

RH11/RP04

READ/WRITE TEST
MD-11-DZRPK-B

EP-DZRPK-B-DL-A

MAR 1976

COPYRIGHT ©1976

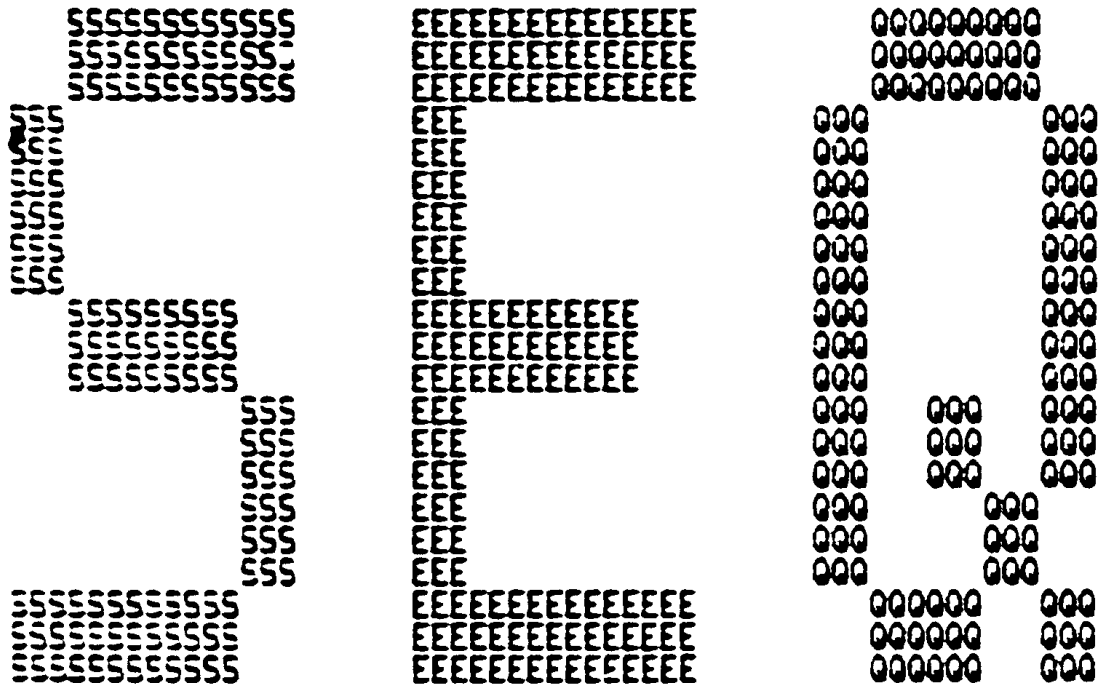
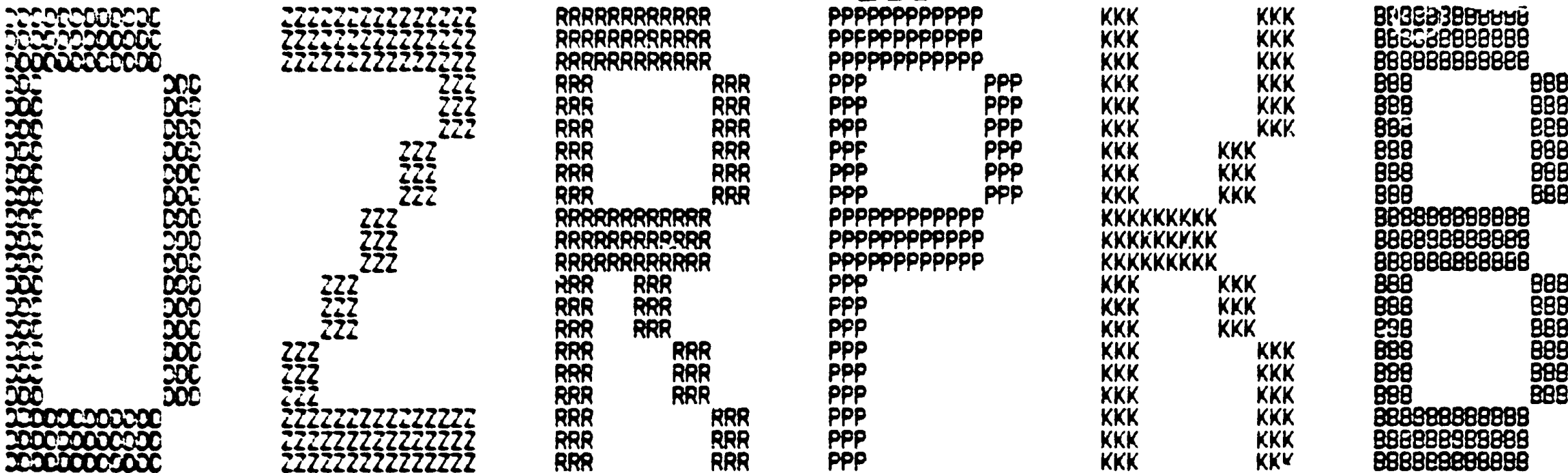
digital

FICHE 1 OF 1

Made in U.S.A.

DZRPKB SEQ

801



DSP Version 101(2111) Running on MTA260
JOB: JPRD User WELINGHAM, A11.21 Job DZRPKB Seq. 1900 Date 13-Feb-76 18:05:18 Monitor R2231D KL10 SYS#1026 *START*
Page: 1 created: 13-Feb-76 17:53:30
Job: MTN: DZRPKB SEQ(055) [404, 3722] Created: 13-Feb-76 15:07:30 Printed: 13-Feb-76 18:05:19
Job Switches: /FILE:ASCII /COPIES:1 /SPACING:1 /LIMIT:400 /FORMS:NORMAL

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZRPK-B-D
PRODUCT NAME: RH11/RP04 MECHANICAL AND READ/WRITE TEST
DATE CREATED: FEBRUARY 21, 1975
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: J. LACEY/C. 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) 1974, 1975 BY DIGITAL EQUIPMENT CORPORATION

CONTENTS

1. ABSTRACT
2. REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 STORAGE
 - 2.3 PRELIMINARY PROGRAMS
3. LOADING PROCEDURE
4. STARTING PROCEDURE
 - 4.1 CONTROL SWITCH SETTINGS
 - 4.2 STARTING ADDRESS
 - 4.3 OPERATOR ACTION
 - 4.4 PROGRAM ACTION
5. OPERATING PROCEDURE
 - 5.1 OPERATIONAL SWITCH SETTINGS
 - 5.2 SUBROUTINE ABSTRACTS
6. ERRORS
 - 6.1 ERROR TYPES
 - 6.2 ERROR RECOVERY
7. RESTRICTIONS
8. MISCELLANEOUS
 - 8.1 EXECUTION TIME
 - 8.2 STACK POINTER
 - 8.3 COMMON TAGS
 - 8.4 TIMING TESTS (TST7-TST12) PRINTOUTS
 - 8.5 END OF TEST
 - 8.6 RM11/RP04 DRIVER
9. PROGRAM DESCRIPTION
10. LISTING

1. ABSTRACT

THIS PROGRAM CONTAINS A SERIES OF TESTS THAT WILL INSURE THE DISK IS CAPABLE OF PERFORMING SEEKS, THE ACCESS TIMES ARE WITHIN TOLERANCE, THE TRACK AND SECTOR ADDRESSING CIRCUITRY OPERATES PROPERLY, AND THE DATA STORAGE AND RETRIEVAL CAPABILITIES ARE FUNCTIONAL.

2. REQUIREMENTS
-----2.1 EQUIPMENT

POP-11 COMPUTER WITH 16K OF MEMORY, CONSOLE TELETYPE, RH11/RP04 DISK WITH FORMATTED PACK, AND A KW11-P CLOCK (IF TIMING TESTS ARE DESIRED).

2.2 STORAGE

THIS PROGRAM WILL LOAD IN 12K BUT REQUIRES. 16K TO RUN

2.3 PRELIMINARY PROGRAMS

MAINDEC-11-DZRPS
MAINDEC-11-DZRPT
MAINDEC-11-DZRPU
MAINDEC-11-DZRPV

3. LOADING PROCEDURE

USE STANDARD PROCEDURE FOR LOADING .ABS TAPES

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

THE CONTROL SWITCH SETTINGS ARE INPUT VIA THE TTY.

TO ENTER THE CONTROL SWITCH SETTING MODE PLACE SW<07>=1 BEFORE PRESSING START. THEN UPON STARTING THE PROGRAM IT WILL TYPE THE PRESENT CONTENTS OF THE CONTROL SWITCH REGISTER (C.SWR) AND WAIT FOR THE NEW SETTING TO BE INPUT. THE INPUT STRING MUST CONSIST OF A SLASH (/), 1 TO 6 OCTAL DIGITS, TWO PERIODS (...) AND A CARRIAGE RETURN.

THE C.SWR SETTINGS ARE:

C.SWR<15>=0...WRITE PACK BEFORE TESTING (TEST16)
 =1...INHIBIT WRITE PACK BEFORE TESTING (TEST16)
 C.SWR<14>=0...NO STALL BETWEEN DRIVE FUNCTIONS
 =1...STALL AFTER EVERY DRIVE FUNCTION
 C.SWR<13>=0...USE SPECIFIC STALL TIMES
 =1...USE RANDOM STALL TIMES
 C.SWR<12>=0...NO INCREMENTING STALLS IN TEST4
 =1...PERFORM INCREMENTING STALLS IN TEST4
 C.SWR<08>=0...DO IMPLIED SEEKS WITH DATA TRANSFERS
 =1...DO EXPLICIT SEEKS BEFORE DATA TRANSFERS
 C.SWR<07>=0...DO READ HEADER AND DATA COMMANDS IN TESTS 0-6
 =1...DO EXPLICIT SEEK COMMANDS IN TESTS 0-6
 C.SWR<06>=0...60 HZ POWER SOURCE
 =1...50 HZ POWER SOURCE
 C.SWR<05>=0...ALLOW SOFTWARE TIMEOUTS(ENABLE WATCHDOG TIMER)
 =1...INHIBIT SOFTWARE TIMEOUTS(DISABLE WATCHDOG TIMER)

THE DEFAULT CONDITION OF C.SWR<15:00>=0.

REFER TO 4.4.1 FOR C.SWR SELECTION

4.2 STARTING ADDRESSES

200 NORMAL STARTING ADDRESS
 204 SELECT OPERATING PARAMETERS
 210 SELECT RH11/PP04 ADDRESSES
 214 COMBINATION OF 204 AND 210

4.3 OPERATOR ACTION

1. LOAD PROGRAM INTO MEMORY (SEE SECTION 3.)
2. LOAD A FORMATTED PACK INTO DRIVE(S) TO BE TESTED
3. BRING DRIVE(S) TO ONLINE STATE, WRITE ENABLED, AND LOCKED ON PORT.
4. LOAD ADDRESS 200.
5. SET SWITCHES (SEE SECTION 4.1 & 5.1)
6. PRESS START.

4.4 PROGRAM ACTION

IN AN EFFORT TO ALLOW CONVERSATION WITH A PROGRAM FOR THE PURPOSE OF CONTROLLING ITS OPERATION AND PARAMETERS THE FOLLOWING CONSTRUCTIONS HAVE BEEN ADOPTED.

NOTE1: IN ALL EXAMPLES BRACKETS ARE USED FOR CLARITY AND ARE NOT TYPED BY THE USER.

NOTE2: THE CARRIAGE RETURN TYPED BY THE USER IS INDICATED BY <CR> AND WILL BE ECHOED AS A "CARRIAGE RETURN-LINE FEED."

<.)<CR> PERIOD

A STATEMENT TERMINATOR: WHEN TYPED AT THE END OF A LINE (LEGAL ON ALL LINES) IT TELLS THE PARAMETER STRING INTERPRETER (PSI) THIS IS THE END OF CHANGES TO THE CURRENT PARAMETER STRING.

<..)<CR> PERIOD PERIOD

AN INPUT TERMINATOR: WHEN A PERIOD IS TYPED AFTER A "STATEMENT TERMINATOR PERIOD" IT TELLS THE PSI THIS IS THE END OF ALL CHANGES (LEGAL ON ALL LINES) BEGIN EXECUTION.

<.,)<CR> COMMA

A CONTINUATION INDICATOR AND FIELD SEPARATOR:

- 1) THIS IS THE CONSTRUCTION USED WHEN NOT USING A PERIOD: THAT IS TO SAY THE COMMA SAYS "TERMINATE THE LINE BUT NOT THE INPUT FOR PARAMETER CHANGES".
- 2) SEPARATE MULTIPLE ENTRIES ON THE SAME TEXT STRING, I.E., A,B,C,D.

- (/) SLASH
A MODIFICATION INDICATOR: AT ANY GIVEN PARAMETER STOP IF A SLASH IS TYPED IT SAYS "OPEN THIS PARAMETER FOR CHANGES".
- (↑U) CONTROL-U
DUMP THE PRESENT INPUT STRING AND START A NEW LINE. TYPED BY DEPRESSING THE "CONTROL KEY" (CTRL) AND THEN STRIKING THE "U".
- (\) RUBOUT
DELETE THE LAST CHARACTER FROM THE INPUT STRING. TYPED BY STRIKING THE "RUBOUT" KEY, WHICH WILL BE ECHOED BY A BACKSLASH (\) FOLLOWED BY THE CHARACTER DELETED.

4.4.1 CONTROL SWITCHES SELECTION

STARTING THE PROGRAM AT ANY OF THE POSSIBLE STARTING ADDRESSES WITH SW<07>=1 WILL RESULT IN ENTERING THE "CONTROL SWITCH SETTING" MODE. THUS, ALLOWING THE OPERATOR TO SPECIFY THE DESIRED STATE OF "C.SWR".

4.4.1.1 CONTROL SWITCH SELECTION EXAMPLES

EXAMPLE #1

SET SW<07>=0
C.SWR=000000/400..

EXAMPLE #2

SET SW<07>=0
C.SWR=000000/220.
C.SWR=000000/220..

4.4.2 ADDRESS SELECTION

STARTING THE PROGRAM AT 200 WILL RESULT IN AUTOMATIC SELECT OF THE DEFAULT VALUES OF BUS ADDRESS (RHCSI), VECTOR ADDRESS, AND PRIORITY LEVEL OF THE RH11/RP04. IF THE DEFAULT VALUE OF THE BUS ADDRESS DOES NOT RESPOND (TIMES OUT) TO ADDRESSING IT IS REPORTED AS AN ERROR. AFTER THE ERROR IS REPORTED ONE OF TWO COURSES OF ACTION WILL BE TAKEN:

1. IF THERE IS A MONITOR -- RETURN TO THE MONITOR
2. IF THERE ISN'T A MONITOR -- ASK FOR NEW ADDRESSES

STARTING THE PROGRAM AT 210 OR 214 ALLOWS THE OPERATOR TO CHANGE THE CONTROL AND STATUS REGISTER #1 (RHCSI) ADDRESS, THE VECTOR ADDRESS, AND THE PRIORITY LEVEL.

4.4.2.1 ADDRESS SELECTION EXAMPLES

EXAMPLE #1

RHCSI=177200/176300..

EXAMPLE #2

RHCSI=177200/176300.
RHVEC=254/260,
RHPRIO=5/6..

EXAMPLE #3

RHCSI=177200,
RHVEC=254/260..

EXAMPLE #4

RH11/RP04 FAILED TO RESPOND TO ADDRESSING
RHCSI ERR PC
177200 XXXXXX
RHCSI=177200/176300..

EXAMPLE #5

RHCSI=177200/1776\67\6300,
RHVEC=254,
RHPRIO=5,
RHCSI=176300..

4.4.3 DRIVE AND PARAMETER SELECTION

STARTING THE PROGRAM AT 200 WILL RESULT IN AUTOMATIC SELECTION OF THE DRIVES TO TEST AND THE TESTS TO RUN.

STARTING THE PROGRAM AT 210 OR 214 ALLOWS THE OPERATOR TO SELECT THE DRIVE(S) TO BE TESTED, THE TESTS TO BE EXECUTED, AND THE PARAMETERS TO USE.

4.4.3.1 DRIVE AND PARAMETER SELECTION DESCRIPTION

THE FOLLOWING IS A TABLE OF TERMS USED BY THE PSI.

"R"	REPEATS (ITERATIONS)
"FC"	FIRST CYLINDER ADDRESS
"LC"	LAST CYLINDER ADDRESS
"IC"	INCREMENT CYLINDER
"FT"	FIRST TRACK ADDRESS
"LT"	LAST TRACK ADDRESS
"IT"	INCREMENT TRACK
"FS"	FIRST SECTOR ADDRESS
"LS"	LAST SECTOR ADDRESS
"PAT"	PATTERN (USED FOR DATA TEST)
"WDX"	WORD OF PATTERN 0 WHERE X IS 1 TO 16
*"S"	ALL SEEK TESTS
*"T"	ALL TIMING TESTS
*"A"	ALL ADDRESS TESTS
*"D"	THE DATA TESTS
*"E"	THE EXERCISER

* USED BY USER AS INPUT.

NOTE1: ALL OTHERS ARE OUTPUT ONLY.

NOTE2: ALL NUMBERS WILL BE IN DECIMAL EXCEPT FOR THE PATTERN (PAT) AND WORDS (WDX) SELECTION. "PAT" WILL BE SELECTED BY A BIT (I.E. 001000(8)=PATTERN 9) AND "WDX" WILL BE IN OCTAL.

SPECIAL CASES OF CONTROL CHARACTERS

IF (<.>) IS TYPED WHILE A TEST IS OPEN FOR MODIFICATION (</>) AND OTHER TESTS IN THE "TEST COMMAND" STRING ARE TO BE OPENED, THEY WILL RECEIVE THE PARAMETERS OF THE TEST THAT IS OPEN WHEN (<.>) IS TYPED.

(<.>) IS ILLEGAL AS A "TEST" LINE TERMINATOR UNLESS A TEST IS TO BE OPENED.

ON STARTING THE PROGRAM ALL TESTS WILL RUN AGAINST ALL DRIVES IN A WORST CASE MANNER. BUT IF THE USER DESIRES TO SELECT THE DRIVES TO BE TESTED, THE TESTS TO BE EXECUTED, AND THE PARAMETERS TO BE USED HE ENTERS CONVERSATION MODE BY TYPING CONTROL C (C).

THE PROGRAM WILL RESPOND WITH:

DRIVE(S)=

WE WILL ASSUME FOR OUR EXAMPLE THAT THE OPERATOR WISHES TO TEST DRIVE #3 USING TESTS 2 THRU 7 AND TEST 11 AND DOES NOT DESIRE TO CHANGE THE PARAMETERS (INITIAL CYLINDER ADDRESS, FINAL CYLINDER ADDRESS, ETC.). THE USER WOULD TYPE "3< >" WHICH SAYS "THIS IS THE END OF ANY CHANGES TO THIS LINE BUT TAKE ME DEEPER FOR MORE CHANGES", AND <CR> "TAKE IT."

FOR CLARITY THE LINE WILL BE REPEATED AS IT LOOKS AFTER USER RESPONSE WITH THE LINE BELOW IT ADDED, THE PROGRAMS RESPONSE:

DRIVE(S)=3,<CR>
TEST=

NOW THE USER INSERTS THE TEST NUMBERS HE DESIRES. IN THIS CASE HE WANTS TESTS 2 THRU 7 AND TEST 11 SO HE TYPES 2-7< > (TO SEPERATE FIELD), 11<..> (END OF ANY CHANGES START EXECUTION), <CR> (TAKE IT).

IT NOW LOOKS LIKE THIS

DRIVE(S)=3,<CR>
TEST=2-7,11.<CR>

IN OUR NEXT EXAMPLE WE WILL ASSUME THE USER WISHES TO TEST DRIVE 4 AND RUN TESTS 1 AND 3 THRU 11 WITH THE PARAMETER FOR TESTS 3 AND 10 MODIFIED. THIS IS HOW HE WOULD RESPOND:

DRIVE(S)=4,<CR>
TEST=

THE USER NOW WILL INSERT A LINE TO GET TESTS 1 AND 3 THRU 11 BUT REMEMBER HE WISHES TO "OPEN" TESTS 3 AND 10 FOR PARAMETER CHANGES. THE ENTIRE ENTRY IS SHOWN BELOW:

DRIVE(S)=4,<CR>
TEST=1,3/4-7,10/11.<CR>

NOTICE THIS SAYS SELECT TEST ONE, CONTINUE< >; SELECT TEST 3, OPEN</>; SELECT TESTS 4-7, CONTINUE<,>; SELECT TEST 10, OPEN</>; SELECT TEST 11, END OF INPUT<.>.

THE PSI SCANS THE TEST STRING AND DETERMINES THAT TEST #1'S PARAMETERS WILL REMAIN THE SAME BUT TEST 3'S MUST BE CHANGED. THEREFORE, TEST 3 IS OPENED FOR CHANGE.

RESPONDS: (FOR CLARITY ENTIRE TRANSACTION REPEATED)

```
DRIVE(S)=4,<CR>
TEST=1,3/4-7,10/11.<CR>
TEST 3
R=X           ;WHERE X IS ITERATION
```

AS IN THE NORMAL MANNER TYPING A (</>) WILL OPEN THIS LINE FOR MODIFICATION OF THE ITERATION COUNT, ENDING THE LINE WITH A (<.>) (PERIOD) WOULD TERMINATE THE CHANGES FOR THIS TEST, AND AS IN THE EXAMPLE SHOWN USING A (<,>) (COMMA) WILL GET US THE NEXT PARAMETER.

```
DRIVE(S)=4,<CR>
TEST=1,3/4-7,10/11,<CR>
TEST 3
R=1,<CR>      ;DO NOT ALTER-BUT CONTINUE
FC=N          ;WHERE N IS FIRST CYLINDER ADDRESS
```

THE USER DOES NOT WISH TO CHANGE "FC" SO THE FOLLOWING ACTION IS TAKEN:

```
DRIVE(S)=4,<CR>
TEST=1,3/4-7,10/11.<CR>
TEST 3
R=1,<CR>      ;DO NOT ALTER THIS LINE BUT CONTINUE
FC=0,<CR>    ;DO NOT ALTER THIS LINE BUT CONTINUE
LC=410
```

THE PROGRAM RESPONDS WITH THE PREVIOUSLY ASSIGNED PARAMETER FOR LAST CYLINDER ADDRESS IN THIS CASE USING 410 AS THE EXAMPLE. THIS IS WHAT THE USER INTENDED TO MODIFY AND IS WHY HE OPENED TEST #3. HE WANTS TO CHANGE IT TO CYLINDER 20. THEREFORE HE TYPES (</>) (OPEN LINE FOR CHANGE), "20" (THE NEW VALUE), (<.>) (THIS IS THE END OF CHANGES FOR THIS DRIVE), (<CR>) (TAKE IT).

THE TOTAL TRANSACTION AND RESPONSE:

```
DRIVE(S)=4,<CR>
TEST=1,3/4-7,10/11.<CR>
TEST 3
R= 1,<CR>
FC=0,<CR>
LC= 410/20.<CR>
TEST 10
R=1
```

THE PROGRAM HAS LOADED TEST #3 WITH ITS NEW PARAMETERS, TESTS NUMBER 4 THRU 7 WITH THEIR PREVIOUSLY ASSIGNED PARAMETERS AND IS NOW WAITING FOR CHANGES TO TEST 10.

THE USER TYPES (</>) (OPEN THIS LINE FOR CHANGE), "10" (MAKE ITERATION COUNT 10), (<.>) (THIS IS THE END OF CHANGES TO THIS TEST) (<CR>) (TAKE IT).

THE PROGRAM NOW LOADS TEST 10 WITH THE NEW PARAMETERS, TEST 11 WITH PREVIOUSLY ASSIGNED PARAMETERS AND RESPONDS WITH:

DRIVE(S)=

SINCE THE USER DID NOT END THE CONVERSATION MODE WITH A (<.>) THE PROGRAM HAS LOOPED BACK TO THE BEGINNING LOOKING FOR MORE CHANGES. THAT IS TO SAY, AFTER THE ENTRY FOR DRIVE SELECTION A (<.>)<CR> WILL CAUSE THE TEST MESSAGE TO BE REPEATED AND FURTHER CHANGES CAN BE MADE. HOWEVER, AT SOME POINT IN ORDER TO EXECUTE THE CHANGES MADE WE MUST NOTIFY THE PROGRAM WE ARE FINISHED BY TYPING (<.>).

EXAMPLE #1

```

DRIVE=4.<CR>           ;SELECT DRIVE #4, TERMINATE AND
                       ;BEGIN EXECUTION USING PREVIOUSLY ASSIGNED
                       ;PARAMETERS

```

EXAMPLE #2

```

DRIVE=0.<CR>           ;SELECT DRIVE #0 AND MAKE CHANGES " "
TEST=1-5.<CR>         ;RUN TEST 1 THRU 5 ONLY, USE DEFAULT
                       ;PARAMETERS AND TERMINATE AND EXECUTE ".."

```

EXAMPLE #3

```

DRIVE=2.<CR>           ;SELECT DRIVE #2 AND MAKE CHANGES " "
TEST=1-5,6/7/10/<CR> ;RUN TEST 1-5 WITH DEFAULT PARAMETERS, OPEN
TEST 6                 ;TEST 6, 7 AND 10 FOR CHANGES
R=1.<CR>               ;LEAVE R AS IS MOVE TO NEXT PARAMETER
FC=0/10.<CR>          ;SET "FC" CYLINDER ADDRESS TO 10 "." END CHANGES
                       ;TO TEST #6

```

```

TEST 7                 ;50 ITERATIONS, MOVE TO NEXT PARAMETER
R=1/50.<CR>            ;DO NOT CHANGE "FC" CYLINDER ADDRESS BUT CONTINUE ","
FC=0.<CR>              ;TEST 10 IS STILL PENDING
LC=410/50.<CR>        ;AND WILL THEREFORE BE
                       ;SET TO THE SAME PARAMETERS
                       ;AS TEST 7 -- START EXECUTION

```

EXAMPLE #4

```

DRIVE=0.<CR>           ;SELECT DRIVE #0 AND MAKE CHANGES
TEST=5,E.<CR>         ;RUN ALL SEEK TESTS AND THE EXERCISER

```

EXAMPLE #5

```

DRIVE=1.<CR>           ;RUN ALL SEEK TESTS (OPEN FOR CHANGES) AND
TEST=3/0.<CR>         ;THE DATA TEST (WITH DEFAULT PARAMETERS).
TEST 0                ;RUN WITH 10 ITERATIONS
R=10.<CR>             ;CHANGE FIRST CYLINDER ADDRESS
FC=0/10.<CR>         ;AND START EXECUTION
                       ;TESTS 1-6 WILL BE ASSIGNED THE
                       ;SAME PARAMETERS AS TEST 0

```

EXAMPLE #6

```

DRIVE=1, <CR>
TEST=S/, <CR>           ; OPEN THE SEEK TESTS (TST 0-6)
TEST 0
  R=10/100, <CR>       ; CHANGE TO 100 ITERATIONS; (<.>) MOVE TO NEXT TEST
TEST 1
  R=100/1000, <CR>     ; CHANGE R TO 1000 ITERATIONS; (<.>) MOVE TO NEXT TEST
TEST 2
  R=1/10, <CR>         ; CHANGE R TO 10 ITERATIONS           ; (<.>) MOVE TO NEXT LINE
  FC=0/50, <CR>        ; CHANGE FC TO 50; (<.>) MOVE TO NEXT LINE
  LC=410/51, <CR>      ; CHANGE LC TO 51; (<.>) MOVE TO NEXT TEST
TEST 3
  R=1, <CR>            ; (<.>) MOVE TO NEXT TEST
TEST 4
  R=1., <CR>           ; USE TESTS 4'S PARAMETERS
                        ; FOR TEST 5 & 6 START EXECUTION

```

EXAMPLE #7

```

DRIVE=1, <CR>
TEST=D/, <CR>           ; SELECT AND OPEN THE DATA TEST
TEST 15
  R=1/1000, <CR>       ; DO 1000 ITERATION OF TEST PATTERN
  FC=0/10, <CR>        ; #8 ON CYLINDER 10, TRACK 2, SECTOR 4
  LC=410/10, <CR>
  IC=64/0, <CR>
  FT=0/2, <CR>
  LT=18/2, <CR>
  IT=1, <CR>
  FS=0.4, <CR>
  LS=22/4, <CR>
  PAT=177777/400., <CR> ; RUN WITH PATTERN #8

```


5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

WITH SW<15:0>=0 THE PROGRAM WILL PRINT OUT ON ERRORS AND CONTINUE IN TEST.
THE SWITCH SETTINGS ARE:

SW<15>=1...HALT ON ERROR
 SW<14>=1...LOOP ON TEST
 SW<13>=1...INHIBIT ERROR TYPEOUTS
 SW<11>=1...INHIBIT ITERATIONS
 SW<10>=1...RING BELL ON ERROR
 SW<09>=1...LOOP ON ERROR
 SW<08>=1...PRINT ERROR MESSAGE ON LINE PRINTER
 SW<07>=1...READ CONTROL SWITCH SETTINGS FROM TTY
 SW<06>=1...INHIBIT TIME REPORTS (TESTS 7-12)
 SW<05>=1...REPORT ONE ERROR PER SECTOR (TESTS 13&14)
 SW<04>=1...INHIBIT WRITES (TEST 15)
 SW<03>=1...INHIBIT WRITE CHECKS (TEST 15)
 SW<02>=1...INHIBIT READ AND SOFTWARE COMPARES (TEST 15)
 SW<01>=1...INHIBIT SOFTWARE COMPARES (TEST 15)
 SW<00>=1...PERFORM READ AFTER WRITE CHECK ERROR (TEST 15)

5.2 SUBROUTINE ABSTRACTS

THE SUBROUTINE CALLS USED BY THIS PROGRAM ARE:

1. IOT (USED FOR SCOPE)
THIS CALL IS PLACED BETWEEN EACH TEST IN THE INSTRUCTION SECTION. IT RECORDS THE STARTING ADDRESS OF EACH TEST IN LOCATIONS "SLPADR" AND "SLPERR" AS IT IS BEING ENTERED.
2. EMT (USED FOR ERROR)
THIS CALL IS USED TO REPORT ALL ERRORS (REFER TO 6).
3. TRAP (USED FOR TYPE, TYPOC, TYPOS, TYPON, TYPDS, ROCHR, RDLIN, SAVREG, AND RESREG)
4. JSR R,ADR

THE FOLLOWING SECTION CONTAINS AN ABSTRACT FOR EACH SUBROUTINE

6. ERRORS

THERE ARE A NUMBER OF ERRORS THAT CAN OCCUR IN THIS PROGRAM. WHEN AN ERROR IS ENCOUNTERED THE CALL TO THE ERROR ROUTINE IS MADE AND IF SW<13> IS NOT SET AN ERROR MESSAGE PERTAINING TO THE ERROR WILL BE TYPED. EACH ERROR TYPE OUT WILL CONTAIN THE FOLLOWING:

1. AN ERROR MESSAGE
2. A DATA HEADER
3. A DATA STRING

REFER TO THE FOLLOWING SECTION FOR THE DIFFERENT ERRORS THAT CAN OCCUR.

6.1 ERROR TYPES

THE ERRORS THAT OCCUR IN THIS PROGRAM FALL INTO THREE (3) CATEGORIES DEFINED AND EXPLAINED AS FOLLOWS:

6.1.1 DRIVER ERROR

THESE ERRORS WILL BE DETECTED BY THE RH11/RP04 DRIVER. THERE ARE TWO CLASSES OF DRIVER ERRORS; THOSE THAT CAN NOT BE IDENTIFIED IN A MANNER THAT ALLOWS THE INFORMATION TO BE RETURNED TO A "DATA PARAMETER BLOCK" (DPB) AND THOSE THAT CAN. THE FIRST CLASS WILL BE REPORTED BY ERROR CALLS (EMT'S) 1-5 WITHIN THE DRIVER. THE SECOND CLASS WILL PASS THE ERROR CODES TO THE STATUS/ERROR WORD (DPB+16) OF THE PROPER DPB.

6.1.2 NON-FATAL ERROR

THESE ERRORS WILL BE DUE TO "DISK" OR "DATA" FAILURES WHICH WILL BE REPORTED AS THEY OCCUR. AFTER REPORTING THE ERROR THE PROGRAM WILL CONTINUE TESTING.

6.1.3 FATAL ERROR

THIS TYPE OF ERROR WILL BE THE RESULT OF ANY KIND OF ERROR THAT INHIBITS THE PROGRAM FROM TESTING THE DISK.

THIS ERROR WILL BE REPORTED WHEN IT OCCURS, THEN THE PROGRAM WILL ABORT THE TEST AND GO TO THE END OF PROGRAM.

6.2 ERROR RECOVERY

6.2.1 PRETEST ERROR

WHEN THIS TYPE OF ERROR OCCURS IT WILL BE REPORTED. THEN DEPENDING ON HOW THE PROGRAM WAS STARTED IT WILL ASK FOR THE DRIVES AND ADDRESSES FOR TESTING OR RETURN TO MONITOR.

6.2.2 NON-FATAL ERROR

WHEN THIS TYPE OF ERROR OCCURS IT WILL BE REPORTED AND THE PROGRAM WILL CONTINUE IN TEST.

6.2.3 FATAL ERROR

WHEN THIS TYPE OF ERROR OCCURS IT WILL BE REPORTED. THE PROGRAM WILL ABORT THE TEST AND GO TO THE END OF PROGRAM.

7. RESTRICTIONS

BEFORE STARTING THE PROGRAM THE OPERATOR MUST INSURE THAT A FORMATTED PACK IS LOADED AND WRITE ENABLED IN THE DRIVE(S) TO BE TESTED.

8. MISCELLANEOUS

8.1 EXECUTION TIME

TO MAKE ONE PASS OF THE PROGRAM TAKES APPROXIMATELY 5 MINUTES. THE RUN TIME INCREASES LINEARLY FOR EACH DRIVE TESTED.

8.2 STACK POINTER

STACK IS INITIALLY SET TO 1100 AND EXTENDS DOWN IN MEMORY.

8.3 COMMON TAGS

THE COMMON TAGS USED IN THIS PROGRAM START AT 1100 AND EXTEND UP IN MEMORY.

8.4 TIMING TESTS (TST7-TST12) PRINTOUTS

AT THE COMPLETION OF EACH OF THE TIMING TESTS THE TIME OF THE MINIMUM SEEK, MAXIMUM SEEK, AND THE AVERAGE OF ALL OF THE SEEKS PERFORMED ARE TYPED ON THE TTY. THE NUMBER OF SEEKS THAT HAD TIMES BELOW THE MINIMUM TIME ALLOWED WILL BE TYPED ON THE SAME LINE AS THE MINIMUM TIME. THE NUMBER ABOVE THE MAXIMUM WILL BE TYPED ON THE SAME LINE AS THE MAXIMUM TIME, AND THE TOTAL NUMBER OF SEEKS PERFORMED WILL BE ON THE SAME LINE AS THE AVERAGE.

NOTE: THE PROGRAM STALLS FOR 2 MILLISECONDS BETWEEN SEEK ORDERS. THIS STALL TIME IS NOT INCLUDED IN THE CALCULATED SEEK TIMES. THE 2 MILLISECOND STALL BETWEEN SEEK ORDERS IS SPECIFIED BY THE RPO4 VENDOR. THE SEEK TIMES SPECIFIED BY THE VENDOR ARE POSITIONER MOVEMENT TIMES ONLY AND ARE NOT A MEASUREMENT OF EFFECTIVE SEEK TIME.

8.4.1 TIMING TOLERANCES

1. TEST 7 -- ROTATIONAL SPEED TIMES

60 HZ
MINIMUM=16340 US
MAXIMUM=17000 US
NOMINAL=16670 US

50 HZ
MINIMUM=16250 US
MAXIMUM=17090 US
NOMINAL=16670 US

2. TEST 10 -- ONE CYLINDER SEEK TIMES

MINIMUM=3000 US
MAXIMUM=10000 US
NOMINAL=7000 US

3. TEST 11 -- AVERAGE SEEK TIMES

MINIMUM=26000 US
MAXIMUM=30000 US
NOMINAL=28000 US

4. TEST 12 -- MAXIMUM SEEK TIMES

MINIMUM=46000 US
MAXIMUM=52000 US
NOMINAL=50000 US

EXAMPLE #1

ROTATIONAL SPEED TIMES

MIN=16670 US
MAX=16690 US
AVG=16680 US 10 SEEKS TIMED

ONE CYLINDER SEEK TIMES

** 0, 1, 2, ... 410 **

MIN=5350 US
MAX=6920 US
AVG=5550 US 409 SEEKS TIMED

** 410, 409, 408, ... 0 **

MIN=5140 US
MAX=5960 US
AVG=5430 US 410 SEEKS TIMED

AVERAGE SEEK TIMES

** 0 TO 136 **

MIN=27773 US
MAX=28640 US
AVG=28230 US 128 SEEKS TIMED

** 136 TO 0 **

MIN=27990 US
MAX=28550 US
AVG=28220 US 128 SEEKS TIMED

MAXIMUM SEEK TIMES

** 0 TO 410 **

MIN=49990 US
MAX=51980 US
AVG=51010 US 128 SEEKS TIMED

** 410 TO 0 **

MIN=48120 US
MAX=50650 US
AVG=49340 US 128 SEEKS TIMED

EXAMPLE #2

ROTATIONAL SPEED TIMES

MIN=16670 US
MAX=16690 US
AVG=16680 US 10 SEEKS TIMED

ONE CYLINDER SEEK TIMES

** 0, 1, 2...410 **
MIN=5470 US
MAX=10940 US 3 ABOVE THE MAXIMUM OF 10000 US
AVG=5830 US 409 SEEKS TIMED
** 410, 409, 408,...0 **
MIN=5040 US
MAX=5970 US
AVG=5330 US 410 SEEKS TIMED

AVERAGE SEEK TIMES

** 0 TO 136 **
MIN=29730 US
MAX=31620 US 73 ABOVE THE MAXIMUM OF 30000 US
AVG=30320 US 128 SEEKS TIMED
** 136 TO 0 **
MIN=28620
MAX=31230 US 128 ABOVE THE MAXIMUM OF 30000 US
AVG=30800 US 128 SEEKS TIMED

MAXIMUM SEEK TIMES

** 0 TO 410 **
MIN=53510 US
MAX=54240 US 128 ABOVE THE MAXIMUM OF 52000 US
AVG=54020 US 128 SEEKS TIMED
** 410 TO 0 **
MIN=52050 US
MAX=54550 US 128 ABOVE THE MAXIMUM OF 52000 US
AVG=52210 US 128 SEEKS TIMED

8.5 END OF TEST

 WITH ALL SWITCHES ON A "0" AN "END OF PASS" MESSAGE WILL BE TYPED AT THE COMPLETION OF TESTING A DRIVE AND THE "END OF TEST" TYPEOUT WILL OCCUR WHEN ALL DRIVES HAVE BEEN TESTED.

8.6 RH11/RP04 DRIVER

 ALL OF THE COMMUNICATION WITH THE RH11/RP04 IS PERFORMED VIA A COMMON DRIVER. THIS SECTION CONTAINS THE USERS GUIDE FOR THE DRIVER.

8.6.1. TO INITIALIZE THE DRIVER:

```
JSR   PC,RPINIT
RETURN
```

UPON RETURN YOU MUST EXAMINE THE "DRVSTA" TABLE TO DETERMINE THE DRIVES THAT ARE ONLINE FOR TESTING. THE DRVSTA TABLE IS EIGHT BYTES; ONE BYTE PER DRIVE. THE STATE OF EACH DRIVE WILL BE INDICATED AS FOLLOWS:

```
>0  ON LINE
=0  OFFLINE
<0  UNSAFE OR NONEXISTENT
```

WITH NON-RP04 DRIVES INDICATED AS OFFLINE. AND THE DRIVE TYPE WILL BE INDICATED IN THE "DRVSTYP" TABLE. USING "DRVSTA" AND "DRVSTYP" THE FOLLOWING CAN BE DETERMINED:

ONLINE	OFFLINE	UNSAFE	NONEXISTENT
-----	-----	-----	-----
DRVSTA>0	DRVSTA=0	DRVSTA<0	DRVSTA<0
DRVSTYP=(RHDT)	DRVSTYP=(RHDT)	DRVSTYP=(RHDT)	DRVSTYP=0

THE RPINIT ROUTINE WILL DO A READIN PRESET AND WILL SET FMT22.

8.6.2. AFTER THE DRIVER HAS BEEN INITIALIZED YOU CAN USE IT TO PERFORM FOR YOU.

```
CALL:      JSR      RD,RPO4      ;MAKE THE CALL
           PNTDPB      ;ADDRESS OF DPB*
           RETURN1     ;RETURN IF QUEUE IS FULL
           RETURN2     ;RETURN IF REQUEST IS IN
                       ;QUEUE OR THERE IS AN
                       ;ERROR CONDITION
```

*DPB (DATA PARAMETER BLOCK)

```
PNTDPB: .BYTE 0      ;(0) DRIVE NUMBER
         .BYTE 0      ;(1) OFFSET VALUE OR FMT22, ECT, AND HCI
         .BYTE 0      ;(2) COMMAND
         .BYTE 0      ;(3) PSEL AND A17 AND A16
         .WORD 0      ;(4) WORD COUNT (MUST BE NEG.)
         .WORD 0      ;(6) BUFFER ADDRESS OR
         ;REGISTER TABLE POINTER
         .BYTE 0      ;(10) SECTOR ADDRESS OR
         ;FIRST REG. INDEX
         .BYTE 0      ;(11) TRACK ADDRESS OR
         ;LAST REG. INDEX
         .WORD 0      ;(12) CYLINDER ADDRESS
         .WORD 0      ;(14) ERROR TABLE POINTER
         ;POINTS TO THE FIRST OF TWENTY
         ;LOCATIONS OF WHERE THE DRIVER
         ;IS TO STORE THE RH11/RPO4
         ;REGISTERS ON AN ERROR. IF LEFT
         ;ZERO REGISTERS ARE NOT SAVED.
         .WORD 0      ;(16) STATUS/ERROR INDICATOR
         ;BIT15=1=>ERROR OCCURRED
         ;BIT07=1=>DONE
         ;BIT14-BIT09 AND BIT06-BIT03
         ;INDICATE TYPE OF ERROR
```

8.6.3. THE DRIVER PROVIDES A SOFTWARE TIMEOUT CAPABILITY. TO UTILIZE THIS CAPABILITY YOU MUST SUPPLY THE "RP TIMER" ROUTINE WITH THE ELAPSED TIME IN THE FOLLOWING MANNER:

```

MOV    #16.,-(SP)    ;16 MILLISECONDS BETWEEN
                    ;CLOCK TICKS
JSR    RO,RPTMR     ;CALL THE TIMER ROUTINE

```

IT SHOULD BE NOTED THAT YOU MUST PROVIDE THE CODE TO DRIVE THE CLOCK. AND THE ELAPSED TIME MUST BE IN MILLISECONDS. THE DRIVER WILL SET THE TIMEOUT TO 1 SECOND FOR ALL POSITIONING AND DATA TRANSFER OPERATIONS AND WILL SET THE TIMEOUT TO 30 SECONDS FOR ERROR RECOVERY OPERATIONS.

8.6.4. EXAMPLE - WRITE 1000. WORDS

```

15:    JSR    RO,RPO3    ;CALL THE DRIVER
        WRTDPB    ;DPB ADDRESS
        BR     15        ;WAIT FOR QUEUE IF FULL
25:    TST    WRTDPB+16  ;WAIT FOR COMMAND TO COMPLETE
        BEQ    25
        BMI    ERROR1   ;ERROR OCCURRED
        .
        .

```

```

WRTDPB: .BYTE    5        ;DRIVE #5
        .BYTE    0
        .BYTE    161     ;WRITE COMMAND
        .BYTE    0
        .WORD    -1000.   ;WORD COUNT
        .WORD    WRTBUF   ;BUFFER ADDRESS
        .BYTE    3        ;SECTOR
        .BYTE    5        ;TRACK
        .WORD    400     ;CYLINDER
        .WORD    ERRTB5   ;ERROR TABLE
        .WORD    0        ;STATUS/ERROR INDICATOR

```

ALTERNATE DPB SETUP

```

WRTDPB: .WORD    5        ;THIS SETUP ACHIEVED
        .WORD    WRITE    ;EVERYTHING THE
        .WORD    -1000.   ;ABOVE TABLE DID, BUT
        .WORD    WRTBUF   ;IN A CLEANER FORMAT
        .BYTE    3,5
        .WORD    400,ERRTB5,0

```


8.6.5. RH11/RP04 REGISTERS

<u>MNEMONIC</u>	<u>INDEX</u>
RHCS1	0
RHHC	1
RHBA	2
RHDA	3
RHCS2	10
RHDS1	11
RHER1	14
RHAS	16
RHLA	20
RHDB	21
RHR	22
RHOT	23
RHSN	30
RHOF	32
RHCA	34
RHCC	36
RHER2	40
RHER3	42
RHEC1	44
RHEC2	46

8.6.6. COMMANDS PERFORMED BY THE DRIVER

<u>COMMAND</u>	<u>CODE</u>	<u>COMMAND TYPE</u>
NO OPERATION	101	N
UNLOAD	103	N
SEEK	105	P
RECALIRATE	107	P
DRIVE CLEAR	111	N
RELEASE	113	N
OFFSET	115	P
RETURN TO CENTER	117	P
READIN PRESET	121	N
PACK ACKNOWLEDGE	123	N
SEARCH	131	P

GET REGISTER(S)	141	S
SET FORMAT	143	S
SELECT DRIVE	145	S
WRITE CHECK DATA	151	D
WRITE CHECK HEADER AND DATA	153	D
WRITE DATA	161	D
WRITE HEADER & DATA	163	D
READ DATA	171	D
READ HEADER & DATA	173	D

N = HOUSEKEEPING
P = POSITIONING
D = DATA TRANSFER
S = SPECIAL PROVIDED BY THE DRIVER

8.6.7. DPB STATUS/ERROR INDICATOR

THIS INDICATOR WILL INFORM THE USER OF THE RESULTS OF THE REQUEST.
THIS IS ACCOMPLISHED BY SETTING VARIOUS BITS OF THE INDICATOR TO
A ONE.

<u>BIT NO.</u>	<u>MEANING IF ON A "1"</u>
15	ERROR OCCURRED DONE (BIT07=0); BITS 14-10 SPECIFIES TYPE DONE (BIT07=1); BITS 06-03 SPECIFIES TYPE
14(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON AN OFFLINE OR UNSAFE DRIVE
13(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON A DRIVE THAT HAS AN UNLOAD REQUEST IN QUEUE.
12(2)	PERSISTENT UNSAFE CONDITION EXIST.
11(2)	PARITY ERROR OCCURRED
10(2)(4)	FATAL PARITY ERROR. A MASSBUS CLEAR WAS PERFORMED, ALL QUEUES WERE EMPTIED, AND ALL DRVACT'S SET TO THE IDLE STATE

- 9(3)(4) SOFTWARE TIMEOUT OCCURRED ON THIS DRIVE
- 8(4) SOFTWARE TIMEOUT OCCURRED ON ANOTHER DRIVE
- 7 DONE
- 6(2) ERROR OCCURRED DURING AN I/O OPERATION
- 5(2) ERROR OCCURRED DURING AN OPERATION
OTHER THAN I/O.
- 4(2) UNSAFE CONDITION OCCURRED
- 3(2) DRIVE ERROR OCCURRED THAT CAUSED AN
AUTOMATIC "RECALIBRATE" SEQUENCE
- (1) => REQUEST WASN'T PUT IN QUEUE. (RH11/RP04
REGISTERS WERE NOT SAVED)
- (2) => REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER
ISSUED A "DRIVE CLEAR" TO THE DRIVE.
NOTE: ALL RH11/RP04 REGISTERS ARE SAVED
AS PER DPB+14 BEFORE THE "DRIVE CLEAR".
- (3) => REQUEST QUEUE HAS BEEN EMPTIED. THE
DRIVER ISSUED A MASSBUS INIT. ALL
RH11/RP04 REGISTERS FOR THE DRIVE WERE
SAVED AS PER DPB+14 BEFORE THE INIT.
- (4) => A "RECALIBRATE" SHOULD BE ISSUED
BEFORE ANY OTHER COMMAND.

8.6.8. ERROR CALLS MADE BY THE DRIVER.

THERE ARE A FEW ERRORS THAT CAN OCCUR THAT CAN NOT BE INDICATED IN A CPB.

WHEN THIS TYPE OF ERROR IS DETECTED BY THE DRIVER IT WILL MAKE AN ERROR CALL OF THE FORM "ERROR N", WHERE "N" IS THE ERROR NUMBER AND THE ERROR WILL BE AN EMT INSTRUCTION.

N	TYPE	DATA AVAILABLE
-	----	-----
1	ILLEGAL SUBSYSTEM INTERRUPT OCCURRED (SC=0)	*R4= RHCS1'S ADDRESS
2	ILLEGAL DRIVE INTERRUPT OCCURRED	R1= DRIVE NUMBER R3= ATA BIT *R4= RHCS1'S ADDRESS R5= (RHAS) RPERRS =RHDS1 RPERRS+2=RHER1 RPERRS+4=RHER2 RPERRS+6=RHER3
3	MASSBUS CONTROL BUS PARITY ERROR OCCURRED(MCPE=1)	RD.ADR= ADDRESS OF REG. READ RD.WRD= WORD READ
4	CONTROL BUS PARITY ERROR OCCURRED(PAR=1)	WRT.AD= ADDRESS OF REG. WRITTEN WRT.WD= WORD WRITTEN RD.WRD= WORD READ BACK
5	ATTENTION FROM AN OFFLINE, NONEXISTENT OR PREVIOUSLY DETECT UNSAFE DRIVE OCCURRED. NOTE: COULD HAVE BEEN CAUSED BY A BIT PICK OR DROP WHEN READING RHAS.	R1= DRIVE NUMBER R3= ATA BIT *R4= RHCS1'S ADDRESS R5= (RHAS) RPERRS =RHDS1 RPERRS+2=RHER1 RPERRS+4=RHER2 RPERRS+6=RHER3
6	RESERVED	
7.	RESERVED	

* THIS IS THE ACTUAL UNIBUS ADDRESS (177200)

9. PROGRAM DESCRIPTION

THIS PROGRAM CONTAINS FIFTEEN TESTS NUMBERED 0-16 IN OCTAL. TESTS 0-6 WILL READ THE CYLINDER, TRACK, AND SECTOR INFORMATION FROM THE HEADER, USING A "READ HEADER AND DATA" COMMAND, AND THEN CHECK THE INFORMATION FOR VALIDITY. THUS, INSURING THE SEEK OPERATION FUNCTIONS PROPERLY. TEST 7-12 WILL MEASURE THE ROTATIONAL SPEED, THE ONE CYLINDER SEEK, THE AVERAGE SEEK, AND THE MAXIMUM SEEK TIMES TO INSURE THEY ARE ALL WITHIN THE TOLERANCES ALLOWED. TEST 13 AND 14 INSURES THE SECTOR AND TRACK ADDRESSING CIRCUITRY WORKS PROPERLY. TEST 15 VERIFIES THE DATA STORAGE AND RETRIEVAL CAPABILITIES ARE FUNCTIONAL. AND TEST 16 WILL STRESS AND CHECK THE READ/WRITE AND SERVO SYSTEMS.

THE PROGRAM WILL START BY IDENTIFYING ITSELF AND DETERMINING ALL DRIVES THAT ARE AVAILABLE FOR TESTING. THEN BEGINNING WITH THE LOWEST NUMERICAL DRIVE AND PROCEEDING IN SEQUENTIAL ORDER ALL OF THE DRIVES WILL BE TESTED. ONE PASS THROUGH THE TEST SEQUENCE (TEST 0-15) WILL BE PERFORMED ON EACH DRIVE BEFORE MOVING TO THE NEXT DRIVE IN SEQUENCE. THE DRIVE TO BE TESTED WILL BE TYPED AT THE BEGINNING OF EACH PASS, AN "END OF PASS" MESSAGE WILL BE TYPED AT THE COMPLETION OF EACH PASS, AND AN "END OF TEST" MESSAGE WILL BE TYPED AFTER TESTING ALL DRIVES.

REFER TO THE FOLLOWING SECTION FOR A DETAILED DESCRIPTION OF EACH TEST.

TABLE OF CONTENTS

4923	CONTROL SWITCH SETTINGS
4944	OPERATIONAL SWITCH SETTINGS
4955	TRAP CATCHER
4957	STARTING ADDRESSES
4968	BASIC DEFINITIONS
5715	COMMON TAGS
(1)	ERROR POINTER TABLE
6151	START OF PROGRAM
6285	#### TESTS ####
6314	*** SEEK TESTS (0-6) ***
6335	T0 RECAL/SEEK TEST
6357	T1 SEEK/SEEK TEST
6382	T2 INCREMENT/SEEK TEST
6409	T3 STEPPING SEEK TEST
6433	T4 OSCILLATING SEEK TEST
6497	T5 CONVERGING/DIVERGING SEEK TEST
6526	T6 SERVO ADDRESSING LOGIC NOISE GENERATOR
6553	*** TIMING TESTS (7-12) ***
6578	T7 ROTATIONAL SPEED TIMING TEST
6651	T10 ONE CYLINDER SEEK TIMING TEST
6712	T11 AVERAGE SEEK TIMING TEST
6777	T12 MAXIMUM SEEK TIMING TEST
6833	*** ADDRESSING TESTS (13-14) ***
6855	T13 SECTOR ADDRESSING TEST
6903	T14 TRACK ADDRESSING TEST
6949	*** DATA TEST (15) ***
7012	T15 DATA TEST
7086	*** EXERCISE TEST (16) ***
7112	T16 RANDOM ADDRESS AND RANDOM PATTERN TEST (TST16)
7165	END OF PASS ROUTINE
7168	*** SUBROUTINES ***
7169	ERROR HANDLER ROUTINE
7172	TYPEERR - TYPE ERROR ROUTINE
7269	TYPE ROUTINE
7270	BINARY TO OCTAL (ASCII) AND TYPE
7271	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
7272	TTY INPUT ROUTINE
7273	SCOPE HANDLER ROUTINE
7274	SAVE AND RESTORE RD-RS ROUTINES
7275	TRAP DECODER
(3)	TRAP TABLE
7276	SINGLE LENGTH BINARY TO DECIMAL ASCII ROUTINE
7277	DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
7278	TYPE NUMERICAL ASCII STRING SUPPRESS LEADING ZEROS
7279	INTEGER DIVIDE ROUTINE
7280	RANDOM NUMBER GENERATOR ROUTINE
7283	LP.AVL - LINE PRINTER AVAILABLE
7303	ST.CLK - CLOCK STARTUP ROUTINE
7373	LDCHD - LOAD COMMAND
7391	CALL.A - CALL RPO4 DRIVER USING "DPB.A"
7412	CALL.B - CALL RPO4 DRIVER USING "DPB.B"
7443	CALL.C - CALL RPO4 DRIVER USING "DPB.C"
7473	DRYCAL - DRIVER (RPO4) CALL USING "DTADPB"
7538	ERINDX - FORM ERROR INDEX

TABLE OF CONTENTS

7590	STALL - USED TO STALL AFTER A DRIVE FUNCTION IS COMPLETE
7619	TWOMS - STALL FOR 2 MS BETWEEN SEEKS IN TESTS 10- - 12
7641	VERIFY - VALIDATE HEADER ON IMPLIED SEEKS
7669	SRCHOO - INITIALIZE THE DRIVE FOR THE TIMING TESTS
7702	DORTI - RETURN FROM INTERRUPT
7709	STRTRM - START THE TIMERS
7728	COUNT - THIS ROUTINE COUNTS THE ELAPSED TIME
7767	TYPTIM - TYPE TIMES
7836	INCTRK - INCREMENT TRACK NUMBER
7855	INCCYL - INCRMENT CYLINDER NUMBER
7873	FILBUF - FILL BUFFER WITH ADDRESS DATA
7895	CLRBUF - CLEAR BUFFER
7920	CKSCIR - CHECK SECTOR DATA
7984	SETBUF - SET BUFFER TO DATA PATTERN
8004	DATCMP - DATA COMPARE
8060	FILRAM - FILL DATA BUFFER WITH RANDOM PATTERN
8078	RANCK - RANDOM PATTERN BUFFER CHECK
8128	RANPAT - RANDOM PATTERN GENERATOR
8149	RANADR - RANDOM ADDRESS GENERATOR
8204	GETSWR - GET THE CONTROL SWITCH SETTINGS
8226	GETADR - GET BUS ADDRESS AND VECTOR ADDRESS
8290	GETNUM - ROUTINE TO GET A NUMBER
8337	GT.PRM - GET PARAMETERS
8620	CK.OCT -- CHECK FOR OCTAL CHARACTER
8640	CK.DEC -- CHECK FOR DECIMAL CHARACTER
8660	CK.CHR -- CHECK CHARACTER
8694	CK.DIG - CHECK DIGIT
8742	CK.NUM - CHECK NUMBER (OCTAL)
8785	RH11/RP04 DRIVER (REV. 0.9)
8787	ASCIZ MESSAGES
8823	ERROR HEADER (EM) MESSAGES
8839	STATUS/ERROR INDICATOR MESSAGES
8853	DATA HEADER (DT) MESSAGES
8877	DATA TABLE (DT)
8904	DATA FORMAT (DF) TABLE

4920
4921
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
4922
4923
4924
4925
4926
4927
4928
4929
4930
4931
4932
4933
4934
4935
4936
4937
4938
4939
4940
4941
4942
4943
4944
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
4945
4946
4947
4948
4949
4950
4951
4952
4953

.ENABLE ABS AMA
.TITLE MAINDEC-11-DZRPK-B "MECHANICAL AND READ/WRITE TEST"
.*COPYRIGHT (C) 1974,1975
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754
.*
.*PROGRAM BY J. LACEY/C. HESS
.*
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-A2).
.*

.SBTTL CONTROL SWITCH SETTINGS

SWITCH	STATE	USE
15	0	WRITE PACK BEFORE TESTING (TEST 16)
	1	INHIBIT WRITING PACK BEFORE TESTING (TEST 16)
14	0	NO STALL BETWEEN DRIVE FUNCTIONS
	1	STALL AFTER EVERY DRIVE FUNCTION
13	0	USE SPECIFIC STALL TIME
	1	USE RANDOM STALL
12	0	NO INCREMENTING STALL IN TEST 4
	1	DO INCREMENTING STALL IN TEST 4
8	0	DO IMPLIED SEEKS WITH DATA TRANSFERS
	1	DO EXPLICIT SEEKS BEFORE DATA TRANSFERS
7	0	DO "READ HEADER AND DATA" IN TESTS 0-6
	1	DO EXPLICIT SEEKS IN TESTS 0-6
6	0	60 HZ
	1	50 HZ
5	0	RUN WATCHDOG TIMER
	1	INHIBIT WATCHDOG TIMER

.SBTTL OPERATIONAL SWITCH SETTINGS

SWITCH	USE
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR TYPEOUTS
11	INHIBIT ITERATIONS
10	BELL ON ERROR
9	LOOP ON ERROR
8	PRINT ERROR MESSAGE ON LINE PRINTER
7	READ CONTROL SWITCH SETTINGS FROM TTY
6	INHIBIT TIME REPORTS (TESTS 7-12)
5	REPORT ONE ERROR PER SECTOR (TESTS 13 & 14)
4	INHIBIT WRITES (TEST 15)
3	INHIBIT WRITE CHECKS (TEST 15)
2	INHIBIT READ AND SOFTWARE COMPARES (TEST 15)
1	INHIBIT SOFTWARE COMPARES (TEST 15)
0	PERFORM READ AFTER WRITE CHECK ERROR (TEST 15)

```

4954
4955      (1)      .SBTTL TRAP CATCHER
         (1)
         (1)      000000      .=0
         (1)      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2.HALT"
         (1)      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
         (1)      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
4956
4957      (1)      .SBTTL STARTING ADDRESSES
         (1)      .=200
         (1)      ;*200 = NORMAL START
4958      000200      000137      004120      JMP      @#START1
4959      ;*204 = SELECT OPERATING PARAMETERS
4960      000204      000137      004210      JMP      @#START2
4961      ;*210 = SELECT RH11/RP04 ADDRESSES
4962      000210      000137      004110      JMP      @#START3
4963      ;*214 = COMBINATION OF 204 AND 210
4964      000214      000137      004200      JMP      @#START4
4965
4966
4967      (1)      .SBTTL BASIC DEFINITIONS
         (1)
         (1)      ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
         (1)      001100      STACK= 1100
         (1)      .EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
         (1)      .EQUI IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
         (1)      177776      PS= 177776      ;;PROCESSOR STATUS WORD
         (1)      .EQUIV PS,PSW
         (1)      177774      STKLMT= 177774      ;;STACK LIMIT REGISTER
         (1)      177772      PIRQ= 177772      ;;PROGRAM INTERRUPT REQUEST REGISTER
         (1)      177570      SWR= 177570      ;;SWITCH REGISTER
         (1)      177570      DISPLAY=SWR
         (1)
         (1)      ;*GENERAL PURPOSE REGISTER DEFINITIONS
         (1)      000000      R0= %0      ;;GENERAL REGISTER
         (1)      000001      R1= %1      ;;GENERAL REGISTER
         (1)      000002      R2= %2      ;;GENERAL REGISTER
         (1)      000003      R3= %3      ;;GENERAL REGISTER
         (1)      000004      R4= %4      ;;GENERAL REGISTER
         (1)      000005      R5= %5      ;;GENERAL REGISTER
         (1)      000006      R6= %6      ;;GENERAL REGISTER
         (1)      000007      R7= %7      ;;GENERAL REGISTER
         (1)      .EQUIV R6,SP      ;;STACK POINTER
         (1)      .EQUIV R7,PC      ;;PROGRAM COUNTER
         (1)
         (1)      ;*PRIORITY LEVEL DEFINITIONS
         (1)      000000      PR0= 0      ;;PRIORITY LEVEL 0
         (1)      000040      PR1= 40      ;;PRIORITY LEVEL 1
         (1)      000100      PR2= 100      ;;PRIORITY LEVEL 2
         (1)      000140      PR3= 140      ;;PRIORITY LEVEL 3
         (1)      000200      PR4= 200      ;;PRIORITY LEVEL 4
         (1)      000240      PR5= 240      ;;PRIORITY LEVEL 5
         (1)      000300      RP6= 300      ;;PRIORITY LEVEL 6

```

```

(1)          000340          PR7=   340          ;;PRIORITY LEVEL 7
(1)
(1)          .:"SWITCH REGISTER" SWITCH DEFINITIONS
(1)          100000          SW15=  100000
(1)          040000          SW14=   40000
(1)          020000          SW13=   20000
(1)          010000          SW12=   10000
(1)          004000          SW11=   4000
(1)          002000          SW10=   2000
(1)          001000          SW09=   1000
(1)          000400          SW08=   400
(1)          000200          SW07=   200
(1)          000100          SW06=   100
(1)          000040          SW05=   40
(1)          000020          SW04=   20
(1)          000010          SW03=   10
(1)          000004          SW02=    4
(1)          000002          SW01=    2
(1)          000001          SW00=    1
(1)          .EQUIV SW09, SW9
(1)          .EQUIV SW08, SW8
(1)          .EQUIV SW07, SW7
(1)          .EQUIV SW06, SW6
(1)          .EQUIV SW05, SW5
(1)          .EQUIV SW04, SW4
(1)          .EQUIV SW03, SW3
(1)          .EQUIV SW02, SW2
(1)          .EQUIV SW01, SW1
(1)          .EQUIV SW00, SW0
(1)
(1)          .:DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1)          100000          BIT15=  100000
(1)          040000          BIT14=   40000
(1)          020000          BIT13=   20000
(1)          010000          BIT12=   10000
(1)          004000          BIT11=   4000
(1)          002000          BIT10=   2000
(1)          001000          BIT09=   1000
(1)          000400          BIT08=   400
(1)          000200          BIT07=   200
(1)          000100          BIT06=   100
(1)          000040          BIT05=   40
(1)          000020          BIT04=   20
(1)          000010          BIT03=   10
(1)          000004          BIT02=    4
(1)          000002          BIT01=    2
(1)          000001          BIT00=    1
(1)          .EQUIV BIT09, BIT9
(1)          .EQUIV BIT08, BIT8
(1)          .EQUIV BIT07, BIT7
(1)          .EQUIV BIT06, BIT6
(1)          .EQUIV BIT05, BIT5
(1)          .EQUIV BIT04, BIT4
(1)          .EQUIV BIT03, BIT3

```



```

(1)          .EQUIV BIT02,BIT2
(1)          .EQUIV BIT01,BIT1
(1)          .EQUIV BIT00,BIT0
(1)
(1)          : *BASIC "CPU" TRAP VECTOR ADDRESSES
(1)          000004 ERRVEC= 4          :: TIME OUT AND OTHER ERRORS
(1)          000010 RESVEC= 10         :: RESERVED AND ILLEGAL INSTRUCTIONS
(1)          000014 TBITVEC=14         :: "T" BIT
(1)          000014 TRTVEC= 14         :: TRACE TRAP
(1)          000014 BPTVEC= 14         :: BREAKPOINT TRAP (BPT)
(1)          000020 IOTVEC= 20         :: INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1)          000024 PWRVEC= 24         :: POWER FAIL
(1)          000030 EMTVEC= 30         :: EMULATOR TRAP (EMT) **ERROR**
(1)          000034 TRAPVEC=34         :: "TRAP" TRAP
(1)          000060 TKVEC= 60          :: TTY KEYBOARD VECTOR
(1)          000064 TPVEC= 64          :: TTY PRINTER VECTOR
(1)          000240 PIRQVEC=240        :: PROGRAM INTERRUPT REQUEST VECTOR

