

# RM03

EXTENDED DRIVE TEST  
MD-11-DZRMF-A

EP-DZRMF-A-DL-A

OCT 1977

COPYRIGHT © 1977

**digital**

FICHE 1 OF 2

MADE IN USA

The microfiche card displays a grid of 100 frames of data, organized into 10 rows and 10 columns. Each frame contains a different set of data, likely test results or system logs, presented in a structured, tabular format. The data is printed in white on a dark background. The frames contain various types of information, including numerical values, text labels, and possibly small diagrams or charts. The overall appearance is that of a dense, organized collection of data points.

# RM03

EXTENDED DRIVE TEST  
MD-11-DZRMF-A

EP-DZRMF-A-DL-A  
COPYRIGHT © 1977  
FICHE 2 OF 2

OCT 1977  
**digital**  
MADE IN USA

This vertical strip contains 12 small, illegible data tables or charts, likely representing test results for different parameters or time intervals. The content is too faint to transcribe accurately.

B01

EOF1DZRMFASBQ411

00010000

770804

POP10 BIE N T I F 038DR1DZMBABEQ

00010000

770804  
SEQ 0001

PRODUCT CODE: MAINDEC-11-DZRMF-A-D  
PRODUCT NAME: RM03 EXTENDED DRIVE TEST  
DATE CREATED: AUGUST 1977  
MAINTAINER: DIAGNOSTIC ENGINEERING  
AUTHOR: H. BLACKSTONE

COPYRIGHT (C) 1977 DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

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

ACTUAL DISTRIBUTION OF THE SOFTWARE DESCRIBED IN THIS DOCUMENT WILL BE SUBJECT TO TERMS AND CONDITIONS TO BE ANNOUNCED ON SOME FUTURE DATE BY DIGITAL EQUIPMENT CORPORATION.

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

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

CONTENTS

1. ABSTRACT
2. REQUIREMENTS
  - 2.1 EQUIPMENT
  - 2.2 PRELIMINARY PROGRAMS
  - 2.3 MEDIA
3. LOADING PROCEDURE
4. STARTING PROCEDURE
  - 4.1 STARTING ADDRESSES
  - 4.2 OPERATOR ACTION
  - 4.3 PROGRAM ACTION
    - 4.3.1 CONTROL SWITCH SELECTION
    - 4.3.2 RH70 ADDRESS SELECTION
    - 4.3.3 DRIVE AND PARAMETER SELECTION
5. OPERATING PROCEDURE
  - 5.1 OPERATIONAL SWITCH SETTINGS
  - 5.2 CONTROL SWITCH SETTINGS
6. ERRORS
  - 6.1 ERROR TYPES
  - 6.2 ERROR RECOVERY
7. RESTRICTIONS
8. MISCELLANEOUS
  - 8.1 EXECUTION TIME
  - 8.2 STACK POINTER
  - 8.3 TIMING TEST (TESTS 12 - 15) PRINTOUTS
  - 8.4 END OF TEST
9. PROGRAM DESCRIPTION
10. RH70/RM03 DRIVER MODULE
11. PROGRAM LISTING

## 1. ABSTRACT

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

## 2. REQUIREMENTS

## 2.1 EQUIPMENT

PDP-11 PROCESSOR  
 16K MEMORY  
 TELETYPE  
 PROGRAM LOADING DEVICE  
 KW11-L OR KW11-P (THE KW11-P IS REQUIRED FOR THE TIMING TESTS)  
 RH70 WITH 1 - 8 RMD3 DISK DRIVES

## 2.2 PRELIMINARY PROGRAMS

RMD3 DISKLESS CONTROLLER TEST DZRMJ-A

RMD3 FUNCTIONAL TEST I DZRMJC-A

RMD3 FUNCTIONAL TEST II DZRMJD-A

RMD3 FUNCTIONAL TEST III DZRMJE-A

## 2.3 MEDIA

THE PROGRAM REQUIRES THAT EACH DRIVE TO BE TESTED HAS A FORMATTED DISK PACK. THE PACK MAY BE FORMATTED IN EITHER 16-BIT OR 18-BIT MODE, DEPENDING ON THE TESTING REQUIREMENTS. NOTE THAT THE PROGRAM WILL NOT TEST A MIXTURE OF DRIVES WITH BOTH 16 AND 18 BIT MODE PACKS.

## 3. LOADING PROCEDURE

THE PROGRAM MAY BE LOADED FROM PAPER TAPE USING THE ABSOLUTE LOADER OR IT MAY BE LOADED FROM THE APPROPRIATE 'XXDP' MEDIA USING THE ASSOCIATED LOADER. THE PROGRAM MAY BE INCLUDED IN AN 'XXDP' CHAIN. IF THE PROGRAM IS BEING RUN ON A PROCESSOR WITH 16K, THE 'XXDP' LOADER WILL NOT BE PRESERVED. THE PROGRAM MUST BE RUN ON A SYSTEM WITH 20K OR MORE TO PRESERVE THE 'XXDP' LOADER. THE 'ABSOLUTE' LOADER WILL BE PRESERVED IN A 16K SYSTEM, HOWEVER.

## 4. STARTING PROCEDURE

## 4.1 STARTING ADDRESSES

200 NORMAL STARTING ADDRESS  
 204 SELECT OPERATING PARAMETERS  
 210 SELECT RM70 ADDRESSES  
 214 COMBINATION OF 204 AND 210

NOTE: STARTING ADDRESSES 210 AND 214 ARE AVAILABLE WHEN THE PROGRAM IS INITIALLY STARTED; THESE STARTING ADDRESSES ARE TREATED AS ADDRESSES 200 OR 204 RESPECTIVELY ON RESTARTS.

## 4.2 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 5.)
6. PRESS START.
7. THE PROGRAM WILL TYPEOUT THE STATUS OF THE DRIVES ATTACHED TO THE SELECTED MASSBUS SUBSYSTEM. TO INHIBIT THIS TYPEOUT, DO NOT RESTART THE PROGRAM FROM ANY OF THE STARTING ADDRESSES; INSTEAD TYPE A 'CONTROL C' ON THE KEYBOARD TO RETURN THE PROGRAM TO COMMAND ENTRY MODE.

## 4.3 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

THE 'PERIOD PERIOD' TERMINATOR IS TYPED TO INDICATE THE END OF TEST PARAMETER MODIFICATION AND TO SIGNAL THE START OF TEST EXECUTION.

<,)<CR> COMMA

THE COMMA IS USED AS A SEPARATOR BETWEEN DRIVE NUMBERS AND TEST NUMBERS.

</> SLASH

A MODIFICATION INDICATOR: IF A SLASH FOLLOWS A TEST

# F01

NUMBER, THE PROGRAM WILL OPEN THAT TEST FOR PARAMETER MODIFICATION.

SEQ 0005

<↑U> CONTROL-U  
DELETE 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.

## 4.3.1 CONTROL SWITCH 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".

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..
```

## 4.3.2 RH70 ADDRESS SELECTION

STARTING THE PROGRAM AT 200 WILL RESULT IN AUTOMATIC SELECT OF THE DEFAULT VALUES OF BUS ADDRESS (RMCS1), VECTOR ADDRESS, AND PRIORITY LEVEL OF THE RH70. IF THE DEFAULT VALUE OF THE BUS ADDRESS DOES NOT RESPOND (TIMES OUT) WHEN ADDRESSED, AN ERROR IS REPORTED. 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 ADDRESS OF THE RH70 AND THE VECTOR ADDRESS. STARTING ADDRESSES 210(8) AND 214(8) ARE TREATED AS ADDRESSES 200(8) OR 204(8) RESPECTIVELY.

ADDRESS SELECTION EXAMPLES

### EXAMPLE #1

RMCSI=176700 / 177200.

EXAMPLE #2

RMCSI=176700 / 176300<CR>  
RHVEC=254 / 260<CR>  
RHPRIO=5 / 6.

EXAMPLE #3

RMCSI=176700<CR>  
RHVEC=254 / 260.

EXAMPLE #4

RH70/RM03 FAILED TO RESPOND TO ADDRESSING  
RMCSI ERR PC  
176300 XXXXXX  
RMCSI=176300 / 176700.

EXAMPLE #5

RMCSI=176700 / 1776\67\6300<CR>  
RHVEC=254<CR>  
RHPRIO=5<CR>  
RMCSI=176300.

#### 4.3.3 DRIVE AND PARAMETER SELECTION

STARTING THE PROGRAM AT 200 OR 210 WILL RESULT IN AUTOMATIC SELECTION OF THE DRIVES TO TEST AND THE TESTS TO RUN.

STARTING THE PROGRAM AT 204 OR 214 ALLOWS THE OPERATOR TO SELECT THE DRIVE(S) TO BE TESTED, THE TESTS TO BE EXECUTED, AND THE PARAMETERS TO USE.

##### 4.3.3.1 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 (TESTS 0 - 10)
*"T"	ALL TIMING TESTS (TESTS 12 - 15)
*"A"	ALL ADDRESS TESTS (TESTS 16 - 17)
*"D"	THE DATA TEST (TEST 20)
*"E"	THE EXERCISER (TEST 21)



# H01

SEQ 0007

\* USED BY THE OPERATOR TO SELECT TEST GROUPS  
NOTE: ALL NUMBERS WILL BE IN DECIMAL EXCEPT FOR THE PATTERN  
(PAT) AND WORDS (WOX) SELECTION. "PAT" WILL BE SELECTED  
BY A BIT (I.E. 001000(8)=PATTERN 9) AND "WOX" 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 MODIFIED, THE REMAINING TESTS WILL BE UNCHANGED.

WHEN THE PROGRAM IS STARTED FROM LOCATION 200 OR 210, TESTS 0-10,  
12-20 WILL BE RUN USING ALL AVAILABLE, ONLINE DRIVES. IF THE OPERATOR  
WISHES TO SELECT THE DRIVES TO BE TESTED, THE TESTS TO BE PERFORMED,  
OR THE PARAMETERS TO BE USED, THE CONVERSATION MODE MAY BE ENTERED  
BY TYPING A 'CONTROL C' OR BY STARTING THE PROGRAM FROM EITHER  
LOCATION 204 OR 214.

THE PROGRAM WILL THEN RESPOND WITH:

DRIVE(S)=

THE FOLLOWING EXAMPLES ASSUME THAT THE OPERATOR IS 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<CR>' WHICH SAYS 'THIS IS THE END OF DRIVE ENTRY'.  
THE PROGRAM WILL THEN REQUEST TEST NUMBERS.

THE TRANSACTION APPEARS AS FOLLOWS:

DRIVE(S)=3<CR>  
TEST=

THE OPERATOR MAY NOW ENTER DESIRED TEST NUMBERS. IN THE EXAMPLE,  
HE WANTS TESTS 2 THRU 7 AND TEST 11 SO HE TYPES 2-7<,> (THE 'COMMA'  
SEPARATES ENTRIES), 11<.><CR> ('PERIOD' 'CARRIAGE RETURN' -  
END OF CHANGES, START TEST EXECUTION.)

IT NOW LOOKS LIKE THIS

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

IN THE NEXT EXAMPLE, IT IS ASSUMED THAT THE OPERATOR WISHES TO TEST  
DRIVE 4 AND TO RUN TESTS 1 AND 3 THRU 11, MODIFYING THE PARAMETERS  
FOR TESTS 3 AND 10.

THE TRANSACTION WOULD BE AS FOLLOWS:

DRIVE(S)=4<CR>  
TEST=

THE OPERATOR NOW ENTERS THE TEST NUMBERS. THE TRANSACTION IS  
GIVEN BELOW:

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

NOTICE THIS SAYS SELECT TEST 1, CONTINUE(<,>); SELECT TEST 3, OPEN(</>);  
SELECT TESTS 4-7, CONTINUE(<,>); SELECT TEST 10, OPEN(</>); SELECT TEST  
11, END OF INPUT (<.>).

THE PROGRAM SCANS THE TEST NUMBER INPUT AND DETERMINES THAT THE  
PARAMETERS FOR TEST 3 AND TEST 10 ARE TO BE CHANGED. THE OTHER  
TESTS WILL NOT BE ALTERED.

(THE ENTIRE TRANSACTION IS REPEATED FOR CLARITY)

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

THE NEW VALUE FOR 'R' MAY BE ENTERED. TERMINATING THE ENTRY WITH  
A (<.>) (PERIOD) WILL TERMINATE THE CHANGES FOR THIS TEST; TYPING A  
<CR> OR TERMINATING THE ENTRY WITH A <CR> WILL CAUSE THE PROGRAM  
TO MOVE TO 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
```

