 266	Op 267	Op 268	Op 269	Op 270	Op 271	Op 272	Op 273	Op 274	Op 275	Op 276	Op 277	Op 278	Op 279	Op 280	Op 281	Op 282	Op 283	Op 284	Op 285	Op 286	Op 287	Op 288	Op 289	Op 290	Op 291	Op 292	Op 293	Op 294	Op 295	Op 296	Op 297	Op 298	Op 299	Op 300	Op 301	Op 302	Op 303	Op 304	Op 305	Op 306	Op 307	Op 308	Op 309	Op 310	Op 311	Op 312	Op 313	Op 314	Op 315	Op 316	Op 317	Op 318	Op 319	Op 320	Op 321	Op 322	Op 323	Op 324	Op 325	Op 326	Op 327	Op 328	Op 329	Op 330	Op 331	Op 332	Op 333	Op 334	Op 335	Op 336	Op 337	Op 338	Op 339	Op 340	Op 341	Op 342	Op 343	Op 344	Op 345	Op 346	Op 347	Op 348	Op 349	Op 350	Op 351	Op 352	Op 353	Op 354	Op 355	Op 356	Op 357	Op 358	Op 359	Op 360	Op 361	Op 362	Op 363	Op 364	Op 365	Op 366	Op 367	Op 368	Op 369	Op 370	Op 371	Op 372	Op 373	Op 374	Op 375	Op 376	Op 377	Op 378	Op 379	Op 380	Op 38
---------	---------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	-------

J04

2415	006352	010537	001166		MOV	R5, \$REG2	; GET EXPCTD PSUEDO-HDR
2416	006356	013737	026446	001170	MOV	OUTBUF, \$REG3	; GET PSUEDO-HDR RECVD
2417	006364	104016			ERROR	16	; IMPLIED SEEK FROM CYLA TO CYLB WAS DONE.
2418							; READ PSUEDO-HEADER OF SEC 0,
2419							; CYLB (IN ERROR MESSAGE), BUT
2420							; THE WRONG PSEUDO-HEADER WAS
2421							; RECEIVED
2422	006366	005237	001254		INC	RETRY2	
2423	006372	001402			BEQ	21\$	
2424	006374	000137	005706		JMP	13\$	
2425							
2426							
2427	006400	004737	006552	21\$:	JSR	PC, SBR2	; GO READ HEADERS (12) FROM
2428							; THIS CYLINDER, & CHECK THEM.
2429							; IF MISCOMPARISON INDX3 WILL
2430							; BE > 0.
2431	006404	005737	001500		TST	INDX3	
2432	006410	001003			BNE	22\$	
2433	006412	104420	001657		TYPMSG	, MSG5	; WRONG PSUEDO-HDR WAS READ
2434							; BUT WHEN HDRS WERE READ
2435							; FROM THE SAME CYLINDER, THEY
2436	006416	000402			BR	23\$	; WERE CORRECT
2437							
2438	006420	104417	000015	22\$:	MESSAGE	, 15	; WRONG PSUEDO-HDR WAS READ
2439							; FROM 'CYLB' (IN ERROR MESSAGE).
2440							; THEN HEADERS WERE READ FROM THE
2441							; SAME CYLINDER. THEY WERE ALSO
2442							; WRONG.
2443	006424	010500		23\$:	MOV	R5, R0	; NOW READ THE PSUEDO-HEADER
2444	006426	005200			INC	R0	; FROM THE NEXT SECTOR (1)
2445	006430	010077	173030		MOV	R0, ZRKDA	; SAME CYLINDER
2446	006434	012777	026446	173020	MOV	OUTBUF, ZRKBA	
2447	006442	012777	177777	173010	MOV	#-1, ZRKWC	
2448	006450	012777	000005	173000	MOV	#5, ZRKCS	
2449	006456	104421			CON.RDY		
2450	006460	010537	001162		MOV	R5, \$REG0	
2451	006464	004737	016434		JSR	PC, GCYL	; GO GET CYL # & STORE IT IN \$REG0
2452	006470	010037	001164		MOV	R0, \$REG1	; GET EXPCT PSUEDO-HDR FROM SEC 1
2453	006474	013737	026446	001166	MOV	OUTBUF, \$REG2	
2454	006502	104417	000017		MESSAGE	, 17	; PSUEDO-HEADER FROM SEC 1, CYLB
2455							; (IN MESSAGE) WAS READ. THE EXPCTD
2456							; & RECVD DATA WORDS ARE REPORTED.
2457	006506	005237	001510		INC	ERCNT3	; ALLOW ONLY LESS THAN 10 ERRORS
2458							; OF THIS TYPE (WRONG PS-HDRS)
2459	006512	001440			BEQ	EXT4	
2460							
2461	006514	005704		24\$:	TST	R4	; SEEKED IN OR OUT LAST TIME?
2462	006516	001266			BNE	19\$	; IF OUT, GO SEEK NXT INNER CYL
2463							; IF IN, GO SEEK BAK TO 0
2464	006520	005204			INC	R4	; INDICATE THAT SEEK OUT (0)
2465	006522	000137	005656		JMP	1\$	; WILL BE DONE NOW
2466							
2467							
2468							
2469							
2470							

; THIS ROUTINE IS USED IN THIS TEST ONLY.
; R4=0 INDICATES SEEK BEING DONE FROM
; CYL 0 TO INNER CYL.

104

;R4=1 INDICATES SEEK BEING DONE FROM
 ;INNER CYL TO 0, THIS ROUTINE POSITIONS
 ;THE HEADS ON 'INADR' CYL IF R4=1

2471
 2472
 2473
 2474
 2475 006526 005704
 2476 006530 001407
 2477 006532 013777 001260 172724
 2478 006540 012777 000011 172710
 2479 006546 104422
 2480 006550 000207

SBR1: TST R4
 BEQ 15
 MOV INADR,ARKDA
 MOV #11,ARKCS
 TST RWS
 RTS PC

;THIS ROUTINE IS USED IN THIS TEST
 ;ONLY. IT READS 12 HEADERS FROM CYLINDER
 ;WHOSE ADRES IS IN R5. THEN IT CHECKS
 ;IF THE EXPECTED HEADER IS RECEIVED.
 ;IF IT IS NOT, INDX3 IS INCREMENTED INDICATING
 ;THE ERROR

2481
 2482
 2483
 2484
 2485
 2486
 2487
 2488
 2489 006552 012700 177764
 2490 006556 012701 026446
 2491 006562 010077 172672
 2492 006566 010177 172670
 2493 006572 010577 172666
 2494 006576 012777 002005 172652
 2495 006604 104421

SBR2: MOV #-14,RD
 MOV #OUTBUF,R1
 MOV RD,ARKWC ;READ 12 HDRS
 MOV R1,ARKBA ;INTO THIS ADRES
 MOV RS,ARKDA ;FROM THIS CYLINDER
 MOV #2005,ARKCS ;RD FMT, GO
 CON.RDY

2496
 2497 006606 004737 007260
 2498
 2499 006612 000207
 2500
 2501 006614 004737 016772

JSR PC,CHKHDRS ;GO CHECK IF CORRECT HEADERS WERE READ
 RTS PC ;EXIT

EXT4: JSR PC,ABRT

 ;TEST 5 PERFORM CONVERGING-DIVERGING (IMPLIED) SEEKS
 ;THIS TEST PERFORMS A CONVERGING-DIVERGING SEEK PATTERN
 ;USING IMPLIED SEEK (READ FORMAT). THE SEEK SEQUENCE IS:
 ;0-312-1-311-2-310-3-307-----310-2-311-1-312
 ;ALL READ FORMATS ARE DONE FROM SURFACE 0.
 ;THE CYLINDER ADDRESSES, BETWEEN WHICH THE IMPLIED SEEK IS
 ;PERFORMED, ARE CONTAINED IN 'OUTADR' & 'INADR'. IF 'SIN' OCCURS
 ;AN ERROR IS REPORTED AND A RETRY IS DONE. ON READING INCORRECT
 ;HEADERS AN ERROR IS REPORTED AND A RETRY IS DONE. NOTE THAT IF
 ;ALL THE 12 HEADERS ARE INCORRECT, IT COULD MEAN THAT THE HEADS
 ;COULD NOT POSITION CORRECTLY. THIS WOULD BE CONFIRMED IF IN
 ;PREVIOUS TESTS BAD HEADERS WERE NOT RECIEVED FROM THE SAME
 ;CYLINDER. IF THAT CYLINDER GAVE BAD HEADERS IN ALL THE PREVIOUS
 ;TESTS THE PROBLEM COULD BE DIFFERENT.
 ;MAXIMUM 12 ERRORS OF ANY KIND ARE ALLOWED.
 ;IF MORE THAN 12 ERRORS OCCUR THE TEST IS ABORTED.

2520
 2521 006620 000004
 2522 006622 104415
 2523 006624 104416
 2524
 2525 006626 005004
 2526

 ;ST5: SCOPE
 CON.RESET ;GO,DO CONTROL RESET
 DRV.RESET ;GO,DO DRIVE RESET
 CLR R4 ;(R4)=0 SEEKING FROM 'OUTADR' TO 'INADR'
 ;(R4)=1 SEEKING FROM 'INADR' TO 'OUTADR'

Address	Op Code	Op 1	Op 2	Op 3	Op 4	Op 5	Op 6	Op 7	Op 8	Op 9	Op 10	Op 11	Op 12	Op 13	Op 14	Op 15	Op 16	Op 17	Op 18	Op 19	Op 20	Op 21	Op 22	Op 23	Op 24	Op 25	Op 26	Op 27	Op 28	Op 29	Op 30	Op 31	Op 32	Op 33	Op 34	Op 35	Op 36	Op 37	Op 38	Op 39	Op 40	Op 41	Op 42	Op 43	Op 44	Op 45	Op 46	Op 47	Op 48	Op 49	Op 50	Op 51	Op 52	Op 53	Op 54	Op 55	Op 56	Op 57	Op 58	Op 59	Op 60	Op 61	Op 62	Op 63	Op 64	Op 65	Op 66	Op 67	Op 68	Op 69	Op 70	Op 71	Op 72	Op 73	Op 74	Op 75	Op 76	Op 77	Op 78	Op 79	Op 80	Op 81	Op 82	Op 83	Op 84	Op 85	Op 86	Op 87	Op 88	Op 89	Op 90	Op 91	Op 92	Op 93	Op 94	Op 95	Op 96	Op 97	Op 98	Op 99	Op 100	Op 101	Op 102	Op 103	Op 104	Op 105	Op 106	Op 107	Op 108	Op 109	Op 110	Op 111	Op 112	Op 113	Op 114	Op 115	Op 116	Op 117	Op 118	Op 119	Op 120	Op 121	Op 122	Op 123	Op 124	Op 125	Op 126	Op 127	Op 128	Op 129	Op 130	Op 131	Op 132	Op 133	Op 134	Op 135	Op 136	Op 137	Op 138	Op 139	Op 140	Op 141	Op 142	Op 143	Op 144	Op 145	Op 146	Op 147	Op 148	Op 149	Op 150	Op 151	Op 152	Op 153	Op 154	Op 155	Op 156	Op 157	Op 158	Op 159	Op 160	Op 161	Op 162	Op 163	Op 164	Op 165	Op 166	Op 167	Op 168	Op 169	Op 170	Op 171	Op 172	Op 173	Op 174	Op 175	Op 176	Op 177	Op 178	Op 179	Op 180	Op 181	Op 182	Op 183	Op 184	Op 185	Op 186	Op 187	Op 188	Op 189	Op 190	Op 191	Op 192	Op 193	Op 194	Op 195	Op 196	Op 197	Op 198	Op 199	Op 200	Op 201	Op 202	Op 203	Op 204	Op 205	Op 206	Op 207	Op 208	Op 209	Op 210	Op 211	Op 212	Op 213	Op 214	Op 215	Op 216	Op 217	Op 218	Op 219	Op 220	Op 221	Op 222	Op 223	Op 224	Op 225	Op 226	Op 227	Op 228	Op 229	Op 230	Op 231	Op 232	Op 233	Op 234	Op 235	Op 236	Op 237	Op 238	Op 239	Op 240	Op 241	Op 242	Op 243	Op 244	Op 245	Op 246	Op 247	Op 248	Op 249	Op 250	Op 251	Op 252	Op 253	Op 254	Op 255	Op 256	Op 257	Op 258	Op 259	Op 260	Op 261	Op 262	Op 263	Op 264	Op 265	Op 266	Op 267	Op 268	Op 269	Op 270	Op 271	Op 272	Op 273	Op 274	Op 275	Op 276	Op 277	Op 278	Op 279	Op 280	Op 281	Op 282	Op 283	Op 284	Op 285	Op 286	Op 287	Op 288	Op 289	Op 290	Op 291	Op 292	Op 293	Op 294	Op 295	Op 296	Op 297	Op 298	Op 299	Op 300	Op 301	Op 302	Op 303	Op 304	Op 305	Op 306	Op 307	Op 308	Op 309	Op 310	Op 311	Op 312	Op 313	Op 314	Op 315	Op 316	Op 317	Op 318	Op 319	Op 320	Op 321	Op 322	Op 323	Op 324	Op 325	Op 326	Op 327	Op 328	Op 329	Op 330	Op 331	Op 332	Op 333	Op 334	Op 335	Op 336	Op 337	Op 338	Op 339	Op 340	Op 341	Op 342	Op 343	Op 344	Op 345	Op 346	Op 347	Op 348	Op 349	Op 350	Op 351	Op 352	Op 353	Op 354	Op 355	Op 356	Op 357	Op 358	Op 359	Op 360	Op 361	Op 362	Op 363	Op 364	Op 365	Op 366	Op 367	Op 368	Op 369	Op 370	Op 371	Op 372	Op 373	Op 374	Op 375	Op 376	Op 377	Op 378	Op 379	Op 380</
---------	---------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	----------

104

2583	007072	001527			BEG	EXT5		: ABOVE KIND
2584								: IF MORE SKIP THIS TEST
2585								: SIN OCCURED WHEN GOING TO CYL (IN
2586								: RS). A DRIVE RESET HAS BEEN DONE.
2587								: NOW TRY POSITIONING HEADS ON
2588								: THAT CYL.
2589	007074	010577	172364		MOV	R5,ARKDA		: SET CYL ADRES
2590	007100	012777	000011	172350	MOV	#11,ARKCS		: SEEK,GO
2591	007106	104422			TST.RWS			
2592	007110	000424			BR	115		
2593								: IF NO SIN, ENTER HERE
2594	007112	004737	007260		JSR	PC,CHKHDRS		: GO CHECK IF CORRECT HEADERS WERE READ
2595								
2596	007116	012737	006716	001110	MOV	#45,\$LPERR		: SET LUP ADDRESS
2597	007124	005737	001500		TST	INDX3		: WAS THERE A BAD HDR?
2598	007130	001414			BEG	115		: NO, BRANCH
2599	007132	104020			ERROR	20		: WRONG HEADERS WERE READ FROM
2600								: 'SEC #'S, ON DOING AN IMPLIED
2601								: SEEK (RDfmt) FROM 'CYLA' TO 'CYLB'
2602	007134	005237	001254		INC	RETRY2		: ALLOW 2 TRIES
2603	007140	022737	000002	001254	CMP	#2,RETRY2		
2604	007146	001263			BNE	45		: GO TRY 2ND TIME
2605	007150	005237	001506		INC	ERCNT2		: ALLOW ONLY 12 ERRORS
2606	007154	001002			BNE	115		
2607	007156	000137	007352		JMP	EXT5		: IF MORE, EXIT THIS TEST
2608								
2609	007162	005704			TST	R4		: GOING WHICH WAY?
2610	007164	001006			BNE	125		: 'INADR' TO 'OUTADR', BRANCH
2611								: 'OUTADR' TO 'INADR'
2612	007166	005204			INC	R4		: INDICATE THAT NXT TIME GOING
2613								: FROM 'INADR' TO 'OUTADR'
2614	007170	062737	000040	001262	ADD	#40,OUTADR		: INCREMENT CYLINDER ADRES
2615	007176	000137	006664		JMP	15		: GO BAK & DO IMPLIED SEEK
2616								: FROM 'INADR' TO 'OUTADR'
2617								
2618	007202	005004			CLR	R4		: INDICATE THAT NXT TIME GOING
2619								: FROM 'OUTADR' TO 'INADR'
2620	007204	162737	000040	001260	SUB	#40,INADR		: DECREMENT CYLINDER ADRES
2621	007212	005237	001476		INC	INDX2		: DONE ALL 312 FORWARD-BACKWARD
2622								: SEEK PATTERNS
2623	007216	001457			BEG	TST6		: IF YES, EXIT
2624								
2625	007220	000137	006664		JMP	15		: IF NOT, GO BAK & DO IMPLIED
2626								: SEEK FROM 'OUTADR' TO 'INADR'
2627								
2628								: THIS SUBROUTINE IS ENTERED AFTER A 'SIN' ERROR OCCURED ON GOING FROM
2629								: 'CYLA' TO 'CYLB' AS INDICATED IN THE ERROR MESSAGE. BEFORE RETRYING THE
2630								: HEADS HAVE TO BE POSITIONED BACK TO 'CYLA'. NOTE THAT A DRIVE RESET
2631								: WAS DONE TO CLEAR SIN.
2632								: R4=0, INDICATES SEEK IS BEING DONE FROM 'OUTADR' CYLINDER TO 'INADR' CYLINDER.
2633								: R4=1, INDICATES THAT SEEK IS BEING DONE FROM 'INADR' TO 'OUTADR'.
2634	007224	005704			SBR3: TST	R4		: GOING WHICH WAY?
2635	007226	001404			BEG	15		: IF FROM 'OUTADR' TO 'INADR', BRANCH.
2636	007230	013777	001260	172226	MOV	INADR,ARKDA		
2637	007236	000403			BR	25		
2638	007240	013777	001262	172216	15: MOV	OUTADR,ARKDA		

2639 007246 012777 000011 172202 25:
2640 007254 104422
2641 007256 000207
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657

MOV #11,ARKCS
TST.RWS
RTS PC

;CHKHORS
;THIS ROUTINE CHECKS THAT THE HEADERS READ PREVIOUSLY WERE CORRECT.
;IF NOT, THE BAD HEADERS AND THE SECTOR #'S FROM WHICH THEY WERE READ
;ARE STORED.
;ON ENTRY:
;R5 CONTAINS DISK ADDRESS FROM WHERE HEADERS WERE READ.
;OUTBUF - 12 HEADERS READ PREVIOUSLY ARE STORED STARTING 'OUTBUF'.
;ON EXIT:
;INX3=0, IF THE HEADERS WERE CORRECT
;INX3=1, IF THE HEADERS WERE INCORRECT
;BUFR - SECTOR #'S GIVING BAD HEADERS ARE STORED STARTING AT 'BUFR'.
;BUFR1 - BAD HEADERS FOR THE ABOVE SECTORS ARE STORED STARTING 'BUFR1'.
;THE BAD SECTOR TABLE IS TERMINATED BY A '177777' WORD.

2658 007260 005000
2659 007262 012701 026446
2660 007266 012702 001266
2661 007272 012703 001320
2662 007276 010537 001120
2663 007302 042737 160037 U01120
2664 007310 005037 001500
2665
2666 007314 023711 001120
2667 007320 001406
2668 007322 011123
2669 007324 010022
2670
2671 007326 012712 177777
2672 007332 005237 001500
2673 007336 005721
2674 007340 005200
2675 007342 022700 000014
2676 007346 001362
2677 007350 000207
2678
2679 007352 004737 016772
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694

CHKHORS: CLR R0 ;INITIALIZE FOR 14 HDRS
MOV #OUTBUF,R1 ;INITIALIZE PTR TO HDRS RECVD
MOV #BUFR,R2 ;INITIALIZE PTR TO SECTOR TABLE
MOV #BUFR1,R3 ;INITIALIZE PTR TO BAD HDR TABLE
MOV R5,\$GDADR
BIC #160037,\$GDADR ;GET EXPCTD HEADER
CLR INX3 ;CLR FLG INDICATING BAD HDRS
9\$: CMP \$GDADR,(R1) ;HEADER OK?
BEQ 10\$;YES, BRANCH
MOV (R1),(R3)+ ;SAVE BAD HDR
MOV R0,(R2)+ ;SAVE BAD SECTR #
MOV #177777,(R2) ;PUT TERMINATR ON SECTR TABLE
INC INX3 ;SET FLG INDICATING BAD HDR
10\$: TST (R1)+ ;INCRMT PTR TO NXT HDR
INC R0 ;ALL HDRS CHKD?
CMP #14,R0
BNE 9\$;IF NOT, LUP BAK
RTS PC

EXT5: JSR PC,ABRT

;;*****

;*TEST 6 WRITE PATTERNS ON THE DISK
;*IN THIS TEST DIFFERENT PATTERNS ARE WRITTEN ON THE ENTIRE DISK. 768
;*WORDS (3 SECTORS) ARE WRITTEN AT A TIME. TWO 768 WORDS LONG
;*BUFFERS HAVE BEEN USED IN THIS TEST. WHILE ONE BUFFER IS
;*USED TO WRITE ON THE DISK, THE OTHER BUFFER GETS FILLED UP WITH
;*PATTERNS TO BE USED NEXT! THIS OVERLAPPING IS DONE TO SAVE TIME.
;*THREE PATTERN-GENERATOR SUB-ROUTINES ARE USED:
;*1. PTGEN0 2. PTGEN1 3. PTGEN2 4. PTGEN3
;*THESE THREE ROUTINES ARE CALLED IN A CYCLIC ORDER:
;* PTGEN0-PTGEN1-PTGEN2-PTGEN3-PTGEN0-PTGEN1.....
;*THE FOLLOWING ORDER IS MAINTAINED IN WRITING BLOCKS (EACH
;*3 SECTORS LONG) USING ONE OF THE FOUR PATTERN GENERATORS:

2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750

007356	000004		
007360	012737	177764	001504
007366	012737	177764	001506
007374	012737	177152	001474
007402	005037	001410	
007406	005037	001412	
007412	013737	001230	001432
007420	012737	001414	001424
007426	004737	010044	
007432	017737	171766	001430
007440	004777	171764	
007444	005005		
007446	005737	001410	
007452	100407		
007454	013737	001406	001434
007462	052737	000200	001412
007470	000406		
007472	013737	001404	001434
007500	052737	000200	001410
007506	012737	007514	001110

```

; *CYL SECTORS CYL SECTORS CYL SECTORS CYL SECTORS
; * 0 0 0-2 PTGEN0 0 0 6-10 PTGEN1 0 0 3-5 PTGEN2 0 0 11-13 PTGEN3
; * 0 1 0-2 PTGEN0 0 1 6-10 PTGEN1 0 1 3-5 PTGEN2 0 1 11-13 PTGEN3
; * 1 0 0-2 PTGEN0 1 0 6-10 PTGEN1 1 0 3-5 PTGEN2 1 0 11-13 PTGEN3
; * 1 1 0-2 PTGEN0 1 1 6-10 PTGEN1 1 1 3-5 PTGEN2 1 1 11-13 PTGEN3
; * 2 0 0-2 PTGEN0 2 0 6-10 PTGEN1 2 0 3-5 PTGEN2 2 0 11-13 PTGEN3
; *ETC, ETC.....
; *THE ABOVE SHOWN STAGGERING (OF SECTORS TO BE WRITTEN) IS DONE TO
; *SAVE ONE REV (40MS) EVERY TIME A BLOCK (3 SECTORS) IS WRITTEN.
; *IF YOU WANT TO USE BUFFERS STARTING AT SOME OTHER MEMORY ADDRESS
; *MAKE THE FOLLOWING CHANGES:
; *CHANGE 'PBUF0' TO STARTING ADDRESS OF THE FIRST 768 WORDS LONG BUFFER.
; *CHANGE 'PBUF1' TO STARTING ADDRESS OF SECOND 768 WORDS LONG BUFFER.

; *IF YOU WANT TO WRITE YOUR OWN PATTERNS USING PATTERN GENERATOR 'PTGENC'
; *CHANGE 'PTRND1' AND 'PTRND2' TO THE PATTERNS YOU WANT.

; *TO WRITE THE SAME TWO (OR ONE) PATTERNS ON THE ENTIRE DISK CHANGE
; *LOCATION 'PAT1' TO 'PTGEN0' THE STARTING ADDRESS OF PAT-GENERATOR 0
; *LOCATION 'PAT2' TO 'PTGEN0' THE STARTING ADDRESS OF PAT-GENERATOR 0
; *LOCATION 'PAT3' TO 'PTGEN0' THE STARTING ADDRESS OF PAT-GENERATOR 0
; *****
; *ST6: SCOPE
; *MOV #14, ER CNT1
; *MOV #14, ER CNT2
; *MOV #626, IND X1
; *CLR BUFLG0
; *CLR BUFLG1
; *SET COUNT FOR 313X2 TRACKS
; *CLR FLAG FOR BUFR 0
; *CLR FLAG FOR BUFR 1
; *BIT 7 OF ABOVE FLAGS WHEN SET
; *INDICATES, THAT BUFR TO BE USED
; *FOR WRITING ON DSK
; *GET DRIVE #, DISK ADRES
; *INITIALIZE PTR TO THE FIRST
; *PATRN GENERATOR

; *GO GENERATE PATRNS FOR
; *3 SECTOR (400X3)
; *INITIALIZE COUNT FOR 4 BLOCKS
; *FIND OUT WHICH BUFR TO USE
; *FOR WRITING ON DSK
; *USE 'IOBUF1' FOR TRANSFER
; *OR THE ONE INDICATED BY THE USER
; *SET FLAG TO INDICATE THAT
; *WRITING ON DSK WILL B DONE FROM
; *THIS BUFR (BUFR 1)

; *USE 'IOBUF0' FOR TRANSFER
; *OR THE ONE INDICATED BY THE USER

; *INDICATE THAT 'IOBUF0' WILL
; *B USED FOR WRITING ON DISK

```

MD-11-DZRLK-E, RK11-RK05 DYNAMIC TEST
DZRLKLE.P11 26-APR-77 12:27

MACY11 30(1046) 14-JUL-77 08:03 PAGE 54
T6 WRITE PATTERNS ON THE DISK

2751	007514	013777	001432	171742	45:	MOV	DSKADR, DSKDA	; SET RKDA
2752	007522	013777	001434	171732		MOV	BUSADR, DSKBA	
2753	007530	012777	176400	171722		MOV	#-1400, DSKWC	
2754	007536	012777	000003	171712		MOV	#3, DSKCS	; WRITE THE 4 SECTORS ON
2755								; DISK
2756	007544	013737	001424	001426		MOV	PRSPAT, NXTPAT	; WHILE THE PATRNS R BEING WRITTEN
2757	007552	023717	001424	001422		CMP	PRSPAT, #PAT3	; GO GENERATE THE NXT PATRNS
2758	007560	001004				BNE	55	; TO B WRITTEN
2759	007562	012737	001414	001426		MOV	#PAT0, NXTPAT	; KEEP GENERATING PATRNS IN THIS
2760	007570	000403				BR	65	; WAY "PAT0"- "PAT1"- "PAT2"- "PAT3"- "PATC"-
2761	007572	062737	000002	001426	55:	ADD	#2, NXTPAT	
2762	007600	004737	010044		65:	JSR	PC, GETBUF	
2763	007604	017737	171616	001430		MOV	#NXTPAT, PGSUBR	
2764	007612	004777	171612			JSR	PC, PGSUBR	; GO GENERATE THESE PATRNS.
2765								; (3 X 400) WORDS
2766	007616	104421				CON. RDY		
2767	007620	032777	140000	171630		BIT	#140000, DSKCS	; ANY ERROR?
2768	007626	001411				BEG	125	; GET RKCS, ER, DS, DA
2769	007630	004737	016162			JSR	PC, GT4RG	
2770	007634	104021				ERROR	21	; ERROR ON DOING WRITE
2771	007636	104415				CON. RESET		; CLEAR IT
2772	007640	005237	001504			INC	ERCNT1	; ALLOW 12 ERRORS AT MOST
2773	007644	001002				BNE	125	
2774	007646	000137	010432			JMP	EXT6	; IF MORE, EXIT
2775	007652	032777	001000	171572	125:	BIT	#BIT9, DSKDS	; SIN, ON DOING WRITE?
2776	007660	001412				BEG	75	
2777	007662	004737	016162			JSR	PC, GT4RG	
2778	007666	104022				ERROR	22	; SIN ERROR ON DOING WRITE
2779	007670	104415				CON. RESET		
2780	007672	104416				CON. RESET		
2781	007674	005237	001506			INC	ERCNT2	; ALLOW 12 ERRORS AT MOST
2782	007700	001002				BNE	75	
2783	007702	000137	010432			JMP	EXT6	
2784								; FIGURE OUT WHICH BUFFER IS
2785								; AVAILABLE FOR USE
2786	007706	105737	001410		75:	TSTB	BUFLG0	; WAS PREVIOUS DSK-WRITE DONE
2787	007712	100003				BPL	85	; USING BUFR 0?
2788	007714	005037	001410			CLR	BUFLG0	; YES, CLR FLAG INDICATING IT'S
2789								; AVAILABLE NOW
2790	007720	000402				BR	95	
2791	007722	005037	001412		85:	CLR	BUFLG1	; CLR FLAG INDICATING BUFR1
2792								; IS AVAILABLE NOW
2793	007726	013737	001426	001424	95:	MOV	NXTPAT, PRSPAT	; PRSPAT'S PATRNS WILL BE USED
2794								; ON NEXT WRITE
2795	007734	010500				MOV	R5, R0	; FORM SEC # TO BE USED NXT TIME
2796	007736	116000	010426			MOVB	SECPTR(R0), R0	; GET SEC #
2797	007742	042737	000017	001432	105:	BIC	#17, DSKADR	; MASK SECTOR BITS FROM DSK-ADRES
2798	007750	050037	001432			BIS	R0, DSKADR	; FORM NXT DSK-ADRES
2799	007754	005205				INC	R5	; DONE WITH 12 SECTRS (3 BLOCKS)?
2800	007756	022705	000004			CMP	#4, R5	
2801	007762	001231				BNE	25	; ON THIS SURFACE? IF NOT, GO
2802								; DO NXT 4 SECTRS
2803	007764	032737	000001	001474		BIT	#BIT0, INDX1	; WHICH SURFACE WAS DONE, 0 OR 1?
2804	007772	001415				BEG	115	; IF 0, GO DO SURFACE 1
2805	007774	005237	001474			INC	INDX1	; COUNT # OF TRACKS
2806	010000	001002				BNE	.+6	

005

2807	010002	000137	010436		JMP	TST7	:EXIT IF DONE
2808	010006	042737	000020	001432	BIC	#20,DSKADR	:GO TO SUR 0, NEXT CYLINDER
2809	010014	062737	000040	001432	ADD	#40,DSKADR	
2810	010022	000137	007444		JMP	IS	:GO BACK AND DO WRITE
2811	010026	005237	001474		115: INC	INDX1	:COUNT # OF TRACKS
2812	010032	052737	000020	001432	BIS	#20,DSKADR	:SET BIT FOR SUR 1
2813	010040	000137	007444		JMP	IS	:GO, WRITE PATRNS ON SURFACE 1
2814					;*GET BUF#		
2815					;*THIS ROUTINE FINDS OUT WHICH BUFFER (IOBUF0, IOBUF1) OR ONE OF		
2816					;*THE TWO BUFFERS SELECTED BY THE USER) TO USE		
2817					;*FOR GENERATING PATTERNS. THEN IT SETS UP A FLAG INDICATING THAT		
2818					;*BUFFER HAS TO BE FILLED UP. THE STARTING ADDRESS OF THE		
2819					;*BUFFER IS STORED IN 'BUF #'.		
2820							
2821	010044	005737	001410		GETBUF: TST	BUFLGO	:BUFR 0 AVAILABLE FOR USE?
2822	010050	100007			BPL	IS	:YES, BRANCH
2823	010052	052737	100000	001412	BIS	#BIT15,BUFLG1	:NO, USE BUFR 1. INDICATE SO.
2824	010060	013737	001406	001446	MOV	PBUF1,BUFNO	:SAVE STARTING ADRES OF BUFR1
2825	010066	000207			RTS	PC	
2826	010070	052737	100000	001410	15: BIS	#BIT15,BUFLGO	:INDICATED, USING BUFR 0
2827	010076	013737	001404	001446	MOV	PBUF0,BUFNO	:SAVE STARTING ADRES OF BUFR 0
2828	010104	000207			RTS	PC	:RETURN
2829					;*PTGENO		
2830					;*THIS ROUTINE GENERATES A 768 (3X256) WORDS LONG		
2831					;*BUFFER CONTAINING THE FOLLOWING PATTERNS		
2832					;*FIRST BLOCK OF 256 WORDS: 125252		
2833					;*SECOND BLOCK OF 256 WORDS: 052525 (COMPLEMENT OF ABOVE)		
2834					;*THIRD BLOCK OF 256 WORDS: 010421		
2835					;*YOU CAN USE ANY OTHER PATTERN/S (& ITS COMPLEMENT)		
2836					;*MAKING THE CHANGES IN THE 2 LOCATIONS SHOWN BELOW.		
2837							
2838	010106	013700	001446		PTGENO: MOV	BUFNO,R0	:GET STARTING ADRES OF BUFR
2839	010112	013701	010164		MOV	PTRNO1,R1	:GET PATRN TO BE GENERATED
2840	010116	012702	177400		MOV	#-400,R2	:IN THE FIRST 400 WORD BLOCK
2841							
2842	010122	010120			15: MOV	R1,(R0)+	:GENERATE THE FIRST BLOCK
2843	010124	005202			INC	R2	:WITH 'PAT01' PATRN
2844	010126	001375			BNE	IS	:ALL DONE?
2845							
2846	010130	012702	177400		MOV	#-400,R2	
2847	010134	005101			COM	R1	:COMPLEMENT 'PAT01' PATRN
2848	010136	010120			25: MOV	R1,(R0)+	:GENERATE 2ND BLOCK WITH
2849	010140	005202			INC	R2	: 'PAT01'S COMPLEMENT PATRN
2850	010142	001375			BNE	25	:ALL DONE?
2851	010144	012702	177400		MOV	#-400,R2	
2852	010150	013701	010166		MOV	PTRNO2,R1	:GET PATRN TO BE GENERATED
2853					:FOR 3RD BLOCK		
2854	010154	010120			35: MOV	R1,(R0)+	:GENERATE 3RD BLOCK USING
2855					: 'PAT02' PATRN		
2856	010156	005202			INC	R2	
2857	010160	001375			BNE	35	:ALL DONE?
2858							
2859	010162	000207			RTS	PC	:RETURN
2860							
2861	010164	125252			PTRNO1: 125252		:CHANGE THESE LOCATIONS IF
2862	010166	010421			PTRNO2: 010421		:YOU WANT ANY OTHER PATTERNS

E05

2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918

010170	012703	000001
010174	012704	177776
010200	013700	001446
010204	012702	177760
010210	010301	
010212	010120	
010214	000261	
010216	006101	
010220	103374	
010222	005202	
010224	001371	
010226	012702	177760
010232	010401	
010234	010120	
010236	000241	
010240	006101	
010242	103774	
010244	005202	
010246	001371	
010250	012702	177760
010254	010301	
010256	010120	
010260	006301	
010262	103375	
010264	005202	
010266	001372	
010270	000207	

;*PTGEN1
;*THIS ROUTINE GENERATES A 768 (3X256) WORDS
;*LONG BUFFER CONTAINING THE FOLLOWING PATTERNS:

;*FIRST BLOCK-256 WORDS 000001 FILL 1'S
;* 000003
;*
;* 177777

;*SECOND BLOCK 177776 FILL 0'S
;* 177774
;*
;* 000000

;*THIRD BLOCK 000001 FLOAT A 1
;* 000020
;*
;* 100000

;* 'BUFNO' CONTAINS THE STARTING ADDRESS OF THE
;* BUFFER.