```

```

4970      .OP CODE DEFINITIONS
4971      000101 NOOP=101
4972      000103 UNLOAD=103
4973      000105 SEEK=105
4974      000107 RECAL=107
4975      000111 DRVCLP=111
4976      000113 RELEASE=113
4977      000115 OFFSET=115
4978      000117 RTC=117
4979      000121 READIN=121
4980      000123 PACK=123
4981      000131 SEARCH=131
4982      000151 WRCKD=151
4983      000153 WRCKHD=153
4984      000161 WRITE=161
4985      000163 WRTHD=163
4986      000171 READ=171
4987      000173 READHD=173
4988      000141 GETREG=141
4989      000143 SETFORM=143
4990      000145 SELDRV=145

4991      ;OTHER EQUATES
4992
4993
4994      165000 TRCKWC = -(256.*22.)
4995      177400 SCTRWC = -256.
4996      010000 FMT22=10000
4997

```

```

:WORD COUNT FOR TRACK
:WORD COUNT FOR SECTOR
:FORMAT 22 BIT

```


(3)	001214	000000		CNTRLC: .WORD	0		; CONTROL "C" FLAG
(3)	001216	000000		BUSADR: .WORD	0		; GET ADDRESSES FROM THE TTY FLAG (0=NO, -1=YES)
(3)	001220	000000		LPTAVL: .WORD	0		; LPT AVAILABLE STATUS (0=NO, 1=YES)
(3)	001222	000000		DRVSEL: .WORD	0		; DRIVES SELECTED FOR TESTING
(3)	001224	037777		TSTNMS: .WORD	37777		; RUN TESTS 0-15
(3)	001226	000000		CLKSTA: .WORD	0		; CLOCK STATUS (0=NO CLOCK, +1=KW11-P, AND -1=KW11-L)
(3)	001230	000020		TICKMS: .WORD	↑D16		; 16 MILLISECONDS PER CLOCK TICK
(3)	001232	040432		TICKUS: .WORD	↑D16666		; 16666 MICROSECONDS PER CLOCK TICK
(3)	001234	000000		BYPASS: .WORD	?		
(3)	001236	000000		CHKDRV: .WORD	0		; DRIVE UNDER TEST
(3)	001240	000000		DRVMASK: .WORD	0		; DRIVE MASK BIT
(3)	001242	000000		SVSTAT: .WORD	?		; STATUS/ERROR INDICATOR IS SAVED HERE ON AN ERROR
(3)	001244	000000		CYL.RD: .WORD	0		; CYLINDER READ
(3)	001246	000000		TRK.RD: .WORD	0		; TRACK READ
(3)	001250	000600		SEC.RD: .WORD	0		; SECTOR READ
(3)	001252	000000		CYL.DS: .WORD	0		; CYLINDER DESIRED
(3)	001254	000000		SEC.DS: .WORD	0		; SECTOR DESIRED
(3)	001256	000000		TRK.DS: .WORD	0		; TRACK DESIRED
(3)	001260	000000		TIM.UP: .WORD	0		; MINIMUM TIME
(3)	001262	000000		.WORD	0		; NUMBER OF COUNTS BELOW MIN. LIMIT
(3)	001264	000000		.WORD	0		; MAXIMUM TIME
(3)	001266	000000		.WORD	0		; NUMBER OF COUNTS ABOVE MAX. LIMIT
(3)	001270	000000	000000	.WORD	0,0		; TOTAL TIME OF ALL SEEKS
(3)	001274	000000		.WORD	0		; NUMBER OF SEEKS PERFORMED
(3)	001276	000000		TIM.DN: .WORD	0		; MINIMUM TIME
(3)	001300	000000		.WORD	0		; NUMBER OF COUNTS BELOW MIN. LIMIT
(3)	001302	000000		.WORD	0		; MAXIMUM TIME
(3)	001304	000000		.WORD	0		; NUMBER OF COUNTS ABOVE MAX. LIMIT
(3)	001306	000000	000000	.WORD	0,0		; TOTAL TIME OF ALL SEEKS
(3)	001312	000000		.WORD	0		; NUMBER OF SEEKS PERFORMED
(3)	001314	000000		TIM.PT: .WORD	0		; POINTS TO TABLE OF TIMES
(3)	001316	000000		WCEFLG: .WORD	0		; FATAL WRITE CHECK ERROR FLAG (TST 15)
(3)	001320	000000		STALLO: .WORD	0		; VARIABLE STALL (TEST 4)
(3)	001322	000000	000000	SVADR: .WORD	0,0		; SAVE DISK ADDRESS (TST16)
(3)				: CONSTANTS STORAGE			
(3)	001326	176700		RH.ADR: .WORD	176700		; RH11/RP04 CONTROL AND STATUS REG. #1
(3)	001330	000254	000240	RHVEC: .WORD	254, 5*32.		; RH11/RP04 VECTOR AND PRIORITY
(3)	001334	000104	000106	PKV: .WORD	104, 106		; KW11-P VECTOR ADDRESS
(3)	001340	172540		PKCS: .WORD	172540		; KW11-P CONTROL AND STATUS REG.
(3)	001342	172542		PKB: .WORD	172542		; KW11-P COUNT SET BUFFER
(3)	001344	172544		PKC: .WORD	172544		; KW11-P COUNTER
(3)	001346	000100	000102	LKV: .WORD	100, 102		; KW11-L VECTOR ADDRESS
(3)	001352	177546		LKS: .WORD	177546		; KW11-L STATUS REGISTER
(3)	001354	177564		TPS: .WORD	177564		; TTY PRINTER STATUS
(3)	001356	177566		TPB: .WORD	177566		; TTY PRINTER BUFFER
(3)	001360	177514		LPS: .WORD	177514		; LINE PRINTER STATUS
(3)	001362	177516		LPB: .WORD	177516		; LINE PRINTER BUFFER
(3)	001364	000012		STALL1: .WORD	↑D10		; 10 MILLISECONDS STALL (TEST 0-6)
(3)	001366	000012		STALL2: .WORD	↑D10		; 10 MILLISECONDS STALL (TEST 13-16)
(3)	001370	000031		MXSTAL: .WORD	↑D25		; MAX. INCREMENTING STALL ALLOWED IN TEST 4
(3)	001372	006		ERR.CT: .BYTE	6		; NUMBER OF ERRORS ALLOWED IN TESTS 13-16

```

(3)                                ;BEFORE GOING TO THE NEXT TEST
(3) 001373 000                    ;RESERVED
(3)                                .BYTE 0
(3)                                ;BIT TABLE
(3)                                BITS:
(3) 001374 000001                .WORD BIT00
(3) 001376 000002                .WORD BIT01
(3) 001400 000004                .WORD BIT02
(3) 001402 000010                .WORD BIT03
(3) 001404 000020                .WORD BIT04
(3) 001406 000040                .WORD BIT05
(3) 001410 000100                .WORD BIT06
(3) 001412 000200                .WORD BIT07
(3) 001414 000400                .WORD BIT08
(3) 001416 001000                .WORD BIT09
(3) 001420 002000                .WORD BIT10
(3) 001422 004000                .WORD BIT11
(3) 001424 010000                .WORD BIT12
(3) 001426 020000                .WORD BIT13
(3) 001430 040000                .WORD BIT14
(3) 001432 100000                .WORD BIT15
(3)
(3)                                ;COMMON STORAGE FOR TEST PARAMETER
(3) 001434 000000                PRM: .WORD 0
(3) 001436 000000                RPT: .WORD 0                                ;REPEAT COUNTS FOR ALL TESTS
(3) 001440 000000                FC: .WORD 0                                ;FIRST CYLINDER
(3) 001442 000000                LC: .WORD 0                                ;LAST CYLINDER
(3) 001444 000000                IC: .WORD 0                                ;INCREMENT CYLINDER
(3) 001446 000000                FT: .WORD 0                                ;FIRST TRACK
(3) 001450 000000                LT: .WORD 0                                ;LAST TRACK
(3) 001452 000000                IT: .WORD 0                                ;INCREMENT TRACK
(3) 001454 000000                FS: .WORD 0                                ;FIRST SECTOR
(3) 001456 000000                LS: .WORD 0                                ;LAST SECTOR
(3) 001460 000000                PAT: .WORD 0                                ;PATTERN CODE
(3)
(3)                                ;TABLE OF PARAMETER POINTERS
(3) 001462 002062                PRMPT: .WORD PRM0
(3) 001464 002074                .WORD PRM1
(3) 001466 002116                .WORD PRM2
(3) 001470 002134                .WORD PRM3
(3) 001472 002152                .WORD PRM4
(3) 001474 002170                .WORD PRM5
(3) 001476 002206                .WORD PRM6
(3) 001500 002224                .WORD PRM7
(3) 001502 002236                .WORD PRM10
(3) 001504 002242                .WORD PRM11
(3) 001506 002246                .WORD PRM12
(3) 001510 002252                .WORD PRM13
(3) 001512 002262                .WORD PRM14
(3) 001514 002272                .WORD PRM15
(3) 001516 002320                .WORD PRM16
(3)
(3)                                ;PARAMETER UPPER LIMIT
(3) 001520 077777                PRMLMT: .WORD ↑D32767                                ;"R"

```

(3)	001522	000632	.WORD	↑D410	:"FC"
(3)	001524	000632	.WORD	↑D410	:"LC"
(3)	001526	000632	.WORD	↑D410	:"IC"
(3)	001530	000022	.WORD	↑D18	:"FT"
(3)	001532	000022	.WORD	↑D18	:"LT"
(3)	001534	000022	.WORD	↑D18	:"IT"
(3)	001536	000025	.WORD	↑D21	:"FS"
(3)	001540	000025	.WORD	↑D21	:"LS"
(3)	001542	177777	.WORD	177777	:"PAT"

:TABLE OF MESSAGE POINTERS

(3)	001544	034376	PRNMSG: .WORD	MSG.R
(3)	001546	034400	.WORD	MSG.FC
(3)	001550	034403	.WORD	MSG.LC
(3)	001552	034406	.WORD	MSG.IC
(3)	001554	034411	.WORD	MSG.FT
(3)	001556	034414	.WORD	MSG.LT
(3)	001560	034417	.WORD	MSG.IT
(3)	001562	034422	.WORD	MSG.FS
(3)	001564	034425	.WORD	MSG.LS

:STATUS/ERROR INDICATOR MESSAGES POINTER TABLE

(3)	001566	036021	STATBL: .WORD	MSG814	:OFFLINE OR UNSAFE DRIVE REQUESTED
(3)	001570	036063	.WORD	MSG813	:UNLOAD DRIVE REQUESTED
(3)	001572	036114	.WORD	MSG812	:PERSISTENT UNSAFE
(3)	001574	036136	.WORD	MSG811	:PARITY ERROR OCCURRED
(3)	001576	036164	.WORD	MSG810	:FATAL PARITY ERROR
(3)	001600	036207	.WORD	MSG809	:SOFTWARE TIMEOUT ON THIS DRIVE
(3)	001602	036246	.WORD	MSG808	:SOFTWARE TIMEOUT ON ANOTHER DRIVE
(3)	001604	036310	.WORD	MSG806	:ERROR OCCURRED DURING I/O OPERATION
(3)	001606	036354	.WORD	MSG805	:ERROR OCCURRED DURING NON-I/O OPERATION
(3)	001610	036424	.WORD	MSG804	:UNSAFE OCCURRED
(3)	001612	036444	.WORD	MSG803	:AUTOMATIC RECALIBRATE SEQUENCE OCCURRED

:DEFAULT VALUES OF TESTS PARAMETERS

(3)	001614	000225	000012	000632	DFLT: .WORD	225,10.,410.,0,0 ;RECAL/SEEK (T0)
(3)	001622	000000	000000		.WORD	677,100.,0,128.,0,0,0,0,0 ;SEEK/SEEK (T1)
(3)	001626	000677	000144	000000		
(3)	001634	000200	000000	000000		
(3)	001642	000000	000000	000000		
(3)	001650	000237	000001	000000	.WORD	237,1,0,410.,1,0,0 ;INCREMENT SEEK (T2)
(3)	001656	000632	000001	000000		
(3)	001664	000000				
(3)	001666	000237	000001	000000	.WORD	237,1,0,256.,1,0,0 ;STEPPING SEEK (T3)
(3)	001674	000400	000001	000000		
(3)	001702	000000				
(3)	001704	000237	000001	000000	.WORD	237,1,0,410.,1,0,0 ;OSCILLATING SEEK (T4)
(3)	001712	000632	000001	000000		
(3)	001720	000000				
(3)	001722	000237	000001	000000	.WORD	237,1,0,410.,1,0,0 ;CONVERGING/DIVERGING SEEK (T5)
(3)	001730	000632	000001	000000		
(3)	001736	000000				
(3)	001740	000237	000001	000000	.WORD	237,1,0,410.,1,0,0 ;SERVO ADDRESSING LOGIC NOISE (T6)
(3)	001746	000632	000001	000000		

```

(3) 001754 000000
(3) 001756 000223 000001 000000 .WORD 223,1,0,0,0 ;ROTATIONAL SPEED TIMING TEST (T7)
(3) 001764 000000 000000
(3) 001770 000001 000001 .WORD 1,1 ;ONE CYLINDER SEEK TIMING TEST (T10)
(3) 001774 000001 000001 .WORD 1,1 ;AVERAGE SEEK TIMING TEST (T11)
(3) 002000 000001 000001 .WORD 1,1 ;MAXIMUM SEEK TIMEING TEST (T12)
(3) 002004 000023 000001 000000 .WORD 23,1,0,0 ;SECTOR ADDRESSING TEST (T13)
(3) 002012 000000
(3) 002014 000203 000001 000000 .WORD 203,1,0,0 ;TRACK ADDRESSING TEST (T14)
(3) 002022 000000
(3) 002024 001777 000001 000000 .WORD 1777,1,0,410.,64.,0.18.,1,1,0,177777 ;DATA TEST (T15)
(3) 002032 000632 000100 000000
(3) 002040 000022 000001 000001
(3) 002046 000000 177777
(3) 002052 000007 047040 000000 .WORD 7,20000.,0,410. ;EXERCISER (T16)
(3) 002060 000632

```

;PARAMETER TABLES

;RECAL/SEEK (T0)

```

(3) 002062 000225 PRM0: .WORD 225
(3) 002064 000012 RPT0: .WORD 10.
(3) 002066 000632 LCO: .WORD 410.
(3) 002070 000000 FTO: .WORD 0
(3) 002072 000000 FSO: .WORD 0

```

;SEEK/SEEK (T1)

```

(3) 002074 000677 PRM1: .WORD 677
(3) 002076 000144 RPT1: .WORD 100.
(3) 002100 000000 FC1: .WORD 0
(3) 002102 000200 LC1: .WORD 128.
(3) 002104 000000 IC1: .WORD 0
(3) 002106 000000 FT1: .WORD 0
(3) 002110 000000 LT1: .WORD 0
(3) 002112 000000 FS1: .WORD 0
(3) 002114 000000 LS1: .WORD 0

```

;INCREMENT SEEK (T2)

```

(3) 002116 000237 PRM2: .WORD 237
(3) 002120 000001 RPT2: .WORD 1
(3) 002122 000000 FC2: .WORD 0
(3) 002124 000632 LC2: .WORD 410.
(3) 002126 000001 IC2: .WORD 1
(3) 002130 000000 FT2: .WORD 0
(3) 002132 000000 FS2: .WORD 0

```

;STEPPING SEEK (T3)

```

(3) 002134 000237 PRM3: .WORD 237
(3) 002136 000001 RPT3: .WORD 1
(3) 002140 000000 FC3: .WORD 0
(3) 002142 000400 LC3: .WORD 256.
(3) 002144 000001 IC3: .WORD 1
(3) 002146 000000 FT3: .WORD 0
(3) 002150 000000 FS3: .WORD 0

```

```

(3)
(3)
(3) 002152 000237
(3) 002154 000001
(3) 002156 000000
(3) 002160 000632
(3) 002162 000001
(3) 002164 000000
(3) 002166 000000
(3)
(3)
(3) 002170 000237
(3) 002172 000001
(3) 002174 000000
(3) 002176 000632
(3) 002200 000001
(3) 002202 000000
(3) 002204 000000
(3)
(3)
(3) 002206 000237
(3) 002210 000001
(3) 002212 000000
(3) 002214 000632
(3) 002216 000001
(3) 002220 000000
(3) 002222 000000
(3)
(3)
(3) 002224 000223
(3) 002226 000001
(3) 002230 000000
(3) 002232 000000
(3) 002234 000000
(3)
(3)
(3) 002236 000001
(3) 002240 000001
(3)
(3)
(3) 002242 000001
(3) 002244 000001
(3)
(3)
(3) 002246 000001
(3) 002250 000001
(3)
(3)
(3) 002252 000023
(3) 002254 000001
(3) 002256 000000
(3) 002260 000000
(3)
(3)

```

: OSCILLATING SEEK (T4)
PRM4: .WORD 237
RPT4: .WORD 1
FC4: .WORD 0
LC4: .WORD 410.
IC4: .WORD 1
FT4: .WORD 0
FS4: .WORD 0

: CONVERGING/DIVERGING SEEK (T5)
PRM5: .WORD 237
RPT5: .WORD 1
FC5: .WORD 0
LC5: .WORD 410.
IC5: .WORD 1
FT5: .WORD 0
FS5: .WORD 0

: SERVO ADDRESSING LOGIC NOISE GENERATOR (T6)
PRM6: .WORD 237
RPT6: .WORD 1
FC6: .WORD 0
LC6: .WORD 410.
IC6: .WORD 1
FT6: .WORD 0
FS6: .WORD 0

: ROTATIONAL SPEED TIMING TEST (T7)
PRM7: .WORD 223
RPT7: .WORD 1
FC7: .WORD 0
FT7: .WORD 0
FS7: .WORD 0

: ONE CYLINDER SEEK TIMING TEST (T10)
PRM10: .WORD 1
RPT10: .WORD 1

: AVERAGE SEEK TIMING TEST (T11)
PRM11: .WORD 1
RPT11: .WORD 1

: MAXIMUM SEEK TIMING TEST (T12)
PRM12: .WORD 1
RPT12: .WORD 1

: SECTOR ADDRESSING TEST (T13)
PRM13: .WORD 23
RPT13: .WORD 1
FC13: .WORD 0
FT13: .WORD 0

; TRACK ADDRESSING TEST (T14)


```

(3) 002262 000203 PRM14: .WORD 203
(3) 002264 000001 RPT14: .WORD 1
(3) 002266 000000 FC14: .WORD 0
(3) 002270 000000 FS14: .WORD 0
(3)
(3) ;DATA TEST (T15)
(3) 002272 001777 PRM15: .WORD 1777
(3) 002274 000001 RPT15: .WORD 1
(3) 002276 000000 FC15: .WORD 0
(3) 002300 000632 LC15: .WORD 410.
(3) 002302 000100 IC15: .WORD 64.
(3) 002304 000000 FT15: .WORD 0
(3) 002306 000022 LT15: .WORD 18.
(3) 002310 000001 IT15: .WORD 1
(3) 002312 000001 FS15: .WORD 1
(3) 002314 000000 LS15: .WORD 0
(3) 002316 177777 PTRN15: .WORD 177777
(3)
(3) ;EXERCISER (T16)
(3) 002320 000007 PRM16: .WORD 7
(3) 002322 047040 RPT16: .WORD 20000.
(3) 002324 000000 FC16: .WORD 0
(3) 002326 000632 LC16: .WORD 410.
(3)
(3) ;SEEK TIMING LIMITS
(3)
(3) 002330 034516 T7A: .WORD MSG7
(3) 002332 000000 .WORD 0
(3) 002334 003142 .WORD 1634. ;(16.67-2%)*2
(3) 002336 003244 .WORD 1700. ;(16.67+2%)*2
(3)
(3) 002340 034516 T7B: .WORD MSG7
(3) 002342 000000 .WORD 0
(3) 002344 003131 .WORD 1625. ;(16.67-2.5%)*2
(3) 002346 003255 .WORD 1709. ;(16.67+2.5%)*2
(3)
(3) 002350 034550 T10: .WORD MSG10A
(3) 002352 034631 .WORD MSG10B
(3) 002354 000454 .WORD 300. ;(7-4)*2
(3) 002356 001750 .WORD 1000. ;(7+3)*2
(3)
(3) 002360 034664 T11: .WORD MSG11A
(3) 002362 034732 .WORD MSG11B
(3) 002364 005050 .WORD 2600. ;(28-2)*2
(3) 002366 005670 .WORD 3000. ;(28+2)*2
(3)
(3) 002370 034753 T12: .WORD MSG12A
(3) 002372 035021 .WORD MSG12B
(3) 002374 010770 .WORD 4600. ;(50-4)*2
(3) 002376 012120 .WORD 5200. ;(50+2)*2
(3)
(3) 002400 002440 PAT.PT: .WORD PAT0 ;TABLE OF POINTERS WHICH POINT TO THE
(3) 002402 002500 .WORD PAT1 ;PATTERNS USED BY THE DATA TEST
(3) 002404 002540 .WORD PAT2

```

(3)	002406	002600	.WORD	PAT3
(3)	002410	002640	.WORD	PAT4
(3)	002412	002700	.WORD	PAT5
(3)	002414	002740	.WORD	PAT6
(3)	002416	003000	.WORD	PAT7
(3)	002420	003040	.WORD	PAT8
(3)	002422	003100	.WORD	PAT9
(3)	002424	003140	.WORD	PAT10
(3)	002426	003200	.WORD	PAT11
(3)	002430	003240	.WORD	PAT12
(3)	002432	003300	.WORD	PAT13
(3)	002434	003340	.WORD	PAT14
(3)	002436	003400	.WORD	PAT15

; PATTERNS 0 THRU 15

(3)	002440	165555	PAT0: .WORD	165555	; PATTERN 0
(3)	002442	133333	.WORD	133333	
(3)	002444	165555	.WORD	165555	
(3)	002446	133333	.WORD	133333	
(3)	002450	165555	.WORD	165555	
(3)	002452	133333	.WORD	133333	
(3)	002454	165555	.WORD	165555	
(3)	002456	133333	.WORD	133333	
(3)	002460	165555	.WORD	165555	
(3)	002462	133333	.WORD	133333	
(3)	002464	165555	.WORD	165555	
(3)	002466	133333	.WORD	133333	
(3)	002470	165555	.WORD	165555	
(3)	002472	133333	.WORD	133333	
(3)	002474	165555	.WORD	165555	
(3)	002476	133333	.WORD	133333	

(3)	002500	000001	PAT1: .WORD	000001	; PATTERN 1
(3)	002502	000003	.WORD	000003	
(3)	002504	000007	.WORD	000007	
(3)	002506	000017	.WORD	000017	
(3)	002510	000037	.WORD	000037	
(3)	002512	000077	.WORD	000077	
(3)	002514	000177	.WORD	000177	
(3)	002516	000377	.WORD	000377	
(3)	002520	000777	.WORD	000777	
(3)	002522	001777	.WORD	001777	
(3)	002524	003777	.WORD	003777	
(3)	002526	007777	.WORD	007777	
(3)	002530	017777	.WORD	017777	
(3)	002532	037777	.WORD	037777	
(3)	002534	077777	.WORD	077777	
(3)	002536	177777	.WORD	177777	

(3)	002540	177776	PAT2: .WORD	177776	; PATTERN 2
(3)	002542	177774	.WORD	177774	
(3)	002544	177770	.WORD	177770	
(3)	002546	177760	.WORD	177760	

(3)	002550	177740	.WORD	177740	
(3)	002552	177700	.WORD	177700	
(3)	002554	177600	.WORD	177600	
(3)	002556	177400	.WORD	177400	
(3)	002560	177000	.WORD	177000	
(3)	002562	176000	.WORD	176000	
(3)	002564	174000	.WORD	174000	
(3)	002566	170000	.WORD	170000	
(3)	002570	160000	.WORD	160000	
(3)	002572	140000	.WORD	140000	
(3)	002574	100000	.WORD	100000	
(3)	002576	000000	.WORD	000000	
(3)					
(3)	002600	000000	PAT3: .WORD	000000	;PATTERN 3
(3)	002602	000000	.WORD	000000	
(3)	002604	000000	.WORD	000000	
(3)	002606	177777	.WORD	177777	
(3)	002610	177777	.WORD	177777	
(3)	002612	177777	.WORD	177777	
(3)	002614	000000	.WORD	000000	
(3)	002616	000000	.WORD	000000	
(3)	002620	177777	.WORD	177777	
(3)	002622	177777	.WORD	177777	
(3)	002624	000000	.WORD	000000	
(3)	002626	177777	.WORD	177777	
(3)	002630	000000	.WORD	000000	
(3)	002632	177777	.WORD	177777	
(3)	002634	000000	.WORD	000000	
(3)	002636	177777	.WORD	177777	
(3)					
(3)	002640	000000	PAT4: .WORD	000000	;PATTERN 4
(3)	002642	010421	.WORD	010421	
(3)	002644	021042	.WORD	021042	
(3)	002646	031463	.WORD	031463	
(3)	002650	042104	.WORD	042104	
(3)	002652	052525	.WORD	052525	
(3)	002654	063146	.WORD	063146	
(3)	002656	073567	.WORD	073567	
(3)	002660	104210	.WORD	104210	
(3)	002662	114631	.WORD	114631	
(3)	002664	125252	.WORD	125252	
(3)	002666	135673	.WORD	135673	
(3)	002670	146314	.WORD	146314	
(3)	002672	156735	.WORD	156735	
(3)	002674	167356	.WORD	167356	
(3)	002676	177777	.WORD	177777	
(3)					
(3)	002700	052525	PAT5: .WORD	052525	;PATTERN 5
(3)	002702	052525	.WORD	052525	
(3)	002704	052525	.WORD	052525	
(3)	002706	125252	.WORD	125252	
(3)	002710	125252	.WORD	125252	
(3)	002712	125252	.WORD	125252	
(3)	002714	052525	.WORD	052525	

(3)	002716	052525		.WORD	052525	
(3)	002720	125252		.WORD	125252	
(3)	002722	125252		.WORD	125252	
(3)	002724	052525		.WORD	052525	
(3)	002726	125252		.WORD	125252	
(3)	002730	052525		.WORD	052525	
(3)	002732	125252		.WORD	125252	
(3)	002734	052525		.WORD	052525	
(3)	002736	125252		.WORD	125252	
(3)						
(3)	002740	007417	PAT6:	.WORD	007417	;PATTERN 6
(3)	002742	007417		.WORD	007417	
(3)	002744	007417		.WORD	007417	
(3)	002746	170360		.WORD	170360	
(3)	002750	170360		.WORD	170360	
(3)	002752	170360		.WORD	170360	
(3)	002754	007417		.WORD	007417	
(3)	002756	007417		.WORD	007417	
(3)	002760	170360		.WORD	170360	
(3)	002762	170360		.WORD	170360	
(3)	002764	007417		.WORD	007417	
(3)	002766	170360		.WORD	170360	
(3)	002770	007417		.WORD	007417	
(3)	002772	170360		.WORD	170360	
(3)	002774	007417		.WORD	007417	
(3)	002776	170360		.WORD	170360	
(3)						
(3)	003000	026455	PAT7:	.WORD	026455	;PATTERN 7
(3)	003002	026455		.WORD	026455	
(3)	003004	026455		.WORD	026455	
(3)	003006	151322		.WORD	151322	
(3)	003010	151322		.WORD	151322	
(3)	003012	151322		.WORD	151322	
(3)	003014	026455		.WORD	026455	
(3)	003016	026455		.WORD	026455	
(3)	003020	151322		.WORD	151322	
(3)	003022	151322		.WORD	151322	
(3)	003024	026455		.WORD	026455	
(3)	003026	151322		.WORD	151322	
(3)	003030	026455		.WORD	026455	
(3)	003032	151322		.WORD	151322	
(3)	003034	026455		.WORD	026455	
(3)	003036	151322		.WORD	151322	
(3)						
(3)	003040	165555	PAT8:	.WORD	165555	;PATTERN 8
(3)	003042	133333		.WORD	133333	
(3)	003044	165555		.WORD	165555	
(3)	003046	133333		.WORD	133333	
(3)	003050	165555		.WORD	165555	
(3)	003052	133333		.WORD	133333	
(3)	003054	165555		.WORD	165555	
(3)	003056	133333		.WORD	133333	
(3)	003060	165555		.WORD	165555	
(3)	003062	133333		.WORD	133333	

(3)	003064	165555	.WORD	165555
(3)	003066	133333	.WORD	133333
(3)	003070	165555	.WORD	165555
(3)	003072	133333	.WORD	133333
(3)	003074	165555	.WORD	165555
(3)	003076	133333	.WORD	133333

(3)	003100	000001	PAT9: .WORD	000001	;PATTERN 9
(3)	003102	000002	.WORD	000002	
(3)	003104	000004	.WORD	000004	
(3)	003106	000010	.WORD	000010	
(3)	003110	000020	.WORD	000020	
(3)	003112	000040	.WORD	000040	
(3)	003114	000100	.WORD	000100	
(3)	003116	000200	.WORD	000200	
(3)	003120	000400	.WORD	000400	
(3)	003122	001000	.WORD	001000	
(3)	003124	002000	.WORD	002000	
(3)	003126	004000	.WORD	004000	
(3)	003130	010000	.WORD	010000	
(3)	003132	020000	.WORD	020000	
(3)	003134	040000	.WORD	040000	
(3)	003136	100000	.WORD	100000	

(3)	003140	177776	PAT10: .WORD	177776	;PATTERN 10
(3)	003142	177775	.WORD	177775	
(3)	003144	177773	.WORD	177773	
(3)	003146	177767	.WORD	177767	
(3)	003150	177757	.WORD	177757	
(3)	003152	177737	.WORD	177737	
(3)	003154	177677	.WORD	177677	
(3)	003156	177577	.WORD	177577	
(3)	003160	177377	.WORD	177377	
(3)	003162	176777	.WORD	176777	
(3)	003164	175777	.WORD	175777	
(3)	003166	173777	.WORD	173777	
(3)	003170	167777	.WORD	167777	
(3)	003172	157777	.WORD	157777	
(3)	003174	137777	.WORD	137777	
(3)	003176	077777	.WORD	077777	

(3)	003200	172666	PAT11: .WORD	172666	;PATTERN 11
(3)	003202	155555	.WORD	155555	
(3)	003204	172666	.WORD	172666	
(3)	003206	155555	.WORD	155555	
(3)	003210	172666	.WORD	172666	
(3)	003212	155555	.WORD	155555	
(3)	003214	172666	.WORD	172666	
(3)	003216	155555	.WORD	155555	
(3)	003220	172666	.WORD	172666	
(3)	003222	155555	.WORD	155555	
(3)	003224	172666	.WORD	172666	
(3)	003226	155555	.WORD	155555	
(3)	003230	172666	.WORD	172666	

(3)	003232	155555		.WORD	155555	
(3)	003234	172666		.WORD	172666	
(3)	003236	155555		.WORD	155555	
(3)						
(3)	003240	077777	PAT12:	.WORD	077777	;PATTERN 12
(3)	003242	137777		.WORD	137777	
(3)	003244	157777		.WORD	157777	
(3)	003246	167777		.WORD	167777	
(3)	003250	173777		.WORD	173777	
(3)	003252	175777		.WORD	175777	
(3)	003254	176777		.WORD	176777	
(3)	003256	177377		.WORD	177377	
(3)	003260	177577		.WORD	177577	
(3)	003262	177677		.WORD	177677	
(3)	003264	177737		.WORD	177737	
(3)	003266	177757		.WORD	177757	
(3)	003270	177767		.WORD	177767	
(3)	003272	177773		.WORD	177773	
(3)	003274	177775		.WORD	177775	
(3)	003276	177776		.WORD	177776	
(3)						
(3)	003300	153333	PAT13:	.WORD	153333	;PATTERN 13
(3)	003302	066667		.WORD	066667	
(3)	003304	153333		.WORD	153333	
(3)	003306	066667		.WORD	066667	
(3)	003310	153333		.WORD	153333	
(3)	003312	066667		.WORD	066667	
(3)	003314	153333		.WORD	153333	
(3)	003316	066667		.WORD	066667	
(3)	003320	153333		.WORD	153333	
(3)	003322	066667		.WORD	066667	
(3)	003324	153333		.WORD	153333	
(3)	003326	066667		.WORD	066667	
(3)	003330	153333		.WORD	153333	
(3)	003332	066667		.WORD	066667	
(3)	003334	153333		.WORD	153333	
(3)	003336	066667		.WORD	066667	
(3)						
(1)	003340	000000	PAT14:	.WORD	000000	;PATTERN 14
(3)	003342	177777		.WORD	177777	
(3)	003344	177777		.WORD	177777	
(3)	003346	177777		.WORD	177777	
(3)	003350	177777		.WORD	177777	
(3)	003352	177777		.WORD	177777	
(3)	003354	177777		.WORD	177777	
(3)	003356	177777		.WORD	177777	
(3)	003360	177777		.WORD	177777	
(3)	003362	177777		.WORD	177777	
(3)	003364	177777		.WORD	177777	
(3)	003366	177777		.WORD	177777	
(3)	003370	177777		.WORD	177777	
(3)	003372	177777		.WORD	177777	
(3)	003374	177777		.WORD	177777	
(3)	003376	177777		.WORD	177777	

(3)
(3) 003400 177777
(3) 003402 000000
(3) 003404 000000
(3) 003406 000000
(3) 003410 000000
(3) 003412 000000
(3) 003414 000000
(3) 003416 000000
(3) 003420 000000
(3) 003422 000000
(3) 003424 000000
(3) 003426 000000
(3) 003430 000000
(3) 003432 000000
(3) 003434 000000
(3) 003436 000000

PAT15: .WORD 177777 ;PATTERN 15
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000

;DPB (DATA PARAMETER BLOCK)

(3)
(3) 003440 000
(3) 003441 000
(3) 003442 000
(3) 003443 000
(3) 003444 000000
(3) 003446 040524
(3)
(3) 003450 000
(3) 003451 000
(3) 003452 000000
(3) 003454 003540
(3)
(3)
(3)
(3) 003456 000000
(3)
(3)
(3)
(3) 003460 000
(3) 003461 000
(3) 003462 000
(3) 003463 000
(3) 003464 177776
(3) 003466 040524
(3)
(3) 003470 000
(3)
(3) 003471 000

DPB.A: .BYTE 0
.BYTE 0
.BYTE 0
.BYTE 0
.WORD 0
.WORD BUFFER
.BYTE 0
.BYTE 0
.WORD 0
.WORD RP.REG
.WORD 0

(0) DRIVE NUMBER
(1) OFFSET VALUE OR FMT22, ECT, AND HCI
(2) COMMAND
(3) PSEL AND A17 AND A16
(4) WORD COUNT (MUST BE NEG.)
(6) BUFFER ADDRESS OR REGISTER TABLE POINTER
(10) SECTOR ADDRESS OR FIRST REG. INDEX
(11) TRACK ADDRESS OR LAST REG. INDEX
(12) CYLINDER ADDRESS
(14) ERROR TABLE POINTER
POINTS TO THE FIRST OF TWENTY LOCATIONS OF WHERE THE DRIVER IS TO STORE THE RH11/RP04 REGISTERS ON AN ERROR. IF LEFT ZERO REGISTERS ARE NOT SAVED.
(16) STATUS/ERROR INDICATOR
BIT15=1=>ERROR OCCURRED
BIT07=1=>DONE
BIT14-BIT09 AND BIT06-BIT03
INDICATE TYPE OF ERROR

DPB.B: .BYTE 0
.BYTE 0
.BYTE 0
.BYTE 0
.WORD -2
.WORD BUFFER
.BYTE 0
.BYTE 0

(0) DRIVE NUMBER
(1) OFFSET VALUE OR FMT22, ECT, AND HCI
(2) COMMAND
(3) PSEL AND A17 AND A16
(4) WORD COUNT (MUST BE NEG.)
(6) BUFFER ADDRESS OR REGISTER TABLE POINTER
(10) SECTOR ADDRESS OR FIRST REG. INDEX
(11) TRACK ADDRESS OR

(3)					:LAST REG. INDEX
(3)	003472	000000	.WORD	0	: (12) CYLINDER ADDRESS
(3)	003474	003540	.WORD	RP.REG	: (14) ERROR TABLE POINTER
(3)					: POINTS TO THE FIRST OF TWENTY
(3)					: LOCATIONS OF WHERE THE DRIVER
(3)					: IS TO STORE THE RH11/RP04
(3)					: REGISTERS ON AN ERROR. IF LEFT
(3)					: ZERO REGISTERS ARE NOT SAVED.
(3)	003476	000000	.WORD	0	: (16) STATUS/ERROR INDICATOR
(3)					: BIT15=1=>ERROR OCCURRED
(3)					: BIT07=1= DONE
(3)					: BIT14-BIT09 AND BIT06-BIT03
(3)					: INDICATE TYPE OF ERROR
(3)	003500	000			: (0) DRIVE NUMBER
(3)	003501	000	.BYTE	0	: (1) OFFSET VALUE OR FMT22, ECT, AND HCI
(3)	003502	000	.BYTE	0	: (2) COMMAND
(3)	003503	000	.BYTE	0	: (3) PSEL AND A17 AND A16
(3)	003504	177776	.WORD	-2	: (4) WORD COUNT (MUST BE NEG.)
(3)	003506	040524	.WORD	BUFFER	: (6) BUFFER ADDRESS OR
(3)					: REGISTER TABLE POINTER
(3)	003510	000	.BYTE	0	: (10) SECTOR ADDRESS OR
(3)					: FIRST REG. INDEX
(3)	003511	000	.BYTE	0	: (11) TRACK ADDRESS OR
(3)					: LAST REG. INDEX
(3)	003512	000000	.WORD	0	: (12) CYLINDER ADDRESS
(3)	003514	003540	.WORD	RP.REG	: (14) ERROR TABLE POINTER
(3)					: POINTS TO THE FIRST OF TWENTY
(3)					: LOCATIONS OF WHERE THE DRIVER
(3)					: IS TO STORE THE RH11/RP04
(3)					: REGISTERS ON AN ERROR. IF LEFT
(3)					: ZERO REGISTERS ARE NOT SAVED.
(3)	003516	000000	.WORD	0	: (16) STATUS/ERROR INDICATOR
(3)					: BIT15=1=>ERROR OCCURRED
(3)					: BIT07=1=>DONE
(3)					: BIT14-BIT09 AND BIT06-BIT03
(3)					: INDICATE TYPE OF ERROR
(3)	003520	000			: (0) DRIVE NUMBER
(3)	003521	000	.BYTE	0	: (1) OFFSET VALUE OR FMT22, ECT, AND HCI
(3)	003522	000	.BYTE	0	: (2) COMMAND
(3)	003523	000	.BYTE	0	: (3) PSEL AND A17 AND A16
(3)	003524	000000	.WORD	0	: (4) WORD COUNT (MUST BE NEG.)
(3)	003526	040524	.WORD	BUFFER	: (6) BUFFER ADDRESS OR
(3)					: REGISTER TABLE POINTER
(3)	003530	000	.BYTE	0	: (10) SECTOR ADDRESS OR
(3)					: FIRST REG. INDEX
(3)	003531	000	.BYTE	0	: (11) TRACK ADDRESS OR
(3)					: LAST REG. INDEX
(3)	003532	000000	.WORD	0	: (12) CYLINDER ADDRESS
(3)	003534	003540	.WORD	RP.REG	: (14) ERROR TABLE POINTER
(3)					: POINTS TO THE FIRST OF TWENTY
(3)					: LOCATIONS OF WHERE THE DRIVER
(3)					: IS TO STORE THE RH11/RP04


```

(3)
(3) 003536 000000
(3)
(3)
(3)
(3)
(3)
(3)
(3)
(3)
(3)
(3)
(3) 003540 000000
(3) 003542 000000
(3) 003544 000000
(3) 003546 000000
(3) 003550 000000
(3) 003552 000000
(3) 003554 000000
(3) 003556 000000
(3) 003560 000000
(3) 003562 000000
(3) 003564 000000
(3) 003566 000000
(3) 003570 000000
(3) 003572 000000
(3) 003574 000000
(3) 003576 000000
(3) 003600 000000
(3) 003602 000000
(3) 003604 000000
(3) 003606 000000

```

;SAVE RH11/RP04 REGISTERS HERE ON ERROR

```

RP.REG: .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0
         .WORD 0

```

```

;REGISTERS ON AN ERROR. IF LEFT
;ZERO REGISTERS ARE NOT SAVED.
;(16) STATUS/ERROR INDICATOR
;BIT15=1=>ERROR OCCURRED
;BIT07=1=>DONE
;BIT14-BIT09 AND BIT06-BIT03
;INDICATE TYPE OF ERROR

;RHCS1 (776700) CONTROL & STATUS #1
;RHWC (776702) WORD COUNT
;RHBA (776704) BUS ADDRESS
;RHDA (776706) DESIRED SECTOR/TRACK
;RHCS2 (776710) CONTROL & STATUS #2
;RHDS1 (776712) DISK STATUS
;RHER1 (776714) ERROR REG. #1
;RHAS (776716) ATTENTION SUMMARY
;RHLA (776720) LOOK AHEAD
;RHDB (776722) DATA BUFFER
;RHMR (776724) MAINTAINABILITY
;RHDT (776726) DRIVE TYPE
;RHSN (776730) SERIAL NUMBER
;RHOF (776732) OFFSET
;RHCA (776734) DESIRED CYLINDER
;RHCC (776736) CURRENT CYLINDER
;RHER2 (776740) ERROR REG #2
;RHER3 (776742) ERROR REG #3
;RHEC1 (776744) ECC POSITION
;RHEC2 (776746) ECC PATTERN

```

(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
003610

5716
5717
5718
5719
5720
5721
5722
5723
5724
5725
5726
5727
5728
5729 003610 035205
5730 003612 036514
5731 003614 037754
5732 003616 040360
5733
5734
5735
5736
5737
5738
5739 003620 035253
5740 003622 036533
5741 003624 037760
5742 003626 040364
5743
5744
5745
5746
5747
5748
5749 003630 035313
5750 003632 036631
5751 003634 040000
5752 003636 040370

```
*****  
.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:  
;*EM AND DH ARE ASCIZ MESSAGES, DT IS A STRING OF WORDS THAT POINT TO THE  
;*DATA TO BE TYPED AND DF IS A STRING OF DATA THAT TELL HOW THE DT WORDS  
;*ARE TO BE TYPED. IF ANY OF THE POINTERS ARE NOT NEEDED FOR A PARTICULAR  
;*ERROR IT IS REPLACED WITH A ZERO.  
;*EACH OF THE ITEMS BELOW REFER TO THE ERROR NUMBER AND INDICATE  
;*THE INFORMATION THAT WILL BE TYPED WHEN THE ERROR OCCURS.  
;*UNLESS STATED OTHER ALL NUMBERS ARE OCTAL
```

```
* ERROR ITEM 1  
;* ILLEGAL RH11(SC=0)INTERRUPT OCCURRED  
;* TEST ERR PC  
;* $TMPO $ERRPC
```

EM1
DH1
DT1
DF1

```
* ERROR ITEM 2  
;* ILLEGAL RPO4 INTERRUPT OCCURRED  
;* TEST ERR PC DRIVE RNAS RACS RHER1 RHER2 RHER3  
;* $TMPO $ERRPC $REG1 $REGS RPERRS RPERRS+2 RPERRS+4 RPERRS+6
```

EM2
DH2
DT2
DF2

```
* ERROR ITEM 3  
;* MASSBUS PARITY ERROR (MCPE=1)  
;* TEST ERR PC ADDRESS DATA  
;* $TMPO $ERRPC RD.ADR RD.WRD
```

EM3
DH3
DT3
DF3

5753										
5754			;	*	ERROR ITEM 4					
5755			;	*	MASSBUS PARITY ERROR (PAR=1)					
5756			;	*	TEST ERR PC ADDRESS GODATA BDDATA					
5757			;	*	\$TMPO \$ERRPC WRT.ADR WRT.WD RD.WRD					
5758										
5759	003640	035350			EM4					
5760	003642	036666			DH4					
5761	003644	040010			DT4					
5762	003646	040374			DF4					
5763										
5764			;	*	ERROR ITEM 5					
5765			;	*	OFFLINE, NONEXISTENT, OR UNSAFE DRIVE INTERRUPTED					
5766			;	*	TEST ERR PC DRIVE RHAS RHDS RHER1 RHER2 RHER3					
5767			;	*	\$TMPO \$ERRPC \$REG: \$REG5 RPERAS RPERRS+2 RPERRS+4 RPERRS+5					
5768										
5769	003650	035404			EM5					
5770	003652	036533			DH2					
5771	003654	037760			DT2					
5772	003656	040364			DF2					
5773										
5774			;	*	ERROR ITEM 6 -- NOT USED					
5775										
5776	003660	000000			0					
5777	003662	000000			0					
5778	003664	000000			0					
5779	003666	000000			0					
5780										
5781			;	*	ERROR ITEM 7 -- NOT USED					
5782										
5783	003670	000000			0					
5784	003672	000000			0					
5785	003674	000000			0					
5786	003676	000000			0					
5787										
5788			;	*	ERROR ITEM 10					
5789			;	*	RH11/RPO4 FAILED TO RESPOND TO ADDRESSING					
5790			;	*	RHCS1 ERR PC					
5791			;	*	RH.ADR \$ERRPC					
5792										
5793	003700	035467			EM10					
5794	003702	036735			DH10					
5795	003704	040042			DT10					
5796	003706	040400			DF10					
5797										
5798			;	*	ERROR ITEM 11					
5799			;	*	DRIVE SELECTED IS NOT ONLINE					
5800			;	*	DRIVE ERR PC					
5801			;	*	\$REG2 \$ERRPC					
5802										
5803	003710	035541			EM11					
5804	003712	036754			DH11					
5805	003714	040046			DT11					
5806	003716	040404			DF11					

5807			
5808			;* ERROR ITEM 12
5809			;* IMPROPER HEADER DATA
5810			;* TEST ERR PC TST PC DRIVE CYLNR TRACK SECTOR
5811			;* \$TMPO \$ERRPC \$REGD CHKDRV CYL.DS TRK.DS SEC.DS
5812			;* GDCYL GDTRK GDSCTR BDCYL BDRK BOSCTR
5813			;* CYL.DS TRK.DS SEC.DS CYL.RD TRK.RD SEC.RD
5814			;* CYLNR, TRACK, AND SECTOR ARE DECIMAL
5815			
5816	003720	035576	EM12
5817	003722	036773	DH12
5818	003724	040052	DT12
5819	003726	040410	DF12
5820			
5821			;* ERROR ITEM 13
5822			;* DATA COMPARE FAILURE
5823			;* TEST ERR PC TST PC DRIVE CYLNR TRACK SECTOR
5824			;* \$TMPO \$ERRPC \$REGD CHKDRV CYL.DS TRK.DS SEC.DS
5825			;* GDDAT BDDAT WRDCNT GDADR BDADR
5826			;* \$GDDAT \$BDDAT \$REG4 \$GDADR \$BDADR
5827			;* CYLNR, TRACK, SECTOR, AND WRDCNT ARE DECIMAL
5828			
5829	003730	035623	EM13
5830	003732	036773	DH12
5831	003734	040104	DT13
5832	003736	040420	DF13
5833			
5834			;* ERROR ITEM 14 -- FOLLOWS #13
5835			;* \$GDDAT \$BDDAT \$REG4 \$GDADR \$BDADR
5836			
5837	003740	000000	0
5838	003742	000000	0
5839	003744	040122	DT13A
5840	003746	040430	DF14
5841			
5842			;* ERROR ITEM 15
5843			;* DATA COMPARE FAILURE
5844			;* TEST ERR PC TST PC DRIVE CYLNR TRACK SECTOR
5845			;* \$TMPO \$ERRPC \$REGD CHKDRV CYL.DS TRK.DS SEC.DS
5846			;* GDDAT BDDAT WRDCNT GDADR BDADR
5847			;* \$GDDAT \$BDDAT \$REG4 \$GDADR \$BDADR
5848			;* CYLNR, TRACK, SECTOR, AND WRDCNT ARE DECIMAL
5849			
5850	003750	035623	EM13
5851	003752	036773	DH12
5852	003754	040104	DT13
5853	003756	040420	DF13
5854			
5855			;* ERROR ITEM 16 -- FOLLOWS #15
5856			;* \$GDDAT \$BDDAT \$REG4 \$GDADR \$BDADR
5857			
5858	003760	000000	0
5859	003762	000000	0
5860	003764	040122	DT13A

5861 003766 040430

DF14

5862

5863

5864

5865

5866

5867

5868

5869

5870

5871

5872

5873

5874

5875

5876

5877

5878

5879

5880

5881

5882

5883

5884

5885

5886

5887

5888

5889

5890

5891

5892

5893

5894

5895

5896

5897

5898

5899

5900

5901

5902

5903

5904

5905

5906

5907

5908

5909

5910

5911

5912

5913

```

:* ERROR ITEM 17
:* DISK ERROR IN TIMING TEST
:* TEST ERR PC DRIVE RHCS1 RHDS RHER1 RHER2 RHER3
:* $TMPD $ERRPC $CHKDRV $P.REG $P.REG+12 $P.REG+14 $P.REG+40 $P.REG+42

```

EM17

DH17

DT17

DF17

```

:* ERROR ITEM 20
:* CLOCK (KW11-P) OVERFLOW IN TIMING TEST
:* TEST ERR PC DRIVE RHCS1 RHDS RHER1 RHER2 RHER3
:* $TMPD $ERRPC $CHKDRV $P.REG $P.REG+12 $P.REG+14 $P.REG+40 $P.REG+42

```

EM20

DH17

DT17

DF17

```

:* ERROR ITEM 21
:* DATA COMPARE FAILURE
:* TEST ERR PC 1ST FC DRIVE CYLNR TRACK
:* $TMPD $ERRPC $REGO $CHKDRV $CYL.DS $TRK.DS
:* $GDDAT $BDDAT $WRDCNT $SECTOR
:* $SREG1 $SBDDAT $SREG4 $SREG1
:* CYLINDER, TRACK, WRDCNT, AND SECTOR ARE DECIMAL

```

EM13

DH21

DT21

DF21

```

:* ERROR ITEM 22--FOLLOWS #21
:* $SREG1 $SBDDAT $SREG4 $SREG1

```

0

0

DT21A

DF22

```

*****
*****

```

```

:* ERROR ITEMS 23-40 NOT USED
:* ERROR ITEMS 41-46 WILL HAVE AN EM THAT
:* VARIES DEPENDING ON THE ERROR, IT WILL BE IN THE FORM:
:* RH11/RP04 ERROR (MESSAGE)
:* WHERE MESSAGE WILL BE ONE OR MORE OF THE FOLLOWING:
:* 1) OFFLINE OR UNSAFE DRIVE REQUESTED
:* 2) UNLOADED DRIVE REQUESTED
:* 3) PERSISTENT UNSAFE
:* 4) PARITY ERROR OCCURRED

```

```

5914      ; * 5) FATAL PARITY ERROR
5915      ; * 6) SOFTWARE TIMEOUT ON THIS DRIVE
5916      ; * 7) SOFTWARE TIMEOUT ON ANOTHER DRIVE
5917      ; * 8) ERROR OCCURRED DURING I/O OPERATION
5918      ; * 9) ERROR OCCURRED DURING NON-I/O OPERATION
5919      ; * 10) UNSAFE OCCURRED
5920      ; * 11) AUTOMATIC RECALIBRATE SEQUENCE OCCURRED
5921
5922 004030 ITEM41:
5923
5924      ; * ERROR ITEM 41
5925      ; * RH11/RP04 ERROR (MESSAGE)
5926      ; * TEST ERR PC TST PC DRIVE
5927      ; * $TMPO $ERRPC $REGO CHKDRV
5928
5929 004030 035751      EM41
5930 004032 037422      DM41
5931 004034 040200      DT41
5932 004036 040454      DF41
5933
5934      ; * ERROR ITEM 42
5935      ; * RH11/RP04 ERROR (MESSAGE)
5936      ; * TEST ERR PC TST PC DRIVE RHCS1 RHCS2 RHDS
5937      ; * $TMPO $ERRPC $REGO CHKDRV RP.REG RP.REG+10 RP.REG+12
5938
5939 004040 035751      EM41
5940 004042 037460      DM42
5941 004044 040210      DT42
5942 004046 040460      DF42
5943
5944      ; * ERROR ITEM 43
5945      ; * RH11/RP04 ERROR (MESSAGE)
5946      ; * TEST ERR PC TST PC DRIVE RHCS1 RHCS2 RHDS
5947      ; * $TMPO $ERRPC $REGO CHKDRV RP.REG RP.REG+10 RP.REG+12
5948      ; * RHER1 RHER2 RHER3
5949      ; * RP.REG+14 RP.REG+40 RP.REG+42
5950
5951 004050 035751      EM41
5952 004052 037460      DM42
5953 004054 040226      DT43
5954 004056 040464      DF43
5955
5956      ; * ERROR ITEM 44
5957      ; * RH11/RP04 ERROR (MESSAGE)
5958      ; * TEST ERR PC TST PC DRIVE CYLNDR TRACK SECTOR
5959      ; * $TMPO $ERRPC $REGO CHKDRV CYL.DS TRK.DS SEC.DS
5960      ; * RHCS1 RHCS2 RHDS RHCC RHCA RHDA
5961      ; * RP.REG RP.REG+10 RP.REG+12 RP.REG+36 RP.REG+34 RP.REG+06
5962      ; * RHER1 RHER2 RHER3
5963      ; * RP.REG+14 RP.REG+40 RP.REG+42
5964      ; * CYLNDR, TRACK, AND SECTOR ARE DECIMAL
5965
5966 004060 035751      EM41
5967 004062 036773      DM12

```

5968 004064 040252
5969 004066 040474
5970
5971
5972
5973
5974
5975
5976
5977
5978
5979
5980
5981 004070 035751
5982 004072 035773
5983 004074 040312
5984 004076 040510
5985
5986
5987
5988
5989
5990
5991
5992
5993
5994
5995
5996 004100 035771
5997 004102 036773
5998 004104 040312
5999 004106 040510
6000

DT44
DF44

```

:* ERROR ITEM 45
:* RH11/RP04 ERROR (MESSAGE)
:* TEST ERR FC TST PC DRIVE CYLNR TRACK SECTOR
:* $TMPD $ERRPC $REGD CHKDRV CYL.DS TRK.DS SEC.DS
:* RHCS1 RHCS2 RHDS RHCC RHCA RHDA
:* RP.REG RP.REG+10 RP.REG+12 RP.REG+36 RP.REG+34 RP.REG+06
:* RHER1 RHER2 RHER3 RHERC RHBA RHOB
:* RP.REG+14 RP.REG+40 RP.REG+42 RP.REG+2 RP.REG+4 RP.REG+22
:* CYLNR,TRACK, AND SECTOR ARE DECIMAL

```

EM41
DH12
DT45
DF45

```

:* ERROR ITEM 46
:* FATAL WRITE CHECK ERROR (MESSAGE)
:* TEST ERR PC TST PC DRIVE CYLNR TRACK SECTOR
:* $TMPD $ERRPC $REGD CHKDRV CYL.DS TRK.DS SEC.DS
:* RHCS1 RHCS2 RHDS RHCC RHCA RHDA
:* RP.REG RP.REG+10 RP.REG+12 RP.REG+36 RP.REG+34 RP.REG+06
:* RHER1 RHER2 RHER3 RHERC RHBA RHOB
:* RP.REG+14 RP.REG+40 RP.REG+42 RP.REG+2 RP.REG+4 RP.REG+22
:* CYLNR,TRACK, AND SECTOR ARE DECIMAL

```

EM46
DH12
DT45
DF45

```

6150
6151          .SBTTL  START OF PROGRAM
6152
6153
6155 004110 012737 177777 001216 START3: MOV    #-1,2#BUSADR ;GET BUSADR FLAG
6156 004116 000402          BR      STRT1A
6157 004120 005037 001216          START1: CLR    2#BUSADR ;CLR BUSADR FLAG
6158 004124 012737 037777 001224 STRT1A: MOV    #37777,TSTNMS ;SELECT TESTS 0-15
6159 004132 005037 001214          CLR    2#CNTRLC ;NO CONTROL "C"
6160 004136 012700 001614          MOV    #DFLT,RO ;DEFAULT PARAMETERS POINTER
6161 004142 012701 002062          MOV    #PRMO,R1 ;TABLE POINTER
6162 004146 010102          MOV    R1,R2 ;STOP ADDRESS
6163 004150 012021          15:  MOV    (RO)+,(R1)+ ;MOVE DEFAULT PARAMETERS INTO
6164 004152 020002          CMP    RO,R2 ;RUN TIME TABLES ** DONE?
6165 004154 103775          BLO   15 ;NO--BRANCH
6166 004156 012700 003040          MOV    #PAT8,RO ;PATO DEFAULTS TO PATTERN 8
6167 004162 012701 002440          MOV    #PATO,R1
6168 004166 012021          25:  MOV    (RO)+,(R1)+
6169 004170 020027 003100          CMP    RO,#PAT9
6170 004174 103774          BLO   25
6171 004176 000411          BR      START
6172 004200 012737 177777 001216 START4: MOV    #-1,2#BUSADR ;SET BUSADR FLAG
6173 004206 000402          BR      STRT2A
6174 004210 005037 001216          START2: CLR    2#BUSADR ;CLR BUSADR FLAG
6175 004214 012737 177777 001214 STRT2A: MOV    #-1,2#CNTRLC ;SET CONTROL "C" FLAG
6176 004222 000005          START:  RESET
6177 004224 012737 000340 177776          MOV    #340,2#PS ;LOCK OUT ALL INTERRUPTS
        (1) 004232 012706 001100          MOV    #SCMTAG,R6 ;FIRST LOCATION TO BE CLEARED
        (1) 004236 005026          CLR    (R6)+ ;CLEAR MEMORY LOCATION
        (1) 004240 022706 001136          CMP    #STKS,R6 ;DONE?
        (1) 004244 001374          BNE   -6 ;LOOP BACK IF NO
        (1) 004246 012706 001100          MOV    #STACK,SP ;SETUP THE STACK POINTER
        (1) 004252 012737 016102 000020          MOV    #SCOPE,2#IOTVEC ;IOT VECTOR FOR SCOPE ROUTINE
        (1) 004260 012737 000340 000022          MOV    #340,2#IOTVEC+2 ;LEVEL 7
        (1) 004266 012737 013762 000030          MOV    #ERROR,2#EMTVEC ;EMT VECTOR FOR ERROR ROUTINE
        (1) 004274 012737 000340 000032          MOV    #340,2#EMTVEC+2 ;LEVEL 7
        (1) 004302 012737 016416 000034          MOV    #STRAP,2#TRAPVEC ;TRAP VECTOR FOR TRAP CALLS
        (1) 004310 012737 000340 000036          MOV    #340,2#TRAPVEC+2 ;LEVEL 7
        (1) 004316 005037 001176          CLR    $TIMES ;INITIALIZE NUMBER OF ITERATIONS
        (1) 004322 005037 001200          CLR    $ESCAPE ;CLEAR THE ESCAPE ON ERROR ADDRESS
        (1) 004326 112737 000001 001115          MOVB  #1,$ERMAX ;ALLOW ONE ERROR PER TEST
        (1) 004334 012737 004334 001106          MOV    #,$SLPADR ;INITIALIZE THE LOOP ADDRESS FOR SCOPE
        (1) 004342 012737 004342 001110          MOV    #,$SLPERA ;SETUP THE ERROR LOOP ADDRESS
6178 004350 012700 001152          MOV    #REGAD,RU ;FIRST ADDRESS
6179 004354 005020          15:  CLR    (RO)+ ;CLEAR VARIABLE STORAGE
6180 004356 022700 001202          CMP    #SBELL,RO ;DONE?
6181 004362 001374          BNE   15 ;NO--BRANCH
6182 004364 013737 001354 001144          MOV    2#TPS,2#STPB ;SETUP THE STATUS AND BUFFER REG'S
6183 004372 013737 001356 001144          MOV    2#TPB,2#STPB ;FOR THE TYPE ROUTINE
6184 004400 063727 001214 000000          ADD    2#CNTRLC,#0 ;TYPE "ID" IF START AT 200 OR
6185 004406 103433          BCS   SRTINT ;THE 1ST START AT 204
6186 004410 104400 004416          TYPE  #4 ;TYPE ASCIZ STRING
        (1) 004414 000430          BR      645 ;GET OVER THE ASCIZ
        (1)          ;;.ASCIZ <15><12>/MD-11-DZRPK-B "MECHANICAL & READ-WRITE" TEST.