IF THE OPERATOR DOES NOT WISH TO CHANGE 'FC', THE FOLLOWING OCCURS:

```
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=822 /
```

THE PROGRAM RESPONDS WITH THE PREVIOUSLY ASSIGNED PARAMETER FOR LAST  
CYLINDER ADDRESS IN THIS CASE USING 822 AS THE EXAMPLE. THIS IS  
WHAT THE OPERATOR INTENDED TO MODIFY AND IS WHY TEST 3 WAS  
OPENED. TO CHANGE THE VALUE TO '20', THE NEW VALUE IS TYPED  
FOLLOWED BY A 'PERIOD' TERMINATOR (<.><CR>).

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= 822 / 20.<CR>
TEST 10
R=1 /
```

THE PROGRAM HAS LOADED TEST 3 WITH ITS NEW PARAMETERS AND THE  
PROGRAM IS WAITING FOR CHANGES TO TEST 10'S PARAMETERS.

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

```

FC=0 / <CR>
LC= 822 / 20.<CR>
TEST 10
R=1 / 10.<CR>

```

THE OPERATOR TYPES THE NEW VALUE (10) AND TERMINATES THE ENTRY WITH A 'PERIOD' 'CARRIAGE RETURN'.

THE PROGRAM NOW LOADS TEST 10 WITH THE NEW PARAMETERS (TEST 11 RETAINS THE PREVIOUSLY ASSIGNED PARAMETERS) AND RESPONDS WITH:

```
DRIVE(S)=
```

SINCE THE USER DID NOT END THE CONVERSATION MODE WITH A 'PERIOD PERIOD', 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 PROGRAM, A 'PERIOD PERIOD' MUST BE TYPED.

IF A SINGLE 'PERIOD' IS TYPED WHILE DRIVE OR TEST NUMBERS ARE BEING ENTERED, THE PROGRAM WILL START EXECUTION IMMEDIATELY. A 'PERIOD PERIOD' MUST BE TYPED BEFORE THE PROGRAM WILL EXIT TEST PARAMETER CHANGE MODE TO GO TO EXECUTION.

#### 4.3.3.2 DRIVE AND PARAMETER SELECTION EXAMPLES

##### 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 AND MOVE TO NEXT PARAMETER
  FC=0 / 10.<CR>       ;SET 'FC' CYLINDER ADDRESS TO 10, END CHANGES
                       ;TO TEST 6.

```

```

TEST 7
  R=1 / 50<CR>        ;50 ITERATIONS, MOVE TO NEXT PARAMETER
  FC=0 / <CR>         ;DO NOT CHANGE 'FC' CYLINDER ADDRESS BUT CONTINUE
  LC=822 / 50.<CR>    ;TEST 10 IS STILL PENDING AND WILL BE
                       ;RETAIN ITS PRESENT PARAMETERS.

```

##### EXAMPLE #4

# K01

SEQ 0010

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

## EXAMPLE #5

```
-----
DRIVE=1<CR>
TEST=S/D<CR>          ;RUN ALL SEEK TESTS (OPEN FOR CHANGES) AND
TEST 0                ;THE DATA TEST (WITH DEFAULT PARAMETERS).
  R=10 / <CR>         ;RUN WITH 10 ITERATIONS
  FC=0 / 10.<CR>      ;CHANGE FIRST CYLINDER ADDRESS
                      ;AND START EXECUTION
                      ;TESTS 1 - 10 WILL RETAIN THEIR PREVIOUSLY
                      ;ASSIGNED PARAMETERS.
```

## EXAMPLE #6

```
-----
DRIVE=1<CR>
TEST=S/<CR>           ;OPEN THE SEEK TESTS (TESTS 0-10)
TEST 0
  R=10 / 100.<CR>     ;CHANGE TO 100 ITERATIONS, TO TO THE 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, GO TO NEXT PARAMETER
  FC=0 / 50<CR>       ;CHANGE 'FC' TO 50, GO TO NEXT PARAMETER
  LC=822 / 51.<CR>    ;CHANGE 'LC' TO 51, GO TO THE NEXT TEST
TEST 3
  R=1.<CR>             ;MOVE TO NEXT TEST
TEST 4
  R=1.<CR>             ;USE TEST 4'S PARAMETERS AND START PROGRAM EXECUTION
```

## EXAMPLE #7

```
-----
DRIVE=1<CR>
TEST=D/<CR>           ;SELECT AND OPEN THE DATA TEST
TEST 20
  R=1 / 1000<CR>      ;DO 1000 ITERATION OF TEST PATTERN
  FC=0 / 10<CR>       ;#8 ON CYLINDER 10, TRACK 2, SECTOR 4
  LC=822 / 10<CR>
  IC=64 / 0<CR>
  FT=0 / 2<CR>
  LT=4 / 2<CR>
  IT=1 / <CR>
  FS=0 / 4<CR>
  LS=32 / 4<CR>
  PAT=177777 / 400.<CR> ;RUN WITH PATTERN #8
```

## EXAMPLE #8

```
-----
DRIVE=1<CR>           ;USE THE SAME PARAMETERS AS IN EXAMPLE
TEST=D/<CR>           ;#7, BUT ALSO SPECIFY A DATA PATTERN (PAT #0).
TEST 20
  R=1000 / <CR>
  FC=10 / <CR>
```

```

LC=10 / <CR>
IC=0 / <CR>
FT=2 / <CR>
LT=2 / <CR>
IT=1 / <CR>
FS=4 / <CR>
LS=4 / <CR>
PAT=000400 / 401<CR> ;RUN WITH PATTERNS #8 & #0 (0=OPERATOR INPUT)
WD1=165555 / 125252<CR> ;FIRST WORD OF PATTERN 0
WD2=133333 / 52525..<CR> ;SECOND WORD OF PATTERN 0
; (...) START EXECUTION

```

## EXAMPLE #9

-----

```

DRIVE=0,1,4<CR> ;TEST DRIVES 0,1, AND 4 IN SEQUENCE
TEST=0-5/<CR> ;CHANGE TEST 5
TEST 0
R=10 / <CR>
FC=0 / <CR>
LC=822 / 1..<CR> ;CHANGE LAST CYLINDER FROM 822 TO 1
;START PROGRAM EXECUTION.