PTGEN1:	MOV	#1,R3	: INITIALIZE PATRNS
	MOV	#177776,R4	
	MOV	BUFNO,R0	: GET STARTING ADRES OF BUFR
	MOV	#-20,R2	: SET COUNT
15:	MOV	R3,R1	: INITIALIZE PATRN
25:	MOV	R1,(R0)+	: GENERATE THE FIRST
	SEC		: BLOCK USING "FILL 1'S"
	ROL	R1	
	BCC	25	
	INC	R2	
	BNE	15	: ALL DONE?
	MOV	#-20,R2	
35:	MOV	R4,R1	: INITIALIZE PATRN
45:	MOV	R1,(R0)+	: GENERATE 2ND BLOCK
	CLC		: USING "FILL 0'S"
	ROL	R1	
	BCS	45	
	INC	R2	
	BNE	35	: DONE?
	MOV	#-20,R2	: SET COUNT
55:	MOV	R3,R1	: INITIALIZE PATRN
65:	MOV	R1,(R0)+	: GENERATE THE 3RD BLOCK
	ASL	R1	: USING "FLOAT A 1"
	BCC	65	
	INC	R2	
	BNE	55	: DONE?
	RTS	PC	: RETURN

;*PTGEN2
;*THIS ROUTINE GENERATES A 768 (3X256) WORDS LONG
;*BUFFER CONTAINING THE FOLLOWING PATTERNS:

F05

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
2969
2970
2971
2972
2973
2974

010272 005001
010274 013700 001446
010300 010120
010302 105201
010304 001375
010306 005001
010310 010120
010312 062701 000400
010316 103374
010320 005001
010322 010120
010324 062701 000401
010330 103374
010332 000207

```

;*FIRST BLOCK-256 WORDS 000000 COUNT PATRN-LOWER BYTE
;*                          000001 0-377
;*                          000002
;*                          :
;*                          000377
;*
;*SECOND BLOCK           000000 COUNT PATRN-HIGHER BYTE
;*                          000400 0-377
;*                          001000
;*                          :
;*                          177400
;*
;*THIRD BLOCK            000000 COUNT PATRN-HIGHER & LOWER BYTE
;*                          000401 0-377, 0-377
;*                          :
;*                          177777

```

;'BUFNO' CONTAINS THE STARTING ADDRESS OF THE BUFFER.

```

PTGEN2: CLR      R1              ;INITIALIZE PATRN

1$:      MOV      BUFNO,R0
        MOV      R1,(R0)+
        INCB     R1
        BNE      1$              ;GENERATE 1ST BLOCK USING
                                ;USING 'COUNT UP LOWER BYTE'
                                ;DONE?

2$:      CLR      R1
        MOV      R1,(R0)+
        ADD      #400,R1
        BCC      2$              ;GENERATE 2ND BLOCK
                                ;USING 'COUNT UP HIGHER BYTE'
                                ;DONE?

3$:      CLR      R1
        MOV      R1,(R0)+
        ADD      #401,R1
        BCC      3$              ;GENERATE 3RD BLOCK USING
                                ;'COUNT UP HIGHER & LOWER BYTE'
                                ;ALL DONE?
        RTS      PC              ;RETURN

```

PTGEN3
;THIS ROUTINE GENERATES A 768 (3X256) WORDS LONG
;BUFFER CONTAINING THE FOLLOWING PATTERNS:

```

;*FIRST BLOCK OF 256 WORDS: 167356 (COMPLEMENT OF 010421)
;*
;*SECOND BLOCK             177776 FLOAT A 0
;*                          177775
;*                          :
;*                          077777
;*
;*THIRD BLOCK              000377 COUNT UP HIGHER BYTE 0-377
;*                          000776 COUNT DOWN LOWER BYTE 377-0
;*                          :
;*                          177400

```

;'BUFNO' CONTAINS THE STARTING ADDRESS OF THE BUFFER.

G05

2975 010334 013700 001446
2976 010340 012702 177400
2977 010344 013701 010166
2978 010350 005101
2979 010352 010120
2980 010354 005202
2981 010356 001375
2982
2983
2984 010360 012702 177760
2985 010364 000261
2986 010366 012701 177776
2987 010372 010120
2988 010374 006101
2989 010376 103775
2990 010400 005202
2991 010402 001370
2992
2993
2994 010404 012701 000377
2995 010410 010102
2996 010412 010120
2997 010414 060201
2998 010416 022701 177777
2999 010422 001373
3000 010424 000207
3001
3002
3003
3004 010426 006
3005 010427 003
3006 010430 011
3007 010431 000
3008
3009 010432 004737 016772
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030

PTGEN3: MOV BUFNO,R0
MOV #400,R2
MOV PTRNO2,R1 ;GET PATTERN
COM R1 ;COMPLEMENT 'PAT02' PATRN
45: MOV R1,(R0)+ ;GENERATE 1ST BLOCK
INC R2
BNE 45 ;ALL DONE?
;2ND BLOCK
MOV #20,R2
75: SEC
MOV #177776,R1
85: MOV R1,(R0)+
ROL R1
BCS 85
INC R2
BNE 75 ;ALL DONE?
;3RD BLOCK
MOV #377,R1
MOV R1,R2 ;GENERATE 3RD BLOCK USING
95: MOV R1,(R0)+ ;'COUNT DOWN LOWER BYTE'
ADD R2,R1 ;'COUNT UP HIGHER BYTE'
CMP #1,R1
BNE 95 ;ALL DONE?
RTS PC ;RETURN

;SECTOR POINTER TABLE. DATA TRANSFERS ARE DONE IN BLOCKS (3 SECTORS
;EACH) IN THE CYCLIC ORDER: 0-2, 6-10, 3-5, 11-13, 0-2, AND SO ON.

SECPTR: .BYTE 6
.BYTE 3
.BYTE 11
.BYTE 0

EXT6: JSR PC,ABRT

;*****
;#TEST 7 READ, SOFTWARE COMPARE, WRITE CHECK OF THE PATTERNS

;****TEST 6 (WRITING PATTERNS ON DISK) SHOULD BE DONE BEFORE DOING****
;****THIS TEST.****

;*IN THIS TEST THE PATTERNS THAT WERE WRITTEN BEFORE (IN THE
;*PREVIOUS TEST) ARE READ BACK AND SOFTWARE COMPARED.
;*WHILE THE SOFTWARE COMPARISON IS TAKING PLACE, AN OVERLAPPING
;*WRITE CHECK IS DONE FOR THE SAME BLOCK. THE READING
;*BACK OF EACH BLOCK (4 SECTORS) IS DONE IN THE SAME
;*MANNER AS DESCRIBED IN THE BEGINNING OF PREVIOUS TEST.
;*OVERLAPPING OPERATIONS AND STAGGERING OF BLOCKS ARE DONE IN
;*A SIMILAR MANNER.
;*THE FOLLOWING IS A DESCRIPTION OF HOW ERRORS ARE HANDLED.
;*IF A 'SIN' OR 'HE' OCCURS IT IS REPORTED AND THAT BLOCK
;*IS SKIPPED. IN THIS STAGE OF TESTING THESE ERRORS SHOULD NOT
;*NORMALLY OCCUR.
;*IF A CHECKSUM ERROR OCCURS, CONTROL IS TRANSFERRED TO

```

3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055 010436 000004
3056 010440 012737 177764 001510
3057 010446 012737 177773 001512
3058 010454 012737 177742 001514
3059 010462 012737 177742 001516
3060 010470 012737 177764 001520
3061 010476 012737 177742 001522
3062 010504 012737 177152 001474
3063 010512 005037 001476
3064
3065 010516 012737 001414 001424
3066 010524 013737 001230 001450
3067
3068
3069 010532 005037 001504
3070 010536 005037 001506
3071 010542 012737 177775 001252
3072 010550 012737 177776 001254
3073 010556 012737 000003 001256
3074
3075 010564 013737 001450 001440
3076 010572 013737 001406 001442
3077
3078 010600 012737 176400 001444
3079
3080 010606 013737 001450 001432
3081 010614 012737 176400 001436
3082 010622 013737 001404 001434
3083
3084
3085 010630 000434
3086

```

*'CSEERROR'. FIRST THE CSE IS REPORTED. THE SECTOR GIVING
 *CSE IS READ AGAIN. IN ALL 3 TRIES ARE DONE. IF STILL
 *THE ERROR PREISTS THAT SECTOR IS ABORTED AND THE REST
 *OF THE SECTORS ARE READ AND CHECKED FOR CSE. IF
 *AGAIN SOME OTHER SECTOR GIVES CSE, REPORTING AND RETRIES
 *ARE DONE AGAIN.
 *NEXT SOFTWARE COMPARISON IS DONE OF THE DATA THAT WAS
 *READ BACK. ON GETTING A DATA MISCOMPARISON
 *THE RELEVANT INFO(GOOD DATA, BAD DATA, ADDRESS ETC.) IS
 *STORED. AFTER THE WHOLE BLOCK HAS BEEN CHECKED, THE DATA
 *ERROR/S IS/ARE REPORTED. THEN THE BLOCK IS READ AGAIN. IN ALL
 *THREE TRIES ARE DONE.
 *WHILE THE DATA COMPARISON (SOFTWARE) IS GOING ON THE 'WRITE
 *CHECK' IS ALSO IN PROGRESS. IF A WRITE CHECK ERROR OCCURS THE
 *CONTROL IS TRANSFERRED TO 'WCEROR'. WRITE CHECK OF THE SECTOR
 *THAT GAVE WCE IS DONE AGAIN. IN ALL THREE TRIES ARE DONE.
 *NOTE THAT EVERY WCE IS REPORTED. IF ALL THE 4 SECTOR OF
 *A BLOCK GAVE WCE'S, ALL 4 WILL BE REPORTED AND RETRIED.
 *DEPENDING ON THE NATURE OF THE PROBLEM, THE ABOVE ERRORS
 *CAN OCCUR IN ANY COMBINATION. IT IS RECOMMENDED THAT ALL
 *THE ERROR MESSAGE BE CAREFULLY CO-RELATED AND EXAMINED. IT
 *WILL PROVIDE A DEEP INSIGHT INTO THE PROBLEM.

```

*****
tST7: SCOPE
      MOV      #-14,ERCNT3      ;ALLOW 12 ERRORS AT THE MOST
      MOV      #-5,ERCNT4      ;ALLOW 5 ERRORS AT THE MOST
      MOV      #-36,ERCNT5     ;ALLOW 10 ERRORS AT THE MOST
      MOV      #-36,ERCNT6     ;ALLOW 10 ERRORS AT THE MOST
      MOV      #-14,ERCNT7     ;ALLOW 12 ERRORS AT THE MOST
      MOV      #-36,ERCNT8     ;ALLOW 10 ERRORS AT THE MOST
      MOV      #-626,INDX1     ;SET COUNT FOR 626 TRACKS
      CLR      INDX2          ;CLR COUNT FOR 4 BLOCKS ON A TRACK

      MOV      #PATO,PRSPAT    ;INITLZE PTR TO PATRN GENRTR
      MOV      DRIVAD,ADRES    ;INITLZE DRV#,ADRES

BEGIN: CLR      ERCNT1        ;IF > 0, MEANS THAT RETRIES
      CLR      ERCNT2        ;DONE AFTER CSE OR CSE CHKD
      MOV      #-3,RETRY1     ;RETRY COUNT FOR CSE
      MOV      #-2,RETRY2     ;RETRY COUNT FOR SFTWRE MISCOMP'N
      MOV      #3,RETRY3      ;RETRY COUNT FOR WCE

      MOV      ADRES,WDSKAD    ;DISK ADRES TO WRT CHK WITH
      MOV      PBUF1,WBUSAD    ;USE THIS BUFR 1 TO WRT CHK
                                ;OR THE BUFR INDICATED BY THE USER
      MOV      #-1400,WWRDCN   ;WRT CHK 1 BLOCK=3SECS=1400 WRDS

READ:  MOV      ADRES,DSKADR    ;DISK ADRES TO READ FROM
      MOV      #-1400,WRCNT    ;1 BLOCK = 3 SECTORS = 1400 WRDS
      MOV      PBUF0,BUSADR    ;USE 'IOBUF0' TO READ INTO
                                ;OR THE BUFR INDICATED BY THE USER

      BR       RDAGAIN

```

MC-11 CRK-E. R11-RK05 DYNAMIC TEST
 CRFILE P11 26-APR-77 12:27

MCY11 30(1046) 14-JUL-77 08:03 PAGE 60
 17 READ, SOFTWARE COMPARE, WRITE CHECK OF THE PATTERNS

3087	010632	104415			LUPSIN: CON.RESET	
3088	010634	104416			DRV.RESET	
3089	010636	000401			BR RDAGAIN	
3090						
3091	010640	104415			LUPHE: CON.RESET	
3092						
3093	010642	013777	001432	170614	RDAGAIN: MOV DSKADR,ARKDA	;READ FROM THIS DSK-ADRES
3094	010650	013777	001436	170602	MOV WRCNT,ARKWC	;THIS # OF WORDS
3095	010656	013777	001434	170576	MOV BUSADR,ARKBA	;INTO THIS BUFR
3096						
3097	010664	012777	000405	170564	MOV #405,ARKCS	;READ,SSE,GO
3098						
3099						
3100	010672	013737	001406	001446	MOV PBUF1,BUFNO	;SET UP STARTING ADRES
3101	010700	017737	170520	001430	MOV APRSPAT,PGSUBR	
3102	010706	004777	170516		JSR PC,PGSUBR	;GO GENERATE A BUFFER
3103						;OF 1400 WORDS USING THIS
3104						;PATRN GENRATR
3105						
3106	010712	104421			CON.RDY	;DONE WITH PATRN GENRTNG.
3107						;WAIT FOR CNT RDY TO SET
3108						; (FROM PREVIOUS READ)
3109						
3110						;CNT RDY SET
3111	010714	032777	040000	170534	BIT #BIT14,ARKCS	;HARD ERROR?
3112	010722	001416			BEQ NOHE	
3113						
3114	010724	012737	010640	001110	MOV #LUPHE,\$LPERR	
3115	010732	004737	016214		JSR PC,GETINF	
3116	010736	104023			ERROR 23	;HARD ERROR
3117	010740	104415			CON.RESET	
3118	010742	005237	001510		INC ERCNT3	;ALLOW 12 ERRORS AT MOST
3119	010746	001002			BNE IS	
3120	010750	000137	012170		JMP EXT7	;IF MORE, EXIT
3121	010754	000137	011512		JMP FINISH	
3122	010760	032777	001000	170464	IS: BIT #BIT9,ARKDS	;SIN SET?
3123	010766	001417			BEQ NOSIN	;NO
3124						
3125	010770	012737	010632	001110	MOV #LUPSIN,\$LPERR	
3126	010776	004737	016214		JSR PC,GETINF	
3127	011002	104011			ERROR 11	;SIN ON READ
3128	011004	104415			CON.RESET	
3129	011006	104416			DRV.RESET	
3130	011010	005237	001512		INC ERCNT4	;ALLOW 5 ERRORS AT MOST
3131	011014	001002			BNE IS	
3132	011016	000137	012170		JMP EXT7	;IF MORE, EXIT
3133	011022	000137	011512		JMP FINISH	
3134						
3135	011026	005737	001504		NOSIN: TST ERCNT1	;CHECKING CSE FOR 1ST TIME
3136	011032	001031			BNE WRTCHK	;NO BRANCH
3137	011034	005237	001504		INC ERCNT1	;INDICATE THAT CSE HAS BEEN
3138						;CHECKED ONCE
3139						
3140	011040	032777	000002	170406	BIT #BIT1,ARKER	;CHECK SUM EROR?
3141	011046	001423			BEQ WRTCHK	
3142	011050	012737	010532	001110	MOV #BEGIN,\$LPERR	

3143	011056	004737	016214		JSR	PC,GETINF	
3144	011052	013237	001252	001202	MOV	RETRY1,SREG10	;GET THE RETRY #
3145	011070	062737	000004	001202	ADD	#4,SREG10	;SAVE IT FOR TYPEOUT
3146	011076	104024			ERROR	24	;CSE
3147	011100	005237	001514		INC	ERCNT5	;ALLOW 10 ERRORS AT MOST
3148	011104	001002			BNE	1\$	
3149	011106	000137	012170		JMP	EXT7	;IF MORE, EXIT
3150	011112	000137	011654		JMP	CSECR0	;GO, SERVICE CSE
3151							
3152	011116	005037	001506		WRTCHK: CLR	ERCNT2	;CLR FLAG INDICATING SOFTWARE
3153							;COMPARE DONE
3154	011122	022737	000003	001256	CMP	#3,RETRY3	;WRT CHK DONE BEFORE OR
3155	011130	001016			BNE	SFTCMP	;IT'S 1ST TIME?
3156							;IF DONE,BRANCH OTHERWISE DO IT
3157	011132	013777	001440	170324	WCAGAIN: MOV	WDSKAD,ARKDA	;WRT CHK FROM THIS DSK-ADRES
3158	011140	013777	001444	170312	MOV	WWRDCN,ARKWC	;THIS # FO WORDS
3159	011146	013777	001442	170306	MOV	WBUSAD,ARKBA	;WITH THIS BUFFER
3160							
3161	011154	012777	000407	170274	MOV	#407,ARKCS	;WRT CHK,GO,SSE
3162							
3163	011162	005337	001256		DEC	RETRY3	;INDICATE WRT CHK DONE
3164							
3165	011166	005737	001506		SFTCMP: TST	ERCNT2	;SOFTWARE CMPARE DONE ONCE BEFORE
3166	011172	001060			BNE	WCREPT	;IF SOFTWARE CMPARE HAS BEEN DONE
3167							;ONCE DON'T DO IT AGAIN. OTHERWISE,
3168	011174	005237	001506		INC	ERCNT2	;DO IT. INDICATE IT IS DONE.
3169							;MORE THAN ONCE BEFORE.
3170							;IF THIS IS 1ST TIME THRU &
3171							;WRT CHK WAS DONE ONCE BEFORE
3172							;DO SOFTWARE COMPARISON OF
3173							;THE DATA THAT WAS READ FROM
3174							;THE DISK
3175							
3176	011200	012702	001266		MOV	#BUFR,R2	;INITLZE PTR TO BUFR STORING
3177							;ADRES OF BAD DATA
3178	011204	012703	001320		MOV	#BUFR1,R3	;STORE EXPCTD DATA STARTING HERE
3179	011210	012704	001352		MOV	#BUFR2,R4	;STORE RECVD (BAD) DATA
3180							;STARTING HERE
3181							
3182	011214	005037	001500		CLR	INDX3	;CLR FLAG INDICATING MISCMPRE
3183							
3184	011220	013700	001404		COMPAR: MOV	PBUFO,R0	;INITLZE PTR TO 'RECVD DATA' BUFR
3185	011224	012737	177764	001502	MOV	#-14,INDX4	;STORE AND REPORT 12 OR LESS DATA ERRORS
3186	011232	013701	001406		MOV	PBUF1,R1	;INITLZE PTR TO 'EXPCTD DATA' BLFR
3187	011236	012737	176400	001260	MOV	#-1400,INADR	;SET COUNT
3188	011244	021011			CMPAGAN: CMP	(R0),(R1)	;CORRECT WORD READ FROM DISK
3189	011246	001402			BEQ	1\$	
3190	011250	000137	012016		JMP	MISCMPE	;BRANCH IF MISCMPRE ERROR
3191	011254	005720			1\$: TST	(R0)+	;INCRMNT PTRS
3192	011256	005721			TST	(R1)+	;TO NXT WORDS
3193	011260	005237	001260		INC	INADR	;DONE WITH CMPRISON?
3194	011264	001367			BNE	CMPAGAN	;IF NOT, COMPARE THE RES
3195							
3196	011266	005737	001500		TST	INDX3	;WAS THERE A BAD DATA WORD
3197							; (EVEN AFTR RETRYING)
3198	011272	001420			BEQ	WCREPT	;NO, BRANCH

K05

3199	011274	012737	010606	001110	REPMS:	MOV	#READ,\$LPERR	
3200	011302	104025				ERROR	25	; DATA ERROR
3201	011304	005237	001516			INC	ERCNT6	; ALLOW 10 ERRORS AT MOST
3202	011310	001002				BNE	1\$	
3203	011312	000137	012170			JMP	EXT7	; IF MORE, EXIT
3204	011316	005737	001254		1\$:	TST	RETRY2	
3205	011322	001404				BEQ	WCREPT	
3206	011324	005237	001254			INC	RETRY2	
3207	011330	000137	010606			JMP	READ	
3208								
3209	011334	104421			WCREPT:	CON.RDY		; WAIT FOR CNTRL RDY FROM
3210								; PREVIOUS WRT CHK
3211	011336	022737	177776	001254		CMP	#-2,RETRY2	; IF THERE WAS A RETRY AFTER MISC
3212								; -OMPARISON, DO WRT CHK AGAIN
3213	011344	001417				BEQ	ERWCHK	
3214	011346	000401				BR	LUPWCE+2	
3215	011350	104415			LUPWCE:	CON.RESET		
3216	011352	013777	001440	170104		MOV	WDSKAD,ARKDA	; WRT CHK WITH THIS DSK-ADRES
3217	011360	013777	001444	170072		MOV	WWRDCN,ARKWC	; THIS # OF WORDS
3218	011366	013777	001442	170066		MOV	WBUSAD,ARKBA	; THIS BUS ADRES
3219	011374	012777	000407	170054		MOV	#407,ARKCS	
3220								
3221	011402	104421				CON.RDY		
3222	011404	012737	011350	001110	ERWCHK:	MOV	#LUPWCE,\$LPERR	
3223	011412	032777	040000	170032		BIT	#BIT14,ARKDS	; HARD ERROR?(FROM WRT CHK)
3224	011420	001410				BEQ	XHE	; NO,BRANCH
3225								
3226	011422	004737	016214			JSR	PC,GETINF	
3227	011426	104026				ERROR	26	; HE ON WRT CHK
3228	011430	005237	001520			INC	ERCNT7	; ALLOW 12 ERRORS AT MOST
3229	011434	001002				BNE	XHE	
3230	011436	000137	012170			JMP	EXT7	; IF MORE, EXIT
3231								
3232	011442	032777	000001	170004	XHE:	BIT	#BIT0,ARKER	; WRITE CHECK ERROR?
3233	011450	001420				BEQ	FINISH	
3234	011452	004737	016214			JSR	PC,GETINF	
3235	011456	012737	000003	001202		MOV	#3,\$REG10	; GET TRY #
3236	011464	163737	001256	001202		SUB	RETRY3,\$REG10	; SAVE IT FOR TYPEOUT
3237	011472	104027				ERROR	27	; WRT CHK ERROR
3238	011474	005237	001522			INC	ERCNT8	; ALLOW 10 ERRORS AT MOST
3239	011500	001002				BNE	1\$	
3240	011502	000137	012170			JMP	EXT7	; IF MORE, EXIT
3241	011506	000137	012062		1\$:	JMP	WCEROR	
3242								
3243								
3244								; THERE WAS NO WCE. DONE
3245								; WITH ALL CHECKING FOR SOFT
3246								; ERORS, ETC. MODIFY PARAMETERS
3247								; TO CHECK NXT BLOCK ON
3248								; THE DISK
3249								
3250	011512	022737	001422	001424	FINISH:	CMP	#PAT3,PRSPAT	; FIND OUT THE NXT PATRN GENRATR
3251	011520	001404				BEQ	1\$	; TO USE FOR GENRATING THE
3252	011522	062737	000002	001424		ADD	#2,PRSPAT	; BUFR, STORE POINTER TO
3253	011530	000403				BR	2\$	; GENRATR ROUTINE IN 'PRSPAT'
3254	011532	012737	001414	001424	1\$:	MOV	#PAT0,PRSPAT	; NOTE THER R 4 PAT-GENRATRS

3311	011712	010102				MOV	R1,R2	
3312	011714	042702	177760			BIC	#177760,R2	
3313	011720	001002			75:	BNE	85	
3314	011722	000137	011116			JMP	WRTCHK	
3315	011726	162702	000003		85:	SUB	#3,R2	
3316	011732	100372				BPL	75	
3317								
3318	011734	010100			65:	MOV	R1,R0	
3319	011736	012737	177775	001252		MOV	#-3,RETRY1	
3320	011744	000403				BR	35	
3321								
3322	011746	012737	177776	001252	25:	MOV	#-2,RETRY1	;ALLOW 2 MORE TRIES FOR ;THE CSE ON SAME DISK-ADRES
3323								
3324								
3325	011754	010037	001432		35:	MOV	R0,DSKADR	;SAVE DSK-ADRES FOR DOING ;THE RETRY-READ
3326								
3327	011760	163700	001450			SUB	ADRES,R0	;MODIFY THE
3328	011764	005200				INC	R0	;BUS ADRES & WORD COUNT
3329	011766	005300			45:	DEC	R0	;TO BE USE ON RETRY-
3330	011770	001407				BEQ	55	;READ
3331	011772	062737	001000	001434		ADD	#1000,BUSADR	
3332	012000	062737	000400	001436		ADD	#400,WORDCNT	
3333	012006	000767				BR	45	
3334								
3335	012010	104415			55:	CON.RESET		;CLR THE CSE IN RKER
3336	012012	000137	010642			JMP	RDAGAIN	;GO BACK, READ AGAIN
3337								;RETRY
3338								
3339								
3340								
3341								
3342								
3343								
3344								
3345								
3346								
3347								
3348	012016	104421						
3349								
3350	012020	032777	040000	167430		BIT	#BIT14,ARKCS	
3351	012026	001401				BEQ	15	
3352	012030	104415				CON.RESET		
3353								
3354	012032	010022			15:	MOV	R0,(R2)+	;CLR WCE IN RKER
3355								;BUT STILL DATA ERROR ENTER HERE
3356	012034	012123				MOV	(R1)+,(R3)+	;STORE MEM ADRES WHERE
3357	012036	012024				MOV	(R0)+,(R4)+	;DATA ERROR OCCURED
3358	012040	005237	001500			INC	INDX3	;SAVE EXPCTD DATA
3359								;SAVE DATA RECVD (BAD)
3360	012044	005237	001502			INC	INDX4	;INDICATE MISCMPRISON
3361	012050	001402				BEQ	45	;STORE ONLY 12(DEC) ERRORS
3362								;IF 12 ERRORS, GO REPORT THEM
3363	012052	000137	011244			JMP	CMRAGAN	;GO BACK & CMPARE THE REST
3364								
3365	012056	000137	011274		45:	JMP	REPMSC	
3366								

;MISCOMP

;THIS IS THE ENTRY POINT
;FOR SERVICE ROUTINE, FOR
;DATA ERROR (MISCOMPARISON)
;ON SOFTWARE COMPARISON
;OF DATA READ FROM THE DISK BLOCK

MISCOMP: CON.RDY

;WAIT FOR CNTRL RDY FROM
;PREVIOUS WRT CHK

```

3367
3368
3369
3370
3371
3372
3373 012062 005737 001256 WCEROR: TST RETRY3 ;THIS THE ENTRY POINT FOR
3374 012066 003035 BGT CLRERR ;EROR SERVICE ON GETTING A
;WRITE CHECK EROR.
3375
3376
3377 012070 012737 000003 001256 MOV #3,RETRY3 ;DONE 3 TRIES?
;IF NOT SKIP, OTHERWISE REPORT
;W C EROR
3378
3379 012076 017700 167362 MOV DRKDA,R0 ;IF WCE WAS IN THE LAST SECTOR
3380 012102 010002 MOV R0,R2 ;OF THE BLOCK, NO MORE SECS
3381 012104 042702 177760 BIC #177760,R2 ;TO CHECK, GO TO 'FINISH'
3382 012110 001002 3$ BNE 4$ ;IF IT WASN'T LAST SECTOR OF THE
3383 012112 000137 011512 JMP FINISH ;BLOCK, THEN CHECK REMAINING
3384 012116 162702 000003 4$ SUB #3,R2 ;SECTORS. (STARTING FROM THE
3385 012122 100372 BPL 3$ ;SEC AFTER THE ONE GIVING WCE)
3386 012124 010001 1$ MOV R0,R1 ;SAVE DISK ADRES
3387 012126 163700 001440 SUB WDSKAD,R0
3388 012132 010137 001440 MOV R1,WDSKAD ;GET SAVED DISK ADRES
3389 012136 005200 INC R0
3390 012140 005300 2$ DEC R0
3391 012142 001407 BEQ CLRERR
3392 012144 062737 001000 001442 ADD #1000,WBUSAD ;FORM THE NEW BUS ADRES
3393 012152 062737 000400 001444 ADD #400,WWORDCN ;FORM THE NEW WORD COUNT
3394 012160 000767 BR 2$
3395 012162 104415 CLRERR: CON.RESET
3396 012164 000137 011132 JMP WCAGAIN
3397
3398 012170 004737 016772 EXT7: JSR PC,ABRT
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422 012174 000004

```

;TEST 10 WRITE, WRITE CHECK ON CYLINDERS 127, 128
;THIS TEST WRITES 12 UNIQUE PATTERNS (ONE FOR EACH
;SECTOR) ON CYLINDERS 127 AND 128, THEN WRITE
;CHECK IS DONE TO SEE IF THEY WERE WRITTEN
;CORRECTLY. IT SHOULD BE NOTED THAT THERE IS
;CHANGE IN 'WRITE' CURRENT AT THIS CYLINDER.
;PATTERNS ARE RELOCATED ON THE CYLINDERS AFTER EACH
;WRITE/WRITE CHECK CYCLE. THUS THE FIRST TIME
;PATTERN 'SP1' IS WRITTEN ON SECTOR 0, PATTERN 'SP2' ON
;SECTOR 2, ETC. AFTER THIS WRITE/WRITE CHECK CYCLE
;IS OVER PATTERN 'SP2' IS WRITTEN ON SECTOR 0, PATTERN
;: 'SP3' ON SECTOR 1 AND SO ON. THIS WRITE/WRITE
;CHECK CYCLE IS REPEATED 12 TIMES, THUS
;THE LAST WRITE/WRITE CHECK IS DONE USING
;PATTERN 'SP12' ON SECTOR 0, PATTERN 'SP1' IS
;WRITTEN ON SECTOR 1, PATTERN 'SP2' ON SECTOR 2,
;ETC. IF YOU WANT TO WRITE ANY OTHER PATTERNS
;FILL IN THE PATTERNS YOU WANT IN LOCATIONS
;'SP1', 'SP2',...ETC.