```


(1)	004476			64\$:			
6187	004476	004737	017316	SRTINT:	JSR	PC, @#LP.AVL	;CHECK FOR A LINE PRINTER
6188	004502	004737	015346		JSR	PC, @#STKINT	;TURN ON THE TTY KEYBOARD INTERRUPT
6189	004506	005037	177776		CLR	@#PS	;INSURE THE PRIORITY = 0
6190	004512	004737	024200		JSR	PC, @#GETADR	;CHECK RH11/RP04 (RHCS1) ADDRESS
6191	004516	004737	024112		JSR	PC, @#GETSWR	;CO CHECK FOR CONTROL SWITCHES
6192	004522	004737	027204		JSR	PC, @#RPINIT	;SETUP DRIVE STATUS
6193	004526	004737	017360		JSR	PC, @#ST.CLK	;INITIALIZE THE CLOCK
6194	004532	005737	001214		TST	@#CNTRLC	;CONTROL "C" START/RESTART?
6195	004536	001403			BEQ	1\$;NO--BRANCH
6196	004540	004737	024550		JSR	PC, @#GT.PRM	;YES--GET PARAMETERS
6197	004544	000416			BR	4\$	
6198	004546	005037	001222	1\$:	CLR	DRVSEL	;NO DRIVES SELECTED
6199	004552	005000			CLR	RO	;DETERMINE THE DRIVES THAT
6200	004554	012701	000001		MOV	#1, R1	;ARE AVAILABLE FOR TESTING
6201	004560	105760	027050	2\$:	TSTB	DRVSTA(RO)	
6202	004564	003403			BLE	3\$	
6203	004566	156037	027154	001222	BISB	ATABIT(RO), @#DRVSEL	
6204	004574	005200		3\$:	INC	RO	
6205	004576	106301			ASLB	R1	
6206	004600	001367			BNE	2\$	
6207	004602	005037	027130	4\$:	CLR	@#SEEKFG	;CLEAR SEEK FLAG
6208	004606	032737	000400	001212	BIT	#SW08, @#C.SWR	;DO SEEK BEFORE DATA TRANSFER?
6209	004614	001002			BNE	5\$;YES--BRANCH
6210	004616	005137	027130		COM	@#SEEKFG	;NO
6211	004622			5\$:			
(1)	004622	104400	004630		TYPE	64\$+4	;TYPE ASCIZ STRING
(1)	004626	000415			BR	64\$;GET OVER THE ASCIZ
(1)					::.ASCIZ	<15><12>/DRIVE(S) TO BE TESTED /	
(1)	004662			64\$:			
6212	004662	005037	013660		CLR	@#SENDOCT	;DETERMINE PASSES TO MAKE AND
6213	004666	005000			CLR	RO	;THE DRIVES TO BE TESTED
6214	004670	013701	001222		MOV	@#DRVSEL, R1	;ANY DRIVES SELECTED?
6215	004674	001010			BNE	6\$;YES--BRANCH
6216	004676	104400	004704		TYPE	65\$+4	;TYPE ASCIZ STRING
(1)	004702	000403			BR	65\$;GET OVER THE ASCIZ
(1)					::.ASCIZ	/NONE/	
(1)	004712			65\$:			
6217	004712	000137	013476		JMP	@#SEOP	;GO TO END OF PROGRAM
6218	004716	006201		6\$:	ASR	R1	;REPORT THE DRIVES TO BE TESTED
6219	004720	103013			BCC	7\$	
6220	004722	005237	013660		INC	@#SENDOCT	;GIVE THIS DRIVE A PASS
6221	004726	010046			MOV	RO, -(SP)	;SAVE RO FOR TYPEOUT
(1)	004730	104404			TYPOS		;GO TYPE--OCTAL ASCII
(1)	004732	001			.BYTE	1	;TYPE 1 DIGIT(S)
(1)	004733	000			.BYTE	0	;SUPPRESS LEADING ZEROS
6222	004734	005701			TST	R1	;MORE DRIVES?
6223	004736	001406			BEQ	8\$;NO--BRANCH
6224	004740	104400	004746		TYPE	66\$+4	;TYPE ASCIZ STRING
(1)	004744	000401			BR	66\$;GET OVER THE ASCIZ
(1)					::.ASCIZ	"."	
(1)	004750			66\$:			
6225	004750	005200		7\$:	INC	RO	;FORM DRIVE NUMBER
6226	004752	000761			BR	6\$	

```

6227 004754 013737 013650 013652 8$: MOV 2#SENDCT,2#SEOPCT
6228 004762 005737 001226 TST 2#CLKSTA ;KW11-P AVAILABLE
6229 004766 003041 BGT RSTRT1 ;YES--BRANCH
6230 004770 032737 003600 001224 BIT 2#3600,2#TSTNMS ;NO--ANY TIMING TESTS TO BE PERFORMED?
6231 004776 001435 BEQ RSTRT1 ;NO--BRANCH
6232 005000 104400 005006 TYPE +4 ;TYPE ASCIZ STRING
(1) 005004 000432 BR 67$ ;GET OVER THE ASCIZ
(1) ;;.ASCIZ <15><12>/NO KW11-P CLOCK--TEST 7-12 WILL NOT BE PERFORMED/
(1) 005072 67$:
6233 005072 005037 001236 RSTRT1: CLR CHKDRV ;INIT. THE CHECK DRIVE KEY
6234 005076 012737 000001 001240 MOV 2#1,DRVMSK ;START TO CHECK DESIRED DRIVES
6235 005104 033737 001240 001222 RSTRT2: BIT DRVMSK,DRVSEL ;IS THIS DRIVE SELECTED?
6236 005112 001006 RSTRT3: BNE DRVOK ;YES--GO CHECK IF DRIVE IS READY FOR TESTING
6237 005114 005237 001236 RESTART: INC CHKDRV ;MOVE TO NEXT DRIVE NUMBER
6238 005120 106337 001240 ASLB DRVMSK ;POSITION THE MASK
6239 005124 103762 BCS RSTRT1 ;BRANCH IF THE DRIVE NUMBER NEEDS INITIALIZED
6240 005126 000766 BR RSTRT2
6241
6242 005130 013702 001236 DRVOK: MOV 2#CHKDRV,R2 ;PICKUP THE DRIVE NUMBER
6243 005134 105762 027050 TSTB DRVSTA(R2) ;IS DESIRED DRIVE ON-LINE?
6244 005140 003002 BGT 1$ ;YES, BRANCH
6245 005142 104011 ERROR 11 ;DRIVE SELECTED IS NOT ONLINE
6246 005144 000763 BR RESTART ;NO-- RETURN
6247 005146 010237 003440 1$: MOV R2,2#DPB.A ;SET THE DRIVE NUMBER INTO THE DPB'S
6248 005152 010237 003460 MOV R2,2#DPB.B
6249 005156 010237 003500 MOV R2,2#DPB.C
6250 005162 010237 003520 MOV R2,2#DTADPB
6251 005166 004737 017636 JSR PC,2#LDCMD ;LOAD COMMAND INTO DPB.B AND DPB.C
6252 005172 012737 013476 001234 MOV 2#SEOP,2#BYPASS ;IF ERROR GO TO END OF PROGRAM
6253 005200 112737 000020 003441 MOV 2#FMT22/10256,2#DPB.A+1
(1) 005206 012737 000143 003442 MOV 2#SETFORM,2#DPB.A+2 ;SETFORM=COMMAND
6254 005214 004037 0117702 JSR R0,2#CALL.A ;GO EXECUTE THE COMMAND
6255 005220 012737 000107 003442 MOV 2#RECAL,2#DPB.A+2 ;RECAL=COMMAND
6256 005226 004037 017702 JSR R0,2#CALL.A ;GO EXECUTE THE COMMAND
6257 005232 012737 006730 003450 MOV 2#RHSN,2#DPB.A+10 ;FIRST REG. INDEX
6258 005240 112737 000630 003451 MOV 2#RHSN,2#DPB.A+11 ;LAST REG. INDEX
6259 005246 012737 000141 003442 MOV 2#GETREG,2#DPB.A+2 ;GETREG=COMMAND
6260 005254 004037 017702 JSR R0,2#CALL.A ;GO EXECUTE THE COMMAND
6261 005260 104400 005266 TYPE +4 ;TYPE ASCIZ STRING
(1) 005264 000411 BR 64$ ;GET OVER THE ASCIZ
(1) ;;.ASCIZ <15><12><12>/TESTING DRIVE /
(1) 005310 64$:
6262 005310 010246 MOV R2,-(SP) ;SAVE R2 FOR TYPEOUT
(1) 005312 104404 TYPOS ;GO TYPE--OCTAL ASCII
(1) 005314 001 .BYTE 1 ;TYPE 1 DIGIT(S)
(1) 005315 000 .BYTE 0 ;SUPPRESS LEADING ZEROS
6263 005316 104400 035202 TYPE ,MSG.SP ;TYPE SPACES
6264 005322 104400 005330 TYPE +4 ;TYPE ASCIZ STRING
(1) 005326 000410 BR 65$ ;GET OVER THE ASCIZ
(1) ;;.ASCIZ /SERIAL NUMBER /
(1) 005350 65$:
6265 005350 012700 000004 MOV 2#4,R0 ;FOUR DIGITS TO TYPE
6266 005354 013701 040524 MOV 2#BUFFER,R1 ;SERIAL NUMBER
6267 005360 005002 2$: CLR R2 ;ZERO

```

6268	005362	006101		ROL	R1	:PUT THE NEXT DIGIT
6269	005364	006102		ROL	R2	:INTO R2
6270	005366	006101		ROL	R1	
6271	005370	006102		ROL	R2	
6272	005372	006101		ROL	R1	
6273	005374	006102		ROL	R2	
6274	005376	006101		ROL	R1	
6275	005400	006102		ROL	R2	
6276	005402	062702	000060	ADD	#'0,R2	:MAKE IT ASCII
6277	005406	010227		MOV	R2,(PC)+	:SAVE IT
6278	005410	000000	3\$:	.WORD	0	
6279	005412	104400	005410	TYPE	,3\$:TYPE
6280	005416	005300		DEC	R0	:ALL DIGITS TYPED?
6281	005420	003357		BGT	2\$:NO -- BRANCH
6282	005422	104400	001207	TYPE	,SCLF	

6335
(3)
(4)
(4)
(4)
(4)
(4)
(4)
(3)
(3)
(3)
(4)
(4)
(4)
(4)
(3)
(3)
(3)
(3)
6336
6337
6338
6339
6340
6341
6342
6343
6344
6345
6357

005426 033737 001374 001224
005426 001002
005434 000137 005554
005442 013737 002064 001176
005450 012737 005536 001106
005456 012737 005426 001110
005464 012737 000000 001102
005472 013737 001102 177570
005500 012737 000107 003442
005506 113737 002072 003470
005514 113737 002070 003471
005522 013737 002066 003472
005530 012737 005552 001234
005536 012706 001100
005542 004037 017702
005546 004037 020010
005552 000004
005554 033737 001376 001224
005562 001002
005564 000137 005716
005570 013737 002076 001176
005576 012737 005700 001106
005604 012737 005554 001110
005612 012737 000001 001102
005620 013737 001102 177570
005626 113737 002112 003470
005634 113737 002114 003510

```
*****  
*TEST 0 RECAL/SEEK TEST  
*****  
* THIS TEST WILL CAUSE THE DRIVE TO EXECUTE A RECALIBRATE  
* COMMAND CYCLE AND THEN SEEK FORWARD TO CYLINDER "LC" AT  
* THE COMPLETION OF BOTH COMMANDS STATUS INDICATIONS ARE  
* CHECKED TO INSURE NO ERRORS OCCURRED.  
* THIS TEST WILL BE REPEATED 10 TIMES.  
*****  
†TST0:  
BIT @#BITS+(0*2),TSTNMS ;DO THIS TEST?  
BNE 64$ ;YES--BRANCH  
JMP TST1 ;NO--GO TO THE NEXT TEST  
64$: MOV @#RPT0,$TIMES ;GET THE ITERATION COUNT  
MOV @#TEST0,@#SLPADR  
MOV @#TST0,@#SLPERR ;SETUP THE ERROR LOOP ADDRESS  
MOV #0,@#$TSTNM ;SET UP TEST NUMBER AND  
;CLEAR THE ERROR FLAG ($ERFLG)  
MOV $TSTNM,@#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER  
MOV @#RECAL,@#DPB.A+2 ;RECAL=COMMAND  
MOVB @#FS0,@#DPB.B+10 ;FS  
MOVB @#FT0,@#DPB.B+11 ;FT  
MOV @#LCO,@#DPB.B+12 ;LC  
MOV @#EXIT0,@#BYPASS ;GO TO EXIT0 ON ERROR  
TEST0: MOV @#STACK,SP ;SET UP STACK POINTER  
JSR RD,@#CALL.A ;GO EXECUTE THE COMMAND  
JSR RD,@#CALL.B ;GO EXECUTE THE COMMAND  
EXIT0: SCOPE ;LOOP  
*****  
*TEST 1 SEEK/SEEK TEST  
*****  
* THIS TEST WILL CAUSE THE DRIVE TO EXECUTE A FORWARD SEEK  
* CYCLE TO "LC" "LT" "LS" FOLLOWED BY A REVERSE SEEK CYCLE TO  
* "FC" "FT" "FS". AT THE COMPLETION OF EACH SEEK, THE PROPER  
* INDICATORS ARE EXAMINED TO INSURE PROPER OPERATION.  
* "LC" WILL DEFAULT TO 128 AND "FC", "FT", "LT", "FS", AND "LS"  
* WILL DEFAULT TO 0  
* THIS TEST WILL BE REPEATED 100 TIMES  
*****  
†TST1:  
BIT @#BITS+(1*2),TSTNMS ;DO THIS TEST?  
BNE 64$ ;YES--BRANCH  
JMP TST2 ;NO--GO TO THE NEXT TEST  
64$: MOV @#RPT1,$TIMES ;GET THE ITERATION COUNT  
MOV @#TEST1,@#SLPADR  
MOV @#TST1,@#SLPERR ;SETUP THE ERROR LOOP ADDRESS  
MOV #1,@#$TSTNM ;SET UP TEST NUMBER AND  
;CLEAR THE ERROR FLAG ($ERFLG)  
MOV $TSTNM,@#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER  
MOVB @#FS1,@#DPB.B+10 ;FS  
MOVB @#LS1,@#DPB.C+10 ;LS
```

```

6360 005642 113737 002106 003471      MOV      @#FT1,@#DPB.B+11 ;FT
6361 005650 113737 002110 003511      MOV      @#LT1,@#DPB.C+11 ;LT
6362 005656 013737 002100 003472      MOV      @#FC1,@#DPB.B+12 ;FC
6363 005664 013737 002102 003512      MOV      @#LC1,@#DPB.C+12 ;LC
6364 005672 012737 005714 001234      MOV      @EXIT1,@#BYPASS ;GO TO EXIT1 ON ERROR
6365 005700 012706 001100          TEST1:  MOV      @STACK,SP ;SET THE STACK POINTER
6366 005704 004037 020154          JSR      RO,@#CALL.C ;GO EXECUTE THE COMMAND
6367 005710 004037 020010          JSR      RO,@#CALL.B ;GO EXECUTE THE COMMAND
6368 005714 000004          EXIT1:  SCOPE ;LOOP
6369
6382 ;*****
(3) ;*TEST 2 INCREMENT/SEEK TEST
(4)
(4) ;* THIS TEST WILL COMMAND FORWARD SEEK CYCLES TO ADVANCE THE
(4) ;* CYLINDER ADDRESS FROM "FC" TO "LC" BY THE INCREMENT "IC".
(4) ;* WHEN THE RESULTANT CYLINDER ADDRESS (NC) EXCEEDS
(4) ;* "LC" REVERSE SEEK CYCLES ARE INITIATED; STARTING
(4) ;* AT THE LAST LEGAL "NC" AND DECREMENTING BY "IC"
(4) ;* UNTIL "NC" IS LESS THAN "FC". AT THE COMPLETION OF EACH
(4) ;* SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO
(4) ;* INSURE PROPER OPERATION.
(4)
(3) ;*****
(3) ;*TEST2:
(4) 005716 033737 001400 001224      BIT      @#BITS+(2*2),TSTNMS ;DO THIS TEST?
(4) 005724 001002          BNE      64$ ;YES--BRANCH
(4) 005726 000137 006100          JMP      TST3 ;NO--GO TO THE NEXT TEST
(4) 005732 013737 002120 001176      64$:  MOV      @#RPT2,$TIMES ;GET THE ITERATION COUNT
(3) 005740 012737 006012 001106      MOV      @#TEST2,@#$LPADR
(3) 005746 012737 005716 001110      MOV      @#TST2,@#$LPERR ;SETUP THE ERROR LOOP ADDRESS
(3) 005754 012737 000002 001102      MOV      @2,@#$TSTNM ;SET UP TEST NUMBER AND
(3) ;* ;CLEAR THE ERROR FLAG ($ERFLG)
(3) ;* ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
6383 005770 113737 002132 003470      MOV      $TSTNM,@#DISPLAY
6384 005776 113737 002130 003471      MOV      @#FS2,@#DPB.B+10 ;FS
6385 006004 012737 006076 001234      MOV      @#FT2,@#DPB.B+11 ;FT
6386 006012 012706 001100          TEST2:  MOV      @EXIT2,@#BYPASS ;GO TO EXIT2 ON ERROR
6387 006016 013737 002122 003472      MOV      @#STACK,SP ;SET UP THE STACK POINTER
6388 006024          INCSK:  MOV      @#FC2,@#DPB.B+12 ;FC
6389 006024 004037 020010          JSR      RO,@#CALL.B ;GO EXECUTE THE COMMAND
6390 006030 063737 002126 003472      ADD      @#IC2,@#DPB.B+12 ;MOVE TO NEXT CYLINDER
6391 006036 023737 002124 003472      CMP      @#LC2,@#DPB.B+12 ;OUT OF CYLINDERS?
6392 006044 002367          BGE      INCSK ;NO--BRANCH
6393 006046 013737 002124 003472      MOV      @#LC2,@#DPB.B+12
6394 006054          DECSK:
6395 006054 004037 020010          JSR      RO,@#CALL.B ;GO EXECUTE THE COMMAND
6396 006060 163737 002126 003472      SUB      @#IC2,@#DPB.B+12
6397 006066 023737 002122 003472      CMP      @#FC2,@#DPB.B+12
6398 006074 002767          BLE      DECSK
6399 006076 000004          EXIT2:  SCOPE ;LOOP
6400
6409 ;*****
(3) ;*TEST 3 STEPPING SEEK TEST
(4)

```

```

(4) ;* THIS TEST WILL COMMAND SEEK CYCLES TO CYLINDER 0,1,2,4
(4) ;* 8,16,32,64,128, AND 256. AT THE COMPLETION OF EACH SEEK
(4) ;* COMMAND THE PROPER INDICATORS ARE EXPLAINED TO INSURE PROPER
(4) ;* OPERATION.
(3) ;*****
(3) 006100 033737 001402 001224 †ST3: BIT 2#BITS+(3*2),TSTNMS ;DO THIS TEST?
(4) 006100 001002 BNE 64$ ;YES--BRANCH
(4) 006110 000137 006240 JMP TST4 ;NO--GO TO THE NEXT TEST
(4) 006114 013737 002136 001176 64$: MOV 2#RPT3,$TIMES ;GET THE ITERATION COUNT
(3) 006122 012737 006174 001106 MOV #TEST3,2#SLPADR
(3) 006130 012737 006100 001110 MOV #TST3,2#SLPERR ;SETUP THE ERROR LOOP ADDRESS
(3) 006136 012737 000003 001102 MOV #3,2#$TSTNM ;SET UP TEST NUMBER AND
;CLEAR THE ERROR FLAG (SERFLG)
(3) 006144 013737 001102 177570 MOV $TSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
6410 006152 113737 002150 003470 MOVB 2#FS3,2#DPB.B+10 ;FS
6411 006160 113737 002146 003471 MOVB 2#FT3,2#DPB.B+11 ;FT
6412 006166 012737 006236 001234 MOV #EXIT3,2#BYPASS ;GO TO BYPASS ON ERROR
6413 006174 012706 001100 TEST3: MOV #STACK,SP ;SET UP THE STACK
6414 006200 013737 002140 003472 MOV 2#FC3,2#DPB.B+12 ;FC
6415 006206 004037 020010 JSR RD,2#CALL.B ;GO EXECUTE THE COMMAND
6416 006212 013701 002144 MOV IC3,R1 ;CYLINDER 1
6417 006216 010137 003472 1$: MOV R1,2#DPB.B+12 ;DESIRED CYLINDER
6418 006222 004037 020010 JSR RD,2#CALL.B ;GO EXECUTE THE COMMAND
6419 006226 006301 ASL R1 ;MOVE TO NEXT CYLINDER
6420 006230 020137 002142 CMP R1,2#LC3 ;DONE?
6421 006234 003770 BLE 1$ ;NO--LOOP
6422 006236 000004 EXIT3: SCOPE
6423
6433 ;*****
(3) ;*TEST 4 OSCILLATING SEEK TEST
(4)
(4) ;* THIS TEST WILL COMMAND SEEK CYCLES FROM "FC" TO "NC" AND BACK
(4) ;* TO "FC". "NC" STARTS AT "FC" AND INCREMENTS BY "IC" UP TO CYLINDER
(4) ;* "LC", THEN IS DECREMENTED BY "IC" BACK TO CYLINDER "FC". AT THE
(4) ;* COMPLETION OF EVERY SEEK COMMAND THE PROPER INDICATORS ARE
(4) ;* EXAMINED TO INSURE PROPER OPERATION.
(3) ;*****
(3) 006240 033737 001404 001224 †ST4: BIT 2#BITS+(4*2),TSTNMS ;DO THIS TEST?
(4) 006240 001002 BNE 64$ ;YES--BRANCH
(4) 006250 000137 006572 JMP TST5 ;NO--GO TO THE NEXT TEST
(4) 006254 013737 002154 001176 64$: MOV 2#RPT4,$TIMES ;GET THE ITERATION COUNT
(3) 006262 012737 006350 001106 MOV #TEST4,2#SLPADR
(3) 006270 012737 006240 001110 MOV #TST4,2#SLPERR ;SETUP THE ERROR LOOP ADDRESS
(3) 006276 012737 000004 001102 MOV #4,2#$TSTNM ;SET UP TEST NUMBER AND
;CLEAR THE ERROR FLAG (SERFLG)
(3) 006304 013737 001102 177570 MOV $TSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
6434 006312 113737 002166 003470 MOVB 2#FS4,2#DPB.B+10 ;FS
6435 006320 113737 002164 003471 MOVB 2#FT4,2#DPB.B+11 ;FT
6436 006326 012737 006570 001234 MOV #EXIT4,2#BYPASS ;GO TO EXIT4 ON ERROR
6437 006334 005002 CLR R2 ;CLEAR STALL SWITCH (NO STALL)

```

```

6438 006336 032737 010000 001212 BIT #SW12,2#C.SWR ;STALL REQUIRED?
6439 006344 001401 BEQ TEST4 ;NO--BRANCH
6440 006346 005102 COM R2 ;YES--SET SWITCH
6441 006350 012706 001100 TEST4: MOV #STACK.SP ;SET UP THE STACK POINTER
6442 006354 013701 002156 MOV 2#FC4,R1 ;SET NC TO FC
6443 006360 005037 001320 CLR 2#STALLO ;START AT ZERO IF STALLS REQUIRED
6444 006364 010137 003472 1$: MOV R1,2#DPB.B+12 ;NC
6445 006370 004037 020010 JSR R0,2#CALL.B ;GO EXECUTE THE COMMAND
6446 006374 005702 TST R2 ;STALL?
6447 006376 001403 BEQ 2$ ;NO--BRANCH
6448 006400 004037 021104 JSR R0,2#STALL ;YES--GO TO STALL ROUTINE
6449 006404 001320 .WORD STALLO ;TIME POINTER
6450 006406 013737 002156 003472 2$: MOV FC4,2#DPB.B+12 ;FC
6451 006414 004037 020010 JSR R0,2#CALL.B ;GO EXECUTE THE COMMAND
6452 006420 005702 TST R2 ;STALL?
6453 006422 001413 BEQ 3$ ;NO--BRANCH
6454 006424 004037 021104 JSR R0,2#STALL ;YES--GO TO STALL ROUTINE
6455 006430 001320 .WORD STALLO ;TIME POINTER
6456 006432 005237 001320 INC 2#STALLO ;UPDATE THE TIME
6457 006436 023737 001370 001320 CMP 2#MXSTAL,2#STALLO ;TIME TO BIG?
6458 006444 003347 BGT 1$ ;NO--BRANCH
6459 006446 005037 001320 CLR 2#STALLO ;YES--START OVER AT ZERO
6460 006452 063701 002162 3$: ADD 2#IC4,R1 ;MOVE TO NEXT CYLINDER
6461 006456 020137 002160 CMP R1,2#LC4 ;LAST CYLINDER COMPLETED?
6462 006462 003740 BLE 1$ ;NO--BRANCH
6463 006464 013701 002160 MOV 2#LC4,R1 ;SET NC TO LC
6464 006470 010137 003472 4$: MOV R1,2#DPB.B+12 ;NC
6465 006474 004037 020010 JSR R0,2#CALL.B ;GO EXECUTE THE COMMAND
6466 006500 005702 TST R2 ;STALL?
6467 006502 001403 BEQ 5$ ;NO--BRANCH
6468 006504 004037 021104 JSR R0,2#STALL ;YES--GO TO STALL ROUTINE
6469 006510 001320 .WORD STALLO ;TIME POINTER
6470 006512 013737 002160 003472 5$: MOV 2#LC4,2#DPB.B+12 ;LC
6471 006520 004037 020010 JSR R0,2#CALL.B ;GO EXECUTE THE COMMAND
6472 006524 005702 TST R2 ;STALL?
6473 006526 001413 BEQ 6$ ;NO--BRANCH
6474 006530 004037 021104 JSR R0,2#STALL ;YES--GO TO STALL ROUTINE
6475 006534 001320 .WORD STALLO ;TIME POINTER
6476 006536 005237 001320 INC 2#STALLO ;UPDATE STALL TIME
6477 006542 023737 001370 001320 CMP 2#MXSTAL,2#STALLO ;TIME TOO BIG?
6478 006550 003347 BGT 4$ ;NO--BRANCH
6479 006552 005037 001320 CLR 2#STALLO ;YES--SET STALL TIME BACK TO ZERO
6480 006556 163701 002162 6$: SUB 2#IC4,R1 ;NEXT CYLINDER
6481 006562 020137 002156 CMP R1,2#FC4 ;DONE?
6482 006566 002373 BGE 6$ ;NO--BRANCH
6483 006570 000004 EXIT4: SCOPE ;LOOP

```

```

6484
6497 *****
(3) ;*TEST 5 CONVERGING/DIVERGING SEEK TEST
(4)
(4) ;* THIS TEST WILL CAUSE THE DRIVE TO EXECUTE FORWARD AND REVERSE
(4) ;* SEEKS FROM "NC1" AND "NC2" RESPECTIVELY, "NC1" WILL BE INCREMENTED
(4) ;* BY "IC" AND "NC2" WILL BE DECREMENTED BY "IC" UNTIL "NC1" IS
(4) ;* GREATER THAN THE INITIAL VALUE OF "NC2" AND "NC2" IS

```



```

(3) 007014 013737 001102 177570      MOV      $STNM,2#DISPLAY      ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
6527 007022 113737 002222 003470      MOVB     2#FS6,2#DPB.B+10    ;FS
6528 007030 113737 002220 003471      MOVB     2#FT6,2#DPB.B+11    ;FT
6529 007036 012737 007170 001234      MOV      #EXIT6,2#BYPASS    ;GO TO EXIT6 ON ERROR
6530 007044 012706 001100      TEST6:  MOV      #STACK,SP    ;SETUP STACK
6531 007050 013701 002212      MOV      2#FC6,R1           ;PICKUP "FC"
6532 007054 013702 002214      MOV      2#LC6,R2           ;FORM LAST CYLINDER THAT
6533 007060 162702 000005      SUB      #5,R2              ;IS AVAILABLE FOR TESTING
6534 007064 020102      IS:      CMP      R1,R2         ;LAST CYLINDER
6535 007066 003040      BGT      EXIT6             ;YES--BRANCH
6536 007070 010137 003472      MOV      R1,2#DPB.B+12      ;NC
6537 007074 004037 020010      JSR      RO,2#CALL.B        ;GO EXECUTE THE COMMAND
6538 007100 062737 000004 003472      ADD      #4,2#DPB.B+12      ;NC+4
6539 007106 004037 020010      JSR      RO,2#CALL.B        ;GO EXECUTE THE COMMAND
6540 007112 162737 000003 003472      SUB      #3,2#DPB.B+12      ;NC+1
6541 007120 004037 020010      JSR      RO,2#CALL.B        ;GO EXECUTE THE COMMAND
6542 007124 062737 000002 003472      ADD      #2,2#DPB.B+12      ;NC+3
6543 007132 004037 020010      JSR      RO,2#CALL.B        ;GO EXECUTE THE COMMAND
6544 007136 162737 000001 003472      SUB      #1,2#DPB.B+12      ;NC+2
6545 007144 004037 020010      JSR      RO,2#CALL.B        ;GO EXECUTE THE COMMAND
6546 007150 062737 000003 003472      ADD      #3,2#DPB.B+12      ;NC+5
6547 007156 004037 020010      JSR      RO,2#CALL.B        ;GO EXECUTE THE COMMAND
6548 007162 063701 002216      ADD      2#IC6,R1
6549 007166 000736      BR      IS
6550 007170 000004      EXIT6:  SCOPE              ;LOOP

```


6590	007320	005005			CLR	R5	:COUNT UP
6591	007322	012703	002330		MOV	#T7A,R3	:60HZ PARAMETERS
6592	007326	032737	000100	001212	BIT	#SW06,@#C.SWR	:60 HZ?
6593	007334	001402			BEQ	TEST7	:YES--BRANCH
6594	007336	012703	002340		MOV	#T7B,R3	:NO--50 HZ PARAMETERS
6595	007342	012706	001100		TEST7:	MOV	:SETUP STACK
6596	007346	012701	000012		MOV	#ID1J,R1	:TIME 10 SEARCHES
6597	007352	004737	021542		JSR	PC,@#STATMR	:INITIALIZE THE TIMERS
6598	007356	012777	007564	171750	MOV	#7\$,@PKV	:SETUP VECTOR IN CASE OF OVERFLOW
6599	007364	012777	021540	017576	MOV	#DOATI,@RVEC	:SETUP RPO4 VECTOR
6600	007372	005077	171744		15:	CLR	:START COUNTING AT ZERO
6601	007376	012777	000131	171734	MOV	#131,@PKCS	:INT.EN., COUNT UP AT 100KHZ
6602	007404	012714	000131		MOV	#SEARCH,(R4)	:START A SEARCH
6603	007410	000001			WAIT		:WAIT ON INTERRUPT
6604	007412	042777	000101	171720	BIC	#101,@PKCS	:STOP THE CLOCK
6605	007420	032764	040000	000012	BIT	#BIT14,RHDS1(R4)	:ERROR?
6606	007426	001415			BEQ	25	:NO--BRANCH
6607	007430	104416			SAVREG		:SAVE R0-R5
(1)	007432	012702	003520		MOV	#DTADPB,R2	:DPB POINTER
(1)	007436	004737	033434		JSR	PC,@#SVRH11	:SAVE ALL THE RH11/RPO4 REGISTERS
(1)	007442	012764	000040	000010	MOV	#BIT05,RHCS2(R4)	:MASSBUS CLEAR
(1)	007450	013764	003520	000010	MOV	@#DTADPB,RHCS2(R4)	:SELECT DRIVE
(1)	007456	104420			RESREG		:RESTORE R0-R5
6608	007460	104017			ERROR	17	
6609	007462	005077	171654		25:	CLR	:START THE COUNT AT ZERO
6610	007466	012714	000131		MOV	#SEARCH,(R4)	:START A SEARCH
6611	007472	012777	000131	171640	MOV	#131,@PKCS	:START THE CLOCK
6612	007500	000001			WAIT		:WAIT ON INTERRUPT
6613	007502	042777	000101	171630	BIC	#101,@PKCS	:STOP THE CLOCK
6614	007510	032764	040000	000012	BIT	#BIT14,RHDS1(R4)	:IS "ERR=1"?
6615	007516	001415			BEQ	35	:NO--BRANCH
6616	007520	104416			SAVREG		:SAVE R0-R5
(1)	007522	012702	003520		MOV	#DTADPB,R2	:DPB POINTER
(1)	007526	004737	033434		JSR	PC,@#SVRH11	:SAVE ALL THE RH11/RPO4 REGISTERS
(1)	007532	012764	000040	000010	MOV	#BIT05,RHCS2(R4)	:MASSBUS CLEAR
(1)	007540	013764	003520	000010	MOV	@#DTADPB,RHCS2(R4)	:SELECT DRIVE
(1)	007546	104420			RESREG		:RESTORE R0-R5
6617	007550	104017			ERROR	17	:DISK ERROR OCCURRED
6618	007552	004737	021606		35:	JSR	PC,@#COUNT
6619	007556	005301			DEC	R1	:UPDATE THE COUNT
6620	007560	003304			BGT	15	:DONE?
6621	007562	000424			BR	EXIT7	:NO--BRANCH
6622	007564	042777	000101	171546	75:	BIC	:YES--GO TO THE EXIT
6623	007572	005037	177776		CLR	#101,@PKCS	:STOP THE CLOCK
6624	007576	012600			MOV	@#PS	:DROP THE PRIORITY
6625	007600	005726			MOV	(SP)+,R0	:PC OF WAIT+2
6626	007602	104416			TST	(SP)+	:POP THE PS FROM THE STACK
(1)	007604	012702	003520		SAVREG		:SAVE R0-R5
(1)	007610	004737	033434		MOV	#DTADPB,R2	:DPB POINTER
(1)	007614	012764	000040	000010	JSR	PC,@#SVRH11	:SAVE ALL THE RH11/RPO4 REGISTERS
(1)	007622	013764	003520	000010	MOV	#BIT05,RHCS2(R4)	:MASSBUS CLEAR
(1)	007630	104420			MOV	@#DTADPB,RHCS2(R4)	:SELECT DRIVE
6627	007632	104020			RESREG		:RESTORE R0-R5
6628	007634	000004			ERROR	20	:CLOCK OVERFLOWED
					EXIT7:	SCOPE	

```

6629 007636 012764 000040 000010 MOV #BIT05,RHCS2(R4) ; MASSBUS INIT.
(1) 007644 013764 003520 000010 MOV #DTADPB,RHCS2(R4) ; SELECT DRIVE
6630 007652 004737 017360 JSR PC,#ST.CLK ; INITIALIZE THE CLOCK
6631 007656 012777 031356 017304 MOV #ISR,#RVEC ; RESTORE RH11/RP04 INT. VECTOR
6632 007664 032737 000100 001212 BIT #SW06,#C.SWR ; 60 HZ?
6633 007672 001004 BNE IS ; NO -- BRANCH
6634 007674 004037 021740 JSR RC,#TYPTIM ; GO TYPE THE TIMES
(1) 007700 002330 T7A ; POINTER
6635 007702 000403 BR TST10 ; GO TO NEXT TEST
6636 007704 ;S: JSR RC,#TYPTIM ; GO TYPE THE TIMES
(1) 007704 004037 021740 T7B ; POINTER
(1) 007710 002340

6637
6638
6651 ;*****
(3) ;*TEST 10 ONE CYLINDER SEEK TIMING TEST
(4)
(4) ;*
(4) ;* THIS TEST WILL COMMAND FORWARD SEEK CYCLES TO ADVANCE THE
(4) ;* CYLINDER BY ONE UNTIL THE INCREMENT IS GREATER THAN THE
(4) ;* NUMBER OF CYLINDERS TO CYLINDER 410, THEN REVERSE SEEK
(4) ;* CYCLES TO RETRACT THE CYLINDER BY ONE UNTIL THE INCREMENT
(4) ;* IS GREATER THAN THE NUMBER OF CYLINDERS TO CYLINDER 0. THE
(4) ;* TIME TO PERFORM EACH SEEK IS CHECKED TO INSURE IT DOES NOT
(4) ;* EXCEED THE MAXIMUM TIME PERMITTED FOR A ONE CYLINDER SEEK.
(4) ;* THE TIME MUST BE LESS THAN 10MS BUT GREATER THAN 3MS.
(4)
(3) ;*****
(3) TST10:
(4) 007712 033737 001414 001224 BIT #BITS+(10*2),TSTNMS ; DO THIS TEST?
(4) 007720 001002 BNE 64$ ; YES--BRANCH
(4) 007722 000137 010334 JMP TST11 ; NO--GO TO THE NEXT TEST
(4) 007726 013737 002240 001176 64$: MOV #RPT10,$TIMES ; GET THE ITERATION COUNT
(3) 007734 012737 007712 001106 MOV #TST10,#SLPADR ; SETUP THE SCOPE LOOP ADDRESS
(3) 007742 012737 007712 001110 MOV #TST10,#SLPERR ; SETUP THE ERROR LOOP ADDRESS
(3) 007750 012737 000010 001102 MOV #10,#$TSTM ; SET UP TEST NUMBER AND
(3) ; CLEAR THE ERROR FLAG ($ERFLG)
6652 007756 013737 001102 177570 MOV $TSTM,#DISPLAY ; LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
6653 007764 005737 001226 TST #CLKSTA ; KW11-P CLOCK?
6654 007770 003002 BGT IS ; YES--START TEST
6655 007772 000137 010334 JMP TST11 ; NO--GO TO NEXT TEST
6656 007776 004037 021356 1$: JSR RC,#SRCH00 ; DO A MASSBUS INIT. AND RECAL
6657 010002 000401 BR 2$ ; NO ERROR RETURN
6658 010004 000534 BR EXIT10 ; ERROR RETURN--SCOPE LOOP CALL
6659 010006 012703 002350 2$: MOV #T10,R3 ; PARAMETER POINTER
6660 010012 012737 010276 001200 MOV #EXIT10,$ESCAPE ; ESCAPE TO EXIT10 ON ERROR
6661 010020 012706 001100 TEST10: MOV #STACK,$P ; SETUP STACK
6662 010032 005005 CLR #1,#DTADPB+12 ; START WITH CYLINDER #1
6663 010034 004737 021542 R5 ; SET THE UP/DOWN SWITCH TO UP
6664 010040 012777 010226 171266 JSR PC,#STRTMR ; INITIALIZE THE TIMERS
6665 010046 012777 021540 017114 MOV #7,$PKV ; SETUP INCASE OF OVERFLOW
6666 010054 005077 171262 1$: CLR #DORTI,#RVEC ; SET RP04 VECTOR
6667 010060 013764 003532 000034 MOV #PKB ; START THE COUNTER AT ZERO
6668 010066 012714 000105 MOV #DTADPB+12,RHCA(R4) ; LOAD DESIRED CYLINDER
MOV #SEEK,(R4) ; START A SEEK

```

```

6669 010072 012777 000131 171240 MOV #131,DPKCS ;START THE CLOCK
6670 010100 000001 WAIT ;WAIT ON INTERRUPT
6671 010102 042777 000101 171230 BIC #101,DPKCS ;STOP THE CLOCK
6672 010110 032764 040000 000012 BIT #BIT14,RHDS1(R4) ;ANY DISK ERRORS?
6673 010116 001415 BEQ 25 ;NO--BRANCH
6674 010120 104416 SAVREG ;SAVE R0-R5
(1) 010122 012702 003520 MOV #DTADPB,R2 ;DPB POINTER
(1) 010126 004737 033434 JSR PC,#SVRH11 ;SAVE ALL THE RH11/RP04 REGISTERS
(1) 010132 012764 000040 000010 MOV #BIT05,RHCS2(R4) ;MASSBUS CLEAR
(1) 010140 013764 003520 000010 MOV #DTADPB,RHCS2(R4) ;SELECT DRIVE
(1) 010146 104420 RESREG ;RESTORE R0-R5
6675 010150 104017 ERROR 17 ;REPORT THE ERROR
6676 010152 004737 021606 25: JSR PC,#COUNT ;COUNT THIS SEEKS TIME
6677 010156 004737 021166 JSR PC,#TWOMS ;STALL FOR 2 MILLISECONDS
6678 010162 005705 TST R5 ;UP OR DOWN?
6679 010164 001011 BNE 45 ;DOWN--BRANCH
6680 010166 005237 003532 35: INC #DTADPB+12 ;MOVE TO NEXT CYLINDER
6681 010172 023727 003532 000632 CMP #DTADPB+12,#104 ;OUT OF CYLINDERS?
6682 010200 002725 BLT 15 ;NO--GO DO THE NEXT SEEK
6683 010202 012705 177777 MOV #-1,R5 ;SET UP/DOWN SWITCH TO DOWN
6684 010206 000722 BR 15 ;GO DO THE NEXT SEEK
6685 010210 005337 003532 45: DEC #DTADPB+12 ;MOVE TO NEXT CYLINDER
6686 010214 023727 003532 000000 CMP #DTADPB+12,#0 ;OUT OF CYLINDERS?
6687 010222 003314 BGT 15 ;NO--GO DO THE NEXT SEEK
6688 010224 000424 BR EXIT10 ;GO TO THE EXIT
6689 010226 042777 000101 171104 75: BIC #101,DPKCS ;STOP THE CLOCK
6690 010234 005037 177776 CLR #PS ;DROP THE PRIORITY
6691 010240 012600 MOV (SP)+,R0 ;PC OF WAIT+2
6692 010242 005726 TST (SP)+ ;POP THE PS FROM THE STACK
6693 010244 104416 SAVREG ;SAVE R0-R5
(1) 010246 012702 003520 MOV #DTADPB,R2 ;DPB POINTER
(1) 010252 004737 033434 JSR PC,#SVRH11 ;SAVE ALL THE RH11/RP04 REGISTERS
(1) 010256 012764 000040 000010 MOV #BIT05,RHCS2(R4) ;MASSBUS CLEAR
(1) 010264 013764 003520 000010 MOV #DTADPB,RHCS2(R4) ;SELECT DRIVE
(1) 010272 104420 RESREG ;RESTORE R0-R5
6694 010274 104020 ERROR 20 ;REPORT CLOCK OVERFLOW
6695 010276 000004 EXIT10: SCOPE
6696 010300 012764 000040 000010 MOV #BIT05,RHCS2(R4) ;MASSBUS INIT.
(1) 010306 013764 003520 000010 MOV #DTADPB,RHCS2(R4) ;SELECT DRIVE
6697 010314 004737 017360 JSR PC,#ST.CLK ;INITIALIZE THE CLOCK
6698 010320 012777 031356 016642 MOV #ISR,DPVEC ;RESTORE RH11/RP04 INT. VECTOR
6699 010326 004037 021740 JSR R0,#TYPTIM ;GO TYPE THE TIMES
(1) 010332 002350 T10 ;POINTER

```

6700
6701
6712
(3)
(4)
(4)
(4)
(4)
(4)
(4)
(4)

;TEST 11 AVERAGE SEEK TIMING TEST

;* THIS TEST WILL COMMAND A FORWARD SEEK FROM CYLINDER 0 TO
;* CYLINDER 136, THEN A REVERSE SEEK FROM CYLINDER 136 TO
;* CYLINDER 0. BOTH SEEKS ARE TIMED AND CHECKED TO INSURE THEY
;* ARE WITHIN THE TOLERANCE ALLOWED FOR THE AVERAGE SEEK TIME.
;* THIS SEQUENCE IS REPEATED 128 TIMES (FOR A TOTAL
;* OF 256 SEEKS). THE AVERAGE SEEK TIME IS 28 MS + OR - 2MS.

```

(4)
(3)
(3) 010334
(4) 010334 033737 001416 001224
(4) 010342 001002
(4) 010344 000137 011034
(4) 010350 013737 002244 001176
(3) 010356 012737 010334 001106
(3) 010364 012737 010334 001110
(3) 010372 012737 000011 001102
(3)
(3) 010400 013737 001102 177570
6713 010406 005737 001226
6714 010412 003002
6715 010414 000137 011034
6716 010420 004037 021356
6717 010424 000401
6718 010426 000563
6719 010430 012703 002360
6720 010434 012737 010776 001200
6721 010442 012706 001100
6722 010446 012701 000200
6723 010452 004737 021542
6724 010456 012777 010726 170650
6725 010464 012777 021540 016476
6726 010472 005077 170644
6727 010476 012764 000210 000034
6728 010504 012764 030105 000000
6729 010512 012777 030131 170620
6730 010520 000001
6731 010522 042777 000101 170610
6732 010530 032764 040000 000012
6733 010536 001415
6734 010540 104416
(1) 010542 012702 003520
(1) 010546 004737 033434
(1) 010552 012764 000040 000010
(1) 010560 013764 003520 000010
(1) 010566 104420
6735 010570 104017
6736 010572 005005
6737 010574 004737 021606
6738 010600 004737 021166
6739 010604 005077 170532
6740 010610 012764 000000 000034
6741 010616 012764 000105 000000
6742 010624 012777 000131 170506
6743 010632 000001
6744 010634 042777 000101 170476
6745 010642 032764 040000 000012
6746 010650 001415
6747 010652 104416
(1) 010654 012702 003520
(1) 010660 004737 033434

```

```

:*****
†ST11:
      BIT      2#BITS+(11*2),TSTNMS ;DO THIS TEST?
      BNE      64$ ;YES--BRANCH
      JMP      TST12 ;NO--GO TO THE NEXT TEST
64$:   MOV      2#RPT11,$TIMES ;GET THE ITERATION COUNT
      MCV      2#TST11,2#SLPADR ;SETUP THE SCOPE LOOP ADDRESS
      MOV      2#TST11,2#SLPERR ;SETUP THE ERROR LOOP ADDRESS
      MOV      2#11,2#$TSTNM ;SET UP TEST NUMBER AND
      ;CLEAR THE ERROR FLAG (SERFLG)
      MOV      2#STSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
      TST      2#CLKSTA ;KW11-P CLOCK?
      BGT      1$ ;YES--START TEST
      JMP      TST12 ;NO--GO TO NEXT TEST
1$:   JSR      2#RO,2#SRCH00 ;DO A MASSBUS INIT & RECAL
      BR      2$ ;RETURN HERE IF NO ERROR
      BR      EXIT11 ;RETURN HERE ON ERROR
2$:   MOV      2#T11,R3 ;PARAMETER POINTER
      MOV      2#EXIT11,$ESCAPE ;ESCAPE TO EXIT11 ON ERROR
      MOV      2#TEST11,$SP ;SETUP STACK
      JSR      2#PC,2#STRMTR ;REPEAT "0-136-0" 128 TIMES
      MOV      2#7$,2#PKV ;INIT. THE COUNTERS
      MOV      2#DOATI,2#RVEC ;SET UP VECTOR IN CASE OF OVERFLOW
15$:  CLR      2#PKB ;SETUP RPO4 VECTOR
      MOV      2#101,2#PKCS ;START COUNT AT ZERO
      MOV      2#136,RHCA(R4) ;CYLINDER=136
      MOV      2#SEEK,RHCS1(R4) ;START A SEEK
      MOV      2#131,2#PKCS ;START THE CLOCK
      WAIT ;WAIT ON INTERRUPT
      BIC      2#101,2#PKCS ;STOP CLOCK
      BIT      2#BIT14,RHCS1(R4) ;ERR=1?
      BEQ      2$ ;NO--BRANCH
      SAVREG ;SAVE R0-R5
      MOV      2#DTADPB,R2 ;DPB POINTER
      JSR      2#PC,2#SVRH11 ;SAVE ALL THE RH11/RPO4 REGISTERS
      MOV      2#BIT05,RHCS2(R4) ;MASSBUS CLEAR
      MOV      2#DTADPB,RHCS2(R4) ;SELECT DRIVE
      RESREG ;RESTORE R0-R5
      ERROR 17
2$:   CLR      R5 ;SET UP/DOWN SWITCH TO UP
      JSR      2#PC,2#COUNT ;UPDATE THE COUNT
      JSR      2#PC,2#TWOMS ;STALL FOR 2 MILLISECONDS
      CLR      2#PKB ;START THE COUNT AT ZERO
      MOV      2#0,RHCA(R4) ;CYLINDER=0
      MOV      2#SEEK,RHCS1(R4) ;START A SEEK
      MOV      2#131,2#PKCS ;START THE CLOCK
      WAIT ;WAIT ON INTERRUPT
      BIC      2#101,2#PKCS ;STOP THE CLOCK
      BIT      2#BIT14,RHCS1(R4) ;ERR=1?
      BEQ      3$ ;NO--BRANCH
      SAVREG ;SAVE R0-R5
      MOV      2#DTADPB,R2 ;DPB POINTER
      JSR      2#PC,2#SVRH11 ;SAVE ALL THE RH11/RPO4 REGISTERS

```



```

(1) 010664 012764 000040 000010      MOV      #BIT05,RHCS2(R4) ;MASSBUS CLEAR
(1) 010672 013764 003520 000010      MOV      @#DTADPB,RHCS2(R4) ;SELECT DRIVE
(1) 010700 104420                      RESREG                      ;RESTORE RO-R5
6748 010702 104017                      ERROR      17
6749 010704 012705 177777          3$:  MOV      #-1,R5 ;SET UP/DOWN SWITCH TO DOWN
6750 010710 004737 021606          JSR      PC,@#COUNT ;UPDATE THE COUNT
6751 010714 004737 021166          JSR      PC,@#TWOMS ;STALL FOR 2 MILLISECOND
6752 010720 005301                      DEC      R1 ;DONE?
6753 010722 003263                      BGT      IS ;NO--BRANCH
6754 010724 000424                      BR       EXIT11 ;YES--EXIT
6755 010726 042777 000101 170404 7$: BIC      #101,@PKCS ;STOP THE CLOCK
6756 010734 005037 177776          CLR      @#PS ;DROP THE PRIORITY
6757 010740 012600                      MOV      (SP)+,RO ;PC OF WAIT+2
6758 010742 005726                      TST     (SP)+ ;POP THE PS FROM THE STACK
6759 010744 104416                      SAVREG                      ;SAVE RO-R5
(1) 010746 012702 003520      MOV      @DTADPB,R2 ;DPB POINTER
(1) 010752 004737 033434      JSR      PC,@#SVRH11 ;SAVE ALL THE RH11/RPO4 REGISTERS
(1) 010756 012764 000040 000010      MOV      #BIT05,RHCS2(R4) ;MASSBUS CLEAR
(1) 010764 013764 003520 000010      MOV      @#DTADPB,RHCS2(R4) ;SELECT DRIVE
(1) 010772 104420                      RESREG                      ;RESTORE RO-R5
6760 010774 104020                      ERROR      20 ;CLOCK OVERFLOWED
6761 010776 000004                      EXIT11: SCOPE
6762 011000 012764 000040 000010      MOV      #BIT05,RHCS2(R4) ;MASSBUS INIT.
(1) 011006 013764 003520 000010      MOV      @#DTADPB,RHCS2(R4) ;SELECT DRIVE
6763 011014 004737 017350      JSR      PC,@#ST.CLK ;INITIALIZE THE CLOCK
6764 011020 012777 031356 016142      MOV      #ISR,@RPVEC ;RESTORE RH11/RPO4 INT. VECTOR
6765 011026 004037 021740      JSR      RO,@#TYPTIM ;GO TYPE THE TIMES
(1) 011032 002360      T11 ;POINTER
6766
6777
(3) ;*****
(4) ;*TEST 12 MAXIMUM SEEK TIMING TEST
(4) ;*
(4) ;* THIS TEST WILL COMMAND A FORWARD SEEK FROM CYLINDER 0 TO
(4) ;* CYLINDER 410, THEN A REVERSE SEEK FROM CYLINDER 410 TO
(4) ;* CYLINDER 0. BOTH SEEKS ARE TIMED AND CHECKED TO INSURE
(4) ;* THEY ARE WITHIN THE TOLERANCE ALLOWED FOR THE MAXIMUM SEEK
(4) ;* TIME. THIS SEQUENCE IS REPEATED 128 TIMES (FOR
(4) ;* A TOTAL OF 256 SEEKS). THE MAXIMUM SEEK TIME IS 50MS + 4MS OR - 2MS.
(3) ;*****
(3) †ST12:
(4) 011034 033737 001420 001224      BIT      @#BITS+(12*2),TSTNMS ;DO THIS TEST?
(4) 011042 001002                      BNE      64$ ;YES--BRANCH
(4) 011044 000137 011534                      JMP      TST13 ;NO--GO TO THE NEXT TEST
(4) 011050 013737 002250 001176 64$: MOV      @#RPT12,$TIMES ;GET THE ITERATION COUNT
(3) 011056 012737 011034 001106      MOV      @#TST12,@#SLPADR ;SETUP THE SCOPE LOOP ADDRESS
(3) 011064 012737 011034 001110      MOV      @#TST12,@#SLPERR ;SETUP THE ERROR LOOP ADDRESS
(3) 011072 012737 000012 001102      MOV      #12,@#TSTNM ;SET UP TEST NUMBER AND
(3) ; CLEAR THE ERROR FLAG (SERFLG)
(3) 011100 013737 001102 177570      MOV      $TSTNM,@#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
6778 011106 005737 001226      TST     @#CLKSTA ;KW11-P CLOCK
6779 011112 003002                      BGT      IS ;YES--START TEST
6780 011114 000137 011534                      JMP      TST13 ;NO--GO TO NEXT TEST
6781 011120 004037 021356          1$:  JSR      RO,@#SRCH00 ;DO A MASSBUS INIT & RECAL

```


6782	011124	000401			BR	25	:RETURN HERE IF NO ERROR
6783	011126	000563			BR	EXIT12	:RETURN HERE ON ERROR
6784	011130	012703	002370		25: MOV	#T12,R3	:PARAMETER POINTER
6785	011134	012737	011476	001200	MOV	#EXIT12,\$ESCAPE	:ESCAPE TO EXIT12 ON ERROR
6786	011142	012706	001100		TEST12: MOV	#STACK,\$P	:SETUP STACK
6787	011146	012701	000200		MOV	#D128,R1	:REPEAT "0-410-0" 128 TIMES
6788	011152	004737	021542		JSR	PC,#STARTMR	:INIT. THE TIMERS
6789	011156	012777	011426	170150	MOV	#7,\$,PKV	:SETUP VECTOR IN CASE OF OVERFLOW
6790	011164	012777	021540	015776	MOV	#DORTI,\$RVEC	:SETUP RPO4 VECTOR
6791	011172	005077	170144		15: CLR	\$PKB	:START COUNTING FROM ZERO
6792	011176	012764	000632	000034	MOV	#410.,RHCA(R4)	:CYLINDER=410
6793	011204	012764	000105	000000	MOV	#SEEK,RHCS1(R4)	:START A SEEK
6794	011212	012777	000131	170120	MOV	#131,\$PKCS	:START THE CLOCK
6795	011220	000001			WAIT		:WAIT ON INTERRUPT
6796	011222	042777	000101	170110	BIC	#101,\$PKCS	:STOP THE CLOCK
6797	011230	032764	040000	000012	BIT	#BIT14,RHDS1(R4)	:ERR=1?
6798	011236	001415			BEQ	25	:NO--BRANCH
6799	011240	104416			SAVREG		:SAVE R0-R5
(1)	011242	012702	003520		MOV	#DTADPB,R2	:DPB POINTER
(1)	011246	004737	033434		JSR	PC,\$SVRH11	:SAVE ALL THE RH11/RPO4 REGISTERS
(1)	011252	012764	000040	000010	MOV	#BIT05,RHCS2(R4)	:MASSBUS CLEAR
(1)	011260	013764	003520	000010	MOV	#DTADPB,RHCS2(R4)	:SELECT DRIVE
(1)	011266	104420			RESREG		:RESTORE R0-R5
6800	011270	104017			ERROR	17	
6801	011272	005005			25: CLR	R5	:SET THE UP/DOWN SWITCH TO UP
6802	011274	004737	021606		JSR	PC,\$COUNT	:UP THE COUNT
6803	011300	004737	021166		JSR	PC,\$TWOMS	:STALL FOR 2 MILLISECOND
6804	011304	005077	170032		CLR	\$PKB	:START COUNT AT ZERO
6805	011310	012764	000000	000034	MOV	#0,RHCA(R4)	:CYLINDER=0
6806	011316	012764	000105	000000	MOV	#SEEK,RHCS1(R4)	:START A SEEK
6807	011324	012777	000131	170006	MOV	#131,\$PKCS	:START THE CLOCK
6808	011332	000001			WAIT		:WAIT ON INTERRUPT
6809	011334	042777	000101	167776	BIC	#101,\$PKCS	:STOP THE CLOCK
6810	011342	032764	040000	000012	BIT	#BIT14,RHDS1(R4)	:ERR=1?
6811	011350	001415			BEQ	35	:NO--BRANCH
6812	011352	104416			SAVREG		:SAVE R0-R5
(1)	011354	012702	003520		MOV	#DTADPB,R2	:DPB POINTER
(1)	011360	004737	033434		JSR	PC,\$SVRH11	:SAVE ALL THE RH11/RPO4 REGISTERS
(1)	011364	012764	000040	000010	MOV	#BIT05,RHCS2(R4)	:MASSBUS CLEAR
(1)	011372	013764	003520	000010	MOV	#DTADPB,RHCS2(R4)	:SELECT DRIVE
(1)	011400	104420			RESREG		:RESTORE R0-R5
6813	011402	104017			ERROR	17	:REPORT THE ERROR
6814	011404	012705	177777		35: MOV	#-1,R5	:SET THE UP/DOWN SWITCH TO DOWN
6815	011410	004737	021606		JSR	PC,\$COUNT	:UPDATE THE COUNT
6816	011414	004737	021166		JSR	PC,\$TWOMS	:STALL FOR 2 MILLISECOND
6817	011420	005301			DEC	R1	:DONE?
6818	011422	003263			BGT	15	:NO--BRANCH
6819	011424	000424			BR	EXIT12	:YES--EXIT
6820	011426	042777	000101	167704	75: BIC	#101,\$PKCS	:STOP THE CLOCK
6821	011434	005037	177776		CLR	\$PS	:DROP THE PRIORITY
6822	011440	012600			MOV	(SP)+,R0	:PC OF WAIT+2
6823	011442	005726			TST	(SP)+	:POP THE PS FROM THE STACK
6824	011444	104416			SAVREG		:SAVE R0-R5
(1)	011446	012702	003520		MOV	#DTADPB,R2	:DPB POINTER

(1)	011452	004737	033434		JSR	PC,@SVRH11	;SAVE ALL THE RH11/RP04 REGISTERS
(1)	011456	012764	000040	000010	MOV	#BIT05,RHCS2(R4)	;MASSBUS CLEAR
(1)	011464	013764	003520	000010	MOV	@DTADPB,RHCS2(R4)	;SELECT DRIVE
(1)	011472	104420			RESREG		;RESTORE R0-R5
6825	011474	104020			ERROR	20	;CLOCK OVERFLOWED
6826	011476	000004			EXIT12:	SCOPE	
6827	011500	012764	000040	000010	MOV	#BIT05,RHCS2(R4)	;MASSBUS INIT.
(1)	011506	013764	003520	000010	MOV	@DTADPB,RHCS2(R4)	;SELECT DRIVE
6826	011514	004737	017363		JSR	PC,@ST.CLK	;INITIALIZE THE CLOCK
6829	011520	012777	031356	015442	MOV	#ISR,@RPVEC	;RESTORE RH11/RP04 INT. VECTOR
6830	011526	004037	021740		JSR	R0,@TYPTIM	;GO TYPE THE TIMES
(1)	011532	002370			T12		;POINTER


```

6870 011716 004037 020320 JSR RO, @#DRVCAL ;START A DATA TRANSFER
6871 011722 004037 022462 JSR RO, @#CKSCTR ;CHECK THE SECTOR DATA READ
6872 011726 012700 040524 MOV #BUFFER, RO ;BUFFER ADDRESS
6873 011732 005001 CLR R1 ;FIRST SECTOR
6874 011734 012737 000161 003522 15: MOV #WRITE, @#DTADPB+2 ;COMMAND=WRITE DATA
6875 011742 012737 177400 003524 MOV #SCTRWC, @#DTADPB+4 ;WORD COUNT
6876 011750 010037 003526 MOV RO, @#DTADPB+6 ;BUFFER ADDRESS
6877 011754 110137 003530 MOV R1, @#DTADPB+10 ;SECTOR
6878 011760 004037 020320 JSR RO, @#DRVCAL ;START A DATA TRANSFER
6879 011764 012737 000151 003522 MOV #WRCKD, @#DTADPB+2 ;COMMAND=WRITE CHECK DATA
6880 011772 012737 165000 003524 MOV #TRCKWC, @#DTADPB+4 ;WORD COUNT
6881 012000 012737 040524 003526 MOV #BUFFER, @#DTADPB+6 ;BUFFER ADDRESS
6882 012006 105037 003530 CLR R1 ;SECTOR
6883 012012 004037 020320 JSR RO, @#DRVCAL ;START A DATA TRANSFER
6884 012016 062700 001000 ADD #D256*2, RO ;MOVE TO NEXT SECTOR
6885 012022 005201 INC R1
6886 012024 022701 000026 CMP #D22, R1 ;DONE?
6887 012030 00334: BGT 15 ;NO--BRANCH
6888 012032 000004 EXIT13: SCOPE ;LOOP
6889
6903 ;*****
(3) ;*TEST 14 TRACK ADDRESSING TEST
(4)
(4) ;* THIS TEST WILL WRITE DATA IN THE FORM OF TRACK ADDRESSES
(4) ;* IN CYLINDER "FC" SECTOR "FS" OF EVERY TRACK WITH EACH TRACK
(4) ;* GETTING ITS OWN TRACK ADDRESS.
(4) ;* A WRITE CHECK IS THEN PERFORMED ON EACH TRACK TO INSURE
(4) ;* THE DATA IS VALID. THEN TRACK 0 IS REWRITTEN AND TRACK 1
(4) ;* THROUGH TRACK 18 IS WRITE CHECKED. THEN TRACK 1 IS
(4) ;* REWRITTEN AND TRACK 2 THROUGH TRACK 18 IS WRITE CHECKED.
(4) ;* THIS PROCEDURE IS CONTINUED UP THROUGH REWRITING TRACK 17
(4) ;* AND WRITE CHECKING TRACK 18.
(4)
(3) ;*****
(3) 012034 †TST14:
(4) 012034 033737 001424 001224 BIT @#BITS+(14*2), TSTNMS ;DO THIS TEST?
(4) 012042 001002 BNE 645 ;YES--BRANCH
(4) 012044 000137 012370 JMP TST15 ;NO--GO TO THE NEXT TEST
(4) 012050 013737 002264 001176 645: MOV @#RPT14, $TIMES ;GET THE ITERATION COUNT
(3) 012056 012737 012114 001106 MOV #TEST14, @#SLPADR
(3) 012064 012737 012034 001110 MOV #TST14, @#SLPERR ;SETUP THE ERROR LOOP ADDRESS
(3) 012072 012737 000014 001102 MOV #14, @#$TSTNM ;SET UP TEST NUMBER AND
;CLEAR THE ERROR FLAG (SERFLG)
(3) 012100 013737 001102 177570 MOV $TSTNM, @#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
6904 012106 012737 012366 001234 TEST14: MOV #EXIT14, @#BYPASS
6905 012114 012706 001100 MOV #STACK, $P ;SET THE STACK POINTER
6906 012120 004737 022346 JSR PC, @#FILBUF ;FILL THE BUFFER WITH TRACK ADDRESS
6907 012124 012737 000161 003522 MOV #WRITE, @#DTADPB+2 ;COMMAND=WRITE DATA
6908 012132 013737 002266 003532 MOV @#FC14, @#DTADPB+12 ;CYLINDER
6909 012140 113737 002270 003530 MOV R1, @#FS14, @#DTADPB+10 ;SECTOR
6910 012146 012737 177400 003524 MOV #SCTRWC, @#DTADPB+4 ;WORD COUNT
6911 012154 012737 040524 003526 MOV #BUFFER, @#DTADPB+6 ;BUFFER ADDRESS
6912 012162 005000 CLR RO ;TRACK=0
6913 012164 110037 003531 15: MOV R0, @#DTADPB+11 ;TRACK ADDRESS

```

6914	012170	004037	020320			JSR
6915	012174	062737	001000	003526		ADD
6916	012202	005200				INC
6917	012204	022700	000023			CMP
6918	012210	003365				BGT
6919	012212	012737	040524	003526		MOV
6920	012220	005000				CLR
6921	012222	012737	000151	003522		MOV
6922	012230	110037	003531		2\$:	MOVB
6923	012234	004037	020320			JSR
6924	012240	062737	001000	003526		ADD
6925	012246	005200				INC
6926	012250	022700	000023			CMP
6927	012254	003365				BGT
6928	012256	005000				CLR
6929	012260	110037	003531		3\$:	MOVB
6930	012264	010001				MOV
6931	012266	012737	040524	003526		MOV
6932	012274	005301			4\$:	DEC
6933	012276	002404				BLT
6934	012300	062737	001000	003526		ADD
6935	012306	000772				BR
6936	012310	012737	000161	003522	5\$:	MOV
6937	012316	004037	020320			JSR
6938	012322	062737	001000	003526	6\$:	ADD
6939	012330	105237	003531			INCB
6940	012334	012737	000151	003522		MOV
6941	012342	004037	020320			JSR
6942	012346	122737	000022	003531		CMPB
6943	012354	003362				BGT
6944	012356	005200				INC
6945	012360	022700	000022			CMP
6946	012364	003335				BGT
6947	012366	000004				EXIT14: SCOPE

```

RO, @#DRVCL          :START A DATA TRANSFER
#↑D256*2, @#DTADPB+6 :UPDATE BUFFER ADDRESS
RO                   :UPDATE TRACK NUMBER
#↑D19, RO            :OUT OF TRACKS?
IS                   :NO--BRANCH
#BUFFER, @#DTADPB+6 ;BUFFER ADDRESS
RO
#WRCKD, @#DTADPB+2  :COMMAND=WRITE CHECK
RO, @#DTADPB+11     :TRACK ADDRESS
RO, @#DRVCL         :START A DATA TRANSFER
#↑D256*2, @#DTADPB+6 :UPDATE BUFFER ADDRESS
RO                   :UPDATE TRACK NUMBER
#↑D19, RO            :OUT OF TRACKS?
2$                   :NO--BRANCH
RO                   :FIRST TRACK ADDRESS
RO, @#DTADPB+11     :TRACK
RO, R1              :FORM BUFFER ADDRESS
#BUFFER, @#DTADPB+6 ;BUFFER ADDRESS
R1
5$
#↑D256*2, @#DTADPB+6
4$
#WRITE, @#DTADPB+2  :COMMAND=WRITE DATA
RO, @#DRVCL         :START A DATA TRANSFER
#↑D256*2, @#DTADPB+6 :UPDATE BUFFER ADDRESS
@#DTADPB+11         :MOVE TO NEXT TRACK
#WRCKD, @#DTADPB+2  :COMMAND=WRITE CHECK DATA
RO, @#DRVCL         :START A DATA TRANSFER
#↑D18, @#DTADPB+11 :OUT OF TRACKS?
6$                   :NO--BRANCH
RO                   :NEXT TRACK TO WRITE
#↑D19, RO            :OUT OF TRACKS?
3$                   :NO--BRANCH

```



```

(4)      ;*      133333 001000 176777 155555 177677 066667 177777 000000
(4)      ;*      165555 002000 175777 172666 177737 153333 177777 000000
(4)      ;*      133333 004000 173777 155555 177757 066667 177777 000000
(4)      ;*      165555 010000 167777 172666 177767 153333 177777 000000
(4)      ;*      133333 020000 157777 155555 177773 066667 177777 000000
(4)      ;*      165555 040000 137777 172666 177775 153333 177777 000000
(4)      ;*      133333 100000 077777 155555 177776 066667 177777 000000
(4)      ;*
(4)
(4)
(3)

```

```

(3) 012370          †ST15:
(4) 012370 033737 001426 001224      BIT      2#BITS+(15*2),TSTNMS ;DO THIS TEST?
(4) 012376 001002          BNE      64$      ;YES--BRANCH
(4) 012400 000137 013030      JMP      TST16      ;NO--GO TO THE NEXT TEST
(4) 012404 013737 002274 001176 64$: MOV      2#RPT15,$TIMES ;GET THE ITERATION COUNT
(3) 012412 012737 012554 001106      MOV      2#TEST15,2#SLPADR
(3) 012420 012737 012370 001110      MOV      2#TST15,2#SLPERR ;SETUP THE ERROR LOOP ADDRESS
(3) 012426 012737 000015 001102      MOV      15,2#$TSTNM ;SET UP TEST NUMBER AND
(3)          ;CLEAR THE ERROR FLAG (SERFLG)
(3) 012434 013737 001102 177570      MOV      $TSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
7013 012442 005000          CLR      R0 ;CLEAR SWITCH
7014 012444 005004          CLR      R4 ;FORM WORD COUNT IN R4
7015 012446 013701 002314      MOV      2#LS15,R1
7016 012452 163701 002312      SUB      2#FS15,R1
7017 012456 002003          BGE      1$ ;BRANCH IF FS < OR = LS
7018 012460 062701 000026      ADD      2#D22,R1 ;MAKE DIFFERENCE POSITIVE
7019 012464 005100          COM      R0 ;SET SWITCH
7020 012466 062704 000400 1$: ADD      2#D256,R4
7021 012472 005301          DEC      R1
7022 012474 002374          BGE      1$
7023 012476 005404          NEG      R4
7024 012500 010405          MOV      R4,R5 ;COPY NORMAL WORD COUNT INTO SMALL WC
7025 012502 005700          TST      R0 ;SWITCH SET?
7026 012504 001412          BEQ      3$ ;NO--BRANCH
7027 012506 005005          CLR      R5 ;FORM WORD COUNT FOR LS < FS
7028 012510 012701 000025      MOV      2#D21,R1
7029 012514 163701 002312      SUB      2#FS15,R1
7030 012520 062705 000400 2$: ADD      2#D256,R5
7031 012524 005301          DEC      R1
7032 012526 002374          BGE      2$
7033 012530 005405          NEG      R5
7034 012532 113737 002312 003530 3$: MOV      2#FS15,2#DTADPB+10 ;SECTOR
7035 012540 012737 040524 003526      MOV      2#BUFFER,2#DTADPB+6 ;DATA BUFFER
7036 012546 012737 013026 001234      MOV      2#EXIT15,2#BYPASS
7037 012554 005037 001316      TEST15: CLR      2#WCEFLG ;CLEAR THE WRITE CHECK ERROR FLAG
7038 012560 013701 002276      MOV      2#FC15,R1 ;PICKUP FIRST CYLINDER
7039 012564 000407          BR      2$
7040 012566 005720 1$: TST      (R0)+ ;MOVE TO NEXT DATA PATTERN
7041 012570 022700 000040      CMP      2#D16*2,R0 ;OUT OF PATTERNS?
7042 012574 003004          BGT      3$ ;NO--BRANCH
7043 012576 004037 022316      JSR      R0,2#INCCYL ;MOVE TO NEXT CYLINDER
7044 012602 000511          BR      EXIT15 ;OUT OF CYLINDERS
7045 012604 005000 2$: CLR      R0 ;START WITH PATTERN G
7046 012606 036037 001374 002316 3$: BIT      BITS(R0),2#PTRN15 ;THIS PATTERN SELECTED?