```

## 5. SWITCH SETTINGS

-----

## 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<12>=1...TYPE OUT TEST NUMBER FOR EACH TEST
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 12-15)
SW<05>=1...REPORT ONE ERROR PER SECTOR (TESTS 16 & 17)
SW<04>=1...INHIBIT WRITES (TEST 20)
SW<03>=1...INHIBIT WRITE CHECKS (TEST 20)
SW<02>=1...INHIBIT READ AND SOFTWARE COMPARES (TEST 20)
SW<01>=1...INHIBIT SOFTWARE COMPARES (TEST 20)
SW<00>=1...PERFORM READ AFTER WRITE CHECK ERROR (TEST 20)

```

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E. AN 11/34) THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE 'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176 (8). THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM IS AT A HIGHER PRIORITY PROCESSING AN RMO3 INTERRUPT. THE

'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

SEQ 0012

'SWR = NNNNNN    NEW ='

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

ON PROCESSORS WITH HARDWARE SWITCH REGISTERS, THE 'SOFTWARE' SWITCH REGISTER MAY BE USED. IF THE PROGRAM FINDS ALL 16 SWITCHES IN THE 'UP' POSITION, ALL SWITCH REGISTER REFERENCES WILL BE TO THE 'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST BE FOLLOWED.

## 5.2 CONTROL SWITCH SETTINGS

THE CONTROL SWITCH SETTINGS ARE ENTERED THROUGH THE KEYBOARD.

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 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)  
 C.SWR<00>=0...OPERATE IN 22 SECTOR (16 BIT) MODE  
 =1...OPERATE IN 20 SECTOR (18 BIT) MODE

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

REFER TO 4.3.1 FOR C.SWR SELECTION

## 5. ERRORS

THERE ARE ANUMBER 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 TIMEOUT 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.

SEQ 0013

## 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 RM70/RM03 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 PROGRAM DPB.

### 6.1.2 NON-FATAL ERRORS

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 ERRORS

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

THE PROGRAM WILL TEST THE DRIVES IN EITHER 16 BIT MODE OR IN 18 BIT MODE DEPENDING ON THE SETTING OF 'S.SWR<00>'. IF 'C.SWR<00>' IS 0, ALL OF THE DRIVES WILL BE TESTED IN 16 BIT MODE; IF 'C.SWR<00>' IS 1, ALL OF THE DRIVES WILL BE TESTED IN 18 BIT MODE. THE PROGRAM HAS NO PROVISIONS FOR TESTING DRIVES WITH INTERMIXED PACKS OR TESTING BOTH 16 BIT MODE AND 18 BIT MODE DRIVES ON THE SAME SYSTEM.

BEFORE THE PROGRAM IS STARTED, PROPERLY FORMATTED PACKS MUST BE MOUNTED ON THE DRIVES WHICH WILL BE TESTED. THE PROGRAM ASSUMES A PROPERLY FORMATTED PACK. THE FORMAT OF THE PACK IS NOT ALTERED BY THE PROGRAM.

8. MISCELLANEOUS

## 8.1 EXECUTION TIME

THE PROGRAM REQUIRES APPROXIMATELY 22 MINUTES TO MAKE ONE PASS PER DRIVE USING AN 11/70 CPU. THIS ASSUMES THE DEFAULT TEST SEQUENCE (TESTS 0-10, 12-21) AND DEFAULT TEST PARAMETERS.

## 8.2 STACK POINTER

THE STACK POINTER IS INITIALLY SET TO 1100.

## 8.3 TIMING TESTS (TESTS 12-15) 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.

## 8.3.1 TIMING TOLERANCES

## 1. TEST 12 -- ROTATIONAL SPEED TIMES

60 HZ  
MINIMUM=16260 US  
MAXIMUM=17300 US  
NOMINAL=16670 US

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



## 2. TEST 13 -- ONE CYLINDER SEEK TIMES

MAXIMUM=8000 US  
NOMINAL=6000 US

## 3. TEST 14 -- ACCESS TIME MEASUREMENT

MAXIMUM=32000 US  
NOMINAL=30000 US

## 4. TEST 15 -- MAXIMUM SEEK TIMES

MAXIMUM=57000 US  
NOMINAL=55000 US

## 8.3 2. TIMING TESTS PRINTOUT EXAMPLES

## EXAMPLE #1

## ROTATIONAL SPEED TIMES

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

## ONE CYLINDER SEEK TIMES

\* FORWARD  
MIN=5350 US  
MAX=6920 US  
AVG=5550 US 822 SEEKS TIMED  
\* REVERSE  
MIN=5140 US  
MAX=5960 US  
AVG=5430 US 822 SEEKS TIMED

## ACCESS TIME MEASUREMENTS

\* FORWARD  
MIN=27770 US  
MAX=28640 US  
AVG=28230 US 128 SEEKS TIMED  
\* REVERSE  
MIN=27990 US  
MAX=28550 US  
AVG=28220 US 128 SEEKS TIMED

## MAXIMUM SEEK TIMES

\* FORWARD  
MIN=56990 US  
MAX=57980 US  
AVG=57010 US 128 SEEKS TIMED  
\* REVERSE  
MIN=55120 US  
MAX=57650 US  
AVG=56340 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

\* FORWARD  
 MIN=5470 US  
 MAX=8940 US 3 ABOVE THE MAXIMUM OF 8000 US  
 AVG=5830 US 822 SEEKS TIMED  
 \* REVERSE  
 MIN=5040 US  
 MAX=5970 US  
 AVG=5330 US 822 SEEKS TIMED

## ACCESS TIME MEASUREMENTS

\* FORWARD  
 MIN=29730 US  
 MAX=32620 US 73 ABOVE THE MAXIMUM OF 32000 US  
 AVG=30320 US 128 SEEKS TIMED  
 \* REVERSE  
 MIN=28620  
 MAX=32230 US 128 ABOVE THE MAXIMUM OF 32000 US  
 AVG=32800 US 128 SEEKS TIMED

## MAXIMUM SEEK TIMES

\* FORWARD  
 MIN=57510 US  
 MAX=57240 US 128 ABOVE THE MAXIMUM OF 57000 US  
 AVG=57020 US 128 SEEKS TIMED  
 \* REVERSE  
 MIN=57050 US  
 MAX=57550 US 128 ABOVE THE MAXIMUM OF 57000 US  
 AVG=57210 US 128 SEEKS TIMED

## 8.4 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.

## 9. PROGRAM DESCRIPTION

-----  
 THIS PROGRAM CONTAINS NINETEEN TESTS NUMBERED 0-22 IN OCTAL. TESTS 0-7 & 11 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. TESTS 12-15 WILL MEASURE THE ROTATIONAL SPEED, THE ONE CYLINDER SEEK, THE ACCESS TIME, AND THE MAXIMUM SEEK TIMES TO ENSURE THEY ARE ALL WITHIN THE TOLERANCES ALLOWED. TEST 16 AND 17 ENSURES THE SECTOR AND TRACK ADDRESSING CIRCUITRY WORKS PROPERLY. TEST 20 VERIFIES THE DATA STORAGE AND RETRIEVAL CAPABILITIES ARE FUNCTIONAL. AND TEST 21 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 (TESTS 0-10,12-20) WILL BE PERFORMED ON EACH DRIVE BEFORE MOVING TO THE NEXT DRIVE IN SEQUENCE. THE ADDRESS OF EACH 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 SECTIONS FOR DETAILED DESCRIPTIONS OF EACH TEST.

### 9.1 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 ENSURE NO ERRORS OCCURRED.

THE PARAMETERS USED BY THE TEST ARE GIVEN BELOW:

R	-	200
LC	-	822
FT	-	0
FS	-	0

### 9.2 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 ENSURE PROPER OPERATION.

THE PARAMETERS USED BY THE TEST ARE GIVEN BELOW:

R	-	100
FC	-	0
LC	-	256
IC	-	0
FT	-	0
LT	-	0
FS	-	0
LS	-	0

### 9.3 TEST 2 - INCREMENTAL SEEK TEST

THIS TEST WILL COMMAND FORWARD SEEK CYCLES TO ADVANCE THE CYLINDER ADDRESS FROM "FC" TO "LC" BY THE INCREMENT "IC". WHEN THE RESULTANT CYLINDER ADDRESS (NC) EXCEEDS "LC" REVERSE SEEK CYCLES ARE INITIATED; STARTING AT THE LAST LEGAL "NC" AND DECREMENTING BY "IC" UNTIL "NC" IS LESS THAN "FC". AT THE COMPLETION OF EACH SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO ENSURE PROPER OPERATION.

THE PARAMETERS USED BY THE TEST ARE GIVEN BELOW:

R	-	1
FC	-	0
LC	-	822
IC	-	1
FT	-	0
FS	-	0

## 9.4 TEST 3 - STEPPING SEEK TEST

THIS TEST WILL COMMAND SEEK CYCLES TO CYLINDER 0, 1, 2, 4, 8, 16, 32, 64, 128, AND 256. AT THE COMPLETION OF EACH SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO VERIFY PROPER OPERATION.

THE PARAMETERS USED BY THE TEST ARE GIVEN BELOW:

R	-	8
FC	-	0
LC	-	256
IC	-	1
FT	-	0
FS	-	0

## 9.5 TEST 4 - OSCILLATING SEEK TEST

THIS TEST WILL COMMAND SEEK CYCLES FROM "FC" TO "NC" AND BACK TO "FC". "NC" STARTS AT "FC" AND INCREMENTS BY "IC" UP TO CYLINDER "LC", THEN IS DECREMENTED BY "IC" BACK TO CYLINDER "FC". AT THE COMPLETION OF EVERY SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO ENSURE PROPER OPERATION.

THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

R	-	1
FC	-	0
LC	-	822
IC	-	1
FT	-	0
FS	-	0

## 9.6 TEST 5 - CONVERGING/DIVERGING SEEK TEST

THIS TEST WILL CAUSE THE DRIVE TO EXECUTE FORWARD AND REVERSE SEEKS FROM "NC1" AND "NC2" RESPECTIVELY. "NC1" WILL BE INCREMENTED BY "IC" AND "NC2" WILL BE DECREMENTED BY "IC" UNTIL "NC1" IS GREATER THAN THE INITIAL VALUE OF "NC2" AND "NC2" IS LESS THAN THE INITIAL VALUE OF "NC1". AT THE COMPLETION OF EACH SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO ENSURE PROPER OPERATION. "NC1" AND "NC2" DEFAULT TO "FC" AND "LC" RESPECTIVELY.

THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

R	-	1
FC	-	0
LC	-	822
IC	-	1
FT	-	0
FS	-	0

## 9.7 TEST 6 - SERVO ADDRESSING LOGIC NOISE GENERATOR TEST

SEQ 0019

IN THIS TEST A SEEK IS DONE TO CYL "NC" THEN A SEEK TO NC+4 THEN NC+1 THEN NC+3 THEN NC+2 THEN NC+5. NOW "NC" IS UPDATED BY "IC" AND THE ABOVE SEQUENCE IS REPEATED UNTIL "LC" IS EXCEEDED BY ANY OF THE ABOVE VALUES. THE INITIAL VALUE OF "NC" IS "FC". AT THE COMPLETION OF EACH SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO ENSURE PROPER OPERATION.

THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

R	-	1
FC	-	0
LC	-	822
IC	-	1
FT	-	0
FS	-	0

## 9.8 TEST 7 - RANDOM SEEK TEST

THIS TEST PERFORMS RANDOM SEEK OPERATIONS BETWEEN CYLINDERS 'FC' 'LC'. AFTER EACH SEEK, THE POSITION OF THE DRIVE IS VERIFIED BY READING A SECTOR FROM THE CURRENTLY ADDRESSED CYLINDER AND TRACK. THE TRACK ADDRESS IS INCREMENTED FOR EACH SEEK SO THAT VERIFICATION OF POSITIONING OCCURS USING EACH HEAD. TRACK ADDRESSES ARE INCREMENTED BETWEEN PARAMETERS 'FT' AND 'LT'.

THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

R	-	5000
FC	-	0
LC	-	822
FT	-	0
LT	-	18