;TEST10: SCOPE

3423											
3424	012176	012737	177764	001504		MOV	#-14,ERCNT1	:	ALLOW 12 ERRORS AT MOST		
3425	012204	012737	177764	001506		MOV	#-14,ERCNT2	:	ALLOW 12 ERRORS AT MOST		
3426	012212	012737	177723	001510		MOV	#-55,ERCNT3	:	ALLOW 15 ERRORS AT MOST		
3427	012220	012702	012702			MOV	#SP1,R2	:	INITIALIZE POINTER TO PATRN		
3428	012224	010201			DOWRT:	MOV	R2,R1				
3429	12236	012700	007740			MOV	#7740,R0	:	SET UP CYL ADRES BITS (127)		
3430	012332	053700	001230			BIS	DRIVAD,R0	:	SET UP DRIVE # BITS		
3431	012336	005003			WRL01:	CLR	R3				
3432	012240	104415			WRERR:	CON.RESET					
3433											
3434	012242	010077	167216		WRL0:	MOV	R0,ARKDA	:	ADRES THE DRIVE		
3435	012246	012777	177400	167204		MOV	#-400,ARKWC	:	WRITE 1 SECTOR		
3436	012254	010177	167202			MOV	R1,ARKBA	:	USE THIS PATTERN		
3437	012260	012777	004003	167170		MOV	#4003,ARKCS	:	WRITE, GO		
3438	012266	104421				CON.RDY					
3439	012270	032777	140000	167160		BIT	#140000,ARKCS	:	ANY ERROR?		
3440	012276	001414				BEG	45				
3441	012300	012737	012240	001110		MOV	WRERR,SLPERR	:	SET ADRES FOR LOOPING ON ERROR		
3442	012306	004737	016162			JSR	PC,GT4RG	:	GET TKCS, ER, DS, DA		
3443	012312	104021				ERROR	21	:	ERROR OCCURRED ON DOING A WRITE		
3444	012314	104415				CON.RESET		:	CLEAR THE ERROR		
3445	012316	005237	001504			INC	ERCNT1	:	ALLOW 12 ERRORS ONLY		
3446	012322	001002				BNE	45				
3447	012324	000137	012732			JMP	EXT10				
3448											
3449	012330	005200			45:	INC	R0				
3450	012332	005203				INC	R3	:	KEEP COUNT		
3451	012334	022701	012730			CMP	#SP12,R1	:	USE PATTERNS IN A CYCLIC FASHION		
3452	012340	001002				BNE	35				
3453	012342	012701	012700			MOV	#SP1-2,R1				
3454	012346	005721			35:	TST	(R1)+	:	INCREMENT POINTERS TO NEXT PATTERN		
3455	012350	020327	000014			CMP	R3,#14	:	DONE SURFACE 0?		
3456	012354	002732				BLT	WRL0	:	NO		
3457	012356	001005				BNE	25	:	YES, IF CHANGING HEADS		
3458	012360	010201				MOV	R2,R1				
3459	012362	042700	000017			BIC	#17,R0	:	SET UP CORRECT ADDRESS ETC.		
3460	012366	052700	000020			BIS	#20,R0				
3461											
3462	012372	020327	000030		25:	CMP	R3,#30	:	DONE WITH WRITING SURFACE 1?		
3463	012376	001321				BNE	WRL0	:	NO, BRANCH		
3464											
3465	012400	032700	007700		WRHI:	BIT	#7700,R0	:	DONE WITH BOTH CYLINDERS - 127 & 128?		
3466	012404	001405				MOV	DOWCHK	:	YES		
3467	012406	012700	010000			MOV	#10000,R0	:	NO, DO CYLINDER 128		
3468	012412	053700	001230			BIS	DRIVAD,R0				
3469	012416	000707				BR	WRL01	:	GO BACK		
3470								:	CYLINDERS 127 AND 128 HAVE BEEN		
3471											
3472											
3473	012420	010201			DOWCHK:	MOV	R2,R1	:	WRITTEN, NOW DO WRITE CHECK		
3474	012422	012737	177775	001252		MOV	#-3,RETRY1	:	INITIALIZE POINTER TO FIRST PATTERN		
3475	012430	012700	010000			MOV	#10000,R0	:	RETRY COUNT		
3476	012434	053700	001230			BIS	DRIVAD,R0	:	DO CYLINDER 128 FIRST		
3477	012440	005003			WCHI1:	CLR	R3				
3478	012442	010077	167016		WCERR:	MOV	R0,ARKDA	:	ADRES THE DRIVE		

```

3479
3480 012446 012777 177400 167004 WCHI: MOV      #400,2RKWC      ;WRITE CHECK 1 SECTOR
3481 012454 010177 167002      MOV      R1,2RKBA      ;WITH THIS PATTERN
3482 012460 012777 004007 166770      MOV      #4007,2RKCS      ;WRITE CHECK, GO
3483 012466 104421      CON.RDY
3484
3485 012470 032777 040000 166754      BIT      #40000,2RKDS      ;HE?
3486 012476 001406      BEQ      1$      ;NO
3487 012500 004737 016214      JSR      PC,GETINF
3488 012504 104026      ERROR      26      ;HE ON DOING WRT CHK
3489 012506 005237 001506      INC      ERCNT2      ;ALLOW 12 ERRORS ONLY
3490 012512 001507      BEQ      EXT10      ;IF MORE, EXIT
3491 012514 032777 000001 166732 1$: BIT      #BIT0,2RKER      ;WCE?
3492 012522 001425      BEQ      4$      ;NO
3493 012524 012737 012442 001110      MOV      #WCERR,SLPERR      ;SET ADRES FOR LOOPING ON ERROR
3494 012532 004737 016214      JSR      PC,GETINF      ;GET INFO ON ERROR
3495 012536 013737 001252 001202      MOV      RETRY1,$REG10
3496 012544 062737 000004 001202      ADD      #4,$REG10      ;WCE ON DOING WRITE CHECK, WITH
3497 012552 104027      ERROR      27      ;PATTERN STORED IN R1
3498 012554 005237 001252      INC      RETRY1      ;DO 3 TIMES IN ALL
3499 012560 001330      BNE      WCERR
3500 012562 005237 001510      INC      ERCNT3      ;ALLOW 15 ERRORS ONLY
3501 012566 001461      BEQ      EXT10      ;IF MORE, EXIT
3502 012570 012737 177775 001252      MOV      #-3,RETRY1
3503
3504 012576 005200      4$: INC      R0      ;KEEP TRACK OF DISK-ADRES
3505 012600 005203      INC      R3      ;AND COUNT
3506 012602 022701 012730      CMP      #SP12,R1      ;USE PATTERNS IN CYCLIC
3507 012606 001002      BNE      3$
3508 012610 012701 012700      MOV      #SP1-2,R1      ;FASHION
3509 012614 005721 000014      3$: TST      (R1)+      ;INCREMENT POINTER TO NEXT PATTERN
3510 012616 020327      CMP      R3,#14      ;DONE SURFACE 0?
3511 012622 002711      BLT      WCHI      ;NO
3512 012624 001005      BNE      2$
3513 012626 010201      MOV      R2,R1      ;IF CHANGING HEADS (0-1), SET CORRECT
3514 012630 042700 000017      BIC      #17,R0      ;ADRES BITS
3515 012634 052700 000020      BIS      #20,R0
3516
3517 012640 020327 000030      2$: CMP      R3,#30      ;DONE WRITE CHECKING SURFACE 1?
3518 012644 001300      BNE      WCHI      ;NO, GO BACK
3519
3520 012646 032700 007700      WCL0: BIT      #7700,R0      ;DONE BOTH CYLINDERS - 127, 128?
3521 012652 001005      BNE      REPEAT      ;YES, BRANCH
3522 012654 012700 007740      MOV      #7740,R0      ;DO CYLINDER 127 NOW
3523 012660 053700 001230      BIS      DRIVAD,R0
3524 012664 000665      BR      WCHI1
3525
3526
3527 012666 005722      REPEAT: TST      (R2)+      ;RELOCATE THE PATTERNS ON THE
3528 012670 020227 012732      CMP      R2,#SP12+2      ;CYLINDERS AND DO IT AGAIN
3529 012674 001420      BEQ      T$11      ;EXIT
3530 012676 000137 012224      JMP      DOWRT      ;THIS TEXT)
3531
3532
3533
3534 ;PATTERNS TO BE USED

```

MC-11 DZRKL-E, RK11-RK05 DYNAMIC TEST
DZRKLE P11 26-APR-77 12:27

MACY11 30(1046) 14-JUL-77 08:03 PAGE 68
T10 WRITE, WRITE CHECK ON CYLINDERS 127, 128

3535
3536 012702 177777
3537 012704 052525
3538 012706 111111
3539 012710 010421
3540 012712 102041
3541 012714 010101
3542 012716 040201
3543 012720 000401
3544 012722 031463
3545 012724 070707
3546 012726 007417
3547 012730 041020

SP1: .WORD 177777
SP2: .WORD 052525
SP3: .WORD 111111
SP4: .WORD 010421
SP5: .WORD 102041
SP6: .WORD 010101
SP7: .WORD 040201
SP8: .WORD 000401
SP9: .WORD 031463
SP10: .WORD 070707
SP11: .WORD 007417
SP12: .WORD 041020

; IF YOU WANT TO WRITE ANY
; OTHER PATTERNS, CHANGE THESE
; 12 LOCATIONS TO ANY PATTERN
; YOU WANT.

3548
3549 012732 004737 016772
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566

EXT10: JSR PC,ABRT

; *TEST 11 SEEK FUNCTION TIMER
; *SEEK TIMER
; *IN THIS PART OF THE PROGRAM SEEKS ARE TIMED BETWEEN A PARTICULAR SET
; *OF CYLINDERS, BOTH IN THE FORWARD DIRECTION (0-312) AND REVERSE(312-0 .
; *****CAUTION*****
; *IT SHOULD BE NOTED THAT THE SECTOR COUNTER (IN RKDS) IS USED AS THE REAL
; *TIME CLOCK TO DO THE SEEK TIMING. FOR THE TIMES TO BE RELIABLE, THE
; *SECTOR COUNTER SHOULD BE ACCURATE WITHIN THE SPECIFICATIONS OF THE DISK
; *SPEED: 1500 RPM = 40 MILI SECS/REV =3.33 MILI SECS/SECTOR.
; *VARIATION: +-30 RPM
; *****

3567 012736 000004
3568 012740 032777 000100 166172
3569 012746 001002
3570 012750 000137 014106
3571 012754 104415
3572 012756 104416
3573
3574 012760 012737 177771 001252
3575
3576
3577 012766 104401 012774
3578 012772 000424
3579
3580 013044
3581 013044 104401 013052
3582 013050 000421
3583
3584 013114
3585 013114 104401 013122
3586 013120 000421
3587
3588 013164
3589
3590 013164 012737 001542 001260

†ST11: SCOPE
BIT #SW6,2SWR ;INHIBIT TIMER?
BNE .+6
JMP PLTGRPH
CON.RESET
DRV.RESET
MOV #-7,RETRY1 ;COUNT FOR 7 DIFRNT SEEK TIMES
;TO BE RECORDED
TYPE 65\$;TYPE ASCIZ STRING
BR 64\$;GET OVER THE ASCIZ
;;65\$: .ASCIZ <15><12>/SEEK TIME SCALE FACTOR=0.01 MILI SECS
64\$:
TYPE 67\$;TYPE ASCIZ STRING
BR 66\$;GET OVER THE ASCIZ
;;67\$: .ASCIZ <15><12><12>/ # OF SEEK # OF SEEK/
66\$:
TYPE 69\$;TYPE ASCIZ STRING
BR 68\$;GET OVER THE ASCIZ
;;69\$: .ASCIZ <15><12>/ SEEKS TIME SEEKS TIME/<15><12>
68\$:
MOV #SIAD,INADR ;INITLZE PTR TO INNER ADRES

```

3591 013172 012737 001524 001262      MOV      #50AD,OUTADR      ;INITLZE PTR TO OUTER ADRES
3592
3593 013200 017777 166056 166256 REPTIM: MOV      @OUTADR,@RKDA      ;POSITION HEADS TO OUTER CYLINDER
3594                                ;BEFORE STARTING TO TIME
3595 013206 053777 001230 166250      BIS      DRIVAD,@RKDA      ;SET DRIVE # BITS
3596 013214 012777 000011 166234      MOV      #11,@RKCS      ;SEEK GO
3597 013222 104421                CON.RDY      ;WAIT FOR CNTRL RDY
3598 013224 104422                TST.RWS      ;WAIT FOR R/W/S RDY
3599
3600 013226 005037 001474                CLR      INDX1      ;INDX1 = 0, GOING IN; OTHERWISE OUT
3601 013232 012704 027026                MOV      #BUFR10,R4      ;STORE FWRD SEEK TIMES IN THIS BUFF
3602 013236 012705 027426                MOV      #BUFR11,R5      ;STORE REVRSE "
3603 013242 012737 177634 001476      MOV      #-144,INDX2      ;SET COUNT FOR # OF SEEKS
3604
3605
3606 013250 013702 001464                BEGSK:  MOV      RKDA,R2
3607 013254 005737 001474                TST      INDX1      ;GOING FWRD OR REVRSE?
3608 013260 001005                BNE      IS      ;REVRSE, BRANCH
3609
3610 013262 017712 165772                MOV      @INADR,@R2      ;FWRD, SET INNER CYL ADRES
3611 013266 053712 001230                BIS      DRIVAD,@R2      ;SET DRIVE # BITS
3612 013272 000404                BR      2$
3613
3614 013274 017712 165762                1$:  MOV      @OUTADR,@R2      ;SET OUTER CYL ADRES
3615 013300 053712 001230                BIS      DRIVAD,@R2      ;SET DRIVE # BITS
3616 013304 004737 014776                2$:  JSR      PC,@TIMSEK      ;GO TIME THE SEEK FROM CYLINDER
3617                                ;0 TO THE ABOVE CYL. RETURN WITH
3618                                ;R3 CONTAINING THE TIME (MS, SCALE
3619                                ;FACTOR= 0.01) REQUIRED FOR THE SEEK.
3620 013310 005737 001474                TST      INDX1
3621 013314 001004                BNE      3$
3622 013316 010324                MOV      R3,(R4)+      ;STORE TIME TAKEN FOR FWRD SEEK
3623 013320 005237 001474                INC      INDX1      ;SET FLAG FOR DOING REVRSE SEEK
3624 013324 000751                BR      BEGSK      ;GO DO IT
3625
3626 013326 010325                3$:  MOV      R3,(R5)+      ;STORE TIME TAKEN FOR REVRSE SEEK
3627 013330 005037 001474                CLR      INDX1      ;CLR FLG FOR DOING FWRD SEEK
3628
3629 013334 005237 001476                INC      INDX2      ;RECORDED 144 SEEK TIMES
3630 013340 001343                BNE      BEGSK      ;IF NOT, GO BAK
3631
3632                                ;AT THIS POINT 100 SEEKS HAVE BEEK PERFORMED BETWEEN TWO
3633                                ;CYLINDERS (FORWARD & REVERSE DIRECTION). FORWARD SEEK
3634                                ;TIMES ARE STORED IN TABLE STARTING AT 'BUFR10' REVERSE
3635                                ;STARTING AT 'BUFR11'. THE FOLLOWING CODE FINDS OUT THE
3636                                ;NUMBERS OF TIMES A PARTICULAR 'SEEK TIME' WAS OBTAINED.
3637                                ;EXAMPLE: OUT OF 100 TIMES THE SEEK WAS DONE BETWEEN
3638                                ;CYLINDER 0 & LAST.
3639                                ;70 TIMES IT TOOK 95 MILI SECS
3640                                ;20 TIMES IT TOOK 85 MILI SECS
3641                                ;10 TIMES IT TOOK 100 MILI SECS
3642                                ;THIS INDICATES HOW CONSISTENT WAS THE SEEKING TIME.
3643                                ;NOTE THAT ONLY 5 DIFFERENT "SEEK TIMES" WILL BE TYPED
3644                                ;OUT.
3645
3646                                ;SORTING ROUTINE

```

```

3647
3648 013342 012705 177776 SORT: MOV # -2,R5 ;COUNT FOR FRWRD, REVRSE
3649 013346 012737 027026 001162 MOV #BUFR10,$REG0 ;INTLZE PTR TO 'SEEK TIME'
3650 013354 012737 027030 001164 MOV #BUFR10+2,$REG1
3651 013362 012737 026526 001166 MOV #BUFR4,$REG2
3652 013370 012737 026606 001170 MOV #BUFR5,$REG3 ;INTLZE PTR TO '# OF TIMES'
3653
3654 013376 013700 001162 1$: MOV $REG0,R0 ;PTR T 'SEEK TIME'
3655 013402 013701 001164 MOV $REG1,R1
3656 013406 012702 177635 MOV #-143,R2 ;COUNT FOR 143 ITEMS TO SORT
3657 013412 005003 CLR R3
3658
3659 013414 021011 2$: CMP (R0),(R1) ;SORT THE ITEMS & PUT THEM
3660 013416 003404 BLE 3$ ;IN DESCENDING ORDER
3661 013420 011004 MOV (R0),R4 ;LARGER ITEMS AT TOP OF LIS*
3662 013422 011110 MOV (R1),(R0) ;SMALLER AT THE BOTTOM
3663 013424 010411 MOV R4,(R1)
3664 013426 005203 INC R3
3665 013430 005720 3$: TST (R0)+
3666 013432 005721 TST (R1)+
3667 013434 005202 INC R2
3668 013436 001366 BNE 2$
3669 013440 005703 TST R3 ;SORTED ALL ITEMS?
3670 013442 001355 BNE 1$ ;IF NOT LOOP BACK
3671
3672 013444 013700 001162 MOV $REG0,R0 ;PTR TO 'SEEK TIME'
3673 013450 013701 001166 MOV $REG2,R1 ;SAVE 'SEEK TIME' HERE
3674 013454 013702 001170 MOV $REG3,R2 ;SAVE '# OF TIMES' HERE
3675 013460 010204 MOV R2,R4
3676 013462 005024 CLR (R4)+ ;CLR OUT 5 WORDS OF
3677 013464 005024 CLR (R4)+ ;'# OF TIMES' BUFR
3678 013466 005024 CLR (R4)+
3679 013470 005024 CLR (R4)+
3680 013472 005024 CLR (R4)+
3681 013474 012703 177773 MOV #-5,R3 ;SAVE ONLY 5 DIFRNT 'SEEK TIMES'
3682
3683 013500 011011 MOV (R0),(R1) ;FIND OUT THE '# OF TIMES'
3684 013502 012703 177634 MOV #-144,R3 ;EACH 'SEEK TIME' WAS
3685 013506 022011 4$: CMP (R0)+,(R1)
3686 013510 001411 BEQ 5$ ;OBTAINED
3687
3688 013512 005721 TST (R1)+
3689 013514 016011 177776 MOV -2(R0),(R1) ;SAVE 'SEEK TIME'
3690 013520 005722 TST (R2)+
3691 013522 012712 000001 MOV #1,(R2) ;KEEP '# OF TIMES'
3692 013526 005203 INC R3
3693 013530 001404 BEQ 6$
3694 013532 000765 BR 4$
3695
3696 013534 005212 5$: INC (R2) ;INCRMNT '# OF TIMES'
3697 013536 005203 INC R3 ;ALL DONE?
3698 013540 001362 BNE 4$ ;IF NOT, GO BAK
3699
3700 013542 005205 6$: INC R5 ;SORTED BOTH FRWRD, REVRSE
3701 013544 001415 BEQ GOTYPE ;'SEEK TIMES'. IF YES GO TYPE
3702

```

G06

MD-11-DZRL-E, RK11-RK05 DYNAMIC TEST
DZRLKLE.P11 26-APR-77 12:27MACY11 30(1046) 14-JUL-77 08:03 PAGE 71
T11 SEEK FUNCTION TIMER

```

3703 013546 012737 027426 001162      MOV      #BUFRI1,$REG0      ;IF NOT, INITLZE PTR TO 'SEEK TIME'
3704 013554 012737 027430 001164      MOV      #BUFRI1+2,$REG1
3705 013562 012737 026666 001166      MOV      #BUFRI6,$REG2      ;SAVE 'SEEK TIME'
3706 013570 012737 026746 001170      MOV      #BUFRI7,$REG3      ;SAVE '# OF TIMES' HERE
3707
3708 013576 000677      BR          1$              ;GO BAK & DO SORTING FOR REVRSE SEEK TIMES
3709
3710                                ;TYPE OUT CYL #'S BETWEEN WHICH SEEK
3711                                ;WAS TIMED. 'OUTADR' TO 'INADR' 'INADR' TO 'OUTADR'
3712 013600 104401      COTYPE: TYPE
3713 013602 001213      SCRLF
3714 013604 104401 013612      TYPE      65$              ;;TYPE ASCIZ STRING
3715 013610 000403      BR          64$              ;;GET OVER THE ASCIZ
3716
3717 013620      ;;65$: .ASCIZ /CYLS:/
3718 64$:
3719 013620 017700 165436      MOV      @OUTADR,RO      ;GET OUTER CYL #
3720 013624 006200      ASR      RO
3721 013626 006200      ASR      RO
3722 013630 006200      ASR      RO
3723 013632 006200      ASR      RO
3724 013634 006200      ASR      RO
3725 013636 010046      MOV      RO,-(SP)
3726 013640 104424      TYPDSS      ;TYPE IT OUT IN DECIMAL
3727 013642 104401 013650      TYPE      67$              ;;TYPE ASCIZ STRING
3728 013646 000401      BR          66$              ;;GET OVER THE ASCIZ
3729
3730      ;;67$: .ASCIZ /-/-
3731 66$:
3732 013652 017701 165402      MOV      @INADR,R1      ;GET INNER CYL #
3733 013656 006201      ASR      R1
3734 013660 006201      ASR      R1
3735 013662 006201      ASR      R1
3736 013664 006201      ASR      R1
3737 013666 006201      ASR      R1
3738 013670 010146      MOV      R1,-(SP)
3739 013672 104424      TYPDSS      ;TYPE IT OUT IN DECIMAL
3740 013674 104401 013702      TYPE      69$              ;;TYPE ASCIZ STRING
3741 013700 000405      BR          68$              ;;GET OVER THE ASCIZ
3742
3743 013714      ;;69$: .ASCIZ <15><12>/ FRWRD/
3744 68$:
3745 013714 104401 002101      TYPE      ,BLNK$9
3746 013720 104401 013726      TYPE      71$              ;;TYPE ASCIZ STRING
3747 013724 000404      BR          70$              ;;GET OVER THE ASCIZ
3748
3749      ;;71$: .ASCIZ /REVRSE/
3750 70$:
3751                                ;TYPE OUT THE '# OF SEEKS' & 'SEEK
3752                                ;TIME' OBTAINED FOR EACH OF THOSE
3753                                ;SEEKS
3754 013736 005000      TYPTIM: CLR      RO
3755 013740 005005      CLR      R5
3756 013742 104401      1$: TYPE
3757 013744 001213      SCRLF
3758 013746 016046 026606      MOV      BUFR5(RO),-(SP) ;GET '# OF SEEKS' IF NONE (0)
3759 013752 001424      BEQ      3$              ;SKIP TYPING (FRWRD SEEK)
3760 013754 104405      TYPDS      ;GO TYPE OUT DECIMAL '# OF SEEKS'

```

```

3759 013756 104401 TYPE
3760 013760 002110 BLNKS2
3761 013762 016046 026526 MOV BUFR4(R0),-(SP) ;GET 'SEEK TIME' FOR EACH OF
3762 013766 104405 TYPDS ;OF THAT '# OF SEEKS'. 'GC
;TYPE OUT IN DECIMAL
3763
3764
3765 013770 016046 026746 2$: MOV BUFR7(R0),-(SP) ;GET '# OF SEEKS' IF NONE .C
3766 013774 001416 BEQ 4$ ;SKIP TYPING (REVERSE SEEK)
3767 013776 005705 TST R5
3768 014000 001402 BEQ 6$
3769 014002 104401 002075 TYPE .BLNK13
3770 014006 104405 5$: TYPDS ;TYPE OUT IN DECIMAL
3771 014010 104401 TYPE
3772 014012 002110 BLNKS2
3773 014014 016046 026666 MOV BUFR6(R0),-(SP) ;GET 'SEEK TIME' & TYPE IT
3774 014020 104405 TYPDS ;OUT IN DECIMAL
3775 014022 000406 BR 5$
3776
3777 014024 005726 3$: TST (SP)+ ;POP STACK
3778 014026 005205 INC R5
3779 014030 000757 BR 2$
3780
3781 014032 005726 4$: TST (SP)+ ;POP STACK
3782 014034 005705 TST R5
3783 014036 001004 BNE TIMDON
3784
3785 014040 005720 5$: TST (R0)+ ;INCREMENT PTR TO TABLES
3786 014042 020027 000012 CMP R0,#12 ;ALL DONE?
3787 014046 001335 BNE 1$ ;IF NOT GO BAK
3788
3789 014050 062737 000002 001260 TIMDON: ADD #2,INADR ;INCRMNT POINTER TO NEXT
3790 014056 062737 000002 001262 ADD #2,OUTADR ;INNER & OUTER ADRES
3791 014064 005237 001252 INC RETRY1 ;ALL DONE?
3792 014070 001406 BEQ PLTGRPH
3793 014072 032777 000100 165040 BIT #SW6,2SWR ;INHIBIT TIMER? FURTHER ?
3794 014100 001402 BEQ PLTGRPH ;YES, BRANCH
3795 014102 000137 013200 JMP REPTIM ;GO, BACK AND TIME REST
3796 ;OF SEEKS
3797
3798
3799
3800 ;PLOT GRAPH OF 'SEEK TIME' V/S 'CYLIDERS SEEKED'
3801
3802 ;PERFORM SEEK FROM CYLINDER 0 TO EVERY OTHER CYLINDER AND TIME IT.
3803 ;0 0.0 1.0----0 312. NOTE 'SECTOR COUNTER' IS USED AS A READ
3804 ;TIME CLOCK TO TIME THERE SEEKS. AFTER OBTAINING THE SEEK TIMES A
3805 ;GRAPH IS PLOTTED OF 'SEEK TIME' V/S 'CYLINDER'.
3806
3807 ;TIME THE SEEKS
3808 014106 032777 000040 165024 PLTGRPH: BIT #SW5,2SWR ;SKIP THE GRAPH?
3809 014114 001002 BNE 16 ;YES, BRANCH
3810 014116 000137 015224 JMP TST12
3811 014122 104415 CON.RESET
3812 014124 104416 DRV.RESET
3813 014126 012737 177465 001500 MOV #-312,INDX3 ;PERFORM 313 SEEKS 0-0.0-1.0-312
3814 014134 012704 027026 MOV #BUFR10,R4 ;STORE 'SEEK TIME' HERE