```

7047	012614	001764				BEQ	1\$: NO--BRANCH
7048	012616	013702	002304			MOV	2#FT15, R2		: FIRST TRACK
7049	012622	010137	003532			MOV	R1, 2#DTADPB+12		: CYLINDER
7050	012626	110237	003531	4\$:		MOV8	R2, 2#DTADPB+11		: TRACK
7051	012632	010437	003524			MOV	R4, 2#DTADPB+4		: WORD COUNT
7052	012636	022701	000632			CMP	#410., R1		: LAST DISK CYLINDER?
7053	012642	003005				BGT	5\$: NO--BRANCH
7054	012644	022702	000022			CMP	#18., R2		: LAST DISK TRACK?
7055	012650	003002				BGT	5\$: NO--BRANCH
7056	012652	010537	003524			MOV	R5, 2#DTADPB+4		: SHORT WORD COUNT
7057	012656	013703	177570	5\$:		MOV	2#SWR, R3		: INHIBIT WRITE AND
7058	012662	005103				COM	R3		: WRITE CHECK?
7059	012664	032703	000030			BIT	#SW04!SW03, R3		
7060	012670	001424				BEQ	7\$: YES--BRANCH
7061	012672	004737	023036			JSR	PC, 2#SETBUF		: MOVE DATA PATTERN INTO THE BUFFER
7062	012676	032737	000020	177570		BIT	#SW04, 2#SWR		: INHIBIT WRITE?
7063	012704	001005				BNE	6\$: YES--BRANCH
7064	012706	012737	000161	003522		MOV	#WRITE, 2#DTADPB+2		: COMMAND=WRITE DATA
7065	012714	004037	020320			JSR	RO, 2#DRVCL		: START A DATA TRANSFER
7066	012720	032737	000010	177570	6\$:	BIT	#SW03, 2#SWR		: INHIBIT WRITE CHECK?
7067	012726	001005				BNE	7\$: YES--BRANCH
7068	012730	012737	000151	003522		MOV	#WRCKD, 2#DTADPB+2		: COMMAND=WRITE CHECK DATA
7069	012736	004037	020320			JSR	RO, 2#DRVCL		: START A DATA TRANSFER
7070	012742	005737	001316		7\$:	TST	2#WCEFLG		: WRITE CHECK ERROR FLAG SET?
7071	012746	001404				BEQ	8\$: NO--BRANCH
7072	012750	032737	000001	177570		BIT	#SW00, 2#SWR		: PERFORM READ AFTER FATAL "WCE"?
7073	012756	001417				BEQ	9\$: NO--BRANCH
7074	012760	032737	000004	177570	8\$:	BIT	#SW02, 2#SWR		: INHIBIT READ DATA AND SOFTWARE COMPARE?
7075	012766	001013				BNE	9\$: YES--BRANCH
7076	012770	012737	000171	003522		MOV	#READ, 2#DTADPB+2		: COMMAND=READ
7077	012776	004037	020320			JSR	RO, 2#DRVCL		: START A DATA TRANSFER
7078	013002	032737	000002	177570		BIT	#SW01, 2#SWR		: COMPARE THE DATA?
7079	013010	001002				BNE	9\$: NO--BRANCH
7080	013012	004737	023132			JSR	PC, 2#DATCMP		: YES--DO IT
7081	013016	004037	022266		9\$:	JSR	RO, 2#INCRK		: MOVE TO NEXT TRACK
7082	013022	000661				BR	1\$: OUT OF TRACKS GO TO NEXT PATTERN
7083	013024	000700				BR	4\$: LOOP
7084	013026	000004				EXIT15:	SCOPE		: SCOPE LOOP

7086
7087
7112
(3)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(3)
(3)
(4)
(4)
(4)
(4)
(3)
(3)
(3)
(3)
(3)
(3)
(1)
7113
7114
7115
7116
7117
7118
7119
7120
7121
7122
7123
7124
(1)
7125
7126
7127
7128

.SBTTL *** EXERCISE TEST (16) ***

;TEST 16 RANDOM ADDRESS AND RANDOM PATTERN TEST (TST16)

;* STARTING AT "FC" AND GOING THROUGH "LC" THE DISK PACK
;* IS WRITTEN WITH A RANDOM PATTERN. THE FIRST TWO WORDS
;* OF EACH SECTOR WILL BE THE BASE OF THE RANDOM GENERATOR
;* FOR THAT SECTOR.
;* THE TEST THEN PERFORMS THE FOLLOWING SEQUENCE "R" TIMES
;* "R" DEFAULTS TO 20,000.

- 1) GENERATE A RANDOM ADDRESS
- 2) WRITE A RANDOM PATTERN AT THE ADDRESS
GENERATED IN 1.
- 3) GENERATE A RANDOM ADDRESS
- 4) READ THE SECTOR AT THE ADDRESS
GENERATED IN 3.
- 5) DO A SOFTWARE CHECK OF THE DATA READ IN 4.
- 6) DO A WRITE CHECK OF THE DATA WRITTEN IN 2
- 7) GENERATE A RANDOM ADDRESS
- 8) READ THE SECTOR AT THE ADDRESS
GENERATED IN 7.
- 9) DO A SOFTWARE CHECK OF THE DATA READ IN 8
- 10) DO A WRITE CHECK OF THE DATA WRITTEN IN 2

†TST16:

```

BIT      2#BITS+<16*2>,TSTNMS ;DO THIS TEST?
BNE      64$                   ;YES--BRANCH
JMP      SEOP                  ;NO--GO TO THE END OF THE PROGRAM
64$:     MOV      2#RPT16,STIMES ;GET THE ITERATION COUNT
        MOV      2#TST16,2#SLPADR ;SETUP THE SCOPE LOOP ADDRESS
        MOV      2#TST16,2#SLPERR ;SETUP THE ERROR LOOP ADDRESS
        MOV      2#16,2#STSTNM  ;SET UP TEST NUMBER AND
                                ;CLEAR THE ERROR FLAG (SERFLG)
        MOV      STSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
        MOV      2#EXIT16,2#BYPASS
        MOV      2#176543,2#SHINUM ;PRIME THE RANDOM NUMBER GENERATOR
        MOV      2#123456,2#SLONUM
        MOV      2#FC16,2#DTADPB+12 ;CYLINDER
        MOV      2#TRACKWC,2#DTADPB+4 ;WORD COUNT
        MOV      2#BUFFER,2#DTADPB+6 ;BUFFER ADDRESS
        MOV      2#WRITE,2#DTADPB+2 ;COMMAND
        BIT      2#SW15,2#C.SWR  ;WRITE THE DISK PACK BEFORE TESTING?
        BNE      3$              ;NO--BRANCH
        JSR      RO,2#FILRAN    ;FILL DATA BUFFER WITH RANDOM DATA
        CLR      2#DTADPB+10    ;SECTOR AND TRACK
1$:      JSR      RO,2#DRVCAL    ;START A DATA TRANSFER
2$:      INCB    2#DTADPB+11    ;NEXT TRACK
        CMPB    2#D19,2#DTADPB+11 ;TIME FOR NEXT CYLINDER
        BGT      2$            ;NO--BRANCH
        INC     2#DTADPB+12

```

(3)	013030			
(4)	013030	033737	001430	001224
(4)	013036	001002		
(4)	013040	000137	013476	
(4)	013044	013737	002322	001176
(3)	013052	012737	013030	001106
(3)	013060	012737	013030	001110
(3)	013066	012737	000016	001102
(3)				
(3)	013074	013737	001102	177570
7113	013102	012737	013474	001234
7114	013110	012737	176543	017312
7115	013116	012737	123456	017314
7116	013124	013737	002324	003532
7117	013132	012737	165000	003524
7118	013140	012737	040524	003526
7119	013146	012737	000161	003522
7120	013154	032737	!00000	001212
7121	013162	001022		
7122	013164	004037	023454	
7123	013170	005037	003530	
7124	013174			
(1)	013174	004037	020320	
7125	013200	105237	003531	
7126	013204	122737	000023	003531
7127	013212	003370		
7128	013214	005237	003532	

7129	013220	023737	002326	003532		CMP	2#LC16,2#DTADPB+12 ;OUT OF CYLINDERS?
7130	013226	002360				BGE	15 ;NO--BRANCH
7131	013230	012737	177400	003524	3\$:	MOV	#SCTRW,2#DTADPB+4 ;WORD COUNT
7132	013236	012737	013252	001106		MOV	#TEST16,2#SLPADR
7133	013244	012737	013252	001110		MOV	#TEST16,2#SLPER?
7134	013252	012706	001100		TEST16:	MOV	#STACK,SP ;SET STACK POINTER
7135	013256	004037	023730			JSR	RO,2#RANADR ;GENERATE A RANDOM ADDRESS
7136	013262	013737	003530	001322		MOV	2#DTADPB+10,2#SVADR ;SAVE THE TRACK/SECTOR
7137	013270	013737	003532	001324		MOV	2#DTADPB+12,2#SVADR+2 ;SAVE THE CYLINDER
7138	013276	012737	000161	003522		MOV	#WRITE,2#DTADPB+2 ;COMMAND=WRITE DATA
7139	013304	012701	040524			MOV	#BUFFER,R1 ;BUFFER ADDRESS
7140	013310	010137	003526			MOV	R1,2#DTADPB+6
7141	013314	004037	023674			JSR	RO,2#RANPAT ;GENERATE RANDOM PATTERN
7142	013320	004037	020320			JSR	RO,2#DRVCAL ;START A DATA TRANSFER
7143	013324	004037	023730			JSR	RO,2#RANADR
7144	013330	012737	000171	003522		MOV	#READ,2#DTADPB+2 ;COMMAND=READ DATA
7145	013336	012737	041524	003526		MOV	#BUFFER+512.,2#DTADPB+6 ;BUFFER ADDRESS
7146	013344	004037	020320			JSR	RO,2#DRVCAL ;START A DATA TRANSFER
7147	013350	004037	023476			JSR	RO,2#RANCK ;CHECK THE DATA
7148	013354	013737	001322	003530		MOV	2#SVADR,2#DTADPB+10 ;GET ADDRESS OF WHERE THE LAST
7149	013362	013737	001324	003532		MOV	2#SVADR+2,2#DTADPB+12 ;WRITE WAS PERFORMED
7150	013370	012737	000151	003522		MOV	#WRCKD,2#DTADPB+2 ;COMMAND=WRITE CHECK DATA
7151	013376	012737	040524	003526		MOV	#BUFFER,2#DTADPB+6 ;DATA BUFFER ADDRESS
7152	013404	004037	020320			JSR	RO,2#DRVCAL ;START A DATA TRANSFER
7153	013410	004037	023730			JSR	RO,2#RANADR ;GENERATE A RANDOM ADDRESS
7154	013414	012737	000171	003522		MOV	#READ,2#DTADPB+2 ;COMMAND=READ
7155	013422	012737	041524	003526		MOV	#BUFFER+512.,2#DTADPB+6 ;DATA BUFFER ADDRESS
7156	013430	004037	020320			JSR	RO,2#DRVCAL ;START A DATA TRANSFER
7157	013434	004037	023476			JSR	RO,2#RANCK ;CHECK THE DATA
7158	013440	013737	001322	003530		MOV	2#SVADR,2#DTADPB+10 ;GET DISK ADDRESS OF THE
7159	013446	013737	001324	003532		MOV	2#SVADR+2,2#DTADPB+12 ;LAST WRITE
7160	013454	012737	000151	003522		MOV	#WRCKD,2#DTADPB+2 ;COMMAND=WRITE CHECK DATA
7161	013462	012737	040524	003526		MOV	#BUFFER,2#DTADPB+6 ;DATA BUFFER ADDRESS
7162	013470	004037	020320			JSR	RO,2#DRVCAL ;START A DATA TRANSFER
7163	013474	000004			EXIT16:	SCOPE	;LOOP

```

7165
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(4) 013476
(4) 013476 104400 013504
(4) 013502 000410
(4)
(4) 013524
(3) 013524 005737 001222
(3) 013530 001434
(4) 013532 104400 013540
(4) 013536 000405
(4)
(4) 013552
(4) 013552 013746 001236
(4) 013556 104404
(4) 013560 002
(4) 013561 000
(4) 013562 104400 013570
(4) 013566 000412
(4)
(4) 013614
(4) 013614 013746 001112
(4) 013620 104402
(3) 013622 005037 001112
(1) 013626 005037 001102
(1) 013632 005037 001176
(1) 013636 005237 001100
(1) 013642 042737 100000 001100
(1) 013650 005327
(1) 013652 000010
(1) 013654 003036
(1) 013656 012737
(1) 013660 000010
(1) 013662 013652
(3) 013664 104400 013672
(3) 013670 000407
(3)
(3) 013710
(2) 013710 104400 013756
(1) 013714 013700 000042
(1) 013720 001414
(1) 013722 022700 013742
(1) 013726 001005
(1) 013730 022760 177777 000002
(1) 013736 001005
(1) 013740 000005
(1) 013742 004710

```

```

;*****
.SBTTL END OF PASS ROUTINE

;*INCREMENT THE PASS NUMBER ($PASS)
;*INDICATE END-OF-PROGRAM AFTER 8. PASSES THRU THE PROGRAM
;*IF THERES A MONITOR GO TO IT
;*IF THERE ISN'T JUMP TO RESTART

$EOP:
TYPE .+4 ;;TYPE ASCIZ STRING
BR 64$ ;;GET OVER THE ASCIZ
;;.ASCIZ <15><12><12>/END OF PASS/

64$:
TST 2#DRVSEL ;;ANY DRIVES SELECTED?
BEQ 1$ ;;NO--BRANCH
TYPE .+4 ;;TYPE ASCIZ STRING
BR 65$ ;;GET OVER THE ASCIZ
;;.ASCIZ / ON DRIVE/

65$:
MOV 2#CHKDRV,-(SP) ;;SAVE 2#CHKDRV FOR TYPEOUT
TYPOS ;;GO TYPE--OCTAL ASCII
.BYTE 2 ;;TYPE 2 DIGIT(S)
.BYTE 0 ;;SUPPRESS LEADING ZEROS
TYPE .+4 ;;TYPE ASCIZ STRING
BR 66$ ;;GET OVER THE ASCIZ
;;.ASCIZ / ERRORS DETECTED=/

66$:
MOV 2#SERTTL,-(SP) ;;SAVE 2#SERTTL FOR TYPEOUT
TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)

1$:
CLR 2#SERTTL ;;ZERO ERROR TOTAL
CLR $STNM ;;ZERO THE TEST NUMBER
CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS
INC $PASS ;;INCREMENT THE PASS NUMBER
BIC 2#100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;;LOOP?

$EOPCT: .WORD 8.
BGT $DOAGN ;;YES
MOV (PC)+,2(PC)+ ;;RESTORE COUNTER

$ENDCT: .WORD 8.
TYPE .+4 ;;TYPE ASCIZ STRING
BR 64$ ;;GET OVER THE ASCIZ
;;.ASCIZ <15><12>/END OF TEST/

64$:
TYPE $ENULL ;;TYPE NULL CHARACTER
$GET42: MOV 2#42,RO ;;GET MONITOR ADDRESS
BEQ $DOAGN ;;BRANCH IF NO MONITOR
CMP #SENDAD,RO ;;IS MONITOR ACT11?
BNE $SENDAD ;;NO--BRANCH
CMP #-1,2(RO) ;;YES--IS THIS THE LAST PASS?
BNE $DOAGN ;;NO--MAKE ANOTHER PASS
RESET ;;CLEAR THE WORLD
$SENDAD: JSR PC,(RO) ;;GO TO MONITOR

```



```

7173      ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE
7174      ;*WHICH ERROR IS TO BE REPORTED. IT THEN OBTAINS FROM THE "ERROR
7175      ;*TABLE" ($ERRTB), AND REPORTS THE APPROPRIATE INFORMATION
7176      ;*CONCERNING THE ERROR.
7177      ;*CALL
7178      ;*
7179      ;*   JSR   PC, @#TYPERR
7180      ;*   RETURN
7181      014164 113737 001102 001170 TYPERR: MOVB  @#STSTNM, @#STMPO ;SAVE THE TEST NUMBER
7182      014172 004037 034334      JSR   RO, @#SAVR05 ;SAVE RO-R5
7183      014176 162700 000004      SUB   #4, RO ;FORM TEST PC
7184      014202 010037 001154      MOV   RO, @#SREG0 ;COPY RO-R5 IN $REG0-$REG5
7185      014206 010137 001156      MOV   R1, @#SREG1
7186      014212 010237 001160      MOV   R2, @#SREG2
7187      014216 010337 001162      MOV   R3, @#SREG3
7188      014222 010437 001164      MOV   R4, @#SREG4
7189      014226 010537 001166      MOV   R5, @#SREG5
7190      014232 113700 001114      MOVB  @#$ITEMB, RO ;PICKUP ERROR ITEM NUMBER
7191      014236 010001      MOV   RO, R1 ;AND COPY IT INTO R1
7192      014240 005300      DEC   RO ;FORM INDEX FOR ERROR TABLE
7193      014242 106300      ASLB  RO
7194      014244 106300      ASLB  RO
7195      014246 106300      ASLB  RO
7196      014250 103002      BCC   1$ ;IS ERROR > 37?
7197      014252 062700 000220      ADD   @#ITEM41-$ERRTB, RO ;YES--FORM OFFSET
7198      014256 062700 003610      ADD   @#ERRTB, RO ;FORM ADDRESS
7199      014262 012037 014276      MOV   (RO)+, @# ;GET ERROR MESSAGE (EM) POINTER
7200      014266 001447      BEQ   7$ ;BRANCH IF THERE ISN'T ONE
7201      014270 104400 001207      TYPE , $CRLF ;"CARRIAGE RETURN - LINE FEED"
7202      014274 104400      TYPE
7203      014276 000000      .WORD 0 ;"EM" POINTER GOES HERE
7204      014300 162701 000041      SUB   #41, R1 ;SPECIAL ERROR ITEM NUMBER?
7205      014304 100440      BMI   7$ ;NO--BRANCH
7206      014306 013701 001242      MOV   @#SVSTAT, R1 ;GET STATUS/ERROR INDICATOR
7207      014312 106301      ASLB  R1 ;STRIP "DONE" BIT (BIT07)
7208      014314 006301      ASL   R1 ;STRIP "ERROR" BIT (BIT15)
7209      014316 012702 001566      MOV   @#STATBL, R2 ;1ST ADDRESS ON STATUS MESSAGE POINTERS
7210      014322 005003      CLR   R3 ;CARRIAGE RETURN-LINE FEED SWITCH
7211      014324 104400 014332      TYPE , @#+4 ;TYPE ASCIZ STRING
7212      (1) 014330 000402      BR    64$ ;GET OVER THE ASCIZ
7213      (1)
7214      (1)
7215      (1) 014336      64$:
7216      7212 014336 012237 014360      3$: MOV   (R2)+, @# ;MESSAGE POINTER
7217      7213 014342 006301      ASL   R1 ;TYPE THIS MESSAGE?
7218      7214 014344 103013      BCC   6$ ;NO--BRANCH
7219      7215 014346 005103      COM   R3 ;YES--TYPE A "CR" & "LF"?
7220      7216 014350 001002      BNE   4$ ;NO--BRANCH
7221      7217 014352 104400 001207      TYPE , $CRLF ;YES
7222      7218 014356 104400      4$: TYPE
7223      7219 014360 000000      5$: .WORD 0 ;MESSAGE POINTER GOES HERE
7224      7220 014362 005701      TST   R1 ;MORE TO TYPE?
7225      7221 014364 001403      BEQ   6$ ;NO--BRANCH
7226      7222 014366 104400 035202      TYPE , MSG.SP ;YES--SPACES
7227      7223 014372 000761      BR    3$ ;LOOP

```

```

7224 014374 001360      6S:  BNE 3S ;BRANCH IF NOT FINISHED
7225 014376 104400 014404  TYPE +4 ;TYPE ASCIZ STRING
(1) 014402 000401  BR 65S ;GET OVER THE ASCIZ
(1) ;:ASCIZ //
(1) 014406 65S:
7226 014406 012037 014422 7S:  MOV (R0)+,8S ;PICK UP DATA HEADER (DH) POINTER
7227 014412 001404  BEQ 9S ;BRANCH IF NONE
7228 014414 104400 001207  TYPE ,SCLF ;CARRIAGE RETURN-LINE FEED
7229 014420 104400  TYPE
7230 014422 000000 8S:  .WORD 0 ;"DH" POINTER GOES HERE
7231 014424 012001 9S:  MOV (R0)+,R1 ;PICKUP DATA TABLE (DT) POINTER
7232 014426 001450  BEQ 20S ;BRANCH IF NONE
7233 014430 005005  CLR R5 ;SET INDENT SWITCH
7234 014432 012000  MOV (R0)+,R0 ;DATA FORMAT (DF) POINTER
7235 014434 012002  MOV (R0)+,R2 ;NUMBER OF DH'S TO TYPE
7236 014436 001441  BEQ 17S ;BRANCH IF DH NUMBER IS 0
7237 014440 005105  COM R5 ;NO INDENT
7238 014442 104400 001207  TYPE ,SCLF ;CARRIAGE RETURN-LINE FEED
7239 014446 112003 10S: MOVB (R0)+,R3 ;NUMBER OF DATA WORDS TO TYPE
7240 014450 112004  MOVB (R0)+,R4 ;AND HOW TO TYPE THEM
7241 014452 006004 11S: ROR R4 ;OCTAL OR DECIMAL?
7242 014454 103403  BCS 12S ;DECIMAL--BRANCH
7243 014456 013146  MOV 2(R1)+,-(SP) ;SAVE 2(R1)+ FOR TYPEOUT
(1) 014460 104402  TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
7244 014462 000402  BR 13S
7245 014464 12S:
(1) 014464 013146  MOV 2(R1)+,-(SP) ;SAVE 2(R1)+ FOR TYPEOUT
(1) 014466 104410  TYPDS ;GO TYPE--DECIMAL ASCII WITH SIGN
7246 014470 005303 13S: DEC R3 ;MORE NUMBERS TO TYPE?
7247 014472 001403  BEQ 14S ;NO--BRANCH
7248 014474 104400 035202  TYPE ,MSG.SP ;YES--TYPE SEPERATORS
7249 014500 000764  BR 11S ;LOOP
7250 014502 005302 14S: DEC R2 ;MORE DH'S?
7251 014504 003421  BLE 20S ;NO--BRANCH
7252 014506 104400 001207  TYPE ,SCLF ;YES--START A NEW LINE
7253 014512 005105  COM R5 ;INDENT?
7254 014514 001002  BNE 15S ;NO--BRANCH
7255 014516 104400 035202  TYPE ,MSG.SP ;YES--TYPE SPACES
7256 014522 012037 014530 15S: MOV (R0)+,16S ;GET NEXT DH
7257 014526 104400  TYPE ;AND TYPE IT
7258 014530 000000 16S: .WORD 0 ;DH POINTER GOES HERE
7259 014532 104400 001207  TYPE ,SCLF ;CARRIAGE RETURN-LINE FEED
7260 014536 005705  TST R5 ;INDENT?
7261 014540 001342  BNE 10S ;NO--BRANCH
7262 014542 104400 035202 17S: TYPE ,MSG.SP ;YES--TYPE SPACES
7263 014546 000737  BR 10S ;LOOP
7264 014550 004037 034360 20S: JSR R0,2#GETROS ;RESTORE R0-R5
7265 014554 000207  RTS PC ;RETURN
7266
7267
7268
7269 ;*****
(1)
(1) .SBTTL TYPE ROUTINE

```

```

(1)
(1) ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
(1) ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
(1) ;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
(1) ;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
(1) ;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
(1) ;*
(1) ;*CALL:
(1) ;*1) USING A TRAP INSTRUCTION
(1) ;* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
(1) ;*OR
(1) ;* TYPE
(1) ;* MESADR
(1) ;*
(1) ;*2) USING A JSR INSTRUCTION
(1) ;* MOV PS,-(SP) ;;PUSH PROCESSOR STATUS WORD ON THE STACK
(1) ;* JSR PC,$TYPE ;;CALL TYPE ROUTINE
(1) ;* MESADDR ;;FIRST ADDRESS OF MESSAGE
(1)
(1) 014556 105737 001151 $TYPE: TSTB $TPFLG ;; IS THERE A TERMINAL?
(1) 014562 100002 BPL $S ;; BR IF YES
(1) 014564 000000 HALT ;; HALT HERE IF NO TERMINAL
(1) 014566 000407 BR $S ;; LEAVE
(1) 014570 010046 1$: MOV RO,-(SP) ;; SAVE RO
(1) 014572 017600 000002 MOV $2(SP),RO ;; GET ADDRESS OF ASCIZ STRING
(1) 014576 112046 2$: MOV $B(RO)+,-(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 014600 001005 BNE $S ;; BR IF IT ISN'T THE TERMINATOR
(1) 014602 005726 TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
(1) 014604 012600 MOV (SP)+,RO ;; RESTORE RO
(1) 014606 062716 3$: ADD $2,(SP) ;; ADJUST RETURN PC
(1) 014612 000002 RTI ;; RETURN
(1) 014614 004737 4$: JSR PC,$TYPEC ;; GO TYPE THIS CHARACTER
(1) 014620 123726 5$: CMPB $FILLC,(SP)+ ;; IS IT TIME FOR FILLER CHARS.?
(1) 014624 001364 BNE $S ;; IF NO GO GET NEXT CHAR.
(1) 014626 013746 MOV $NULL,-(SP) ;; GET # OF FILLER CHARS. NEEDED
(1) ;* AND THE NULL CHAR.
(1) 014632 105366 6$: DECB 1(SP) ;; DOES A NULL NEED TO BE TYPED?
(1) 014636 002770 BLT $S ;; BR IF NO--GO POP THE NULL OFF OF STACK
(1) 014640 004737 JSR PC,$TYPEC ;; GO TYPE A NULL
(1) 014644 000772 BR $S ;; LOOP
(1) 014646 105777 164270 $TYPEC: TSTB $STPS ;; WAIT UNTIL PRINTER IS READY
(1) 014652 100375 BPL $TYPEC
(1) 014654 116677 000002 164262 MOV $B(2(SP),$STPB) ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 014662 000207 RTS PC
(1) ;*****
(1)
(1) ;SBTTL BINARY TO OCTAL (ASCII) AND TYPE
(1)
(1) ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
(1) ;*OCTAL (ASCII) NUMBER AND TYPE IT.
(1) ;*$TYPCS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
(1) ;*CALL:
(1) ;* MOV NUM,-(SP) ;;NUMBER TO BE TYPED
(1) ;* TYPOS ;;CALL FOR TYPEOUT

```


N07

```

(1)      ;*      .BYTE  N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
(1)      ;*      .BYTE  M      ;;M=1 OR 0
(1)      ;*      ;*      ;;I=TYPE LEADING ZEROS
(1)      ;*      ;*      ;;O=SUPPRESS LEADING ZEROS
(1)      ;*STYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
(1)      ;*STYPOS OR STYPOC
(1)      ;*CALL:
(1)      ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
(1)      ;*      TYPON      ;;CALL FOR TYPEOUT
(1)      ;*
(1)      ;*STYPOC----ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(1)      ;*CALL:
(1)      ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
(1)      ;*      TYPOC      ;;CALL FOR TYPEOUT
(1)
(1) 014664 017646 000000      STYPOS: MOV      0(SP),-(SP)      ;;PICKUP THE MODE
(1) 014670 116637 000001 015107      MOVVB   1(SP),$OFILL      ;;LOAD ZERO FILL SWITCH
(1) 014676 112637 015111      MOVVB   (SP)+,$OMODE+1      ;;NUMBER OF DIGITS TO TYPE
(1) 014702 062716 000002      ADD     #2,(SP)      ;;ADJUST RETURN ADDRESS
(1) 014706 000406      BR      STYPON
(1) 014710 112737 000001 015107      STYPOC: MOVVB  #1,$OFILL      ;;SET THE ZERO FILL SWITCH
(1) 014716 112737 000006 015111      MOVVB   #6,$OMODE+1      ;;SET FOR SIX(6) DIGITS
(1) 014724 112737 000005 015106      STYPON: MOVVB  #5,$OCNT      ;;SET THE ITERATION COUNT
(1) 014732 010346      MOV     R3,-(SP)      ;;SAVE R3
(1) 014734 010446      MOV     R4,-(SP)      ;;SAVE R4
(1) 014736 010546      MOV     R5,-(SP)      ;;SAVE R5
(1) 014740 113704 015111      MOVVB   $OMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
(1) 014744 005404      NEG     R4
(1) 014746 062704 000006      ADD     #6,R4      ;;SUBTRACT IT FOR MAX. ALLOWED
(1) 014752 110437 015110      MOVVB   R4,$OMODE      ;;SAVE IT FOR USE
(1) 014756 113704 015107      MOVVB   $OFILL,R4      ;;GET THE ZERO FILL SWITCH
(1) 014762 016605 000012      MOV     12(SP),R5      ;;PICKUP THE INPUT NUMBER
(1) 014766 005003      CLR     R3      ;;CLEAR THE OUTPUT WORD
(1) 014770 006105      1$:    ROL     R5      ;;ROTATE MSB INTO "C"
(1) 014772 000404      BR      3$      ;;GO DO MSB
(1) 014774 006105      2$:    ROL     R5      ;;FORM THIS DIGIT
(1) 014776 006105      ROL     R5
(1) 015000 006105      ROL     R5
(1) 015002 010503      MOV     R5,R3
(1) 015004 006103      3$:    ROL     R3      ;;GET LSB OF THIS DIGIT
(1) 015006 105337 015110      DECB   $OMODE      ;;TYPE THIS DIGIT?
(1) 015012 100016      BPL    7$      ;;BR IF NO
(1) 015014 042703 177770      BIC    #177770,R3      ;;GET RID OF JUNK
(1) 015020 001002      BNE    4$      ;;TEST FOR 0
(1) 015022 005704      TST    R4      ;;SUPPRESS THIS 0?
(1) 015024 001403      BEQ    5$      ;;BR IF YES
(1) 015026 005204      4$:    INC     R4      ;;DON'T SUPPRESS ANYMORE 0'S
(1) 015030 052703 000060      BIS    #'0,R3      ;;MAKE THIS DIGIT ASCII
(1) 015034 052703 000040      5$:    BIS    #' ,R3      ;;MAKE ASCII IF NOT ALREADY
(1) 015040 110337 015104      MOVVB  R3,#$      ;;SAVE FOR TYPING
(1) 015044 104400 015104      TYPE   #8$      ;;GO TYPE THIS DIGIT
(1) 015050 105337 015106      7$:    DECB   $OCNT      ;;COUNT BY 1
(1) 015054 003347      BGT    2$      ;;BR IF MORE TO DO

```

(1) 015056 002402 BLT 6S :: BR IF DONE
(1) 015060 005204 INC R4 :: INSURE LAST DIGIT ISN'T A BLANK
(1) 015062 000744 BR 2S :: GO DO THE LAST DIGIT
(1) 015064 012605 6S: MOV (SP)+,R5 :: RESTORE R5
(1) 015066 012604 MOV (SP)+,R4 :: RESTORE R4
(1) 015070 012603 MOV (SF)+,R3 :: RESTORE R3
(1) 015072 016666 000002 000004 MOV 2(SP),4(SP) :: SET THE STACK FOR RETURNING
(1) 015100 012616 MOV (SP)+,(SP)
(1) 015102 000002 RTI :: RETURN
(1) 015104 000 8S: .BYTE 0 :: STORAGE FOR ASCII DIGIT
(1) 015105 000 .BYTE 0 :: TERMINATOR FOR TYPE ROUTINE
(1) 015106 000 \$OCNT: .BYTE 0 :: OCTAL DIGIT COUNTER
(1) 015107 000 \$OFILL: .BYTE 0 :: ZERO FILL SWITCH
(1) 015110 000000 \$OMODE: .WORD 0 :: NUMBER OF DIGITS TO TYPE

.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 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 R5,-(SP) :: PUSH R5 ON STACK
MOV #20200,-(SP) :: SET BLANK SWITCH AND SIGN
MOV 20(SP),R5 :: GET THE INPUT NUMBER
BPL 1S :: BR IF INPUT IS POS.
NEG R5 :: MAKE THE BINARY NUMBER POS.
MOVB #'-,1(SP) :: MAKE THE ASCII NUMBER NEG.
1S: CLR R0 :: ZERO THE CONSTANTS INDEX
MOV #0BLK,R3 :: SETUP THE OUTPUT POINTER
MOVB #'',(R3)+ :: SET THE FIRST CHARACTER TO A BLANK
2S: CLR R2 :: CLEAR THE BCD NUMBER
MOV \$DTBL(R0),R1 :: GET THE CONSTANT
3S: SUB R1,R5 :: FORM THIS BCD DIGIT
BLT 4S :: BR IF DONE
INC R2 :: INCREASE THE BCD DIGIT BY 1
BR 3S
4S: ADD R1,R5 :: ADD BACK THE CONSTANT
TST R2 :: CHECK IF BCD DIGIT=0
BNE 5S :: FALL THROUGH IF 0
TSTB (SP) :: STILL DOING LEADING 0'S?
BMI 7S :: BR IF YES
5S: ASLB (SP) :: MSD?
BCC 6S :: BR IF NO

```

(1) 015214 116663 000001 177777          MCVB 1(SP),-1(R3)  ;;YES--SET THE SIGN
(1) 015222 052702 000060                6S:  BIS  #'0,R2      ;;MAKE THE BCD DIGIT ASCII
(1) 015226 052702 000040                7S:  BIS  #' R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1) 015232 110223                MOVB  R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1) 015234 005720                TST  (R0)+       ;;JUST INCREMENTING
(1) 015236 020027 000010                CMP  R0,#1C     ;;CHECK THE TABLE INDEX
(1) 015242 002746                BLT  Z$         ;;GO DO THE NEXT DIGIT
(1) 015244 003002                BGT  B$         ;;GO TO EXIT
(1) 015246 010502                MOV  R5,R2      ;;GET THE LSD
(1) 015250 000764                BR   6$         ;;GO CHANGE TO ASCII
(1) 015252 105726                8S:  TSTB (SP)+   ;;WAS THE LSD THE FIRST NON-ZERO?
(1) 015254 100003                BPL  9$         ;;BR IF NO
(1) 015256 116663 177777 177776        9S:  MOVB -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
(1) 015264 105013                CLRB (R3)       ;;SET THE TERMINATOR
(3) 015266 012605                MOV  (SP)+,R5   ;;POP STACK INTO R5
(3) 015270 012603                MOV  (SP)+,R3   ;;POP STACK INTO R3
(3) 015272 012602                MOV  (SP)+,R2   ;;POP STACK INTO R2
(3) 015274 012601                MOV  (SP)+,R1   ;;POP STACK INTO R1
(3) 015276 012600                MOV  (SP)+,R0   ;;POP STACK INTO R0
(1) 015300 104400 015326                TYPE  $DBLK     ;;NOW TYPE THE NUMBER
(1) 015304 016666 000002 000004        MOV  2(SP),4(SP) ;;ADJUST THE STACK
(1) 015312 012616                MOV  (SP)+,(SP)
(1) 015314 000002                RTI
(1) 015316 023420                SOTBL: 10000.   ;;RETURN TO USER
(1) 015320 001750                1000.
(1) 015322 000144                100.
(1) 015324 000012                10.
(1) 015326 000004                SOBKL: .BLKW 4

```

7272

.SBTTL TTY INPUT ROUTINE

```

(1) 015336 000000                STKCNT: .WORD 0  ;;NUMBER OF ITEMS IN QUEUE
(1) 015340 000000                STKQIN: .WORD 0  ;;INPUT POINTER
(1) 015342 000000                STKQOUT: .WORD 0 ;;OUTPUT POINTER
(1) 015344 000002                STKQSR: .BLKB 2  ;;TTY KEYBOARD QUEUE
(1) 015346 015346                STKQEND=.

; *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
; * RETURN

(1) 015346 005037 015336                $TKINT: CLR  STKCNT  ;;CLEAR COUNT OF ITEMS IN QUEUE
(1) 015352 012737 015344 015340        MOV  #STKQSR,STKQIN ;;MOVE THE STARTING ADDRESS OF THE
(1) 015360 013737 015340 015342        MOV  STKQIN,STKQOUT ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
(1) 015366 012737 015416 000060        MOV  #STKSRV,#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
(1) 015374 012737 000200 000062        MOV  #200,#TKVEC+2 ;;BR" LEVEL 4
(1) 015402 005777 163532                TST  $TKB       ;;CLEAR DONE FLAG
(1) 015406 012777 000100 163522        MOV  #BZT06,$TKS ;;ENABLE INTERRUPT
(1) 015414 000207                RTS  PC         ;;RETURN TO CALLER

```

```

(1)
(1)
(1)
(1)
(1) 015416 117746 163516
(1) 015422 042716 177600
(1) 015426 021627 000003
(1) 015432 001006
(1) 015434 104400 016075
(1) 015440 004737 015346
(1) 015444 000137 004210
(1) 015450 022737 000002 015336 1$:
(1) 015456 001004
(1) 015460 104400 001202
(1) 015464 005726
(1) 015466 000002
(1) 015470 005237 015336 2$:
(1) 015474 112677 177640
(1) 015500 005237 015340
(1) 015504 023727 015340 015346
(1) 015512 001003
(1) 015514 012737 015344 015340
(1) 015522 000002 3$:
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 015524 011646
(1) 015526 016666 000004 000002
(1) 015534 005066 000004
(1) 015540 005037 177776
(1) 015544 005737 015336 1$:
(1) 015550 001775
(1) 015552 005337 015336
(1) 015556 117766 177560 000004
(1) 015564 005237 015342
(1) 015570 023727 015342 015346
(1) 015576 001003
(1) 015600 012737 015344 015342
(1) 015606 000002 2$:
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 015610 010346
(1) 015612 005046
(1) 015614 012703 016051
(1) 015620 022703 016075

```

```

: *TK SERVICE ROUTINE
: *THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
$TKSRV: MOVB    2$TKB,-(SP)          ;; PICKUP THE CHARACTER
          BIC    2$C177,(SP)        ;; STRIP THE JUNK
          CMP    (SP),#3            ;; IS IT A CONTROL C?
          BNE    1$                  ;; BRANCH IF NO
          TYPE   $CNTLC              ;; TYPE A CONTROL-C (1C)
          JSR    PC,$TKINT           ;; INIT THE KEYBOARD
          JMP    START2              ;; CONTROL C RESTART
          CMP    2$,2$STKCNT         ;; IS THE QUEUE FULL?
          BNE    2$                  ;; BRANCH IF NO
          TYPE   $BELL               ;; RING THE TTY BELL
          TST    (SP)+              ;; CLEAN CHARACTER OFF OF STACK
          RTI                          ;; RETURN
2$: INC    $TKCNT                    ;; COUNT THIS CHARACTER
          MOVB   (SP)+,2$STKQIN      ;; AND PUT IT IN QUEUE
          INC    $TKQIN              ;; UPDATE THE POINTER
          CMP    $TKQIN,#$STKQEND    ;; GO OFF THE END?
          BNE    3$                  ;; BRANCH IF NO
          MOV    #2$TKQSRT,$TKQIN    ;; RESET THE POINTER
          RTI                          ;; RETURN
3$:
*****
: *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
: *CALL:
: *
: * RDCHR                                     ;; INPUT A SINGLE CHARACTER FROM THE TTY
: * RETURN HERE                               ;; CHARACTER IS ON THE STACK
:
$RDCHR: MOV    (SP),-(SP)            ;; PUSH DOWN THE PC AND
          MOV    4(SP),2(SP)         ;; THE PS
          CLR    4(SP)               ;; GET READY FOR A CHARACTER
          CLR    2$PS                 ;; ALLOW INTERRUPTS
          TST    $TKCNT               ;; WAIT ON A CHARACTER
          SEQ    1$
          DEC    $TKCNT               ;; DECREMENT THE COUNTER
          MOVB  2$TKQOUT,4(SP)       ;; GET ONE CHARACTER
          INC    $TKQOUT              ;; UPDATE THE POINTER
          CMP    $TKQOUT,#$STKQEND   ;; DID IT GO OFF OF THE END?
          BNE    2$                  ;; BRANCH IF NO
          MOV    #2$TKQSRT,$TKQOUT   ;; RESET THE POINTER
          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
:
$RDLIN: MOV    R3,-(SP)              ;; SAVE R3
          CLR    -(SP)               ;; CLEAR THE RUBOUT KEY
          MOV    #2$TTYIN,R3         ;; GET ADDRESS
          CMP    #2$TTYIN+20.,R3     ;; BUFFER FULL?

```

```

(1) 015624 101456      BLOS      4$      ;; BR IF YES
(1) 015626 104412      RDCHR                      ;; GO READ ONE CHARACTER FROM THE TTY
(1) 015630 112613      MOV      (SP)+, (R3)      ;; GET CHARACTER
(1) 015632 122713      CMP      #177, (R3)      ;; IS IT A RUBOUT
(1) 015636 001022      BNE      5$      ;; BR IF NO
(1) 015640 005716      TST      (SP)      ;; IS THIS THE FIRST RUBOUT?
(1) 015642 001007      BNE      6$      ;; BR IF NO
(1) 015644 112737      MOV      #' \, 9$      ;; TYPE A BACK SLASH
(1) 015652 104400      TYPE      9$
(1) 015656 012716      MOV      #-1, (SP)      ;; SET THE RUBOUT KEY
(1) 015662 005303      6$: DEC      R3      ;; BACKUP BY ONE
(1) 015664 020327      016051      CMP      R3, #STTYIN      ;; STACK EMPTY?
(1) 015670 103434      BLO      4$      ;; BR IF YES
(1) 015672 111337      016042      MOV      (R3), 9$      ;; SETUP TO TYPEOUT THE DELETED CHAR.
(1) 015676 104400      TYPE      9$      ;; GO TYPE
(1) 015702 000746      BR      2$      ;; GO READ ANOTHER CHAR.
(1) 015704 005716      5$: TST      (SP)      ;; RUBOUT KEY SET?
(1) 015706 001406      BEQ      7$      ;; BR IF NO
(1) 015710 112737      000134      016042      MOV      #' \, 9$      ;; TYPE A BACK SLASH
(1) 015716 104400      TYPE      9$
(1) 015722 005016      CLR      (SP)      ;; CLEAR THE RUBOUT KEY
(1) 015724 122713      000025      7$: CMP      #25, (R3)      ;; IS CHARACTER A CTRL U?
(1) 015730 001003      BNE      8$      ;; BR IF NO
(1) 015732 104400      TYPE      $CNTLU      ;; TYPE A CONTROL "U"
(1) 015736 000726      BR      1$      ;; GO START OVER
(1) 015740 122713      8$: CMP      #12, (R3)      ;; IS CHARACTER A "LF"?
(1) 015744 001011      BNE      3$      ;; BRANCH IF NO
(1) 015746 105013      CLRB      (R3)      ;; CLEAR THE CHARACTER
(1) 015750 104400      TYPE      $CRLF      ;; TYPE A "CR" & "LF"
(1) 015754 104400      TYPE      $TTYIN      ;; TYPE THE INPUT STRING
(1) 015760 000717      BR      2$      ;; GO PICKUP ANOTHER CHARACTER
(1) 015762 104400      4$: TYPE      $QUES      ;; TYPE A '?'
(1) 015766 000712      BR      1$      ;; CLEAR THE BUFFER AND LOOP
(1) 015770 111337      3$: MOV      (R3), 9$      ;; ECHO THE CHARACTER
(1) 015774 104400      TYPE      9$
(1) 016000 122723      000015      CMP      #15, (R3)+      ;; CHECK FOR RETURN
(1) 016004 001305      BNE      2$      ;; LOOP IF NOT RETURN
(1) 016006 105063      177777      CLRB      -1(R3)      ;; CLEAR RETURN (THE 15)
(1) 016012 104400      001210      TYPE      $LF      ;; TYPE A LINE FEED
(1) 016016 005726      TST      (SP)+      ;; CLEAN RUBOUT KEY FROM THE STACK
(1) 016020 012603      MOV      (SP)+, R3      ;; RESTORE R3
(1) 016022 011646      MOV      (SP), -(SP)      ;; ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 016024 016666      000004      000002      MOV      4(SP), 2(SP)      ;; FIRST ASCII CHARACTER ON IT
(1) 016032 012766      016051      000004      MOV      #STTYIN, 4(SP)
(1) 016040 000002      RTI      ;; RETURN
(1) 016042 000      9$: .BYTE      0      ;; STORAGE FOR ASCII CHAR. TO TYPE
(1) 016043 000      .BYTE      0      ;; TERMINATOR
(1) 016044 052536      005015      000      $CNTLU: .ASCIZ /?U/(15)<(12)      ;; CONTROL "U"
(1) 016051 000024      $TTYIN: .BLKB      20.      ;; RESERVE 20. BYTES FOR TTY INPUT
(1) 016075 136      006503      000012      $CNTLC: .ASCIZ /?C/(15)<(12)      ;; CONTROL "C"
7273 ;*****
(1)
(1)
(1) .SBTTL SCOPE HANDLER ROUTINE

```

```

(1) ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(1) ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
(1) ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
(1) ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1) ;*SW14=1 LOOP ON TEST
(1) ;*SW11=1 INHIBIT ITERATIONS
(1) ;*SW09=1 LOOP ON ERROR
(1) ;*CALL
(1) ;* SCOPE ;;SCOPE=IOT
(1)
(1) 016102 $SCOPE:
(1) 016102 006137 177570 ROL 2#SWR ;;LOOP ON PRESENT TEST?
(1) 016106 100475 BHI $OVER ;;YES IF SW14=1
(1) ;*****START OF CODE FOR THE XOR TESTER*****
(1) 016110 000416 $XTSTR: BR 6$ ;;IF RUNNING ON THE "XOR" TESTER CHANGE
(1) ; THIS INSTRUCTION TO A "NOP" (NOP=240)
(1) 016112 013746 000004 MOV 2#ERRVEC, -(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1) 016116 012737 016136 000004 MOV 2$S, 2#ERRVEC ;;SET FOR TIMEOUT
(1) 016124 005737 177060 TST 2#177060 ;;TIME OUT ON XOR?
(1) 016130 012637 000004 MOV (SP)+, 2#ERRVEC ;;RESTORE THE ERROR VECTOR
(1) 016134 000450 BR $SVLAD ;;GO TO THE NEXT TEST
(1) 016136 022626 5$: CMP (SP)+, (SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
(1) 016140 012637 000004 MOV (SP)+, 2#ERRVEC ;;RESTORE THE ERROR VECTOR
(1) 016144 000413 BR 7$ ;;LOOP ON THE PRESENT TEST
(1) 016146 6$: ;*****END OF CODE FOR THE XOR TESTER*****
(1) 016146 105737 001103 2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
(1) 016152 001421 BEQ 3$ ;;BR IF NO
(1) 016154 123737 001115 001103 CMPB $ERMAX, $ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1) 016162 101015 BHI 3$ ;;BR IF NO
(1) 016164 032737 001000 177570 BIT 2#BIT09, 2#SWR ;;LOOP ON ERROR?
(1) 016172 001404 BEQ 4$ ;;BR IF NO
(1) 016174 013737 001110 001106 7$: MOV $LPERR, $LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
(1) 016202 000440 BR $OVER
(1) 016204 105037 001103 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
(1) 016210 005037 001176 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1) 016214 000412 BR 1$ ;;ESCAPE TO THE NEXT TEST
(1) 016216 032737 004000 177570 3$: BIT 2#BIT11, 2#SWR ;;INHIBIT ITERATIONS?
(1) 016224 001006 BNE 1$ ;;BR IF YES
(1) 016226 005237 001104 INC $ICNT ;;INCREMENT ITERATION COUNT
(1) 016232 023737 001176 001104 CMP $TIMES, $ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
(1) 016240 002021 BGE $OVER ;;BR IF MORE ITERATION REQUIRED
(1) 016242 012737 000001 001104 1$: MOV 2#1, $ICNT ;;REINITIALIZE THE ITERATION COUNTER
(1) 016250 013737 016320 001176 MOV $SMXCNT, $TIMES ;;SET NUMBER OF ITERATIONS TO DO
(1) 016256 105237 001102 $SVLAD: INCB $TSTNM ;;COUNT TEST NUMBERS
(1) 016262 011637 001106 MOV (SP), $LPADR ;;SAVE SCOPE LOOP ADDRESS
(1) 016266 011637 001110 MOV (SP), $LPERR ;;SAVE ERROR LOOP ADDRESS
(1) 016272 005037 001200 CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1) 016276 112737 000001 001115 MOVB 2#1, $ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1) 016304 013737 001102 177570 $OVER: MOV $TSTNM, 2#DISPLAY ;;DISPLAY TEST NUMBER
(1) 016312 013716 001106 MOV $LPADR, (SP) ;;FUDGE RETURN ADDRESS
(1) 016316 000002 RTI ;;FIXES PS
(1) 016320 000001 $MXCNT: 1 ;;MAX. NUMBER OF ITERATIONS
7274 ;*****
(1)

```

```

(1)          .SBTTL  SAVE AND RESTORE RO-R5 ROUTINES
(1)          ;*SAVE RO-R5
(1)          ;*CALL:
(1)          ;*   SAVREG
(1)          ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
(1)          ;*
(1)          ;*TOP---(+16)
(1)          ;* +2---(+18)
(1)          ;* +4---R5
(1)          ;* +6---R4
(1)          ;* +8---R3
(1)          ;*+10---R2
(1)          ;*+12---R1
(1)          ;*+14---R0
(1)          $SAVREG:
(3) 016322   C10046   MOV      RO,-(SP)      ;; PUSH RO ON STACK
(3) 016324   010146   MOV      R1,-(SP)      ;; PUSH R1 ON STACK
(3) 016326   010246   MOV      R2,-(SP)      ;; PUSH R2 ON STACK
(3) 016330   010346   MOV      R3,-(SP)      ;; PUSH R3 ON STACK
(3) 016332   010446   MOV      R4,-(SP)      ;; PUSH R4 ON STACK
(3) 016334   010546   MOV      R5,-(SP)      ;; PUSH R5 ON STACK
(1) 016336   016646   000022   MOV      22(SP),-(SP)  ;; SAVE PS OF MAIN FLOW
(1) 016342   016646   000022   MOV      22(SP),-(SP)  ;; SAVE PC OF MAIN FLOW
(1) 016346   016646   000022   MOV      22(SP),-(SP)  ;; SAVE PS OF CALL
(1) 016352   016646   000022   MOV      22(SP),-(SP)  ;; SAVE PC OF CALL
(1) 016356   000002   RTI
(1)          ;*RESTORE RO-R5
(1)          ;*CALL:
(1)          ;*   RESREG
(1)          $RESREG:
(1) 016360   012666   000022   MOV      (SP)+,22(SP)  ;; RESTORE PC OF CALL
(1) 016364   012666   000022   MOV      (SP)+,22(SP)  ;; RESTORE PS OF CALL
(1) 016370   012666   000022   MOV      (SP)+,22(SP)  ;; RESTORE PC OF MAIN FLOW
(1) 016374   012666   000022   MOV      (SP)+,22(SP)  ;; RESTORE PS OF MAIN FLOW
(3) 016400   012605   MOV      (SP)+,R5      ;; POP STACK INTO R5
(3) 016402   012604   MOV      (SP)+,R4      ;; POP STACK INTO R4
(3) 016404   012603   MOV      (SP)+,R3      ;; POP STACK INTO R3
(3) 016406   012602   MOV      (SP)+,R2      ;; POP STACK INTO R2
(3) 016410   012601   MOV      (SP)+,R1      ;; POP STACK INTO R1
(3) 016412   012600   MOV      (SP)+,R0      ;; POP STACK INTO R0
(1) 016414   000002   RTI
7275          ;*****
(1)          .SBTTL  TRAP DECODER
(1)          ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
(1)          ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
(1)          ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
(1)          ;*GO TO THAT ROUTINE.
(1) 016416   010046   $TRAP:  MOV      RO,-(SP)      ;;SAVE RO

```

```

(1) 016420 016600 000002
(1) 016424 005740
(1) 016426 111000
(1) 016430 016000 016436
(1) 016434 000200

```

```

MOV 2(SP),RO ;;GET TRAP ADDRESS
TST -(RO) ;;BACKUP BY 2
MOVB (RO),RO ;;GET RIGHT BYTE OF TRAP
MOV $TRPAD(RO),RO ;;INDEX TO TABLE
RTS RO ;;GO TO ROUTINE

```

.SBTTL TRAP TABLE

```

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

```

: ROUTINE

;\$TRPAD:

```

$STYPE ;;CALL=TYPE TRAP+0(104400) TTY TYPEOUT ROUTINE
$STYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
$STYPOS ;;CALL=TYPOS TRAP+4(104404) TYPE OCTAL NUMBER (NO LEADING ZEROS)
$STYPON ;;CALL=TYPON TRAP+6(104406) TYPE OCTAL NUMBER (AS PER LAST CALL)
$STYPOS ;;CALL=TYPOS TRAP+10(104410) TYPE DECIMAL NUMBER (WITH SIGN)
$SRDCHR ;;CALL=ROCHR TRAP+12(104412) TTY TYPEIN CHARACTER ROUTINE
$SRDLIN ;;CALL=ROLIN TRAP+14(104414) TTY TYPEIN STRING ROUTINE
$SAVREG ;;CALL=SAVREG TRAP+16(104416) SAVE RO-R5 ROUTINE
$RESREG ;;CALL=RESREG TRAP+20(104420) RESTORE RO-R5 ROUTINE

```

.SBTTL SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE

```

; *THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
; *UNSIGNED DECIMAL ASCIZ NUMBER.

```

;\$CALL

```

; * MOV NUMBER, -(SP) ;;PUT BINARY NUMBER ON THE STACK
; * JSR PC, @#$SB2D ;;CALL
; * RETURN ;;ADDRESS OF THE 1ST ASCIZ CHAR. IS ON THE STACK

```

```

(1) 016460 016637 000002 016504
(1) 016466 012746 016504
(1) 016472 004737 016510
(1) 016476 012666 000002
(1) 016502 000207
(1) 016504 000000 000000

```

```

$SB2D: MOV 2(SP),1$ ;;SAVE BINARY NUMBER
MOV #1$, -(SP) ;;SET POINTER
JSR PC, @#$DB2D ;;CALL DOUBLE LENGTH CONVERT
MOV (SP)+, 2(SP) ;;PICKUP POINTER
RTS PC ;;RETURN
1$: .WORD 0,0

```

.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, @#$DB2D
; * RETURN ;;THE FIRST ADDRESS OF ASCIZ
; * IS ON THE STACK

```



```

(1)
(1)
(1) 016510 104416          $D820: SAVREG          ;; SAVE REGISTERS
(1) 016512 016602 000002  MOV      2(SP),R2      ;; PICKUP THE DATA POINTER
(1) 016516 012700 016670  MOV      #SDECVL,R0    ;; GET ADDRESS OF "SDECVL" STRING
(1) 016522 010066 000002  MOV      R0,2(SP)     ;; PUT ADDRESS OF ASCII STRING ON STACK
(1) 016526 012201          MOV      (R2)+,R1     ;; PICKUP THE BINARY NUMBER
(1) 016530 012202          MOV      (R2)+,R2
(1) 016532 012737 000012 016606  MOV      #10.,4$     ;; SET UP TO DO 10 CONVERSIONS
(1) 016540 012704 016620  MOV      #STNPNR,R4   ;; ADDRESS OF TEN POWER
(1) 016544 012705 016622  MOV      #STNPNR+2,R5
(1) 016550 005003          1$: CLR      R3          ;; CLEAR PARTIAL
(1) 016552 161401          2$: SUB      (R4),R1     ;; SUBTRACT TEN POWER
(1) 016554 005602          SBC      R2
(1) 016556 161502          SUB      (R5),R2
(1) 016560 002402          BLT      3$          ;; BR IF TEN POWER TO LARGE
(1) 016562 005203          INC      R3          ;; ADD 1 TO PARTIAL
(1) 016564 000772          BR      2$          ;; LOOP
(1) 016566 062401          3$: ADD      (R4)+,R1   ;; RESTORE SUBTRACTED VALUE
(1) 016570 005502          ADC      R2
(1) 016572 062402          ADD      (R4)+,R2
(1) 016574 022525          CMP      (R5)+,(R5)+ ;; MOVE TO NEXT TEN POWER
(1) 016576 052703 000060  BIS      #'0,R3      ;; CHANGE PARTIAL TO ASCII
(1) 016602 110320          MOVB    R3,(R0)+    ;; SAVE IT
(1) 016604 005327          DEC      (PC)+      ;; DONE?
(1) 016606 000000          4$: .WORD 0
(1) 016610 001357          BNE     1$          ;; BR IF NO
(1) 016612 105020          CLRB    (R0)+      ;; TERMINATOR
(1) 016614 104420          RESREG          ;; RESTORE REGISTERS
(1) 016616 000207          RTS     PC         ;; RETURN
(1) 016620 145000          STNPNR: 145000     ;; 1.0E09
(1) 016622 035632          35632
(1) 016624 160400          160400           ;; 1.0E08
(1) 016626 002765          2765
(1) 016630 113200          113200           ;; 1.0E07
(1) 016632 000230          230
(1) 016634 041100          041100           ;; 1.0E06
(1) 016636 000017          17
(1) 016640 103240          103240           ;; 1.0E05
(1) 016642 000001          1
(1) 016644 023420          23420           ;; 1.0E04
(1) 016646 000000          0
(1) 016650 001750          1750           ;; 1.0E03
(1) 016652 000000          0
(1) 016654 000144          144           ;; 1.0E02
(1) 016656 000000          0
(1) 016660 000012          12           ;; 1.0E01
(1) 016662 000000          0
(1) 016664 000001          1           ;; 1.0E00
(1) 016666 000000          0
(1) 016670 000014          SDECVL: .BLKB 12. ;; RESERVE STORAGE FOR ASCII STRING
7278 ;*****
(1)
(1) .SBTTL TYPE NUMERICAL ASCII STRING SUPPRESS LEADING ZEROS

```



```

(1) 016766 012746 000021      MOV      #17,-(SP)      ;; SETUP THE ITERATION COUNTER
(1) 016772 016601 000024      MOV      24(SP),R1     ;; PICKUP THE DIVIDEND
(1) 016776 016600 000022      MOV      22(SP),R0
(1) 017002 100005              BPL      1$           ;; CHECK THE SIGN
(1) 017004 105366 000003      DECB     3(SP)        ;; KEEP TRACK OF THE SIGN
(1) 017010 005400              NEG      R0           ;; AND NEGATE THE ORIGINAL
(1) 017012 005401              NEG      R1           ;; NUMBER
(1) 017014 005600              SBC      R0
(1) 017016 016602 000020      1$:     MOV      20(SP),R2     ;; PICKUP THE DIVISOR
(1) 017022 002407              BLT      2$           ;; CHECK THE SIGN
(1) 017024 003011              BGT      3$           ;; DIVISOR OF 0 IS A NO-NO
(1) 017026 052766 000003 000014  BIS      #3,14(SP)    ;; SET "V" & "C"
(1) 017034 012700 177777      MOV      #-1,R0      ;; SET REMAINDER TO ALL ONES
(1) 017040 000424              BR       7$           ;; EXIT
(1) 017042 005266 000002      2$:     INC      2(SP)     ;; KEEP TRACK OF DIVISORS SIGN
(1) 017046 000401              BR       4$
(1) 017050 005402              3$:     NEG      R2           ;; NEGATE THE ORIGINAL NUMBER
(1) 017052 000241              4$:     CLC                    ;; CLEAR "C"
(1) 017054 000405              BR       6$           ;; START FORMING QUOTIENT
(1) 017056 006100              5$:     ROL      R0           ;; POSITION MSB'S
(1) 017060 010003              MOV      R0,R3        ;; COPY
(1) 017062 060203              ADD      R2,R3        ;; COMPARE DIVIDEND & DIVISOR
(1) 017064 103001              BCC      6$           ;; BR IF DIVIDEND > DIVISOR
(1) 017066 010300              MOV      R3,R0        ;; REMAINDER AFTER THIS LOOP
(1) 017070 006101              6$:     ROL      R1           ;; QUOTIENT BIT ENTERS HERE
(1) 017072 005316              DEC      (SP)         ;; DONE?
(1) 017074 001370              BNE      5$           ;; BR IF NO
(1) 017076 005701              TST      R1           ;; OVERFLOW?
(1) 017100 100005              BPL      8$           ;; BR IF NO
(1) 017102 052766 000002 000014  BIS      #2,14(SP)    ;; SET "V" IN RETURN STATUS WORD
(1) 017110 005000              CLR      R0           ;; SET REMAINDER TO ALL ZEROS
(1) 017112 010001              7$:     MOV      R0,R1        ;; COPY REMAINDER INTO QUOTIENT
(1) 017114 005726              8$:     TST      (SP)+      ;; CLEAR COUNTER FROM STACK
(1) 017116 005716              TST      (SP)         ;; REMAINDER SIGN CORRECTION NEEDED?
(1) 017120 002004              BGE      9$           ;; BR IF NO
(1) 017122 005400              NEG      R0           ;; NEGATE REMAINDER
(1) 017124 105066 000001      CLRB     1(SP)        ;; CLEAR SIGN
(1) 017130 005316              DEC      (SP)         ;; BUT DON'T FORGET QUOTIENT
(1) 017132 005726              9$:     TST      (SP)+      ;; QUOTIENT SIGN CORRECTION NEEDED?
(1) 017134 001401              BEQ      10$          ;; BR IF NO
(1) 017136 005401              NEG      R1           ;; NEGATE QUOTIENT
(1) 017140 010166 000020      10$:    MOV      R1,20(SP)    ;; RETURN QUOTIENT AND
(1) 017144 010066 000016      MOV      R0,16(SP)   ;; REMAINDER TO USER
(3) 017150 012603              MOV      (SP)+,R3    ;; POP STACK INTO R3
(3) 017152 012602              MOV      (SP)+,R2    ;; POP STACK INTO R2
(3) 017154 012601              MOV      (SP)+,R1    ;; POP STACK INTO R1
(3) 017156 012600              MOV      (SP)+,R0    ;; POP STACK INTO R0
(1) 017160 012666 000002      MOV      (SP)+,2(SP) ;; SETUP TO RETURN CONDITION CODES
(1) 017164 000002              RTI                    ;; RETURN

```

7280
(1)
(1)
(1)
(1)

```

;*****
.SBTTL  RANDOM NUMBER GENERATOR ROUTINE
;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR

```

```

(1) ;*WITH A RANGE OF 0 TO 2(+33)-1.
(1) ;*CALL:
(1) ;* JSR PC,$RAND ;:CALL THE ROUTINE
(1) ;* RETURN ;:RETURN HERE THE RANDOM
(1) ;* ;:NUMBER WILL BE IN
(1) ;* ;:$HINUM,$LONUM
(1)
(2) 017166 $RAND:
(3) 017166 010046 MOV R0,-(SP) ;:PUSH R0 ON STACK
(3) 017170 010146 MOV R1,-(SP) ;:PUSH R1 ON STACK
(3) 017172 010246 MOV R2,-(SP) ;:PUSH R2 ON STACK
(3) 017174 010346 MOV R3,-(SP) ;:PUSH R3 ON STACK
(1) 017176 013700 017314 MOV $LONUM,R0 ;:SET R0 WITH LOW
(1) 017202 013701 017312 MOV $HINUM,R1 ;:SET R1 WITH HIGH
(1) 017206 012703 177771 MOV #-7,R3 ;:SET SHIFT COUNT
(1) 017212 005002 CLR R2 ;:ZERO R2
(1) 017214 006300 15: ASL R0 ;:SHIFT R0 LEFT AND
(1) 017216 006101 ROL R1 ;:ROTATE CARRY INTO R1 AND
(1) 017220 006102 ROL R2 ;:ROTATE CARRY INTO R2
(1) 017222 005203 INC R3 ;:CHECK FOR DONE
(1) 017224 001373 BNE 15 ;:CONTINUE SHIFT LOOP
(1) 017226 063700 017314 ADD $LONUM,R0 ;:ADD NUMBER TO MAKE X 129
(1) 017232 005501 ADC R1 ;:PROPOGATE CARRY
(1) 017234 063701 017312 ADD $HINUM,R1 ;:ADD NUMBER TO MAKE X 129
(1) 017240 005502 ADC R2 ;:PROPOGATE CARRY
(1) 017242 062700 001057 ADD #1057,R0 ;:ADD LOW CONSTANT
(1) 017246 005501 ADC R1 ;:PROPOGATE CARRY
(1) 017250 005502 ADC R2 ;:PROPOGATE CARRY
(1) 017252 062701 047401 ADD #47401,R1 ;:ADD HIGH CONSTANT
(1) 017256 005502 ADC R2 ;:PROPOGATE CARRY
(1) 017260 062702 000006 ADD #6,R2 ;:ADD HIGHEST CONSTART
(1) 017264 060200 ADD R2,R0 ;:REPRIME R0 WITH HIGHEST DIGIT
(1) 017266 005501 ADC R1 ;:PROPOGATE CARRY
(1) 017270 010037 017314 MOV R0,$LONUM ;:SAVE R0
(1) 017274 010137 017312 MOV R1,$HINUM ;:SAVE R1
(3) 017300 012603 MOV (SP)+,R3 ;:POP STACK INTO R3
(3) 017302 012602 MOV (SP)+,R2 ;:POP STACK INTO R2
(3) 017304 012601 MOV (SP)+,R1 ;:POP STACK INTO R1
(3) 017306 012600 MOV (SP)+,R0 ;:POP STACK INTO R0
(1) 017310 000207 RTS PC ;:RETURN
(1) 017312 176543 $HINUM: .WORD 176543
(1) 017314 123456 $LONUM: .WORD 123456

```

```

7281
7282 ;*****
7283 ;SBTTL LP.AVL - LINE PRINTER AVAILABLE
7284 ;*THIS ROUTINE WILL CHECK IF THERE IS A LINE PRINTER AVAILABLE AND SET
7285 ;*"LPTAVL" TO THE PROPER STATE.
7286 ;* LPTAVL = 0 IF NO LINE PRINTER AVAILABLE
7287 ;* LPTAVL = 1 IF LINE PRINTER IS AVAILABLE
7288 ;*CALL
7289 ;* JSR PC,@#LP.AVL
7290 ;* RETURN
7291
7292 017316 005037 001220 LP.AVL: CLR @#LPTAVL ;:START WITH NO PRINTER AVAIBLE

```