## 9.9 TEST 10 - SERVO SETTLE DOWN TEST

THIS TEST VERIFIES THAT THE SERVO HAS SETTLED DOWN AND THAT THE DRIVE IS ON CYLINDER WHEN THE DRIVE INDICATES SEEK COMPLETE. RANDOM SEEKS ARE ISSUED BETWEEN CYLINDERS 'NC1' AND 'NC1+IC' ('NC1' STARTS AT VALUE 'FC'). AT THE COMPLETION OF 1000 (10) SEEKS, 'NC1' IS INCREMENTED BY VALUE 'IC' AND THE SEQUENCE IS REPEATED. THE TEST IS COMPLETED WHEN 'NC1' HAS BEEN INCREMENTED BEYOND 'LC'.

WHEN THE SEEK COMPLETES, THE PROGRAM READS THE DRIVE'S LOOK-AHEAD REGISTER (RLA) TO DETERMINE THE ADDRESS OF THE SECTOR ROTATING INTO POSITION. THE PROGRAM THEN ISSUES A WRITE HEADER AND DATA COMMAND FOR THAT SECTOR. ERRORS IN THIS TEST INDICATE THAT THE SERVO SYSTEM MAY NOT BE ADJUSTED CORRECTLY, THAT THE DRIVE IS MALFUNCTIONING, OR THAT A PACK WITH MARGINAL SERVO TRACKS IS MOUNTED ON THE DRIVE.

THIS TEST IS VALID ONLY IF THE OPERATION IS STARTED WITHIN A FEW HUNDRED MICRO-SECONDS AFTER SEEK DONE OCCURS. THE NECESSARY TIME DEPENDENT PARAMETERS OCCUR WITHIN THE REQUIRED TIME RANGE FREQUENTLY ENOUGH TO PERMIT THIS TEST TO BE EFFECTIVE.

THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

R	-	1
FC	-	0
LC	-	822
IC	-	100
FT	-	0

#### 9.10 TEST 11 - ALL SEEKS TEST

THIS TEST VERIFIES THAT THE DISK DRIVE CAN SEEK FROM EACH CYLINDER TO ALL OTHER CYLINDERS.

BEGINNING WITH CYLINDER 'FC', THE TEST SEEKS TO EACH CYLINDER BETWEEN 'FC' AND 'LC' FROM CYLINDER 'FC'. THE BEGINNING CYLINDER ADDRESS IS INCREMENTED AND THE TEST SEEKS BETWEEN THE NEW CYLINDER ADDRESS AND ALL CYLINDERS BETWEEN 'FC' AND 'LC'. THE SEQUENCE CONTINUES UNTIL ALL CYLINDERS HAVE BEEN CHECKED.

THE FOLLOWING PARAMETERS ARE USED BY THIS TEST:

R	-	1
FC	-	0
LC	-	822
IC	-	1
FT	-	0
FS	-	0

#### 9.11 TEST 12 - ROTATIONAL SPEED TIMING TEST

THIS TEST WILL START A SEARCH TO CYLINDER 0, TRACK 0, SECTOR 0. AS SOON AS THE INTERRUPT OCCURS, THE GO BIT IS SET AGAIN AND THE OPERATION IS TIMED. THIS PROCEDURE IS REPEATED 10 TIMES THEN THE AVERAGE TIME IS CALCULATED AND CHECKED TO ENSURE IT IS WITHIN TOLERANCE:

16.67 MS/REV + 2% OR - 3.5% IF 60HZ  
16.67 MS/REV + 2% OR - 3.5% IF 50HZ.

THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

R	-	1
FC	-	0
FT	-	0
FS	-	0

## 9.12 TEST 13 - ONE CYLINDER SEEK TIMING TEST

THIS TEST WILL COMMAND FORWARD SEEK CYCLES TO ADVANCE THE CYLINDER BY ONE UNTIL THE INCREMENT IS GREATER THAN THE CYLINDER 'LC'. THEN REVERSE SEEK TO CYLINDER 'FC'. THE TIME TO PERFORM EACH SEEK IS CHECKED TO ENSURE IT DOES NOT EXCEED THE MAXIMUM TIME PERMITTED FOR A ONE CYLINDER SEEK. THE TIME MUST BE LESS THAN 10MS.

THE TEST USES THE FOLLOWING PARAMETERS:

R	-	1
FC	-	0
LC	-	822

## 9.13 TEST 14 - ACCESS TIME MEASUREMENT

THIS TEST WILL COMMAND A FORWARD SEEK FROM CYLINDER 0 TO CYLINDER 'LC'. THEN A REVERSE SEEK FROM CYLINDER 'LC' TO CYLINDER 0. BOTH SEEKS ARE TIMED AND CHECKED TO ENSURE THEY ARE WITHIN THE TOLERANCE ALLOWED FOR THE ACCESS TIME MEASUREMENT. THIS SEQUENCE IS REPEATED 128 TIMES (FOR A TOTAL OF 256 SEEKS). THE AVERAGE ACCESS TIME MUST BE LESS THAN 32 MS. CYLINDER 'LC' DEFAULTS TO 255 (10).

THE TEST USES THE FOLLOWING PARAMETERS:

R	-	1
FC	-	0
LC	-	255

## 9.14 TEST 15 - MAXIMUM SEEK TIMING TEST

THIS TEST WILL COMMAND A FORWARD SEEK FROM CYLINDER 0 TO CYLINDER 'LC'. THEN A REVERSE SEEK FROM CYLINDER 'LC' TO CYLINDER 0. BOTH SEEKS ARE TIMED AND CHECKED TO ENSURE THEY ARE WITHIN THE TOLERANCE ALLOWED FOR THE MAXIMUM SEEK TIME. THIS SEQUENCE IS REPEATED 128 TIMES (FOR A TOTAL OF 256 SEEKS). THE MAXIMUM SEEK TIME MUST BE LESS THAN 54 MS. 'LC' DEFAULTS TO 822 (10).

THE TEST USES THE FOLLOWING PARAMETERS:

R	-	1
FC	-	0
LC	-	822

## 9.15 TEST 16 - SECTOR ADDRESSING TEST

THIS TEST WRITES DATA INTO ALL SECTORS OF TRACK "FT". THE DATA WILL BE 256 WORDS OF THE SECTOR ADDRESS OF THE SECTOR BEING WRITTEN. A WRITE CHECK IS PERFORMED, THE BUFFER IS CLEARED (TO 177400) AND THE DATA IS READ AND COMPARED. THEN SECTOR 0 IS REWRITTEN AND SECTORS 0 - 31 ARE WRITE CHECKED. THEN SECTOR 1 IS REWRITTEN AND SECTORS 0 - 31 ARE WRITE CHECKED. THIS REWRITE AND WRITE CHECK PROCEDURE IS CONTINUED UP THROUGH REWRITE SECTOR 31 AND WRITE CHECK SECTORS 0-31.

THE TEST USES THE FOLLOWING PARAMETERS:

R - 1  
 FC - 0  
 FT - 0

9.16 TEST 17 - TRACK ADDRESSING TEST

THIS TEST WILL WRITE DATA IN THE FORM OF TRACK ADDRESSES IN CYLINDER "FC" SECTOR "FS" OF EVERY TRACK WITH EACH TRACK GETTING ITS OWN TRACK ADDRESS. A WRITE CHECK IS THEN PERFORMED ON EACH TRACK TO INSURE THE DATA IS VALID. THEN TRACK 0 IS REWRITTEN AND TRACK 1 THROUGH TRACK 4 IS WRITE CHECKED. THEN TRACK 1 IS REWRITTEN AND TRACK 2 THROUGH TRACK 4 IS WRITE CHECKED. THIS PROCEDURE IS CONTINUED UP THROUGH REWRITING TRACK 3 AND WRITE CHECKING TRACK 4.

THE TEST USES THE FOLLOWING PARAMETERS:

R - 1  
 FC - 0  
 FS - 0

9.17 TEST 20 - DATA TEST

THIS TEST PERFORMS DATA STORAGE AND RETRIEVAL ON CYLINDERS "FC" THROUGH "LC" BY THE INCREMENT "IC" USING THE DATA PATTERNS SPECIFIED. THE FOLLOWING SEQUENCE OCCURS FOR EACH CYLINDER:

1. SET "NT" TO "FT" THEN REPEAT 2-4 UNTIL "NT" > "LT"
2. WRITE THEN WRITE CHECK "FS" THROUGH "LS" OF TRACK "NT"
3. READ THEN SOFTWARE COMPARE "FS" THROUGH "LS" OF TRACK "NT"
4. INCREMENT "NT" BY "IT"
5. REPEAT STEPS 1-4 FOR EACH DATA PATTERN
6. REPEAT STEPS 1-5 FOR "FC" THROUGH "LC" ADVANCING BY "IC"

IF A WRITE CHECK ERROR OCCURS THE ERROR IS REPORTED AND THE TRACK IN ERROR IS REWRITTEN AND CHECKED. THIS CHECK IS ACCOMPLISHED BY PERFORMING TWO(2) SUCCESSIVE ERROR FREE WRITE CHECKS. IF THE CHECK FAILS THE ERROR IS REPORTED AS FATAL AND NO READ OCCURS.

FS DEFAULTS TO 1 AND LS DEFAULTS TO 0  
 PAT DEFAULTS TO 177777 (ALL POSSIBLE PATTERNS)  
 THE POSSIBLE PATTERNS ARE:

PAT 0	PAT 1	PAT 2	PAT 3	PAT 4	PAT 5	PAT 6	PAT 7
165555	000001	177776	000000	000000	052525	007417	026455
133333	000003	177774	000000	010421	052525	007417	026455
165555	000007	177770	000000	021042	052525	007417	026455
133333	000017	177760	177777	031463	125252	170360	151322
165555	000037	177740	177777	042104	125252	170360	151322
133333	000077	177700	177777	052525	125252	170360	151322
165555	000177	177600	000000	063146	052525	007417	026455
133333	000377	177400	000000	073567	052525	007417	026455
165555	000777	177000	177777	104210	125252	170360	151322
133333	001777	176000	177777	114631	125252	170360	151322
165555	003777	174000	000000	125252	052525	007417	026455
133333	007777	170000	177777	135673	125252	170360	151322
165555	017777	160000	000000	146314	052525	007417	026455
133333	037777	140000	177777	156735	125252	170360	151322
165555	077777	100000	000000	167356	052525	007417	026455



K02

SEQ 0023

133333	177777	000000	177777	177777	125252	170360	151322
PAT 8	PAT 9	PAT 10	PAT 11	PAT 12	PAT 13	PAT 14	PAT 15
165555	000001	177776	172666	077777	153333	000000	177777
133333	000002	177775	155555	137777	066667	177777	000000
165555	000004	177773	172666	157777	153333	177777	000000
133333	000010	177767	155555	167777	066667	177777	000000
165555	000020	177757	172666	173777	153333	177777	000000
133333	000040	177737	155555	175777	066667	177777	000000
165555	000100	177677	172666	176777	153333	177777	000000
133333	000200	177577	155555	177377	066667	177777	000000
165555	000400	177377	172666	177577	153333	177777	000000
133333	001000	176777	155555	177677	066667	177777	000000
165555	002000	175777	172666	177377	153333	177777	000000
133333	004000	173777	155555	177577	066667	177777	000000
165555	010000	167777	172666	177677	153333	177777	000000
133333	020000	157777	155555	177773	066667	177777	000000
165555	040000	137777	172666	177775	153333	177777	000000
133333	100000	077777	155555	177776	066667	177777	000000

THE TEST USES THE FOLLOWING PARAMETERS:

R	-	1
FC	--	0
LC	--	821
IC	--	64
FT	--	0
LT	--	4
IT	--	1
FS	--	1
LS	--	0
PAT	-	177777

9.18 TEST 21 - RANDOM ADDRESS AND RANDOM PATTERN TEST

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

THE TEST USES THE FOLLOWING PARAMETERS:

R	-	20000
---	---	-------

L02

FC - 0  
LC - 821

SEQ 0024

9.19 TEST 22 - RMO3 ACCESS TIME TEST

THIS TEST PERFORMS SEEKS BETWEEN CYLINDERS 0 & 255 TO ALLOW THE OPERATOR TO ADJUST THE ACCESS TIME ON AN RMO3 USING THE DOU. THE PROGRAM STALLS APPROXIMATELY 5 SECONDS BETWEEN SEEKS SO THAT THE ACCESS TIME INDICATORS ON THE DOU MAY BE OBSERVED.

THE TEST USES THE FOLLOWING PARAMETERS:

R - 5000  
FC - 0  
LC - 255

I D E N T I F I C A T I O N

PRODUCT NAME: RMO3 SOFTWARE DRIVER MODULE

DATE CREATED: AUGUST 1977

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: J. LACEY/C. HESS/C. CHEN

COPYRIGHT (C) 1977 DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

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

ACTUAL DISTRIBUTION OF THE SOFTWARE DESCRIBED IN THIS DOCUMENT WILL BE SUBJECT TO TERMS AND CONDITIONS TO BE ANNOUNCED ON SOME FUTURE DATE BY DIGITAL EQUIPMENT CORPORATION.

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

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

10.1 RH70/RM03 DRIVER MODULE  
-----

THIS DOCUMENT IS THE USER'S GUIDE FOR THE RH70/RM03 DRIVER.

10.2 TO INITIALIZE THE DRIVER:

```
JSR    PC,RMINIT
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:

DRVSTA -----	DRIVE STATE -----
>0	ONLINE RM03
=0	OFFLINE RM03, DRIVE IS NOT AN RM03, OR NONEXISTENT DRIVE
<0	UNSAFE RM03

THE DRIVE TYPE IS DEFINED IN AN 8 BYTE LONG TABLE TAGGED 'DRVTYP'. THE TABLE CONTAINS ONE BYTE FOR EACH DRIVE AND IS INDEXED BY THE DRIVE NUMBER. ENTRIES ARE ENCODED AS FOLLOWS:

DRVTYP -----	CONDITION -----
0	NONEXISTENT DRIVE
4	RM03
-1	NOT AN RM03

THE 'RMINIT' ROUTINE WILL DO A READIN PRESET AND WILL SET FMT22.

10.3 AFTER THE DRIVER HAS BEEN INITIALIZED, IT IS CALLED USING THE FOLLOWING SEQUENCE.

```
CALL: JSR    RO,RM03      ;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
        .WORD 0      ;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 PCINTER
              ;POINTS TO THE FIRST OF TWENTY
              ;LOCATIONS OF WHERE THE DRIVER
              ;IS TO STORE THE r170/RM03
              ;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

```

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

```

MOV    #16.,-(SP) ;16 MILLISECONDS BETWEEN
              ;CLOCK TICKS
JSR    PC,RMTMR  ;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.

#### 10.4.1 EXAMPLE - WRITE 1000. WORDS

```

1$:    JSR    R0,RM03      ;CALL THE DRIVER
        WRTDPB      ;DPB ADDRESS
        BR     1$         ;WAIT FOR QUEUE IF FULL
2$:    TST    WRTDPB+16    ;WAIT FOR COMMAND TO COMPLETE
        BEQ    2$
        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,ERRTBS,0

## 10.5 RH70/RM03 REGISTERS

<u>MNEMONIC</u>	<u>INDEX</u>
RMCS1	0
RMWC	2
RMBA	4
RMDA	6
RMCS2	10
RMDS	12
RMER1	14
RMAS	16
RMLA	20
RMOB	22
RMR1	24
RMDT	26
RMSN	30
RMOF	32
RMDC	34
RMR	36
RMR2	40
RMER2	42
RMEC1	44
RMEC2	46

## 10.6 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

D03

SEQ 0029

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

#### 10.7 DPB STATUS/ERROR INDICATOR WORD

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)	UNCORRECTABLE 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) CORRECTABLE UNSAFE CONDITION OCCURRED
- 3(2) DRIVE ERROR OCCURRED THAT CAUSED AN AUTOMATIC "RECALIBRATE" SEQUENCE
- 2 PORT REQUEST TIMEOUT. THE DRIVER REQUESTED THE DRIVE BUT THE OPPOSITE PORT DID NOT RELEASE THE DRIVE WITHIN 20 SECONDS.
- 1 NON-EXISTENT DRIVE REQUESTED. USER MADE A REQUEST FOR A NON-EXISTENT DRIVE.
- (1) => REQUEST WASN'T PUT IN QUEUE. (rh70/RM03 REGISTERS WERE NOT SAVED)
- (2) => REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A "DRIVE CLEAR" TO THE DRIVE.  
NOTE: ALL rh70/RM03 REGISTERS ARE SAVED AS PER DPB+14 BEFORE THE "DRIVE CLEAR".
- (3) => REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A MASSBUS INIT. ALL rh70/RM03 REGISTERS FOR THE DRIVE WERE SAVED AS PER DPB+14 BEFORE THE INIT.
- (4) => A "RECALIBRATE" SHOULD BE ISSUED BEFORE ANY OTHER COMMAND.

## 10.8 ERROR CALLS MADE BY THE DRIVER.

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

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	RH70 INTERRUPT OCCURRED (RHAS=0)	*R4= RMCS1'S ADDRESS
2	UNEXPECTED ATTENTION OCCURRED	R1= DRIVE NUMBER R3= ATA BIT *R4= RMCS1'S ADDRESS R5= (RMAS) RMERRS =RMDS RMERRS+2=RMER1 RMERRS+4=RMER2 RMERRS+6=RMER2
3	MASSBUS PARITY ERROR (MCPE=1)	RD.ADR= ADDRESS OF REG. READ RD.WRD= WORD READ
4	MASSBUS PARITY ERROR (PAR=1)	WRT.AD= ADDRESS OF REG. WRITTEN WRT.WD= WORD WRITTEN RD.WRD= WORD READ BACK
5	ADDRESS PLUG CHANGE	R1= DRIVE NUMBER



F03

BIT SET ('OPE' ERROR)

R3= ATA BIT  
\*R4= RMCS1'S ADDRESS  
R5= (RMAS)  
RMERRS =RMS  
RMERRS+2=RMR1  
RMERRS+4=RMR2  
RMERRS+6=RMR2

SEQ 0031

\* THIS IS THE ACTUAL UNIBUS ADDRESS (176700)

11. PROGRAM LISTING

14	CONTROL SWITCH SETTINGS
37	OPERATIONAL SWITCH SETTINGS
56	TRAP CATCHER
66	ACT11 HOOKS
77	STARTING ADDRESSES
88	BASIC DEFINITIONS
200	RH70 REGISTERS
246	RMO3 REGISTERS
454	COMMON TAGS
1370	ERROR POINTER TABLE
1698	START OF PROGRAM
1711	INITIALIZE THE COMMON TAGS
1771	GET VALUE FOR SOFTWARE SWITCH REGISTER
1949	**** TESTS ****
1984	*** SEEK TESTS ***
2002	T0 RECAL/SEEK TEST
2050	T1 SEEK/SEEK TEST
2102	T2 INCREMENT/SEEK TEST
2165	T3 STEPPING SEEK TEST
2220	T4 OSCILLATING SEEK TEST
2313	T5 CONVERGING/DIVERGING SEEK TEST
2374	T6 SERVO ADDRESSING LOGIC NOISE GENERATOR
2440	T7 RANDOM SEEK TEST
2532	T10 SERVO SETTLE DOWN TEST
2716	T11 ALL SEEKS TEST
2779	*** TIMING TESTS ***
2799	T12 ROTATIONAL SPEED TIMING TEST
2920	T13 ONE CYLINDER SEEK TIMING TEST
3024	T14 ACCESS TIME MEASUREMENT TEST
3138	T15 MAXIMUM SEEK TIMING TEST
3252	*** ADDRESSING TESTS ***
3268	T16 SECTOR ADDRESSING TEST
3354	T17 TRACK ADDRESSING TEST
3449	*** DATA TEST ***
3451	T20 DATA TEST
3645	*** EXERCISE TEST ***
3647	T21 RANDOM ADDRESS AND RANDOM PATTERN TEST
3777	*** RMO3 ACCESS TIME ADJUSTMENT TEST ***
3779	T22 RMO3 ACCESS TIME ADJUSTMENT TEST
3817	END OF PASS ROUTINE
3875	*** SYSMAC SUBROUTINES ***
3877	ERROR HANDLER ROUTINE
3930	TYPEERR - TYPE ERROR ROUTINE
4034	TYPE ROUTINE
4105	BINARY TO OCTAL (ASCII) AND TYPE
4183	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
4251	TTY INPUT ROUTINE
4508	SCOPE HANDLER ROUTINE
4566	SAVE AND RESTORE RD-RS ROUTINES
4612	TRAP DECODER
4635	TRAP TABLE
4657	SINGLE LENGTH BINARY TO DECIMAL ASCII ROUTINE
4676	DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
4739	TYPE NUMERICAL ASCII STRING SUPPRESS LEADING ZEROS
4763	RANDOM NUMBER GENERATOR ROUTINE
4800	INTEGER DIVIDE ROUTINE

H03

MD-11-DZRMF-A RMO3 EXTENDED DRIVE TEST MACY11 30(1046) 22-JUL-77 16:19  
DZRMFA.P11 22-JUL-77 14:59 TABLE OF CONTENTS

SEQ 0033

4883	*** PROGRAM SUBROUTINES ***
6728	SINGLE/DUAL PORT RH70/RMO3 DRIVER (REV 1.0)
8123	ASCIZ MESSAGES
8242	ERROR HEADER (EM) MESSAGES
8322	STATUS/ERROR INDICATOR MESSAGES
8394	DATA HEADER (DT) MESSAGES
8534	DATA TABLE (DT)
8594	DATA FORMAT (DF) TABLE
8747	ROUTINE TO SIZE MEMORY
8779	GETADR - GET BUS ADDRESS AND VECTOR ADDRESS

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11

.TITLE MD-11-DZRMF-A RMO3 EXTENDED DRIVE TEST  
:\*COPYRIGHT (C) 1977  
:\*DIGITAL EQUIPMENT CORP.  
:\*MAYNARD, MASS. 01754  
:\*  
:\*PROGRAM BY H. BLACKSTONE  
:\*  
:\*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC  
:\*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.  
:\*

12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67

.SBTTL CONTROL SWITCH SETTINGS

```

*
* SWITCH STATE USE
*-----
* 15 0 WRITE PACK BEFORE TESTING (TEST 21)
* 14 1 INHIBIT WRITING PACK BEFORE TESTING (TEST 21)
* 13 0 NO STALL BETWEEN DRIVE FUNCTIONS
* 12 1 STALL AFTER EVERY DRIVE FUNCTION
* 11 0 USE SPECIFIC STALL TIME
* 10 1 USE RANDOM STALL
* 9 0 NO INCREMENTING STALL IN TEST 4
* 8 1 DO INCREMENTING STALL IN TEST 4
* 7 0 DO IMPLIED SEEKS WITH DATA TRANSFERS
* 6 1 DO EXPLICIT SEEKS BEFORE DATA TRANSFERS
* 5 0 DO "READ HEADER AND DATA" IN TESTS 0-11
* 4 1 DO EXPLICIT SEEKS IN TESTS 0-11
* 3 0 60 HZ
* 2 1 50 HZ
* 1 0 RUN WATCHDOG TIMER
* 0 1 INHIBIT WATCHDOG TIMER
* 0 0 TEST DRIVE(S) IN 22 SECTOR (16 BIT) MODE
* 1 TEST DRIVE(S) IN 20 SECTOR (18 BIT) MODE

```

.SBTTL OPERATIONAL SWITCH SETTINGS

```

*
* SWITCH USE
*-----
* 15 HALT ON ERROR
* 14 LOOP ON TEST
* 13 INHIBIT ERROR TIMEOUTS
* 12 BELL ON ERROR
* 11 LOOP ON ERROR
* 10 PRINT ERROR MESSAGE ON LINE PRINTER
* 9 READ CONTROL SWITCH SETTINGS FROM TTY
* 8 INHIBIT TIME REPORTS (TESTS 12-15)
* 7 REPORT ONE ERROR PER SECTOR (TESTS 16 & 17)
* 6 INHIBIT WRITES (TEST 15)
* 5 INHIBIT WRITE CHECKS (TEST 20)
* 4 INHIBIT READ AND SOFTWARE COMPARES (TEST 20)
* 3 INHIBIT SOFTWARE COMPARES (TEST 20)
* 2 PERFORM READ AFTER WRITE CHECK ERROR (TEST 20)
* 1
* 0