```

106

```

3815 014140 005037 001260          CLR      INADR          ;CLR CYL ADRES BITS
3816
3817 014144 013777 001260 165312 1$: MOV      INADR,ARKDA      ;ADRES THE RIGHT CYLINDER
3818 014152 053777 001230 165304     BIS      DRIVAD,ARKDA    ;ADRES THE RIGHT DRIVE
3819
3820 014160 004737 014776          JSR      PC,TIMSEK          ;GO TIME THE SEEK FROM CYL 0
3821                                ;TO THE ABOVE CYL. RETURN WITH
3822                                ;R3 CONTAINING 'SEEK TIME' IN MS
3823                                ;SCALE FACTOR OF 0.01
3824 014164 010324                                ;STORE 'SEEK TIME'
3825 014166 042777 017777 165270     BIC      #17777,ARKDA      ;SEEK BACK TO CYL 0 FOR
3826 014174 012777 000011 165254     MOV      #11,ARKCS          ;TIMING NXT CYL SEEK
3827 014202 104421                                ;WAIT FOR CNTRL RDY?
3828 014204 104422                                ;WAIT FOR R/W/S RDY
3829 014206 062737 000040 001260     ADD      #40,INADR        ;FORM NXT CYL ADRES
3830 014214 005237 001500          INC      INDX3
3831 014220 001351          BNE      1$
3832
3833                                ;PLOT A GRAPH USING 'SEEK TIMES' RECORDED BEFORE
3834
3835 014222                                PLOT:
3836 014222 104401 014230          TYPE      65$              ;;TYPE ASCIZ STRING
3837 014226 000422                                BR      64$              ;;GET OVER THE ASCIZ
3838                                ;;65$: .ASCIZ <15><12><12><12>/X AXIS - SEEK TIME - MILI SECS/
3839 014274                                64$:
3840 014274 104401 014302          TYPE      67$              ;;TYPE ASCIZ STRING
3841 014300 000423                                BR      66$              ;;GET OVER THE ASCIZ
3842                                ;;67$: .ASCIZ <15><12>/Y AXIS - CYLINDER SEEKED FROM 0/<15>'12'<12'
3843 014350                                66$:
3844
3845 014350 104401          TYPE      BLNKS7
3846 014352 002103          CLR      R0
3847 014354 005000          1$: MOV      R0,-(SP)          ;TYPE OUT THE TIME UNITS
3848 014356 010046                                ; (MILI SECS) FOR THE X-AXIS
3849 014360 104424          TYPDSS                                ;LIKE THIS:
3850 014362 005700          TST      R0                          ;0 20 30 40.....
3851 014364 001411          BEQ      2$
3852 014366 022700 000144          CMP      #144,R0
3853 014372 003010          BGT      4$
3854 014374 022700 000170          CMP      #170,R0
3855 014400 002412          BLT      5$
3856 014402 104401          TYPE      BLNKS2
3857 014404 002110          BR      3$
3858 014406 000404          2$: TYPE
3859 014410 104401          BLNKS1
3860 014412 002111          4$: TYPE
3861 014414 104401          BLNKS3
3862 014416 002107          3$: ADD      #12,R0
3863 014420 062700 000012          BR      1$
3864 014424 000754          5$: TYPE
3865                                SCRLF
3866 014426 104401          TYPE
3867 014430 001213          BLNKS7
3868 014432 104401
3869 014434 002103
3870

```

J06

3871 014436 012700 177763
3872 014442
3873 014442 104401 014450
3874 014446 000403
3875
3876 014456
3877 014456 005200
3878 014460 001370
3879
3880
3881
3882 014462 032777 000020 164450
3883 014470 001054
3884
3885
3886
3887
3888 014472 005000
3889 014474 032777 000040 164436
3890 014502 001445
3891 014504 104401
3892 014506 001213
3893
3894
3895
3896 014510 010046
3897 014512 104405
3898 014514 104401 014522
3899 014520 000401
3900
3901 014524
3902 014524 010001
3903 014526 006301
3904 014530 016103 027026
3905 014534 004737 014736
3906 014540 022700 000004
3907 014544 003402
3908 014546 005200
3909 014550 000751
3910 014552 022700 000024
3911 014556 003403
3912 014560 062700 000002
3913 014564 000743
3914 014566 022700 000310
3915 014572 003403
3916 014574 062700 000005
3917 014600 000735
3918 014602 022700 000312
3919 014606 001403
3920 014610 062700 000002
3921 014614 000727
3922 014616 000137 015224
3923
3924
3925
3926

PLT1: MOV #15,R0 ;TYPE OUT THE X-AXIS MARKERS
1\$:
TYPE 65\$;TYPE ASCIZ STRING
BR 64\$;GET OVER THE ASCIZ
65\$: .ASCIZ /I-----/
64\$:
INC R0 ;I-----I-----I-----
BNE 1\$
; IF SW 4 IS SET THEN TYPE THE COMPLETE GRAPH. IF NOT TYPE THE SMALL GRAPH.
BIT #SW4,2SWR ;TYPE COMPLETE GRAPH?
BNE CMPGRP ;YES BRANCH
;IF NOT, TYPE SMALL GRAPH
SMGRP: CLR R0
1\$: BIT #SW5,2SWR ;SKIP REST OF GRAPH?
BEQ 5\$;YES
TYPE ;IN THIS GRAPH SEEK TIMES ARE
\$CRLF ;PLOTTED ONLY FOR SELECTED
;CYLINDERS (NOT ALL) SHOWN BELOW:
;0,1,2,3,4, 6,8,10,12,14,16,18,20.
;25,30,35,....,190,195,200, 203
MOV R0,-(SP) ;TYPE THE MARKERS
TYPDS
TYPE 65\$;TYPE ASCIZ STRING
BR 64\$;GET OVER THE ASCIZ
65\$: .ASCIZ /-/
64\$:
MOV R0,R1 ;FORM THE ADRES OF 'SEEK TIME'
ASL R1
MOV BUFR10(R1),R3 ;GET THE SEEK TIME
JSR PC,PLTPT ;GO PLOT IT
CMP #4,R0 ;PLOTTED UPTO CYL 4?
BLE 2\$;YES
INC R0
BR 1\$
2\$: CMP #24,R3 ;PLOTTED UPTO CYL 20?
BLE 3\$
ADD #2,R0
BR 1\$
3\$: CMP #310,R0 ;PLOTTED UPTO CYL 200?
BLE 4\$
ADD #5,R0
BR 1\$
4\$: CMP #312,R0 ;PLOTTED ALL CYLS?
BEQ 5\$
ADD #2,R0
BR 1\$
5\$: JMP TST12

; IF SW 4 IS SET THE COMPLETE GRAPH IS PRINTED OUT. IT GIVES TIMES FOR
;SEEKS FROM CYLINDER 0 TO ALL OTHER CYLINDERS (0,1,2,3...202).

K06

```

3927
3928 014622 005000
3929 014624 012701 177773
3930 014630 012702 027026
3931 014634 104401
3932 014636 001213
3933 014640 000412
3934
3935 014642 032777 000040 164270 2$: BIT #SW5,JSWR ;SKIP REST OF GRAPH?
3936 014650 001002 BNE +6
3937 014652 000137 015224 JMP TST12
3938 014656 005201 INC R1 ;TYPE OUT Y-AXIS MARKER 'CYL #'
3939 014660 001005 BNE 4$ ;IF REQUIRED
3940 014662 012701 177773 MOV #-5,R1
3941 014666 010046 3$: MOV RO,-(SP) ;TYPE 'CYL #' ON Y-AXIS
3942 014670 104405 TYPDS ;(IN DECIMAL)
3943 014672 000402 BR 5$
3944 014674 104401 4$: TYPE
3945 014676 002104 BLNKS6
3946 014700 5$:
3947 014700 104401 014706 TYPE 65$ ;:TYPE ASCII STRING
3948 014704 000401 BR 64$ ;:GET OVER THE ASCII
3949
3950 014710 64$: .ASCIIZ /-/
3951
3952 014710 012203 MOV (R2)+,R3 ;GET SEEK TIME
3953 014712 004737 014736 JSR PC,PLTPT ;GO PLOT THE POINT
3954
3955
3956 014716 104401 TYPE
3957 014720 001213 SCRLF
3958 014722 005200 INC RO ;ALL DONE?
3959 014724 022700 000312 CMP #312,RO
3960 014730 001344 BNE 2$ ;IF NOT, GO BAK
3961 014732 000137 015224 6$: JMP TST12
3962
3963 ;PLTPT
3964 ;THE ROUTINE IS ENTERED WITH R3 CONTAINING HORIZONTAL AXIS
3965 ;COORDINATE- SEEK TIME
3966 ;PLOT THE ACTUAL TIME ON THE GRAPH. IN KEEPING WITH NORMAL
3967 ;CONVENTION A NUMBER LESS THAN HALF THE CELL WIDTH IS
3968 ;CONSIDERED AS FALLING UNDER THE PREVIOUS CELL, A NUMBER
3969 ;GREATER THAN OR EQUAL TO HALF THE CELL WIDTH FALLS UNDER THE NEXT CELL
3970 ;EX: IF SEEK TIME IS 11.5 MS, IT'S BETW'N 10 & 12, BUT > 11
3971 ;HENCE IT WILL BE PLOTTED AS 12 IF SEEK TIME IS 10.8 MS,
3972 ;IT'S BETW'N 10 & 12, BUT < 11 HENCE IT WILL BE PLOTTED
3973 ;AS 10.0 MS
3974
3975 014736 162703 000310 PLTPT: SUB #310,R3 ;FIND OUT HOW MANY BLANKS TO
3976 014742 002403 BLT 7$ ;INSERT TO PLOT THE POINT
3977 ;NOTE THE FIRST CELL = 0 MS
3978 014744 104401 TYPE
3979 014746 002111 BLNKS1
3980 014750 000772 BR PLTPT
3981 014752 062703 000144 7$: ADD #144,R3
3982 014756 002402 BLT 8$

```

```

3983 014760 104401
3984 014762 002111
3985
3986 014764
3987 014764 104401 014772
3988 014770 104401
3989
3990 014774
3991 014774 000207
3992
3993
3994
3995
3996
3997
3998
3999
4000
4001
4002
4003 014776 010246
4004 015000 005003
4005 015002 013701 001452
4006 015006 011102
4007 015010 032702 000400
4008 015014 001774
4009
4010 015016 032702 000010
4011 015022 001771
4012
4013
4014 015024 011102
4015 015026 032702 000400
4016 015032 001774
4017 015034 021102
4018 015036 001372
4019 015040 032702 000017
4020 015044 001367
4021
4022
4023 015046 012777 000011 164402
4024
4025 015054 104421
4026
4027 015056 011102
4028 015060 032702 000400
4029 015064 001774
4030 015066 020211
4031 015070 001372
4032 015072 032702 000100
4033 015076 001025
4034 015100 032702 000017
4035 015104 001764
4036
4037 015106 011102
4038 015110 032702 000400

```

```

TYPE
BLNKS1

8$:
TYPE 655 ;:TYPE ASCII STRING
BR 645 ;:GET OVER THE ASCII
655: .ASCIIZ /X/
645:
RTS PC

;TIMSEK
;THIS ROUTINE FINDS OUT THE TIME REQUIRED TO SEEK TO THE CYLINDER
;INDICATED IN RKDA.
;***CAUTION*** SECTOR COUNTER IS USED AS A REAL TIME CLOCK TO KEEP TIME.
;ENTRY: JSR PC,TIMSEK
;RKDA CONTAINS THE CYLINDER TO BE SEEKED FROM THE PRESENT POSITION.
;RETURN: R3 CONTAINS THE SEEK TIME IN MILI SECS. SCALE FACTOR = 0.01

TIMSEK: MOV R2, -(SP) ;R3 WILL COUNT REVOLUTIONS OF
CLR R3 ;DISK (FROM INDEX MARK TO INDEX MARK.
;40 MILI SECS FOR EACH REV
1$: MOV RKDS, R1
MOV R1, R2
BIT #400, R2 ;WAIT FOR SOK
BEQ 1$

BIT #BIT3, R2 ;WAIT FOR SECTOR 10 SO THAT
BEQ 1$ ;U CAN START WAITING FOR
;INDEX, SEC 0

2$: MOV R1, R2
BIT #400, R2 ;WAIT FOR SEC OK
BEQ 2$
CMP R1, R2
BNE 2$
BIT #17, R2 ;WAIT FOR SEC 0, INDEX MARK
BNE 2$ ;AS SOON AS IT IS SEC 0, ISSUE
;A SEEK & START TIMING

MOV #11, R1 ;ISSUE A SEEK, START TIMING
CON.RDY ;THE SEC COUNTER
;WAIT FOR CNTRL RDY

3$: MOV R1, R2 ;GET RKDS
BIT #400, R2 ;WAIT FOR SOK
BEQ 3$
CMP R2, R1 ;INFO CORRECT?
BNE 3$ ;NO
BIT #100, R2 ;R/W/S RDY SET?
BNE SKDON ;IF YES, BRANCH
BIT #17, R2 ;WAIT FOR SEC CNTR TO MOVE
BEQ 3$ ;FROM 0 TO 1

4$: MOV R1, R2
BIT #400, R2 ;WAIT FOR SOK

```

MO6

4039	015114	001774		BEG	45	
4040	015116	020211		CMP	R2, 2P1	
4041	015120	001372		BNE	45	
4042	015122	032702	000100	BIT	#100, R2	; R/W/S ROY SET, SEEK DONE?
4043	015126	001005		BNE	55	; YES, BRANCH
4044	015130	032702	000017	BIT	#17, R2	; IF NOT KEEP TRACK OF SEC
4045	015134	001364		BNE	45	; COUNTER. INCREMENT R3 AT
4046						; EVERY INDEX MARK, EVERY
4047	015136	005203		INC	R3	; 40 MILI SECS
4048	015140	000746		BR	35	; GO BAK, KEEP TIME
4049						
4050	015142	032702	000017	55:	BIT	#17, R2
4051	015146	001001		BNE	SKDON	; CHECK, IS IT INDEX MARK -SEC 0
4052	015150	005203		INC	R3	; IF NOT, SKIP
4053						; IF YES, INCREMENT COUNT
4054						; SEEK DONE, SAVE RKDS-SEC COUNTER.
4055	015152			SKDON:		
4056	015152	012746	000014	MOV	#14, -(SP)	; PUT THE MULTIPLIER ON THE STACK
4057	015156	010346		MOV	R3, -(SP)	; PUT THE MULTIPLICAND ON THE STACK
4058	015160	004737	020500	JSR	PC, 2#MULT	; CALL THE MULTIPLY ROUTINE
4059	015164	012616		MOV	(SP)+, (SP)	; DISREGARD THE MSB'S
4060	015166	012603		MOV	(SP)+, R3	; GET THE LSB'S OF THE PRODUCT
4061	015170	042702	177760	BIC	#177760, R2	; SEEK. TOTAL TIME=(IN DECIMAL)
4062	015174	060203		ADD	R2, R3	; [(R3)X12+SEC COUNTER]X330X0.01
4063						; NOTE THERE IS A SCALE FACTOR
4064	015176	012746	000512	MOV	#512, -(SP)	; PUT THE MULTIPLIER ON THE STACK
4065	015202	010346		MOV	R3, -(SP)	; PUT THE MULTIPLICAND ON THE STACK
4066	015204	004737	020500	JSR	PC, 2#MULT	; CALL THE MULTIPLY ROUTINE
4067	015210	012616		MOV	(SP)+, (SP)	; DISREGARD THE MSB'S
4068	015212	012603		MOV	(SP)+, R3	; GET THE LSB'S OF THE PRODUCT
4069	015214	062703	000245	ADD	#245, R3	; ASSUMPTION THAT EACH SECTOR
4070						; TAKES 3.3 MILI SECS. IF THE
4071						; DISK SPEED IS VERY MUCH DIFRNT
4072						; FROM THE SPEC SPEED OF
4073						; 1500 RPM (40 MS/REV), THEN
4074						; SEC COUNTER WOULD NOT BE AN
4075						; ACCURATE TIME CLOCK.
4076	015220	012602		MOV	(SP)+, R2	; POP R2 BAK
4077	015222	000207		RTS	PC	; RETURN
4078						
4079						
4080						
4081						
4082						
4083						
4084						
4085	015224	000004		†ST12:	SCOPE	
4086	015226	105237	001223	INCB	DRVDON	
4087	015232	123737	001223 001224	BTEOP:	CMPB	DRVDON, DRVS
4088	015240	001402		BEG	.+6	
4089	015242	000137	003760	JMP	NXTDRV	
4090						
4091						
4092						
4093						
4094						

.SBTTL END OF PASS ROUTINE

;*****
; *INCREMENT THE PASS NUMBER (\$PASS)
;*****

N06

4095
4096
4097
4098
4099
4100 015246
4101 015246 000004
4102 015250 005037 001102
4103 015254 005237 001100
4104 015260 042737 100000 001100
4105 015266 005327
4106 015270 000001
4107 015272 003022
4108 015274 012737
4109 015276 000001
4110 015300 015270
4111 015302 104401 015347
4112 015306 013746 001100
4113 015312 104405
4114 015314 104401 015344
4115 015320 013700 000042
4116 015324 001405
4117 015326 000005
4118 015330 004710
4119 015332 000240
4120 015334 000240
4121 015336 000240
4122 015340
4123 015340 000137
4124 015342 003742
4125 015344 377 377 000
4126 015347 015 042412 042116
4127 015354 050040 051501 020123
4128 015362 000043
4129

```

; *INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
; *TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
; *IF THERES A MONITOR GO TO IT
; *IF THERE ISN'T JUMP TO ST3

SEOP:
    SCOPE
    CLR STSTNM
    INC SPASS
    BIC #100000, SPASS
    DEC (PC)+
    SEOPCT: .WORD 1
    BGT SDOAGN
    MOV (PC)+, 2(PC)+
    SENDCT: .WORD 1
    SEOPCT
    TYPE SENDMG
    MOV SPASS, -(SP)
    TYPDS
    TYPE SENULL
    SGET42: MOV 2#42, R0
    BEQ SDOAGN
    RESET
    SENDAD: JSR PC, (R0)
    NOP
    NOP
    NOP
    SDOAGN:
    JMP 2(PC)+
    SRTNAD: .WORD ST3
    SENULL: .BYTE -1, -1, 0
    SENDMG: .ASCIZ <15><12>/END PASS #/

;; ZERO THE TEST NUMBER
;; INCREMENT THE PASS NUMBER
;; DON'T ALLOW A NEG. NUMBER
;; LOOP?
;; YES
;; RESTORE COUNTER
;; TYPE "END PASS #"
;; SAVE SPASS FOR TYPEOUT
;; GO TYPE--DECIMAL ASCII WITH SIGN
;; TYPE A NULL CHARACTER
;; GET MONITOR ADDRESS
;; BRANCH IF NO MONITOR
;; CLEAR THE WORLD
;; GO TO MONITOR
;; SAVE ROOM
;; FOR
;; ACT11
;; RETURN
;; NULL CHARACTER STRING

```

B07

;COMMON SUBROUTINES AND HANDLERS

.SBTTL ESR15

;ESR15
;THIS ROUTINE IS USED TO TYPE OUT ERROR DATA FOR ITEM 15
;OF THE ERROR TABLE. AT THE TIME OF ENTRY INTO THIS
;ROUTINE RS CONTAINS THE DISK ADDRESS FROM WHICH THE 12
;HEADERS WERE READ. THE SECTOR #'S WHICH GAVE BAD HEADERS HAVE
;BEEN STORED STARTING AT 'BUFR'. THE CORRESPONDING BAD HEADERS
;HAVE BEEN STORED STARTING AT 'BUFRI'.

;THE PRINTOUT LOOKS LIKE:

;SEC# HDR RECVD
;AA BBBBBB AA=BAD SEC # BBBBBB=BAD HEADER
;EXPCD HDR=XXXXXX TRY# Y

ESR15:

MOV R1,-(SP) ;PUSH R1 ON STACK
MOV R2,-(SP) ;PUSH R2 ON STACK
MOV #BUFR,R1 ;SEC #'S STORED HERE PREVIOUSLY
MOV #BUFRI,R2 ;BAD HDRS STORED HERE PRVSLY

IS:

MOV (R1)+,-(SP)
TYP0S ;GO TYPE OUT BAD SEC # (OCTAL)
.BYTE 2 ;ONLY 2 DIGITS
.BYTE 0 ;SUPRES LDG 0'S
TYPE ;TYPE 3 BLNKS

BLNKS3
MOV (R2)+,-(SP) ;GO TYPE OUT BAD HEADER

TYP0C
TYPE
BLNKS4

TYPE
\$CRLF
CMP #177777,(R1) ;ALL BAD SEC #'S TYPD OUT?
BNE IS ;IF NOT GO BAK

TYPE
MSG6
MOV RS,-(SP) ;TYPE OUT EXPCD HEADER FOR
BIC #160037,(SP)
TYP0C ;THAT CYLINDER

MOV (SP)+,R2 ;POP STACK INTO R2
MOV (SP)+,R1 ;POP STACK INTO R1
RTS PC

.SBTTL ESR13

;ESR13
;THIS ROUTINE IS USED WITH 'ERROR 13' TO TYPEOUT OUT ERROR
;DATA. THE SECTOR #'S WHICH GAVE BAD HEADERS HAVE BEEN STORED
;STARTING AT 'BUFR'. THE CORRESPONDING BAD HEADERS HAVE
;BEEN STORED STARTING AT 'BUFRI'. RS CONTAINS THE EXPECTED

4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149 015364
4150 015364 010146
4151 015366 010246
4152 015370 012701 001266
4153 015374 012702 001320
4154 015400 012146
4155 015402 104403
4156 015404 002
4157 015405 000
4158 015406 104401
4159 015410 002107
4160 015412 012246
4161 015414 104402
4162 015416 104401
4163 015420 002106
4164 015422 104401
4165 015424 001213
4166 015426 022711 177777
4167 015432 001362
4168
4169 015434 104401
4170 015436 001716
4171 015440 010546
4172 015442 042716 160037
4173 015446 104402
4174
4175 015450 012602
4176 015452 012601
4177 015454 000207
4178
4179
4180
4181
4182
4183
4184
4185

007

4186
4187
4188
4189
4190
4191
4192
4193 015456 004737 015364
4194 015462 104401 015470
4195 015466 000404
4196
4197 015500
4198 015500 005046
4199 015502 032705 000020
4200 015506 001401
4201 015510 005216
4202 015512 104402
4203
4204 015514 104401 002053
4205 015520 013746 001254
4206 015524 005216
4207 015526 104402
4208 015530 000207
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219 015532 004737 015456
4220 015536 004737 016320
4221
4222 015542 104401 015550
4223 015546 000404
4224
4225 015560
4226 015560 013746 001162
4227 015564 104403
4228 015566 003
4229 015567 000
4230 015570 104401 015576
4231 015574 000404
4232
4233 015606
4234 015606 013746 001164
4235 015612 104403
4236 015614 003
4237 015615 000
4238 015616 000207
4239
4240
4241

;HEADER FOR THAT CYLINDER. THE TYPEOUT LOOKS LIKE

;SEC# HDR RCVD
;AA BBBBBB AA=BAD SEC #
;EXPTD HDR=XXXXXX BBBBBB=BAD HEADER
TRY# Y SUR=Z

ESR13: JSR PC,ESR15
TYPE 65\$;:TYPE ASCIZ STRING
BR 64\$;:GET OVER THE ASCIZ

;65\$: .ASCIZ / SUR=/
64\$: CLR -(SP)
BIT #20,RS ;SUR 0 OR 11'
BEQ 1\$
INC (SP)

1\$: TPOC
TYPE MSG13
MOV RETRY2,-(SP)
INC (SP)
TPOC
RTS PC

.SBTTL ESR20

;ESR20
;SUBROUTINE TO TYPE OUT ERROR DATA FOR 'ERROR 20'. AT THE TIME
;OF ENTRY, TABLE STARTING AT 'BUFR' CONTAINS SECTOR #'S THAT GAVE BAD
;HEADERS. TABLE AT 'BUFR1' CONTAINS BAD HEADERS, RS CONTAINS EXPECTED
;HEADER FOR THE CYLINDER. 'INADR' AND 'OUTADR' CONTAIN THE CYLINDER
;ADDRESSES BETWEEN WHICH THE IMPLIED SEEK WAS TRIED.

ESR20: JSR PC,ESR13 ;GO TYPE OUT SEC #'S, BAD HORS
JSR PC,ERR2 ;GET CYL #'S BETWEEN WHICH SEEK
;WAS TRIED

TYPE 65\$;:TYPE ASCIZ STRING
BR 64\$;:GET OVER THE ASCIZ

;65\$: .ASCIZ / CYLA=/
64\$: MOV \$REG0,-(SP) ;GO TYPE CYL # FROM WHERE
TPOC ;SEEK BEGAN

.BYTE 3 ;TYPE 3 DIGITS
.BYTE 0 ;SUPRES LOG 0'S
TYPE 67\$;:TYPE ASCIZ STRING
BR 66\$;:GET OVER THE ASCIZ

;67\$: .ASCIZ / CYLB=/
66\$: MOV \$REG1,-(SP) ;TYPE CYL # TO WHICH SEEK
TPOC ;WAS DONE
.BYTE 3 ;TYPE 3 DIGITS
.BYTE 0 ;SUPRES LOG 0'S
RTS PC ;RETURN

.SBTTL ESR25

4242	015620	010205		ESR25:	MOV	R2,R5	:SAVE ADRES OF TERMINATOR
4243							
4244	015622	012702	001266		MOV	#BUFR,R2	:INITLZE PTR TO TABLE STORING
4245							:ADRES OF BAD DATA
4246	015626	012703	001320		MOV	#BUFR1,R3	:INITLZE PTR TO 'EXPCTD' DATA
4247	015632	012704	001352		MOV	#BUFR2,R4	:INITLZE PTR TO 'RECVD' DATA
4248							
4249	015636	032777	020000 163274 1\$:		BIT	#SW13,2SWR	:INHIBIT TYPE OUT?
4250	015644	001076			BNE	4\$	:YES, EXIT
4251	015646	104401			TYPE		:TYPE CR,LF
4252	015650	001213			\$CRIF		
4253							
4254	015652	163712	001404		SUB	PBUFO,(R2)	:GET WORD # IN BUFR (0,1,2...
4255	015656	006212			ASR	(R2)	
4256	015660	011246			MOV	(R2),-(SP)	:WHICH WAS BAD. NOTE YOU
4257							:CAN HAVE THE ACTUAL MEMORY
4258							:ADRES BY ADDING 'IOBUFO'
4259							:TO THIS
4260	015662	104403			TYPOS		:GO TYPE WORD # THAT WAS BAD
4261	015664	004			.BYTE	4	
4262	015665	000			.BYTE	0	
4263	015666	104401			TYPE		
4264	015670	002107			BLNKS3		:2 BLANKS
4265							
4266	015672	012346			MOV	(R3)+,-(SP)	:GET EXPCTD DATA
4267	015674	104402			TYPOC		:GO TYPE IT
4268	015676	104401			TYPE		
4269	015700	002110			BLNKS2		
4270	015702	012446			MOV	(R4)+,-(SP)	:GET RECVD DATA (BAD)
4271	015704	104402			TYPOC		:GO TYPE IT
4272	015706	104401			TYPE		
4273	015710	002110			BLNKS2		
4274							
4275	015712	012700	000400		MOV	#400,R0	:GET THE DISK ADRES FROM
4276	015716	021200		2\$:	CMP	(R2),R0	:WHICH THIS (BAD) DATA WAS
4277	015720	002405			BLT	3\$	:READ
4278	015722	062700	000400		ADD	#400,R0	
4279	015726	022700	002400		CMP	#2400,R0	
4280	015732	001371			BNE	2\$	
4281							
4282	015734	000300		3\$:	SWAB	R0	
4283	015736	005300			DEC	R0	
4284	015740	063700	001450		ADD	ADRES,R0	:R0 CONTAINS THE DISK
4285							:ADRES FROM WHICH THE (BAD,
4286	015744	010037	001170		MOV	R0,\$REG3	:DATA WAS READ
4287							
4288	015750	004737	016220		JSR	PC,BRKDA	:GO BREAK ABOVE DISK ADRES
4289							:INTO CYL#, SUR#, SEC#
4290							
4291	015754	013746	001174		MOV	\$REG5,-(SP)	:GET THE CYL#
4292	015760	104403			TYPOS		:TYPE IT
4293	015762	003			.BYTE	3	:ONLY 3 DIGITS
4294	015763	000			.BYTE	0	:NO LEADING 0'S
4295	015764	104401			TYPE		
4296	015766	002107			BLNKS3		
4297							

```

4298 015770 013746 001176      MOV      $REG6,-(SP)      ;GET SUR #
4299 015774 104403              TYPOS              ;TYPE
4300 015776      001              .BYTE      1          ;1 DIGIT ONLY
4301 015777      000              .BYTE      0
4302
4303 016000 104401              TYPE
4304 016002 002106              BLNKS4
4305
4306 016004 013746 001200      MOV      $REG7,-(SP)      ;GET SEC#
4307 016010 104403              TYPOS              ;TYPE
4308 016012      002              .BYTE      2          ;2 DIGITS
4309 016013      000              .BYTE      0
4310
4311 016014 005722              TST      (R2)+          ;INCREMNT PTR
4312 016016 020205              CMP      R2,R5          ;TYPED OUT ALL BAD DATA
4313
4314 016020 001306              BNE      1$              ;INFO?
4315 016022 104401              TYPE              ;IF NOT LUP BAK
4316 016024 002053              MSG13
4317 016026 013746 001254      MOV      RETRY2,-(SP)      ;' TRY #:'
4318 016032 062716 000003      ADD      #3,(SP)          ;GET RETRY COUNT
4319 016036 104403              TYPOS              ;FORM THE RETRY NO.
4320 016040      001              .BYTE      1          ;TYPE IT OUT
4321 016041      000              .BYTE      0
4322
4323 016042 000207      4$:      RTS      PC              ;IF YES, RETURN
4324
4325      ;MESSAGE HANDLER
4326      ;THE MESSAGE HANDLER IS USED FOR TYPING OUT MESSAGES & DATA
4327      ;RELATED TO THE MESSAGE. IF SW13 IS SET, THE TYPEOUT IS
4328      ;INHIBITED. THE CALL IS:
4329      ;      MESSAGE,XX
4330      ;XXX IS THE MESSAGE NUMBER & PROVIDES AN INDEX TO THE
4331      ;'ERROR ITEMS TABLE' WHERE THAT MESSAGE ITEM
4332      ;IS LOCATED.
4333      ;THE MESSAGE ITEM CONTAINS:
4334      ;      MS:      POINTER TO THE ASCII MESSAGE
4335      ;      DH:      POINTER TO THE DATA HEADER
4336      ;      DT:      POINTER TO THE DATA
4337      ;      0        TERMINATOR
4338      ;IF 'DT' IS 0 THE DATA IS TO BE PRINTED USING THE SUBROUTINE
4339      ;INDICATED IN PLACE OF THE TERMINATOR
4340 016044 032777 020000 163066 MSGE:  BIT      #SW13,$SWR      ;INHIBIT TYPEOUT?
4341 016052 001012              BNE      1$              ;IF YES, EXIT
4342 016054 011637 001116      MOV      (SP),SERRPC      ;GET ADRES OF 'MESSAGE' CALL
4343 016060 162737 000002 001116      SUB      #2,SERRPC      ;STORE IT
4344 016066 117637 000000 001114      MOVB     2(SP),SITEMB      ;GET MESSAGE # (INDEX TO ITEM TABLE)
4345 016074 004737 017446      JSR      PC,2$ERRTYP      ;GO TO 'ERRTYP' & TYPE OUT
4346
4347 016100 062716 000002      1$:      ADD      #2,(SP)          ;INFO
4348 016104 000002              RTI              ;ADJUST RETURN ADDRES
4349
4350      ;THIS ROUTINE IS USED FOR TYPING OUT ASCII MESSAGES. BEFORE
4351      ;THE MESSAGE IS TYPED SW13 IS CHECKED & IF SET THE
4352      ;TYPEOUT IS INHIBITED & AN EXIT IS MADE.
4353      ;THE CALL FOR THIS ROUTINE IS "TYPMSG", AN ENCODED

```

F07

4354
4355
4356
4357
4358 016106 032777 020000 163024
4359 016114 001005
4360 016116 017537 000000 016126
4361 016124 104401
4362 016126 000000
4363 016130 062716 000002
4364
4365 016134 000002
4366
4367
4368
4369
4370
4371
4372
4373 016136 017746 163322
4374 016142 042716 160037
4375 016146 006316
4376 016150 006316
4377 016152 006316
4378 016154 000316
4379 016156 112637 001172
4380
4381
4382
4383
4384
4385
4386
4387
4388 016162 017737 163270 001162
4389 016170 017737 163260 001164
4390 016176 017737 163250 001166
4391 016204 017737 163254 001170
4392 016212 000207
4393
4394
4395
4396
4397
4398
4399
4400 016214 004737 016162
4401 016220 010046
4402 016222 010146
4403 016224 010246
4404 016226 012700 001202
4405 016232 013701 001170
4406 016236 010102
4407 016240 042702 177760
4408 016244 010240
4409 016246 006201

;TRAP INSTRUCTION.
;THE POINTER TO THE ASCII MESSAGE TO BE TYPED IS LOCATED IN THE
;WORD FOLLOWING THE "TYPMSG" CALL.