```

7293 017322 012737 046 000004 MOV #15, @#ERRVEC ; SETUP THE TIMEOUT VECTOR
7294 017330 005037 006 CLR @#ERRVEC+2
7295 017334 005777 162020 TST @LPS ; IS THERE A LINE PRINTER?
7296 017340 005237 001220 INC @#LPTAVL ; YES--SET AVAILABLE SWITCH
7297 017344 000401 BR 2$
7298 017346 022626 1$: CMP (SP)+, (SP)+ ; NO--POP STACK
7299 017350 012737 000006 000004 2$: MOV #ERRVEC+2, @#ERRVEC ; RESTORE TIMEOUT VECTOR
7300 017356 000207 RTS PC ; RETURN
7301
7302 ;*****
7303 .SBTTL ST.CLK - CLOCK STARTUP ROUTINE
7304 ;*THIS ROUTINE WILL DETERMINE IF THERE IS A CLOCK ON THE SYSTEM
7305 ;*AND IF THERE IS IT WILL SETUP THE VECTOR AND START THE CLOCK
7306 ;*"CLKSTA" WILL INDICATE THE CLOCK TYPE
7307 ;* 0= NO CLOCK
7308 ;*+1= KW11-P
7309 ;*-1= KW11-L
7310 ;*THIS ROUTINE WILL ALSO SETUP "TICKMS" (TIME
7311 ;*PER CLOCK TICK IN MILLISECONDS) AND "TICKUS"
7312 ;*(TIME PER CLOCK TICK IN MICROSECONDS) AS
7313 ;*PER SW00.
7314 ;*SW00=0 -- 60HZ
7315 ;*SW00=1 -- 50HZ
7316 ;*CALL
7317 ;* JSR PC, @#ST.CLK
7318 ;* RETURN
7319
7320 017360 010146 ST.CLK: MOV R1, -(SP) ; SAVE R1
7321 017362 012701 000006 MOV #ERRVEC+2, R1 ; SAVE AND SETUP TIMEOUT VECTOR
7322 017366 011146 MOV (R1), -(SP)
7323 017370 005011 CLR (R1) ; LEVEL 0
7324 017372 014146 MOV -(R1), -(SP)
7325 017374 012711 017424 MOV #1$, (R1) ; GO TO 1$ ON TIMEOUT
7326 017400 005037 001226 CLR CLKSTA ; SET CLOCK STATUS TO NO CLOCK
7327 017404 005777 161730 TST @PKCS ; IS THERE A KW11-P?
7328 017410 012737 000001 001226 MOV #1, CLKSTA ; YES--SET STATUS TO KW11-P
7329 017416 004737 017526 JSR PC, ST.PCLK ; START THE KW11-P
7330 017422 000414 BR 3$ ; GO TO EXIT
7331 017424 022626 1$: CMP (SP)+, (SP)+ ; CLEAN UP THE STACK
7332 017426 012711 017452 MOV #2$, (R1) ; IF TIMEOUT GO TO 2$
7333 017432 005777 161714 TST @LKS ; IS THERE A KW11-L?
7334 017436 012737 177777 001226 MOV #-1, CLKSTA ; YES-- SET STATUS TO KW11-L
7335 017444 004737 017570 JSR PC, ST.LCLK ; START THE KW11-L
7336 017450 000401 BR 3$ ; EXIT
7337 017452 022626 2$: CMP (SP)+, (SP)+ ; CLEAN UP THE STACK
7338 017454 012621 3$: MOV (SP)+, (R1)+ ; RESTORE THE TIMEOUT VECTOR
7339 017456 012621 MOV (SP)+, (R1)+
7340 017460 012601 MOV (SP)+, R1 ; RESTORE R1
7341 017462 032737 000100 001212 BIT #SW06, @#C.SWR ; 50HZ OR 60HZ?
7342 017470 001407 BEQ 4$ ; BRANCH IF 60
7343 017472 012737 000020 001230 MOV #20, @#TICKMS ; SETUP TIME PER
7344 017500 012737 047040 001232 MOV #20000., @#TICKUS ; TICK FOR 50HZ
7345 017506 000406 BR 5$
7346 017510 012737 000016 001230 4$: MOV #16, @#TICKMS ; SETUP TIME PER

```

```

7347 017516 012737 040432 001232      MOV      #16666.,@#TICKUS ;TICK FOR 60HZ
7348 017524 000207      SS:      RTS          PC          ;RETURN
7349
7350 017526      ST.PCLK:
7351 017526 032737 000040 001212      BIT      #SW05,@#C.SWR   ;ALLOW SOFTWARE TIMEOUTS?
7352 017534 001014      BNE     15             ;NO--BRANCH
7353 017536 012777 017624 161570      MOV      #SRVCLK,@PKV   ;SETUP THE KW11-P VECTOR
7354 017544 012777 000300 161564      MOV      #300,@PKV+2
7355 017552 012777 000001 161562      MOV      #1,@PKB        ;COUNT ONE TICK
7356 017560 012777 000115 161552      MOV      #115,@PKCS     ;"INT.EN." COUNT DOWN" "MODE 1 (REPEAT)",
7357                                     ;"LINE FREQ", AND "RUN"
7358 017566 000207      15:      RTS          PC          ;RETURN
7359
7360 017570      ST.LCLK:
7361 017570 032737 000040 001212      BIT      #SW05,@#C.SWR   ;ALLOW SOFTWARE TIMEOUTS?
7362 017576 001011      BNE     15             ;NO--BRANCH
7363 017600 012777 017624 161540      MOV      #SRVCLK,@LKV   ;SETUP THE KW11-L VECTOR
7364 017606 012777 000300 161534      MOV      #300,@LKV+2
7365 017614 012777 000100 161530      MOV      #100,@LKS      ;START THE KW11-L
7366 017622 000207      15:      RTS          PC          ;RETURN
7367
7368 017624 013746 001230      SRVCLK: MOV      @#TICKMS,-(SP) ;TIME PER TICK IN MILLISECONDS
7369 017630 004037 032534      JSR     RO,@#RPTMR      ;COUNT THE ELAPSED TIME
7370 017634 000002      RTI          ;RETURN AFTER INTERRUPT
7371
7372                                     ;*****
7373                                     ;SBTTL LDCMD - LOAD COMMAND
7374                                     ;*THIS ROUTINE LOADS A READ HEADER AND DATA COMMAND OR A SEEK COMMAND
7375                                     ;*INTO DPB.B+2 AND DPB.C+2, DEPENDING ON THE STATE OF "CONTROL SWITCH"
7376                                     ;*BIT07.
7377                                     ;*CALL
7378                                     ;*   JSR      PC,@#LDCMD
7379                                     ;*   RETURN
7380
7381 017636 032737 000200 001212      LDCMD:  BIT      #SW07,@#C.SWR ;DO EXPLICIT SEEKS?
7382 017644 001007      BNE     15             ;YES--BRANCH
7383 017646 012737 000173 003462      MOV      #READHD,@#DPB.B+2 ;NO--SET UP FOR READ HEADER AND
7384 017654 012737 000173 003502      MOV      #READHD,@#DPB.C+2 ;DATA COMMAND
7385 017662 000406      BR      25
7386 017664 012737 000105 003462      15:     MOV      #SEEK,@#DPB.B+2 ;SETUP FOR SEEK COMMAND
7387 017672 012737 000105 003502      MOV      #SEEK,@#DPB.C+2
7388 017700 000207      25:     RTS          PC
7389
7390                                     ;*****
7391                                     ;SBTTL CALL.A - CALL RPO4 DRIVER USING "DPB.A"
7392                                     ;*THIS ROUTINE WILL CALL THE RPO4 DRIVER AND THEN WAIT ON THE FUNCTION
7393                                     ;*TO COMPLETE. IF AN ERROR OCCURS IT IS REPORTED.
7394                                     ;*CALL
7395                                     ;*   FILL "DPB" WITH COMMAND INFORMATION
7396                                     ;*   JSR     RO,@#CALL.A
7397                                     ;*   RETURN
7398
7399 017702 005037 001200      CALL.A: CLR      @#$ESCAPE ;NO ESCAPE ADDRESS
7400 017706 004037 027574      JSR     RO,@#RPO4      ;CALL RPO4 DRIVER

```

```

7401 017712 003440          DPB.A
7402 017714 000732          BR
7403 017716 005737 003456 15:  TST      2#DPB.A+16      :DONE?
7404 017722 001775          BEQ      15              :NO--LOOP
7405 017724 100030          BPL      35              :BRANCH IF NO ERROR
7406 017726 012737 020002 301200  MOV      2#5,SESCAPE     :ESCAPE TO 2$ ON ERROR
7407 017734 013737 003452 001252  MOV      2#DPB.A+12,2#CYL.DS :CYLINDER
(1) 017742 113737 003451 001255  MOVVB   2#DPB.A+11,2#TRK.DS :TRACK
(1) 017750 113737 003450 001254  MOVVB   2#DPB.A+10,2#SEC.DS :SECTOR
(1) 017756 012746 003456          MOV      #DPB.A+16,-(SP) :STATUS/ERROR INDICATOR ADDRESS
(1) 017762 004737 020766          JSR      PC,2#ERINDX    :FORM DISPATCH INDEX
(1) 017765 062607          ADD      (SP)+,PC      :REPORT PROPER ERROR
(1) 017770 104041          ERROR   41
(1) 017772 104042          ERROR   42              :PARITY ERROR
(1) 017774 104043          ERROR   43              :UNSAFE ERROR
(1) 017776 104044          ERROR   44              :NON-I/O ERROR
(1) 020000 104045          ERROR   45              :I/O ERROR
7408 020002 013700 001234 2$:  MOV      2#BYPASS,RO    :TAKE ERROR EXIT
7409 020006 000200          3$:  RTS      RO          :RETURN

```

```

;*****
;SBTTL CALL.B - CALL RPO4 DRIVER USING "DPB.B"
;THIS ROUTINE IS THE SAME AS "CALL.A" EXCEPT FOR THE DPB USED AND IF
;THE COMMAND IS A READ HEADER AND DATA THE HEADER (CYLINDER, TRACK,
;AND SECTOR) READ IS CHECKED FOR VALIDITY.
;CALL
;*
;* FILL DPB
;* JSR      RO,2#CALL.B
;* RETURN

```

```

7421 020010 005037 001200  CALL.B: CLR      2#SESCAPE     :NO ESCAPE ADDRESS
7422 020014 004037 027574          JSR      RO,2#RPO4      :CALL RPO4 DRIVER
7423 020020 003460          DPB.B
7424 020022 000772          BR      CALL.B
7425 020024 005737 003476 15:  TST      DPB.B+16      :DONE?
7426 020030 001775          BEQ      15              :NO--BRANCH
7427 020032 100031          BPL      35              :BRANCH IF NO ERROR
7428 020034 012737 020110 001200  MOV      2#5,SESCAPE     :ESCAPE TO 2$ ON ERROR
7429 020042 013737 003472 001252  MOV      2#DPB.B+12,2#CYL.DS :CYLINDER
(1) 020050 113737 003471 001256  MOVVB   2#DPB.B+11,2#TRK.DS :TRACK
(1) 020056 113737 003470 001254  MOVVB   2#DPB.B+10,2#SEC.DS :SECTOR
(1) 020064 012746 003476          MOV      #DPB.B+16,-(SP) :STATUS/ERROR INDICATOR ADDRESS
(1) 020070 004737 020766          JSR      PC,2#ERINDX    :FORM DISPATCH INDEX
(1) 020074 062607          ADD      (SP)+,PC      :REPORT PROPER ERROR
(1) 020076 104041          ERROR   41
(1) 020100 104042          ERROR   42              :PARITY ERROR
(1) 020102 104043          ERROR   43              :UNSAFE ERROR
(1) 020104 104044          ERROR   44              :NON-I/O ERROR
(1) 020106 104045          ERROR   45              :I/O ERROR
7430 020110 013700 001234 2$:  MOV      2#BYPASS,RO    :TAKE ERROR EXIT
7431 020114 000407          BR      45
7432 020116 123727 003462 000173 3$:  CMPB   2#DPB.B+2,#READHD :DOING IMPLIED SEEKS?
7433 020124 001003          BNE     45              :NO--BRANCH
7434 020126 004037 021246          JSR      RO,2#VERIFY    :YES--GO CHECK THE DATA

```

```

7435 020132 003470          DPB.B+10
7436 020134 032737 040000 001212 45:  BIT      #SW14,2#C.SWR  ;STALL?
7437 020142 001403          BEQ      55          ;NO--BRANCH
7438 020144 004037 021104          JSR      RO,2#STALL ;YES--CALL STALL ROUTINE
7439 020150 001364          .WORD   STALL1     ;STALL TIME POINTER
7440 020152 000200          55:    RTS      RO      ;RETURN

```

```

7441
7442 ;*****
7443 .SBTTL CALL.C - CALL RPO4 DRIVER USING "DPB.C"
7444 ;*THIS ROUTINE IS THE SAME AS "CALL.B" EXCEPT FOR THE DPB USED.
7445 ;*CALL
7446 ;*      FILL DPB
7447 ;*      JSR      RO,2#CALL.C
7448 ;*      RETURN
7449

```

```

7450 020154 005037 031200          CALL.C: CLR      2#SESCAPE ;NO ESCAPE ADDRESS
7451 020160 004037 027574          JSR      RO,2#RPO4   ;CALL RPO4 DRIVER
7452 020164 003500          DPB.C
7453 020166 000772          BR      CALL.C
7454 020170 005737 003516          15:    TST      2#DPB.C+16 ;DONE?
7455 020174 001775          BEQ      15          ;NO--LOOP
7456 020176 100031          BPL      35          ;YES--BRANCH IF NO ERROR
7457 020200 012737 020254 001200          MOV      25,SESCAPE ;ESCAPE TO 25 ON ERROR
7458 020206 013737 003512 001252          MOV      2#DPB.C+12,2#CYL.DS ;CYLINDER
(1) 020214 113737 003511 001256          MOV      2#DPB.C+11,2#TRK.DS ;TRACK
(1) 020222 113737 003510 001254          MOV      2#DPB.C+10,2#SEC.DS ;SECTOR
(1) 020230 012746 003516          MOV      2#DPB.C+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
(1) 020234 004737 020766          JSR      PC,2#ERINDX ;FORM DISPATCH INDEX
(1) 020240 062607          ADD      (SP)+,PC   ;REPORT PROPER ERROR
(1) 020242 104041          ERROR   41
(1) 020244 104042          ERROR   42          ;PARITY ERROR
(1) 020246 104043          ERROR   43          ;UNSAFE ERROR
(1) 020250 104044          ERROR   44          ;NON-I/O ERROR
(1) 020252 104045          ERROR   45          ;I/O ERROR
7459 020254 013700 001234          25:    MOV      2#BYPASS,RO ;TAKE ERROR EXIT
7460 020260 000400          BR      35
7461 020262 123727 003502 000173          35:    CMPB    2#DPB.C+2,#READHD ;DOING IMPLIED SEEK?
7462 020270 001003          BNE     45          ;NO--EXIT
7463 020272 004037 021246          JSR      RC,2#VERIFY ;YES--CHECK THE DATA
7464 020276 003510          DPB.C+10
7465 020300 032737 040000 001212          45:    BIT      #SW14,2#C.SWR ;STALL?
7466 020306 001403          BEQ      55          ;NO--BRANCH
7467 020310 004037 021104          JSR      RO,2#STALL ;YES--CALL STALL ROUTINE
7468 020314 001364          .WORD   STALL1     ;STALL TIME POINTER
7469 020316 000200          55:    RTS      RO

```

```

7470
7471 ;*****
7472 .SBTTL DRVCAL - DRIVER (RPO4) CALL USING "DTADPB"
7473 ;*THIS ROUTINE IS THE SAME AS "CALL.A" EXCEPT FOR THE DPB USED AND
7474 ;*ON AN ERROR LOCATION "ERR.CT" IS EXAMINED. IF ERR.CT IS EQUAL TO
7475 ;*SERFLG EXIT IS TO THE NEXT TEST.
7476 ;*CALL
7477 ;*      FILL DPB
7478

```



```

7479      ;*      JSR      RO,2#DRVCAL
7480      ;*      RETURN
7481
7482      020320 005037 001200      DRVCAL: CLR      2#SESCAPE      ;NO ESCAPE ADDRESS
7483      020324 005037 001316      CLR      2#WCEFLG      ;CLEAR WRITE CHECK ERROR FLAG
7484      020330 004037 027574      JSR      RO,2#RPO4      ;CALL RPO4 DRIVER
7485      020334 003520      DTADPB
7486      020336 000770      BR      DRVCAL
7487      020340 005737 003536      15:  TST      2#DTADPB+16      ;DONE
7488      020344 001775      BEQ      15              ;NO--LOOP
7489      020346 100177      BPL      105             ;YES--BRANCH IF NO ERRORS
7490      020350 012737 020424 001200  MOV      25,SESCAPE      ;ESCAPE TO 25 ON ERROR
7491      020356 013737 003532 001252  MOV      2#DTADPB+12,2#CYL.DS ;CYLINDER
(1)      020364 113737 003531 001256  MOVB    2#DTADPB+11,2#TRK.DS ;TRACK
(1)      020372 113737 003530 001254  MOVB    2#DTADPB+10,2#SEC.DS ;SECTOR
(1)      020400 012746 003536      MOV      2#DTADPB+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
(1)      020404 004737 020766      JSR      PC,2#ERINDX     ;FORM DISPATCH INDEX
(1)      020410 062607      ADD      (SP)+,PC        ;REPORT PROPER ERROR
(1)      020412 104041      ERROR   41
(1)      020414 104042      ERROR   42              ;PARITY ERROR
(1)      020416 104043      ERROR   43              ;UNSAFE ERROR
(1)      020420 104044      ERROR   44              ;NON-I/O ERROR
(1)      020422 104045      ERROR   45              ;I/O ERROR
7492      020424 122737 000015 001102  25:  CMPB    15,2#STSTNM     ;TEST 15?
7493      020432 001133      BNE      85              ;NO--BRANCH
7494      020434 122737 000151 003522  CMPB    2#WRCKD,2#DTADPB+2 ;DOING A WRITE CHECK?
7495      020442 001127      BNE      85              ;NO--BRANCH
7496      020444 032737 040000 003550  BIT     2#BIT14,2#RP.REG+10 ;IS "WCE"=1?
7497      020452 001523      BEQ      85              ;NO--BRANCH
7498      020454 032737 000020 177570  BIT     2#SW04,2#SWR      ;INHIBIT WRITES?
7499      020462 001117      BNE      85              ;YES--BRANCH
7500      020464 112737 000161 003522  MOVB    2#WRITE,2#DTADPB+2 ;SETUP FOR A WRITE
7501      020472 005037 001200      CLR      2#SESCAPE      ;NO ESCAPE ADDRESS
7502      020476 004037 027574      JSR      RO,2#RPO4      ;DO THE WRITE
7503      020502 003520      DTADPB
7504      020504 000240      NOP
7505      020506 005737 003536      35:  TST      2#DTADPB+16      ;DONE?
7506      020512 001775      BEQ      35              ;NO--LOOP
7507      020514 100026      BPL      45              ;YES--BRANCH IF NO ERROR
7508      020516 012737 020722 001200  MOV      35,SESCAPE      ;ESCAPE TO 35 ON ERROR
7509      020524 013737 003532 001252  MOV      2#DTADPB+12,2#CYL.DS ;CYLINDER
(1)      020532 113737 003531 001256  MOVB    2#DTADPB+11,2#TRK.DS ;TRACK
(1)      020540 113737 003530 001254  MOVB    2#DTADPB+10,2#SEC.DS ;SECTOR
(1)      020546 012746 003536      MOV      2#DTADPB+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
(1)      020552 004737 020766      JSR      PC,2#ERINDX     ;FORM DISPATCH INDEX
(1)      020556 062607      ADD      (SP)+,PC        ;REPORT PROPER ERROR
(1)      020560 104041      ERROR   41
(1)      020562 104042      ERROR   42              ;PARITY ERROR
(1)      020564 104043      ERROR   43              ;UNSAFE ERROR
(1)      020566 104044      ERROR   44              ;NON-I/O ERROR
(1)      020570 104045      ERROR   45              ;I/O ERROR
7510      020572 112737 000151 003522  45:  MOVB    2#WRCKD,2#DTADPB+2 ;COMMAND=WRITE CHECK DATA
7511      020600 004037 027574      JSR      RO,2#RPO4      ;DO THE WRITE CHECK
7512      020604 003520      DTADPB

```

```

7513 020606 000240      NOP
7514 020610 005737 003536 5$:  TST      @#DTADPB+16      ;DONE?
7515 020614 001775      BEQ      5$                ;NO--LOOP
7516 020616 100410      BMI      7$                ;YES--BRANCH IF ERROR
7517 020620 004037 027574      JSR      RO,@#RPO4        ;DO A 2ND WRITE CHECK
7518 020624 003520      DTADPB
7519 020626 000240      NOP
7520 020630 005737 003536 6$:  TST      @#DTADPB+16      ;DONE?
7521 020634 001775      BEQ      6$                ;NO--LOOP
7522 020636 100031      BPL      8$                ;YES--BRANCH IF NO ERROR
7523 020640 012737 000001 001316 7$:  MOV      #1,@#WCEFLG      ;SET THE WRITE CHECK ERROR FLAG
7524 020646 012737 020722 001200      MOV      @#$,@#SECAPE     ;ESCAPE TO 8$ ON ERROR
7525 020654 013737 003532 001252      MOV      @#DTADPB+12,@#CYL.DS ;CYLINDER
(1) 020662 113737 003531 001256      MOV      @#DTADPB+11,@#TRK.DS ;TRACK
(1) 020670 113737 003530 001254      MOV      @#DTADPB+10,@#SEC.DS ;SECTOR
(1) 020676 012746 003536      MOV      @#DTADPB+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
(1) 020702 004737 020766      JSR      PC,@#ERINDX      ;FORM DISPATCH INDEX
(1) 020706 062607      ADD      (SP)+,PC         ;REPORT PROPER ERROR
(1) 020710 104041      ERROR   41
(1) 020712 104042      ERROR   42                ;PARITY ERROR
(1) 020714 104043      ERROR   43                ;UNSAFE ERROR
(1) 020716 104044      ERROR   44                ;NON-I/O ERROR
(1) 020720 104046      ERROR   46                ;FATAL WRITE CHECK
7526 020722 032737 001000 177570 8$:  BIT      @#SW09,@#SWR      ;LOOP ON ERROR?
7527 020730 001004      BNE      9$                ;YES--BRANCH
7528 020732 123737 001372 001103      CMPB    @#ERR.CT,@#SERFLG ;GO TO NEXT TEST?
7529 020740 101002      BHI      10$               ;NO--BRANCH
7530 020742 013700 001234 9$:  MOV      @#BYPASS,RO      ;YES--GET EXIT ADDRESS
7531 020746 032737 040000 001212 10$: BIT      @#SW14,@#C.SWR    ;STALL?
7532 020754 001403      BEQ      11$               ;NO--BRANCH
7533 020756 004037 021104      JSR      RO,@#STALL       ;YES--CALL STALL ROUTINE
7534 020762 001366      .WORD   STALL2            ;STALL TIME POINTER
7535 020764 000200      11$:   RTS      RO
7536
7537 ;*****
7538 ;SBTTL ERINDX - FORM ERROR INDEX
7539 ;*THIS ROUTINE FORMS AN INDEX THAT WILL BE USED TO DISPATCH
7540 ;*TO THE PROPER ERROR CALL. THE INDEX IS FORMED BY EXAMINING
7541 ;*THE STATUS/ERROR INDICATOR OF THE APPLICABLE DPB.
7542 ;*INDEX
7543 ;-----
7544 ;*      0      BIT14!BIT13!BIT08
7545 ;*      2      BIT11!BIT10
7546 ;*      4      BIT12!BIT04
7547 ;*      6      BIT05!BIT03!<BIT09 & COMMAND=NON-I/O>
7548 ;*     10     BIT06!<BIT09 & COMMAND=I/O>
7549 ;*CALL
7550 ;*      JSR      @#DPB+16,-(SP) ;ADDRESS OF STATUS/ERROR INDICATOR
7551 ;*      JSR      PC,@#ERINDX   ;FORM INDEX
7552 ;*      RETURN                      ;INDEX IS ON THE STACK
7553
7554 020766 010046      ERINDX: MOV      RO,-(SP)      ;SAVE RO
7555 020770 010146      MOV      R1,-(SP)        ;SAVE R1
7556 020772 016600 000006      MOV      6(SP),RO        ;GET STATUS/ERROR INDICATOR POINTER

```

```

7557 020776 011037 001242      MOV      (R0),@#SVSTAT      ;SAVE THE STATUS/ERROR INDICATOR
7558 021002 005001      CLR      R1                  ;START INDEX AT ZERO
7559 021004 032710      BIT      (PC)+ (R0)         ;FORM INDEX OF 0?
7560 021006 060400      .WORD   BIT14!BIT13!BIT08
7561 021010 001027      BNE      5$                  ;YES--BRANCH
7562 021012 032710      BIT      (PC)+ (R0)         ;FORM PARITY ERROR INDEX (2)?
7563 021014 006000      .WORD   BIT11!BIT10
7564 021016 001023      BNE      4$                  ;YES--BRANCH
7565 021020 032710      BIT      (PC)+ (R0)         ;FORM UNSAFE INDEX (4)?
7566 021022 010020      .WORD   BIT12!BIT04
7567 021024 001017      BNE      3$                  ;YES--BRANCH
7568 021026 032710      BIT      (PC)+ (R0)         ;FORM NON-I/O ERROR INDEX (6)?
7569 021030 000050      .WORD   BIT05!BIT03
7570 021032 001013      BNE      2$                  ;YES--BRANCH
7571 021034 032710      BIT      (PC)+ (R0)         ;FORM I/O ERROR INDEX (10)?
7572 021036 000100      .WORD   BIT06
7573 021040 001007      BNE      1$                  ;YES--BRANCH
7574 021042 032710      BIT      (PC)+ (R0)         ;SOFTWARE TIMEOUT?
7575 021044 001000      .WORD   BIT09
7576 021046 001410      BEQ      5$                  ;NO--FORM INDEX OF 0
7577 021050 122760 000150 177762      CMPB    #150,-16(R0)       ;YES--I/O?
7578 021056 003001      BGT      2$                  ;NO--BRANCH
7579 021060 005201      1$: INC      R1                ;INDEX=10---ERROR=45 OR 46
7580 021062 005201      2$: INC      R1                ;INDEX=6---ERROR=44
7581 021064 005201      3$: INC      R1                ;INDEX=4---ERROR=43
7582 021066 005201      4$: INC      R1                ;INDEX=2---ERROR=42
7583 021070 006301      5$: ASL      R1                ;INDEX=0---ERROR=41
7584 021072 010166 000006      MOV      R1,6(SP)           ;RETURN INDEX TO USER
7585 021076 012601      MOV      (SP)+,R1           ;RESTORE R1
7586 021100 012600      MOV      (SP)+,R0           ;RESTORE R0
7587 021102 000207      RTS      PC                  ;RETURN FROM CALL

```

```

:*****
:SBTTL STALL - USED TO STALL AFTER A DRIVE FUNCTION IS COMPLETE
:*THIS ROUTINE WILL PROVIDE A STALL IN MILLISECONDS FOR A SPECIFIC
:*AMOUNT OF TIME IF BIT13 OF C.SWR = 0 OR A RANDOM AMOUNT OF TIME
:*IF BIT 13 OF C.SWR = 1.
:*STALL1 CONTAINS SPECIFIED TIME FOR TESTS 0-6 AND STALL2
:*CONTAINS THE TIME FOR TESTS 13-16.
:*CALL

```

```

7597      *      JSR      R0,@#STALL
7598      *      TIME POINTER      ;WHERE TO FIND THE STALL TIME
7599
7600 021104 013046      STALL: MOV      @ (R0)+, -(SP)    ;PICKUP STALL TIME
7601 021106 032737 020000 001212      BIT      @SW13,@#C.SWR      ;USE A RANDOM TIME?
7602 021114 001406      BEQ      1$                  ;NO--BRANCH
7603 021116 004737 017166      JSR      PC,@#SRAND          ;YES--FORM RANDOM NUMBER
7604 021122 013716 017314      MOV      @#$LONUM,(SP)      ;AND USE IT FOR THE STALL TIME
7605 021126 042716 177700      BIC      @#C77,(SP)         ;BUT NEVER > 64 MILLISECONDS
7606 021132 005046      1$: CLR      -(SP)           ;CLEAR TEMP. LOCATION
7607 021134 162766 000001 000002      2$: SUB      #1,2(SP)         ;MORE STALL REQUIRED?
7608 021142 103407      BLO      4$                  ;NO--BRANCH
7609 021144 012716 000144      MOV      @#C100,(SP)        ;STALL FOR ABOUT 1 MILLISECOND
7610 021150 005700      3$: TST      R0              ;NOP TO KILL TIME

```

STALL - USED TO STALL AFTER A DRIVE FUNCTION IS COMPLETE

```

7611 021152 005366 000000          DEC      0(SP)          ;COUNT
7612 021156 001374          BNE      3$           ;LOOP IF MORE COUNTS NEEDED
7613 021160 000765          BR       2$           ;
7614 021162 022626          4$:    CMP      (SP)+,(SP)+ ;CLEAN OFF THE STACK
7615 021164 000200          RTS      RD           ;EXIT
7616
7617
7618 ;*****
7619 ;SBTTL TWOMS - STALL FOR 2 MS BETWEEN SEEKS IN TESTS 10- - 12
7620 ;ROUTINE TO PROVIDE A 2 MS STALL AFTER A SEEK OPERATION IN THE SEEK TIMING
7621 ;TESTS (TESTS 10, 11, & 12). THIS STALL IS SPECIFIED BY THE VENDOR (ISS)
7622 ;OF THE DISK DRIVE. THIS STALL TIME IS NOT INCLUDED IN THE CALCULATED
7623 ;SEEK TIMES.
7624 ;CALL
7625 ;
7626 ;      JSR      PC,@TWOMS
7627 ;      RETURN
7628 021166 013746 177776          TWOMS:  MOV      @#PS,-(SP) ;SAVE THE PRESENT PROCESSOR STATUS
7629 021172 012737 000240 177776          MOV      @(<5*32.)@#PS ;SET THE PROCESSOR PRIORITY TO 5
7630 021200 017746 160130          MOV      @PKV,-(SP) ;SAVE THE OLD CLOCK VECTOR ADDRESS
7631 021204 012777 021230 160122          MOV      @1$,@PKV ;SETUP NEW VECTOR ADDRESS
7632 021212 012777 000310 160122          MOV      @200,@PKB ;LOAD THE CLOCK BUFFER
7633 021220 012777 000101 160112          MOV      @101,@PKCS ;START THE CLOCK
7634 021226 000001          WAIT ;WAIT FOR 2 MS
7635 021230 062706 000004          1$:    ADD      @4,SP ;INCREMENT STACK FOR RETURN
7636 021234 012677 160074          MOV      (SP)+,@PKV ;RESTORE OLD CLOCK VECTOR
7637 021240 012637 177776          MOV      (SP)+,@#PS ;RESTORE THE OLD PROCESSOR STATUS
7638 021244 000207          RTS      PC           ;RETURN
7639
7640 ;*****
7641 ;SBTTL VERIFY - VALIDATE HEADER ON IMPLIED SEEKS
7642 ;ROUTINE TO SOFTWARE COMPARE HEADER ON IMPLIED SEEKS
7643 ;CALL
7644 ;      JSR      RD,@VERIFY
7645 ;      ADR POINTER ;ADDRESS OF DPB+10 (SECTOR NUMBER)
7646 ;      RETURN
7647
7648 021246 010146          VERIFY: MOV      R1,-(SP) ;SAVE R1
7649 021250 012001          MOV      (R0)+,R1 ;GET ADDRESS OF DPB+10
7650 021252 042737 010000 040524          BIC      @FMT22,@#BUFFER ;STRIP FORMAT BIT FROM CYLINDER NUMBER
7651 021260 023761 040524 000002          CMP      @#BUFFER,2(R1) ;CYLINDER NUMBER OK?
7652 021266 001003          BNE      1$ ;NO--BRANCH
7653 021270 023711 040526          CMP      @#BUFFER+2,(R1) ;YES--HOW ABOUT TRACK/SECTOR?
7654 021274 001426          BEQ      3$ ;BRANCH IF GOOD
7655 021276 013737 040524 001244          1$:    MOV      @#BUFFER,@#CYL,RD ;SAVE THE EXPECTED AND THE
7656 021304 113737 040527 001246          MOV      @#BUFFER+3,@#TRK,RD ;RECIEVED CYLINDER, TRACK,
7657 021312 113737 040526 001250          MOV      @#BUFFER+2,@#SEC,RD ;AND SECTOR
7658 021320 112137 001254          MOV      (R1)+,@#SEC,DS
7659 021324 112137 001256          MOV      (R1)+,@#TRK,DS
7660 021330 011137 001252          MOV      (R1),@#CYL,DS
7661 021334 012737 021346 001200          MOV      @2$, $ESCAPE ;: ESCAPE TO 2$ ON ERROR
7662 021342 005740          TST      -(RD) ;MAKE IT TEST PC+4
7663 021344 104012          ERROR  12 ;REPORT THE ERROR
7664 021346 013700 001234          2$:    MOV      @#BYPASS,R0 ;TAKE ERROR EXIT

```

VERIFY - VALIDATE HEADER ON IMPLIED SEEKS

```

7665 021352 012601          3$:  MOV    (SP)+,R1      ;RESTORE R1
7666 021354 000200          RTS      RO              ;EXIT
7667
7668 ;*****
7669 .SBTTL SRCHOO - INITIALIZE THE DRIVE FOR THE TIMING TESTS
7670 ;*THIS ROUTINE WILL PERFORM A "MASSBUS" INIT. FOLLOWED BY
7671 ;*A "RECALIBRATE" ON THE DRIVE UNDER TEST.
7672 ;*NOTE: THIS ROUTINE DESTROYS R1 AND R4
7673 ;*CALL
7674 ;*   JSR    RO,SRCHOO      ;DO A MASSBUS INIT. AND RECAL
7675 ;*   RETURN1                ;RETURN HERE IF NO ERROR
7676 ;*   RETURN2                ;RETURN HERE ON ERROR
7677
7678 021356 005001          SRCHOO: CLR    R1              ;INCASE OF ERROR (TYPTIM)
7679 021360 005037 177776    CLR    @#PS
7680 021364 012777 031356 005576  MOV    @ISR,@RPEC      ;SETUP INTERRUPT VECTOR
7681 021372 013704 027166    MOV    @#RPAR,R4      ;PICKUP ADDRESS OF RHCS1
7682 021376 012764 000040 000010  MOV    @BIT05,RHCS2(R4) ;MASSBUS INIT.
7683 021404 005037 003530    CLR    @#DTADPB+10    ;TRACK=0; SECTOR=0
7684 021410 005037 003532    CLR    @#DTADPB+12    ;CYLINDER =0
7685 021414 012737 000107 003522  MOV    @#RECAL,@#DADPB+2 ;COMMAND = RECALIBRATE
7686 021422 005037 001200    CLR    @#SESCAPE      ;NO ESCAPE ADDRESS
7687 021426 004037 027574    JSR    RO,@#RPO4      ;CALL THE DRIVER
7688 021432 003520          DTAOPB ;DPB POINTER
7689 021434 000440          BR      4$             ;QUEUE IS FULL
7690 021436 005737 003536 1$:  TST    DTADPB+16      ;WAIT ON DONE
7691 021442 001775          BEQ    1$
7692 021444 100030          BPL    3$             ;TAKE NORMAL EXIT IF NO ERROR
7693 021446 012737 021522 001200  MOV    @#SESCAPE      ;ESCAPE TO 2$ ON ERROR
7694 021454 013737 003532 001252  MOV    @#DTADPB+12,@#CYL.DS ;CYLINDER
(1) 021462 113737 003531 001256  MOVB   @#DTADPB+11,@#TRK.DS ;TRACK
(1) 021470 113737 003530 001254  MOVB   @#DTADPB+10,@#SEC.DS ;SECTOR
(1) 021476 012746 003536    MOV    @#DTADPB+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
(1) 021502 004737 020766    JSR    PC,@#ERINDX    ;FORM DISPATCH INDEX
(1) 021506 062607          ADD    (SP)+,PC      ;REPORT PROPER ERROR
(1) 021510 104041          ERROR  41
(1) 021512 104042          ERROR  42             ;PARITY ERROR
(1) 021514 104043          ERROR  43             ;UNSAFE ERROR
(1) 021516 104044          ERROR  44             ;NON-I/O ERROR
(1) 021520 104045          ERROR  45             ;I/O ERROR
7695 021522 005720          2$:  TST    (RO)+          ;ADJUST FOR ERROR EXIT
7696 021524 000404          BR      4$             ;GO TO THE EXIT
7697 021526 005064 000006  3$:  CLR    RHOA(R4)      ;TRACK AND SECTOR = 0
7698 021532 005064 000034    CLR    RHCA(R4)      ;CYLINDER = 0
7699 021536 000200          4$:  RTS      RO              ;RETURN
7700
7701 ;*****
7702 .SBTTL DORTI - RETURN FROM INTERRUPT
7703 ;*THIS IS AN RTI WHICH IS USED BY THE TIMING TESTS
7704
7705 021540 000002          DORTI: RTI              ;RETURN FROM INTERRUPT
7706
7707 ;*****
7708 .SBTTL STRTMR - START THE TIMERS

```

```

7709      ;*THIS ROUTINE WILL INITIALIZE THE TIMERS
7710      ;*USED BY THE "TIMING ROUTINES"
7711      ;*CALL
7712      ;*      JSR      PC,@#STRTMR
7713      ;*      RETURN
7714
7715      021542 104416      STRTMR: SAVREG      ;SAVE R0-R5
7716      021544 012700 001260      MOV      @TIM.UP,R0      ;START AT TIM.UP (MINIMUM)
7717      021550 012701 001314      MOV      @TIM.PT,R1      ;STOP AT TIM.PT
7718      021554 005020      1$:      CLR      (R0)+      ;CLEAR
7719      021556 020001      CMP      R0,R1      ;DONE?
7720      021560 103775      BLO      1$      ;NO--BRANCH
7721      021562 012710 040524      MOV      @BUFFER,(R0)      ;SETUP POINTER
7722      021566 012737 077777 001260      MOV      @+CBIT15,@#TIM.UP ;SET MINIMUM TIME TO MAXIMUM
7723      021574 012737 077777 001276      MOV      @+CBIT15,@#TIM.DN ;POSITIVE NUMBER
7724      021602 104420      RESREG      ;RESTORE R0-R5
7725      021604 000207      RTS      PC      ;RETURN
7726
7727      ;*****
7728      ;SBTTL COUNT - THIS ROUTINE COUNTS THE ELAPSED TIME
7729      ;*THIS ROUTINE WILL ADD THE ELAPSED TIME TO THE AVERAGE COUNTER AND
7730      ;*MAINTAIN THE MINIMUM AND MAXIMUM TIMES.
7731      ;*NOTE: THIS ROUTINE DESTROYS R2
7732      ;*CALL
7733      ;*      MOV      @TP,R3      ;PARAMETER POINTER
7734      ;*      MOV      FLAG,R5      ;FLAG=0=COUNT UP
7735      ;*      ;FLAG=-1=COUNT DOWN
7736      ;*      JSR      PC,@#COUNT
7737      ;*      RETURN
7738
7739      021606 012702 001260      COUNT: MOV      @TIM.UP,R2      ;PICKUP THE "UP" POINTER
7740      021612 005705      TST      R5      ;USE IT?
7741      021614 001402      BEQ      1$      ;YES--BRANCH
7742      021616 012702 001276      MOV      @TIM.DN,R2      ;NO--PICKUP "DOWN" POINTER
7743      021622 027722 157516      1$:      CMP      @PKC,(R2)+      ;LESS THAN PREVIOUS LOW?
7744      021626 002003      BGE      2$      ;NO--BRANCH
7745      021630 017762 157510 177776      MOV      @PKC,-2(R2)      ;YES--SAVE IT
7746      021636 027763 157502 000004      2$:      CMP      @PKC,4(R3)      ;LESS THAN THE LOW LIMIT?
7747      021644 002001      BGE      3$      ;NO--BRANCH
7748      021646 005212      INC      (R2)      ;YES--COUNT IT
7749      021650 005722      3$:      TST      (R2)+      ;ADVANCE THE POINTER
7750      021652 027722 157466      CMP      @PKC,(R2)+      ;GREATER THAN PREVIOUS HIGH?
7751      021656 003403      BLE      4$      ;NO--BRANCH
7752      021660 017762 157460 177776      MOV      @PKC,-2(R2)      ;YES--SAVE IT
7753      021666 027763 157452 000006      4$:      CMP      @PKC,6(R3)      ;GREATER THAN THE HIGH LIMIT?
7754      021674 003401      BLE      5$      ;NO--BRANCH
7755      021676 005212      INC      (R2)      ;YES--COUNT IT
7756      021700 005722      5$:      TST      (R2)+      ;ADVANCE THE POINTER
7757      021702 067722 157436      ADD      @PKC,(R2)+      ;ADD THIS COUNT TO THE TOTAL
7758      021706 005522      ADC      (R2)+
7759      021710 005212      INC      (R2)      ;COUNT THIS READING
7760      021712 022737 043674 001314      CMP      @BUFFER+<4*410.>,@#TIM.PT ;SAVE THIS COUNT?
7761      021720 101406      BLOS      6$      ;NO--BRANCH
7762      021722 017777 157416 157364      MOV      @PKC,@TIM.PT      ;YES--WELL SAVE IT THEN

```

```

7763 021730 062737 000002 001314      ADD      #2,#TIM.PT      ;ADVANCE THE POINTER
7764 021736 090207                6$:      RTS          PC      ;RETURN
7765
7766 ;*****
7767 ;SBTTL TYPTIM - TYPE TIMES
7768 ;*THIS ROUTINE IS USED TO TYPE THE MINIMUM,
7769 ;*MAXIMUM, AND AVERAGE TIMES FOR TESTS 7,10,11, AND 12.
7770 ;*IT WILL ALSO CHECK THE TIMES TO INSURE
7771 ;*THEY ARE WITHIN TOLERANCE AND IF NOT FLAG THE BAD TIMES.
7772 ;*NOTE: THIS ROUTINE DESTROYS R2-R5
7773 ;*CALL
7774 ;*      JSR      RD,#TYPTIM      ;GO REPORT THE TIMES
7775 ;*      TABLE      ;POINT TO THE PROPER TABLE
7776 ;*      RETURN
7777 ;*
7778 ;*TABLE:MSGADR1      ;ADDRESS OF ASCIZ MESSAGE NUMBER 1
7779 ;*      MSGADR2      ;ADDRESS OF ASCIZ MESSAGE NUMBER 2
7780 ;*      MIN.ALLOWED  ;MINIMUM TIME ALLOWED
7781 ;*      MAX.ALLOWED  ;MAXIMUM TIME ALLOWED
7782
7783 021740 012002                TYPTIM:  MOV      (R0)+,R2      ;PICKUP THE TABLE POINTER
7784 021742 032737 000100 177570      BIT      #SW06,#SWR      ;INHIBIT TIME REPORTS?
7785 021750 001145                BNE      7$              ;YES--BRANCH
7786 021752 012237 021772                MOV      (R2)+,2$        ;ADDRESS OF MESSAGE NUMBER 1
7787 021756 012205                MOV      (R2)+,R5        ;ADDRESS OF MESSAGE NUMBER 2
7788 021760 012203                MOV      (R2)+,R3        ;PICKUP THE LOW LIMIT
7789 021762 011202                MOV      (R2),R2         ;AND THE HIGH LIMIT
7790 021764 012704 001260                MOV      #TIM.UP,R4      ;PARAMETER POINTER
7791 021770 104400                1$:      TYPE          ;TYPE THE MESSAGE
7792 021772 000000                2$:      .WORD      0      ;ASCIZ MESSAGE POINTER GOES HERE
7793 021774 005764 000014                TST      14(R4)          ;DID ANY COUNTS OCCUR?
7794 022000 001527                BEQ      6$              ;NO--BRANCH
7795 022002 104400 035042                TYPE          ,MSGMIN    ;"MIN="
7796 022006 012446                MOV      (R4)+,-(SP)     ;PUT (R4)+ ON THE STACK
(1) 022010 004737 016460                JSR      PC,#$SB2D       ;CHANGE (R4)+ TO DECIMAL ASCIZ
(1) 022014 004737 016704                JSR      PC,#$SUPRS      ;TYPE WITHOUT LEADING ZEROS
7797 022020 104400 035067                TYPE          ,MSGOUS    ;"0 US"
7798 022024 005724                TST      (R4)+          ;ANY SEEKS BELOW THE LOW LIMIT
7799 022026 001421                BEQ      3$              ;NO--BRANCH
7800 022030 104400 035202                TYPE          ,MSG.SP    ;" "
7801 022034 016446 177776                MOV      -2(R4),-(SP)   ;PUT -2(R4) ON THE STACK
(1) 022040 004737 016460                JSR      PC,#$SB2D       ;CHANGE -2(R4) TO DECIMAL ASCIZ
(1) 022044 004737 016704                JSR      PC,#$SUPRS      ;TYPE WITHOUT LEADING ZEROS
7802 022050 104400 035074                TYPE          ,M BELOW  ;"BELOW THE MINIMUM OF"
7803 022054 010346                MOV      R3,-(SP)       ;PUT R3 ON THE STACK
(1) 022056 004737 016460                JSR      PC,#$SB2D       ;CHANGE R3 TO DECIMAL ASCIZ
(1) 022062 004737 016704                JSR      PC,#$SUPRS      ;TYPE WITHOUT LEADING ZEROS
7804 022066 104400 035067                TYPE          ,MSGOUS    ;"MAX="
7805 022072 104400 035051                3$:      TYPE          ;"MAX="
7806 022076 012446                MOV      (R4)+,-(SP)   ;PUT (R4)+ ON THE STACK
(1) 022100 004737 016460                JSR      PC,#$SB2D       ;CHANGE (R4)+ TO DECIMAL ASCIZ
(1) 022104 004737 016704                JSR      PC,#$SUPRS      ;TYPE WITHOUT LEADING ZEROS
7807 022110 104400 035067                TYPE          ,MSGOUS    ;" "
7808 022114 005724                TST      (R4)+          ;ANY SEEKS ABOVE THE HIGH LIMIT

```



```

7809 022116 001421 BEQ 4$ ;NO--BRANCH
7810 022120 104400 035202 TYPE ,MSG.SP ;YES--REPORT HOW MANY
7811 022124 016446 177776 MOV -2(R4),-(SP) ;PUT -2(R4) ON THE STACK
(1) 022130 004737 016460 JSR PC,2$SSB2D ;CHANGE -2(R4) TO DECIMAL ASCIZ
(1) 022134 004737 016704 JSR PC,2$SSUPRS ;TYPE WITHOUT LEADING ZEROS
7812 022140 104400 035123 TYPE ,MABOVE ;"ABOVE THE MAXIMUM OF"
7813 022144 010246 MOV R2, -(SP) ;PUT R2 ON THE STACK
(1) 022146 004737 016460 JSR PC,2$SSB2D ;CHANGE R2 TO DECIMAL ASCIZ
(1) 022152 004737 016704 JSR PC,2$SSUPRS ;TYPE WITHOUT LEADING ZEROS
7814 022156 104400 035067 TYPE ,MSGOUS
7815 022162 104400 035060 4$: TYPE ,MSGAVG ;"AVG="
7816 022166 012446 MOV (R4)+, -(SP) ;FORM THE AVERAGE
7817 022170 012446 MOV (R4)+, -(SP)
7818 022172 012446 MOV (R4)+, -(SP)
7819 022174 004737 016744 JSR PC,2$SDIV
7820 022200 006126 ROL (SP)+ ;IS THE REMAINDER OVER HALF?
7821 022202 100001 BPL 5$ ;NO--BRANCH
7822 022204 005216 INC (SP) ;YES--ROUND UP
7823 022206 5$: JSR PC,2$SSB2D ;CHANGE TO DECIMAL ASCIZ
(1) 022206 004737 016460 JSR PC,2$SSUPRS ;TYPE WITHOUT LEADING ZEROS
(1) 022212 004737 016704 JSR PC,2$SSUPRS
7824 022216 104400 035067 TYPE ,MSGOUS
7825 022222 104400 035202 TYPE ,MSG.SP
7826 022226 016446 177776 MOV -2(R4), -(SP) ;PUT -2(R4) ON THE STACK
(1) 022232 004737 016460 JSR PC,2$SSB2D ;CHANGE -2(R4) TO DECIMAL ASCIZ
(1) 022236 004737 016704 JSR PC,2$SSUPRS ;TYPE WITHOUT LEADING ZEROS
7827 022242 104400 035152 TYPE ,MSGNUM ;"SEEKS TIMED"
7828 022246 010537 021772 MOV R5, 2$ ;NEXT MESSAGE POINTER
7829 022252 001404 BEQ 7$ ;IF NONE EXIT
7830 022254 005005 CLR R5 ;NO MORE THAN 2
7831 022256 000644 BR 1$
7832 022260 104400 035167 6$: TYPE ,MSGNON
7833 022264 000200 7$: RTS R0 ;EXIT
7834
7835 ;*****
7836 .SBTTL INCTRK - INCREMENT TRACK NUMBER
7837 ;*THIS SUBROUTINE WILL INCREMENT THE TRACK
7838 ;*NUMBER (R2) BY THE AMOUNT SPECIFIED BY IT15.
7839 ;*CALL
7840 ;* JSR R0,2$INCTRK
7841 ;* RETURN1 ;TRACK NUMBER GREATER THAN LT15
7842 ;* RETURN2 ;TRACK NUMBER INCREMENTED
7843
7844 INCTRK: CMP R2,2$LT15 ;LAST TRACK COMPLETED?
7845 BEQ 2$ ;YES--EXIT
7846 ADD 2$IT15,R2 ;NO--UPDATE TRACK
7847 CMP R2,2$LT15 ;TRACK TO BIG?
7848 BLE 1$ ;NO--EXIT
7849 MOV 2$LT15,R2 ;YES--SET TRACK TO LAST TRACK
7850 1$: TST (R0)+ ;ADJUST FOR RETURN 2
7851 2$: RTS R0 ;RETURN
7852
7853
7854 ;*****

```



```

7855      .SBTTL INCCYL - INCRMENT CYLINDER NUMBER
7856      ;*THIS SUBROUTINE WILL INCREMENT THE CYLINDER
7857      ;*NUMBER (R1) BY THE AMOUNT SPECIFIED BY IC15.
7858      ;*CALL
7859      ;*      JSR      RO,@#INCCYL
7860      ;*      RETURN1  ;CYLINDER NUMBER GREATER THAN LC15
7861      ;*      RETURN2  ;CYLINDER NUMBER INCREMENTED
7862
7863      INCCYL: CMP      R1,@#LC15      ;LAST CYLINDER COMPLETED?
7864      BEQ      2$      ;YES--EXIT
7865      ADD      @#IC15,R1      ;NO--UPDATE CYLINDER
7866      CMP      R1,@#LC15      ;CYLINDER TO BIG?
7867      BLE      1$      ;NO--EXIT
7868      MOV      @#LC15,R1      ;YES--SET CYLINDER TO LAST CYLINDER
7869      1$:      TST      (R0)+    ;ADJUST FOR RETURN 2
7870      2$:      RTS      RO      ;RETURN
7871
7872      ;*****
7873      .SBTTL FILBUF - FILL BUFFER WITH ADDRESS DATA
7874      ;*THIS SUBROUTINE IS USED TO FILL THE DATA BUFFER
7875      ;*WITH ADDRESSES FROM 0 TO 21 WITH EACH ADDRESS
7876      ;*BEING STORED IN 256 CONSECUTIVE LOCATIONS
7877      ;*CALL
7878      ;*      JSR      PC,@#FILBUF
7879      ;*      RETURN
7880
7881      FILBUF: JSR      RO,@#SAVR05    ;SAVE R0-R5
7882      CLR      RO      ;FIRST DISK ADDRESS
7883      MOV      #BUFFER,R1      ;START FILLING HERE
7884      1$:      MOV      #10256,R2    ;DO 256 WORDS
7885      2$:      MOV      RO,(R1)+    ;STORE
7886      DEC      R2      ;MORE?
7887      BGT      2$      ;YES--BRANCH
7888      INC      RO      ;NO--UPDATE DISK ADDRESS
7889      CMP      #1022,RO      ;DONE?
7890      BGT      1$      ;NO--BRANCH
7891      JSR      RO,@#GETR05    ;RESTORE R0-R5
7892      RTS      PC      ;RETURN
7893
7894      ;*****
7895      .SBTTL CLRBUF - CLEAR BUFFER
7896      ;*THIS ROUTINE WILL CLEAR THE BUFFER BY
7897      ;*SETTING EACH WORD TO "177400".
7898      ;*CALL
7899      ;*      JSR      RO,@#CLRBUF
7900      ;*      RETURN
7901
7902      CLRBUF: JSR      RO,@#SAVR15    ;SAVE R1-R5
7903      MOV      #177400,R1      ;WORD TO FILL BUFFER WITH
7904      MOV      #BUFFER,R2      ;FIRST ADDRESS OF BUFFER
7905      MOV      #BUFFER+(512.*22.),R3 ;LAST ADDRESS+2 OF BUFFER
7906      1$:      MOV      R1,(R2)+    ;FILL WORDS 1, 9,...249,...5625
7907      MOV      R1,(R2)+    ;FILL WORDS 2,10,...250,...5626
7908      MOV      R1,(R2)+    ;FILL WORDS 3,11,...251....5627

```

```

7909 022436 010122      MOV      R1,(R2)+      ;FILL WORDS 4,12,...252,...5628
7910 022440 010122      MOV      R1,(R2)+      ;FILL WORDS 5,13,...253,...5629
7911 022442 010122      MOV      R1,(R2)+      ;FILL WORDS 6,14,...254,...5630
7912 022444 010122      MOV      R1,(R2)+      ;FILL WORDS 7,15,...255,...5631
7913 022446 010122      MOV      R1,(R2)+      ;FILL WORDS 8,16,...256,...5632
7914 022450 020203      CMP      R2,R3         ;DONE?
7915 022452 103766      BLO      1$           ;NO--BRANCH
7916 022454 004037 034354      JSR      R0,#GETR15    ;RESTORE R1-R5
7917 022460 000200      RTS      R0           ;RETURN FROM CALL

```

```

7918
7919
7920      ;SBTTL CKSCTR - CHECK SECTOR DATA
7921      ;*THIS ROUTINE IS USED TO CHECK THE DATA BUFFER
7922      ;*FOR ADDRESSES 0 THROUGH 21 WITH EACH ADDRESS
7923      ;*BEING STORED IN 256 CONSECUTIVE LOCATIONS
7924      ;*CALL
7925      ;*
7926      ;*      JSR      R0,#CKSCTR
7927      ;*      RETURN
7928 022462 004037 034334      CKSCTR: JSR      R0,#SAVR15   ;SAVE R1-R5
7929 022466 162706 000004      SUB      #4,SP        ;RESERVE TEMP. STORAGE AREA
7930 022472 005001      CLR      R1           ;FIRST SECTOR
7931 022474 012716 040524      MOV      #BUFFER,(SP) ;FIRST ADDRESS OF DATA BUFFER
7932 022500 005066 000002      CLR      2(SP)        ;NO ERRORS
7933 022504 012702 000020      1$:     MOV      #1016,R2 ;LOOP COUNT (16*16=256)
7934 022510 011603      MOV      (SP),R3      ;GET 1ST ADDRESS OF THIS SECTORS DATA
7935 022512
7938      2$:     CMP      R1,(R3)+      ;WORD 1
(1) 022514 001064      BNE      7$           ;BRANCH IF BAD
(1) 022516 020123      CMP      R1,(R3)+      ;WORD 2
(1) 022520 001062      BNE      7$           ;BRANCH IF BAD
(1) 022522 020123      CMP      R1,(R3)+      ;WORD 3
(1) 022524 001060      BNE      7$           ;BRANCH IF BAD
(1) 022526 020123      CMP      R1,(R3)+      ;WORD 4
(1) 022530 001056      BNE      7$           ;BRANCH IF BAD
(1) 022532 020123      CMP      R1,(R3)+      ;WORD 5
(1) 022534 001054      BNE      7$           ;BRANCH IF BAD
(1) 022536 020123      CMP      R1,(R3)+      ;WORD 6
(1) 022540 001052      BNE      7$           ;BRANCH IF BAD
(1) 022542 020123      CMP      R1,(R3)+      ;WORD 7
(1) 022544 001050      BNE      7$           ;BRANCH IF BAD
(1) 022546 020123      CMP      R1,(R3)+      ;WORD 8
(1) 022550 001046      BNE      7$           ;BRANCH IF BAD
(1) 022552 020123      CMP      R1,(R3)+      ;WORD 9
(1) 022554 001044      BNE      7$           ;BRANCH IF BAD
(1) 022556 020123      CMP      R1,(R3)+      ;WORD 10
(1) 022560 001042      BNE      7$           ;BRANCH IF BAD
(1) 022562 020123      CMP      R1,(R3)+      ;WORD 11
(1) 022564 001040      BNE      7$           ;BRANCH IF BAD
(1) 022566 020123      CMP      R1,(R3)+      ;WORD 12
(1) 022570 001036      BNE      7$           ;BRANCH IF BAD
(1) 022572 020123      CMP      R1,(R3)+      ;WORD 13
(1) 022574 001034      BNE      7$           ;BRANCH IF BAD
(1) 022576 020123      CMP      R1,(R3)+      ;WORD 14

```

```

(1) 022600 001032      BNE      7$      ;BRANCH IF BAD
(1) 022602 020123      CMP      R1,(R3)+ ;WORD 15
(1) 022604 001030      BNE      7$      ;BRANCH IF BAD
(1) 022606 020123      CMP      R1,(R3)+ ;WORD 16
(1) 022610 001026      BNE      7$      ;BRANCH IF BAD
7939 022612 005302      DEC      R2      ;FINISHED WITH THIS SECTORS DATA?
7940 022614 001336      BNE      2$      ;NO--BRANCH
7941 022616 062716 001000 3$:  ADD      #D512,(SP) ;YES--FIRST ADDRESS OF NEXT SECTOR
7942 022622 005201      INC      R1      ;MOVE TO NEXT SECTOR
7943 022624 022701 000026  CMP      #D22,R1  ;DONE?
7944 022630 003325      BGT      1$      ;NO--BRANCH
7945 022632 005766 000002 4$:  TST      2(SP)   ;ERROR OCCUR?
7946 022636 001406      BEQ      6$      ;NO--BRANCH
7947 022640 123737 001372 001103  CMPB    #ERR.CT,#SERFLG ;MAX. ERROR OCCURRED?
7948 022646 101002      BHI      6$      ;NO--BRANCH
7949 022650 013700 001234 5$:  MOV      #BYPASS,R0 ;TAKE ERROR EXIT
7950 022654 062706 000004 6$:  ADD      #4,SP    ;FREE TEMP. AREA
7951 022660 004037 034354  JSR      R0,#GETR15 ;RESTORE R1-R5
7952 022664 000200      RTS      R0      ;RETURN FROM CALL
7953 022666 010304      7$:  MOV      R3,R4   ;FORM WORD NUMBER AND
7954 022670 161604      SUB      (SP),R4 ;ADDRESS TO CONTINUE FROM
7955 022672 010405      MOV      R4,R5   ;WORD NUMBER
7956 022674 006204      ASR      R4      ;WORD NUMBER
7957 022676 042705 177740  BIC      #C37,R5
7958 022702 001002      BNE      8$      ;BRANCH IF NOT A MULTIPLE OF 16
7959 022704 012705 000040  MOV      #40,R5  ;SET TO WORD 16
7960 022710 006305      8$:  ASL      R5
7961 022712 062705      ADD      #2$ R5  ;ADDRESS
7962 022716 016337 77776 001126  MOV      -2(R3),#SBO0AT ;SAVE BAD DATA
7963 022724 005766 000002  TST      2(SP)   ;FIRST ERROR?
7964 022730 001015      BNE      10$     ;NO--BRANCH
7965 022732 013737 003532 001252  MOV      #DADPB+12,#CYL.DS ;CYLINDER NUMBER
7966 022740 113737 003531 001256  MOVB    #DADPB+11,#TRK.DS ;TRACK NUMBER
7967 022746 012737 022756 001200  MOV      #9$,#ESCAPE ;:ESCAPE TO 9$ ON ERROR
7968 022754 104021      ERROR   21      ;REPORT THE ERROR
7969 022756 105166 000002 9$:  COMB    2(SP)   ;SET ERROR SWITCH
7970 022762 000404      BR      11$
7971 022764      10$:  MOV      #11$,#ESCAPE ;:ESCAPE TO 11$ ON ERROR
(1) 022764 012737 022774 001200  ERROR   22      ;REPORT THE ERROR
7972 022772 104022      11$:  BIT      #SW09,#SWR ;LOOP ON ERROR?
7973 022774 032737 001000 177570  BNE      5$      ;YES
7974 023002 001322      BIT      #SW01,#SWR ;STOP DATA COMPARE?
7975 023004 032737 000002 177570  BNE      4$      ;YES--BRANCH
7976 023012 001307      CMPB    #ERR.CT,#SERFLG ;MAX. ERRORS?
7977 023014 123737 001372 001103  BLOS    5$      ;YES--BRANCH
7978 023022 101712      BIT      #SW05,#SWR ;REPORT ONLY 1ST ERROR PER SECTOR?
7979 023024 032737 000040 177570  BNE      3$      ;YES--BRANCH
7980 023032 001271      JMP      (R5)
7981 023034 000115
7982
7983 ;*****
7984 ;SBTTL SETBUF - SET BUFFER TO DATA PATTERN
7985 ;*THIS ROUTINE WILL MOVE THE 16 WORDS OF THE
7986 ;*DESIRED PATTERN INTO THE DATA BUFFER.

```

```

7987          :CALL
7988          :*      MOV      #NX,R0      ;PATTERN NUMBER INDEX TO R0
7989          :*      JSR      PC,@#SETBUF
7990
7991 023036 004037 034334  ST TBUF: JSR      R0,@#SAVR15 ;SAVE R1-R5
7992 023042 012701 040524      MOV      #BUFFER,R1 ;FIRST ADDRESS
7993 023046 013702 003524      MOV      @#DTADPB+4,R2 ;WORD COUNT
7994 023052 016003 002400      S:      MOV      PAT.PT(R0),R3 ;PICKUP PATTERN POINTER
7997 023055 012321      MOV      (R3)+,(R1)+ ;MOVE WORD 1 INTO DATA BUFFER
(1) 023060 012321      MOV      (R3)+,(R1)+ ;MOVE WORD 2 INTO DATA BUFFER
(1) 023062 012321      MOV      (R3)+,(R1)+ ;MOVE WORD 3 INTO DATA BUFFER
(1) 023064 012321      MOV      (R3)+,(R1)+ ;MOVE WORD 4 INTO DATA BUFFER
(1) 023066 012321      MOV      (R3)+,(R1)+ ;MOVE WORD 5 INTO DATA BUFFER
(1) 023070 012321      MOV      (R3)+,(R1)+ ;MOVE WORD 6 INTO DATA BUFFER
(1) 023072 012321      MOV      (R3)+,(R1)+ ;MOVE WORD 7 INTO DATA BUFFER
(1) 023074 012321      MOV      (R3)+,(R1)+ ;MOVE WORD 8 INTO DATA BUFFER
(1) 023076 012321      MOV      (R3)+,(R1)+ ;MOVE WORD 9 INTO DATA BUFFER
(1) 023100 012321      MOV      (R3)+,(R1)+ ;MOVE WORD 10 INTO DATA BUFFER
(1) 023102 012321      MOV      (R3)+,(R1)+ ;MOVE WORD 11 INTO DATA BUFFER
(1) 023104 012321      MOV      (R3)+,(R1)+ ;MOVE WORD 12 INTO DATA BUFFER
(1) 023106 012321      MOV      (R3)+,(R1)+ ;MOVE WORD 13 INTO DATA BUFFER
(1) 023110 012321      MOV      (R3)+,(R1)+ ;MOVE WORD 14 INTO DATA BUFFER
(1) 023112 012321      MOV      (R3)+,(R1)+ ;MOVE WORD 15 INTO DATA BUFFER
(1) 023114 012321      MOV      (R3)+,(R1)+ ;MOVE WORD 16 INTO DATA BUFFER
7998 023116 062702 000020      ADD      #D16,R2 ;DONE?
7999 023122 001353      BNE      1$ ;NO--BRANCH
8000 023124 004037 034354      JSR      R0,@#GETR15 ;YES--RESTORE R1-R5
8001 023130 000207      RTS      PC ;RETURN

```

```

*****
:SBTTL DATCMP - DATA COMPARE
:*THIS ROUTINE COMPARES A 16 WORD DATA PATTERN
:*AGAINST THE DATA BUFFER

```

```

8007          :CALL
8008          :*      MOV      #NX,R0      ;PATTERN NUMBER INDEX TO R0
8009          :*      JSR      PC,@#DATCMP
8010          :*      RETURN
8011
8012 023132 004037 034334  DATCMP: JSR      R0,@#SAVR05 ;SAVE R0-R5
8013 023136 012701 040524      MOV      #BUFFER,R1 ;FIRST ADDRESS OF BUFFER
8014 023142 013702 003524      MOV      @#DTADPB+4,R2 ;WORD COUNT
8015 023146 005046      CLR      -(SP) ;NO ERROR
8016 023150 016003 002400      1$:      MOV      PAT.PT(R0),R3 ;PATTERN POINTER
8017 023154      2$:
8020 023154 162321      SUB      (R3)+,(R1)+ ;CHECK WORD 1
(1) 023156 001045      BNE      4$ ;BRANCH IF DIFFERENT
(1) 023160 162321      SUB      (R3)+,(R1)+ ;CHECK WORD 2
(1) 023162 001043      BNE      4$ ;BRANCH IF DIFFERENT
(1) 023164 162321      SUB      (R3)+,(R1)+ ;CHECK WORD 3
(1) 023166 001041      BNE      4$ ;BRANCH IF DIFFERENT
(1) 023170 162321      SUB      (R3)+,(R1)+ ;CHECK WORD 4
(1) 023172 001037      BNE      4$ ;BRANCH IF DIFFERENT
(1) 023174 162321      SUB      (R3)+,(R1)+ ;CHECK WORD 5
(1) 023176 001035      BNE      4$ ;BRANCH IF DIFFERENT

```

(1)	023200	162321		SUB	(R3)+,(R1)+	:CHECK WORD 6
(1)	023202	001033		BNE	4\$:BRANCH IF DIFFERENT
(1)	023204	162321		SUB	(R3)+,(R1)+	:CHECK WORD 7
(1)	023206	001031		BNE	4\$:BRANCH IF DIFFERENT
(1)	023210	162321		SUB	(R3)+,(R1)+	:CHECK WORD 8
(1)	023212	001027		BNE	4\$:BRANCH IF DIFFERENT
(1)	023214	162321		SUB	(R3)+,(R1)+	:CHECK WORD 9
(1)	023216	001025		BNE	4\$:BRANCH IF DIFFERENT
(1)	023220	162321		SUB	(R3)+,(R1)+	:CHECK WORD 10
(1)	023222	001023		BNE	4\$:BRANCH IF DIFFERENT
(1)	023224	162321		SUB	(R3)+,(R1)+	:CHECK WORD 11
(1)	023226	001021		BNE	4\$:BRANCH IF DIFFERENT
(1)	023230	162321		SUB	(R3)+,(R1)+	:CHECK WORD 12
(1)	023232	001017		BNE	4\$:BRANCH IF DIFFERENT
(1)	023234	162321		SUB	(R3)+,(R1)+	:CHECK WORD 13
(1)	023236	001015		BNE	4\$:BRANCH IF DIFFERENT
(1)	023240	162321		SUB	(R3)+,(R1)+	:CHECK WORD 14
(1)	023242	001013		BNE	4\$:BRANCH IF DIFFERENT
(1)	023244	162321		SUB	(R3)+,(R1)+	:CHECK WORD 15
(1)	023246	001011		BNE	4\$:BRANCH IF DIFFERENT
(1)	023250	162321		SUB	(R3)+,(R1)+	:CHECK WORD 16
(1)	023252	001007		BNE	4\$:BRANCH IF DIFFERENT
8021	023254	062702	000020	ADD	#D16,R2	:DONE ?
8022	023260	001333		BNE	1\$:NO--BRANCH
8023	023262	005726		TST	(SP)+	:YES -- CLEAN UP STACK
8024	023264	004037	034360	JSR	RO,2#GETRUS	:RESTORE RO-R5
8025	023270	000207		RTS	PC	
8026	023272	010104		MOV	R1,R4	:FORM THE WORD NUMBER
8027	023274	162704	040524	SUB	#BUFFER,R4	
8028	023300	006204		ASR	R4	:WORD NUMBER
8029	023302	010305		MOV	R3,R5	:FORM ADDRESS TO CONTINUE FROM
8030	023304	166005	002400	SUB	PAT.PT(RO),R5	
8031	023310	006305		ASL	R5	
8032	023312	062705	023154	ADD	#2\$,R5	:ADDRESS
8033	023316	064341		ADD	-(R3),-(R1)	:RECONSTRUCT THE BAD WORD
8034	023320	010137	001122	MOV	R1,2#\$8020R	:SAVE THE ERROR INFORMATION
8035	023324	010337	001120	MOV	R3,2#\$8020R	
8036	023330	012137	001126	MOV	(R1)+,2#\$8020R	
8037	023334	012337	001124	MOV	(R3)+,2#\$8020R	
8038	023340	005716		TST	(SP)	:1ST DATA COMPARE ERROR?
8039	023342	001023		BNE	6\$:NO--BRANCH
8040	023344	013737	003532 001252	MOV	2#DTADPB+12,2#CYL.DS	:CYLINDER
8041	023352	113737	003531 001256	MOVB	2#DTADPB+11,2#TRK.DS	:TRACK
8042	023360	113737	003530 001254	MOVB	2#DTADPB+10,2#SEC.DS	:SECTOR
8043	023365	016600	000016	MOV	16(SP),RO	:GET TEST PC+4
8044	023372	012737	023402 001200	MOV	#5\$,SESCAPE	:ESCAPE TO 5\$ ON ERROR
8045	023400	104013		ERROR	13	:REPORT THE ERROR
8046	023402	016600	000014	MOV	14(SP),RO	:PATTERN NUMBER INDEX
8047	023406	105116		COMB	(SP)	:SET THE ERROR SWITCH
8048	023410	000404		BR	7\$	
8049	023412					
(1)	023412	012737	023422 001200	MOV	#7\$,SESCAPE	:ESCAPE TO 7\$ ON ERROR
8050	023420	104014		ERROR	14	:REPORT THE ERROR
8051	023422	032737	000002 177570	BIT	#SW01,2#SWR	:STOP DATA COMPARE?