```

.SBTTL TRAP CATCHER

000000

```

;=0
;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
;=174
DISPREG: .WORD 0 ;;SOFTWARE DISPLAY REGISTER
SWREG: .WORD 0 ;;SOFTWARE SWITCH REGISTER

```

000174 000000  
000176 000000

.SBTTL ACT11 HOOKS

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

```

68      ;HOOKS REQUIRED BY ACT11
69      000200      $SVPC=      ;SAVE PC
70      000046      .=46
71      000046      020634      SENDAD      ;;1)SET LOC.46 TO ADDRESS OF SENDAD IN .SEOP
72      000052      000052      .=52
73      000052      000000      .WORD 0      ;;2)SET LOC.52 TO ZERO
74      000200      .=$SVPC      ;; RESTORE PC
75
76      .SBTTL  STARTING ADDRESSES
77      000200      .=200
78      000200      000137      004636      ;*200 = NORMAL START
79      JMP          @#START1
80      000204      000137      004660      ;*204 = SELECT OPERATING PARAMETERS
81      JMP          @#START2
82      000210      000137      004626      ;*210 = SELECT RM70/RM03 ADDRESSES
83      JMP          @#START3
84      000214      000137      004650      ;*214 = COMBINATION OF 204 AND 210
85      JMP          @#START4
86
87      .SBTTL  BASIC DEFINITIONS
88
89      ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
90      001100      STACK= 1100
91      .EQUIV  EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
92      .EQUIV  IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
93
94      ;*MISCELLANEOUS DEFINITIONS
95      000011      HT= 11      ;;CODE FOR HORIZONTAL TAB
96      000012      LF= 12      ;;CODE FOR LINE FEED
97      000015      CR= 15      ;;CODE FOR CARRIAGE RETURN
98      000200      CRLF= 200      ;;CODE FOR CARRIAGE RETURN-LINE FEED
99      177776      PS= 177776      ;;PROCESSOR STATUS WORD
100     .EQUIV  PS,PSW
101     177774      STKLMT= 177774      ;;STACK LIMIT REGISTER
102     177772      PIRQ= 177772      ;;PROGRAM INTERRUPT REQUEST REGISTER
103     177570      DSWR= 177570      ;;HARDWARE SWITCH REGISTER
104     177570      DDISP= 177570      ;;HARDWARE DISPLAY REGISTER
105
106     ;*GENERAL PURPOSE REGISTER DEFINITIONS
107     0C0000      R0= %0      ;;GENERAL REGISTER
108     003001      R1= %1      ;;GENERAL REGISTER
109     006002      R2= %2      ;;GENERAL REGISTER
110     000003      R3= %3      ;;GENERAL REGISTER
111     000004      R4= %4      ;;GENERAL REGISTER
112     000005      R5= %5      ;;GENERAL REGISTER
113     000006      R6= %6      ;;GENERAL REGISTER
114     000007      R7= %7      ;;GENERAL REGISTER
115     000006      SP= %6      ;;STACK POINTER
116     000007      PC= %7      ;;PROGRAM COUNTER
117
118     ;*PRIORITY LEVEL DEFINITIONS
119     000000      PR0= 0      ;;PRIORITY LEVEL 0
120     000040      PR1= 40      ;;PRIORITY LEVEL 1
121     000100      PR2= 100      ;;PRIORITY LEVEL 2
122     000140      PR3= 140      ;;PRIORITY LEVEL 3
123     000200      PR4= 200      ;;PRIORITY LEVEL 4

```

124	000240	PRS=	240	::	PRIORITY LEVEL 5
125	000300	PR6=	300	::	PRIORITY LEVEL 6
126	000340	PR7=	340	::	PRIORITY LEVEL 7

```

127
128      .*"SWITCH REGISTER" SWITCH DEFINITIONS

```

129	100000	SW15=	100000
130	040000	SW14=	40000
131	020000	SW13=	20000
132	010000	SW12=	10000
133	004000	SW11=	4000
134	002000	SW10=	2000
135	001000	SW09=	1000
136	000400	SW08=	400
137	000200	SW07=	200
138	000100	SW06=	100
139	000040	SW05=	40
140	000020	SW04=	20
141	000010	SW03=	10
142	000004	SW02=	4
143	000002	SW01=	2
144	000001	SW00=	1
145		.EQUIV	SW09, SW9
146		.EQUIV	SW08, SW8
147		.EQUIV	SW07, SW7
148		.EQUIV	SW06, SW6
149		.EQUIV	SW05, SW5
150		.EQUIV	SW04, SW4
151		.EQUIV	SW03, SW3
152		.EQUIV	SW02, SW2
153		.EQUIV	SW01, SW1
154		.EQUIV	SW00, SW0

```

155
156      .*DATA BIT DEFINITIONS (BIT00 TO BIT15)

```

157	100000	BIT15=	100000
158	040000	BIT14=	40000
159	020000	BIT13=	20000
160	010000	BIT12=	10000
161	004000	BIT11=	4000
162	002000	BIT10=	2000
163	001000	BIT09=	1000
164	000400	BIT08=	400
165	000200	BIT07=	200
166	000100	BIT06=	100
167	000040	BIT05=	40
168	000020	BIT04=	20
169	000010	BIT03=	10
170	000004	BIT02=	4
171	000002	BIT01=	2
172	000001	BIT00=	1
173		.EQUIV	BIT09, BIT9
174		.EQUIV	BIT08, BIT8
175		.EQUIV	BIT07, BIT7
176		.EQUIV	BIT06, BIT6
177		.EQUIV	BIT05, BIT5
178		.EQUIV	BIT04, BIT4
179		.EQUIV	BIT03, BIT3

```

180 .EQUIV BIT02,BIT2
181 .EQUIV BIT01,BIT1
182 .EQUIV BIT00,BIT0
183
184 ;*BASIC "CPU" TRAP VECTOR ADDRESSES
185 ERRVEC= 4 ;: TIME OUT AND OTHER ERRORS
186 RESVEC= 10 ;: RESERVED AND ILLEGAL INSTRUCTIONS
187 TBITVEC=14 ;: "T" BIT
188 TRTVEC= 14 ;: TRACE TRAP
189 BPTVEC= 14 ;: BREAKPOINT TRAP (BPi)
190 IOTVEC= 20 ;: INPUT/OUTPUT TRAP (IOT) **SCOPE**
191 PWRVEC= 24 ;: POWER FAIL
192 EMTVEC= 30 ;: EMULATOR TRAP (EMT) **ERROR**
193 TRAPVEC=34 ;: "TRAP" TRAP
194 TKVEC= 60 ;: TTY KEYBOARD VECTOR
195 TPVEC= 64 ;: TTY PRINTER VECTOR
196 PIRQVEC=240 ;: PROGRAM INTERRUPT REQUEST VECTOR
197 ;:*****
198
199 .SBTTL RM70 REGISTERS
200
201 ;:*****
202
203 ;CONTROL AND STATUS REGISTER 1 (RMCS1)
204
205 IE= 100 ;: INTERRUPT ENABLE (BIT #6)
206 RDY= 200 ;: READY (BIT #7)
207 A16= 400 ;: HIGH ORDER BUS ADDRESS BIT (BIT #8)
208 A17= 1000 ;: HIGH ORDER BUS ADDRESS BIT (BIT #9)
209 PSEL= 2000 ;: PORT SELECT (BIT #10)
210 MCPE= 20000 ;: MASSBUS PARITY ERROR (BIT #13)
211 TRE= 40000 ;: TRANSFER ERROR (BIT #14)
212 ;SC= 100000 ;: SPECIAL CONDITION (BIT #15)
213
214 ;WORD COUNT REGISTER (RMWC)
215 ;(EACH BIT IS CALLED BY BIT NUMBER)
216
217 ;BUS ADDRESS REGISTER (RMBA)
218 ;(EACH BIT IS CALLED BY BIT NUMBER)
219
220 ;CONTROL AND STATUS REGISTER 2 (RMCS2)
221
222 US1= 1 ;: UNIT SELECT (BIT #0)
223 US2= 2 ;: UNIT SELECT (BIT #1)
224 US3= 4 ;: UNIT SELECT (BIT #2)
225 BAI= 10 ;: BUS ADDRESS INCREMENT INHIBIT (BIT #3)
226 PAT= 20 ;: MASSBUS PARITY TEST (BIT #4)
227 CLR= 40 ;: CLEAR (BIT #5)
228 IR= 100 ;: INPUT READY (BIT #6)
229 OR= 200 ;: OUTPUT READY (BIT #7)
230 MPE= 400 ;: MASS BUS PARITY ERROR (BIT #8)
231 MXF= 1000 ;: MISSED TRANSFER ERROR (BIT #9)
232 PGE= 2000 ;: PROGRAM ERROR (BIT #10)
233 NEM= 4000 ;: NON EXISTENT MEMORY (BIT #11)
234 NED= 10000 ;: NON EXISTENT DRIVE (BIT #12)
235 UPE= 20000 ;: UNIBUS PARITY ERROR (BIT #13)