TY.MSG: BIT #SW13,2SWR ;INHIBIT TYPEOUT?
BNE 25 ;YES, EXIT
MOV 2(SP),15 ;GET POINTER TO ASCII MESSAGE
TYPE ;GO TYPE ASCII STRING
15: 0
25: ADD #2,(SP) ;ADJUST RETURN ADRES. SKIP OVER
;POINTER ON RETURN
RTI ;EXIT

;GT5RG
;THIS ROUTINE EXTRACTS THE CYLINDER # FROM RKDA AND STORES IT
;IN \$REG4. THEN TRANSFERS RKCS, ER, DS, DA TO \$REG0, \$REG1, \$REG2, \$REG3
GT5RG: MOV 2RKDA,-(SP) ;PUSH RKDA ONTO STACK
BIC #160037,(SP) ;MASK OUT NON-CYLINDER BITS
ASL (SP) ;SHIFT 8 BITS OF CYL ADRES TO LO BYTE
ASL (SP)
ASL (SP)
SWAB (SP)
MOVB (SP)+,\$REG4 ;UP STACK

;GT4RG
;THIS ROUTINE TRANSFERS THE CONTENTS OF RKCS, RKER, RKDS
;RKDA TO \$REG0, \$REG1, \$REG2, \$REG3 RESPECTIVELY. \$REG'S
;ARE USED FOR TYPING OUT THERE CONTENTS AT THE TIME OF ERROR

GT4RG: MOV 2RKCS,\$REG0 ;GET RKCS
MOV 2RKER,\$REG1 ;RKER
MOV 2RKDS,\$REG2 ;RKDS
MOV 2RKDA,\$REG3 ;RKDA
RTS PC ;EXIT FROM THIS ROUTINE

;GETINF
;THIS ROUTINE SAVES THE CONTENTS OF RKCS IN \$REG0
;RKER IN \$REG1, RKDS IN \$REG2. THEN IT BREAKS RKDA
;INTO DRIVE NO, CYLINDER #, SURFACE AND SECTOR #.
;AND SAVES THEM IN \$REG4, \$REG5, \$REG6, \$REG7.

GETINF: JSR PC,GT4RG
BRKDA: MOV R0,-(SP)
MOV R1,-(SP)
MOV R2,-(SP)
MOV #2,\$REG7+2,R0
MOV \$REG3,R1
MOV R1,R2
BIC #177760,R2
MOV R2,-(R0)
ASR R1

```

4410 016250 006201
4411 015252 006201
4412 016254 006201
4413 016256 010102
4414 016260 042702 177776
4415 016264 010240
4416 016266 006201
4417 016270 010102
4418 016272 042702 177400
4419 016276 010240
4420 016300 000301
4421 016302 042701 177770
4422 016306 010140
4423 016310 012602
4424 016312 012601
4425 016314 012600
4426 016316 000207

```

```

ASR R1
ASR R1
ASR R1
MOV R1,R2
BIC #177776,R2
MOV R2,-(R0)
ASR R1
MOV R1,R2
BIC #177400,R2
MOV R2,-(R0)
SWAB R1
BIC #177770,R1
MOV R1,-(R0)
MOV (SP)+,R2
MOV (SP)+,R1
MOV (SP)+,R0
RTS PC

```

```

4427
4428
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439 016320 013737 001260 001162
4440 016326 004737 016434
4441 016332 013737 001162 001164
4442 016340 013737 001262 001162
4443 016346 004737 016434
4444 016352 005704
4445 016354 001407
4446 016356 013746 001162
4447 016362 013737 001164 001162
4448 016370 012637 001164
4449 016374 000207
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4460 016376 010537 001162
4461 016402 004737 016434
4462 016406 005704
4463 016410 001006
4464 016412 013737 001162 001164
4465 016420 005037 001162

```

.SBTTL ERR2

```

;ERR2
;THIS ROUTINE GETS THE CYLINDER NUMBERS BETWEEN WHICH (IMPLIED) SEEK
;WAS DONE. (R4)=0 INDICATES SEEK FROM 'OUTADR' TO 'INADR'
;(R4)=1 INDICATES SEEK TO 'OUTADR'. ON EXIT $REG0 CONTAINS CYL #
;FROM WHICH SEEK WAS INITIATED, $REG1 CONTAINS CYL # TO WHICH SEEK WAS DONE
ERR2: MOV INADR,$REG0 ;GET CYL ADRES
JSR PC,GCYL ;GO GET CYL# FROM IT
MOV $REG0,$REG1 ;SAVE
MOV OUTADR,$REG0 ;GET CYL ADRES
JSR PC,GCYL ;GO GET CYL # FROM IT
TST R4 ;GOING WHICH WAY?
BEQ 1$ ;'OUTADR' TO 'INADR'. BRANCH
MOV $REG0,-(SP) ;EXCHANG CYL# TO GET
MOV $REG1,$REG0 ;CORRECT 'TO' & 'FROM' CYLS
MOV (SP)+,$REG1
1$: RTS PC ;RETURN

```

.SBTTL ERR1

```

;ERR1
;THIS SUBROUTINE FINDS OUT THE CYLINDER NOS. BETWEEN WHICH THE SEEK
;IS DONE. THE CYLINDER # WHERE THE HEADS WERE PRIOR TO MOVING, IS
;DEPOSITED IN $REG0. THE CYLINDER # WHERE THE HEADS SHOULD BE AFTER
;MOVEMENT, IS DEPOSITED IN $REG1. R4 INDICATES WHICH DIRECTION THE
;HEADS WERE MOVING, IN OR OUT. R5 CONTAINS THE
;DISK ADDRESS (IN OR OUT AS THE CASE MAY BE).

```

```

ERR1: MOV R5,$REG0
JSR PC,GCYL ;GO GET CYL #
TST R4 ;WAS GOING IN OR OUT?
BNE 1$ ;OUT
MOV $REG0,$REG1
CLR $REG0

```

4466	016424	000207			RTS	PC	
4467	016426	005037	001164		1\$: CLR	\$REG1	
4468	016432	000207			RTS	PC	
4469							
4470							
4471							
4472							
4473							
4474							
4475	016434	010046					
4476	016436	013700	001162				
4477	016442	042700	160037				
4478							
4479	016446	006200					
4480	016450	006200					
4481	016452	006200					
4482	016454	006200					
4483	016456	006200					
4484	016460	010037	001162				
4485	016464	012600					
4486	016466	000207					
4487							
4488							
4489							
4490							
4491							
4492							
4493							
4494							
4495							
4496							
4497							
4498	016470	005037	001174				
4499	016474	013777	001230	162762			
4500	016502	012777	000015	162746			
4501	016510	104421					
4502	016512	000402					
4503	016514	005037	001174				
4504	016520	032777	000100	162724			
4505	016526	001024					
4506	016530	012746	177770				
4507	016534	005216					
4508	016536	001376					
4509	016540	005726					
4510	016542	005237	001174				
4511	016546	001364					
4512	016550	032777	020000	162362			
4513	016556	001010					
4514	016560	104420					
4515	016562	002027					
4516	016564	104420	001733				
4517	016570	011646					
4518	016572	162716	000002				
4519	016576	104402					
4520	016600	000002					
4521							


```

      .SBTTL GCYL
:GCYL
:THIS ROUTINE EXTRACTS THE CYLINDER NO. FROM THE DISK ADDRESS
:CONTAINED IN '$REG0' AND THEN STORES IT BACK IN '$REG0'
GCYL:  MOV    RO, -(SP)      ;PUSH RO ONTO STACK
      MOV    $REG0, RO
      BIC    #160037, RO    ;MASK OUT DRV # BITS &
                              ;SUR, SEC BITS IF PRESENT
      ASR    RO
      ASR    RO              ;SHIFT CYL BITS RIGHT
      ASR    RO              ;BY 5
      ASR    RO
      MOV    RO, $REG0      ;STORE CYL # IN $REG0
      MOV    (SP)+, RO      ;POP RO FROM STACK
      RTS    PC             ;EXIT

      .SBTTL DRV.RESET - DRIVE RESET ROUTINE
      .SBTTL RESDON - WAIT FOR DRIVE RESET TO BE DONE
:DR.RST
:THIS ROUTINE DOES A DRIVE RESET ON THE DRIVE WHOOSE ADDRESS IS IN
:RKDA. MULTIPLE RETURN ADDRESSES FOR THIS ROUTINE ARE PROVIDED.
:IF THERE IS NO ERROR (R/W/S RDY SETS WITHIN CERTAIN TIME) THEN
:A NORMAL EXIT IS MADE. IF R/W/S RDY DOES NOT SET ERROR IS REPORTED.

DR.RST: CLR    $REG5        ;INITIALIZE THE COUNT
      MOV    DRIVAD, DRKDA
      MOV    #15, DRKCS    ;DRIVE RESET, GO
      CON.RDY
      BR     RES.D0+4
RES.D0: CLR    $REG5
      1$: BIT    #100, DRKDS    ;DID R/W/S RDY SET?
      BNE    2$
      MOV    #-10, -(SP)      ;PUSH COUNT ON SP
      INC    (SP)            ;COUNT IT DOWN
      BNE    -2
      TST    (SP)+          ;POP UP SP
      INC    $REG5          ;IF NOT WAIT
      BNE    1$             ;WAITED LONG?
      BIT    #SW13, ASWR
      BNE    2$
      TYPMSG MSG12
      TYPMSG MSG7
      MOV    (SP), -(SP)
      SUB    #2, (SP)
      2$: RTI

```

4522
4523
4524
4525
4526
4527
4528
4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4540
4541
4542
4543
4544
4545
4546
4547
4548
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4570
4571
4572
4573
4574
4575
4576
4577

016602	012777	000001	162646
016610	012737	177500	001170
016616	000402		
016620	005037	001170	
016624	105777	162626	
016630	100431		
016632	005237	001170	
016636	001372		
016640	104420		
016642	001742		
016644	104401	016652	
016650	000403		
016660			
016660	011646		
016662	162716	000002	
016666	104402		
016670	104401	016676	
016674	000404		
016706			
016706	017746	162544	
016712	104402		
016714	000002		

```

.SBTTL CON.RESET - CONTROL RESET ROUTINE
.SBTTL CON.RDY - WAIT FOR CONTROL READY

:CON.RESET
:CON.RDY
:THIS ROUTINE IS CALLED BY USING 'CNT.RESET' WHICH IS ACTUALLY
:'TRAP' INSTRUCTION WITH THE LOWER BYTE ENCODED TO PROVIDE
:AN INDEX TO THE CONTROL-RESET ROUTINE BELOW.
:THE ROUTINE ISSUES A CONTROL RESET AND WAITS FOR
:THE 'CNTRL.RDY' FLAG TO SET. WHEN THE FLAG SETS
:AN EXIT IS MADE OUT OF THE ROUTINE. IF 'CNTRL-RDY'
:DOES NOT SET WITHIN A CERTAIN TIME AN ERROR MESSAGE
:   CNT RDY DIDN'T SET
:   PC=XXXXXX RKCS=XXXXXX
:IS GIVEN.
:THIS ROUTINE IS CALLED THROUGH THE 'TRAP' INSTRUCTION
:USING THE LOWER BYTE AS AN INDEX TO THIS ROUTINE.
:THE TRAP DECODER LOCATED AT 'STRAP'.

:CN.RDY
:THE CN.RDY ROUTINE IS CALLED BY USING CNT.RDY WHICH IS A TRAP
:INSTRUCTION WITH ITS LOWER BYTE ENCODED.
:THIS ROUTINE WAITS FOR THE CONTROL READY BIT TO SET AND WHEN IT
:SETS EXITS OUT. IF WITHIN A CERTAIN TIME CNTRL RDY DOES
:NOT SET AN ERROR IS REPORTED. WAITING TIME IS 883 MS FOR 11/20
:175 MS FOR 11/45 WITH BIPOLAR MEMORY.
CN.RST: MOV     #1, @RKCS      ;ISSUE A CONTROL RESET
        MOV     @-300, $REG3   ;SET UP COUNT
        BR      CN.RDY+4      ;SKIP OVER CN.RDY

CN.RDY: CLR     $REG3
1$:     TSTB     @RKCS          ;DID CNTRL-RDY SET?
        BMI     2$             ;YES, EXIT
        INC     $REG3          ;WAITED LONG?
        BNE     1$             ;IF NOT, GO BAK & WAIT
        TYPMSG
        MSG10
        TYPE     ,65$          ;;TYPE ASCIZ STRING
        BR       ,64$          ;;GET OVER THE ASCIZ
;;65$: .ASCIZ  <15><12>/PC=/
64$:   MOV     (SP), -(SP)
        SUB     #2, (SP)
        TYPOC          ;GO TYPE PC IN THE MAIN PROGRAM.
                        ; WHERE ERROR OCCURRED
        TYPE     ,67$          ;;TYPE ASCIZ STRING
        BR       ,66$          ;;GET OVER THE ASCIZ
;;67$: .ASCIZ  / RKCS=/
66$:   MOV     @RKCS, -(SP)    ;GET RKCS
        TYPOC          ;GO TYPE IT
2$:    RTI                  ;RETURN FROM THIS
                        ;ROUTINE TO THE MAIN

```

;PROGRAM

.SBTTL TST.RWS - WAIT FOR R/W/S RDY
;TST.RWS
;THIS ROUTINE WAITS FOR THE R/W/S READY TO SET AND RETURNS
;TO THE MAIN PROGRAM WHEN IT SETS. IF IT DOES NOT SET
;WITHIN A CERTAIN TIME AN ERROR IS REPORTED.
;WAITING TIME APPROX. 1040 MS FOR 11/20. 208 MS FOR 11/45

4588 016716 005037 001264
4589 016722 032777 000100 162522
4590 016730 001017
4591 016732 005237 001264
4592 016736 001371
4593 016740 032777 020000 162172
4594 016746 001010
4595 016750 104420 002027
4596 016754 104420 001733
4597 016760 011646
4598 016762 162716 000002
4599 016766 104402
4600 016770 000002
4601
4602
4603
4604
4605 016772 104401 001616
4606 016776 113746 001102
4607 017002 104402
4608 017004 000207
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4620
4621
4622
4623
4624
4625 017006
4626 017006 104407
4627 017010 032777 000400 162122
4628 017016 001053
4629
4630 017020 032777 040000 162112
4631 017026 001047
4632
4633 017030 000416

TST.RWS: CLR TIMER
1\$: BIT #100,ARKDS
BNE 2\$
INC TIMER
BNE 1\$
BIT #BIT13,2SWR
BNE 2\$
TYPMSG ,MSG12
TYPMSG ,MSG7
MOV (SP),-(SP)
SUB #2,(SP)
TYP0C
2\$: RTI

.SBTTL TEST ABORT ROUTINE
;ABRT

ABRT: TYPE MSG3
MOV \$TSTNM, -(SP)
TYP0C
RTS PC

;COMMON SUBROUTINES & HANDLERS

.SBTTL SCOPE HANDLER ROUTINE

;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
;AND LOAD THE TEST NUMBER(\$TSTNM) INTO THE DISPLAY REG.(DISPLAY(7:0))
;AND LOAD THE ERROR FLAG (\$ERFLG) INTO DISPLAY(15:08)
;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;SW14=1 LOOP ON TEST
;SW09=1 LOOP ON ERROR
;CALL
;* SCOPE ;;SCOPE=IOT
\$SCOPE:
CKSWR
BIT #SWB,2SWR ;;TEST FOR CHANGE IN SOFT-SWR
BNE \$OVER ;WAS SWB USED TO SELECT
;A TEST? IF YES, SKIP OVER
;THE REST, U ARE LOOPING ON
1\$: BIT #BIT14,2SWR
BNE \$OVER ;LOOP ON PRESENT TEST?
;YES IF SW14=1
;*****START OF CODE FOR THE XOR TESTER*****
\$XSTR: BR 6\$;IF RUNNING ON THE "XOR" TESTER CHANGE

```

4634      017032 013746 000004      MOV      2#ERRVEC -(SP)  ;: THIS INSTRUCTION TO A "NOP" (NOP=240)
4635      017036 012737 017056 000004      MOV      2#ERRVEC  ;: SAVE THE CONTENTS OF THE ERROR VEC OF
4636      017044 005737 177060      TST      2#177060  ;: SET FOR TIMEOUT
4637      017050 012637 000004      MOV      (SP)+, 2#ERRVEC  ;: TIME OUT ON XOR?
4638      017054 000421      BR      $SVLAD  ;: RESTORE THE ERROR VECTOR
4639      017056 022626      5$: CMP      (SP)+, (SP)+  ;: GO TO THE NEXT TEST
4640      017060 012637 000004      MOV      (SP)+, 2#ERRVEC  ;: CLEAR THE STACK AFTER A TIME OUT
4641      017064 000407      BR      7$  ;: RESTORE THE ERROR VECTOR
4642      017066      6$: ;*****END OF CODE FOR THE XOR TESTER*****  ;: LOOP ON THE PRESENT TEST
4643      017072 001412      2$: TSTB     $ERFLG  ;: HAS AN ERROR OCCURRED?
4644      017074 032777 001000 162036      BEQ      $SVLAD  ;: BR IF NO
4645      017102 001404      BIT      #BIT09, 2$SWR  ;: LOOP ON ERROR?
4646      017104 013737 001110 001106 7$: BEQ      4$  ;: BR IF NO
4647      017112 000415      MOV      $LPERR, $LPADR  ;: SET LOOP ADDRESS TO LAST SCOPE
4648      017114 105037 001103      BR      $OVER  ;:
4649      017120 105237 001102      4$: CLRB     $ERFLG  ;: ZERO THE ERROR FLAG
4650      017124 011637 001106      $SVLAD: INCB     $STSTM  ;: COUNT TEST NUMBERS
4651      017130 011637 001110      MOV      (SP), $LPADR  ;: SAVE SCOPE LOOP ADDRESS
4652      017134 005037 001204      MOV      (SP), $LPERR  ;: SAVE ERROR LOOP ADDRESS
4653      017140 112737 000001 001115      CLR      $ESCAPE  ;: CLEAR THE ESCAPE FROM ERROR ADDRESS
4654      017146 013777 001102 161766 $OVER: MOVB     #1, $ERMAX  ;: ONLY ALLOW ONE(1) ERROR ON NEXT "ES"
4655      017154 013716 001106      MOV      $STSTM, 2$DISPLAY  ;: DISPLAY TEST NUMBER
4656      017160 000002      MOV      $LPADR, (SP)  ;: FUDGE RETURN ADDRESS
4657      RTI  ;: FIXES PS

```

;*****

.SBTTL ERROR HANDLER ROUTINE

```

;: *SW15=1      HALT ON ERROR
;: *SW13=1      INHIBIT ERROR TYPEOUTS
;: *SW10=1      BELL ON ERROR
;: *SW09=1      LOOP ON ERROR
;: *SW12=1      CYCLE ON ERROR TO PREVIOUS 'SCOPE' STATEMENT
;: *GO TO ERRYP ON ERROR
;: *NOT FROM SYSMAC

```

```

4673      017162 104407      $ERROR: CKSWR  ;: CHECK FOR SOFTWARE SWITCH REGISTER REQUEST
4674      017164 105237 001103 7$: INCB     $ERFLG  ;: SET THE ERROR FLAG
4675      017170 001775      BEQ      7$  ;: DON'T LET THE FLAG GO TO ZERO
4676      017172 013777 001102 161742      MOV      $STSTM, 2$DISPLAY
4677      017200 032777 002000 161732      BIT      #SW10, 2$SWR
4678      017206 001402      BEQ      1$  ;:
4679      017210 104401 001206      TYPE     $BELL
4680      017214 005237 001112      1$: INC      $ERTTL
4681      017220 011637 001116      MOV      (SP), $ERRPC
4682
4683      017224 032777 000004 161706      BIT      #SW2, 2$SWR  ;: DROP THE DRIVE?
4684      017232 001404      BEQ      5$  ;: SW NOT SET, SKIP
4685      017234 023727 001112 000006      CMP      $ERTTL, #6  ;: MORE THAN 6 ERRORS ON THIS DRIVE?
4686      017242 101044      BHI      6$  ;: YES, DROP THE DRIVE
4687
4688      017244 162737 000002 001116 5$: SUB      #2, $ERRPC
4689      017252 117737 161640 001114      MOVB     2$ERRPC, $ITEMB

```

LO7

4690	017260	032777	020000	161652	BIT	#SW13,2SWR		
4691	017266	001004			BNE	25		
4692	017270	004737	017446		JSR	PC,2ERRTYP		
4693	017274	104401	001213		TYPE	\$CRLF		
4694	017300	023737	000042	000046	25:	CMP	2#42,2#46	ARE WE IN ACT11 AUTO MODE?
4695	017306	001403			BEQ	.+10		YES, HALT ON ERROR
4696	017310	005777	161624		TST	2SWR		SWR15 (HALT ON ERROR) SET?
4697	017314	100002			BPL	35		BRANCH IF NOT
4698	017316	000000			HALT			HALT ON ERROR
4699	017320	104407			CKSWR			CHECK FOR SOFTWARE SWITCH REGISTER REQUEST
4700	017322	032777	010000	161610	35:	BIT	#SW12,2SWR	
4701	017330	001402			BEQ	.+6		
4702	017332	013716	001106		MOV	\$LPADR,(SP)		
4703	017336	032777	001000	161574	BIT	#SW09,2SWR		
4704	017344	001402			BEQ	45		
4705	017346	013716	001110		MOV	\$LPERR,(SP)		
4706	017352	000002			45:	RTI		
4707								
4708	017354	013746	001226		65:	MOV	DRVPT,-(SP)	GET POINTER TO DRIVE #
4709	017360	162716	000002		SUB	#2,(SP)		
4710	017364	042736	000377		BIC	#377,2(SP)+		CLEAR THE DRIVE PRESENT FLAG
4711	017370	104401	002064		TYPE	MSG14		
4712	017374	013746	001230		MOV	DRIVAD,-(SP)		
4713	017400	000241			CLC			GET THE DRIVE #
4714	017402	006116			ROL	(SP)		
4715	017404	006116			ROL	(SP)		
4716	017406	006116			ROL	(SP)		
4717	017410	006116			ROL	(SP)		
4718	017412	104402			TYPOC			TYPE IT OUT
4719	017414	104401	017422		TYPE	655		TYPE ASCIZ STRING
4720	017420	000405			BR	645		GET OVER THE ASCIZ
4721					655:	.ASCIZ	/ DROPPED/	
4722	017434				645:			
4723	017434	105337	001224		DECB	DRIVS		DECRMT # OF DRIVS PRESENT
4724	017440	022626			CMP	(SP)+,(SP)+		RESTORE STACK
4725	017442	000137	015232		JMP	BTEOP		EXIT
4726								
4727	017446				ERRTYP:			
4728	017446	104401	001213		TYPE	\$CRLF		"CARRIAGE RETURN" & LINE FEED"
4729	017452	010046			MOV	RO,-(SP)		SAVE RO
4730	017454	005000			CLR	RO		PICKUP THE ITEM INDEX
4731	017456	153700	001114		BISB	2#5ITEMB,RO		
4732	017462	001011			BNE	15		IF ITEM NUMBER IS ZERO, JUST
4733								
4734								
4735	017464	013746	001116		MOV	\$ERRPC,-(SP)		TYPE THE PC OF THE ERROR
4736								SAVE \$ERRPC FOR TYPEOUT
4737	017470	104402			TYPOC			ERROR ADDRESS
4738	017472	104401			TYPE			GO TYPE--OCTAL ASCII(ALL DIGITS)
4739	017474	001733			MSG7			
4740	017476	013746	001116		MOV	\$ERRPC,-(SP)		
4741	017502	104402			TYPOC			
4742	017504	000440			BR	65		GET OUT
4743	017506	005300			15:	DEC	RO	ADJUST THE INDEX SO THAT IT WILL
4744	017510	006300			ASL	RO		WORK FOR THE ERROR TABLE
4745	017512	006300			ASL	RO		

MO7

4746	017514	006300		ASL	RO	
4747	017516	062700	002122	ADD	#SERRTB,RO	; FORM TABLE POINTER
4748	017522	012037	017532	MOV	(RO)+,2\$	; PICKUP "ERROR MESSAGE" POINTER
4749	017526	001404		BEQ	3\$	; SKIP TYPEOUT IF NOT POINTER
4750	017530	104401		TYPE		; TYPE THE "ERROR MESSAGE"
4751	017532	000000		2\$: .WORD	0	; "CARRIAGE RETURN" & LINE FEED"
4752	017534	104401	001213	TYPE	, \$CRLF	; PICKUP "DATA HEADER" POINTER
4753	017540	032777	004000	3\$: BIT	#SW11,2SWR	; DUMP OUT ALL RK REGISTERS
4754	017546	001042		BNE	10\$	; YES, BRANCH
4755	017550	012037	017560	MOV	(RO)+,4\$	; PICKUP "DATA HEADER" POINTER
4756	017554	001412		BEQ	5\$	; SKIP TYPEOUT IF 0
4757	017556	104401		TYPE		; TYPE THE "DATA HEADER"
4758	017560	000000		4\$: .WORD	0	; "DATA HEADER" POINTER GOES HERE
4759	017562	104401	001213	TYPE	, \$CRLF	; "CARRIAGE RETURN" & LINE FEED"
4760	017566	062700	000002	ADD	#2,RO	; FORM POINTER TO TERMINATOR
4761	017572	005710		TST	(RO)	; IS THE TERMINATOR 0?
4762	017574	001017		BNE	9\$	; IF NOT, BRANCH
4763	017576	162700	000002	SUB	#2,RO	; YES, IT IS 0. REPOINT TO "DATA"
4764						; GO TYPE OUT DATA AS USUAL
4765	017602	011000		5\$: MOV	(RO),RO	; PICKUP "DATA TABLE" POINTER
4766	017604	001004		BNE	7\$	; GO TYPE THE DATA
4767	017606	012600		6\$: MOV	(SP)+,RO	; RESTORE RO
4768	017610	104401	001213	TYPE	, \$CRLF	; "CARRIAGE RETURN" & LINE FEED"
4769	017614	000207		RTS	PC	; RETURN
4770	017616			7\$:		
4771	017616	013046		MOV	2(RO)+,-(SP)	; SAVE 2(RO)+ FOR TYPEOUT
4772	017620	104402		TYPOC		; GO TYPE--OCTAL ASCII(ALL DIGITS)
4773	017622	005710		TST	(RO)	; IS THERE ANOTHER NUMBER?
4774	017624	001770		BEQ	6\$	; BR IF NO
4775	017626	104401	002110	TYPE	,BLNKS2	
4776	017632	000771		BR	7\$	
4777	017634	004770	000000	9\$: JSR	PC,2(RO)	; GO TO THE SPECIAL ERROR
4778						; DATA HANDLING SUBROUTINE
4779						; NOTE THAT THIS ROUTINE IS
4780						; THE ONE INDICATED IN THE
4781						; LAST WORD OF AN ERROR
4782						; ITEM IN THE ERROR TABLE
4783						; (STARTING AT SERRTB)
4784	017640	104401		TYPE		
4785	017642	001733		MSG7		
4786	017644	013746	001116	MOV	\$ERRPC,-(SP)	
4787	017650	104402		TYPOC		
4788	017652	000755		BR	6\$	; GO BACK TO THE EXIT POINT
4789						; FOR 'ERRATYP'
4790						
4791	017654	004737	017662	10\$: JSR	PC,DMPREG	
4792	017660	000752		BR	6\$	
4793						
4794						
4795						
4796						
4797						
4798						
4799	017662					
4800	017662	104401	017670	DMPREG: TYPE	65\$	; TYPE ASCII STRING
4801	017666	000441		BR	64\$	; GET OVER THE ASCII

;DMPREG

;DUMPS OUT ALL RK REGISTERS WHEN SW 11 IS SET

DMPREG:

TYPE 65\$
 BR 64\$

; TYPE ASCII STRING
 ; GET OVER THE ASCII

NO7

```

4802
4803 017772
4804 017772 013746 001116
4805 017776 104402
4806 020000 104401 002110
4807 020004 010046
4808 020006 012700 001452
4809 020012 013046
4810 020014 104402
4811 020016 104401 002110
4812 020022 020027 001466
4813 020026 003771
4814 020030 012600
4815 020032 000207
4816
4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4830
4831
4832
4833 020034
4834 020034 010046
4835 020036 010146
4836 020040 010246
4837 020042 010346
4838 020044 010546
4839 020046 012746 020200
4840 020052 016605 000020
4841 020056 100004
4842 020060 005405
4843 020062 112766 000055 000001
4844 020070 005000
4845 020072 012703 020250
4846 020076 112723 000040
4847 020102 005002
4848 020104 016001 020240
4849 020110 160105
4850 020112 002402
4851 020114 005202
4852 020116 000774
4853 020120 060105
4854 020122 005702
4855 020124 001002
4856 020126 105716
4857 020130 100407

```

```

655: .ASCIZ <15><12>/ PC RKDS RKER RKCS RKWC RKBA RKDA RKDB//
645:
MOV $ERRPC, -(SP)
TYPDC
TYPE BLNKS2
MOV RO, -(SP)
MOV #RKDS, RO
1$: MOV 2(RO)+, -(SP)
TYPDC
TYPE BLNKS2
CMP RO, #RKDB
BLE 1$
MOV (SP)+, RO
RTS PC