```

8052 023430 001314          BNE      3$      ;YES--EXIT
8053 023432 123737 001372 001103  CMPB    2$ERR.CT,2$SERFLG ;MAX. ERRORS?
8054 023440 101004          BHI      8$      ;NO--BRANCH
8055 023442 013766 001234 300016  MOV     2$BYPASS,16(SP) ;YES--ERROR EXIT
8056 023450 000704          BR       3$
8057 023452 000115          BS:     JMP      (R5)      ;NO--CONTINUE AT NEXT WORD
8058
8059 ;*****
8060 ;SBTTL FILRAN - FILL DATA BUFFER WITH RANDOM PATTERN
8061 ;*THIS ROUTINE WILL FILL THE DATA BUFFER (256*22 WORDS) WITH
8062 ;*A RANDOM PATTERN. THE FIRST TWO WORDS OF EVERY 256 WILL
8063 ;*BE THE BASE OF THE RANDOM NUMBER GENERATOR FOR THE
8064 ;*NEXT 254 WORDS.
8065 ;*NOTE: THIS ROUTINE DESTROYS R1 AND R2
8066 ;*CALL
8067 ;*      JSR      RO,2$FILRAN
8068 ;*      RETURN
8069
8070 023454 012701 040524  FILRAN: MOV     $BUFFER,R1
8071 023460 012702 000026  MOV     #1022,R2
8072 023464 004037 023674  BS:     JSR      RO,2$RANPAT
8073 023470 005302          DEC     R2
8074 023472 003374          BGT    1$
8075 023474 000200          RTS     RO
8076
8077 ;*****
8078 ;SBTTL RANCK - RANDOM PATTERN BUFFER CHECK
8079 ;*THIS ROUTINE USES THE FIRST TWO WORDS OF THE
8080 ;*READ BUFFER TO GENERATED A RANDOM PATTERN. THEN
8081 ;*THE READ BUFFER IS COMPARED TO THE PATTERN GENERATED.
8082 ;*NOTE: THIS ROUTINE DESTROYS R1-R4
8083 ;*CALL
8084 ;*      JSR      RO,2$RANCK
8085 ;*      RETURN
8086
8087 023476 013746 017312  RANCK: MOV     2$SHINUM,-(SP) ;SAVE THE PRESENT RANDOM NUMBER
8088 023502 013746 017314  MOV     2$SLONUM,-(SP)
8089 023506 012702 041524  MOV     $BUFFER+512.,R2 ;READ BUFFER ADDRESS
8090 023512 012701 042524  MOV     $BUFFER+1024.,R1 ;RANDOM PATTERN ADDRESS
8091 023516 010103          MOV     R1,R3 ;COPY IT INTO R3 FOR LATER USE
8092 023520 011237 017314  MOV     (R2),2$SLONUM ;PRIME THE RANDOM NUMBER GENERATOR
8093 023524 016237 000002 017312  MOV     2(R2),2$SHINUM
8094 023532 004037 023674          JSR     RO,2$RANPAT ;GENERATE A PANDOM PATTERN
8095 023536 012637 017314  MOV     (SP)+,2$SLONUM ;RESTORE PRESENT RANDOM NUMBER
8096 023542 012637 017312  MOV     (SP)+,2$SHINUM
8097 023546 005046          CLR    -(SP) ;NO ERRORS
8098 023550 162322          BS:     SUB     (R3)+,(R2)+ ;ARE THESE TWO WORDS DIFFERENT?
8099 023552 001441          BEQ    4$      ;NO--BRANCH
8100 023554 012737 023626 001200  MOV     3$,$ESCAPE ;ESCAPE TO 3$ ON ERROR
8101 023562 064342          ADD    -(R3),-(R2) ;RECREATE THE BAD WORD
8102 023564 010237 001122  MOV     R2,2$BDDADR ;ADDRESS OF BAD DATA
8103 023570 010337 001120  MOV     R3,2$GDDADR ;ADDRESS OF GOOD DATA
8104 023574 012237 001126  MOV     (R2)+,2$BDDAT ;BAD DATA
8105 023600 012337 001124  MOV     (R3)+,2$GDDAT ;GOOD DATA

```

```

8106 023604 010204      MOV      R2,R4      ;FORM WORD NUMBER (1 TO 256)
8107 023606 162704 041524  SUB      #BUFFER+512.,R4
8108 023612 006204      ASR      R4
8109 023614 005716      TST     (SP)      ;FIRST ERROR
8110 023616 001002      BNE     2$        ;NO--BRANCH
8111 023620 105116      COMB   (SP)      ;YES--SET ERROR SWITCH
8112 023622 104015      ERROR  15       ;REPORT THE ERROR
8113 023624 104016      ERROR  16       ;REPORT THE ERROR
8114 023626 032737 001000 177570 2$:      BIT     #SW09,2#SWR ;LOOP ON ERROR?
8115 023634 001012      BNE     5$        ;YES--BRANCH
8116 023636 123737 001372 001103  CMPB   2#ERR.CT,2#SERFLG ;MAX. ERRORS OCCURRED?
8117 023644 101406      BLOS   5$        ;YES--BRANCH
8118 023646 032737 000002 177570  BIT     #SW01,2#SWR ;STOP COMPARING?
8119 023654 001002      BNE     5$        ;YES--BRANCH
8120 023656 020103      4$:     CMP      R1,R3 ;ALL DATA BEEN COMPARED?
8121 023660 101333      BHI     1$        ;NO--BRANCH
8122 023662 005726      5$:     TST     (SP)+ ;ERROR OCCUR?
8123 023664 001402      BEQ    6$        ;NO--BRANCH
8124 023666 013700 001234  MOV     2#BYPASS,R0 ;TAKE ERROR EXIT
8125 023672 000200      6$:     RTS     R0   ;EXIT

```

```

8126
8127 ;*****
8128 .SBTTL RANPAT - RANDOM PATTERN GENERATOR
8129 ;*THIS ROUTINE FILLS A 256 WORD BUFFER WITH A RANDOM
8130 ;*PATTERN OF WHICH THE FIRST TWO WORDS ARE THE BASE
8131 ;*OF THE PATTERN.
8132 ;*CALL

```

```

8133 ;*      MOV      #ADR,R1      ;ADDRESS OF THE BUFFER
8134 ;*      JSR     R0,2#RANPAT
8135 ;*      RETURN

```

```

8136
8137 RANPAT: MOV     R2,-(SP)      ;SAVE R2
8138 023674 010246 000200  MOV     #10256/2,R2      ;GENERATE 256 WORDS
8139 023702 000402      BR     2$
8140 023704 004737 017166 1$:     JSR     PC,2#SRAND      ;GENERATE A RANDOM NUMBER
8141 023710 013721 017314 2$:     MOV     2#$LONUM,(R1)+ ;PUT LOW WORD IN BUFFER
8142 023714 013721 017312  MOV     2#$HINUM,(R1)+ ;PUT HIGH WORD IN BUFFER
8143 023720 005302      DEC     R2        ;DONE?
8144 023722 003370      BGT    1$        ;NO--BRANCH
8145 023724 012602      MOV     (SP)+,R2    ;RESTORE R2
8146 023726 000200      RTS     R0        ;EXIT

```

```

8147
8148 ;*****
8149 .SBTTL RANADR - RANDOM ADDRESS GENERATOR
8150 ;*THIS ROUTINE GENERATES RANDOM CYLINDER, TRACK, AND SECTOR
8151 ;*ADDRESSES AND SAVES THEM IN THE DPB (DTADPB+10 AND DTADPB+12).
8152 ;*NOTE: THIS ROUTINE DESTROYS R1-R3
8153 ;*CALL

```

```

8154 ;*      JSR     R0,2#RANADR
8155 ;*      RETURN
8156
8157 RANADR: JSR     PC,2#SRAND      ;GENERATE A RANDOM NUMBER
8158 023730 004737 017166  MOVB   2#$LONUM,R1    ;FORM SECTOR IN R1
8159 023734 113701 017314  MOVB   2#$LONUM+1,R2  ;FORM TRACK IN R2

```

```

8160 023744 013703 017312      MOV      2#SHINUM,R3      ;FORM CYLINDER IN R3
8161 023750 105701              TSTB     R1              ;INSURE THE SECTOR IS BETWEEN 0 AND 21
8162 023752 002403              BLT      2$
8163 023754 122701 000026      1$:     CMPB     8#D22,R1
8164 023760 003003              BGT      3$
8165 023762 000241              2$:     CLC
8166 023764 106001              RORB     R1
8167 023766 000772              BR       1$
8168 023770 105702              3$:     TSTB     R2              ;INSURE THE TRACK IS BETWEEN 0 AND 18
8169 023772 002403              BLT      5$
8170 023774 122702 000023      4$:     CMPB     8#D19,R2
8171 024000 003003              BGT      6$
8172 024002 000241              5$:     CLC
8173 024004 106002              RORB     R2
8174 024006 000772              BR       4$
8175 024010 023703 002324      6$:     CMP      2#FC16,R3      ;INSURE THE CYLINDER IS BETWEEN FC AND LC
8176 024014 003413              BLE     7$
8177 024016 000241              CLC
8178 024020 006003              ROR      R3
8179 024022 005503              ADC      R3
8180 024024 001371              BNE     6$
8181 024026 010103              MOV     R1,R3
8182 024030 000303              SWAB    R3
8183 024032 060203              ADD     R2,R3
8184 024034 005203              INC     R3
8185 024036 003364              BGT     6$
8186 024040 005403              NEG     R3
8187 024042 000762              BR      6$
8188 024044 023703 002326      7$:     CMP      2#LC16,R3
8189 024050 002003              BGE     8$
8190 024052 000241              CLC
8191 024054 006003              ROR     R3
8192 024056 000772              BR      7$
8193 024060 023703 002324      8$:     CMP      2#FC16,R3
8194 024064 003403              BLE     9$
8195 024066 005203              INC     R3
8196 024070 000303              SWAB    R3
8197 024072 000764              BR      7$
8198 024074 110137 003530      9$:     MOVB    R1,2#DTADPB+10 ;SAVE SECTOR ADDRESS
8199 024100 110237 003531      MOVB    R2,2#DTADPB+11 ;SAVE TRACK ADDRESS
8200 024104 010337 003532      MOV     R3,2#DTADPB+12 ;SAVE CYLINDER ADDRESS
8201 024110 000200              RTS     R0              ;RETURN

```

```

8202
8203 ;*****
8204 .SBTTL GETSWR - GET THE CONTROL SWITCH SETTINGS
8205 ;*THIS ROUTINE IS USED TO INPUT THE "CONTROL SWITCHES".
8206 ;*IF SWR<07>=1 THE PRESENT SETTING WILL BE TYPED AND THE NEW
8207 ;*SETTING IS READ AND STORED.
8208 ;*NOTE: THIS ROUTINE DESTROYS R3 AND R4
8209 ;*CALL
8210 ;* JSR PC,2#GETSWR
8211 ;* RETURN ;(C.SWR)=DESIRED CONTROL SWITCHES
8212
8213 024112 032737 000200 177570 GETSWR: BIT 8#SW07,2#SWR ;READ CONTROL SWITCHES?

```



```

8214 024120 001426 BEQ 25 ;NO--BRANCH
8215 024122 104400 024130 TYPE ;TYPE ASCIZ STRING
      (1) 024126 000410 BR 645 ;GET OVER THE ASCIZ
      (1) ;.ASCIZ (15)<(12)/SET SWR(07)=0/
      (1) 024150 645:
8216 024150 012703 034436 15: MOV #MSG.CS,R3 ;"CONTROL SWITCHES="
8217 024154 013704 001212 MOV #C.SWR,R4 ;PRESENT CONTROL SWITCH SETTINGS
8218 024160 004037 024420 JSR RO,#GETNUM ;GET THE NEW SWITCH SETTINGS
8219 024164 000771 BR 15 ;COMMA
8220 024166 000770 BR 15 ;PERIOD
8221 024170 010437 001212 MOV R4,#C.SWR ;DOUBLE PERIOD-SAVE NEW SWITCH SETTING
8222 024174 000746 BR GETSWR ;LOOP
8223 024176 000207 25: RTS PC ;RETURN FROM CALL
8224
8225 ;*****
8226 .SBTTL GETADR - GET BUS ADDRESS AND VECTOR ADDRESS
8227 ;*THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS
8228 ;*OF THE RH11/RPO4 IS SETUP TO READ THE PROPER VALUE.
8229 ;*IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
8230 ;*REQUIRED.
8231 ;*NOTE: THIS ROUTINE DESTROYS RO-R4
8232 ;*CALL
8233 ;*
8234 ;* JSR PC,#GETADR
8235 ;* RETURN
8236
8237 024200 005737 001216 GETADR: TST #BUSADR ;INPUT FROM TTY REQUESTED?
8238 024204 001447 BEQ 75 ;NO--BRANCH
8239 024206 005037 001216 CLR #BUSADR ;YES--CLEAR THE REQUEST FLAG
8240 024212 012700 001326 15: MOV #RH.ADR,RO ;FIRST ADDRESS
8241 024216 012703 034462 MOV #MRHCS1,R3 ;"RHCS1="
8242 024222 011004 MOV (RO),R4 ;PRESENT RHCS1 ADDRESS
8243 024224 004037 024420 JSR RO,#GETNUM ;GET NEW RHCS1
8244 024230 000402 BR 25 ;COMMA
8245 024232 000767 BR 15 ;PERIOD
8246 024234 000430 BR 55 ;DOUBLE PERIOD
8247 024236 010420 25: MOV R4,(RO)+ ;SAVE NEW RHCS1
8248 024240 012703 034473 MOV #MRHVEC,R3 ;"RHVEC="
8249 024244 011004 MOV (RO),R4 ;PRESENT RH11 VECTOR ADDRESS
8250 024246 004037 024420 JSR RO,#GETNUM ;GET NEW RHVEC
8251 024252 000402 BR 35 ;COMMA
8252 024254 000756 BR 15 ;PERIOD
8253 024256 000417 BR 55 ;DOUBLE PERIOD
8254 024260 010420 35: MOV R4,(RO)+ ;SAVE NEW RHVEC
8255 024262 012703 034504 MOV #MRHPRI,R3 ;"RHPRI="
8256 024266 011004 MOV (RO),R4 ;PRESENT RH11 PRIORITY LEVEL
8257 024270 006304 ASL R4 ;POSITION FOR TYPEOUT
8258 024272 006304 ASL R4
8259 024274 006304 ASL R4
8260 024276 000304 SWAB R4
8261 024300 004037 024420 JSR RO,#GETNUM ;GET NEW RHPRIO
8262 024304 000402 BR 45 ;COMMA
8263 024306 000401 BR 45 ;PERIOD
8264 024310 000404 BR 65 ;DOUBLE PERIOD

```

```

8265 024312 010210      4$:  MOV    R2,(R0)      ;SAVE NEW RHPRIO
8266 024314 000736      BR     1$              ;LOOP
8267 024316 010410      5$:  MOV    R4,(R0)      ;SAVE INPUT
8268 024320 000401      BR     7$
8269 024322 010210      6$:  MOV    R2,(R0)      ;SAVE PRIORITY
8270 024324 013701 000004 7$:  MOV    2#ERRVEC,R1    ;SAVE THE ERROR VECTOR
8271 024330 012737 024366 000004  MOV    8$ ,2#ERRVEC    ;SETUP FOR TRAP
8272 024336 005777 154764      TST    2#RH.ADR        ;CHECK FOR RH11/RP04
8273 024342 010137 000004      MOV    R1,2#ERRVEC    ;RESTORE ERROR VECTOR
8274 024346 012700 001326      MOV    2#RH.ADR,R0    ;FIRST ADDRESS OF NEW PARAMETERS
8275 024352 012701 027166      MOV    2#RPADR,R1     ;FIRST ADDRESS OF WHERE TO PUT THEM
8276 024356 012021      MOV    (R0)+,(R1)+    ;BUS ADDRESS
8277 024360 012021      MOV    (R0)+,(R1)+    ;VECTOR ADDRESS
8278 024362 012021      MOV    (R0)+,(R1)+    ;PRIORITY LEVEL
8279 024364 000207      RTS    PC              ;RETURN
8280 024366 010137 000004  8$:  MOV    R1,2#ERRVEC    ;RESTORE ERROR VECTOR
8281 024372 022626      CMP    (SP)+,(SP)+    ;CLEAN OFF THE STACK
8282 024374 104010      ERROR 10              ;REPORT THE ERROR
8283 024376 005737 000042      TST    2#42           ;IS THERE A MONITOR?
8284 024402 001703      BEQ    1$              ;NO--GO ASK FOR ADDRESS
8285 024404 005037 001222      CLR    2#DRVSEL       ;YES--NO DRIVES SELECTED
8286 024410 005037 013652      CLR    2#SEOPCT       ;NO PASSES
8287 024414 000137 013476      JMP    2#SEOP         ;GO TO END OF PROGRAM

```

```

8288
8289 ;*****
8290 .SBTTL GETNUM - ROUTINE TO GET A NUMBER
8291 ;*THIS ROUTINE WILL TYPE AN ASCIZ MESSAGE AND THEN
8292 ;*INPUT AN ASCIZ STRING AND CHANGE THE STRING TO OCTAL
8293 ;*IF REQUIRED.
8294 ;*NOTE: THIS ROUTINE DESTROYS R1
8295 ;*CALL
8296 ;*
8297 ;*   MOV    #ADR,R3      ;ADDRESS OF ASCIZ MESSAGE
8298 ;*   MOV    #NUM,R4      ;OCTAL NUMBER
8299 ;*   JSR    R0,2#GETNUM
8300 ;*   RETURN1              ;INPUT TERMINATED WITH A COMMA
8301 ;*   RETURN2              ;WITH A PERIOD
8302 ;*   RETURN3              ;WITH A DOUBLE PERIOD
8303 ;*                       ;R4=INPUT NUMBER AND
8304 ;*                       ;R2=R4*32 FOR ALL
8305 ;*                       ;THREE RETURNS

```

```

8306 024420 010337 024426  GETNUM: MOV    R3,2$      ;SAVE MESSAGE POINTER
8307 024424 104400 1$:  TYPE              ;TYPE THE MESSAGE
8308 024426 000000 2$:  .WORD 0           ;MESSAGE POINTER GOES HERE
8309 024430 010446      MOV    R4,-(SP)      ;SAVE R4 FOR TYPEOUT
(1) 024432 104402      TYPOC              ;GO TYPE--OCTAL ASCII(ALL DIGITS)
8310 024434 104414      RDLIN              ;READ AN ASCIZ STRING
8311 024436 012601      MOV    (SP)+,R1     ;ADDRESS OF FIRST CHARACTER
8312 024440 004037 026454  JSR    R0,2#CK.CHR  ;CHECK ONE CHARACTER
(1) 024444 024424      1$                ;ILLEGAL CHARACTER
(1) 024446 024424      1$                ;CARRIAGE RETURN
(1) 024450 024460      3$                ;"/"
(1) 024452 024504      7$                ;" "
(1) 024454 024512      8$                ;"'"

```

```

(1) 024456 024424
8313 024460 004037 026710 3$: 1$ ;DIGIT 0-9
(1) 024460 004037 026710 JSR RO, @#CK.NUM ;CHECK THE NUMBER
(1) 024464 024424 1$ ;ILLEGAL INPUT
(1) 024466 024500 6$ ;TERMINATED WITH A "."
(1) 024470 024476 5$ ;TERMINATED WITH A ","
(1) 024472 024474 4$ ;TERMINATED WITH A "..."
8314 024474 005720 4$: TST (R0)+ ;DOUBLE PERIOD
8315 024476 005720 5$: TST (R0)+ ;SINGLE PERIOD
8316 024500 010204 6$: MOV R2, R4 ;COMMA--SAVE INPUT NUMBER
8317 024502 000414 BR 10$ ;GO TO EXIT
8318 024504 105711 7$: TSTB (R1) ;TERMINATOR AFTER A COMMA?
8319 024506 001346 BNE 1$ ;NO--LOOP
8320 024510 000411 BR 10$ ;YES--EXIT
8321 024512 105711 8$: TSTB (R1) ;TERMINATOR AFTER A PERIOD?
8322 024514 001406 BEQ 9$ ;YES--EXIT
8323 024516 122721 000056 CMPB #'.,(R1)+ ;NO--DOUBLE PERIOD?
8324 024522 001340 BNE 1$ ;NO--LOOP
8325 024524 105711 TSTB (R1) ;YES--TERMINATOR?
8326 024526 001336 BNE 1$ ;NO--LOOP
8327 024530 005720 TST (R0)+ ;DOUBLE PERIOD
8328 024532 005720 9$: TST (R0)+ ;PERIOD
8329 024534 010402 10$: MOV R4, R2 ;COMMA--POSITION THE
8330 024536 000302 SWAB R2 ;NUMBER IN CASE IT
8331 024540 006202 ASR R2 ;IS THE PRIORITY LEVEL
8332 024542 006202 ASR R2
8333 024544 006202 ASR R2
8334 024546 000200 RTS R0 ;EXIT

```

```

8335
8336 ;*****
8337 ;SBTTL GT.PRM - GET PARAMETERS
8338 ;*THIS ROUTINE IS USED TO CHANGE OR MODIFY
8339 ;*THE TEST PARAMETERS. IT GIVES THE OPERATOR
8340 ;*THE CAPABILITY OF SPECIFYING WHICH DRIVES TO TEST, WHICH
8341 ;*TESTS TO RUN AND HOW MANY TIMES TO
8342 ;*REPEAT EACH TEST
8343 024550 004037 034334 GT.PRM: JSR RO, @#SAVROS ;SAVE RO-R5
8344 024554 005037 001222 GT.PRI: CLR DRVSEL ;NO DRIVE SELECTED
8345 024560 104400 024566 TYPE +4 ;TYPE ASCIZ STRING
(1) 024564 000406 BR 64$ ;GET OVER THE ASCIZ
(1) ;;.ASCIZ <15><12>/DRIVE(S)=/
(1) 64$:
8346 024602 104414 RDLIN ;READ TTY
8347 024604 012601 MOV (SP)+, R1 ;ADDRESS OF ASCIZ STRING
8348 024606 004037 026454 JSR RO, @#CK.CHR ;CHECK ONE CHARACTER
(1) 024612 024554 GT.PRI ;ILLEGAL CHARACTER
(1) 024614 024554 GT.PRI ;CARRIAGE RETURN
(1) 024616 024554 GT.PRI ;"/"
(1) 024620 024554 GT.PRI ;"."
(1) 024622 024554 GT.PRI ;"'"
(1) 024624 024626 1$ ;DIGIT 0-9
8349 024626 005301 1$: DEC R1
8350 024630 012702 000007 2$: MOV #7, R2 ;UPPER LIMIT OF INPUT
(1) 024630 012702 000007

```

```

(1) 024634 004037 026530 JSR RO, @CK.DIG ;CHECK THE DIGIT(S)
(1) 024640 024554 GT.PRI ;ILLEGAL INPUT
(1) 024642 024554 GT.PRI ;INPUT TO LARGE
(1) 024644 024652 3$ ;TERMINATED WITH A "."
(1) 024646 024672 4$ ;TERMINATED WITH A "."
(1) 024650 024672 4$ ;TERMINATED WITH A "."
8351 024652 156237 027154 001222 3$: BISB ATABIT(R2), DRVSEL ;SET THE DRIVE SELECTED BIT
8352 024660 105741 TSTB -(R1) ;WAS THE LINE TERMINATED?
8353 024662 001362 BNE 2$ ;NO-GET THE NEXT DRIVE
8354 024664 005037 001224 CLR @TSTNMS ;DESELECT ALL TESTS
8355 024670 000406 BR GTTST1 ;YES--SELECT TEST
8356 024672 156237 027154 001222 4$: BISB ATABIT(R2), DRVSEL ;SET THE SELECTED DRIVE BITS
8357 024700 004037 034360 GT.PR2: JSR RO, @GETROS ;RESTORE RO-R5
8358 024704 000207 RTS PC ;EXIT
8359
8360 024706 GTTST1:
(1) 024706 104400 024714 TYPE +4 ;:TYPE ASCIZ STRING
(1) 024712 000403 BR 64$ ;:GET OVER THE ASCIZ
(1) ;:ASCIZ /TEST=/
(1) 024722 64$:
8361 024722 104414 RDLIN ;READ AN ASCIZ STRING
8362 024724 012601 MOV (SP)+, R1 ;POINTER TO R1
8363 024726 122711 000056 CMPB #'., (R1) ;DOUBLE PERIOD?
8364 024732 001007 BNE 1$ ;NO--BRANCH
8365 024734 122761 000056 000001 CMPB #'., 1(R1)
8366 024742 001003 BNE 1$
8367 024744 105761 000002 TSTB 2(R1) ;"CR"?
8368 024750 001753 BEQ GT.PR2 ;YES--EXIT
8369 024752 005037 001224 1$: CLR TSTNMS ;NO TEST SELECTED
8370 024756 005003 CLR R3 ;NO TEST TO BE OPENED
8371 024760 005004 GTTST2: CLR R4 ;NO TEST BITS SET
8372 024762 121127 000123 CMPB (R1), #'S ;ALL SEEK TESTS?
8373 024766 001003 BNE 1$ ;NO--BRANCH
8374 024770 012704 000177 MOV #177, R4 ;YES--SELECT TESTS 0-6
8375 024774 000512 BR GTTST3
8376 024776 121127 000124 1$: CMPB (R1), #'T ;ALL TIMING TESTS?
8377 025002 001003 BNE 2$ ;NO--BRANCH
8378 025004 012704 003600 MOV #3600, R4 ;YES--SELECT TESTS 7-12
8379 025010 000504 BR GTTST3
8380 025012 121127 000101 2$: CMPB (R1), #'A ;ALL ADDRESSING TESTS?
8381 025016 001003 BNE 3$ ;NO--BRANCH
8382 025020 012704 014000 MOV #14000, R4 ;YES--SELECT TESTS 13 & 14
8383 025024 000476 BR GTTST3
8384 025026 121127 000104 3$: CMPB (R1), #'D ;DATA TEST?
8385 025032 001003 BNE 4$ ;NO--BRANCH
8386 025034 012704 020000 MOV #20000, R4 ;YES--SELECT TEST 15
8387 025040 000470 BR GTTST3
8388 025042 121127 000105 4$: CMPB (R1), #'E ;EXERCISER TEST?
8389 025046 001003 BNE 5$ ;NO--BRANCH
8390 025050 012704 040000 MOV #40000, R4 ;YES--SELECT TEST 16
8391 025054 000462 BR GTTST3
8392 025056 004037 026400 5$: JSR RO, @CK.OCT ;OCTAL DIGIT?
8393 025062 000460 BR GTTST4 ;NO--BRANCH
8394 025064 010205 MOV R2, R5 ;YES--SAVE IT

```

8395	025066	005201		INC	R1	:MOVE TO NEXT CHARACTER
8396	025070	004037	026400	JSR	R0, @#CK.OCT	:OCTAL DIGIT
8397	025074	000405		BR	6\$:NO--BRANCH
8398	025076	005201		INC	R1	:MOVE TO NEXT CHARACTER
8399	025100	006305		ASL	R5	:SCALE HIGH DIGIT
8400	025102	006305		ASL	R5	
8401	025104	006305		ASL	R5	
8402	025106	060502		ADD	R5, R2	:COMBINE HIGH & LOW DIGITS
8403	025110	020227	000016	6\$: CMP	R2, #16	:LEGAL TEST NUMBER?
8404	025114	003274		BGT	GTTST1	:NO--BRANCH
8405	025116	006302		ASL	R2	
8406	025120	016204	001374	MOV	BITS(R2), R4	:SELECT TEST
8407	025124	121127	000055	CMPB	(R1), #'-	:TEST STRING?
8408	025130	001035		BNE	GTTST4	:NO--BRANCH
8409	025132	005201		INC	R1	:YES--MOVE TO NEXT CHARACTER
8410	025134	004037	026400	JSR	R0, @#CK.OCT	:OCTAL DIGIT?
8411	025140	000662		BR	GTTST1	:NO--BRANCH
8412	025142	010205		MOV	R2, R5	:YES--SAVE IT
8413	025144	005201		INC	R1	:MOVE TO NEXT CHARACTER
8414	025146	004037	026400	JSR	R0, @#CK.OCT	:OCTAL DIGIT?
8415	025152	000405		BR	7\$:NO--BRANCH
8416	025154	005201		INC	R1	:YES--MOVE TO NEXT CHARACTER
8417	025156	006305		ASL	R5	:SCALE HIGH DIGIT
8418	025160	006305		ASL	R5	
8419	025162	006305		ASL	R5	
8420	025164	060502		ADD	R5, R2	:COMBINE HIGH & LOW DIGIT
8421	025166	020227	000016	7\$: CMP	R2, #16	:LEGAL TEST NUMBER?
8422	025172	003245		BGT	GTTST1	:NO--BRANCH
8423	025174	006302		ASL	R2	
8424	025176	020462	001374	CMP	R4, BITS(R2)	:IS THE FIRST NUMBER OF THE
8425						:STRING SMALLER THAN THE LAST?
8426	025202	002241		BGE	GTTST1	:NO--BRANCH
8427	025204	056204	001374	8\$: BIS	BITS(R2), R4	:YES--SET TEST SELECT BIT FOR SELECTED TESTS
8428	025210	005742		TST	-(R2)	:ADJUST POINTER
8429	025212	036204	001374	BIT	BITS(R2), R4	:DONE?
8430	025216	001772		BEQ	8\$:NO--BRANCH
8431	025220	000401		BR	GTTST4	:SKIP THE INCREMENT
8432	025222	005201		GTTST3: INC	R1	:MOVE TO NEXT CHARACTER
8433	025224	050437	001224	GTTST4: BIS	R4, TSTNMS	:SET SELECTED TEST BITS INTO
8434						:TEST SELECT WORD
8435	025230	121127	000056	CMPB	(R1), #'.	: "PERIOD"?
8436	025234	001420		BEQ	GTTST5	:YES--BRANCH
8437	025236	005704		TST	R4	:ANY TEST SELECTED THIS CYCLE?
8438	025240	001622		BEQ	GTTST1	:NO--BRANCH
8439	025242	121127	000057	CMPB	(R1), #' /	: "OPEN"?
8440	025246	001002		BNE	1\$:NO--BRANCH
8441	025250	050403		BIS	R4, R3	:YES--SET BITS FOR TEST TO OPEN
8442	025252	000403		BR	2\$	
8443	025254	121127	000054	1\$: CMPB	(R1), #' ,	: "COMMA"?
8444	025260	001212		BNE	GTTST1	:NO--BRANCH
8445	025262	005201		2\$: INC	R1	:MOVE TO NEXT CHARACTER
8446	025264	105711		TSTB	(R1)	: "CR"?
8447	025266	001234		BNE	GTTST2	:NO--GO GET NEXT CHARACTER
8448	025270	005703		TST	R3	:ANY TESTS TO OPEN?

```

8449 025272 001026      BNE      OPNTST      ;YES--BRANCH
8450 025274 000604      BR       GTTST1     ;NO--START AGAIN
8451 025276 005201      GTTST5: INC      R1      ;MOVE TO NEXT CHARACTER
8452 025300 121127 000056      CMPB    (R1),#'.'   ;"PERIOD"?
8453 025304 001410      BEQ     GTTST6     ;YES--BRANCH
8454 025306 105711      TSTB   (R1)       ;"CR"?
8455 025310 001402      BEQ     1$        ;YES--BRANCH
8456 025312 000137 024706      JMP     GTTST1
8457 025316 005703      1$:    TST      R3      ;ANY TEST TO BE OPENED?
8458 025320 001013      BNE     OPNTST     ;YES--BRANCH
8459 025322 000137 024700      JMP     GT.PR2
8460 025326 005201      GTTST6: INC      R1      ;MOVE TO NEXT CHARACTER
8461 025330 105711      TSTB   (R1)       ;"CR"?
8462 025332 001402      BEQ     1$        ;YES--BRANCH
8463 025334 000137 024706      JMP     GTTST1
8464 025340 005703      1$:    TST      R3      ;ANY TEST TO BE OPENED?
8465 025342 001002      BNE     OPNTST     ;YES--BRANCH
8466 025344 000137 024700      JMP     GT.PR2     ;NO--GO START TESTING
8467      ;OPEN THE TEST FOR CHANGES
8468 025350 004037 034334      OPNTST: JSR     RO,2#SAVROS
8469 025354 005027      CLR     (PC)+
8470 025356 000000      OPN.CT: .WORD   0      ;START WITH TEST 0
8471 025360 000412      3R      OPN.2      ;COUNT STORED HERE
8472 025362 005237 025356      OPN.1: INC     OPN.CT  ;SKIP THE INCREMENT
8473 025366 022737 000016 025356      CMP     #16,OPN.CT  ;MOVE TO THE NEXT TEST
8474 025374 002004      BGE     OPN.2      ;TEST NUMBER TO BIG?
8475 025376 004037 034360      JSR     RO,2#GETROS ;NO--OPEN THE NEXT TEST
8476 025402 000137 024706      JMP     GTTST1
8477 025406 013705 025356      OPN.2: MOV     OPN.CT,R5 ;YES--GO ASK FOR MORE TESTS
8478 025412 006305      ASL     R5        ;SETUP TO USE THE
8479 025414 036503 001374      BIT     BITS(R5),R3 ;TEST NUMBER AS AN INDEX
8480 025420 001760      BEQ     OPN.1     ;OPEN THIS TEST?
8481 025422 104400 025430      TYPE   +4        ;NO--MOVE TO NEXT TEST
8482 025426 000404      BR      64$      ;TYPE ASCIZ STRING
8483      ;;.ASCIZ      / TEST / ;GET OVER THE ASCIZ
8484      64$:
8485 025440 013746 025356      MOV     OPN.CT,-(SP) ;SAVE OPN.CT FOR TYPEOUT
8486 025444 104404      TYPOS   ;TEST NUMBER
8487 025446 002      .BYTE   2        ;GO TYPE--OCTAL ASCII
8488 025447 000      .BYTE   0        ;TYPE 2 DIGIT(S)
8489 025450 104400 001207      TYPE   $CRLF     ;SUPPRESS LEADING ZEROS
8490 025454 016500 001462      MOV     PRMPT(R5),RO ;TYPE "CR" & "LF"
8491 025460 011046      MOV     (RO),-(SP) ;PICKUP PARAMETER POINTER
8492 025462 012702 001434      MOV     #PRM,R2   ;SAVE THE VARIABLE INDICATOR
8493 025466 000405      BR      2$      ;FIRST ADDRESS OF TABLE
8494 025470 006216      1$:    ASR     (SP)   ;CHECK FOR A VARIABLE
8495 025472 103403      BCS     2$      ;GO MOVE THIS ONE
8496 025474 001404      BEQ     OPNPRM   ;DONE
8497 025476 005722      TST     (R2)+    ;BUMP THE POINTER
8498 025500 000773      BR      1$
8499 025502 012022      2$:    MOV     (RO)+,(R2)+ ;MOVE THIS VARIABLE INTO THE
8500 025504 000771      BR      1$      ;COMMON AREA
8501 025506 013716 001434      OPNPRM: MOV     2#PRM,(SP) ;GET THE VARIABLE INDICATOR

```

8496	025512	005004		CLR	R4	;ZERO THE INDEX	
8497	025514	006216		ASR	(SP)	;CHECK FOR A VARIABLE	
8498	025516	103403		BCS	3\$;GO GET IT	
8499	025520	001772		BEQ	OPNPRM	;OUT OF VARIABLES	
8500	025522	005724		TST	(R4)+	;UPDATE THE INDEX	
8501	025524	000773		BR	1\$		
8502	025526	005764	001520	TST	PRMLMT(R4)	;IS THE MAX. MAGNITUDE NEG?	
8503	025532	100463		BMI	OPNPAT	;YES--THEN IT IS THE PATTERN	
8504	025534	104400	035202	TYPE	MSG.SP	;TYPE SPACES	
8505	025540	016437	001544 025550	MOV	PRMSG(R4),4\$;TYPE THE NAME OF THIS VARIABLE	
8506	025546	104400		TYPE			
8507	025550	000000		.WORD	0		
8508	025552	104400	034434	TYPE	MSG.EQ	;TYPE "="	
8509	025556	016446	001436	MOV	RPT(R4),-(SP)	;PUT RPT(R4) ON THE STACK	
(1)	025562	004737	016460	JSR	PC,2\$SSB2D	;CHANGE RPT(R4) TO DECIMAL ASCIZ	
(1)	025566	004737	016704	JSR	PC,2\$SSUPRS	;TYPE WITHOUT LEADING ZEROS	
8510	025572	104414		RDLIN			
8511	025574	012601		MOV	(SP)+,R1	;READ AN ASCIZ STRING	
8512	025576	004037	026454	JSR	RD,2\$CK.CHR	;CHECK ONE CHARACTER	
(1)	025602	025526		3\$;ILLEGAL CHARACTER	
(1)	025604	025526		3\$;CARRIAGE RETURN	
(1)	025606	025652		7\$		"/	
(1)	025610	025616		5\$:"	
(1)	025612	025624		6\$:"	
(1)	025614	025526		3\$;DIGIT 0-9	
8513	025616	105711		TSTB	(R1)	"CR"?	
8514	025620	001342		BNE	3\$;NO--STAY ON THIS VARIABLE	
8515	025622	000737		BR	2\$;YES--MOVE TO NEXT VARIABLE	
8516	025624	105711		TSTB	(R1)	;IS THERE A "CR" AFTER THE PERIOD?	
8517	025626	001002		BNE	64\$;NO	
8518	025630	000137	026240	JMP	OPN.N2	;YES--GO CLOSE THIS TEST	
8519	025634	122721	000056	64\$:	CMPB	8'.,(R1)+	;DOUBLE PERIOD?
8520	025640	001332		BNE	3\$;NO--GO ASK FOR THIS VARIABLE	
8521	025642	105711		TSTB	(R1)	;YES--IS A "CR" AFTER THE DOUBLE PERIOD?	
8522	025644	001330		BNE	3\$;NO--ASK FOR THIS VARIABLE AGAIN	
8523	025646	000137	026256	JMP	OPN.X2	;YES--CLOSE ALL TEST	
8524	025652			7\$:			
(1)	025652	016402	001520	MOV	PRMLMT(R4),R2	;UPPER LIMIT OF INPUT	
(1)	025656	004037	026530	JSR	RD,2\$CK.DIG	;CHECK THE DIGIT(S)	
(1)	025662	025526		3\$;ILLEGAL INPUT	
(1)	025664	025526		3\$;INPUT TO LARGE	
(1)	025666	025674		8\$;TERMINATED WITH A ","	
(1)	025670	026234		OPN.N1		;TERMINATED WITH A "."	
(1)	025672	026252		OPN.X1		;TERMINATED WITH A "..."	
8525	025674	010264	001436	8\$:	MOV	R2,RPT(R4)	;SAVE THIS VARIABLE
8526	025700	000710		BR	2\$;MOVE TO NEXT VARIABLE	
8527	025702	104400	035202	OPNPAT:	TYPE	MSG.SP	;TYPE SPACES
8528	025706	104400	034430	TYPE	MSG.PAT	;TYPE "PAT"	
8529	025712	104400	034434	TYPE	MSG.EQ	;TYPE "="	
8530	025716	016446	001436	MOV	RPT(R4),-(SP)	;SAVE RPT(R4) FOR TYPEOUT	
(1)	025722	104402		TYPOC		;GO TYPE--OCTAL ASCII(ALL DIGITS)	
8531	025724	104414		RDLIN		;READ ASCIZ STRING	
8532	025726	012601		MOV	(SP)+,R1	;PICKUP POINTER	
8533	025730	004037	026454	JSR	RD,2\$CK.CHR	;CHECK ONE CHARACTER	

(1)	025734	025702		OPNPAT		: ILLEGAL CHARACTER
(1)	025736	025702		OPNPAT		: CARRIAGE RETURN
(1)	025740	025770		2\$		"/"
(1)	025742	025506		OPNPRM		","
(1)	025744	025750		1\$		","
(1)	025746	025702		OPNPAT		: DIGIT 0-9
8534	025750	105711	1\$:	TSTB (R1)		: "CR" AFTER THE PERIOD?
8535	025752	001532		BEG OPN.N2		: YES--GO CLOSE THIS TEST
8536	025754	122721	000056	CMPB #' (R1)+		: NO--PERIOD?
8537	025760	001350		BNE OPNPAT		: NO--LOOP
8538	025762	105711		TSTB (R1)		: "CR" AFTER A DOUBLE PERIOD?
8539	025764	001534		BEG OPN.X2		: YES--GO START TESTING
8540	025766	000745		BR OPNPAT		: NO--LOOP
8541	025770		2\$:			
(1)	025770	004037	026710	JSR RO,#CHK.NUM		: CHECK THE NUMBER
(1)	025774	025702		OPNPAT		: ILLEGAL INPUT
(1)	025776	026004		3\$: TERMINATED WITH A ","
(1)	026000	026234		OPN.N1		: TERMINATED WITH A "."
(1)	026002	026252		OPN.X1		: TERMINATED WITH A "..."
8542	026004	010264	001436	3\$:	MOV R2,RPT(R4)	: SAVE THE INPUT NUMBER
8543	026010	006002		ROR R2		: OPEN PATTERN 0?
8544	026012	103235		BCC OPNPRM		: NO--START AT BEGINNING OF PARAMETER TABLE
8545	026014	004037	034334	JSR RO,#SAVROS		: SAVE R1-R5
8546	026020	005000	OPNWD5:	CLR RO		: START WITH WORD C
8547	026022	012704	002440	MOV #PATO,R4		
8548	026026		1\$:			
(1)	026026	104400	026034	TYPE		: TYPE ASCIZ STRING
(1)	026032	000403		BR		: GET OVER THE ASCIZ
(1)				;;.ASCIZ / WD/		
(1)	026042		64\$:			
8549	026042	010046		MOV RO,-(SP)		: PUT RO ON THE STACK
(1)	026044	004737	016460	JSR PC,#SSB20		: CHANGE RO TO DECIMAL ASCIZ
(1)	026050	004737	016704	JSR PC,#SSUPRS		: TYPE WITHOUT LEADING ZEROS
8550	026054	104400	034434	TYPE MSG.EQ		: TYPE "="
8551	026060	011446		MOV (R4),-(SP)		: SAVE (R4) FOR TYPEOUT
(1)	026062	104402		TYPOC		: GO TYPE--OCTAL ASCII(ALL DIGITS)
8552	026064	104414		RDLIN		: READ ASCIZ STRING
8553	026066	012601		MOV (SP)+,R1		: PICKUP THE POINTER
8554	026070	004037	026454	JSR RO,#CHK.CHR		: CHECK ONE CHARACTER
(1)	026074	026026		1\$: ILLEGAL CHARACTER
(1)	026076	026026		1\$: CARRIAGE RETURN
(1)	026100	026110		2\$		"/"
(1)	026102	026126		4\$		","
(1)	026104	026142		5\$		","
(1)	026106	026026		1\$: DIGIT 0-9
8555	026110		2\$:			
(1)	026110	004037	026710	JSR RO,#CHK.NUM		: CHECK THE NUMBER
(1)	026114	026026		1\$: ILLEGAL INPUT
(1)	026116	026124		3\$: TERMINATED WITH A ","
(1)	026120	026162		6\$: TERMINATED WITH A "."
(1)	026122	026176		8\$: TERMINATED WITH A "..."
8556	026124	010214	3\$:	MOV R2,(R4)		: SAVE THE INPUT
8557	026126	005724	4\$:	TST (R4)+		: MOVE TO NEXT WORD
8558	026130	005200		INC RO		: INCREMENT THE COUNT

8553	026132	022700	000020		CMP	#16.,R0	:COUNT TO LARGE?
8550	026136	003333			BGT	1\$:NO--BRANCH
8561	026140	000727			BR	OPN.WDS	:YES--BRANCH
8562	026142	105711		5\$:	TSTB	(R1)	: "CR" AFTER THE PERIOD?
8563	026144	001407			BEQ	7\$:YES--GO CLOSE THIS TEST
8564	026146	122721	000056		CMPB	#',.(R1)+	:NO--PERIOD?
8565	026152	001325			BNE	1\$:NO--BRANCH ILLEGAL INPUT STRING
8566	026154	105711			TSTB	(R1)	: "CR" AFTER THE "PERIOD-PERIOD"?
8567	026156	001410			BEQ	9\$:YES--GO START TESTING
8568	026160	000722			BR	1\$:NO--LOOP
8569	026162	010224		6\$:	MOV	R2,(R4)+	:SAVE THE INPUT
8570	026164	004737	026212	7\$:	JSR	PC,2#CLSWDS	:CLOSE THE DATA PATTERN
8571	026170	004037	034360		JSR	RO,2#GETROS	:GET RO-R5
8572	026174	000421			BR	OPN.N2	:MOVE TO NEXT TEST
8573	026176	010224		9\$:	MOV	R2,(R4)+	:SAVE THE INPUT
8574	026200	004737	026212	9\$:	JSR	PC,2#CLSWDS	:CLOSE THE DATA PATTERN
8575	026204	004037	034360		JSR	RO,2#GETROS	:GET RO-R5
8576	026210	000422			BR	OPN.X2	:START TESTING
8577	026212	012701	002440	CLSWDS:	MOV	#PAT0,R1	:FIRST ADDRESS OF DATA PATTERN
8578	026216	005200		1\$:	INC	RO	:COUNT THE LAST WORD THAT WAS STORED
8579	026220	022700	000017		CMP	#15.,R0	:END OF TABLE
8580	026224	002402			BLT	2\$:YES--EXIT
8581	026226	012124			MOV	(R1)+,(R4)+	:COPY
8582	026230	000772			BR	1\$:LOOP
8583	026232	000207		2\$:	RTS	PC	:RETURN
8584	026234	010264	001436	OPN.N1:	MOV	R2,RPT(R4)	:SAVE THIS VARIABLE
8585	026240	005726		OPN.N2:	TST	(SP)+	:CLEAN OFF THE STACK
8586	026242	004737	026314		JSR	PC,CLOSE	:CLOSE THIS TEST
8587	026246	000137	025362		JMP	OPN.1	:GO OPEN THE NEXT TEST
8588	026252	010264	001436	OPN.X1:	MOV	R2,RPT(R4)	:SAVE THIS VARIABLE
8589	026256	005726		OPN.X2:	TST	(SP)+	:CLEAN OFF THE STACK
8590	026260	004737	026314	1\$:	JSR	PC,CLOSE	:CLOSE THIS TEST
8591	026264	005725		2\$:	TST	(R5)+	:UPDATE THE INDEX
8592	026266	020527	000034		CMP	R5,#16*2	:INDEX TO BIG?
8593	026272	002404			BLT	3\$:NO--BRANCH
8594	026274	004037	034360		JSR	RO,2#GETROS	:YES--RESTORE RO-R5
8595	026300	000137	024700		JMP	GT.PR2	:GO TO EXIT
8596	026304	036503	001374	3\$:	BIT	BITS(R5),R3	:IS THIS TEST OPEN FOR CHANGE?
8597	026310	001363			BNE	1\$:YES--GO CLOSE IT
8598	026312	000764			BR	2\$:NO--MOVE TO NEXT TEST
8599	026314	004037	034334	CLOSE:	JSR	RO,2#SAVROS	:SAVE RO-R5
8600	026320	012700	001434		MOV	#PRM,RO	: "FROM" ADDRESS
8601	026324	016501	001462		MOV	PRMP+(R5),R1	: "TO" ADDRESS
8602	026330	012002			MOV	(RO)+,R2	: "FROM" INDICATOR
8603	026332	012103			MOV	(R1)+,R3	: "TO" INDICATOR
8604	026334	012704	000001		MOV	#1,R4	:TEST BIT START A "RPT"
8605	026340	030402		1\$:	BIT	R4,R2	:PARAMETER TO BE MOVED?
8606	026342	001403			BEQ	2\$:NO--BRANCH
8607	026344	030402			BIT	R4,R3	:A PLACE TO PUT IT?
8608	026346	001404			BEQ	3\$:NO--BRANCH
8609	026350	011011			MOV	(RO),(R1)	:YES--MOVE "FROM" TO "TO"
8610	026352	030402		2\$:	BIT	R4,R3	: "TO" PARAMETER?
8611	026354	001401			BEQ	3\$:NO--BRANCH
8612	026356	005721			TST	(R1)+	:YES--UPDATE THE POINTER

```

8613 026360 005720          35:   TST   (R0)+      ;UPDATE FROM POINTER
8614 026362 006304          ASL   R4          ;POSITION THE TEST BIT
8615 026364 032704 002000   BIT   #BIT10,R4  ;DONE?
8616 026370 001763          BEQ   IS          ;NO--BRANCH
8617 026372 004037 034360   JSR   R0,#GETROS ;YES--RESTORE R0-R5
8618 026376 000207          RTS   PC         ;RETURN
8619                                     ;*****
8620 .SBTTL CK.OCT -- CHECK FOR OCTAL CHARACTER
8621 ;THIS ROUTINE IS USED TO CHECK IF AN
8622 ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
8623 ;CALL
8624 ;   MOV   #ADR,R1      ;ADDRESS OF ASCII CHARACTER
8625 ;   JSR   R0,#CK.OCT  ;CHECK THE CHARACTER
8626 ;   RETURN1          ;CHARACTER IS NOT BETWEEN 0-7
8627 ;   RETURN2          ;CHARACTER IS IN R2 AS A
8628 ;                   ;OCTAL DIGIT
8629
8630 026400 121127 000060   CK.OCT: CMPB   (R1),#'0  ;LESS THAN ZERO?
8631 026404 103407          BLO   IS          ;YES -- BRANCH
8632 026406 121127 000067   CMPB   (R1),#'7     ;GREATER THAN SEVEN?
8633 026412 101004          BHI   IS          ;YES -- BRANCH
8634 026414 111102          MOVB  (R1),R2       ;GET THE CHARACTER
8635 026416 042702 177770   BIC   #7,R2        ;STRIP AWAY THE ASCII
8636 026422 005720          TST   (R0)+        ;ADJUST FOR RETURN
8637 026424 000200          IS:   RTS   R0     ;RETURN
8638
8639                                     ;*****
8640 .SBTTL CK.DEC -- CHECK FOR DECIMAL CHARACTER
8641 ;THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
8642 ;AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
8643 ;CALL
8644 ;   MOV   #ADR,R1      ;ADDRESS OF ASCII CHARACTER
8645 ;   JSR   R0,#CK.DEC  ;CHECK THE CHARACTER
8646 ;   RETURN1          ;NOT BETWEEN 0 AND 9
8647 ;   RETURN2          ;BETWEEN 0 AND 9
8648 ;                   ;R2 = DIGIT
8649
8650 026426 121127 000060   CK.DEC: CMPB   (R1),#'0  ;LESS THAN ZERO?
8651 026432 103407          BLO   IS          ;YES -- BRANCH
8652 026434 121127 000071   CMPB   (R1),#'9     ;GREATER THAN NINE?
8653 026440 101004          BHI   IS          ;YES -- BRANCH
8654 026442 111102          MOVB  (R1),R2       ;GET THE CHARACTER
8655 026444 042702 000060   BIC   #9,R2        ;STRIP AWAY THE ASCII
8656 026450 005720          TST   (R0)+        ;ADJUST FOR RETURN
8657 026452 000200          IS:   RTS   R0     ;RETURN
8658
8659                                     ;*****
8660 .SBTTL CK.CHR -- CHECK CHARACTER
8661 ;THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
8662 ;DETERMINE WHAT IT IS.
8663 ;CALL
8664 ;   MOV   #ADR,R1      ;ADDRESS OF ASCII CHARACTER
8665 ;   JSR   R0,#CK.CHR  ;CHECK CHARACTER
8666 ;   RETURN  ADA1       ;UNKNOWN CHARACTER

```

```

8667          : *      RETURN  ADR2      ; CARRIAGE RETURN * (R1)=ADR+1
8668          : *      RETURN  ADR3      ; SLASH * (R1)=ADR+1
8669          : *      RETURN  ADR4      ; COMMA * (R1)=ADR+1
8670          : *      RETURN  ADR5      ; PERIOD * (R1)=ADR+1
8671          : *      RETURN  ADR6      ; DIGIT BETWEEN 0 AND 9.
8672          : *                                     ; R2 = DIGIT * (R1)=ADR+1
8673
8674 026454 005711 CK.CHR: TSTB (R1) ; "CARRIAGE RETURN"?
8675 026456 001420      BEQ 4$      ; YES -- BRANCH
8676 026460 121127 000057      CMPB (R1),#'/' ; "SLASH"?
8677 026464 001414      BEQ 3$      ; YES -- BRANCH
8678 026466 121127 000054      CMPB (R1),#',' ; "COMMA"?
8679 026472 001410      BEQ 2$      ; YES -- BRANCH
8680 026474 121127 000056      CMPB (R1),#'. ; "PERIOD"?
8681 026500 001404      BEQ 1$      ; YES -- BRANCH
8682 026502 004037 026426      JSR RO,#CK.DEC ; "DIGIT"?
8683 026506 000406      BR 5$      ; NO -- BRANCH
8684 026510 005720      TST (RO)+   ; DIGIT BETWEEN 0-9
8685 026512 005720      1$: TST (RO)+   ; PERIOD
8686 026514 005720      2$: TST (RO)+   ; COMMA
8687 026516 005720      3$: TST (RO)+   ; SLASH
8688 026520 005720      4$: TST (RO)+   ; CARRIAGE RETURN
8689 026522 005201      INC R1      ; MOVE POINTER TO NEXT CHARACTER
8690 026524 011000      5$: MOV (RO),RO ; UNKNOWN CHARACTER
8691 026526 000200      RTS RO      ; RETURN
8692
8693          : *****
8694          : SBTTL CK.DIG - CHECK DIGIT
8695          : *THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
8696          : *CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
8697          : *CALL
8698          : *      MOV #ADR,R1      ; ADDRESS OF ASCII STRING
8699          : *      MOV #NUM,R2      ; MAX. MAGNITUDE OF INPUT NUMBER
8700          : *      JSR RO,#CK.DIG ; CHECK DIGITS
8701          : *      RETURN ADR1      ; ILLEGAL CHARACTER -- R2=?
8702          : *      RETURN ADR2      ; INPUT NUMBER TOO LARGE -- R2=?
8703          : *      RETURN ADR3      ; "COMMA" -- R2 = NUMBER
8704          : *      RETURN ADR4      ; "PERIOD" -- R2 = NUMBER
8705          : *      RETURN ADR5      ; "PERIOD-PERIOD" -- R2 = NUMBER
8706
8707 026530 010446 CK.DIG: MOV R4,-(SP) ; SAVE R4
8708 026532 010346      MOV R3,-(SP) ; SAVE R3
8709 026534 010246      MOV R2,-(SP) ; SAVE THE MAX. SIZE ON THE STACK
8710 026536 005002      CLR R2      ; START WITH 0
8711 026540 005003      CLR R3
8712 026542 005004      CLR R4
8713 026544 004037 026454      JSR RO,#CK.CHR ; CHECK ONE CHARACTER
8714          (1) 026550 026674      BS ; ILLEGAL CHARACTER
8714          (1) 026552 026674      BS ; CARRIAGE RETURN
8714          (1) 026554 026674      BS ; "/"
8714          (1) 026556 026674      BS ; "."
8714          (1) 026560 026674      BS ; " "
8714          (1) 026562 026564      1$: BS ; DIGIT 0-9
8714          (1) 026564 006303      1$: ASL R3 ; *2

```

```

8715 026566 010346      MOV      R3,-(SP)      ;SAVE #2
8716 026570 006303      ASL      R3           ;*4
8717 026572 006303      ASL      R3           ;*8
8718 026574 062603      ADD      (SP)+,R3     ;(*8)+(*2)=*10.
8719 026576 060203      ADD      R2,R3       ;UPDATE THE INPUT NUMBER
8720 026600 004037 026454      JSR      R0,@CK.CHR  ;CHECK ONE CHARACTER
(1) 026604 026674      8$      ;ILLEGAL CHARACTER
(1) 026606 026674      9$      ;CARRIAGE RETURN
(1) 026610 026674      8$      ;"/"
(1) 026612 026622      3$      ;"."
(1) 026614 026620      2$      ;"'"
(1) 026616 026564      1$      ;DIGIT 0-9
8721 026620 005724      2$: TST      (R4)+      ;"PERIOD"
8722 026622 005724      3$: TST      (R4)+      ;"COMMA"
8723 026624 004037 026454      JSR      R0,@CK.CHR  ;CHECK ONE CHARACTER
(1) 026630 026674      8$      ;ILLEGAL CHARACTER
(1) 026632 026664      6$      ;CARRIAGE RETURN
(1) 026634 026674      9$      ;"/"
(1) 026636 026674      9$      ;"."
(1) 026640 026644      4$      ;"'"
(1) 026642 026654      5$      ;DIGIT 0-9
8724 026644 005724      4$: TST      (R4)+      ;"PERIOD-PERIOD"
8725 026646 105711      TSTB     (R1)        ;"CR"
8726 026650 001405      BEQ      6$          ;YES--BRANCH
8727 026652 000410      BR       8$
8728 026654 126127 177776 000054 5$: CMPB     -2(R1),#' , ;WAS CHARACTER BEFORE THE DIGIT A COMMA?
8729 026662 001004      BNE      8$          ;NO--EXIT
8730 026664 020316      6$: CMP      R3,(SP)  ;INPUT TO LARGE?
8731 026666 101001      BHI      7$          ;YES -- BRANCH
8732 026670 060400      ADD      R4,R0       ;ADJUST RETURN ADDRESS
8733 026672 005720      7$: TST      (R0)+      ;NUMBER TO R2
8734 026674 010302      8$: MOV      R3,R2     ;CLEAN MAX. SIZE OFF OF STACK
8735 026676 005726      TST      (SP)+,R3    ;RESTORE R3
8736 026700 012603      MOV      (SP)+,R4    ;RESTORE R4
8737 026702 012604      MOV      (SP)+,R4    ;RESTORE R4
8738 026704 011000      MOV      (R0),R0     ;GET RETURN ADDRESS
8739 026706 000200      RTS      R0          ;RETURN

```

```

;SBTTL CK.NUM - CHECK NUMBER (OCTAL)
;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
;AND FORMS AN OCTAL NUMBER IN R2
;CALL:
;*      MOV      #ADR,R1      ;ADDRESS OF ASCIZ STRING
;*      JSR      R0,@CK.NUM   ;GO FORM THE NUMBER
;*      RETURN   ADR1        ;ILLEGAL CHARACTER IN THE INPUT STRING
;*      RETURN   ADR2        ;"COMMA"--R2=NUMBER
;*      RETURN   ADR3        ;"PERIOD"--R2=NUMBER
;*      RETURN   ADR4        ;"PERIOD-PERIOD"--R2=NUMBER

```

```

8753 026710 010346      CK.NUM: MOV      R3,-(SP) ;SAVE R3
8754 026712 005003      CLR      R3           ;START NUMBER AT ZERO
8755 026714 004037 026400      JSR      R0,@CK.OCT  ;OCTAL DIGIT?
8756 026720 000440      BR       6$          ;NO--BRANCH

```

8757	026722	005201		1\$:	INC	R1	: MOVE TO NEXT CHARACTER
8758	026724	006303			ASL	R3	: FOR THE OCTAL NUMBER IN R3
8759	026726	103435			BCS	6\$: DON'T LET IT GET TO BIG
8760	026730	006303			ASL	R3	
8761	026732	103433			BCS	6\$	
8762	026734	006303			ASL	R3	
8763	026736	103431			BCS	6\$	
8764	026740	060203			ADD	R2, R3	
8765	026742	004037	026400		JSR	RO, @CK.OCT	: IS THIS AN OCTAL DIGIT?
8766	026746	000401			BR	2\$: NO--FIND OUT WHAT IT IS
8767	026750	000764			BR	1\$: YES--MAKE IT PART OF THE NUMBER
8768	026752	010302		2\$:	MOV	R3, R2	: SAVE THE OCTAL NUMBER
8769	026754	005003			CLR	R3	: START WITH ZERO INDEX
8770	026756	004037	026454		JSR	RO, @CK.CHR	: CHECK ONE CHARACTER
(1)	026762	027022			6\$: ILLEGAL CHARACTER
(1)	026764	027022			6\$: CARRIAGE RETURN
(1)	026766	027022			6\$		"/"
(1)	026770	027012			5\$		":"
(1)	026772	026776			3\$		":."
(1)	026774	027022			6\$: DIGIT 0-9
8771	026776	005723		3\$:	TST	(R3)+	"PERIOD"
8772	027000	121127	000056		CMPB	(R1), #'.	"PERIOD-PERIOD"?
8773	027004	001002			BNE	5\$: NO--BRANCH
8774	027006	005201			INC	R1	: YES--ADVANCE THE POINTER
8775	027010	005723		4\$:	TST	(R3)+	"PERIOD-PERIOD"
8776	027012	005723		5\$:	TST	(R3)+	"COMMA"
8777	027014	105711			TSTB	(R1)	"CR"?
8778	027016	001001			BNE	6\$: NO--BRANCH
8779	027020	060300			ADD	R3, RO	: YES--SAVE THE OCTAL NUMBER
8780	027022	012603		6\$:	MOV	(SP)+, R3	: RESTORE R3
8781	027024	011000			MOV	(RO), RO	: PICKUP EXIT ADDRESS
8782	027026	000200			RTS	RO	: RETURN