```



```

236          040000      WCE=      40000      ;WRITE CHECK ERROR (BIT #14)
237          100000      DLT=      100000     ;DATA LATE (BIT #15)
238
239          ;DATA BUFFER REGISTER (RMOB)
240          ;(EACH BIT IS CALLED BY BIT NUMBER)
241
242
243          ;*****
244          .SBTTL  RMO3 REGISTERS
245          ;*****
246
247          ;CONTROL AND STATUS 1 REGISTER. (#00)
248
249
250
251          000001      GO=         1          ;GO BIT (BIT #0)
252          000002      F1=         2          ;FUNCTION CODE BIT #1
253          000004      F2=         4          ;FUNCTION CODE BIT #2
254          000010      F3=        10          ;FUNCTION CODE BIT #3
255          000020      F4=        20          ;FUNCTION CODE BIT #4
256          000040      F5=        40          ;FUNCTION CODE BIT #5
257          004000      DVA=        4000      ;DEVICE AVAILABLE (BIT #11)
258
259          ;DRIVE STATUS REGISTER (RMD5) (#01)
260
261          000001      OM=          1          ;
262          000100      VV=         100        ;VOLUME VALID (BIT #6)
263          000200      DRY=        200        ;DRIVE READY (BIT #7)
264          000400      DPR=        400        ;DRIVE PRESENT (BIT #8)
265          001000      PGM=       1000        ;PROGRAMMABLE (BIT #9)
266          002000      LST=       2000        ;LAST SECTOR TRANSFERRED (BIT #10)
267          004000      WRL=       4000        ;WRITE LOCK (BIT #11)
268          010000      MOL=      10000        ;MEDIUM ON-LINE (BIT #12)
269          020000      PIP=      20000        ;POSITIONING OPERATION IN PROGRESS (BIT #13)
270          040000      ERR=      40000        ;COMPOSITE ERROR (BIT #14)
271          100000      ATA=     100000        ;ATTENTION ACTIVE (BIT #15)
272
273          ;ERROR REGISTER #01 (RMER1) (#02)
274
275          000001      ILF=         1          ;ILLEGAL FUNCTION (BIT #0)
276          000002      ILR=         2          ;ILLEGAL REGISTER (BIT #1)
277          000004      RMR=         4          ;REGISTER MODIFICATION REFUSED (BIT #2)
278          000010      PAR=        10          ;PARITY ERROR (BIT #3)
279          000020      FER=        20          ;FORMAT ERROR (BIT #4)
280          000040      WCF=        40          ;WRITE CLOCK FAIL (BIT #5)
281          000100      ECH=        100        ;ECC HARD ERROR (BIT #6)
282          000200      HCE=        200        ;HEADER COMPARE ERROR (BIT #7)
283          000400      HCRC=       400        ;HEADER CRC ERROR (BIT #8)
284          001000      AOE=       1000        ;ADDRESS OVERFLOW ERROR (BIT #9)
285          002000      IAE=       2000        ;INVALID ADDRESS ERROR (BIT #10)
286          004000      WLE=       4000        ;WRITE LOCK ERROR (BIT #11)
287          010000      DTE=      10000        ;DRIVE TIMING ERROR (BIT #12)
288          020000      OPI=      20000        ;OPERATION INCOMPLETE (BIT #13)
289          040000      UNS=      40000        ;DRIVE UNSAFE (BIT #14)
290          100000      DCK=     100000        ;DATA CHECK ERROR (BIT 15)
291

```

```

292 ;MAINTAINABILITY REGISTER #01 (RMMR1)(#03) - READ ONLY BITS
293
294 000001 DMD= 1 ;DIAGNOSTIC MODE
295 000002 LSIT= 2
296 ;LS=4
297 000010 MD= 10
298 000020 EECC= 20
299 000040 WC= 40
300 000100 CONT= 100
301 000200 PHA= 200
302 000400 PDA= 400
303 001000 ECRC= 1000
304 002000 PLFS= 2000
305 004000 ESRC= 4000
306 010000 REX= 10000
307 020000 EBL= 20000
308 ;R/G= 40000
309 100000 OCC= 100000

```

```

310 ;MAINTAINABILITY REGISTER #01 (RMMR1) (#03) - WRITE ONLY BITS
311
312 ;DIAGNOSTIC MODE BIT
313 000301 DMD= 1
314 000002 MSC= 2
315 000004 MI= 4
316 000010 MWP= 10
317 000020 DTG= 20
318 000040 MS= 40
319 000100 MDF= 100
320 000200 MSER= 200
321 000400 MOC= 400
322 001000 MUR= 1000
323 002000 MRD= 2000
324 004000 MCLK= 4000
325 010000 MSEN= 10000
326 020000 DTO= 20000
327 040000 OBEN=40000
328 100000 OBCK= 100000

```

```

329 ;ATTENTION SUMMARY PSEUDO-REGISTER (RMAS) (#04)
330
331
332 000001 AT0= 1 ;DEVICE 0 (BIT #0)
333 000002 AT1= 2 ;DEVICE 1 (BIT #1)
334 000004 AT2= 4 ;DEVICE 2 (BIT #2)
335 000010 AT3= 10 ;DEVICE 3 (BIT #3)
336 000020 AT4= 20 ;DEVICE 4 (BIT #4)
337 000040 AT5= 40 ;DEVICE 5 (BIT #5)
338 000100 AT6= 100 ;DEVICE 6 (BIT #6)
339 000200 AT7= 200 ;DEVICE 7 (BIT #7)
340

```

```

341 ;DESIRED SECTOR/TRACK ADDRESS REGISTER (RMDA) (#05)
342 ;(EACH BIT IS CALLED BY BIT NUMBER)
343

```

```

344 ;DRIVE TYPE REGISTER (RMDT) (#06)
345
346 000001 DT00= 1 ;DRIVE TYPE NUMBER BIT 1
347 000002 DT01= 2 ;DRIVE TYPE NUMBER BIT 2

```

```

348      000004      DT02= 4      ;DRIVE TYPE NUMBER BIT 3
349      000010      DT03= 10     ;DRIVE TYPE NUMBER BIT 4
350      000020      DT04= 20     ;DRIVE TYPE NUMBER BIT 5
351      000040      DT05= 40     ;DRIVE TYPE NUMBER BIT 6
352      000100      DT06= 100    ;DRIVE TYPE NUMBER BIT 7
353      000200      DT07= 200    ;DRIVE TYPE NUMBER BIT 8
354      000400      DT08= 400    ;DRIVE TYPE NUMBER BIT 9
355      004000      DRQ= 4000   ;DRIVE REQUEST REQUIRED (BIT #11)
356      020000      MOH= 20000  ;MOVING HEAD (BIT #13)
357      040000      TAP= 40000  ;TAPE DRIVE (BIT #14)
358      100000      NBA= 100000 ;NOT BLOCK ADDRESSED (BIT #15)
359
360      ;LOOK-AHEAD REGISTER (RMLA) (#07)
361
362
363      000100      SC0= 100    ;SECTOR COUNT FIELD 0 (BIT #6)
364      000200      SC1= 200    ;SECTOR COUNT FIELD 1 (BIT #7)
365      000400      SC2= 400    ;SECTOR COUNT FIELD 2 (BIT #8)
366      ;SC3= 1000  ;SECTOR COUNT FIELD 3 (BIT #9)
367      ;SC4= 2000  ;SECTOR COUNT FIELD 4 (BIT #10)
368
369
370      ;RMO3 MAINTAINABILITY REGISTER #2 (RMMR2) (#10)
371
372
373
374
375
376
377
378
379
380
381
382      ;RMO3 ERROR REGISTER #02 (RMMR2) (#10)
383
384
385
386
387
388
389
390
391
392
393
394      ;OFFSET REGISTER (RMOF) (#11)
395
396      000001      OFFDIR= 1    ;RMO3 OFFSET DIRECTION
397      002000      HCI= 2000  ;HEADER COMPARE INHIBIT (BIT #10)
398      004000      ECI= 4000  ;ERROR CORRECTION CODE INHIBIT (BIT #11)
399      010000      FMT22= 10000 ;FORMAT BIT (BIT #12)
400
401      ;DESIRED CYLINDER ADDRESS (RMDC) (#12)
402      ;(EACH BIT IS CALLED BY BIT NUMBER)
403

```

```

404 ;CURRENT CYLINDER ADDRESS (RMHR) (#13)
405 ;(EACH BIT IS CALLED BY BIT NUMBER)
406
407 ;SERIAL NUMBER REGISTER (RMSN) (#14)
408 ;(EACH IS CALLED BY BIT NUMBER)
409
410
411 ;RMO3 ERROR REGISTER #02 (RMER2) (#15)
412
413
414
415 020000 OPE= 20000 ;OPERATOR PLUG ERROR (BIT #13)
416 040000 SKI= 40000 ;SEEK INCOMPLETE (BIT #15)
417
418 ;ECC POSITION REGISTER (RMEC1) (#16)
419 ;(EACH BIT IS CALLED BY BIT NUMBER)
420
421 ;ECC PATTERN REGISTER (RMEC2) (#17)
422 ;(EACH BIT IS CALLED BY BIT NUMBER)
423
424 ;*****
425
426 ;OP CODE DEFINITIONS
427 000101 NOOP=101
428 000103 UNLOAD=103
429 000105 SEEK=105
430 000107 RECAL=107
431 000111 DRVCLR=111
432 000113 RELEASE=113
433 000115 OFFSET=115
434 000117 RTC=117
435 000121 READIN=121
436 000123 PACK=123
437 000131 SEARCH=131
438 000151 WRCKD=151
439 000153 WRCKHD=153
440 000161 WRITE=161
441 000163 WRTHD=163
442 000171 READ=171
443 000173 READHD=173
444 000141 GETREG=141
445 000143 SETFORM=143
446 000145 SELDRV=145
447
448 ;OTHER EQUATES
449
450 177400 SCTRWC = -256. ;WORD COUNT FOR SECTOR
451 010000 FMT22=10000 ;FORMAT 22 BIT
452

```

453  
454  
455  
456  
457  
458  
459 001100  
460 001100  
461 001100 000000  
462 001102 000  
463 001103 000  
464 001104 000000  
465 001106 000000  
466 001110 000000  
467 001112 000000  
468 001114 000  
469 001115 001  
470 001116 000000  
471 001120 000000  
472 001122 000000  
473 001124 000000  
474 001126 000000  
475 001130 000000  
476 001132 000000  
477 001134 000  
478 001135 000  
479 001136 000000  
480 001140 177570  
481 001142 177570  
482 001144 177560  
483 001146 177562  
484 001150 177564  
485 001152 177566  
486 001154 000  
487 001155 002  
488 001156 012  
489 001157 000  
490 001160 000000  
491  
492 001162 000000  
493 001164 000000  
494 001166 000000  
495 001170 000000  
496 001172 000000  
497 001174 000000  
498 001176 000000  
499 001200 000000  
500 001202 000000  
501 001204 000000  
502 001206 000000  
503 001210 177607 000377  
504 001214 077  
505 001215 015  
506 001216 000012  
507  
508 000015

.SBTTL COMMON TAGS

\*\*\*\*\*  
; THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS  
; USED IN THE PROGRAM.