```

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

;*****
;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
;SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
;NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
;BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
;REPLACED WITH SPACES.
;CALL:
;* MOV NUM, -(SP) ;PUT THE BINARY NUMBER ON THE STACK
;* TYPDS ;GO TO THE ROUTINE
$TYPDS:
MOV RO, -(SP) ;PUSH RO ON STACK
MOV R1, -(SP) ;PUSH R1 ON STACK
MOV R2, -(SP) ;PUSH R2 ON STACK
MOV R3, -(SP) ;PUSH R3 ON STACK
MOV R5, -(SP) ;PUSH R5 ON STACK
MOV #20200, -(SP) ;SET BLANK SWITCH AND SIGN
MOV 20(SP), R5 ;GET THE INPUT NUMBER
BPL 1$ ;BR IF INPUT IS POS.
NEG R5 ;MAKE THE BINARY NUMBER POS.
MOVB #'-, 1(SP) ;MAKE THE ASCII NUMBER NEG.
1$: CLR RO ;ZERO THE CONSTANTS INDEX
MOV #SDBLK, R3 ;SETUP THE OUTPUT POINTER
MOVB #' , (R3)+ ;SET THE FIRST CHARACTER TO A BLANK
2$: CLR R2 ;CLEAR THE BCD NUMBER
MOV $DTBL(RO), R1 ;GET THE CONSTANT
3$: SUB R1, R5 ;FORM THIS BCD DIGIT
BLT 4$ ;BR IF DONE
INC R2 ;INCREASE THE BCD DIGIT BY 1
BR 3$
4$: ADD R1, R5 ;ADD BACK THE CONSTANT
TST R2 ;CHECK IF BCD DIGIT=0
BNE 5$ ;FALL THROUGH IF 0
TSTB (SP) ;STILL DOING LEADING 0'S?
BMI 7$ ;BR IF YES

```

MD-11-DZRL-E, RK11-RK05 DYNAMIC TEST
DZRL-E.P11 26-APR-77 12:27

MACY11 30(1046) 14-JUL-77 08:03 PAGE 92
CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

4858	020132	106316		55:	ASLB	(SP)	MS0?
4859	020134	103003			BCC	65	BR IF NO
4860	020136	116663	000001 177777		MOVB	1(SP),-1(R3)	YES--SET THE SIGN
4861	020144	052702	000063	65:	BIS	8'0,R2	MAKE THE BCD DIGIT ASCII
4862	020150	052702	000040	75:	BIS	8' R2	MAKE IT A SPACE IF NOT ALREADY A DIGIT
4863	020154	110223			MOVB	R2,(R3)+	PUT THIS CHARACTER IN THE OUTPUT BUFFER
4864	020156	005720			TST	(R0)+	JUST INCREMENTING
4865	020160	020027	000010		CHP	R0,810	CHECK THE TABLE INDEX
4866	020164	002746			BLT	25	GO DO THE NEXT DIGIT
4867	020166	003002			BGT	85	GO TO EXIT
4868	020170	010502			MOV	R5,R2	GET THE LSD
4869	020172	000764			BR	65	GO CHANGE TO ASCII
4870	020174	105726		85:	TSTB	(SP)+	WAS THE LSD THE FIRST NON-ZERO?
4871	020176	100003			BPL	95	BR IF NO
4872	020200	116663	177777 177776		MOVB	-1(SP),-2(R3)	YES--SET THE SIGN FOR TYPING
4873	020206	105013		95:	CLRB	(R3)	SET THE TERMINATOR
4874	020210	012605			MOV	(SP)+,R5	POP STACK INTO R5
4875	020212	012603			MOV	(SP)+,R3	POP STACK INTO R3
4876	020214	012602			MOV	(SP)+,R2	POP STACK INTO R2
4877	020216	012601			MOV	(SP)+,R1	POP STACK INTO R1
4878	020220	012600			MOV	(SP)+,R0	POP STACK INTO R0
4879	020222	104401	020250		TYPE	\$DBLK	NOW TYPE THE NUMBER
4880	020226	016666	000002 000004		MOV	2(SP),4(SP)	ADJUST THE STACK
4881	020234	012616			MOV	(SP)+,(SP)	
4882	020236	000002			RTI		::RETURN TO USER
4883	020240	023420		\$DTBL:	10000.		
4884	020242	001750			1000.		
4885	020244	000144			100.		
4886	020246	000012			10.		
4887	020250	000004		\$DBLK:	.BLKW 4		
4888							
4889					.SBTTL	TYPE ROUTINE	
4890							
4891					*****		
4892					*ROUTINE TO TYPE ASCII MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.		
4893					*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.		
4894					*NOTE1: \$NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.		
4895					*NOTE2: \$FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.		
4896					*NOTE3: \$FILLC CONTAINS THE CHARACTER TO FILL AFTER.		
4897					*		
4898					*CALL:		
4899					*1) USING A TRAP INSTRUCTION		
4900					* TYPE ,MESADR	::MESADR IS FIRST ADDRESS OF AN ASCII STRING	
4901					*OR		
4902					* TYPE		
4903					* MESADR		
4904					*		
4905							
4906	020260	105737	001157	\$TYPE:	TSTB	\$TPFLG	::IS THERE A TERMINAL?
4907	020264	100002			BPL	15	BR IF YES
4908	020266	000000			HALT		HALT HERE IF NO TERMINAL
4909	020270	000407			BR	35	LEAVE
4910	020272	010046		15:	MOV	R0,-(SP)	SAVE R0
4911	020274	017600	000002		MOV	22(SP),R0	GET ADDRESS OF ASCII STRING
4912	020300	112046		25:	MOVB	(R0)+,-(SP)	PUSH CHARACTER TO BE TYPED ONTO STACK
4913	020302	001005			BNE	45	BR IF IT ISN'T THE TERMINATOR

008

4914	020304	005726			TST	(SP)+	:: IF TERMINATOR POP IT OFF THE STACK
4915	020306	012600			60S: MOV	(SP)+, R0	:: RESTORE R0
4916	020310	062716	000002		3S: ADD	#2, (SP)	:: ADJUST RETURN PC
4917	020314	000002			RTI		:: RETURN
4918	020316	122716	000011		4S: CMPB	#HT, (SP)	:: BRANCH IF <HT>
4919	020322	001430			BEQ	#S	
4920	020324	122716	000200		CMPB	#CRLF, (SP)	:: BRANCH IF NOT <CRLF>
4921	020330	001006			BNE	#S	
4922	020332	005726			TST	(SP)+	:: POP <CR><LF> EQUIV
4923	020334	104401			TYPE		:: TYPE A CR AND LF
4924	020336	001213			SCRLF		
4925	020340	105037	020474		CLRB	\$CHARCNT	:: CLEAR CHARACTER COUNT
4926	020344	000755			BR	#S	:: GET NEXT CHARACTER
4927	020346	004737	020430		5S: JSR	PC, \$TYPEC	:: GO TYPE THIS CHARACTER
4928	020352	123726	001156		6S: CMPB	\$FILLC, (SP)+	:: IS IT TIME FOR FILLER CHARS.?
4929	020356	001350			BNE	#S	:: IF NO GO GET NEXT CHAR.
4930	020360	013746	001154		MOV	\$NULL, -(SP)	:: GET # OF FILLER CHARS. NEEDED
4931							:: AND THE NULL CHAR.
4932	020364	105366	000001		7S: DECB	1(SP)	:: DOES A NULL NEED TO BE TYPED?
4933	020370	002770			BLT	#S	:: BR IF NO--GO POP THE NULL OFF OF STACK
4934	020372	004737	020430		JSR	PC, \$TYPEC	:: GO TYPE A NULL
4935	020376	105337	020474		DECB	\$CHARCNT	:: DO NOT COUNT AS A COUNT
4936	020402	000770			BR	#S	:: LOOP
4937							
4938							
4939							
4940	020404	112716	000040		8S: MOVB	#', (SP)	:: REPLACE TAB WITH SPACE
4941	020410	004737	020430		9S: JSR	PC, \$TYPEC	:: TYPE A SPACE
4942	020414	132737	000007	020474	BITB	#7, \$CHARCNT	:: BRANCH IF NOT AT
4943	020422	001372			BNE	#S	:: TAB STOP
4944	020424	005726			TST	(SP)+	:: POP SPACE OFF STACK
4945	020426	000724			BR	#S	:: GET NEXT CHARACTER
4946	020430	105777	160514		\$TYPEC: TSTB	\$STPS	:: WAIT UNTIL PRINTER IS READY
4947	020434	100375			BPL	\$TYPEC	
4948	020436	116677	000002	160506	MOVB	2(SP), \$STPB	:: LOAD CHAR TO BE TYPED INTO DATA REG.
4949	020444	122766	000015	000002	CMPB	#CR, 2(SP)	:: IS CHARACTER A CARRIAGE RETURN?
4950	020452	001003			BNE	#S	:: BRANCH IF NO
4951	020454	105037	020474		CLRB	\$CHARCNT	:: YES--CLEAR CHARACTER COUNT
4952	020460	000406			BR	\$TYPEX	:: EXIT
4953	020462	122766	000012	000002	1S: CMPB	#LF, 2(SP)	:: IS CHARACTER A LINE FEED?
4954	020470	001402			BEQ	\$TYPEX	:: BRANCH IF YES
4955	020472	105227			INCB	(PC)+	:: COUNT THE CHARACTER
4956	020474	000000			\$CHARCNT: .WORD	0	:: CHARACTER COUNT STORAGE
4957	020476	000207			\$TYPEX: RTS	PC	
4958							
4959							
4960							
4961							
4962							
4963							
4964							
4965							
4966							
4967							
4968							
4969							

.SBTTL INTEGER MULTIPLY ROUTINE

```

;*****
;CALL
;*  MOV    MULTIPLIER, -(SP)
;*  MOV    MULTIPLICAND, -(SP)
;*  JSR    PC, $MULT
;*  RETURN ;:PRODUCT IS ON THE STACK
;*
```

008

4970			;	*	STACK	PRODUCT	
4971			;	*	-----	-----	
4972			;	*	TOP	LSB'S	
4973			;	*	+2	MSB'S	
4974							
4975	020500						
4976	020500	010046					
4977	020502	010146					
4978	020504	010246					
4979	020506	005046					
4980	020510	016601	000012				
4981	020514	100002					
4982	020516	005216					
4983	020520	005401					
4984	020522	016602	000014				
4985	020526	100002					
4986	020530	005316					
4987	020532	005402					
4988	020534	012746	000021				
4989	020540	005000					
4990	020542	103001					
4991	020544	060200					
4992	020546	006000					
4993	020550	006001					
4994	020552	005316					
4995	020554	001372					
4996	020556	022616					
4997	020560	001403					
4998	020562	005400					
4999	020564	005401					
5000	020566	005600					
5001	020570	005726					
5002	020572	010066	000012				
5003	020576	010166	000010				
5004	020602	012602					
5005	020604	012601					
5006	020606	012600					
5007	020610	000207					
5008							
5009							
5010							
5011							
5012							
5013	020612	000000					
5014	020614	000000					
5015	020616	000000					
5016	020620	000001					
5017		020621					
5018		020622					
5019							
5020							
5021							
5022							
5023							
5024							
5025							

SMULT:	MOV	R0,-(SP)	;; PUSH R0 ON STACK
	MOV	R1,-(SP)	;; PUSH R1 ON STACK
	MOV	R2,-(SP)	;; PUSH R2 ON STACK
	CLR	-(SP)	;; CLEAR THE SIGN KEY
	MOV	12(SP),R1	;; GET THE MULTIPLICAND
	BPL	1\$	;; BR IF PLUS
	INC	(SP)	;; SET THE SIGN KEY
	NEG	R1	;; MAKE THE MULTIPLICAND POSTIVE
1\$:	MOV	14(SP),R2	;; GET THE MULTIPLIER
	BPL	2\$	;; BR IF PLUS
	DEC	(SP)	;; UPDATE THE SIGN KEY
	NEG	R2	;; MAKE THE MULTIPLIER POSTIVE
2\$:	MOV	#17,-(SP)	;; SET THE LOOP COUNT
	CLR	R0	;; SETUP FOR THE MULTIPLY LOOP
3\$:	BCC	4\$	;; DON'T ADD IF MULTIPLICAND = 0
	ADD	R2,R0	
4\$:	ROR	R0	;; POSITION THE PARTIAL PRODUCT AND
	ROR	R1	;; THE MULTIPLICAND
	DEC	(SP)	;; HAS ALL BITS OF THE MULTIPLICAND BEEN DONE?
	BNE	3\$	;; BR IF NO
	CMP	(SP)+,(SP)	;; SHOULD PRODUCT BE NEGATIVE?
	BEQ	5\$	;; GO TO EXIT IF NO
	NEG	R0	;; YES--SO MAKE IT SO
	NEG	R1	
	SBC	R0	
5\$:	TST	(SP)+	;; CLEAR SIGN INFO. OFF OF STACK
	MOV	R0,12(SP)	;; PUT THE PRODUCT ON THE STACK (MSB'S.
	MOV	R1,10(SP)	;; LSB'S
	MOV	(SP)+,R2	;; POP STACK INTO R2
	MOV	(SP)+,R1	;; POP STACK INTO R1
	MOV	(SP)+,R0	;; POP STACK INTO R0
	RTS	PC	

.SBTTL TTY INPUT ROUTINE

;;*****

.ENABL LSB

STKCNT: .WORD 0 ;; NUMBER OF ITEMS IN QUEUE

STKQIN: .WORD 0 ;; INPUT POINTER

STKQOUT: .WORD 0 ;; OUTPUT POINTER

STKQSRT: .BLKB 1 ;; TTY KEYBOARD QUEUE

STKQEND=.

.EVEN

;;TK INITIALIZE ROUTINE

;;THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE

;;SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT

;;

;;CALL:

;; JSR PC,STKINT

```

5026          : *      RETURN
5027          :
5028 020622 005037 020612 $TKINT: CLR      $TKCNT      ;; CLEAR COUNT OF ITEMS IN QUEUE
5029 020626 012737 020620 020614 MOV      $TKQSR, $TKQIN ;; MOVE THE STARTING ADDRESS OF THE
5030 020634 013737 020614 020616 MOV      $TKQIN, $TKQOUT ;; QUEUE INTO THE INPUT & OUTPUT POINTERS.
5031 020642 012737 020672 000060 MOV      $TKSRV, $TKVEC ;; INITIALIZE THE KEYBOARD VECTOR
5032 020650 012737 000200 000062 MOV      $200, $TKVEC+2 ;; "BR" LEVEL 4
5033 020656 005777 160264 TST      $TKB      ;; CLEAR DONE FLAG
5034 020662 012777 000100 160254 MOV      $100, $TKS ;; ENABLE TTY KEYBOARD INTERRUPT
5035 020670 000207 RTS      PC      ;; RETURN TO CALLER
5036
5037          : *TK SERVICE ROUTINE
5038          : *THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
5039          : *BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
5040          : *IT IN THE QUEUE.
5041
5042 020672 117746 160250 $TKSRV: MOVB    $TKB, -(SP) ;; PICKUP THE CHARACTER
5043 020676 042716 177600 BIC      $1C177, (SP) ;; STRIP THE JUNK
5044 020702 021627 000007 1$: CMP      (SP), $7 ;; IS IT A CONTROL G?
5045 020706 001004 BNE      2$ ;; BRANCH IF NO
5046 020710 022737 000176 001140 CMP      $SWREG, SWR ;; IS SOFT-SWR SELECTED?
5047 020716 001500 BEQ      6$ ;; GO TO SWR CHANGE
5048
5049 020720 022737 000001 020612 2$: CMP      $1, $TKCNT ;; IS THE QUEUE FULL?
5050 020726 001004 BNE      3$ ;; BRANCH IF NO
5051 020730 104401 001206 TYPE      $BELL ;; RING THE TTY BELL
5052 020734 005726 TST      (SP)+ ;; CLEAN CHARACTER OFF OF STACK
5053 020736 000451 BR      5$ ;; EXIT
5054 020740 021627 000023 3$: CMP      (SP), $23 ;; IS IT A CONTROL-S?
5055 020744 001021 BNE      32$ ;; BRANCH IF NO
5056 020746 005077 160172 CLR      $TKS ;; DISABLE TTY KEYBOARD INTERRUPTS
5057 020752 005726 TST      (SP)+ ;; CLEAN CHAR OFF STACK
5058 020754 105777 160164 31$: TSTB    $TKS ;; WAIT FOR A CHAR
5059 020760 100375 BPL      31$ ;; LOOP UNTIL ITS THERE
5060 020762 117746 160160 MOVB     $TKB, -(SP) ;; GET THE CHARACTER
5061 020766 042716 177600 BIC      $1C177, (SP) ;; MAKE IT 7-BIT ASCII
5062 020772 022627 000021 CMP      (SP)+, $21 ;; IS IT A CONTROL-Q?
5063 020776 001366 BNE      31$ ;; BRANCH IF NO
5064 021000 012777 000100 160136 MOV      $100, $TKS ;; REENABLE TTY KEYBOARD INTERRUPTS
5065 021006 000002 RTI      ;; RETURN
5066 021010 005237 020612 32$: INC      $TKCNT ;; COUNT THIS CHARACTER
5067 021014 021627 000140 CMP      (SP), $140 ;; IS IT UPPER CASE?
5068 021020 002405 BLT      4$ ;; BRANCH IF YES
5069 021022 021627 000175 CMP      (SP), $175 ;; IS IT A SPECIAL CHAR?
5070 021026 003002 BGT      4$ ;; BRANCH IF YES
5071 021030 042716 000040 BIC      $40, (SP) ;; MAKE IT UPPER CASE
5072 021034 112677 177554 4$: MOVB    (SP)+, $TKQIN ;; AND PUT IT IN QUEUE
5073 021040 005237 020614 INC      $TKQIN ;; UPDATE THE POINTER
5074 021044 023727 020614 020621 CMP      $TKQIN, $TKQEND ;; GO OFF THE END?
5075 021052 001003 BNE      5$ ;; BRANCH IF NO
5076 021054 012737 020620 020614 MOV      $TKQSR, $TKQIN ;; RESET THE POINTER
5077 021062 000002 5$: RTI      ;; RETURN
5078
5079          : *****
5080          : *SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
5081

```

```

5082 ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
5083 ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
5084 ;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
5085 021064 022737 000176 001140 $CKSWR: CMP      $SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
5086 021072 001104          BNE      15$          ;;EXIT IF NOT
5087 021074 105777 160044          TSTB     $STKS          ;;IS A CHAR WAITING?
5088 021100 100101          BPL      15$          ;;IF NOT, EXIT
5089 021102 117746 160040          MOVB     $STKB,-(SP)      ;;YES
5090 021106 042716 177600          BIC      $C177,(SP)      ;;MAKE IT 7-BIT ASCII
5091 021112 021627 000007          CMP      (SP),#7        ;;IS IT A CONTROL-G?
5092 021116 001300          BNE      25$          ;;IF NOT, PUT IT IN THE TTY QUEUE
5093                                     ;;AND EXIT
5094
5095 ;*****
5096 ;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
5097 ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
5098 ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
5099 021120 123727 001134 000001 6$: CMPB     $AUTOB,#1      ;;ARE WE RUNNING IN AUTO-MODE?
5100 021126 001674          BEQ      25$          ;;BRANCH IF YES
5101 021130 005726          TST      (SP)+        ;;CLEAR CONTROL-G OFF STACK
5102 021132 004737 020622          JSR      PC,$TKINT      ;;FLUSH THE TTY INPUT QUEUE
5103 021136 005077 160002          CLR      $STKS          ;;DISABLE TTY KEYBOARD INTERRUPTS
5104 021142 112737 000001 001135          MOVB     #1,$INTAG      ;;SET INTERRUPT MODE INDICATOR
5105
5106 021150 104401 021727          TYPE      ,SCNTLG          ;;ECHO THE CONTROL-G (+G)
5107 021154 104401 021734          $GTSWR: TYPE      ,SMSWR          ;;TYPE CURRENT CONTENTS
5108 021160 013746 000176          MOV      $SWREG,-(SP)      ;;SAVE SWREG FOR TYPEOUT
5109 021164 104402          TYPEOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
5110 021166 104401 021745          TYPE      ,SMNEW          ;;PROMPT FOR NEW SWR
5111 021172 005046          19$: CLR      -(SP)          ;;CLEAR COUNTER
5112 021174 005046          CLR      -(SP)          ;;THE NEW SWR
5113 021176 105777 157742          7$: TSTB     $STKS          ;;CHAR THERE?
5114 021202 100375          BPL      7$          ;;IF NOT TRY AGAIN
5115
5116 021204 117746 157736          MOVB     $STKB,-(SP)      ;;PICK UP CHAR
5117 021210 042716 177600          BIC      $C177,(SP)      ;;MAKE IT 7-BIT ASCII
5118
5119
5120
5121 021214 021627 000025          9$: CMP      (SP),#25        ;;IS IT A CONTROL-U?
5122 021220 001005          BNE      10$          ;;BRANCH IF NOT
5123 021222 104401 021722          TYPE      ,SCNTLU          ;;YES, ECHO CONTROL-U (+U)
5124 021226 062706 000006          20$: ADD      #6,SP          ;;IGNORE PREVIOUS INPUT
5125 021232 000757          BR       19$          ;;LET'S TRY IT AGAIN
5126
5127
5128 021234 021627 000015          10$: CMP      (SP),#15        ;;IS IT A <CR>?
5129 021240 001022          BNE      16$          ;;BRANCH IF NO
5130 021242 005766 000004          TST      4(SP)          ;;YES, IS IT THE FIRST CHAR?
5131 021246 001403          BEQ      11$          ;;BRANCH IF YES
5132 021250 016677 000002 157662          MOV      2(SP),2SWR      ;;SAVE NEW SWR
5133 021256 062706 000006          11$: ADD      #6,SP          ;;CLEAR UP STACK
5134 021262 104401 001213          14$: TYPE      ,SCRLF          ;;ECHO <CR> AND <LF>
5135 021266 123727 001135 000001          CMPB     $INTAG,#1      ;;RE-ENABLE TTY KBD INTERRUPTS?
5136 021274 001003          BNE      15$          ;;BRANCH IF NOT
5137 021276 012777 000100 157640          MOV      #100,$STKS      ;;RE-ENABLE TTY KBD INTERRUPTS

```

G08

5138 021304 000002
5139 021306 004737 020430
5140 021312 021627 000060
5141 021316 002420
5142 021320 021627 000067
5143 021324 003015
5144 021326 042726 000060
5145 021332 005766 000002
5146 021336 001403
5147 021340 006316
5148 021342 006316
5149 021344 006316
5150 021346 005266 000002
5151 021352 056616 177776
5152 021356 000707
5153 021360 104401 001212
5154 021364 000720
5155
5156
5157
5158
5159
5160
5161
5162
5163
5164
5165
5166 021366 011646
5167 021370 016666 000004 000002
5168 021376 005066 000004
5169 021402 005046
5170 021404 012746 021412
5171 021410 000002
5172 021412
5173 021412 005737 020612
5174 021416 001775
5175 021420 005337 020612
5176 021424 117766 177166 000004
5177 021432 005237 020616
5178 021436 023727 020616 020621
5179 021444 001003
5180 021446 012737 020620 020616
5181 021454 000002
5182
5183
5184
5185
5186
5187
5188
5189 021456 010346
5190 021460 005046
5191 021462 012703 021712
5192 021466 022703 021722
5193 021472 101456

```

15$: RTI                                ;; RETURN
16$: JSR PC,STPEC                        ;; ECHO CHAR
    CMP (SP),#60                        ;; CHAR < 0?
    BLT 18$                             ;; BRANCH IF YES
    CMP (SP),#67                        ;; CHAR > 7?
    BGT 18$                             ;; BRANCH IF YES
    BIC #60,(SP)+                       ;; STRIP-OFF ASCII
    TST 2(SP)                           ;; IS THIS THE FIRST CHAR
    BEQ 17$                             ;; BRANCH IF YES
    ASL (SP)                            ;; NO, SHIFT PRESENT
    ASL (SP)                            ;; CHAR OVER TO MAKE
    ASL (SP)                            ;; ROOM FOR NEW ONE.
17$: INC 2(SP)                           ;; KEEP COUNT OF CHAR
    BIS -2(SP),(SP)                    ;; SET IN NEW CHAR
    BR 7$                               ;; GET THE NEXT ONE
18$: TYPE $QUES                         ;; TYPE ?(CR)(LF)
    BR 20$                             ;; SIMULATE CONTROL-U
.DSABL L56

;*****
;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
;CALL:
;* RDCHR                                ;; GET A CHARACTER FROM THE QUEUE
;* RETURN HERE                        ;; CHARACTER IS ON THE STACK
;*                                     ;; WITH PARITY BIT STRIPPED OFF
;

$RDCHR: MOV (SP),-(SP)                  ;; PUSH DOWN THE PC AND
    MOV 4(SP),2(SP)                    ;; THE PS
    CLR 4(SP)                          ;; GET READY FOR A CHARACTER
    CLR -(SP)                          ;; PUT NEW PS ON STACK
    MOV #64$,-(SP)                     ;; PUT NEW PC ON STACK
    RTI                                ;; POP NEW PC AND PS

64$:   TST $TKCNT                       ;; WAIT ON A CHARACTER
1$:    BEQ 1$                           ;;
    DEC $TKCNT                         ;; DECREMENT THE COUNTER
    MOVB $TKQOUT,4(SP)                 ;; GET ONE CHARACTER
    INC $TKQOUT                        ;; UPDATE THE POINTER
    CMP $TKQOUT,#$TKQEND               ;; DID IT GO OFF OF THE END?
    BNE 2$                             ;; BRANCH IF NO
    MOV # $TKQSRT,$TKQOUT              ;; RESET THE POINTER
2$:    RTI                             ;; RETURN
;*****
;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
;CALL:
;* RDLIN                                ;; INPUT A STRING FROM THE TTY
;* RETURN HERE                        ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
;*                                     ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
;

$RDLIN: MOV R3,-(SP)                   ;; SAVE R3
    CLR -(SP)                          ;; CLEAR THE RUBOUT KEY
1$:    MOV # $TTYIN,R3                 ;; GET ADDRESS
2$:    CMP # $TTYIN+8.,R3              ;; BUFFER FULL?
    BLOS 4$                            ;; BR IF YES

```

5194	021474	104410			RDCHR		:: GO READ ONE CHARACTER FROM THE TTY
5195	021476	112613			MOVB	(SP)+, R3	:: GET CHARACTER
5196	021500	122713	000177		CMPB	#177, (R3)	:: IS IT A RUBOUT
5197	021504	001022			BNE	5\$	:: BR IF NO
5198	021506	005716			TST	(3P)	:: IS THIS THE FIRST RUBOUT?
5199	021510	001007			BNE	6\$	:: BR IF NO
5200	021512	112737	000134	021710	MOVB	#'\, 9\$	:: TYPE A BACK SLASH
5201	021520	104401	021710		TYPE	9\$	
5202	021524	012716	177777		MOV	#-1, (SP)	:: SET THE RUBOUT KEY
5203	021530	005303			DEC	R3	:: BACKUP BY ONE
5204	021532	020327	021712		CMP	R3, #STTYIN	:: STACK EMPTY?
5205	021536	103434			BLO	4\$	:: BR IF YES
5206	021540	111337	021710		MOVB	(R3), 9\$	:: SETUP TO TYPEOUT THE DELETED CHAR.
5207	021544	104401	021710		TYPE	9\$	:: GO TYPE
5208	021550	000746			BR	2\$	:: GO READ ANOTHER CHAR.
5209	021552	005716			TST	(SP)	:: RUBOUT KEY SET?
5210	021554	001406			BEG	7\$	:: BR IF NO
5211	021556	112737	000134	021710	MOVB	#'\, 9\$	:: TYPE A BACK SLASH
5212	021564	104401	021710		TYPE	9\$	
5213	021570	005016			CLR	(SP)	:: CLEAR THE RUBOUT KEY
5214	021572	122713	000025		CMPB	#25, (R3)	:: IS CHARACTER A CTRL U?
5215	021576	001003			BNE	8\$	:: BR IF NO
5216	021600	104401	021722		TYPE	\$CNTLU	:: TYPE A CONTROL "U"
5217	021604	000726			BR	1\$	:: GO START OVER
5218	021606	122713	000022		CMPB	#22, (R3)	:: IS CHARACTER A "r"?
5219	021612	001011			BNE	3\$	:: BRANCH IF NO
5220	021614	105013			CLRB	(R3)	:: CLEAR THE CHARACTER
5221	021616	104401	001213		TYPE	\$CRLF	:: TYPE A "CR" & "LF"
5222	021622	104401	021712		TYPE	\$TTYIN	:: TYPE THE INPUT STRING
5223	021626	000717			BR	2\$	:: GO PICKUP ANOTHER CHARACTER
5224	021630	104401	001212		TYPE	\$QUES	:: TYPE A "?"
5225	021634	000712			BR	1\$	:: CLEAR THE BUFFER AND LOOP
5226	021636	111337	021710		MOVB	(R3), 9\$	:: ECHO THE CHARACTER
5227	021642	104401	021710		TYPE	9\$	
5228	021646	122723	000015		CMPB	#15, (R3)+	:: CHECK FOR RETURN
5229	021652	001305			BNE	2\$	:: LOOP IF NOT RETURN
5230	021654	105063	177777		CLRB	-1(R3)	:: CLEAR RETURN (THE 15)
5231	021660	104401	001214		TYPE	\$LF	:: TYPE A LINE FEED
5232	021664	005726			TST	(SP)+	:: CLEAN RUBOUT KEY FROM THE STACK
5233	021666	012603			MOV	(SP)+, R3	:: RESTORE R3
5234	021670	011646			MOV	(SP), -(SP)	:: ADJUST THE STACK AND PUT ADDRESS OF THE
5235	021672	016666	000004	000002	MOV	4(SP), 2(SP)	:: FIRST ASCII CHARACTER ON IT
5236	021700	012766	021712	000004	MOV	#STTYIN, 4(SP)	
5237	021706	000002			RTI		:: RETURN
5238	021710	000			.BYTE	0	:: STORAGE FOR ASCII CHAR. TO TYPE
5239	021711	000			.BYTE	0	:: TERMINATOR
5240	021712	000010			\$TTYIN:	.BLKB 8	:: RESERVE 8 BYTES FOR TTY INPUT
5241	021722	052536	005015	000	\$CNTLU:	.ASCIZ /tU<15><12>	:: CONTROL "U"
5242	021727	136	006507	000012	\$CNTLG:	.ASCIZ /tG<15><12>	:: CONTROL "G"
5243	021734	005015	053523	020122	\$MSWR:	.ASCIZ <15><12>/SWR = /	
5244	021742	020075	000				
5245	021745	040	047040	053505	\$MNEW:	.ASCIZ / NEW = /	
5246	021752	036440	000040				
5247							
5248							
5249							
					.SETTL	READ AN OCTAL NUMBER FROM THE TTY	

103

5250
 5251
 5252
 5253
 5254
 5255
 5256
 5257
 5258 021756 011646
 5259 021760 016666 000004 000002
 5260 021766 010046
 5261 021770 010146
 5262 021772 010246
 5263 021774 104411
 5264 021776 012600
 5265 022000 005001
 5266 022002 005002
 5267 022004 112046
 5268 022006 001412
 5269 022010 006301
 5270 022012 006102
 5271 022014 006301
 5272 022016 006102
 5273 022020 006301
 5274 022022 006102
 5275 022024 042716 177770
 5276 022030 062601
 5277 022032 000764
 5278 022034 005726
 5279 022036 010166 000012
 5280 022042 010237 022056
 5281 022046 012602
 5282 022050 012601
 5283 022052 012600
 5284 022054 000002
 5285 022056 000000
 5286
 5287
 5288
 5289
 5290
 5291
 5292
 5293
 5294
 5295
 5296
 5297
 5298
 5299
 5300
 5301
 5302
 5303
 5304
 5305