```

8784 ;*****
8785
(1) .SBTTL RH11/RP04 DRIVER (REV. 0.9)
(1)
(1) ;*COPYRIGHT (C) 1974
(1) ;*DIGITAL EQUIPMENT CORP.
(1) ;*MAYNARD, MA 01754
(1) ;*AUTHOR: JIM LACEY
(1)
(1) ;*STORAGE FOR RH0S1, RHER1, RHER2, AND RHER3 ON AN ERROR "2" OR "5"
(1) ;*RPERRS = RH0S1
(1) ;*RPERRS+2 = RHER1
(1) ;*RPERRS+4 = RHER2
(1) ;*RPERRS+6 = RHER3
(1)
(1) RPERRS: .WORD 0
(1) .WORD 0
(1) .WORD 0
(1) .WORD 0
(1)
(1) ;*TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
(1) ;*DRVACT=0 IMPLIES DRIVE IS IDLE
(1) ;*DRVACT>0 IMPLIES DRIVE IS ACTIVE WITH A COMMAND
(1) ;*DRVACT<0 IMPLIES DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION
(1)
(1) DRVACT: .BYTE 0 ;DRIVE 0
(1) .BYTE 0 ;DRIVE 1
(1) .BYTE 0 ;DRIVE 2
(1) .BYTE 0 ;DRIVE 3
(1) .BYTE 0 ;DRIVE 4
(1) .BYTE 0 ;DRIVE 5
(1) .BYTE 0 ;DRIVE 6
(1) .BYTE 0 ;DRIVE 7
(1)
(1) ;*TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
(1) ;*DRVSTA=0 IMPLIES DRIVE IS OFFLINE
(1) ;*DRVSTA>0 IMPLIES DRIVE IS ONLINE
(1) ;*DRVSTA<0 IMPLIES DRIVE IS UNSAFE OR NONEXISTENT
(1)
(1) DRVSTA: .BYTE 0 ;DRIVE 0
(1) .BYTE 0 ;DRIVE 1
(1) .BYTE 0 ;DRIVE 2
(1) .BYTE 0 ;DRIVE 3
(1) .BYTE 0 ;DRIVE 4
(1) .BYTE 0 ;DRIVE 5
(1) .BYTE 0 ;DRIVE 6
(1) .BYTE 0 ;DRIVE 7
(1)
(1) ;*TABLE OF DRIVE TYPES (DRVTyp=8 WORDS)
(1) ;*DRVTyp WILL CONTAIN THE DRIVE TYPE OF ALL ONLINE, OFFLINE, AND
(1) ;*UNSAFE DRIVES. IF A DRIVE IS NONEXISTENT DRVTyp WILL BE ZERO.
(1)
(1) DRVTyp: .WORD 0 ;DRIVE 0
(1)

```

```

(1) 027062 000000 .WORD 0 ;DRIVE 1
(1) 027064 000000 .WORD 0 ;DRIVE 2
(1) 027066 000000 .WORD 0 ;DRIVE 3
(1) 027070 000000 .WORD 0 ;DRIVE 4
(1) 027072 000000 .WORD 0 ;DRIVE 5
(1) 027074 000000 .WORD 0 ;DRIVE 6
(1) 027076 000000 .WORD 0 ;DRIVE 7
(1)
(1) ;*TRANSFER WAIT FLAG (TRNSWT=1 WORD)
(1) ;*THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
(1) ;*"DPB" OF THE I/O OPERATION.
(1)
(1) 027100 000000 TRNSWT: .WORD 0
(1)
(1) ;*SEARCH WAIT KEYS (SRCHWT=1 WORD)
(1) ;*THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
(1) ;*THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
(1) ;*REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
(1) ;*EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
(1)
(1) 027102 000000 SRCHWT: .WORD 0
(1)
(1) ;*RP04 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
(1) ;*ACTDRV=0 IMPLIES DRIVER IS INACTIVE
(1) ;*ACTDRV>0 IMPLIES DRIVER IS ACTIVE
(1)
(1) 027104 000 ACTDRV: .BYTE 0
(1)
(1) ;*SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
(1) ;*ACTSTR=0 IMPLIES SOFTWARE TIMER ROUTINE IS INACTIVE
(1) ;*ACTSTR>0 IMPLIES SOFTWARE TIMER ROUTINE IS ACTIVE
(1)
(1) 027105 000 ACTSTR: .BYTE 0
(1)
(1) ;*UNLOAD FLAG (ULDFLG=8 BYTES)
(1) ;*ULDFLG=0 IMPLIES NO UNLOAD COMMAND
(1) ;*ULDFLG>0 IMPLIES UNLOAD COMMAND IN PROGRESS
(1) ;*ULDFLG<0 IMPLIES UNLOAD COMMAND IN WAIT QUEUE
(1)
(1) 027106 000 ULDFLG: .BYTE 0 ;DRIVE 0
(1) 027107 000 .BYTE 0 ;DRIVE 1
(1) 027110 000 .BYTE 0 ;DRIVE 2
(1) 027111 000 .BYTE 0 ;DRIVE 3
(1) 027112 000 .BYTE 0 ;DRIVE 4
(1) 027113 000 .BYTE 0 ;DRIVE 5
(1) 027114 000 .BYTE 0 ;DRIVE 6
(1) 027115 000 .BYTE 0 ;DRIVE 7
(1)
(1) ;*LOOK AHEAD COUNT (LACNT=8 BYTES)
(1) ;*LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
(1)
(1) 027116 000 LACNT: .BYTE 0 ;DRIVE 0
(1) 027117 000 .BYTE 0 ;DRIVE 1
(1) 027120 000 .BYTE 0 ;DRIVE 2

```

I11

```
(1) 027121 000 .BYTE 0 ;DRIVE 3
(1) 027122 000 .BYTE 0 ;DRIVE 4
(1) 027123 000 .BYTE 0 ;DRIVE 5
(1) 027124 000 .BYTE 0 ;DRIVE 6
(1) 027125 000 .BYTE 0 ;DRIVE 7
(1)
(1) ;*SAVE REGISTERS FLAG (SAVEFG =1 WORD)
(1) ;*SAVEFG <0 IMPLIES SAVE THE RH11/RP04 REGISTERS WHEN THE
(1) ;*OPERATION IS COMPLETED AS PER (DPB+14).
(1) ;*SAVEFG=0 IMPLIES SAVE THE RH11/RP04 REGISTERS, AS PER
(1) ;*(DPB+14), AFTER AN ERROR.
(1)
(1) 027126 000000 SAVEFG: .WORD 0
(1)
(1) ;*SEEK FLAG (SEEKFG=1 WORD)
(1) ;*SEEKFG=0 IMPLIES WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
(1) ;*FOR A DATA TRANSFER START A SEARCH COMMAND
(1) ;*SEEKFG<0 IMPLIES DATA TRANSFER WILL DO IMPLIED SEEKS,
(1) ;*DISREGARD THE WINDOW
(1)
(1) 027130 000000 SEEKFG: .WORD 0
(1)
(1) ;*TIMEOUT TABLE (TIMER=8 WORDS)
(1) ;*THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
(1)
(1) 027132 177777 TIMER: .WORD -1 ;DRIVE 0
(1) 027134 177777 .WORD -1 ;DRIVE 1
(1) 027136 177777 .WORD -1 ;DRIVE 2
(1) 027140 177777 .WORD -1 ;DRIVE 3
(1) 027142 177777 .WORD -1 ;DRIVE 4
(1) 027144 177777 .WORD -1 ;DRIVE 5
(1) 027146 177777 .WORD -1 ;DRIVE 6
(1) 027150 177777 .WORD -1 ;DRIVE 7
(1)
(1) ;*DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
(1) ;*DTUW<0 IMPLIES NO DATA TRANSFER UNDERWAY
(1) ;*DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
(1)
(1) 027152 177777 DTUW: .WORD -1
(1)
(1) ;*ATTENTION BITS TABLE (ATABIT=8 BYTES)
(1) ;*THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
(1) ;*ATTENTION BIT
(1)
(1) 027154 001 ATABIT: .BYTE 1 ;DRIVE 0
(1) 027155 002 .BYTE 2 ;DRIVE 1
(1) 027156 004 .BYTE 4 ;DRIVE 2
(1) 027157 010 .BYTE 10 ;DRIVE 3
(1) 027160 020 .BYTE 20 ;DRIVE 4
(1) 027161 040 .BYTE 40 ;DRIVE 5
(1) 027162 100 .BYTE 100 ;DRIVE 6
(1) 027163 200 .BYTE 200 ;DRIVE 7
(1)
(1) ;*RP04 TO RH11 "MASS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE
```



```

(1) ;*CALLING IT FATAL (MCPEMX=1 WORD)
(1)
(1) 027164 000003 MCPEMX: .WORD 3
(1)
(1) ;*STORAGE FOR RPADR (THE FIRST ADDRESS (776700) OF THE RH11/RP04),
(1) ;*RPVEC (THE VECTOR ADDRESS (254)), AND RPVEC+2 (THE BR LEVEL (5)).
(1) 027166 176700 RPADR: .WORD 176700
(1) 027170 000254 000240 RPVEC: .WORD 254,5*32.
(1)
(1) ;*MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)
(1) 027174 000004 MXLACT: .WORD 4
(1) ;*MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)
(1) 027176 001000 MXDLTA: .WORD 8*64.
(1) ;*MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)
(1) 027200 000200 MNDLTA: .WORD 2*64.
(1) ;*MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWINDW=1 WORD)
(1) 027202 000005 MXWINDW: .WORD 5
(1)
(1) ;*DEFINITIONS OF THE RH11/RP04 ADDRESS INDEXES
(1)
(1) 000000 RHCS1=0 ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
(1) 000002 RHWC=2 ;WORD COUNT REGISTER (NOT A DRIVE REG)
(1) 000004 RHBA=4 ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
(1) 000006 RHDA=6 ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
(1) 000010 RHCS2=10 ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
(1) 000012 RHCS1=12 ;DRIVE STATUS REGISTER (DRIVE REG 01)
(1) 000014 RHER1=14 ;ERROR REGISTER #1 (DRIVE REG. 02)
(1) 000016 RHAS=16 ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
(1) 000020 RHLA=20 ;LOOK AHEAD REGISTER (DRIVE REG. 07)
(1) 000022 RHOB=22 ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
(1) 000024 RHMR=24 ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
(1) 000026 RHOT=26 ;DRIVE TYPE REGISTER (DRIVE REG. 06)
(1) 000030 RYSN=30 ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
(1) 000032 RHOF=32 ;OFFSET REGISTER (DRIVE REG. 11)
(1) 000034 RHCA=34 ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
(1) 000036 RHCC=36 ;CURRENT CYLINDER ADDRESS REGISTER (DRIVE REG. 13)
(1) 000040 RHER2=40 ;ERROR REGISTER #2 (DRIVE REG. 14)
(1) 000042 RHER3=42 ;ERROR REGISTER #3 (DRIVE REG. 15)
(1) 000044 RHEC1=44 ;ECC POSITION REGISTER (DRIVE REG. 16)
(1) 000046 RHEC2=46 ;ECC PATTERN REGISTER (DRIVE REG. 17)
(1)
(1) ;*RH11/RP04 DRIVER INIT. CODE
(1) ;*THIS ROUTINE WILL DETERMINE WHICH RP04 DRIVES ARE
(1) ;*AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
(1) ;*TO THE PROPER STATE FOR EACH DRIVE.
(1) ;*NOTE: THIS ROUTINE CALLS DRVINT
(1)
(1) ;*CALL
(1)
(1) ;* JSR PC,RPINIT
(1) ;* RETURN
(1)
(1) 027204 013746 177776 RPINIT: MOV @#PS, SP) ;SAVE PROCESSOR STATUS
(1) 027210 013737 027172 177776 MOV RPVEC+2,@#PS ;SET PROCESSOR STATUS TO RH11/RP04 LEVEL

```

```

(1) 027216 004037 034334 JSR RO,SAVR15 ;GO SAVE R1-R5
(1) 027222 004737 034072 JSR PC,CLRQUE ;CLEAR ALL REQUEST QUEUES
(1) 027226 012701 027030 MOV #RPERRS,R1 ;FIRST ADDRESS TO BE CLEARED
(1) 027232 012702 027130 MOV #SEEKFG,R2 ;LAST ADDRESS TO BE CLEARED
(1) 027236 005021 15: CLR (R1)+ ;CLEAR
(1) 027240 020102 CMP R1,R2 ;ARE WE DONE?
(1) 027242 101775 BLOS 15 ;BRANCH IF NO
(1) 027244 012702 027152 MOV #DTUW,R2 ;LAST ADDRESS
(1) 027250 012721 177777 25: MOV #-1,(R1)+ ;INITIALIZE
(1) 027254 020102 CMP R1,R2 ;DONE?
(1) 027256 101774 BLOS 25 ;LOOP IF NO
(1) 027260 005001 CLR R1 ;DRIVE NUMBER WILL BE IN R1
(1) 027262 013704 027166 MOV RPADR,R4 ;FIRST ADDRESS OF RH11/RP04
(1) 027266 012764 000040 000010 MOV #BIT05,RHCS2(R4) ;MASSBUS INIT.
(1) 027274 004037 027346 35: JSR RO,DRVINT ;GO INIT. A DRIVE
(1) 027300 000417 BR 75
(1) 027302 005201 45: INC R1 ;MOVE TO THE NEXT DRIVE
(1) 027304 042701 177770 BIC #1C7,R1 ;DON'T LET THE DRIVE NUMBER GET TO BIG
(1) 027310 001371 BNE 35 ;BRANCH IF MORE DRIVES TO INIT.
(1) 027312 013703 027170 MOV RPVEC,R3 ;SETUP THE RH11/RP04 VECTOR
(1) 027316 012723 031356 MOV #ISR,(R3)+
(1) 027322 013713 027172 MOV RPVEC+2,(R3)
(1) 027326 004037 034354 JSR RO,GETR15 ;RESTORE R1-R5
(1) 027332 012637 177776 MOV (SP)+,#FS ;RESTORE THE PROCESSOR STATUS
(1) 027336 000207 RTS PC ;BYE-BYE
(1) 027340 105061 027050 75: CLRB DRVSTA(R1) ;SET THE DRIVE STATUS TO OFFLINE
(1) 027344 000756 BR 45 ;GO DO THE NEXT DRIVE

; *DRIVE INIT. ROUTINE
; *THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
; *AN RPO4. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT22
; *IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
; *INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
; *DRVSTA IS SET TO THE PROPER CONDITION.

; *CALL
; *
; * MOV #DRVNUM,R1 ;DRIVE NUMBER TO R1
; * MOV RPADR,R4 ;UNIBUS ADDRESS OF RH11/RP04 (RHCS1)
; * JSR RO,DRVINT ;CALLED BY A JSR
; * RETURN1 ;ERROR OCCURRED (PARITY)
; * RETURN2 ;NORMAL RETURN

DRVINT: JSR RO,SAVR15 ;SAVE R1-R5
MOV R1,RHCS2(R4) ;SELECT A DRIVE
MOV R1,R3 ;COPY THE DRIVE NUMBER INTO
ASL R3 ;R3 AND POSITION IT FOR TABLE INDEXING
MOVB #-1,DRVSTA(R1) ;START DRIVE STATUS AS NONEXISTENT
CLR DRVSTYP(R3) ;CLEAR THE DRIVE TYPE INDICATOR
MOVB #111,RPADR ;DO A "DRIVE CLEAR" COMMAND
BIT #BIT12,RHCS2(R4) ;NONEXISTENT DRIVE?
BEQ 15 ;NO---BRANCH
JSR PC,SET.IE ;GO SET "IE" WITHOUT A "TRE"
BR 65 ;LEAVE THIS ROUTINE
15: JSR RO,RD.RP ;READ THE DRIVE TYPE REG.
RHD
  
```

```

(1) 027426 027566          7$          ;ERROR RETURN ADDRESS
(1) 027430 012605          MOV          (SP)+,R5          ;PUT DRIVE TYPE IN R5
(1) 027432 010563 027060  MOV          R5,DRVSTYP(R3) ;SAVE THE DRIVE TYPE
(1) 027436 022705 020020  CMP          #20020,R5       ;IS IT A SINGLE PORT RP04?
(1) 027442 001403          BEQ          2$              ;BRANCH IF YES
(1) 027444 022705 024020  CMP          #24020,R5       ;IS IT A DUAL PORT RP04?
(1) 027450 001043          BNE          5$              ;BRANCH IF NO
(1) 027452 012746 000121 2$: MOV          #121,-(SP)      ;DO A "READ-IN PRESET"
(1) 027456 004037 033272  JSR          RO,WRT.RP
(1) 027462 000000          RHCS1
(1) 027464 027566          7$
(1) 027466 012746 010000  MOV          #BIT12,-(SP)    ;SET FMT22=1
(1) 027472 004037 033272  JSR          RO,WRT.RP
(1) 027476 000032          RHOF
(1) 027500 027566          7$
(1) 027502 004037 033134  JSR          RO,RO.RP        ;READ RHDS1
(1) 027506 000012          RHDS1
(1) 027510 027566          7$
(1) 027512 012605          MOV          (SP)+,R5        ;AND SAVE IT IN R5
(1) 027514 100011          BPL          4$              ;BRANCH IF ATA=0
(1) 027516 116164 027154 000016  MOVB        ATABIT(R1),RHAS(R4) ;CLEAR ATTENTION BIT
(1) 027524 004037 033134  JSR          RO,RO.RP        ;FIND OUT WHY ATA=1
(1) 027530 000014          RHER1
(1) 027532 027566          7$
(1) 027534 006126          ROL          (SP)+          ;IS IT UNSAFE?
(1) 027536 100412          BMI          6$              ;BRANCH IF YES
(1) 027540 005105          COM          R5              ;CHECK MOL, DPF, DRY, AND VV
(1) 027542 042705 167077  BIC          #1<BIT12:BIT08:BIT07:BIT06>,R5
(1) 027546 001004          BNE          5$              ;BRANCH IF MOL, DPF, DRY, OR VV IS CLEAR
(1) 027550 112761 000001 027050  MOVB        #1,DRVSTA(R1)    ;SET DRIVE STATUS TO ONLINE
(1) 027556 000402          BR          6$
(1) 027560 105061 027050  5$: CLRB        DRVSTA(R1)    ;DRIVE STATUS = OFFLINE
(1) 027564 005720          6$: TST          (RO)+        ;STEP OVER THE ERROR RETURN
(1) 027566 004037 034354  7$: JSR          RO,GETR15     ;RESTORE R1-R5
(1) 027572 000200          RTS          RO              ;RETURN

(1)
(1) ;*REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
(1)
(1) ;*CALL
(1)
(1) ;* JSR          RO,#RP04    ;CALL THE RP04 DRIVER
(1) ;* PNTADR        ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
(1) ;* RETURN1       ;RETURN HERE IF QUEUE IS FULL
(1) ;* RETURN2       ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
(1) ;*              ;IS AN ERROR CONDITION
(1)
(1) RP04: MOV          2#PS,-(SP) ;SAVE THE CALLING STATUS
(1) 027574 013746 177776  MOV          RPVEC+2,2#PS    ;DON'T ALLOW ANY RP04 INTERRUPTS
(1) 027600 013737 027172 177776  MOVB        #1,ACTDRV       ;SET "ACTIVE DRIVER" FLAG
(1) 027606 112737 000001 027104  JSR          RO,SAVR15      ;SAVE R1-R5
(1) 027614 004037 034334  MOV          (RO)+,R2        ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
(1) 027620 012002          CLR          16(R2)         ;CLEAR THE STATUS/ERROR INDICATOR
(1) 027622 005062 000016  MOVB        (R2),R1         ;PICKUP THE DRIVE NUMBER
(1) 027626 111201          MOVB        DRVSTA(R1)     ;CHECK DRIVES STATUS
(1) 027630 105761 027050  TST

```

```

(1) 027634 003017          BGT      1$          ;BRANCH IF ONLINE
(1) 027636 105761 027106  TSTB    ULDFLG(R1)  ;UNLOAD COMMAND IN QUEUE?
(1) 027642 001034          BNE     3$          ;BRANCH IF YES
(1) 027644 013704 027166  MOV     RPAOR,R4   ;UNIBUS ADDRESS OF RHCS1
(1) 027650 004037 027346  JSR    RO,DRVINT  ;GO INIT. THE DRIVE
(1) 027654 000442          BR      6$          ;ERROR RETURN
(1) 027656 105761 027050  TSTB    DRVSTA(R1) ;IS DRIVE STATUS ONLINE?
(1) 027662 003004          BGT     1$          ;BRANCH IF YES
(1) 027664 052762 140000 000016  BIS    #BIT15:BIT14,16(R2) ;ERROR--DRIVE IS OFFLINE OR NONEXISTENT
(1) 027672 000423          BR      4$          ;GO TO EXIT
(1) 027674 004037 034174          1$:    JSR    RO,DRVQUE  ;PUT THIS REQUEST IN QUEUE
(1) 027700 000421          BR      5$          ;QUEUE IS FULL
(1) 027702 122762 000103 000002  CMPB   #103,2(R2)  ;IS THIS REQ. FOR AN UNLOAD?
(1) 027710 001003          BNE     2$          ;BR IF NO
(1) 027712 112761 177777 027106  MOVB   #-1,ULDFLG(R1) ;SET THE "UNLOAD IN QUEUE" FLAG
(1) 027720 105761 027040          2$:    TSTB    DRVACT(R1) ;IS THIS DRIVE ACTIVE?
(1) 027724 001006          BNE     4$          ;BR IS YES
(1) 027726 004737 027770  JSR    PC,OPT     ;CALL THE OPTIMIZER
(1) 027732 000403          BR      4$
(1) 027734 052762 120000 000016  3$:    BIS    #BIT15:BIT13,16(R2) ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
(1) 027742 005720          4$:    TST    (R0)+
(1) 027744 004037 034354          5$:    JSR    RO,GETR15 ;RESTORE R1-R5
(1) 027750 105037 027104          CLRB   ACTDRV     ;CLEAR "ACTIVE DRIVER" FLAG
(1) 027754 012637 177776          MOV    (SP)+,2#PS ;RETURN "PS" TO USER LEVEL
(1) 027760 000200          RTS    RO         ;RETURN TO CALLER
(1) 027762 004737 030772          6$:    JSR    PC,C17   ;GO HANDLE THE PARITY ERROR
(1) 027766 000765          BR      4$

(1)
(1) ;*OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
(1)
(1) ;*CALL
(1) ;*
(1) ;*   MOV    #DRVNUM,R1 ;DRIVE NUMBER TO R1
(1) ;*   JSR    PC,OPT     ;SETUP A COMMAND
(1)
(1) OPT:  JSR    RO,SAVR15 ;SAVE R1-R5
(1) 027774 013746 177776          MOV    2#PS,-(SP) ;SAVE PROC. STATUS
(1) 030000 146137 027154 027102  BICB   ATABIT(R1),SRCHWT ;CLEAR "SEARCH WAIT" KEY
(1) 030006 004737 034250          JSR    PC,GETREQ  ;GET "DPB" POINTER OF REQUEST
(1) 030012 005702          TST    R2         ;IS THERE A REQUEST IN QUEUE?
(1) 030014 001444          BEQ    6$          ;NO--BRANCH TO EXIT
(1) 030016 105761 027050  TSTB    DRVSTA(R1) ;IS DRIVE ONLINE?
(1) 030022 003006          BGT     1$          ;YES--BRANCH
(1) 030024 004737 034272          JSR    PC,POPQUE  ;NO--REMOVE REQUEST FROM QUEUE
(1) 030030 052762 140000 000016  BIS    #BIT15:BIT14,16(R2) ;SET ERROR BIT OF STATUS/ERROR INDICATOR
(1) 030036 000426          BR      5$          ;BRANCH TO EXIT
(1) 030040 122762 000150 000002  1$:    CMPB   #150,2(R2) ;IS THE REQUEST FOR I/O?
(1) 030046 002403          BLT    2$          ;YES--BRANCH
(1) 030050 004737 030350          JSR    PC,C14     ;CALL THE COMMAND INITIATOR
(1) 030054 000417          BR      5$          ;BRANCH TO EXIT
(1) 030056 005737 027152          2$:    TST    DTUW     ;DATA TRANSFER UNDERWAY?
(1) 030062 002012          BGE    4$          ;YES--GO START A SEARCH
(1) 030064 005737 027130          TST    SEEKFG    ;DO IMPLIED SEEKS?
(1) 030070 100404          BMI    3$          ;YES---BRANCH
(1) 030072 004037 031220          JSR    RO,LA     ;NO--DO LOOK AHEAD

```

```

(1) 030076 000406 BR 5$ ;RETURN HERE ON A PARITY ERROR
(1) 030100 000403 BR 4$ ;GO START A SEARCH
(1) 030102 004737 030134 3$: JSR PC,C11 ;START A DATA TRANSFER
(1) 030106 000402 BR 5$
(1) 030110 004737 030242 4$: JSR PC,C13 ;START A SEARCH
(1) 030114 012637 177776 5$: MOV (SP)+,2*PS ;RESTORE PROC. STATUS
(1) 030120 004037 034354 JSR R0,GETR15 ;RESTORE R1-R5
(1) 030124 000207 RTS PC
(1) 030126 004737 033526 6$: JSR PC,SET.IE ;SET "IE" WITHOUT A "TRE"
(1) 030132 000770 BR 5$ ;EXIT THE ROUTINE
(1)
(1) ;*COMMAND INITIATOR
(1)
(1) ;*CALL
(1) ;* MOV #DRVNUM,R1 ;DRIVE NUMBER
(1) ;* MOV #DPB,R2 ;ADDRESS OF DPB
(1) ;* JSR PC,C1? ;C1?= C11,C13, OR C14
(1) ;* ;WHERE:
(1) ;* ;C11=DATA TRANSFER
(1) ;* ;C12=SEARCH REQUESTED BY DATA XFER
(1) ;* ;C14=NOT DATA TRANSFER
(1)
(1) 030134 004737 034272 C11: JSR PC,POPQUE ;REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
(1) 030140 010237 027100 MOV R2,TRNSW ;PUT REQ. IN TRANSFER WAIT QUEUE
(1) 030144 010203 C12: MOV R2,R3 ;DPB ADDRESS TO R3
(1) 030146 013704 027166 MOV RPADR,R4 ;RHCS1 ADDRESS
(1) 030152 010164 000010 MOV R1,RHCS2(R4) ;SELECT DRIVE
(1) 030156 062703 000004 ADD #4,R3 ;DESIRED WORD COUNT
(1) 030162 062704 000002 ADD #2,R4 ;RHC ADDRESS
(1) 030166 012324 MOV (R3)+,(R4)+ ;LOAD WORD COUNT
(1) 030170 012324 MOV (R3)+,(R4)+ ;LOAD BUFFER ADDRESS
(1) 030172 012346 MOV (R3)+,-(SP) ;LOAD SECTOR AND TRACK
(1) 030174 004037 033272 JSR R0,WRT.RP ;CALL THE LOAD(WRITE) ROUTINE
(1) 030200 000006 R1,DA ;INDEX OF REGISTER TO LOAD
(1) 030202 030772 C17 ;ERROR RETURN ADDRESS
(1) 030204 012346 MOV (R3)+,-(SP) ;LOAD CYLINDER ADDRESS
(1) 030206 004037 033272 JSR R0,WRT.RP
(1) 030212 000034 RHCA
(1) 030214 030772 C17
(1) 030216 016246 000002 MOV 2(R2),-(SP) ;LOAD "COMMAND+GO", "A17&A16". AND "PSEL"
(1) 030222 004037 033272 JSR R0,WRT.RP
(1) 030226 000000 RHCS1
(1) 030230 030772 C17
(1) 030232 010137 027152 MOV R1,DTUW ;SET "DATA TRANSFER UNDERWAY"
(1) 030236 000137 030726 C15
(1) 030242 013704 027166 C13: MOV RPADR,R4 ;RHCS1 ADDRESS
(1) 030246 010164 000010 MOV R1,RHCS2(R4) ;SELECT DRIVE
(1) 030252 016246 000012 MOV 12(R2),-(SP) ;DESIRED CYLINDER ADDRESS
(1) 030256 004037 033272 JSR R0,WRT.RP
(1) 030262 000034 RHCA
(1) 030264 030772 C17
(1) 030266 116203 000010 MOV# 10(R2),R3 ;PICKUP SECTOR ADDRESS
(1) 030272 163703 027202 SUB MXWINDW,R3 ;BACKUP BY MAX. SEARCH FOR I/O WINDOW
(1) 030276 002002 BGE 1$

```

(1)	030300	062703	000026		ADD	#22,R3	
(1)	030304	010346		1\$:	MOV	R3,-(SP)	:COMBINE THE ADJUSTED SECTOR WITH
(1)	030306	116266	000011	000001	MOV	11(R2),1,SP	:THE DESIRED TRACK
(1)	030314	004037	033272		JSR	RD,WRT.RP	:LOAD DESIRED TRACK & SECTOR
(1)	030320	000006			RHDA		
(1)	030322	030772			CI7		
(1)	030324	012746	000131		MOV	#131,-(SP)	:START A SEARCH
(1)	030330	004037	033272		JSR	RD,WRT.RP	
(1)	030334	000000			RHCS1		
(1)	030336	030772			CI7		
(1)	030340	156137	027154	027102	B15B	ATABIT(R1),SACHWT	:SET "SEARCH WAIT" KEY
(1)	030346	000557			BR	CI5	
(1)	030350	013704	027166		MOV	RPADR,R4	:RHCS1 ADDRESS
(1)	030354	010164	000010	CI4:	MOV	R1,RHCS2(R4)	:SELECT DRIVE
(1)	030360	116203	000002		MOV	2(R2),R3	:PICKUP THE REQUESTED COMMAND
(1)	030364	122703	000131		CMPB	#131,R3	:IS IT A SEARCH COMMAND?
(1)	030370	001007			BNE	1\$:BRANCH IF NO
(1)	030372	016246	000010		MOV	10(R2),-(SP)	:LOAD DESIRED TRACK & SECTOR
(1)	030376	004037	033272		JSR	RD,WRT.RP	
(1)	030402	000006			RHDA		
(1)	030404	030772			CI7		
(1)	030406	000403			BR	2\$:GO LOAD CYLINDER
(1)	030410	122703	000105	1\$:	CMPB	#105,R3	:IS IT A SEEK COMMAND
(1)	030414	001007			BNE	3\$:BRANCH IF NO
(1)	030416	016246	000012	2\$:	MOV	12(R2),-(SP)	:LOAD DESIRED CYLINDER
(1)	030422	004037	033272		JSR	RD,WRT.RP	
(1)	030426	000034			RHCA		
(1)	030430	030772			CI7		
(1)	030432	000546			BR	CI5	
(1)	030434	122703	000115	3\$:	CMPB	#115,R3	:IS IT AN "OFFSET" COMMAND?
(1)	030440	001013			BNE	4\$:BR IF NO
(1)	030442	004037	033134		JSR	RD,RD.RP	:MERGE THE OFFSET VALUE INTO RHOF
(1)	030446	000032			RHOF		:BUT DON'T CHANGE THE UPPER
(1)	030450	030772			CI7		
(1)	030452	116216	000001		MOV	11(R2),(SP)	:BYTE WHEN LOADING THE
(1)	030456	004037	033272		JSR	RD,WRT.RP	:REGISTER (RHOF)
(1)	030462	000032			RHOF		
(1)	030464	030772			CI7		
(1)	030466	000530			BR	CI5	:GO START THE COMMAND
(1)	030470	122703	000107	4\$:	CMPB	#107,R3	:IS IT A "RECALIBRATE" COMMAND?
(1)	030474	001525			BEQ	CI5	:BRANCH IF YES
(1)	030476	122703	000117		CMPB	#117,R3	:IS IT A RETURN TO CENTER?
(1)	030502	001522			BEQ	CI5	:BRANCH IF YES
(1)	030504	122703	000103		CMPB	#103,R3	:IS IT AN "UNLOAD" COMMAND?
(1)	030510	001016			BNE	5\$:BRANCH IF NO
(1)	030512	112761	000001	02704C	MOV	#1,DRVACT(R1)	:SET THE DRIVE ACTIVE INDICATOR
(1)	030520	105061	027050		CLRB	DRVSTA(R1)	:PUT DRIVE STATUS TO OFFLINE
(1)	030524	112761	000001	027106	MOV	#,ULDFLG(R1)	:SET "UNLOAD IN PROGRESS" FLAG
(1)	030532	010346			MOV	R3,-(SP)	:START THE "UNLOAD" COMMAND
(1)	030534	004037	033272		JSR	RD,WRT.RP	
(1)	030540	000000			RHCS1		
(1)	030542	030772			CI7		
(1)	030544	000207			RTS	PC	:RETURN TO USER
(1)	030546	122703	000143	5\$:	CMPB	#143,R3	:IS IT A "SET FORMAT" COMMAND?

```

(1) 030552 001014      BNE      6$          ;BRANCH IF NO
(1) 030554 004037 033134 JSR      RD, RD.RP   ;READ THE OFFSET REGISTER
(1) 030560 000032      RHOF
(1) 030562 030772      CI7
(1) 030564 116266 000001 000001 MOVB     1(R2), 1(SP) ;COMBINE "FMT22" "ECI" AND "HCI"
(1) 030572 004037 033272 JSR      RD, WRT.RP ;LOAD "FMT22", "ECI", AND/OR "HCI".
(1) 030576 000032      RHOF
(1) 030600 030772      CI7
(1) 030602 000436      BR
(1) 030604 122703 000141 6$: CMPB     #141, R3    ;IS IT A "GET REGISTER" COMMAND?
(1) 030610 001023      BNE
(1) 030612 016203 000006 7$: MOV      6(R2), R3   ;BRANCH IF NO
;POINTS TO 1ST ADDRESS OF WHERE
;TO PUT THE REGISTER(S)
(1) 030616 116237 000010 030634 MOVB     10(R2), 9$  ;INIT. THE INDEX FOR THE FIRST REG.
(1) 030624 116205 000011      MOVB     11(R2), R5  ;INDEX OF LAST REG. TO MOVE
(1) 030630 004037 033134 8$: JSR      RD, RD.RP   ;READ RP04 REGISTER
(1) 030634 000000 9$: RHCSI
;INDEX OF REG. TO READ
(1) 030636 030772      CI7
(1) 030640 012623      MOV      (SP)+, (R3)+ ;GET THE CONTENTS OF RH11/RP04 REG.
(1) 030642 023705 030634 CMP      9$, R5      ;LAST REG. BEEN READ?
(1) 030646 001414      BEQ      12$         ;GET OUT IF YES
(1) 030650 062737 000002 030634 ADD      #2, 9$      ;INCREASE THE INDEX BY 2
(1) 030656 000764      BR      8$          ;LOOP--MORE TO READ
(1) 030660 122703 000145 10$: CMPB    #145, R3   ;IS IT A "SELECT DRIVE" COMMAND?
(1) 030664 001405      BEQ      12$         ;BRANCH IF YES
(1) 030666 010346 11$: MOV      R3, -(SP) ;LOAD THE COMMAND
(1) 030670 004037 033272 JSR      RD, WRT.RP
(1) 030674 000000      RHCSI
(1) 030676 030772      CI7
(1) 030700 004737 034272 12$: JSR      PC, POPQUE  ;REMOVE REQ. FROM QUEUE
(1) 030704 052762 000200 000016 BIS      #BIT07, 16(R2) ;SET THE "DONE" BIT
(1) 030712 005737 027126 TST      SAVEFG     ;SAVE THE RH11/RP04 REGISTERS?
(1) 030716 100002      BPL      13$         ;BRANCH IF NO
(1) 030720 004737 033434 JSR      PC, SVRH11 ;YES--GO SAVE THE REGISTERS
(1) 030724 000207 13$: RTS      PC        ;RETURN TO USER
(1) 030726 006301 15$: ASL      R1
(1) 030730 012761 001750 027132 MOV      #1000., TIMER(R1) ;SET A ONE SECOND TIMER
(1) 030736 006201      ASR      R1
(1) 030740 112761 000001 027040 MOVB     #1, DRVACT(R1) ;SET THE DRIVE ACTIVE
(1) 030746 000207      RTS      PC        ;RETURN TO THE USER
(1) 030750 010346 16$: MOV      R3, -(SP) ;LOAD THE COMMAND
(1) 030752 004037 033272 JSR      RD, WRT.RP
(1) 030756 000000      RHCSI
(1) 030760 030772      CI7
(1) 030762 000761      BR      C15
(1) 030764 105761 027040 C17A: TSTB     DRVACT(R1) ;IS THE DRIVE ACTIVE?
(1) 030770 001405      BEQ      C17B       ;BRANCH IF NO
(1) 030772 012762 104000 000016 C17: MOV      #BIT15!BIT11, 16(R2) ;SET "PARITY" ERROR INDICATOR
(1) 031000 004737 033434 JSR      PC, SVRH11 ;GO SAVE THE RH11/RP04 REGISTERS
(1) 031004 012746 000111 C17B: MOV      #111, -(SP) ;DO A "DRIVE CLEAR"
(1) 031010 004037 033272 JSR      RD, WRT.RP
(1) 031014 000000      RHCSI
(1) 031016 031056      CI8
(1) 031020 004737 034154 JSR      PC, EMPTYQ  ;EMPTY THE QUEUE

```



```

(1) 031024 105061 027106 CLR8 ULDFLG(R1) ;CLEAR THE UNLOAD IN QUEUE FLAG
(1) 031030 105061 027040 CLR8 DRVACT(R1) ;DRIVE IS IDLE
(1) 031034 020137 027152 CMP R1,DTUM ;IF THIS DRIVE HAD AN I/O REQUEST
(1) 031040 001005 BNE IS ;IN PROGRESS CLEAR ALL OF THE FLAGS
(1) 031042 005037 027100 CLR TRNSWT
(1) 031046 012737 177777 027152 MOV #1,DTUM
(1) 031054 000207 1S: RTS PC
(1) 031056 004037 034334 C18: JSR RO,SAVR15 ;SAVE R1-R5
(1) 031062 013704 027166 MOV RPADR,R4 ;PICKUP THE ADDRESS OF THE FIRST REGISTER
(1) 031066 005001 CLR R1
(1) 031070 005003 CLR R3
(1) 031072 105761 027040 1S: TSTB DRVACT(R1) ;DRIVE ACTIVE?
(1) 031076 001421 BEQ 3S ;BRANCH IF NO
(1) 031100 013702 027100 MOV TRNSWT,R2 ;GET THE "TRANSFER WAIT" QUEUE
(1) 031104 020137 027152 CMP R1,DTUM ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
(1) 031110 001402 BEQ 2S ;BRANCH IF YES
(1) 031112 004737 034250 JSR PC,GETREQ ;GET THE DPB POINTER
(1) 031116 004737 033434 2S: JSR PC,SVRH11 ;SAVE RH11/RP04 REGISTERS
(1) 031122 052762 102000 C00C16 BIS #BIT15:BIT10,16(R2) ;SET "NON-CLEARABLE PARITY" ERROR INDICATOR
(1) 031130 012763 177777 027132 MOV #1,TIMER(R3) ;STOP THE TIMER
(1) 031136 105061 027040 CLR8 DRVACT(R1) ;SET "DRIVE ACTIVE" TO IDLE
(1) 031142 105061 027106 3S: CLR8 ULDFLG(R1) ;CLEAR UNLOAD FLAG
(1) 031146 005201 INC R1 ;MOVE TO THE NEXT DRIVE
(1) 031150 062703 000002 ADD #2,R3
(1) 031154 042701 177770 BIC #1C7,R1
(1) 031160 001344 BNE IS ;BRANCH IF MORE DRIVES
(1) 031162 012737 177777 027152 MOV #1,DTUM ;NO DATA TRANSFERS UNDERWAY
(1) 031170 005037 027100 CLR TRNSWT ;CLEAR THE "TRANSFER WAIT" QUEUE
(1) 031174 004737 034072 JSR PC,CLRQUE ;CLEAR ALL OF THE REQUEST QUEUES
(1) 031200 012764 000040 C00010 MOV #BIT05,RHCS2(R4) ;DO A MASSBUS INIT.
(1) 031206 004737 033526 JSR PC,SET.IE ;SET "IE" WITHOUT "TRE"
(1) 031212 004037 034354 JSR RO,GETR15 ;RESTORE THE REGISTERS
(1) 031216 000207 RTS PC ;RETURN

(1) ;#LOOK AHEAD ROUTINE
(1) ;#CALL
(1) ;#
(1) ;# MOV #DRVNUM,R1 ;DRIVE NUMBER
(1) ;# MOV #DPB,R2 ;POINT TO DPB
(1) ;# JSR RO,LA ;GO CHECK THE WINDOW
(1) ;# RETURN1 ;ERROR RETURN
(1) ;# RETURN2 ;START A SEARCH
(1) ;# RETURN3 ;START A DATA TRANSFER

(1) 031220 013704 027166 LA: MOV RPADR,R4 ;GET RHCS1'S ADDRESS
(1) 031224 010164 000010 MOV R1,RHCS2(R4) ;SELECT DRIVE
(1) 031230 004037 033134 JSR RO,RO.RP ;READ CURRENT CYLINDER
(1) 031234 000036 RHCC
(1) 031236 031350 4S ;ERROR RETURN ADDRESS
(1) 031240 022662 000012 CMP (SP)+,12(R2) ;IS CURRENT CYLINDER=DESIRED
(1) ;CYLINDER?
(1) 031244 001037 BNE 3S ;EXIT IF NO
(1) 031246 105261 027116 INCB LACNT(R1) ;INCREMENT THE LOOK AHEAD COUNT
(1) 031252 126137 027116 027174 CMPB LACNT(R1),MXLACT ;EXCEED MAX?

```


(1)	031740	106303			ASLB	R3	
(1)	031742	001373			BNE	SC4	: BRANCH IF MORE TO CHECK?
(1)	031744	005726			TST	(SP)+	: CLEAN OFF THE STACK
(1)	031746	000207			RTS	PC	: RETURN TO USER
(1)	031750	023701	027152		CMP	DTUW,R1	: IS THIS DRIVE SETUP FOR I/O?
(1)	031754	001002			BNE	1\$: NO---BRANCH
(1)	031756	005726			TST	(SP)+	: YES---CLEAN OFF THE STACK (RHAS)
(1)	031760	000622			BR	TD	: BRANCH TO "TRANSFER DONE"
(1)	031762	105761	027050		TSTB	DRVSTA(R1)	: CHECK THE DRIVE STATUS
(1)	031766	003030			BGT	SC6	: BRANCH IF ONLINE
(1)	031770	105761	027106		TSTB	ULDFLG(R1)	: UNLOAD IN PROGRESS?
(1)	031774	003420			BLE	2\$: BRANCH IF NO
(1)	031776	004737	034250		JSR	PC,GETREQ	: GET DPB POINTER
(1)	032002	004737	033434		JSR	PC,SVRH11	: SAVE THE RH11/RP04 REGISTERS
(1)	032006	004737	032464		JSR	PC,SC12	: SAVE RH0S1, RHER1, RHER2, AND RHER3
(1)							: ALSO DO A DRIVE INIT (DRVINT)
(1)	032012	105761	027050		TSTB	DRVSTA(R1)	: DID DRIVE COME ONLINE?
(1)	032016	003411			BLE	3\$: NO---BRANCH
(1)	032020	032737	040000	027030	BIT	#BIT14,RPERRS	: WAS THERE AN ERROR?
(1)	032026	001565			BEQ	SC11	: NO -- BRANCH
(1)	032030	013705	027032		MOV	RPERRS+2,R5	: YES -- PICKUP RHER1 AND
(1)	032034	000447			BR	SC6A	: GO PROCESS THE ERROR
(1)	032036	004737	032464		JSR	PC,SC12	: SAVE RH0S1, RHER1, RHER2, AND RHER3
(1)							: ALSO DO A DRVINT
(1)	032042	011605			MOV	(SP),R5	: PICKUP (RHAS) BEFORE THE ERROR CALL
(2)	032044	104005			ERROR	5	: REPORT THE ERROR
(1)	032046	000733			BR	SC3	: GO CHECK FOR MORE ATA'S
(1)	032050	006301			ASL	R1	
(1)	032052	012761	177777	027132	MOV	#-1,TIMER(R1)	: STOP THE TIMER
(1)	032060	006201			ASR	R1	
(1)	032062	004737	034250		JSR	PC,GETREQ	: GET THE DPB POINTER FROM THE QUEUE
(1)	032066	010364	000016		MOV	R3,RHAS(R4)	: CLEAR ATTENTION
(1)	032072	010164	000010		MOV	R1,RHCS2(R4)	: SELECT DRIVE
(1)	032076	004037	033134		JSR	RD,RD.RP	: READ THE RP04'S STATUS REG.
(1)	032102	000012			RH0S1		
(1)	032104	032322			SC8		
(1)	032106	011605			MOV	(SP),R5	: AND PUT IT IN R5
(1)	032110	006126			ROL	(SP)+	: WAS THERE AN ERROR?
(1)	032112	100115			BPL	SC9	: BR IF NO
(1)	032114	105761	027040		TSTB	DRVACT(R1)	: CHECK THE DRIVE ACTIVE INDICATOR
(1)	032120	001522			BEQ	SC10	: BRANCH IF IDLE
(1)	032122	004037	033134		JSR	RD,RD.RP	: READ ERROR REGISTER #1
(1)	032126	000014			RHER1		
(1)	032130	032322			SC8		
(1)	032132	012605			MOV	(SP)+,R5	: AND SAVE IT IN R5
(1)	032134	004737	033434		JSR	PC,SVRH11	: SAVE RH11/RP04 REGISTERS
(1)	032140	012746	000111		MOV	#111,-(SP)	: ISSUE A DRIVE CLEAR
(1)	032144	004037	033272		JSR	RD,WAT.RP	
(1)	032150	000000			RHCS1		
(1)	032152	032322			SC8		
(1)	032154	006105			ROL	R5	: WAS "UNSAFE" CONDITION =1?
(1)	032156	100404			B'I	1\$: BRANCH IF YES
(1)	032160	052762	100240	000016	BIS	#BIT15:BIT07:BIT05,16(R2)	: INFORM USER OF ERROR
(1)	032166	000443			BR	SC7	

```

(1) 032170 004037 033134 1$: JSR RO, RD.RP ; READ DRIVE STATUS REG. #1
(1) 032174 000012 RVD0S1
(1) 032176 032322 SC2
(1) 032200 011605 MOV (SP), R5 ; SAVE RHDS1 IN R5
(1) 032202 006126 ROL (SP)+ ; "ERR"=1?
(1) 032204 100013 BPL 2$ ; BR IF NO--UNSAFE CLEARED
(1) 032206 112761 177777 027050 MOVB #-1, DRVSTA(R1) ; DRIVE IS UNSAFE
(1) 032214 004737 033434 JSR PC, SVRH11 ; SAVE RH11/RP04 REGISTERS
(1) 032220 010364 000016 MOV R3, RHAS(R4) ; CLEAR ATTENTION
(1) 032224 052762 110000 000016 BIS #BIT15!BIT12, 16(R2) ; INFORM USER OF UNSAFE ERROR
(1) 032232 000421 BR SC7
(1) 032234 105705 2$: TSTB R5 ; "DRY" =1?
(1) 032236 100414 BMI 3$ ; BRANCH IF YES
(1) 032240 112761 177777 027040 MOVB #-1, DRVACT(R1) ; ACTIVE ERROR RECOVER
(1) 032246 112761 000001 027050 MOVB #1, DRVSTA(R1) ; ONLINE
(1) 032254 006301 R1
(1) 032256 012761 072460 027132 MOV #30000., TIMER(R1) ; START 30 SECOND TIMER
(1) 032264 006201 ASR R1
(1) 032266 000623 BR SC3
(1) 032270 052762 100220 000016 3$: BIS #BIT15!BIT07!BIT04, 16(R2) ; INFORM USER OF ERROR
(1) 032276 105061 027040 SC7: CLRB DRVACT(R1) ; DRIVE IS IDLE
(1) 032302 004737 034154 JSR PC, EMPTYQ ; DUMP THE QUEUE
(1) 032306 105761 027106 TSTB ULDFLG(R1) ; UNLOAD IN PROGRESS OR QUEUE?
(1) 032312 001611 BEQ SC3 ; BR IF NO
(1) 032314 105061 027106 CLRB ULDFLG(R1) ; CLEAR UNLOAD FLAG
(1) 032320 000606 BR SC3
(1) 032322 005726 SC8: TST (SP)+ ; REMOVE (RHAS) FROM THE STACK
(1) 032324 105761 027040 TSTB DRVACT(R1) ; IS DRIVE IDLE?
(1) 032330 001404 BEQ 1$ ; YES--BRANCH
(1) 032332 004737 034250 JSR PC, GETREQ ; GET DPB POINTER
(1) 032336 000137 030772 JMP CI7 ; PROCESS THE PARITY ERROR
(1) 032342 000137 031004 1$: JMP CI7B ; PROCESS THE PARITY ERROR
(1) 032346 105761 027040 SC9: TSTB DRVACT(R1) ; TEST DRIVE ACTIVE
(1) 032352 003013 BGT SC11 ; BRANCH IF DRIVE IS ACTIVE
(1) 032354 001404 BEQ SC10 ; BRANCH IF DRIVE IS IDLE
(1) 032356 052762 100210 000016 BIS #BIT15!BIT07!BIT03, 16(R2) ; INFORM USER OF ERROR RECOVER COMPLETION
(1) 032364 000744 BR SC7
(1) 032366 011605 SC10: MOV (SP), R5 ; PUT (RHAS) IN R5
(1) 032370 004737 032464 JSR PC, SC12 ; SAVE RHDS1, RHER1, RHER2, AND RHER3
(2) 032374 104002 ERROR 2 ; REPORT THE ERROR
(1) 032376 000137 031736 JMP SC3 ; GO CHECK FOR MORE ATA'S
(1) 032402 105761 027106 SC11: TSTB ULDFLG(R1) ; "UNLOAD IN PROGRESS"?
(1) 032406 003402 BLE 1$ ; BRANCH IF NO
(1) 032410 105061 027106 CLRB ULDFLG(R1) ; CLEAR UNLOAD FLAG
(1) 032414 105061 027040 1$: CLRB DRVACT(R1) ; SET DRIVE IDLE
(1) 032420 136137 027154 027102 BITB ATABIT(R1), SRCHWT ; DOING A SEARCH OPERATION FOR AN I/O COMMAND?
(1) 032426 001012 BNE 2$ ; BRANCH IF YES
(1) 032430 004737 034272 JSR PC, POPQUE ; REMOVE REQUEST FROM QUEUE
(1) 032434 052762 000200 000016 BIS #BIT07, 16(R2) ; SET "DONE" BIT
(1) 032442 005737 027126 TST SAVEFG ; SAVE THE RH11/RP04 REGISTERS?
(1) 032446 100002 BPL 2$ ; BRANCH IF NO
(1) 032450 004737 033434 JSR PC, SVRH11 ; YES--SAVE ALL OF THE RH11/RP04 REG'S
(1) 032454 004737 027770 2$: JSR PC, OPT ; START A REQUEST

```

```

(1) 032460 000137 031736          JMP      SC3
(1) 032464 010164 000010          MOV     SC12: R1,RHCS2(R4) ;SELECT DRIVE
(1) 032470 016437 000012 027030  MOV     RHDS1(R4),RPERRS ;SAVE THE FOUR REGISTERS THAT
(1) 032476 016437 000014 027032  MOV     RHER1(R4),RPERRS+2 ;WILL TELL US SOMETHING
(1) 032504 016437 000040 027034  MOV     RHER2(R4),RPERRS+4
(1) 032512 016437 000042 027036  MOV     RHER3(R4),RPERRS+6
(1) 032520 004037 027346          JSR     RO,DRVINT ;INIT. THE STATE OF THE DRIVE
(1) 032524 000401          BR      1$ ;TAKE ERROR EXIT
(1) 032526 000207          RTS    PC ;RETURN
(1) 032530 005726          1$:   TST   (SP)+ ;POP PC OFF OF THE STACK
(1) 032532 000673          BR      SC8 ;PROCESS THE PARITY ERROR

(1)
(1)
(1) ;*RPO4 TIMER ROUTINE
(1) ;*CALL
(1) ;*
(1) ;*   MOV     #TIME, -(SP) ;ELAPSED TIME IN MILLISECONDS ON THE STACK
(1) ;*   JSR     RO,RPTMR ;CALL RPO4 TIME ROUTINE
(1)
(1) RPTMR: TST     ACTDRV ;CHECK "ACTDRV & ACTSTR"
(1) 032534 005737 027104          BNE     4$ ;IF NON ZERO EXIT
(1) 032540 001032          MOVB   #1,ACTSTR ;SET "ACTSTR"
(1) 032542 112737 000001 027105  JSR     RO,SAVR15 ;SAVE R1-R5
(1) 032550 004037 034334          CLR    R1 ;START WITH DRIVE 0
(1) 032554 005001          CLR    R3
(1) 032556 005003          TST   TIMER(R3) ;IS THE TIMER RUNNING?
(1) 032560 005763 027132          1$:   BLT   2$ ;BRANCH IF NO
(1) 032564 002407          SUB    16(SP),TIMER(R3) ;COUNT THE INTERVAL
(1) 032566 166663 000016 027132  BGT    2$ ;BR IF NO SOFTWARE TIMEOUT
(1) 032574 003003          JSR     RO,STO ;CALL SOFTWARE TIMEOUT ROUTINE
(1) 032576 004037 032632          BR     3$ ;GO TO THE EXIT
(1) 032602 000405          2$:   INC   R1 ;MOVE TO NEXT DRIVE
(1) 032604 005201          TST   (R3)+
(1) 032606 005723          CMP    #8.,R1 ;OUT OF DRIVES?
(1) 032610 022701 000010          BGT    1$ ;BRANCH IF NO
(1) 032614 003361          JSR     RO,GETR15 ;RESTORE R1-R5
(1) 032616 004037 034354          CLRB  ACTSTR ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
(1) 032622 105037 027105          3$:   MOV   (SP)+,(SP) ;ADJUST THE STACK
(1) 032626 012616          4$:   RTS   RO ;RETURN
(1) 032630 000200

(1) ;*SOFTWARE TIMEOUT ROUTINE
(1) ;*
(1) ;*CALL: STO
(1) ;*   MOV     #DRVNUM,R1 ;DRIVE NUMBER
(1) ;*   JSR     RO,STO ;CALL--DRVACT MUST BE NONZERO
(1) ;*   RETURN
(1)
(1) STO:   MOV     @#PS, -(SP) ;SAVE THE PROCESSOR STATUS
(1) 032632 013746 177776          MOV     RPVEC+2,@#PS ;SET THE "PS" TO RPO4 BR LEVEL
(1) 032636 013737 027172 177776  JSR     RO,SAVR15 ;SAVE R1-R5
(1) 032644 004037 034334          MOV     RPADR,R4 ;GET ADDRESS OF "RHCS1"
(1) 032650 013704 027166          MOV     R1,RHCS2(R4) ;SELECT THE DRIVE
(1) 032654 010164 000010          JSR     RO,RO.RP ;READ "DRIVE STATUS REG"
(1) 032660 004037 033134          RHDS1
(1) 032664 000012          STOS
(1) 032666 033126          TSTB  (SP)+ ;IS "DRY"=1?
(1) 032670 105726

```

```

(1) 032672 100471          BMI      ST02          ;BR IF YES
(1) 032674 013702 027100  ST01:  MOV      TRANSW,R2  ;PICKUP TRANSFER WAIT QUEUE
(1) 032700 020137 027152          CMP      R1,DTUW    ;TRANSFER UNDERWAY ON THIS DRIVE?
(1) 032704 001402          BEQ      1$         ;BRANCH IF YES
(1) 032706 004737 034250          JSR      PC,GETREQ  ;GET DPB ADDRESS
(1) 032712 052762 101000 000016 1$:  BIS      #BIT15:BIT09,16(R2) ;SET THE ERROR FLAGS
(1) 032720 004737 033434          JSR      PC,SVRH11 ;SAVE RH11/RP04 REGISTERS
(1) 032724 012764 000040 000010  MOV      #BIT05,RHCS2(R4) ;"INIT" THE MASS BUS
(1) 032732 105061 027040          CLRB    DRVACT(R1)  ;DRIVE IS IDLE
(1) 032736 105061 027106          CLRB    ULDFLG(R1)  ;CLEAR THE UNLOAD FLAG
(1) 032742 005001          CLR     R1         ;START WITH DRIVE 0
(1) 032744 005003          CLR     R3
(1) 032746 004037 027346 2$:  JSR      RD,DRVINT  ;INIT. THIS DRIVE
(1) 032752 000465          BR      ST05       ;PARITY ERROR RETURN
(1) 032754 105761 027040          TSTB   DRVACT(R1)  ;DRIVE IDLE BEFORE THE INIT.?
(1) 032760 001414          BEQ      4$         ;YES--BRANCH
(1) 032762 013702 027100          MOV      TRANSW,R2  ;GET TRANSFER WAIT QUEUE
(1) 032766 023701 027152          CMP      DTUW,R1    ;WAS THERE I/O ON THIS DRIVE?
(1) 032772 001402          BEQ      3$         ;YES--BRANCH
(1) 032774 004737 034250          JSR      PC,GETREQ  ;GET THE DPB POINTER FROM QUEUE
(1) 033000 052762 100400 000016 3$:  BIS      #BIT15:BIT08,16(R2) ;INFORM USER OF INIT.
(1) 033006 105061 027040          CLRB    DRVACT(R1)  ;SET DRIVE ACTIVE TO IDLE
(1) 033012 105061 027106 4$:  CLRB    ULDFLG(R1)  ;NO UNLOAD
(1) 033016 012763 177777 027132  MOV      #-1,TIMER(R3) ;STOP THE TIMER
(1) 033024 005723          TST     (R3)+      ;UPDATE THE INDEX
(1) 033026 005201          INC     R1         ;INCREMENT THE DRIVE NUMBER
(1) 033030 022701 000010          CMP     #8.,R1     ;LAST DRIVE BEEN CHECKED?
(1) 033034 003344          BGT     2$         ;NO--LOOP
(1) 033036 012737 177777 027152  MOV      #-1,DTUW   ;NO DATA TRANSFERS UNDERWAY
(1) 033044 005037 027100          CLR     TRANSW     ;CLEAR TRANSFER WAIT QUEUE
(1) 033050 004737 034072          JSR      PC,CLRQUE  ;CLEAR ALL REQUEST QUEUES
(1) 033054 000417          BR      ST04       ;EXIT
(1) 033056 016405 000016  ST02:  MOV      RHAS(R4),R5 ;IS ATTENTION FOR THIS
(1) 033062 136105 027154          BITB   ATABIT(R1),R5 ;DRIVE UP?
(1) 033066 001011          BNE     ST03       ;YES--BRANCH
(1) 033070 020137 027152          CMP     R1,DTUW    ;DATA TRANSFER UNDERWAY FOR THIS DRIVE
(1) 033074 001277          BNE     ST01       ;BR IF NO
(1) 033076 004037 033134          JSR      RD,RD.RP  ;YES--CHECK "RDY"
(1) 033102 000000          RHCS1
(1) 033104 033126          ST05:  ST05
(1) 033106 105726          TSTB   (SP)+
(1) 033110 100271          BPL     ST01       ;BR IF "RDY"=0
(1) 033112 005720  ST03:  TST     (RD)+      ;ADJUST FOR THE PROPER RETURN
(1) 033114 004037 034354  ST04:  JSR      RD,GETR15 ;RESTORE R1-R5
(1) 033120 012637 177776          MOV     (SP)+,2#PS ;RESTORE PROCESSOR STATUS
(1) 033124 000200          RTS     RD         ;RETURN
(1) 033126 004737 031056  ST05:  JSR      PC,CIB    ;GO HANDLE THE PARITY ERROR
(1) 033132 000770          BR      ST04
(1)
(1) ;*ROUTINE TO READ A RH11/RP04 REGISTER
(1) ;
(1) ;*CALL
(1) ;*   JSR      RD,RD.RP ;GO READ A REGISTER
(1) ;*   INDEX    ;REG. INDEX FROM BASE

```

```

(1) ;* ERRADR ;ERROR ADDRESS--PROCESS ERROR STARTING
(1) ;* ;AT THIS ADDRESS
(1) ;* RETURN ;CONTENTS OF REG. IS ON THE STACK
(1) ;
(1) 033134 013737 027164 033260 RD.RP: MOV MCPEMX, RD.RP2 ;MAX. RETRYS ALLOWED
(1) 033142 011646 MOV (SP), -(SP) ;SAVE RD FOR RETURN
(1) 033144 013737 027166 033160 MOV RPADR, RD.ADR ;FORM THE DESIRED ADDRESS
(1) 033152 062037 033160 ADD (RD)+, RD.ADR ;USING THE BASE AND THE INDEX
(1) 033156 013727 RD.RP1: MOV @ (PC)+, (PC)+ ;READ THE DESIRED REGISTER OF THE RP04
(1) 033160 000000 RD.ADR: .WORD 0 ;ADDRESS IS FORMED HERE
(1) 033162 000000 RD.WRD: .WORD 0 ;REG. CONTENTS PUT HERE
(1) 033164 013766 033162 000002 MOV RD.WRD, 2(SP) ;RETURN IT TO THE USER
(1) 033172 017746 173770 MOV @RPADR, -(SP) ;READ RHCS1
(1) 033176 032716 020000 BIT #BIT13, (SP) ;DID MCPE SET?
(1) 033202 001002 BNE 1$ ;BRANCH IF YES
(1) 033204 022620 CMP (SP)+, (RD)+ ;ADJUST FOR RETURN
(1) 033206 000200 RTS RD ;RETURN IF NO
(2) 033210 1$:
(2) 033210 104003 ERROR 3 ;REPORT THE ERROR
(1) 033212 005737 027152 TST DTUW ;DATA TRANSFER UNDERWAY?
(1) 033216 100405 BMI 2$ ;NO--BRANCH
(1) 033220 032716 040000 BIT #BIT14, (SP) ;NO--"TRE"=1?
(1) 033224 001402 BEQ 2$ ;NO--BRANCH
(1) 033226 005726 TST (SP)+ ;YES--CLEAN OFF THE STACK AND
(1) 033230 000415 BR RD.RP3 ;TAKE THE FATAL ERROR EXIT
(1) 033232 052716 040000 2$: BIS #BIT14, (SP) ;CLEAR "MCPE" BY SENDING A "1" TO "TRE"
(1) 033236 000316 SWAB (SP) ;POSITION BEFORE WRITING
(1) 033240 013737 027166 033254 MOV RPADR, 3$ ;FORM ADDRESS OF HIGH BYTE
(1) 033246 005237 033254 INC 3$
(1) 033252 112637 MOVW (SP)+, @ (PC)+ ;WRITE THE HIGH BYTE OF RHCS1
(1) 033254 000000 3$: .WORD 0 ;ADDRESS STORAGE
(1) 033256 005327 DEC (PC)+ ;EXCEEDED MAX. RETRYS
(1) 033260 000003 RD.RP2: .WORD 3
(1) 033262 002335 RD.RP1: BGE RD.RP1 ;BRANCH IF NO
(1) 033264 011000 RD.RP3: MOV (RD), RD ;FATAL ERROR EXIT
(1) 033266 012616 MOV (SP)+, (SP)
(1) 033270 000200 RTS RD
(1) ;
(1) ;*ROUTINE TO WRITE A RH11/RP04 REGISTER
(1) ;*CALL
(1) ;* MOV DATA, -(SP) ;DATA TO BE LOADED ON THE STACK
(1) ;* JSR RD, WRT.RP ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
(1) ;* INDEX ;INDEX OF THE REGISTER TO BE LOADED
(1) ;* ERRADR ;ADDRESS TO RETURN TO ON AN ERROR
(1) ;* RETURN ;ERROR FREE RETURN
(1) ;
(1) 033272 016637 000002 033360 WRT.RP: MOV 2(SP), WRT.WD ;SAVE THE WORD TO WRITE
(1) 033300 012616 MOV (SP)+, (SP) ;ADJUST THE STACK
(1) 033302 012037 033362 MOV (RD)+, WRT.AD ;GET INDEX OF REGISTER TO BE WRITTEN
(1) 033306 001020 BNE 2$ ;BRANCH IF NOT RHCS1
(1) 033310 122737 000150 033360 CMPB #150, WRT.WD ;IS THE COMMAND FOR DATA TRANSFERS?
(1) 033316 002414 BLT 2$ ;YES--DON'T GET THE OLD A16&A17, & PSEL
(1) 033320 004037 033134 JSR RD, RD.RP ;NO---COMBINE A16&A17, & PSEL WITH

```



```

(1) 033324 000000 RHCS1 ;THE COMMAND BEFORE SENDING IT TO
(1) 033326 033344 1$ ;THE RH11/RP04
(1) 033330 000316 SWAB (SP)
(1) 033332 042716 177770 BIC #C7,(SP)
(1) 033336 112637 033361 MOVB (SP)+,WRT.WD+1
(1) 033342 000402 BR 2$
(1) 033344 011000 1$: MOV (R0),R0 ;TAKE THE ERROR EXIT
(1) 033346 000200 RTS R0
(1) 033350 063737 027166 033362 2$: ADD RPADR,WRT.AD ;FORM THE ADDRESS OF THE DISK REG.
(1) 033356 012737 MOVB (PC)+,2(PC)+ ;LOAD THE DESIRED REG.
(1) 033360 000000 WRT.WD: .WORD 0 ;WORD TO WRITE GOES HERE
(1) 033362 000000 WRT.AD: .WORD 0 ;ADDRESS IS FORMED HERE
(1) 033364 004037 033134 JSR R0,RD.RP ;CHECK FOR PARITY ERROR ON WRITE
(1) 033370 000014 RHER1
(1) 033372 033424 2$
(1) 033374 032726 000010 BIT #BIT03,(SP)+
(1) 033400 001413 BEQ 3$ ;BRANCH IF "PAR=0"
(1) 033402 016037 177776 033414 MOV -2(R0),1$ ;PICKUP THE INDEX
(1) 033410 004037 033134 JSR R0,RD.RP ;READ THE REG.
(1) 033414 000000 1$: .WORD 0 ; REG. INDEX
(1) 033416 033424 2$ ;RETURN TO THIS ADDRESS ON ERROR
(2) 033420 104004 ERROR 4 ;REPORT THE ERROR
(1) 033422 005726 TST (SP)+ ;CLEAN OFF THE STACK
(1) 033424 011000 2$: MOV (R0),R0 ;TAKE THE "PARITY ON WRITE" ERROR EXIT
(1) 033426 000200 RTS R0
(1) 033430 005720 3$: TST (R0)+
(1) 033432 000200 RTS R0 ;ADJUST FOR ERROR FREE EXIT