```

*****
;THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
;CHANGE IT TO BINARY.
;CALL:
;      RDOCT          ;: READ AN OCTAL NUMBER
;      RETURN HERE    ;: LOW ORDER BITS ARE ON TOP OF THE STACK
;                      ;: HIGH ORDER BITS ARE IN $HI OCT
$RDOCT: MOV      (SP), -(SP)      ;: PROVIDE SPACE FOR THE
      MOV      4(SP), 2(SP)      ;: INPUT NUMBER
      MOV      R0, -(SP)         ;: PUSH R0 ON STACK
      MOV      R1, -(SP)         ;: PUSH R1 ON STACK
      MOV      R2, -(SP)         ;: PUSH R2 ON STACK
1$:  RDLIN          ;: READ AN ASCII LINE
      MOV      (SP)+, R0         ;: GET ADDRESS OF 1ST CHARACTER
      CLR      R1                ;: CLEAR DATA WORD
      CLR      R2
2$:  MOVB      (R0)+, -(SP)       ;: PICKUP THIS CHARACTER
      BEQ      3$                ;: IF ZERO GET OUT
      ASL      R1                ;: *2
      ROL      R2                ;: *4
      ASL      R1                ;: *8
      ROL      R2
      BIC      #1C7, (SP)        ;: STRIP THE ASCII JUNK
      ADD      (SP)+, R1         ;: ADD IN THIS DIGIT
      BR       2$                ;: LOOP
3$:  TST      (SP)+              ;: CLEAN TERMINATOR FROM STACK
      MOV      R1, 12(SP)        ;: SAVE THE RESULT
      MOV      R2, $HI OCT
      MOV      (SP)+, R2         ;: POP STACK INTO R2
      MOV      (SP)+, R1         ;: POP STACK INTO R1
      MOV      (SP)+, R0         ;: POP STACK INTO R0
      RTI                      ;: RETURN
$HI OCT: .WORD      0            ;: HIGH ORDER BITS GO HERE

.SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
*****
;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
;OCTAL (ASCII) NUMBER AND TYPE IT.
;$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
;CALL:
;      MOV      NUM, -(SP)        ;: NUMBER TO BE TYPED
;      TYPOS          ;: CALL FOR TYPEOUT
;      .BYTE    N            ;: N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
;      .BYTE    M            ;: M=1 OR 0
;                                ;: 1=TYPE LEADING ZEROS
;                                ;: 0=SUPPRESS LEADING ZEROS
;$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
;$TYPOS OR $TYPOC
;CALL:
;      MOV      NUM, -(SP)        ;: NUMBER TO BE TYPED
;      TYPON          ;: CALL FOR TYPEOUT

```

```

5306
5307
5308
5309
5310
5311
5312 022060 017646 000000
5313 022064 116637 000001 022303
5314 022072 112637 022305
5315 022076 062716 000002
5316 022102 000406
5317 022104 112737 000001 022303
5318 022112 112737 000006 022305
5319 022120 112737 000005 022302
5320 022126 010346
5321 022130 010446
5322 022132 010546
5323 022134 113704 022305
5324 022140 005404
5325 022142 062704 000006
5326 022146 110437 022304
5327 022152 113704 022303
5328 022156 016605 000012
5329 022162 005003
5330 022164 006105
5331 022166 000404
5332 022170 006105
5333 022172 006105
5334 022174 006105
5335 022176 010503
5336 022200 006103
5337 022202 105337 022304
5338 022206 100016
5339 022210 042703 177770
5340 022214 001002
5341 022216 005704
5342 022220 001403
5343 022222 005204
5344 022224 052703 000060
5345 022230 052703 000040
5346 022234 110337 022302
5347 022240 104401 022302
5348 022244 105337 022302
5349 022250 003347
5350 022252 002402
5351 022254 005204
5352 022256 000744
5353 022260 012605
5354 022262 012604
5355 022264 012603
5356 022266 016666 000002 000004
5357 022274 012616
5358 022276 000002
5359 022300 000
5360 022301 000
5361 022302 000

; *
; * STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
; * CALL:
; *      MOV      NUM,-(SP)      ; NUMBER TO BE TYPED
; *      TYPOC      ; CALL FOR TYPEOUT

STYPOS: MOV      2(SP),-(SP)      ; PICKUP THE MODE
        MOV      1(SP),SOFILL    ; LOAD ZERO FILL SWITCH
        MOV      (SP)+,SOMODE+1  ; NUMBER OF DIGITS TO TYPE
        ADD      #2,(SP)         ; ADJUST RETURN ADDRESS
        BR       STYPON

STYPOC: MOV      #1,SOFILL        ; SET THE ZERO FILL SWITCH
        MOV      #6,SOMODE+1      ; SET FOR SIX(6) DIGITS
STYPON: MOV      #5,SOCNT         ; SET THE ITERATION COUNT
        MOV      R3,-(SP)         ; SAVE R3
        MOV      R4,-(SP)         ; SAVE R4
        MOV      R5,-(SP)         ; SAVE R5
        MOV      SOMODE+1,R4      ; GET THE NUMBER OF DIGITS TO TYPE
        NEG      R4
        ADD      #6,R4            ; SUBTRACT IT FOR MAX. ALLOWED
        MOV      R4,SOMODE        ; SAVE IT FOR USE
        MOV      SOFILL,R4        ; GET THE ZERO FILL SWITCH
        MOV      12(SP),R5        ; PICKUP THE INPUT NUMBER
        CLR      R3              ; CLEAR THE OUTPUT WORD
        ROL      R5              ; ROTATE MSB INTO "C"
        BR       1$             ; GO DO MSB
        ROL      R5              ; FORM THIS DIGIT
        BR       2$
        ROL      R5
        ROL      R5
        MOV      R5,R3
        ROL      R3              ; GET LSB OF THIS DIGIT
        DECB     SOMODE           ; TYPE THIS DIGIT?
        BPL      7$              ; BR IF NO
        BIC      #177770,R3      ; GET RID OF JUNK
        BNE      4$              ; TEST FOR 0
        TST      R4              ; SUPPRESS THIS 0?
        BEQ      5$              ; BR IF YES
        INC      R4              ; DON'T SUPPRESS ANYMORE 0'S
        BIS      #'0,R3          ; MAKE THIS DIGIT ASCII
        BIS      #' ,R3          ; MAKE ASCII IF NOT ALREADY
        MOV      R3,R5           ; SAVE FOR TYPING
        TYPE     R5              ; GO TYPE THIS DIGIT
        DECB     SOCNT           ; COUNT BY 1
        BGT      2$              ; BR IF MORE TO DO
        BLT      6$              ; BR IF DONE
        INC      R4              ; INSURE LAST DIGIT ISN'T A BLANK
        BR       2$              ; GO DO THE LAST DIGIT
        MOV      (SP)+,R5        ; RESTORE R5
        MOV      (SP)+,R4        ; RESTORE R4
        MOV      (SP)+,R3        ; RESTORE R3
        MOV      2(SP),4(SP)     ; SET THE STACK FOR RETURNING
        MOV      (SP)+,(SP)
        RTI                     ; RETURN
        .BYTE    0               ; STORAGE FOR ASCII DIGIT
        .BYTE    0               ; TERMINATOR FOR TYPE ROUTINE
        .BYTE    0               ; OCTAL DIGIT COUNTER

```

5362 022303 000
5363 022304 000000\$OFILL: .BYTE 0 ;:ZERO FILL SWITCH
\$OMODE: .WORD 0 ;:NUMBER OF DIGITS TO TYPE5364
5365
5366
5367
5368
5369
5370
5371
5372
5373.SBTTL TYPDSS - TYPE DECIMAL, LEADING ZEROES SUPPRESSED
;TYPDSS
;ROUTINE FOR TYPING OUT DECIMAL NUMBERS, LEADING 0'S ARE SUPPRESSED
;THE NUMBER IS LEFT JUSTIFIED. NOTE THE 16 BIT BINARY NUMBER SHOULD
;BE POSITIVE (BIT 15= 0)
;CALL: MOV NUMBER, -(SP) ;PUT BINARY NUMBER ON STACK
;TYPDSS ;GO TYPE DECIMAL5374 022306 016637 000004 022346
5375 022314 012746 022346
5376 022320 004737 022506
5377
5378 022324 004737 022352
5379
5380 022330 016666 000002 000004
5381 022336 011666 000002
5382 022342 005726
5383 022344 000002
5384TYPDES: MOV 4(SP), 1\$;GET THE NUMBER
MOV 1\$, -(SP) ;PUT PTR ON THE STACK
JSR PC, 2\$S0B2D ;GO CONVERT BINARY NO. TO
;ASCII STRING
JSR PC, 2\$S\$SUPRS ;GO TYPE OUT DECIMAL STRING
;SUPRESING LEADING 0'S
MOV 2(SP), 4(SP) ;ADJUST RETURN
MOV (SP), 2(SP) ;ADJUST RETURN ADRES
TST (SP)+ ;POP STACK
RTI ;RETURN

5385 022346 000000 000000

1\$: .WORD 0,0

5386
5387
5388
5389
5390
5391
5392
5393
5394
5395
5396
5397
5398
5399
5400
5401
5402
5403
5404
5405
5406
5407
5408
5409
5410
5411
5412
5413
5414
5415
5416
5417

.SBTTL TYPE NUMERICAL ASCII STRING SUPPRESS LEADING ZEROS

;*****
;THIS ROUTINE IS USED TO TYPE AN ASCII NUMBER SUPPRESSING THE
;LEADING NUMBERS.
;CALL
;* MOV \$NUMADR, -(SP) ;:FIRST ADDRESS OF ASCII STRING
;* JSR PC, 2\$S\$SUPRS5398 022352 010046
5399 022354 016600 000004
5400 022360 105710
5401 022362 001403
5402 022364 122720 000060
5403 022370 001773
5404 022372 005300
5405 022374 010037 022402
5406 022400 104401
5407 022402 000000
5408 022404 012600
5409 022406 012616
5410 022410 000207\$SUPRS: MOV R0, -(SP) ;:SAVE R0
MOV 4(SP), R0 ;:PICKUP THE POINTER
1\$: TSTB (R0) ;:TERMINATEOR?
BEQ 2\$;:BR IF YES
CMPB #'0, (R0)+ ;:IS THIS AN ASCII "0" ?
BEQ 1\$;:BR IF YES
2\$: DEC R0 ;:BACKUP BY "1"
MOV R0, 3\$;:SAVE FOR TYPING
TYPE ;:GO TYPE
3\$: .WORD 0 ;:ASCII POINTER GOES HERE
MOV (SP)+, R0 ;:RESTORE R0
MOV (SP)+, (SP) ;:RESTORE THE STACK
RTS PC ;:RETURN

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

;*****
;SAVE R0-R5
;CALL:
;* SAVREG

```

5418
5419
5420
5421
5422
5423
5424
5425
5426
5427
5428
5429
5430 022412
5431 022412 010046
5432 022414 010146
5433 022416 010246
5434 022420 010346
5435 022422 010446
5436 022424 010546
5437 022426 016646 000022
5438 022432 016646 000022
5439 022436 016646 000022
5440 022442 016646 000022
5441 022446 000002
5442
5443
5444
5445 022450
5446 022450 012666 000022
5447 022454 012666 000022
5448 022460 012666 000022
5449 022464 012666 000022
5450 022470 012605
5451 022472 012604
5452 022474 012603
5453 022476 012602
5454 022500 012601
5455 022502 012600
5456 022504 000002
5457
5458
5459
5460
5461
5462
5463
5464
5465
5466
5467
5468
5469
5470
5471 022506 104413
5472 022510 016602 000002
5473 022514 012700 022666

```

```

; *UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
; *

```

```

; *TOP---(+16)
; * +2---(+18)
; * +4---R5
; * +6---R4
; * +8---R3
; * +10---R2
; * +12---R1
; * +14---R0

```

```

$SAVREG:

```

```

MOV R0, -(SP) ;: PUSH R0 ON STACK
MOV R1, -(SP) ;: PUSH R1 ON STACK
MOV R2, -(SP) ;: PUSH R2 ON STACK
MOV R3, -(SP) ;: PUSH R3 ON STACK
MOV R4, -(SP) ;: PUSH R4 ON STACK
MOV R5, -(SP) ;: PUSH R5 ON STACK
MOV 22(SP), -(SP) ;: SAVE PS OF MAIN FLOW
MOV 22(SP), -(SP) ;: SAVE PC OF MAIN FLOW
MOV 22(SP), -(SP) ;: SAVE PS OF CALL
MOV 22(SP), -(SP) ;: SAVE PC OF CALL
RTI

```

```

; *RESTORE RO-R5

```

```

; *CALL:

```

```

; * RESREG

```

```

$RESREG:

```

```

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

```

```

.SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

```

```

; *****
; *THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
; *DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
; *POSITIVE.

```

```

; *CALL

```

```

; * MOV #PNTR, -(SP) ;: POINTER TO LOW WORD OF BINARY NUMBER
; * JSR PC, @SDB2D ;: THE FIRST ADDRESS OF ASCII
; * RETURN ;: IS ON THE STACK

```

```

$SDB2D: SAVREG

```

```

MOV 2(SP), R2 ;: SAVE REGISTERS
MOV #SDECVL, R0 ;: PICKUP THE DATA POINTER
;: GET ADDRESS OF "SDECVL" STRING

```

```

5474 022520 010066 000002      MOV      R0,2(SP)      ;;PUT ADDRESS OF ASCII STRING ON STACK
5475 022524 012201      MOV      (R2)+,R1      ;;PICKUP THE BINARY NUMBER
5476 022526 012202      MOV      (R2)+,R2
5477 022530 012737 000012 022604  MOV      #10,R4      ;;SET UP TO DO 10 CONVERSIONS
5478 022536 012704 022616      MOV      $STNPWR,R4      ;;ADDRESS OF TEN POWER
5479 022542 012705 022620      MOV      $STNPWR+2,R5
5480 022546 005003      1$:  CLR      R3      ;;CLEAR PARTIAL
5481 022550 161401      2$:  SUB      (R4),R1      ;;SUBTRACT TEN POWER
5482 022552 005602      SBC      R2
5483 022554 161502      SUB      (R5),R2
5484 022556 002402      BLT      3$      ;;BR IF TEN POWER TOO LARGE
5485 022560 005203      INC      R3      ;;ADD 1 TO PARTIAL
5486 022562 000772      BR      2$      ;;LOOP
5487 022564 062401      3$:  ADD      (R4)+,R1      ;;RESTORE SUBTRACTED VALUE
5488 022566 005502      ADC      R2
5489 022570 062402      ADD      (R4)+,R2
5490 022572 022525      CMP      (R5)+,(R5)+      ;;MOVE TO NEXT TEN POWER
5491 022574 052703 000060      BIS      #0,R3      ;;CHANGE PARTIAL TO ASCII
5492 022600 110320      MOV      R3,(R0)+      ;;SAVE IT
5493 022602 005327      DEC      (PC)+      ;;DONE?
5494 022604 000000      4$:  .WORD      0
5495 022606 001357      BNE      1$      ;;BR IF NO
5496 022610 105020      CLRB     (R0)+      ;;TERMINATOR
5497 022612 104414      RESREG
5498 022614 000207      RTS      PC      ;;RETURN
5499 022616 145000      $TNPWR: 145000      ;;1.0E09
5500 022620 035632      35632
5501 022622 160400      160400      ;;1.0E08
5502 022624 002765      2765
5503 022626 113200      113200      ;;1.0E07
5504 022630 000230      230
5505 022632 041100      041100      ;;1.0E06
5506 022634 000017      17
5507 022636 103240      103240      ;;1.0E05
5508 022640 000001      1
5509 022642 023420      23420      ;;1.0E04
5510 022644 000000      0
5511 022646 001750      1750      ;;1.0E03
5512 022650 000000      0
5513 022652 000144      144      ;;1.0E02
5514 022654 000000      0
5515 022656 000012      12      ;;1.0E01
5516 022660 000000      0
5517 022662 000001      1      ;;1.0E00
5518 022664 000000      0
5519 022666 000014      $DECVL: .BLKB 12.      ;;RESERVE STORAGE FOR ASCII STRING
5520
5521
5522
5523      .SBTTL  TRAP DECODER
5524
5525      ;*****
5526      ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
5527      ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
5528      ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
5529      ;*GO TO THAT ROUTINE.

```

```

5530
5531 022702 010046 $TRAP: MOV R0, -(SP) ;; SAVE R0
5532 022704 016600 000002 MOV 2(SP), R0 ;; GET TRAP ADDRESS
5533 022710 005740 TST -(R0) ;; BACKUP BY 2
5534 022712 111000 MOVB (R0), R0 ;; GET RIGHT BYTE OF TRAP
5535 022714 006300 ASL R0 ;; POSITION FOR INDEXING
5536 022716 016000 022736 MOV $TRAPD(R0), R0 ;; INDEX TO TABLE
5537 022722 000200 RTS R0 ;; GO TO ROUTINE
5538
5539
5540 ;; THIS IS USE TO HANDLE THE "GETPRI" MACRO
5541
5542 022724 011646 $TRAP2: MOV (SP), -(SP) ;; MOVE THE PC DOWN
5543 022726 016666 000004 000002 MOV 4(SP), 2(SP) ;; MOVE THE PSW DOWN
5544 022734 000002 RTI ;; RESTORE THE PSW
5545
5546 .SBTTL TRAP TABLE
5547
5548 ; *THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
5549 ; BY THE "TRAP" INSTRUCTION.
5550
5551 ; ROUTINE
5552 ; -----
5553 022736 022724 $TRPAD: .WORD $TRAP2
5554 022740 020260 $TYPE ;; CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
5555 022742 022104 $TYPOC ;; CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
5556 022744 022060 $TYPOS ;; CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
5557 022746 022120 $TYPON ;; CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
5558 022750 020034 $TYPDS ;; CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
5559
5560 022752 021154 $GTSWR ;; CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING
5561
5562 022754 021064 $CKSWR ;; CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
5563 022756 021366 $RDCHR ;; CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
5564 022760 021456 $RDLIN ;; CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE
5565 022762 021756 $RDOCT ;; CALL=RDOCT TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
5566 022764 022412 $SAVREG ;; CALL=SAVREG TRAP+13(104413) SAVE R0-R5 ROUTINE
5567 022766 022450 $RESREG ;; CALL=RESREG TRAP+14(104414) RESTORE R0-R5 ROUTINE
5568
5569 022770 016602 CN.RST ;; CALL=CON.RESET TRAP+15(104415) CONTROL RESET ROUTINE
5570
5571 022772 016470 DR.RST ;; CALL=DRV.RESET TRAP+16(104416) DRIVE RESET ROUTINE
5572
5573 022774 016044 MSGE ;; CALL=MESSAGE TRAP+17(104417) MESSAGE HANDLER
5574
5575 022776 016106 TY.MSG ;; CALL=TYPMMSG TRAP+20(104420) MESSAGE TYPEOUT ROUTINE
5576
5577 023000 016620 CN.RDY ;; CALL=CON.RDY TRAP+21(104421) WAIT FOR CONTROL READY
5578
5579 023002 016716 TSTAWS ;; CALL=TST.RWS TRAP+22(104422) TEST R/W/S RDY SET
5580
5581 023004 016514 RES.DO ;; CALL=RESDON TRAP+23(104423) DRIVE RESET DONE?
5582
5583 023006 022306 TYPDES ;; CALL=TYPDSS TRAP+24(104424) TYPE DECIMAL, SUPRES LOG 0'S
5584
5585

```

.SBTTL POWER DOWN AND UP ROUTINES

```

5586
5587
5588
5589
5590 023010 012737 023154 000024
5591 023016 012737 000340 000026
5592 023024 010046
5593 023026 010146
5594 023030 010246
5595 023032 010346
5596 023034 010446
5597 023036 010546
5598 023040 017746 156074
5599 023044 010637 023160
5600 023050 012737 023062 000024
5601 023056 000000
5602 023060 000776
5603
5604
5605
5606 023062 012737 023154 000024
5607 023070 013706 023160
5608 023074 005037 023160
5609 023100 005337 023160
5610 023104 001375
5611 023106 012677 156026
5612 023112 012605
5613 023114 012604
5614 023116 012603
5615 023120 012602
5616 023122 012601
5617 023124 012600
5618 023126 012737 023010 000024
5619 023134 012737 000340 000026
5620 023142 104401
5621 023144 023162
5622 023146 012716
5623 023150 004244
5624 023152 000002
5625 023154 000000
5626 023156 000776
5627 023160 000000
5628 023162 005015 047520 042527
5629 023170 000122
5630
5631
5632

*****
: POWER DOWN ROUTINE
$PWRDN: MOV $SILLUP, 2*PWRVEC ;; SET FOR FAST UP
        MOV #340, 2*PWRVEC+2 ;; PRI0:7
        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 2SWR, -(SP) ;; PUSH 2SWR ON STACK
        MOV SP, $SAVR6 ;; SAVE SP
        MOV $PWRUP, 2*PWRVEC ;; SET UP VECTOR
        HALT
        BR -2 ;; HANG UP

*****
: POWER UP ROUTINE
$PWRUP: MOV $SILLUP, 2*PWRVEC ;; SET FOR FAST DOWN
        MOV $SAVR6, SP ;; GET SP
        CLR $SAVR6 ;; WAIT LOOP FOR THE TTY
        IS: INC $SAVR6 ;; WAIT FOR THE INC
        BNE IS ;; OF WORD
        MOV (SP)+, 2SWR ;; POP STACK INTO 2SWR
        MOV (SP)+, R5 ;; POP STACK INTO R5
        MOV (SP)+, R4 ;; POP STACK INTO R4
        MOV (SP)+, R3 ;; POP STACK INTO R3
        MOV (SP)+, R2 ;; POP STACK INTO R2
        MOV (SP)+, R1 ;; POP STACK INTO R1
        MOV (SP)+, R0 ;; POP STACK INTO R0
        MOV $PWRDN, 2*PWRVEC ;; SET UP THE POWER DOWN VECTOR
        MOV #340, 2*PWRVEC+2 ;; PRI0:7
        TYPE ;; REPORT THE POWER FAILURE
        $PWRMG: .WORD $POWER ;; POWER FAIL MESSAGE POINTER
        MOV (PC)+, (SP) ;; RESTART AT PWRFL
        $PWRAD: .WORD PWRFL ;; RESTART ADDRESS
        RTI
        $SILLUP: HALT ;; THE POWER UP SEQUENCE WAS STARTED
        BR -2 ;; BEFORE THE POWER DOWN WAS COMPLETE
        $SAVR6: 0 ;; PUT THE SP HERE
        $POWER: .ASCIZ <15><12>"POWER"
                .EVEN

```

.SBTTL FUNCTION SELECTION PROGRAM

```

;THIS IS THE FUNCTION SELECTION PROGRAM.
;ON ENTERING THIS SUB-PROGRAM THE FIRST QUESTION ASKED IS
;FUNCTION? IN REPLY TYPE IN ONE OF THE FOLLOWING COMMANDS
;COMMANDS: CR - CONTROL RESET
;           DR - DRIVE RESET
;           SK - SEEK
;           WR - WRITE
;           RD - READ
;           WC - WRITE CHECK
;           RC - READ CHECK
;TERMINATE EVERY COMMAND WITH <CR>. FURTHER QUESTIONS (RKBA? RKDA?
;WORDS? ) WILL BE ASKED. TYPE IN THE BUS ADDRESS (OCTAL), DISK ADDRESS
;(OCTAL), AND NUMBER OF WORDS TO BE TRANSFERRED (OCTAL). IF A NON-EXISTENT
;CYLINDER OR A NON-EXISTENT SECTOR IS TYPED IN, THE QUESTION (RKDA?) WILL
;BE ASKED AGAIN.

;IN CASE OF SEEK FUNCTION, SEEK IS DONE BETWEEN A SET OF CYLINDERS GIVEN
;BY THE USER (CYL1? , CYL2?). IN REPLY TO (CYL1? , CYL2?) TYPE IN THE OCTAL
;CYLINDER NUMBERS BETWEEN WHICH THE SEEK IS TO BE DONE. SET SWITCH
;REGISTER BITS <15-13> TO THE DRIVE # ON WHICH SEEK IS TO BE DONE.

;IN CASE OF A WRITE FUNCTION IF SW 0 IS SET TO 1 THE PROGRAM WILL ASK
;THE USER FOR THE PATTERN TO BE WRITTEN:
;PATRN?125252<CR>
;THE USER SHOULD TYPE IN THE (OCTAL) PATTERN HE WANTS TO WRITE.
;NOTE THAT THE NUMBER OF WORDS TRANSFERRED SHOULD BE WITHIN THE
;BOUNDS OF THE SYSTEM.

;IN CASE OF A WRITE CHECK FUNCTION IF SW 0 IS SET TO 1 THE PROGRAM
;WILL ASK THE USER FOR THE PATTERN TO BE WRITE CHECKED:
;PATRN?125252<CR>
;THE USER SHOULD TYPE IN THE (OCTAL) PATTERN TO BE WRITE CHECKED.

;LOCATIONS "IOBUF0" ONWARDS ARE RESERVED FOR DATA BUFFERS. YOU CAN USE
;THESE LOCATIONS FOR DOING DATA TRANSFERS IN THIS PROGRAM (OR YOU CAN
;USE ANY OTHER BUFFER FOR DATA TRANSFER AS LONG AS IT DOES NOT OVERLAY
;THE PROGRAM).

;THE SAME FUNCTION IS PERFORMED AGAIN AND AGAIN, THUS PROVIDING
;A VERY GOOD SCOPE LOOP. IF YOU WANT TO GIVE A NEW FUNCTION PUT SW 3 UP.
;THE QUESTION (FUNCTION?) WILL BE ASKED AND YOU CAN START ALL OVER AGAIN.
;IF ON EXECUTING A FUNCTION AN ERROR OCCURS, IT IS REPOTRED . WITH
;RELEVANT REGISTER CONTENTS GIVEN.

```

```

;R2 CONTAINS RKCS CONTENTS
;R3 CONTAINS RKDA CONTENTS
;R4,R5 CONTAIN THE CYLINDER #S BETWEEN
;WHICH SEEK IS TO BE DONE.

```

FUNBEG:

```

TYPE 65$ ;TYPE ASCIZ STRING
BR 64$ ;GET OVER THE ASCIZ
;65$: .ASCIZ <15><12>/FUNCTION? /
64$: ROLIN

```

```

5633
5634
5635
5636
5637
5638
5639
5640
5641
5642
5643
5644
5645
5646
5647
5648
5649
5650
5651
5652
5653
5654
5655
5656
5657
5658
5659
5660
5661
5662
5663
5664
5665
5666
5667
5668
5669
5670
5671
5672
5673
5674
5675
5676
5677
5678
5679
5680
5681
5682
5683 023172
5684 023172 104401 023200
5685 023176 000407
5686
5687 023216
5688 023216 104411

```

5689	023220	012600		MOV	(SP)+,R0	
5690	023222	112001		MOV	(R0)+,R1	
5691	023224	120127	000127	CMP	R1,#'W	
5692	023230	001026		BNE	2\$	
5693	023232	121027	000122	CMP	(R0),#'R	
5694	023236	001010		BNE	1\$	
5695	023240	004737	024064	JSR	PC,CHKSWD	;CHECK SW 0 SET?
5696	023244	012702	004003	MOV	#4003,R2	
5697	023250	000536		BR	NXTDA	
5698						
5699	023252	012702	000003	9\$: MOV	#3,R2	
5700	023256	000521		BR	NXTBA	
5701	023260	121027	000103	1\$: CMP	(R0),#'C	
5702	023264	001342		BNE	FUNBEG	
5703	023266	004737	024064	JSR	PC,CHKSWD	;CHECK SW 0 SET?
5704	023272	012702	004007	MOV	#4007,R2	
5705	023276	000523		BR	NXTDA	
5706						
5707	023300	012702	000007	MOV	#7,R2	
5708	023304	000506		BR	NXTBA	
5709						
5710	023306	120127	000122	2\$: CMP	R1,#'R	
5711	023312	001014		BNE	3\$	
5712	023314	121027	000104	CMP	(R0),#'D	
5713	023320	001003		BNE	8\$	
5714	023322	012702	000005	MOV	#5,R2	
5715	023326	000475		BR	NXTBA	
5716	023330	121027	000103	8\$: CMP	(R0),#'C	
5717	023334	001316		BNE	FUNBEG	
5718	023336	012702	000013	MOV	#13,R2	
5719	023342	000501		BR	NXTDA	
5720						
5721	023344	120127	000104	3\$: CMP	R1,#'D	
5722	023350	001006		BNE	4\$	
5723	023352	121027	000122	CMP	(R0),#'R	
5724	023356	001305		BNE	FUNBEG	
5725	023360	012702	000015	MOV	#15,R2	
5726	023364	000470		BR	NXTDA	
5727						
5728	023366	120127	000103	4\$: CMP	R1,#'C	
5729	023372	001006		BNE	5\$	
5730	023374	121027	000122	CMP	(R0),#'R	
5731	023400	001274		BNE	FUNBEG	
5732	023402	012702	000001	MOV	#1,R2	
5733	023406	000533		BR	EXEC	
5734						
5735	023410	120127	000123	5\$: CMP	R1,#'S	
5736	023414	001256		BNE	FUNBEG	
5737	023416	121027	000113	CMP	(R0),#'K	
5738	023422	001263		BNE	FUNBEG	
5739	023424	012702	000011	MOV	#11,R2	
5740						
5741	023430			6\$: TYPE	67\$	::TYPE ASCII STRING
5742	023430	104401	023436	BR	66\$	::GET OVER THE ASCII
5743	023434	000404				
5744				::67\$: .ASCII	/CYL1? /	

```

5745 023446
5746 023446 004737 024032
5747 023452 000766
5748 023454 010004
5749
5750 023456
5751 023456 104401 023464
5752 023462 000404
5753
5754 023474
5755 023474 004737 024032
5756 023500 000766
5757 023502 010005
5758 023504 017700 155430
5759 023510 042700 017777
5760 023514 050004
5761 023516 050005
5762 023520 000466
5763
5764
5765 023522
5766 023522 104401 023530
5767 023526 000404
5768
5769 023540
5770 023540 104412
5771 023542 012637 001476
5772
5773 023546
5774 023546 104401 023554
5775 023552 000404
5776
5777 023564
5778 023564 104412
5779 023566 012600
5780 023570 010001
5781 023572 006201
5782 023574 006201
5783 023576 006201
5784 023600 006201
5785 023602 006201
5786 023604 042701 177400
5787 023610 020127 000312
5788 023614 003354
5789 023616 010001
5790 023620 042701 177760
5791 023624 020127 000013
5792 023630 003346
5793 023632 010003
5794 023634 022702 000015
5795 023640 001416
5796
5797
5798 023642
5799 023642 104401 023650
5900 023646 000405

```