(1) ;*ROUTINE TO SAVE THE RH11/RP04 REGISTERS AS PER DPB+14
(1) ;*CALL
(1) ;* MOV #DPBNUM,R2 ;DPB POINTER TO R2
(1) ;* JSR PC,SVRH11 ;SAVE THE DRIVES REG'S
(1) SVRH11: JSR R0,SAVR15 ;SAVE REG.'S R1-R5
(1) 033440 013704 027166 MOV RPADR,R4
(1) 033444 111264 000010 MOVB (R2),RHCS2(R4) ;SELECT DRIVE
(1) 033450 016202 000014 MOV 14(R2),R2 ;GET THE ERROR TABLE POINTER
(1) 033454 001421 BEQ 4$ ;EXIT IF 0
(1) 033456 005003 CLR R3 ;COUNTER & POINTER
(1) 033460 012705 000022 MOV #RHDB,R5 ;PROBLEM REGISTER
(1) 033464 020305 1$: CMP R3,R5 ;REACHED RHDB?
(1) 033466 001005 BNE 2$ ;BR IF NO
(1) 033470 105764 000010 TSTB RHCS2(R4) ;CHECK "OR"
(1) 033474 100402 BMI 2$ ;BRANCH IF RHDB CAN BE READ
(1) 033476 005022 CLR (R2)+ ;ELSE SAVE IT AS 0'S
(1) 033500 000403 BR 3$
(1) 033502 010446 2$: MOV R4,-(SP) ;FORM RH11/RP04 ADDRESS THAT IS
(1) 033504 060316 ADD R3,(SP) ;TO BE READ
(1) 033506 013622 MOVB 2(SP)+,(R2)+ ;AND READ IT
(1) 033510 005723 3$: TST (R3)+ ;MOVE TO NEXT REG INDEX
(1) 033512 020327 000046 CMP R3,#RHEC2 ;DONE?
(1) 033516 003762 BLE 1$ ;BRANCH IF NO
(1) 033520 004037 034354 4$: JSR R0,GETR15 ;GET REGISTERS R1-R5

```



```

(1) 033524 000207          RTS      PC          ;RETURN TO USER
(1)
(1)                        ;*ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
(1)                        ;*CALL
(1)                        ;*
(1)                        ;*   MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
(1)                        ;*   JSR      PC,SET.IE      ;SET "IE"
(1)                        ;*   RETURN
(1)
(1) 033526 010446          SET.IE: MOV     R4,-(SP)      ;SAVE R4
(1) 033530 013704 027166   MOV     RPAOR,R4      ;PICKUP ADDRESS OF RHCS1
(1) 033534 010164 000010   MOV     R1,RHCS2(R4) ;SELECT DRIVE
(1) 033540 011446          MOV     (R4),-(SP)    ;READ RHCS1
(1) 033542 052716 040000   BIS     #BIT14,(SP)  ;SET THE "TRE" BIT OF THE WORD READ
(1) 033546 000316          SWAB    (SP)         ;ADJUST FOR DATO
(1) 033550 112714 000100   MOVB   #BIT06,(R4)  ;SET "IE"
(1) 033554 032764 010000 000010 BIT     #BIT12,RHCS2(R4) ;IS "NED"=1?
(1) 033562 001002          BNE     1$          ;YES--CLEAR "TRE"
(1) 033564 005726          TST     (SP)+       ;CLEAN OFF THE STACK
(1) 033566 000402          BR      2$
(1) 033570 112664 000001 1$:  MOVB   (SP)+,1(R4)  ;CLEAR "TRE"
(1) 033574 012604          2$:  MOV     (SP)+,R4   ;RESTORE R4
(1) 033576 000207          RTS      PC          ;RETURN TO CALLER
(1)
(1)                        ;*QUEUE COUNT
(1) 033600          000          QCNT:  .BYTE 0          ;DRIVE 0
(1) 033601          000          .BYTE 0          ;DRIVE 1
(1) 033602          000          .BYTE 0          ;DRIVE 2
(1) 033603          000          .BYTE 0          ;DRIVE 3
(1) 033604          000          .BYTE 0          ;DRIVE 4
(1) 033605          000          .BYTE 0          ;DRIVE 5
(1) 033606          000          .BYTE 0          ;DRIVE 6
(1) 033607          000          .BYTE 0          ;DRIVE 7
(1)
(1)                        ;QUEUE INPUT POINTERS
(1) 033610 033672          QINPT: .WORD  QDRV0      ;DRIVE 0
(1) 033612 033712          .WORD  QDRV1      ;DRIVE 1
(1) 033614 033732          .WORD  QDRV2      ;DRIVE 2
(1) 033616 033752          .WORD  QDRV3      ;DRIVE 3
(1) 033620 033772          .WORD  QDRV4      ;DRIVE 4
(1) 033622 034012          .WORD  QDRV5      ;DRIVE 5
(1) 033624 034032          .WORD  QDRV6      ;DRIVE 6
(1) 033626 034052          .WORD  QDRV7      ;DRIVE 7
(1)
(1)                        ;QUEUE OUTPUT POINTERS
(1) 033630 033672          QOUTPT: .WORD  QDRV0     ;DRIVE 0
(1) 033632 033712          .WORD  QDRV1     ;DRIVE 1
(1) 033634 033732          .WORD  QDRV2     ;DRIVE 2
(1) 033636 033752          .WORD  QDRV3     ;DRIVE 3
(1) 033640 033772          .WORD  QDRV4     ;DRIVE 4
(1) 033642 034012          .WORD  QDRV5     ;DRIVE 5
(1) 033644 034032          .WORD  QDRV6     ;DRIVE 6
(1) 033646 034052          .WORD  QDRV7     ;DRIVE 7

```

```

(1)
(1) 033650 033672 QSTART: .WORD QDRV0 ;DRIVE 0 START ADDRESS
(1) 033652 033712 QSTOP: .WORD QDRV1 ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
(1) 033654 033732 .WORD QDRV2 ;STOP DRIVE 1--START DRIVE 2
(1) 033656 033752 .WORD QDRV3 ;STOP DRIVE 2--START DRIVE 3
(1) 033660 033772 .WORD QDRV4 ;STOP DRIVE 3--START DRIVE 4
(1) 033662 034012 .WORD QDRV5 ;STOP DRIVE 4--START DRIVE 5
(1) 033664 034032 .WORD QDRV6 ;STOP DRIVE 5--START DRIVE 6
(1) 033666 034052 .WORD QDRV7 ;STOP DRIVE 6--START DRIVE 7
(1) 033670 034072 .WORD QTERM ;STOP DRIVE 7
(1)
(1) ;DRIVE REQUEST QUEUES
(1)
(1) 033672 000010 QDRV0: .BLKW 10
(1) 033712 000010 QDRV1: .BLKW 10
(1) 033732 000010 QDRV2: .BLKW 10
(1) 033752 000010 QDRV3: .BLKW 10
(1) 033772 000010 QDRV4: .BLKW 10
(1) 034012 000010 QDRV5: .BLKW 10
(1) 034032 000010 QDRV6: .BLKW 10
(1) 034052 000010 QDRV7: .BLKW 10
(1) 034072 034072 QTERM=.
(1)
(1) ;*ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
(1)
(1) ;*CALL
(1) ;* JSR PC,CLRQUE
(1)
(1) 034072 004037 034334 CLRQUE: JSR R0,SAVR15 ;SAVE R1-R5
(1) 034076 012702 033600 MOV #QCNT,R2 ;ZERO THE QUEUE COUNTS
(1) 034102 005022 CLR (R2)+ ;DRIVES 0 & 1
(1) 034104 005022 CLR (R2)+ ;DRIVES 2 & 3
(1) 034106 005022 CLR (R2)+ ;DRIVES 4 & 5
(1) 034110 005022 CLR (R2)+ ;DRIVES 6 & 7
(1) 034112 012703 000010 MOV #8,R3 ;MOVE THE STARTING
(1) 034116 012701 033650 MOV #QSTART,R1 ;ADDRESS OF THE QUEUE INTO
(1) 034122 012122 1$: MOV (R1)+,(R2)+ ;THE QUEUE INPUT POINTER
(1) 034124 005303 DEC R3
(1) 034126 001375 BNE 1$
(1) 034130 012703 000010 MOV #8,R3 ;MOVE THE STARTING ADDRESS
(1) 034134 012701 033650 MOV #QSTART,R1 ;OF THE QUEUE INTO THE
(1) 034140 012122 2$: MOV (R1)+,(R2)+ ;QUEUE OUTPUT POINTER
(1) 034142 005303 DEC R3
(1) 034144 001375 BNE 2$
(1) 034146 004037 034354 JSR R0,GETR15 ;RESTORE R1-R5
(1) 034152 000207 RTS PC
(1)
(1) ;*EMPTY THE QUEUE SPECIFIED BY R1
(1)
(1) ;*CALL
(1) ;* MOV DRVNUM,R1 ;DRIVE NUMBER TO R1
(1) ;* JSR PC,EMPTYQ
(1)
(1) 034154 105061 033500 EMPTYQ: CLRB QCNT(R1) ;CLEAR NUMBER OF ITEMS IN QUEUE

```

```

(1) 034160 006301 ASL R1
(1) 034162 015161 033610 033630 MOV QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
(1) 034170 006201 ASR R1
(1) 034172 000207 RTS PC

;#ROUTINE TO PUT A REQUEST IN QUEUE
;#CALL
;# MOV #DRVNUM,R1 ;DRIVE NUMBER
;# MOV #DPB,R2 ;ADDRESS OF PARAMETER BLOCK
;# JSR RO,DRVQUE ;GO PUT REQUEST IN QUEUE
;# RETURN1 ;RETURN HERE IF QUEUE IS FULL
;# RETURN2 ;RETURN HERE IF REQUEST IS IN QUEUE

(1) 034174 122761 000010 033650 DRVQUE: CMPB #10,QCNT(R1) ;IS QUEUE FULL?
(1) 034202 001421 BEQ 2$ ;BR IF YES-TAKE RETURN1
(1) 034204 105261 03360C INCB QCNT(R1) ;INCREMENT QUEUE COUNT
(1) 034210 006301 ASL R1
(1) 034212 010271 033610 MOV R2,QINPT(R1) ;PUT THIS REQUEST IN QUEUE
(1) 034216 062761 000002 033610 ADD #2,QINPT(R1) ;UPDATE THE QUEUE POINTER
(1) 034224 026161 033610 033652 CMP QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
(1) 034232 001003 BNE 1$ ;BRANCH IF NO
(1) 034234 016161 033650 033610 MOV QSTART(R1),QINPT(R1) ;YES--RESET POINTER
(1) 034242 006201 1$: ASR R1
(1) 034244 005720 TST (R0)+ ;TAKE RETURN 2
(1) 034246 000200 2$: RTS RO ;RETURN TO USER

;#ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
;#CALL
;# MOV #DRVNUM,R1 ;DRIVE NUMBER TO R1
;# JSR PC,GETREQ ;GO GET THE REQUEST
;# RETURN ;R2="DPB" ADDRESS OF THE REQUEST
;# ;R2=0 IF NO REQUEST IN QUEUE

(1) 034250 005002 GETREQ: CLR R2
(1) 034252 105761 033600 TSTB QCNT(R1) ;IS THERE ANY REQUEST IN QUEUE?
(1) 034256 001404 BEQ 2$ ;NO---BRANCH
(1) 034260 006301 1$: ASL R1
(1) 034262 017102 033630 MOV #QOUTPT(R1),R2 ;PICKUP "DPB" POINTER FOR THIS DRIVE
(1) 034266 006201 ASR R1
(1) 034270 000207 2$: RTS PC ;RETURN TO USER

;#ROUTINE TO "POP" THE REQUEST FROM QUEUE
;#CALL
;# MOV #DRVNUM,R1 ;DRIVE NUMBER TO R1
;# JSR PC,POPQUE ;CALL TO REMOVE REQUEST
;# RETURN ;R2=ADDRESS OF DPB REMOVED

(1) 034272 105361 03360C POPQUE: DECB QCNT(R1) ;DECREMENT QUEUE COUNT
(1) 034276 006301 ASL R1
(1) 034300 017102 033630 MOV #QOUTPT(R1),R2 ;GET THE "DPB" POINTER
(1) 034304 062761 000002 033630 ADD #2,QOUTPT(R1) ;UPDATE THE QUEUE POINTER
  
```

```

(1) 034312 026161 033630 033652      CMP      QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
(1) 034320 001003                      BNE      15 ;NO--BRANCH TO EXIT
(1) 034322 016161 033650 033630      MOV      QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
(1) 034330 006201                      15:     ASR      R1
(1) 034332 000207                      RTS      PC ;RETURN TO USER

```

```

(1)                                     ;ROUTINES TO SAVE R0-R5 AND R1-R5
(1) *
(1) *CALL: SAVROS
(1) *      JSR      R0,SAVROS
(1) *      RETURN
(1) *                                     ;R0-R5 IS ON THE STACK
(1) *
(1) *CALL: SAVR15
(1) *      JSR      R0,SAVR15
(1) *      RETURN
(1) *                                     ;R1-R5 IS ON THE STACK
(1) *
(1) *UPON RETURN FROM SAVROS AND SAVR15 THE STACK WILL LOOK LIKE:
(1) *
(1) *++12  R0
(1) *++10  R1
(1) *++06  R2
(1) *++04  R3
(1) *++02  R4
(1) *TOP   R5

```

```

(1) 034334                                SAVROS:
(1) 034334 010146                        SAVR15: MOV      R1,-(SP) ;SAVE R0 BY THE JSR INST.
(1) 034336 010246                        MOV      R2,-(SP) ;SAVE R1
(1) 034340 010346                        MOV      R3,-(SP) ;SAVE R2
(1) 034342 010446                        MOV      R4,-(SP) ;SAVE R3
(1) 034344 010546                        MOV      R5,-(SP) ;SAVE R4
(1) 034346 016646 000012                MOV      12(SP),-(SP) ;SAVE R5
(1) 034352 000200                        RTS      R0 ;GET R0

```

```

(1)                                     ;ROUTINES TO RESTORE R0-R5 AND R1-R5
(1) *
(1) *CALL: GETROS
(1) *      JSR      R0,GETROS
(1) *      RETURN
(1) *                                     ;R0-R5 HAVE BEEN RESTORED
(1) *
(1) *CALL: GETR15
(1) *      JSR      R0,GETR15
(1) *      RETURN
(1) *                                     ;R1-R5 HAVE BEEN RESTORED

```

```

(1) 034354 011566 000014                GETR15: MOV      (SP),14(SP) ;POSITION R0
(1) 034360 005726                        GETROS: TST      (SP)+ ;POP R0 OF THE JSR OFF OF THE STACK
(1) 034362 012605                        MOV      (SP)+,R5 ;RESTORE R5
(1) 034364 012604                        MOV      (SP)+,R4 ;RESTORE R4
(1) 034366 012603                        MOV      (SP)+,R3 ;RESTORE R3
(1) 034370 012602                        MOV      (SP)+,R2 ;RESTORE R2
(1) 034372 012601                        MOV      (SP)+,R1 ;RESTORE R1
(1) 034374 000200                        RTS      R0 ;RESTORE R0

```

.SBTTL ASCIZ MESSAGES

8787					
8788					
8789	034376	000122			MSG.R: .ASCIZ /R/
9790	034400	041506	000		MSG.FC: .ASCIZ /FC/
8791	034403	114	000103		MSG.LC: .ASCIZ /LC/
8792	034406	041511	000		MSG.IC: .ASCIZ /IC/
8793	034411	106	000124		MSG.FT: .ASCIZ /FT/
8794	034414	052114	000		MSG.LT: .ASCIZ /LT/
8795	034417	111	000124		MSG.IT: .ASCIZ /IT/
8796	034422	051506	000		MSG.FS: .ASCIZ /FS/
8797	034425	114	000123		MSG.LS: .ASCIZ /LS/
8798	034430	040520	000124		MSG.PAT: .ASCIZ /PAT/
8799	034434	000075			MSG.EQ: .ASCIZ /=/
8800	034436	005015	047503	052116	MSG.CS: .ASCIZ <15><12>/CONTROL SWITCHES=/ 034444 047522 020114 053523 034452 052111 044103 051505 034460 000075
8801	034462	005015	044122	051503	MRHCS1: .ASCIZ <15><12>/RHCS1=/ 034470 036461 000
8802	034473	015	051012	053110	MRHVEC: .ASCIZ <15><12>/RHVEC=/ 034500 041505 000075
8803	034504	005015	044122	051120	MRHPRI: .ASCIZ <15><12>/RHPRI=/ 034512 047511 000075
8804					
8805	034516	005015	051012	052117	MSG7: .ASCIZ <15><12><12>/ROTATIONAL SPEED TIMES/ 034524 052101 047511 040516 034532 020114 050123 042505 034540 020104 044524 042515 034546 000123
8806	034550	005015	047412	042516	MSG10A: .ASCIZ <15><12><12>/ONE CYLINDER SEEK TIMES/<15><12>/** 0, 1, 2,...410 **/ 034556 041440 046131 047111 034564 042504 020122 042523 034572 045505 052040 046511 034600 051505 005015 025052 034606 030040 020054 026061 034614 031040 027054 027056 034622 030464 020060 025052 034630 000
8807	034631	015	025012	020052	MSG10B: .ASCIZ <15><12>/** 410, 409, 408,...0 **/ 034636 030464 026060 032040 034644 034460 020054 030064 034652 026070 027056 030056 034660 025040 000052
8808	034664	005015	040412	042526	MSG11A: .ASCIZ <15><12><12>/AVERAGE SEEK TIMES/<15><12>/** 0 TO 136 **/ 034672 040522 042507 051440 034700 042505 020113 044524 034706 042515 006523 025012 034714 020052 020060 047524 034722 030440 033063 025040 034730 000052
8809	034732	005015	025052	030440	MSG11B: .ASCIZ <15><12>/** 136 TO 0 **/ 034740 033063 052040 020117 034746 020060 025052 000
8810	034753	015	005012	040515	MSG12A: .ASCIZ <15><12><12>/MAXIMUM SEEK TIMES/<15><12>/** 0 TO 410 **/

	034760	044530	052515	020115	
	034766	042523	045505	052040	
	034774	046511	051505	005015	
	035002	025052	030040	052040	
	035010	020117	030464	020060	
	035016	025052	000		
8811	035021	015	025012	020052	MSG128: .ASCIZ <15><12>/** 410 TO 0 **/
	035026	030464	020060	047524	
	035034	030040	025040	000052	
8812					
8813	035042	005015	044515	036516	MSGMIN: .ASCIZ <15><12>/MIN=/
	035050	000			
8814	035051	015	046412	054101	MSGMAX: .ASCIZ <15><12>/MAX=/
	035056	000075			
8815	035060	005015	053101	036507	MSGAVG: .ASCIZ <15><12>/AVG=/
	035066	000			
8816	035067	060	052440	000123	MSG2US: .ASCIZ /0 US/
8817	035074	041040	046105	053517	MSBELOW: .ASCIZ / BELOW THE MINIMUM OF /
	035102	052040	042510	046440	
	035110	047111	046511	046525	
	035116	047440	020106	000	
8818	035123	040	041101	053117	MABOVE: .ASCIZ / ABOVE THE MAXIMUM OF /
	035130	020105	044124	020105	
	035136	040515	044530	052515	
	035144	020115	043117	000040	
8819	035152	051440	042505	051513	MSGNUM: .ASCIZ / SEEKS TIMED/
	035160	052040	046511	042105	
	035166	000			
8820	035167	040	047516	020124	MSGNON: .ASCIZ / NOT TIMED/
	035174	044524	042515	000104	
8821	035202	020040	000		MSG.SP: .ASCIZ / / ;TWO (2) SPACES
8822					
8823					.SBTTL ERROR HEADER (EM) MESSAGES
8824					
8825	035205	111	046114	043505	EM1: .ASCIZ /ILLEGAL RH11(SC=0), INTERRUPT OCCURRED/
	035212	046101	051040	030510	
	035220	024061	041523	030075	
	035226	026051	047111	042524	
	035234	051122	050125	020124	
	035242	041517	052503	051122	
	035250	042105	000		
8826	035253	111	046114	043505	EM2: .ASCIZ /ILLEGAL RPO4 INTERRUPT OCCURRED/
	035260	046101	051040	030120	
	035266	020064	047111	042524	
	035274	051122	050125	020124	
	035302	041517	052503	051122	
	035310	042105	000		
8827	035313	115	051501	041123	EM3: .ASCIZ /MASSBUS PARITY ERROR(MCPE=1)/
	035320	051525	050040	051101	
	035326	052111	020131	051105	
	035334	047522	024122	041515	
	035342	042520	030475	000051	
8828	035350	040515	051523	052502	EM4: .ASCIZ /MASSBUS PARITY ERROR(PAR=1)/
	035356	020123	040520	044522	

	035364	054524	042440	051122		
	035372	051117	050050	051101		
	035400	030475	000051			
8829	035404	043117	046106	047111	EMS:	.ASCIZ /OFFLINE, NON-EXISTENT, OR UNSAFE DRIVE INTERRUPTED/
	035412	026105	047040	047117		
	035420	042455	044530	052123		
	035426	047105	026124	047440		
	035434	020122	047125	040523		
	035442	042506	042040	044522		
	035450	042526	044440	052116		
	035456	051105	052522	052120		
	035464	042105	000			
8830	035467	122	030510	027461	EM10:	.ASCIZ "RH11/RP04 FAILED TO RESPOND TO ADDRESSING"
	035474	050122	032060	043040		
	035502	044501	042514	020104		
	035510	047524	051040	051505		
	035516	047520	042116	052040		
	035524	020117	042101	051104		
	035532	051505	044523	043516		
	035540	000				
8831	035541	104	044522	042526	EM11:	.ASCIZ /DRIVE SELECTED IS NOT ONLINE/
	035546	051440	046105	041505		
	035554	042524	020104	051511		
	035562	047040	052117	047440		
	035570	046116	047111	000105		
8832	035576	046511	051120	050117	EM12:	.ASCIZ /IMPROPER HEADER DATA/
	035604	051105	044040	040505		
	035612	042504	020122	040504		
	035620	040524	000			
8833	035623	104	052101	020101	EM13:	.ASCIZ /DATA COMPARE FAILURE/
	035630	047503	050115	051101		
	035636	020105	040506	046111		
	035644	051125	000105			
8834	035650	044504	045523	042440	EM17:	.ASCIZ /DISK ERROR IN TIMING TEST/
	035656	051122	051117	044440		
	035664	020116	044524	044515		
	0356	043516	052040	051505		
	0357	000124				
8835	03570c	046103	041517	020113	EM20:	.ASCIZ /CLOCK (KW11-P) OVERFLOW IN TIMING TEST/
	035710	045450	030527	026461		
	035716	024520	047440	042526		
	035724	043122	047514	020127		
	035732	047111	052040	046511		
	035740	047111	020107	042524		
	035746	052123	000			
8836	035751	122	030510	027461	EM41:	.ASCIZ "RH11/RP04 ERROR"
	035756	050122	032060	042440		
	035764	051122	051117	000		
8837	035771	106	052101	046101	EM46:	.ASCIZ /FATAL WRITE CHECK ERROR/
	035776	053440	044522	042524		
	036004	041440	042510	045503		
	036012	042440	051122	051117		
	036020	000				
8838						

Line	Code	Code	Code	Code	Message
8839					.SBTTL STATUS/ERROR INDICATOR MESSAGES
8840					
8841	036021	117	043106	044514	MSG814: .ASCIZ /OFFLINE OR UNSAFE DRIVE REQUESTED/
	036026	042516	047440	020122	
	036034	047125	040523	042506	
	036042	042040	044522	042526	
	036050	051040	050505	042525	
	036056	052123	042105	000	
8842	036063	125	046116	040517	MSG813: .ASCIZ /UNLOADED DRIVE REQUESTED/
	036070	042504	020104	051104	
	036076	053111	020105	042522	
	036104	052521	051505	042524	
	036112	000104			
8843	036114	042520	051522	051511	MSG812: .ASCIZ /PERSISTENT UNSAFE/
	036122	042524	052116	052440	
	036130	051516	043101	000105	
8844	036136	040520	044522	054524	MSG811: .ASCIZ /PARITY ERROR OCCURRED/
	036144	042440	051122	051117	
	036152	047440	041503	051125	
	036160	042522	000104		
8845	036164	040506	040524	020114	MSG810: .ASCIZ /FATAL PARITY ERROR/
	036172	040520	044522	054524	
	036200	042440	051122	051117	
	036206	000			
8846	036207	123	043117	053524	MSG809: .ASCIZ /SOFTWARE TIMEOUT ON THIS DRIVE/
	036214	051101	020105	044524	
	036222	042515	052517	020124	
	036230	047117	052040	044510	
	036236	020123	051104	053111	
	036244	000105			
8847	036246	047523	052106	040527	MSG808: .ASCIZ /SOFTWARE TIMEOUT ON ANOTHER DRIVE/
	036254	042522	052040	046511	
	036262	047505	052125	047440	
	036270	020116	047101	052117	
	036276	042510	020122	051104	
	036304	053111	000105		
8848	036310	051105	047522	020122	MSG806: .ASCIZ "ERROR OCCURRED DURING I/O OPERATION"
	036316	041517	052503	051122	
	036324	042105	042040	051125	
	036332	047111	020107	027511	
	036340	020117	050117	051105	
	036346	052101	047511	000116	
8849	036354	051105	047522	020122	MSG805: .ASCIZ "ERROR OCCURRED DURING NON-I/O OPERATION"
	036362	041517	052503	051122	
	036370	042105	042040	051125	
	036376	047111	020107	047516	
	036404	026516	027511	020117	
	036412	050117	051105	052101	
	036420	047511	000116		
8850	036424	047125	040523	042506	MSG804: .ASCIZ /UNSAFE OCCURRED/
	036432	047440	041503	051125	
	036440	042522	000104		
8851	036444	052501	047524	040515	MSG803: .ASCIZ /AUTOMATIC RECALIBRATE SEQUENCE OCCURRED/
	036452	044524	020103	042522	

036460	040503	044514	051102
036466	052101	020105	042523
036474	052521	047105	042503
036502	047440	041503	051125
036510	042522	000104	

8852
8853
8854
8855

.SBTTL DATA HEADER (DT) MESSAGES

036514	042524	052123	020040
036522	020040	051105	020122
036530	041520	000	
036533	124	051505	020124
036540	020040	042440	051122
036546	050040	020103	042040
036554	044522	042526	020040
036562	051040	040510	020123
036570	020040	051040	042110
036576	020123	020040	051040
036604	042510	030522	020040
036612	051040	042510	031122
036620	020040	051040	042510
036626	031522	000	

8857
8858
8859

DH1: .ASCIZ /TEST ERR PC/

DH2: .ASCIZ /TEST ERR PC DRIVE RNAS RND5 RHER1 RHER2 RHER3/

036631	124	051505	020124
036636	020040	042440	051122
036644	050040	020103	040440
036652	042104	042522	051523
036660	042040	052101	000101
036666	042524	052123	020040
036674	020040	051105	020122
036702	041520	020040	042101
036710	051104	051505	020123
036716	042107	040504	040524
036724	020040	042102	040504
036732	040524	000	

DH3: .ASCIZ /TEST ERR PC ADDRESS DATA/

DH4: .ASCIZ /TEST ERR PC ADDRESS GDDATA BDDATA/

036735	122	041510	030523
036742	020040	042440	051122
036750	050040	000103	

DH10: .ASCIZ /RHCS1 ERR PC/

8860
8861

036754	051104	053111	020105
036762	020040	051105	020122
036770	041520	000	

DH11: .ASCIZ /DRIVE ERR PC/

036773	124	051505	020124
037000	020040	042440	051122
037006	050040	020103	052040
037014	052123	050040	020103
037022	042040	044522	042526
037030	020040	041440	046131
037036	042116	020122	052040
037044	040522	045503	020040
037052	051440	041505	047524
037060	000122		

DH12: .ASCIZ /TEST ERR PC TST PC DRIVE CYLNR TRACK SECTOR/

8862

03706?	042107	054503	020114
037070	020040	042107	051124
037076	020113	020040	042107
037104	041523	051124	020040

DH12A: .ASCIZ /GDCYL GDTRK GDSCTR BDCYL BDRK BOSCTR/

8870	037573	122	041510	030523	DH44A:	.ASCIZ	/RHCS1	RHCS2	RHDS	RHCC	RHCA	RHDA/
	037600	020040	051040	041510								
	037606	031123	020040	051040								
	037614	042110	020123	020040								
	037622	051040	041510	020103								
	037630	020040	051040	041510								
	037636	020101	020040	051040								
	037644	042110	000101									
8871	037650	044122	051105	020061	DH44B:	.ASCIZ	/RHER1	RHER2	RHER3/			
	037656	020040	044122	051105								
	037664	020062	020040	044122								
	037672	051105	000063									
8872	037676	044122	051105	020061	DH45A:	.ASCIZ	/RHER1	RHER2	RHER3	RHWC	RHBA	RHDB/
	037704	020040	044122	051105								
	037712	020062	020040	044122								
	037720	051105	020063	020040								
	037726	044122	041527	020040								
	037734	020040	044122	040502								
	037742	020040	020040	044122								
	037750	041104	000									

8873
8874
8875
8876
8877
8878

037754

.EVEN

.SBTTL DATA TABLE (DT)

8879	037754	001170	001116		DT1:	.WORD	\$TMP0,\$ERRPC
8880	037760	001170	001116	001156	DT2:	.WORD	\$TMP0,\$ERRPC,\$REG1,\$REG5,RPERRS,RPERRS+2,RPERRS+4,RPERRS+6
	037766	001166	027030	027032			
	037774	027034	027036				
8881	040000	001170	001116	033160	DT3:	.WORD	\$TMP0,\$ERRPC,RD.ADR,RD.WRD
	040006	033162					
8882	040010	001170	001116	033362	DT4:	.WORD	\$TMP0,\$ERRPC,WRT.ADR,WRT.WD,RD.WRD
	040016	033360	033162				
8883	040022	001170	001116	001156	DT5:	.WORD	\$TMP0,\$ERRPC,\$REG1,\$REG5,RPERRS,RPERRS+2,RPERRS+4,RPERRS+6
	040030	001166	027030	027032			
	040036	027034	027036				
8884	040042	001326	001116		DT10:	.WORD	RH.ADR,\$ERRPC
8885	040046	001160	001116		DT11:	.WORD	\$REG2,\$ERRPC
8886	040052	001170	001116	001154	DT12:	.WORD	\$TMP0,\$ERRPC,\$REG0,CHKDRV,CYL.DS,TRK.DS,SEC.DS
	040060	001236	001252	001256			
	040066	001254					
8887	040070	001252	001256	001254	DT12A:	.WORD	CYL.DS,TRK.DS,SEC.DS,CYL.RD,TRK.RD,SEC.RD
	040076	001244	001246	001250			
8888	040104	001170	001116	001154	DT13:	.WORD	\$TMP0,\$ERRPC,\$REG0,CHKDRV,CYL.DS,TRK.DS,SEC.DS
	040112	001236	001252	001256			
	040120	001254					
8889	040122	001124	001126	001164	DT13A:	.WORD	\$GDDAT,\$BDDAT,\$REG4,\$GDAOR,\$BDAOR
	040130	001120	001122				
8890	040134	001170	001116	001236	DT17:	.WORD	\$TMP0,\$ERRPC,CHKDRV,RP.REG,RP.REG+12,RP.REG+14,RP.REG+40,RP.REG+42
	040142	003540	003552	003554			
	040150	003600	003602				
8891	040154	001170	001116	001154	DT21:	.WORD	\$TMP0,\$ERRPC,\$REG0,CHKDRV,CYL.DS,TRK.DS
	040162	001236	001252	001256			

8892	040170	001156	001126	001164	DT21A:	.WORD	SREG1, SBDDAT, SREG4, SREG1
	040176	001156					
8893	040200	001170	001116	001154	DT41:	.WORD	STMPD, SERRPC, SREGO, CHKDRV
	040206	001236					
8894	040210	001170	001116	001154	DT42:	.WORD	STMPD, SERRPC, SREGO, CHKDRV, RP.REG, RP.REG+10, RP.REG+12
	040216	001236	003540	003550			
	040224	003552					
8895	040226	001170	001116	001154	DT43:	.WORD	STMPD, SERRPC, SREGO, CHKDRV, RP.REG, RP.REG+10, RP.REG+12
	040234	001236	003540	003550			
	040242	003552					
8896	040244	003554	003600	003602	DT43A:	.WORD	RP.REG+14, RP.REG+40, RP.REG+42
8897	040252	001170	001116	001154	DT44:	.WORD	STMPD, SERRPC, SREGO, CHKDRV, CYL.DS, TRK.DS, SEC.DS
	040260	001236	001252	001256			
	040266	001254					
8898	040270	003540	003550	003552	DT44A:	.WORD	RP.REG, RP.REG+10, RP.REG+12, RP.REG+36, RP.REG+34, RP.REG+06
	040276	003576	003574	003546			
8899	040304	003554	003600	003602	DT44B:	.WORD	RP.REG+14, RP.REG+40, RP.REG+42
8900	040312	001170	001116	001154	DT45:	.WORD	STMPD, SERRPC, SREGO, CHKDRV, CYL.DS, TRK.DS, SEC.DS
	040320	001236	001252	001256			
	040326	001254					
8901	040330	003540	003550	003552	DT45A:	.WORD	RP.REG, RP.REG+10, RP.REG+12, RP.REG+36, RP.REG+34, RP.REG+06
	040336	003576	003574	003546			
8902	040344	003554	003600	003602	DT45B:	.WORD	RP.REG+14, RP.REG+40, RP.REG+42, RP.REG+2, RP.REG+4, RP.REG+22
	040352	003542	003544	003562			

.SBTTL DATA FORMAT (DF) TABLE

8903							
8904							
8905							
8906	040360	000001			DF1:	.WORD	1
8907	040362	002				.BYTE	2
8908	040363	000				.BYTE	0
8909							
8910	040364	000001			DF2:	.WORD	1
8911	040366	010				.BYTE	108
8912	040367	000				.BYTE	0
8913							
8914	040370	000001			DF3:	.WORD	1
8915	040372	004				.BYTE	4
8916	040373	000				.BYTE	0
8917							
8918	040374	000001			DF4:	.WORD	1
8919	040376	005				.BYTE	5
8920	040377	000				.BYTE	0
8921							
8922	040400	000001			DF10:	.WORD	1
8923	040402	002				.BYTE	2
8924	040403	000				.BYTE	0
8925							
8926	040404	000001			DF11:	.WORD	1
8927	040406	002				.BYTE	2
8928	040407	000				.BYTE	0
8929							
8930	040410	000002			DF12:	.WORD	2
8931	040412	007				.BYTE	7
8932	040413	160				.BYTE	160

; NUMBER OF DATA HEADERS
 ; NUMBER OF WORDS IN DATA TABLE
 ; BOTH NUMBERS ARE OCTAL

; 2 DH'S TO BE TYPED
 ; 7 DATA WORDS FOLLOW THE 1ST DH
 ; WORDS 1-4 ARE OCTAL 5-7 ARE DECIMAL

8933	040414	037062	.WORD	DH12A	: ADDRESS OF 2ND DH
8934	040416	006	.BYTE	6	: 6 DATA WORDS FOLLOW THE 2ND DH
8935	040417	000	.BYTE	0	: ALL WORDS ARE OCTAL
8936					
8937	040420	000002	DF13: .WORD	2	
8938	040422	007	.BYTE	7	
8939	040423	160	.BYTE	160	
8940	040424	037141	.WORD	DH13A	
8941	040426	005	.BYTE	5	
8942	040427	004	.BYTE	4	;WORD 3 IS DECIMAL
8943					
8944	040430	000000	DF14: .WORD	0	
8945	040432	005	.BYTE	5	
8946	040433	004	.BYTE	4	;WORD 3 IS DECIMAL
8947					
8948	040434	000001	DF17: .WORD	1	
8949	040436	010	.BYTE	108	
8950	040437	000	.BYTE	0	
8951					
8952	040440	000002	DF21: .WORD	2	
8953	040442	006	.BYTE	5	
8954	040443	060	.BYTE	60	
8955	040444	037363	.WORD	DH21A	
8956	040446	004	.BYTE	4	
8957	040447	014	.BYTE	14	
8958					
8959	040450	000000	DF22: .WORD	0	
8960	040452	004	.BYTE	4	
8961	040453	014	.BYTE	14	
8962					
8963	040454	000001	DF41: .WORD	1	
8964	040456	004	.BYTE	4	
8965	040457	000	.BYTE	0	
8966					
8967	040460	000001	DF42: .WORD	1	
8968	040462	007	.BYTE	7	
8969	040463	000	.BYTE	0	
8970					
8971	040464	000002	DF43: .WORD	2	
8972	040466	007	.BYTE	7	
8973	040467	000	.BYTE	0	
8974	040470	037545	.WORD	DH43A	
8975	040472	003	.BYTE	3	
8976	040473	000	.BYTE	0	
8977					
8978	040474	000003	DF44: .WORD	3	
8979	040476	007	.BYTE	7	
8980	040477	160	.BYTE	160	
8981	040500	037573	.WORD	DH44A	
8982	040502	006	.BYTE	6	
8983	040503	000	.BYTE	0	
8984	040504	037650	.WORD	DH44B	
8985	040506	003	.BYTE	3	
8986	040507	000	.BYTE	0	

8987					
8988	040510	000003	DF45:	.WORD	3
8989	040512	007		.BYTE	7
8990	040513	160		.BYTE	160
8991	040514	037573		.WORD	DH44A
8992	040516	006		.BYTE	6
8993	040517	000		.BYTE	0
8994	040520	037676		.WORD	DH45A
8995	040522	006		.BYTE	6
8996	040523	000		.BYTE	0
8997					
8998			.EVEN		
8999	040524		BUFFER=.		
9000					
9001	000001		.END		

IT15	002310	5715#	7846		
LA	031220	8785#			
LACNT	027116	8785#*			
LC	001442	5715#			
LCO	002066	5715#	6339		
LC1	002102	5715#	6363		
LC15	002300	5715#	7863	7866	7868
LC16	002326	5715#	7129	8188	
LC2	002124	5715#	6391	6393	
LC3	002142	5715#	6420		
LC4	002160	5715#	6461	6463	6470
LC5	002176	5715#	6503	6510	
LC6	002214	5715#	6532		
LDCMD	017636	6251	7381#		
LKS	001352	5715#	7333	7365*	
LKV	001346	5715#	7363*	7364*	
LPB	001362	5715#	7169		
LPS	001360	5715#	7169	7295	
LPTAVL	001220	5715#	7169	7292*	7295*
LP.AVL	017316	6187	7292#		
LS	001456	5715#			
LS1	002114	5715#	6359		
LS15	002314	5715#	7015		
LT	001450	5715#			
LT1	002110	5715#	6361		
LT15	002306	5715#	7844	7847	7849
MABOVE	035123	7812	8818#		
MBELOW	035074	7802	8817#		
MCPENX	027164	8785#			
MNOLTA	027200	8785#			
MRHCSI	034462	8241	8801#		
MRHPRI	034504	8255	8803#		
MRHVEC	034473	8248	8802#		
MSGAVG	035060	7815	8815#		
MSG803	036444	5715	8851#		
MSG804	036424	5715	8850#		
MSG805	036354	5715	8849#		
MSG806	036310	5715	8848#		
MSG808	036246	5715	8847#		
MSG809	036207	5715	8846#		
MSG810	036164	5715	8845#		
MSG811	036136	5715	8844#		
MSG812	036114	5715	8843#		
MSG813	036063	5715	8842#		
MSG814	036021	5715	8841#		
MSGMAX	035051	7805	8814#		
MSGMIN	035042	7795	8813#		
MSGNON	035167	7832	8820#		
MSGNUM	035152	7827	8819#		
MSG.CS	034436	8216	8800#		
MSG.EQ	034434	8508	8529	8550	8799#
MSG.FC	034400	5715	8790#		
MSG.FS	034422	5715	8796#		
MSG.FT	024411	5715	8793#		

MSG.IC	034406	5715	8792#																	
MSG.IT	034417	5715	8795#																	
MSG.LC	034403	5715	8791#																	
MSG.LS	034425	5715	8797#																	
MSG.LT	034414	5715	8794#																	
MSG.PA	034430	8528	8798#																	
MSG.R	034376	5715	8789#																	
MSG.SP	035202	6263	7222	7248	7255	7262	7800	7810	7825	8504	8527	8821#								
MSGOUS	035067	7797	7804	7807	7814	7824	8816#													
MSG10A	034450	5715	8806#																	
MSG10B	034531	5715	8807#																	
MSG11A	034564	5715	8808#																	
MSG11B	034732	5715	8809#																	
MSG12A	034753	5715	8810#																	
MSG12B	035021	5715	8811#																	
MSG7	034516	5715	8805#																	
MXDLTA	027176	8785#																		
MXLACT	027174	8785#																		
MXSTAL	001370	5715#	6457	6477																
MXWINDW	027202	8785#																		
NOOP =	000101	4971#																		
OFFSET =	000115	4977#																		
OPMPAT	025702	8503	8527#	8533	8537	8540	8541													
OPNPRM	025506	8490	8495#	8499	8533	8544														
OPNTST	025350	8449	8458	8465	8468#															
OPNWDS	026020	8546#	8561																	
OPN.CT	025356	8470#	8472*	8473	8477	8482														
OPN.N1	026234	8524	8541	8584#																
OPN.N2	026240	8518	8535	8572	8585#															
OPN.X1	026252	8524	8541	8588#																
OPN.X2	026256	8523	8539	8576	8589#															
OPN.1	025362	8472#	8480	8587																
OPN.2	025406	8471	8474	8477#																
OPT	027770	8785#																		
PACK =	000123	4980#																		
PAT	001460	5715#																		
PAT.PT	002400	5715#	7994	8016	8030															
PAT0	002440	5715#	6167	8547	8577															
PAT1	002500	5715#																		
PAT10	003140	5715#																		
PAT11	003200	5715#																		
PAT12	003240	5715#																		
PAT13	003300	5715#																		
PAT14	003340	5715#																		
PAT15	003400	5715#																		
PAT2	002540	5715#																		
PAT3	002600	5715#																		
PAT4	002640	5715#																		
PAT5	002700	5715#																		
PAT6	002740	5715#																		
PAT7	003000	5715#																		
PAT8	003040	5715#	6166																	
PAT9	003100	5715#	6169																	
PC	=:000007	4968#	6187*	6188*	6190*	6191*	6192*	6193*	6196*	6251*	6277*	6597*	6607*	6616*						

SHNUM	017312	7114*	7280**	8087	8093*	8096*	8142	8160								
SYCNT	001104	5715*	7273*													
SITEMB	001114	5715*	7169*	7190												
SLF	001210	5715*	7169	7272												
SLONUM	017314	7115*	728J**	7604	8088	8092*	8095*	8141	8158	8159						
SLPADR	001106	5715*	6177*	6335*	6357*	6382*	6409*	6433*	6497*	6526*	6578*	6651*	6712*	6777*		
		6855*	6903*	7012*	7112*	7132*	7273*									
SLPERR	001110	5715*	6177*	6335*	6357*	6382*	6409*	6433*	6497*	6526*	6578*	6651*	6712*	6777*		
		6855*	6903*	7012*	7112*	7133*	7169	7273*								
SMXCNT	016320	7273*														
SMULL	001146	5715*	7269													
SNWTST=	000001	6335*	6357*	6382*	6409*	6433*	6497*	6526*	6578*	6651*	6712*	6777*	6855*	6903*		
		7012*	7112*													
SCCNT	015106	7270**														
SOMODE	015110	7270**														
SOVER	016304	7273*														
SPASS	001100	5715*	7165*													
SQUES	001206	5715*	7169	7272												
SRAND	017166	7280*	7603	8140	8157											
SROCHR	015524	7272*	7275													
SRODEC=	***** U	7275														
SROLIN	015610	7272*	7275													
SROCT=	***** U	7275														
SROSZ =	000024	7272*														
SREGAD	001152	5715*	6178													
SREG0	001154	5715*	7184*	8886	8888	8891	8893	8894	8895	8897	8900					
SREG1	001156	5715*	7185*	8880	8883	8892										
SREG2	001160	5715*	7186*	8885												
SREG3	001162	5715*	7187*													
SREG4	001164	5715*	7188*	8889	8892											
SREG5	001166	5715*	7189*	8880	8883											
SRESRE	016360	7274*	7275													
SSAVRE	016322	7274*	7275													
SSB2D	016460	7276*	7796	7801	7803	7806	7811	7813	7823	7826	8509	8549				
SSCOPE	016102	6177	7273*													
SSSETUP=	000007	6154*	6177	7165												
SSSTUP =	177777	6154*														
SSUPRS	016704	7278*	7796	7801	7803	7806	7811	7813	7823	7826	8509	8549				
SSVLAD	016256	7273*														
SSWR =	167000	4914*	4921	4944	5715	6177	6335	6357	6382	6409	6433	6497	6526	6578		
		6651	6712	6777	6855	6903	7012	7112	7165	7169	7273					
SSWRNK=	000000	7273														
STIMES	001176	5715*	6177*	6335*	6357*	6382*	6409*	6433*	6497*	6526*	6578*	6651*	6712*	6777*		
		6855*	6903*	7012*	7112*	7165*	7273*									
STKB	001140	5715*	7272													
STKCNT	015336	7272**														
STKINT	015346	6188	7272*													
STKQEN=	015346	7272*														
STKQIN	015340	7272**														
STKQOU	015342	7272**														
STKQSR	015344	7272*														
STKS	001136	5715*	6177	7272*												
STKSRV	015416	7272*														
STMPD	001170	5715*	7181*	8879	888J	8881	8882	8883	8886	8889	8890	8891	8893	8894		

ADC	7277	7280	7758	8179															
ADD	6184	6276	6390	6460	6508	6538	6542	6546	6548	6884	6915	6924	6934	6938	7018				
	7020	7030	7197	7198	7269	7270	7271	7277	7279	7280	7407	7429	7458	7491	7509				
	7525	7635	7694	7757	7763	7846	7865	7941	7950	7961	7998	8021	8032	8033	8101				
	8193	8402	8420	8719	8719	8732	8764	8779	8785										
ASL	6419	7208	7213	7280	7583	7960	8031	8257	8258	8259	8399	8400	8401	8405	8417				
	8418	8419	8423	8478	8614	8714	8716	8717	8758	8760	8762	8785							
ASL B	6205	6238	7193	7194	7195	7207	7271	8785											
ASR	6218	7956	8029	8108	8331	8332	8333	8488	8497	8785									
BCC	6219	7196	7214	7271	7279	8544	8785												
BOS	6185	6239	7242	8489	8498	8759	8761	8763											
BEO	6195	6223	6231	6439	6447	6453	6467	6473	6593	6606	6615	6673	6733	6746	6798				
	6811	7026	7047	7060	7071	7073	7165	7169	7200	7221	7227	7232	7236	7247	7270				
	7272	7273	7278	7279	7342	7404	7426	7437	7455	7466	7488	7497	7506	7515	7521				
	7532	7576	7602	7654	7691	7741	7794	7799	7909	7829	7845	7864	7946	8099	8123				
	8214	8238	8284	8322	8368	8430	8436	8438	8453	8455	8462	8480	8490	8499	8535				
	8539	8563	8567	8606	8608	8611	8616	8675	8677	8679	8681	8726	8785						
BGE	6392	6482	6513	7017	7022	7032	7130	7273	7279	7744	7747	8189	8426	8474	8795				
BGT	6229	6244	6281	6458	6478	6511	6535	6580	6620	6653	6687	6714	6753	6779	6818				
	6887	6918	6927	6943	6946	7042	7053	7055	7127	7165	7270	7271	7279	7578	7987				
	7890	7944	8074	8144	8164	8171	8185	8404	8422	8560	8785								
BHI	7273	7529	7948	8054	8121	8633	8653	8731											
BIC	6604	6613	6622	6671	6689	6731	6744	6755	6796	6809	6820	7165	7270	7272	7279				
	7605	7650	7957	8635	8655	8785													
BICB	8785																		
BIS	7270	7271	7277	7279	9427	8433	8441	8785											
BISB	6203	8351	8356	8785															
BIT	6208	6230	6235	6335	6357	6382	6409	6433	6438	6497	6526	6578	6592	6605	6614				
	6632	6651	6672	6712	6732	6745	6777	6797	6810	6855	6903	7012	7046	7059	7062				
	7066	7072	7074	7078	7112	7120	7169	7273	7341	7351	7361	7381	7436	7465	7496				
	7498	7526	7531	7559	7562	7565	7568	7571	7574	7601	7784	7973	7975	7979	8051				
	8114	8118	8213	8429	8479	8596	8605	8607	8610	8615	8785								
BITB	8785																		
BLE	6202	6398	6421	6462	7251	7751	7754	7848	7867	8176	8194	8785							
BLO	6165	6170	7272	7608	7720	7915	8631	8651											
BLOS	7272	7761	7978	8117	8785														
BLT	6682	6933	7269	7270	7271	7277	7279	8162	8169	8580	8593	8785							
BMI	7205	7271	7273	7516	8503	8785													
BME	6177	6181	6206	6209	6215	6236	6335	6357	6382	6409	6433	6497	6526	6578	6633				
	6651	6679	6712	6777	6855	6903	7012	7063	7067	7075	7079	7112	7121	7165	7169				
	7216	7224	7254	7261	7269	7270	7271	7272	7273	7277	7279	7280	7352	7362	7382				
	7433	7462	7493	7495	7499	7527	7561	7564	7567	7570	7573	7612	7652	7785	7938				
	7940	7958	7964	7974	7976	7980	7999	8020	8022	8039	8052	8110	8115	8119	8180				
	8319	8324	8326	8353	8364	8366	8373	8377	8381	8385	8389	8408	8440	8444	8447				
	8449	8458	8465	8514	8517	8520	8522	8537	8565	8597	8729	8773	8778	8785					
BPL	7169	7269	7270	7271	7279	7405	7427	7456	7489	7507	7522	7692	7821	8785					
BR	6156	6171	6173	6186	6197	6211	6216	6224	6226	6232	6240	6246	6261	6264	6549				
	6583	6584	6621	6635	6656	6657	6684	6688	6717	6718	6754	6782	6783	6819	6935				
	7039	7044	7082	7083	7165	7211	7223	7225	7244	7249	7263	7269	7270	7271	7272				
	7273	7277	7279	7297	7330	7336	7345	7385	7402	7424	7431	7453	7460	7486	7613				
	7689	7696	7831	7970	8048	8056	8139	8167	8174	8187	8192	8197	8215	8219	8220				
	8222	8244	8245	8246	8251	8252	8253	8262	8263	8264	8266	8268	8317	8320	8345				
	8355	8360	8375	8379	8383	8387	8391	8393	8397	8411	8415	8431	8442	8450	8471				
	8481	8487	8492	8494	8501	8515	8526	8540	8548	8561	8568	8572	8576	8582	8598				

H15

MAINDEC-11-DZRPK-B "MECHANICAL AND READ/WRITE TEST"
DZRPKB.P11 CROSS REFERENCE TABLE

MACY11 27(663) 23-JAN-76 10:16 PAGE 43-157

SEQ 0188

	8683	8727	8756	8766	8767	8785											
CLC	7279	8165	8172	8177	8190												
CLR	6157	6159	6174	6177	6179	6189	6198	6199	6207	6212	6212	6233	6267	6437	6443		
	6459	6479	6590	6600	6609	6623	6662	6666	6690	6726	6736	6739	6756	6791	6801		
	6804	6821	6873	6912	6920	6928	7013	7014	7027	7037	7045	7123	7165	7210	7233		
	7270	7271	7272	7273	7277	7279	7280	7292	7294	7323	7326	7399	7421	7450	7482		
	7483	7501	7558	7606	7678	7679	7683	7684	7686	7697	7698	7718	7830	7882	7930		
	7932	8015	8097	8239	8285	8286	8344	8354	8369	8370	8371	8469	8496	8546	8710		
	8711	8712	8754	8769	8785												
CLRB	6862	6882	7271	7272	7273	7277	7279	8785									
CMP	6164	6169	6177	6180	6391	6397	6420	6457	6461	6477	6481	6510	6512	6534	6681		
	6586	6886	6917	6926	6945	7041	7052	7054	7129	7165	7271	7272	7273	7277	7298		
	7331	7337	7614	7651	7653	7719	7743	7746	7750	7753	7760	7844	7847	7863	7866		
	7889	7914	7938	7943	8120	8175	8188	8193	8281	8403	8421	8424	8473	8559	8579		
	8592	8730	8785														
CMPB	6942	7126	7269	7272	7273	7278	7432	7461	7492	7494	7528	7577	7947	7977	8053		
	8116	8163	8170	8323	8363	8365	8372	8376	8380	8384	8388	8407	8435	8439	8443		
	8452	8519	8536	8564	8630	8632	8650	8652	8676	8678	8680	8728	8772	8785			
COM	6210	6440	7019	7058	7215	7237	7253	8785									
COMB	7969	8047	8111														
DEC	6280	6619	6685	6752	6817	6932	7021	7031	7165	7192	7246	7250	7272	7277	7278		
	7279	7611	7886	7939	8073	8143	8349	8785									
DECB	7269	7270	7279	8785													
EMT	4968																
HALT	4955	7169	7269														
INC	6204	6220	6225	6237	6456	6476	6680	6885	6916	6925	6944	7128	7165	7169	7270		
	7271	7272	7273	7277	7279	7280	7296	7579	7580	7591	7582	7748	7755	7759	7822		
	7888	7942	8184	8195	8395	8398	8409	8413	8416	8432	8445	8451	8460	8472	8558		
	8578	8689	8757	8774	8785												
INCB	6939	7125	7169	7273	8785												
ICT	4962																
JMP	4960	4962	4964	4966	6217	6335	6357	6382	6409	6433	6497	6526	6578	6581	6651		
	6654	6712	6715	6777	6780	6855	6903	7012	7112	7165	7272	7981	8057	8287	8456		
	8459	8463	8466	8476	8518	8523	8587	8595	8785								
JSR	6187	6188	6190	6191	6192	6193	6196	6251	6254	6256	6260	6342	6343	6366	6367		
	6389	6395	6415	6418	6445	6448	6451	6454	6465	6468	6471	6474	6505	6507	6537		
	6539	6541	6543	6545	6547	6582	6597	6607	6616	6618	6626	6630	6634	6636	6655		
	6663	6674	6676	6677	6693	6697	6699	6716	6723	6734	6737	6738	6747	6750	6751		
	6759	6763	6765	6781	6788	6799	6802	6803	6812	6815	6816	6824	6828	6830	6858		
	6865	6867	6868	6870	6871	6878	6883	6906	6914	6923	6937	6941	7043	7061	7065		
	7069	7077	7080	7081	7122	7124	7135	7141	7142	7143	7146	7147	7152	7153	7156		
	7157	7162	7165	7169	7182	7264	7269	7272	7276	7329	7335	7369	7400	7407	7422		
	7429	7434	7438	7451	7458	7463	7467	7484	7491	7502	7509	7511	7517	7525	7533		
	7603	7687	7694	7796	7801	7803	7806	7811	7813	7819	7823	7826	7881	7891	7902		
	7916	7928	7951	7991	8000	8012	8024	8072	8094	8140	8157	8218	8243	8250	8261		
	8312	8313	8343	8348	8350	8357	8392	8396	8410	8414	8468	8475	8509	8512	8524		
	8533	8541	8545	8549	8554	8555	8570	8571	8574	8575	8586	8590	8594	8599	8617		
	8682	8713	8720	8723	8755	8765	8770	8785									
MOV	6155	6158	6160	6161	6162	6163	6166	6167	6168	6172	6175	6177	6178	6182	6183		
	6200	6214	6221	6227	6234	6242	6247	6248	6249	6250	6252	6253	6255	6257	6259		
	6262	6265	6266	6277	6335	6336	6339	6340	6341	6357	6362	6363	6364	6365	6382		
	6385	6386	6387	6393	6409	6412	6413	6414	6416	6417	6433	6436	6441	6442	6444		
	6450	6463	6464	6470	6497	6500	6501	6502	6503	6504	6506	6526	6529	6530	6531		
	6532	6536	6578	6585	6586	6588	6589	6591	6594	6595	6596	6598	6599	6601	6602		

	6607	6610	6611	6615	6624	6626	6629	6631	6651	6658	6659	6660	6661	6664	6665
	6667	6668	6669	6674	6683	6691	6693	6696	6698	6712	6719	6720	6721	6722	6724
	6725	6727	6728	6729	6734	6740	6741	6742	6747	6749	6757	6759	6762	6764	6777
	6784	6785	6786	6787	6789	6790	6792	6793	6794	6799	6805	6806	6807	6812	6814
	6822	6824	6827	6829	6855	6856	6857	6859	6860	6863	6864	6866	6869	6872	6874
	6875	6876	6879	6880	6881	6903	6904	6905	6907	6908	6910	6911	6919	6921	6930
	6931	6936	6940	7012	7015	7024	7028	7035	7036	7038	7048	7049	7051	7056	7057
	7064	7068	7076	7112	7113	7114	7115	7116	7117	7118	7119	7131	7132	7133	7134
	7135	7137	7138	7139	7140	7144	7145	7148	7149	7150	7151	7154	7155	7158	7159
	7160	7161	7165	7169	7184	7185	7186	7187	7188	7189	7191	7199	7206	7209	7212
	7226	7231	7234	7235	7243	7245	7256	7269	7270	7271	7272	7273	7274	7275	7276
	7277	7278	7279	7280	7293	7299	7320	7321	7322	7324	7325	7328	7332	7334	7338
	7339	7340	7343	7344	7346	7347	7353	7354	7355	7356	7363	7364	7365	7368	7383
	7384	7386	7387	7406	7407	7408	7428	7429	7430	7457	7458	7459	7490	7491	7508
	7509	7523	7524	7525	7530	7554	7555	7556	7557	7584	7585	7586	7600	7604	7609
	7628	7629	7630	7631	7632	7633	7636	7637	7648	7649	7655	7660	7661	7664	7655
	7680	7681	7682	7685	7693	7694	7716	7717	7721	7722	7723	7739	7742	7745	7752
	7762	7783	7786	7787	7788	7789	7790	7796	7801	7803	7806	7811	7813	7816	7817
	7818	7826	7828	7849	7868	7883	7884	7885	7903	7904	7905	7906	7907	7908	7909
	7910	7911	7912	7913	7931	7933	7934	7949	7953	7955	7959	7962	7965	7967	7971
	7992	7993	7994	7997	8013	8014	8016	8026	8029	8034	8035	8036	8037	8040	8043
	8044	8046	8049	8055	8070	8071	8087	8088	8089	8090	8091	8092	8093	8095	8096
	8100	8102	8103	8104	8105	8106	8124	8137	8138	8141	8142	8145	8160	8181	8200
	8216	8217	8221	8240	8241	8242	8247	8248	8249	8254	8255	8256	8265	8267	8269
	8270	8271	8273	9274	8275	8276	8277	8278	8280	8306	8309	8311	8316	8329	8347
	8350	8362	8374	8378	8382	8386	8390	8394	8406	8412	8477	8492	8484	8485	8486
	8493	8495	8505	8509	8511	8524	8525	8530	8532	8542	8547	8549	8551	8553	8556
	8569	8573	8577	8581	8584	8588	8600	8601	8602	8603	8604	8609	8690	8707	8708
	8709	8715	8734	8736	8737	8738	8753	8768	8780	8781	8785				
MOV8	6177	6253	6258	6337	6338	6358	6359	6360	6361	6383	6384	6410	6411	6434	6435
	6498	6499	6527	6528	6587	6861	6877	6909	6913	6922	6929	7034	7050	7169	7181
	7190	7239	7240	7269	7270	7271	7272	7273	7275	7277	7407	7429	7458	7491	7500
	7509	7510	7525	7656	7657	7658	7659	7694	7966	8041	8042	8158	8159	8198	8199
	8634	8654	8785												
NEG	7023	7033	7270	7271	7279	8186									
NOP	7165	7504	7513	7519											
RESET	6176	7165													
ROL	6268	6269	6270	6271	6272	6273	6274	6275	7270	7273	7279	7280	7820	8785	
ROLB	8785														
ROR	7241	8178	8191	8543											
RORB	8166	8173													
RTI	7169	7269	7270	7271	7272	7273	7274	7279	7370	7705	8785				
RTS	7265	7269	7272	7275	7276	7277	7278	7280	7300	7348	7358	7366	7388	7409	7440
	7469	7535	7587	7615	7638	7666	7699	7725	7764	7833	7851	7870	7892	7917	7952
	8001	8025	8075	8125	8146	8201	8223	8279	8334	8358	9583	8618	8637	8657	8691
	8739	8782	8785												
SBC	7277	7279													
SUB	6396	6480	6509	6533	6540	6544	7016	7029	7169	7183	7204	7271	7277	7607	7929
	7954	8020	8027	8030	8098	8107	8785								
SWAB	8182	8196	8260	8330	8785										
TRAP	7275														
TST	6194	6222	6228	6446	6452	6466	6472	6579	6625	6652	6678	6692	6713	6758	6778
	6823	7025	7040	7070	7165	7169	7220	7260	7269	7270	7271	7272	7273	7275	7279
	7295	7327	7333	7403	7425	7454	7487	7505	7514	7520	7610	7662	7690	7695	7740

K15

MAINDEC-11-DZRPK-B "MECHANICAL AND READ/WRITE TEST"
DZRPKB.P1: CROSS REFERENCE TABLE

MACY11 27(663) 23-JAN-76 10:16 PAGE 43-160

SEQ 0191

.IIF	4921	4944	4945	4946	4947	4948	4949	4950	4951	4952	4955	5715	6177	6221	6262
	7165	7169	7243	7245	7269	7272	7273	7275	7796	7801	7803	7806	7811	7813	7823
.IRP	5715	6154	6335	6357	6382	6409	6433	6497	6526	6578	6651	6712	6777	6855	6903
	7012	7112	7165	7169	7271	7274	7279	7280	7935	7995	8017				
.LIST	2	3664	4912	4919	4955	4968	5715	5904	6148	6154	6186	6211	6216	6224	6232
	6261	6264	6287	6312	6316	6324	6335	6357	6382	6409	6433	6497	6526	6555	6565
	6578	6651	6712	6777	6835	6841	6855	6903	7012	7112	7165	7211	7225	7272	7273
.MACRO	39	81	164	300	478	535	684	738	825	856	904	916	956	985	1018
	1031	1052	1065	1094	1140	1177	1224	1257	1287	1343	1351	1399	1575	1773	1928
	2016	2094	2191	2269	2354	2568	2660	2736	2837	2964	3021	3083	3201	3239	3303
	3401	3486	3525	3588	3626	3665	4944	4998	5715	6002	6010	6014	6020	6025	6035
	6051	6060	6067	6071	6076	6087	6105	6114	6121	6130	6140	6326	6346	6370	6401
	6424	6485	6516	6566	6639	6702	6767	6843	6890	6951	7088	7275	8785		
.MCALL	4916	4917	4918	4968											
.MLIST	1	3	4911	4913	4955	4968	5715	5904	6001	6154	6186	6211	6216	6224	6232
	6261	6264	6287	6312	6316	6324	6335	6357	6382	6409	6433	6497	6526	6555	6565
	6578	6651	6712	6777	6835	6841	6855	6903	7012	7112	7165	7211	7225	7272	7273
	7275	8215	8345	8360	8481	8548									
.PAGE	4969	5715	6149	6283	6551	6831	6948	7085	7164	7166	8783				
.REPT	4955	5715	5904	6287	6312	6316	6324	6555	6565	6835	6841				
.SBTTL	4923	4944	4955	4957	4968	5715	6151	6285	6314	6335	6357	6382	6409	6433	6497
	6525	6553	6578	6651	6712	6777	6833	6855	6903	6949	7012	7086	7112	7165	7168
	7169	7172	7269	7270	7271	7272	7273	7274	7275	7276	7277	7278	7279	7280	7283
	7303	7373	7391	7412	7443	7473	7538	7590	7619	7641	7669	7702	7708	7728	7767
	7836	7855	7873	7895	7920	7984	8004	8060	8078	8126	8149	8204	8226	8290	8337
	8620	8640	8660	8694	8742	8785	8787	8823	8839	8853	8877	8904			
.TITLE	4921														
.WORD	4955	5715	6278	6449	6455	6469	6475	7165	7203	7219	7230	7258	7270	7272	7276
	7277	7278	7280	7439	7468	7534	7560	7563	7566	7569	7572	7575	7792	8308	8470
	8507	8785	8879	8880	8881	8882	8883	8884	8885	8886	8887	8888	8889	8890	8891
	8892	8893	8894	8895	8896	8897	8898	8899	8900	8901	8902	8906	8910	8914	8918
	8922	8926	8930	8933	8937	8940	8944	8948	8952	8955	8959	8963	8967	8971	8974
	8978	8981	8984	8988	8991	8994									

ERRORS DETECTED: 0

*DZRPKB, DZRPKB/CRF=SYSMAC.SMA, RPO4.009, DZRPKB
RUN-TIME: 70 83 9 SECONDS
CORE USED: 39K