```

66$: JSR PC, INPT
BR 6$
MOV RO, R4

7$: TYPE 69$ ;;TYPE ASCIZ STRING
BR 68$ ;;GET OVER THE ASCIZ
;;69$: .ASCIZ /CYL2? /
68$: JSR PC, INPT
BR 7$
MOV RO, R5
MOV 254R, RO ;GET DRIVE # FROM SW REG<15-13>
BIC #17777, RO ;CLR UNWANTED BITS
BIS RO, R4 ;SET DRIVE # IN DSK ADRES
BIS RO, R5
BR EXEC

NXTBA: TYPE 65$ ;;TYPE ASCIZ STRING
BR 64$ ;;GET OVER THE ASCIZ
;;65$: .ASCIZ /RKBA? /
64$: RDOCT
MOV (SP)+, INDX2

NXTDA: TYPE 65$ ;;TYPE ASCIZ STRING
BR 64$ ;;GET OVER THE ASCIZ
;;65$: .ASCIZ /RKDA? /
64$: RDOCT
MOV (SP)+, RO
MOV RO, R1
ASR R1
ASR R1
ASR R1
ASR R1
ASR R1
BIC #177400, R1
CMP R1, #312
BGT NXTDA
MOV RO, R1
BIC #177760, R1
CMP R1, #13
BGT NXTDA
MOV RO, R3
CMP #15, R2
BEQ EXEC

NXTWC: TYPE 65$ ;;TYPE ASCIZ STRING
BR 64$ ;;GET OVER THE ASCIZ

```

```

5801      655: .ASCIZ  /#WORDS? /
5802      645:
5803      RDOCT
5804      NEG      (SP)
5805      MOV      (SP)+,INDX4
5806      BR       EXEC
5807
5808      023662 104412
5809      023662 005416
5810      023664 012637 001502
5811      023672 000401
5812
5813      023674 104415
5814      023676 022702 000011
5815      023702 001005
5816      023704 020403
5817      023706 001402
5818      023710 010403
5819      023712 000401
5820      023714 010503
5821      023716 013777 001476 155536
5822      023724 010377 155534
5823      023730 013777 001502 155522
5824      023736 010277 155514
5825      023742 105777 155510
5826      023746 100375
5827      023750 022702 000001
5828      023754 001401
5829      023756 104423
5830
5831      023760 032777 140000 155470
5832      023766 001006
5833      023770 032777 000010 155142
5834      023776 001737
5835      024000 000137 023172
5836
5837      024004 012737 023674 001110
5838      024012 012737 023674 001106
5839      024020 004737 016162
5840      024024 104030
5841      024026 104415
5842      024030 000757
5843
5844      024032 104412
5845      024034 012600
5846      024036 020027 000312
5847      024042 003007
5848      024044 006300
5849      024046 006300
5850      024050 006300
5851      024052 006300
5852      024054 006300
5853      024056 062716 000002
5854      024062 000207
5855
5856      024064 032777 000001 155046
5857      024072 001416
5858
5859      024074 104401 024102
5860      024100 000404

```

655: .ASCIZ /#WORDS? /
 645:
 RDOCT
 NEG (SP)
 MOV (SP)+,INDX4
 BR EXEC

 EXEC1: CON.RESET ;CLR ERROR CONTROL RESET
 EXEC: CMP #11,R2 ;SEEK FUNCTION?
 BNE 25 ;NO
 CMP R4,R3 ;IF SEEK, INSERT THE RIGHT
 BEQ 35 ;CYLINDER ADDRESS
 MOV R4,R3
 BR 25
 35: MOV R5,R3
 25: MOV INDX2,ARKBA
 MOV F,ARKDA
 MOV INDX4,ARKWC
 MOV R2,ARKCS
 15: TSTB ARKCS ;WAIT FOR CNTROL RDY
 BPL 15
 CMP #1,R2 ;IF IT'S CON RESET FUNCTION
 BEQ 45 ;DONT WAIT FOR R/W/S RDY
 RESDON ;R/W/S RDY?
 45: BIT #140000,ARKCS ;ERROR?
 BNE FUNERR ;YES
 CHSW: BIT #SW3,ASWR ;TERMINATE THIS FUNCTION OR REPEAT?
 BEQ EXEC ;REPEAT
 JMP FUNBEG ;TERMINATE

 FUNERR: MOV #EXEC1,SLPERR ;SET UP FOR LUPING
 MOV #EXEC1,SLPADR
 JSR PC,GT4RG
 ERROR 30 ;REPORT ERROR
 CON.RESET ;CLR ERROR
 BR CHSW

 INPT: MOV RDOCT
 MOV (SP)+,R0
 CMP R0,#312
 BGT 15
 ASL R0
 ASL R0
 ASL R0
 ASL R0
 ASL R0
 ASL R0
 ADD #2,(SP)
 15: RTS PC

 CHKSWO: BIT #SWO,ASWR ;WRITE A PATTERN GIVEN BY USER?
 BEQ 15 ;NO
 TYPE 655 ;YES, ASK FOR PATTERN
 BR 645 ;TYPE ASCIZ STRING
 ;GET OVER THE ASCIZ

```

5857      ;:655: .ASCIZ /PATRN?/
5858      645:
5859      RDOCT
5860      MOV      (SP)+,IOBUF1      ;SAVE THE PATTERN
5861      MOV      #IOBUF1,INDX2
5862      RTS      PC
5863      15:      ADD      #6,(SP)      ;SKIP NXT 2 INST ON RETURN
5864      RTS      PC
5865
5866      FCHECK:  DRV.RESET
5867      CON.RESET
5868      MOV      DRIVAD,DRHOLD      ;SAVE DRIVE ADDR
5869      BIT      #20000,DRIVAD      ;SEE IF ODD
5870      BEQ      15
5871      BIC      #20000,DRIVAD      ;MAKE EVEN
5872      BR       25
5873      15:      BIS      #20000,DRIVAD      ;MAKE ODD
5874      25:      MOV      DRIVAD,ARKDA      ;DRIVE ADDR
5875      MOV      #11,ARKCS      ;DRIVE SEEK
5876      CON.RDY
5877      MOV      DRHOLD,ARKDA      ;OTHER DRIVE
5878      CON.RDY
5879      BIT      #100,ARKDS      ;HEADS IN MOTION?
5880      BNE      35
5881      TST      (R5)+
5882      35:      MOV      DRHOLD,DRIVAD      ;NO 50 RK-05J
5883      DRV.RESET      ;YES RK-05F
5884      RTS      R5      ;RESTORE ADDR
5885      SIZEF:  CLR      DRIVAD      ;START AT DR0
5886      MOV      #DRIV0,RO      ;TABLE OF AVAIL DRIVES
5887      45:      TSTB      (RO)      ;THIS DRIVE HERE?
5888      BEQ      25
5889      TSTB      2(RO)
5890      BEQ      25
5891      JSR      R5,FCHECK      ;NO
5892      BR       25
5893      BIS      #2,(RO)      ;SEE IF F MODEL
5894      BIS      #2,2(RO)      ;J MODEL
5895      25:      TST      (RO)+      ;SET SIGN FOR F
5896      TST      (RO)+      ;BOTH DRIVES
5897      ADD      #40000,DRIVAD      ;NEXT PAIR OF DRIVES
5898      CMP      #DRIV7+1,RO      ;NEXT ACTUL ADDR
5899      BGT      45
5900      RTS      PC      ;CHECKED ALL?
5901      ;NOT YET
5902
5903      ;ERROR MESSAGES
5904
5905      EM1:      .ASCIZ /CNTRL RDY DIDN'T SET AFTR SEEK/
5906
5907
5908
5909
5910
5911
5912      EM2:      .ASCIZ /SIN ON SEEK/

```

5913	024400	020116	042523	045505	
5914	024406	000			
5915					
5916	024407	104	042522	047440	EM3: .ASCIZ /DRE ON SEEK/
5917	024414	020116	042523	045505	
5918	024422	000			
5919					
5920	024423	047	051105	023522	EM4: .ASCIZ /'ERR' ON SEEK/
5921	024430	047440	020116	042523	
5922	024436	045505	000		
5923					
5924	024441	104	052522	047440	EM5: .ASCIZ /DRU ON SEEK, PUT DRV ON 'LOAD' & 'RUN'.
5925	024446	020116	042523	045505	
5926	024454	020054	052520	020124	
5927	024462	051104	020126	047117	
5928	024470	023440	047514	042101	
5929	024476	020047	020046	051047	
5930	024504	047125	000047		
5931					
5932	024510	027522	027527	020123	EM6: .ASCIZ "R/W/S RDY NOT SET AFTER SEEK"
5933	024516	042122	020131	047516	
5934	024524	020124	042523	020124	
5935	024532	043101	042524	020122	
5936	024540	042523	045505	000	
5937					
5938	024545	123	047111	047440	EM7: .ASCIZ /SIN ON WRT FMT/
5939	024552	020116	051127	020124	
5940	024560	046506	000124		
5941					
5942	024564	042447	051122	020047	EM10: .ASCIZ /'ERR' ON DOING WRITE FMT/
5943	024572	047117	042040	044517	
5944	024600	043516	053440	044522	
5945	024606	042524	043040	052115	
5946	024614	000			
5947	024615	123	047111	047440	EM11: .ASCIZ "SIN ON RD/FMT"
5948	024622	020116	042122	043057	
5949	024630	052115	000		
5950	024633	047	051105	023522	EM12: .ASCIZ /'ERR' ON READ FMT/
5951	024640	047440	020116	042522	
5952	024646	042101	043040	052115	
5953	024654	000			
5954	024655	127	047522	043516	EM13: .ASCIZ /WRONG HDRS FROM 'SEC#'.
5955	024662	044040	051104	020123	
5956	024670	051106	046517	023440	
5957	024676	042523	021503	000047	
5958					
5959	024704	051105	051117	047440	EM14: .ASCIZ /ERROR ON IMPLIED SEEK FROM 'CYLA' TO 'CYLB'
5960	024712	020116	046511	046120	
5961	024720	042511	020104	042523	
5962	024726	045505	043040	047522	
5963	024734	020115	041447	046131	
5964	024742	023501	052040	020117	
5965	024750	041447	046131	023502	
5966	024756	000			
5967					
5968	024757	122	040505	020104	MS15: .ASCIZ /READ WRONG HDRS FROM 'CYLB' ABOVE/

MO-11-DZRKL-E, RK11-RK05 DYNAMIC TEST
DZRKLE.P11 26-APR-77 12:27

MACY11 30(1046) 14-JUL-77 08:03 PAGE 112
FUNCTION SELECTION PROGRAM

5969	024764	051127	047117	020107	
5970	024772	042110	051522	043040	
5971	025000	047522	020115	041447	
5972	025006	046131	023502	040440	
5973	025014	047502	042526	000	
5974					
5975	025021	122	040505	020104	EM16: .ASCIZ /READ WRONG, 1ST WRD FROM SEC 0, 'CYLB' (ON IMPLIED SEEK FROM 'CYLA')/
5976	025026	051127	047117	026107	
5977	025034	030440	052123	053440	
5978	025042	042122	043040	047522	
5979	025050	020115	042523	020103	
5980	025056	026060	023440	054503	
5981	025064	041114	020047	047450	
5982	025072	020116	046511	046120	
5983	025100	042511	020104	042523	
5984	025106	045505	043040	047522	
5985	025114	020115	041447	046131	
5986	025122	023501	000051		
5987					
5988	025126	042522	042101	030440	MS17: .ASCIZ /READ 1ST WRD FROM SEC 1, 'CYLB' ABOVE/
5989	025134	052123	053440	042122	
5990	025142	043040	047522	020115	
5991	025150	042523	020103	026061	
5992	025156	023440	054503	041114	
5993	025164	020047	041101	053117	
5994	025172	000105			
5995					
5996	025174	042522	042101	053440	EM20: .ASCIZ /READ WRONG HDRS ON IMPLIED SEEK FROM 'CYLA' TO 'CYLB' /
5997	025202	047522	043516	044040	
5998	025210	051104	020123	047117	
5999	025216	044440	050115	044514	
6000	025224	042105	051440	042505	
6001	025232	020113	051106	046517	
6002	025240	023440	054503	040514	
6003	025246	020047	047524	023440	
6004	025254	054503	041114	000047	
6005					
6006	025262	051105	051117	047440	EM21: .ASCIZ /EROR ON DOING WRITE ON DISK/
6007	025270	020116	047504	047111	
6008	025276	020107	051127	052111	
6009	025304	020105	047117	042040	
6010	025312	051511	000113		
6011					
6012	025316	044523	020116	047117	EM22: .ASCIZ /SIN ON DOING WRITE/
6013	025324	042040	044517	043516	
6014	025332	053440	044522	042524	
6015	025340	000			
6016					
6017	025341	110	020105	047117	EM23: .ASCIZ /HE ON DOING READ/
6018	025346	042040	044517	043516	
6019	025354	051040	040505	000104	
6020					
6021	025362	051503	020105	047117	EM24: .ASCIZ /CSE ON READ/
6022	025370	051040	040505	000104	
6023					
6024	025376	040504	040524	042440	EM25: .ASCIZ /DATA EROR ON READ FROM DISK ADRES/

6025	025404	047522	020122	047117	
6026	025412	051040	040505	020104	
6027	025420	051106	046517	042040	
6028	025426	051511	020113	042101	
6029	025434	042522	000123		
6030					
6031	025440	042510	047440	020116	EM26: .ASCIZ /HE ON WRT CHK/
6032	025446	051127	020124	044103	
6033	025454	000113			
6034					
6035	025456	051127	020124	044103	EM27: .ASCIZ /WRT CHK EROR/
6036	025464	020113	051105	051117	
6037	025472	000			
6038					
6039	025473	105	047522	000122	EM30: .ASCIZ /EROR/
6040					
6041					:DATA HEADERS
6042					
6043					
6044	025500	020040	041520	020040	DH1: .ASCIZ / PC RKCS RKER RKDS . RKDA/
6045	025506	020040	051040	041513	
6046	025514	020123	020040	051040	
6047	025522	042513	020122	020040	
6048	025530	051040	042113	020123	
6049	025536	020040	051040	042113	
6050	025544	000101			
6051					
6052	025546	042523	045505	021440	DH6: .ASCIZ /SEEK # SEEKING ERRORS/
6053	025554	020040	042523	045505	
6054	025562	047111	020107	042440	
6055	025570	051122	051117	000123	
6056					
6057	025576	020040	041520	020040	DH7: .ASCIZ / PC RKCS RKER RKDS RKDA CYL/
6058	025604	020040	051040	041513	
6059	025612	020123	020040	051040	
6060	025620	042513	020122	020040	
6061	025626	051040	042113	020123	
6062	025634	020040	051040	042113	
6063	025642	020101	020040	041440	
6064	025650	046131	000		
6065					
6066	025653	040	050040	020103	DH11: .ASCIZI/ PC RKCS RKER RKDS RKDA: DR CYL SUR SEC/
6067	025660	020040	051040	041513	
6068	025666	020123	020040	051040	
6069	025674	042513	020122	020040	
6070	025702	051040	042113	020123	
6071	025710	020040	045522	040504	
6072	025716	020072	051104	020040	
6073	025724	054503	020114	020040	
6074	025732	020040	052523	020122	
6075	025740	020040	020040	042523	
6076	025746	000103			
6077	025750	042523	021503	020040	DH13: .ASCIZ /SEC# HDR RCVD/
6078	025756	042110	020122	041522	
6079	025764	042126	000		
6080					

6081	025767	040	050040	020103	DM14:	.ASCIZ	/	PC	CYLA	CYLB	RKER	RKDS	TRY#	/
6082	025774	020040	020040	054503										
6083	026002	040514	020040	020040										
6084	026010	054503	041114	020040										
6085	026016	020040	051040	042513										
6086	026024	020122	020040	045522										
6087	026032	051504	020040	020040										
6088	026040	052040	054522	000043										
6089														
6090	026046	020040	041520	020040	DM16:	.ASCIZ	/	PC	CYLA	CYLB	EXPCT	RECVD	TRY#	/
6091	026054	020040	041440	046131										
6092	026062	020101	020040	054503										
6093	026070	041114	020040	020040										
6094	026076	054105	041520	020124										
6095	026104	020040	042522	053103										
6096	026112	020104	020040	051124										
6097	026120	021531	000											
6098														
6099	026123	040	050040	020103	DM17:	.ASCIZ	/	PC	CYLB	EXPCT	RECVD	/		
6100	026130	020040	020040	054503										
6101	026136	041114	020040	042440										
6102	026144	050130	052103	020040										
6103	026152	051040	041505	042126										
6104	026160	000												
6105														
6106	026161	040	050040	020103	DM24:	.ASCIZ	/	PC	TRY#	RKCS	RKER	RKDS	RKDA: DR	CYL SUR SEC/
6107	026166	020040	020040	051124										
6108	026174	021531	020040	051040										
6109	026202	041513	020123	020040										
6110	026210	051040	042513	020122										
6111	026216	020040	051040	042113										
6112	026224	020123	051040	042113										
6113	026232	035101	042040	020122										
6114	026240	041440	046131	020040										
6115	026246	020040	051440	051125										
6116	026254	020040	020040	020040										
6117	026262	042523	000103											
6118														
6119	026266	047527	042122	020043	DM25:	.ASCIZ	/	WORD#	EXPCT	RECVD	CYL	SUR	SEC/	
6120	026274	042440	050130	052103										
6121	026302	020040	051040	041505										
6122	026310	042126	020040	041440										
6123	026316	046131	020040	052523										
6124	026324	020122	051440	041505										
6125	026332	000												
6126														
6127														
6128														
6129														
6130														
6131		026334												
6132														
6133	026334	001116	001162	001164	DT1:	.WORD		\$ERRPC,\$REG0,\$REG1,\$REG2,\$REG3,0						
6134	026342	001166	001170	000000										
6135														
6136	026350	001116	001162	001164	DT7:	.WORD		\$ERRPC,\$REG0,\$REG1,\$REG2,\$REG3,\$REG4,0						

:ERROR DATA POINTERS

.EVEN

```

6137 026356 001166 001170 001172
6138 026364 000000
6139
6140 026366 001116 001162 001164 DT11: .WORD $ERRPC,$REG0,$REG1,$REG2,$REG4,$REG5,$REG6,$REG7,C
6141 026374 001166 001172 001174
6142 026402 001176 001200 000000
6143
6144 026410 001116 001162 001164 DT17: .WORD $ERRPC,$REG0,$REG1,$REG2,0
6145 026416 001166 000000
6146
6147 026422 001116 001202 001162 DT24: .WORD $ERRPC,$REG10,$REG0,$REG1,$REG2,$REG4,$REG5,$REG6,$REG7,C
6148 026430 001164 001166 001172
6149 026436 001174 001176 001200
6150 026444 000000
6151
6152 ;DATA BUFFERS
6153
6154 026446 IOBUF0:
6155 026446 000030 OUTBUF: .BLKW 24. ;IOBUF0 AND IOBUF1 ARE
6156 026526 000030 BUFR4: .BLKW 24. ;TWO - 768 WORDS LONG BUFFERS
6157 026606 000030 BUFR5: .BLKW 24. ;NORMALLY USED FOR DATA TRANSFERS
6158 026666 000030 BUFR6: .BLKW 24. ;TO AND FROM DISK
6159 026746 000030 BUFR7: .BLKW 24.
6160 027026 000200 BUFR10: .BLKW 128.
6161 027426 000200 BUFR11: .BLKW 128.
6162 030026 000200 BUFR12: .BLKW 128.
6163 030426 000200 BUFR13: .BLKW 128.
6164 031026 000210 BUFR14: .BLKW 136.
6165
6166 031446 001400 IOBUF1: .BLKW 768.
6167
6168
6169
6170 000001 .END

```

ABRT	016722	BUFR13	030426	DRV DON	001223	EXT1C	012732	NOSIN	011026
ADRES	001450	BUFR14	031026	DRV PTR	001226	EXT4	006614	NXTBA	023522
BAOTMO	002422	BUFR2	001352	DRV.RE=	104416	EXT5	007352	NXTDA	023546
BEGIN	010532	BUFR4	026526	DR.RST	016470	EXT6	010432	NXTDRV	003760
BEGSK	013250	BUFR5	026606	DSKADR	001432	EXT7	012170	NXTPAT	001426
BIT0	= 000001	BUFR6	026666	DSWR =	177570	FCHECK	024136	NXTWC	023642
BIT00	= 000001	BUFR7	026746	DT1	026334	FDRIVE	002114	OUTADR	001262
BIT01	= 000002	BUSAOR	001434	DT11	026366	FDRVE1	002116	OUTBUF	026446
BIT02	= 000004	CHKHOR	007260	DT17	026410	FFUNC	001216	PAT0	001414
BIT03	= 000010	CHKSWO	024064	DT24	026422	FINISH	011512	PAT1	001416
BIT04	= 000020	CHSW	023770	DT7	026350	FUNBEG	023172	PAT2	001420
BIT05	= 000040	CKSWR =	104407	EMTVEC=	000030	FUNERR	024004	PAT3	001422
BIT06	= 000100	CLERRR	012162	EM1	024334	GCYL	016434	PBUFO	001404
BIT07	= 000200	CMPAGA	011244	EM10	024564	GETBUF	010044	PBUF1	001406
BIT08	= 000400	CMPGRP	014622	EM11	024615	GETINF	016214	PGSUBR	001430
BIT09	= 001000	CN.RDY	016620	EM12	024633	GOBAK	011650	PIRQ =	177772
BIT1	= 000002	CN.RST	016602	EM13	024655	GOTYPE	013600	PIRGVE=	000240
BIT10	= 002000	COMPAR	011220	EM14	024704	GTSWR =	104406	PLOT	014222
BIT11	= 004000	CON.RD=	104421	EM16	025021	GT4RG	016162	PLTGRP	014106
BIT12	= 010000	CON.RE=	104415	EM2	024373	GT5RG	016136	PLTPT	014736
BIT13	= 020000	CR =	000015	EM20	025174	HT =	000011	PLT1	014436
BIT14	= 040000	CRLF =	000200	EM21	025262	INADR	001260	PRSPAT	001424
BIT15	= 100000	CSEORR	011654	EM22	025316	INDX1	001474	PRO =	000000
BIT2	= 000004	DOISP =	177570	EM23	025341	INDX2	001476	PR1 =	000040
BIT3	= 000010	DH1	025500	EM24	025362	INDX3	001500	PR2 =	000100
BIT4	= 000020	DH11	025653	EM25	025376	INDX4	001502	PR3 =	000140
BIT5	= 000040	DH13	025750	EM26	025440	INPT	024032	PR4 =	000200
BIT6	= 000100	DH14	025767	EM27	025456	IOBUFO	026446	PR5 =	000240
BIT7	= 000200	DH16	026046	EM3	024407	IOBUF1	031446	PR6 =	000300
BIT8	= 000400	DH17	026123	EM30	025473	IOTVEC=	000020	PR7 =	000340
BIT9	= 001000	DH24	026161	EM4	024423	LF =	000012	PS	= 177776
BLNKS1	002111	DH25	026266	EM5	024441	LUPHE	010640	PSW	= 177776
BLNKS2	002110	DH6	025546	EM6	024510	LUPSIN	010632	PTGEN0	010106
BLNKS3	002107	DH7	025576	EM7	024545	LUPSW	001222	PTGEN1	010170
BLNKS4	002106	DISPLA	001142	ERCNT1	001504	LUPWCE	011350	PTGEN2	010272
BLNKS5	002105	DISPRE	000174	ERCNT2	001506	MESSAGE=	104417	PTGEN3	010334
BLNKS6	002104	DMPREG	017662	ERCNT3	001510	MISCMP	012016	PTRNO1	010164
BLNKS7	002103	DONE	011636	ERCNT4	001512	MSGE	016044	PTRNO2	010166
BLNKS8	002102	DONTRK	011576	ERCNT5	001514	MSG1	001602	PT1	001560
BLNKS9	002101	DOSUR1	011630	ERCNT6	001516	MSG10	001742	PT10	001576
BLNK10	002100	DOWCHK	012420	ERCNT7	001520	MSG11	001771	PT11	001600
BLNK13	002075	DOWRT	012224	ERCNT8	001522	MSG12	002027	PT2	001562
BPTVEC=	000014	DRHOLD	002120	ERRTYP	017446	MSG13	002053	PT3	001564
BRKDA	016220	DRIVAD	001230	ERRVEC=	000004	MSG14	002064	PT4	001566
BTEOP	015232	DRIVS	001224	ERR1	016376	MSG2	001610	PT5	001570
BUFLG0	001410	DRIVO	001232	ERR2	016320	MSG3	001616	PT6	001572
BUFLG1	001412	DRIV1	001234	ERWCHK	011404	MSG4	001640	PT7	001574
BUFNO	001446	DRIV2	001236	ESR13	015456	MSG5	001657	PWRFL	004244
BUFR	001266	DRIV3	001240	ESR15	015364	MSG6	001716	PWRVEC=	000024
BUFR1	001320	DRIV4	001242	ESR20	015532	MSG7	001733	RDAGAI	010642
BUFR10	027026	DRIV5	001244	ESR25	015620	MS15	024757	RDCHR =	104410
BUFR11	027426	DRIV6	001246	EXEC	023676	MS17	025126	RDLIN =	104411
BUFR12	030026	DRIV7	001250	EXEC1	023674	NOHE	010760	RDOCT =	104412

READ	010606	STKLMT=	177774	TST7	010436	SENULL	015344	\$REG3	001170
REPEAT	012666	ST2	003512	TYPDES	022306	SEOP	015246	\$REG4	001172
REPMS	011274	ST3	003742	TYPDS =	104405	SEOPCT	015270	\$REG5	001174
REPTIM	013200	SWR	001140	TYPDSS=	104424	SERFLG	001103	\$REG6	001176
RESOON=	104423	SWREG	000176	TYPE =	104401	SERMAX	001115	\$REG7	001200
RESREG=	104414	SW0	= 000001	TYPMSG=	104420	SERROR	017162	\$RESRE	022450
RESVEC=	000010	SW00	= 000001	TYPOC =	104402	SERRPC	001116	\$RTNAD	015342
RES.DO	016514	SW01	= 000002	TYPON =	104404	SERRTB	002122	\$SAVRE	022412
RETRY1	001252	SW02	= 000004	TYPOS =	104403	SERTTL	001112	\$SAVR6	023160
RETRY2	001254	SW03	= 000010	TYPTIM	013736	SESCAP	001204	\$SCOPE	017006
RETRY3	001256	SW04	= 000020	TY.MSG	016106	SFILLC	001156	\$SETUP=	000117
RKBA	001462	SW05	= 000040	WBUSAD	001442	SFILLS	001155	\$STUP =	177777
RKCS	001456	SW06	= 000100	WCAGAI	011132	\$GDADR	001120	\$SUPRS	022352
RKDA	001464	SW07	= 000200	WCEROR	012062	\$GDOAT	001124	\$SVLAD	017120
RKDB	001466	SW08	= 000400	WCERR	012442	\$GET42	015320	\$SVPC =	000220
RKDS	001452	SW09	= 001000	WCHI	012446	\$GTSWR	021154	\$SWR =	163000
RKER	001454	SW1	= 000002	WCHI1	012440	\$HD =	000000	\$SWRMK=	000000
RKPRI	001470	SW10	= 002000	WCLO	012646	\$HIOCT	022056	\$TKB	001146
RKVEC	001472	SW11	= 004000	WCREPT	011334	\$ICNT	001104	\$TKCNT	020612
RKWC	001460	SW12	= 010000	WDSKAD	001440	\$ILLUP	023154	\$TKINT	020622
R6	=%000006	SW13	= 020000	WRDCNT	001436	\$INTAG	001135	\$TKQEN=	020621
R7	=%000007	SW14	= 040000	WRERR	012240	\$ITEMB	001114	\$TKQIN	020614
SAVREG=	104413	SW15	= 100000	WRHI	012400	\$LF	001214	\$TKQOU	020616
SBR1	006526	SW2	= 000004	WRLO	012242	\$LPADR	001106	\$TKQSR	020620
SBR2	006552	SW3	= 000010	WRLO1	012236	\$LPERR	001110	\$TKS	001144
SBR3	007224	SW4	= 000020	WRTCHK	011116	\$MNEW	021745	\$TKSRV	020672
SECPTR	010426	SW5	= 000040	WRDCN	001444	\$MSWR	021734	\$TN =	000013
SFTCMP	011166	SW6	= 000100	XHE	011442	\$MUL =	000003	\$TNPWR	022616
SIAO	011542	SW7	= 000200	XXDPMO	001220	\$MULT	020500	\$TPB	001152
SIZEF	024250	SW8	= 000400	\$AUTOB	001134	\$NULL	001154	\$TPFLG	001157
SKDON	015152	SW9	= 001000	\$BODR	001122	\$NWTST=	000001	\$TPS	001150
SMGRP	014472	TBITVE=	000014	\$BODAT	001126	\$OCNT	022302	\$TRAP	022702
SOAD	001524	TIMDON	014050	\$BELL	001206	\$OMODE	022304	\$TRAP2	022724
SORT	013342	TIMER	001264	\$CHARC	020474	\$OVER	017146	\$TRP =	000025
SP1	012702	TIMSEK	014776	\$CKSWR	021064	\$PASS	001100	\$TRPAD	022736
SP10	012724	TKVEC =	000060	\$CMTAG	001100	\$POWER	023162	\$TSTNM	001102
SP11	012726	TPVEC =	000064	\$CM1 =	000011	\$PWRAD	023150	\$TTYIN	021712
SP12	012730	TRAPVE=	000034	\$CM2 =	000022	\$PWRDN	023010	\$TYPDS	020034
SP2	012704	TRTVEC=	000014	\$CM3 =	000011	\$PWRMG	023144	\$TYPE	020260
SP3	012706	TSTRWS	016716	\$CNTLG	021727	\$PWRUP	023062	\$TYPEC	020430
SP4	012710	TST.RW=	104422	\$CNTLU	021722	\$QUES	001212	\$TYPEX	020476
SP5	012712	TST1	004260	\$CRLF	001213	\$RDCHR	021366	\$TYPOC	022104
SP6	012714	TST10	012174	\$DBLK	020250	\$RDLIN	021456	\$TYPON	022120
SP7	012716	TST11	012736	\$DB20	022506	\$RDOCT	021756	\$TYPDS	022060
SP8	012720	TST12	015224	\$DECVL	022666	\$RDSZ =	000010	\$XTSTR	017030
SP9	012722	TST2	004624	\$DOAGN	015340	\$REGAD	001160	\$SGETH=	000000
STACK =	001100	TST3	005122	\$DTBL	020240	\$REGD	001162	\$OFILL	022303
START	002432	TST4	005612	\$ENDAD	015330	\$REG1	001164	.	= 034446
STARTA	002530	TST5	006620	\$ENDCT	015276	\$REG10	001202		
START1	003J74	TST6	007356	\$ENDMG	015347	\$REG2	001166		

. ABS. 034446 000

ERRORS DETECTED: 0

B10

MD-11-DZKLE-E, RK11-RK05 DYNAMIC TEST MACY11 30(1046) 14-JUL-77 08:03 PAGE 119
DZKLE.P11 26-APR-77 12:27 SYMBOL TABLE

DSKM:DZKLE,DSKZ:DZKLE/SOL=DSKZ:SYSMAC.SML,DSKZ:DZKLE.P11
RUN-TIME: 15 21 .5 SECONDS
RUN-TIME RATIO: 124/37=3.3
CORE USED: 35K (69 PAGES)