. = 1100

\$CMTAG:	.WORD	0	START OF COMMON TAGS
\$PASS:	.WORD	0	CONTAINS PASS COUNT
\$TSTNM:	.BYTE	0	CONTAINS THE TEST NUMBER
\$ERFLG:	.BYTE	0	CONTAINS ERROR FLAG
\$ICNT:	.WORD	0	CONTAINS SUBTEST ITERATION COUNT
\$LPAOR:	.WORD	0	CONTAINS SCOPE LOOP ADDRESS
\$LPERR:	.WORD	0	CONTAINS SCOPE RETURN FOR ERRORS
\$ERTTL:	.WORD	0	CONTAINS TOTAL ERRORS DETECTED
\$ITEMB:	.BYTE	0	CONTAINS ITEM CONTROL BYTE
\$ERMAX:	.BYTE	1	CONTAINS MAX. ERRORS PER TEST
\$ERRPC:	.WORD	0	CONTAINS PC OF LAST ERROR INSTRUCTION
\$GDADR:	.WORD	0	CONTAINS ADDRESS OF 'GOOD' DATA
\$BDADR:	.WORD	0	CONTAINS ADDRESS OF 'BAD' DATA
\$GDDAT:	.WORD	0	CONTAINS 'GOOD' DATA
\$BDDAT:	.WORD	0	CONTAINS 'BAD' DATA
	.WORD	0	RESERVED--NOT TO BE USED
	.WORD	0	
\$AUTOB:	.BYTE	0	AUTOMATIC MODE INDICATOR
\$INTAG:	.BYTE	0	INTERRUPT MODE INDICATOR
	.WORD	0	
\$SWR:	.WORD	DSWR	ADDRESS OF SWITCH REGISTER
\$DISPLAY:	.WORD	DDISP	ADDRESS OF DISPLAY REGISTER
\$TKS:	177560		TTY KBD STATUS
\$TKB:	177562		TTY KBD BUFFER
\$TPS:	177564		TTY PRINTER STATUS REG. ADDRESS
\$TPB:	177566		TTY PRINTER BUFFER REG. ADDRESS
\$NULL:	.BYTE	0	CONTAINS NULL CHARACTER FOR FILLS
\$FILLS:	.BYTE	2	CONTAINS # OF FILLER CHARACTERS REQUIRED
\$FILLC:	.BYTE	12	INSERT FILL CHARS. AFTER A "LINE FEED"
\$TPFLG:	.BYTE	0	"TERMINAL AVAILABLE" FLAG (BIT(07)=0=YES)
\$REGAD:	.WORD	0	CONTAINS THE ADDRESS FROM WHICH (\$REGO) WAS OBTAINED
\$REGO:	.WORD	0	CONTAINS (( \$REGAD )+0)
\$REG1:	.WORD	0	CONTAINS (( \$REGAD )+2)
\$REG2:	.WORD	0	CONTAINS (( \$REGAD )+4)
\$REG3:	.WORD	0	CONTAINS (( \$REGAD )+6)
\$REG4:	.WORD	0	CONTAINS (( \$REGAD )+10)
\$REG5:	.WORD	0	CONTAINS (( \$REGAD )+12)
\$TMP0:	.WORD	0	USER DEFINED
\$TMP1:	.WORD	0	USER DEFINED
\$TMP2:	.WORD	0	USER DEFINED
\$TIMES:	0		MAX. NUMBER OF ITERATIONS
\$ESCAPE:	0		ESCAPE ON ERROR ADDRESS
\$BELL:	.ASCIZ	<207><377><377>	CODE FOR BELL
\$QUES:	.ASCII	/?/	QUESTION MARK
\$CRLF:	.ASCII	<15>	CARRIAGE RETURN
\$LF:	.ASCIZ	<12>	LINE FEED
*****			
CR	=	15	

509		000012		LF	=	12		
510	001220	000000		C.SWR:	.WORD	0		:CONTROL SWITCHES
511	001222	000000		SAVCSW:	.WORD	0		:PREVIOUS CONTENTS OF 'C.SWR'
512	001224	000000		CNTRLC:	.WORD	0		:CONTROL "C" FLAG
513	001226	000000		BUSADR:	.WORD	0		:GET ADDRESSES FROM THE TTY FLAG (0=NO, -1=YES)
514	001230	000000		LPTAVL:	.WORD	0		:LPT AVAILABLE STATUS (0=NO, 1=YES)
515	001232	000000		DRVSEL:	.WORD	0		:DRIVES SELECTED FOR TESTING
516	001234	176777	000003	TSTNMS:	.WORD	176777,3		:DEFAULT IS NO; TESTS 0-10 & 12-21
517	001240	000000	000000	OPNFLG:	.WORD	0,0		:MODIFY TEST PARAMETER FLAGS
518	001244	000000		CLKSTA:	.WORD	0		:CLOCK STATUS (0=NO CLOCK,+1=KW11-P, AND -1=KW11-L)
519								:16 MILLISECONDS PER CLOCK TICK
520	001246	000020		TICKMS:	.WORD	16.		:16666 MICROSECONDS PER CLOCK TICK
521	001250	040432		TICKUS:	.WORD	16666.		
522	001252	000000		BYPASS:	.WORD	0		
523	001254	000000		CHKDRV:	.WORD	0		:DRIVE UNDER TEST
524	001256	000000		DRVMASK:	.WORD	0		:DRIVE MASK BIT
525	001260	000000		SVSTAT:	.WORD	0		:STATUS/ERROR INDICATOR IS SAVED HERE ON AN ERROR
526								:CYLINDER READ
527	001262	000000		CYL.RD:	.WORD	0		:TRACK READ
528	001264	000000		TRK.RD:	.WORD	0		:SECTOR READ
529	001266	000000		SEC.RD:	.WORD	0		:CYLINDER DESIRED
530	001270	000000		CYL.DS:	.WORD	0		:SECTOR DESIRED
531	001272	000000		SEC.DS:	.WORD	0		:TRACK DESIRED
532	001274	000000		TRK.DS:	.WORD	0		:MINIMUM TIME
533	001276	000000		TIM.UP:	.WORD	0		:NUMBER OF COUNTS BELOW MIN. LIMIT
534	001300	000000			.WORD	0		:MAXIMUM TIME
535	001302	000000			.WORD	0		:NUMBER OF COUNTS ABOVE MAX. LIMIT
536	001304	000000			.WORD	0		:TOTAL TIME OF ALL SEEKS
537	001306	000000	000000		.WORD	0,0		:NUMBER OF SEEKS PERFORMED
538	001312	000000		TIM.DN:	.WORD	0		:MINIMUM TIME
539	001314	000000			.WORD	0		:NUMBER OF COUNTS BELOW MIN. LIMIT
540	001316	000000			.WORD	0		:MAXIMUM TIME
541	001320	000000			.WORD	0		:NUMBER OF COUNTS ABOVE MAX. LIMIT
542	001322	000000			.WORD	0		:TOTAL TIME OF ALL SEEKS
543	001324	000000	000000		.WORD	0,0		:NUMBER OF SEEKS PERFORMED
544	001330	000000		TIM.PT:	.WORD	0		:POINTS TO TABLE OF TIMES
545	001332	000000		WCEFLG:	.WORD	0		:FATAL WRITE CHECK ERROR FLAG (TEST 20)
546	001334	000000		STALLO:	.WORD	0		:VARIABLE STALL (TEST 4)
547	001336	000000		SVADR:	.WORD	0,0		:SAVE DISK ADDRESS (TEST 22)
548	001340	000000	000000	SEKTR:	.WORD	0		:SEEK TIMER (TEST 10)
549	001344	000000		SEKCN:	.WORD	0		:SEEK COUNTER
550	001346	000000		DELTA:	.WORD	0		:TESTING RANGE FOR SERVO SETTLE DOWN TEST
551	001350	000000		TRCKWC:	.WORD	-(256.*32.)		:WORD COUNT FOR A FULL TRACK IN 16 BIT MODE
552	001352	160000		STALL1:	.WORD	10.		:10 MILLISECONDS STALL (TEST 0-11)
553	001354	000012		STALL2:	.WORD	10.		:10 MILLISECONDS STALL (TEST 16-21)
554	001356	000012		STALL3:	.WORD	5000.		:5 SEC STALL (TEST 22)
555	001360	011610		MXSTAL:	.WORD	25.		:MAX. INCREMENTING STALL ALLOWED IN TEST 4
556	001362	000031		ERR.CT:	.BYTE	100.		:NUMBER OF ERRORS ALLOWED IN TESTS 16 - 21
557	001364	144						:BEFORE GOING TO THE NEXT TEST
558								:RESERVED
559	001365	000			.BYTE	0		
560								
561				:ADDRESSES AND VECTORS				
562	001366	176700		RH.ADR:	.WORD	176700		:RH70 UNIBUS ADDRESS
563	001370	000254	000240	RHVEC:	.WORD	254,5*32.		:RH70 VECTOR ADDRESS AND PRIORITY
564	001374	000104	000106	PKV:	.WORD	104,106		:KW11-P VECTOR ADDRESS

565 001400 172540  
566 001402 172542  
567 001404 172544  
568 001406 000100 000102  
569 001412 177546  
570 001414 177564  
571 001416 177566  
572 001420 177514  
573 001422 177516  
574  
575

PKCS: .WORD 172540  
PKB: .WORD 172542  
PKC: .WORD 172544  
LKV: .WORD 100,102  
LKS: .WORD 177546  
TPS: .WORD 177564  
TPB: .WORD 177566  
LPS: .WORD 177514  
LPB: .WORD 177516

;KW11-P CONTROL AND STATUS REG.  
;KW11-P COUNT SET BUFFER  
;KW11-P COUNTER  
;KW11-L VECTOR ADDRESS  
;KW11-L STATUS REGISTER  
;TTY PRINTER STATUS  
;TTY PRINTER BUFFER  
;LINE PRINTER STATUS  
;LINE PRINTER BUFFER

576 001424 000001  
577 001426 000002  
578 001430 000004  
579 001432 000010  
580 001434 000020  
581 001436 000040  
582 001440 000100  
583 001442 000200  
584 001444 000400  
585 001446 001000  
586 001450 002000  
587 001452 004000  
588 001454 010000  
589 001456 020000  
590 001460 040000  
591 001462 100000  
592 001464 000001  
593 001466 000002  
594 001470 000004  
595 001472 000010  
596 001474 000020  
597 001476 000040  
598 001500 000100  
599 001502 000200  
600  
601  
602

.BIT TABLE  
BITS: .WORD BIT00  
.WORD BIT01  
.WORD BIT02  
.WORD BIT03  
.WORD BIT04  
.WORD BIT05  
.WORD BIT06  
.WORD BIT07  
.WORD BIT08  
.WORD BIT09  
.WORD BIT10  
.WORD BIT11  
.WORD BIT12  
.WORD BIT13  
.WORD BIT14  
.WORD BIT15  
.WORD BIT00  
.WORD BIT01  
.WORD BIT02  
.WORD BIT03  
.WORD BIT04  
.WORD BIT05  
.WORD BIT06  
.WORD BIT07

603 001504 000000  
604  
605 001506 000000  
606 001510 000000  
607 001512 000000  
608 001514 000000  
609 001516 000000  
610 001520 000000  
611 001522 000000  
612 001524 000000  
613 001526 000000  
614 001530 000000  
615 001532 000000  
616 001534 000000  
617  
618  
619 001536 002330  
620 001540 002344

.COMMON STORAGE FOR TEST PARAMETER  
PRM: .WORD 0 ;THIS WORD TELLS WHICH OF THE  
;FOLLOWING PARAMETERS ARE TO BE USED  
RPT: .WORD 0 ;REPEAT COUNTS FOR ALL TESTS  
FC: .WORD 0 ;FIRST CYLINDER  
LC: .WORD 0 ;LAST CYLINDER  
IC: .WORD 0 ;INCREMENT CYLINDER  
FT: .WORD 0 ;FIRST TRACK  
LT: .WORD 0 ;LAST TRACK  
IT: .WORD 0 ;INCREMENT TRACK  
FS: .WORD 0 ;FIRST SECTOR  
LS: .WORD 0 ;LAST SECTOR  
PAT: .WORD 0 ;PATTERN CODE  
NC1: .WORD 0 ;NEW CYLINDER ADDRESS  
NC2: .WORD 0 ;NEW CYLINDER ADDRESS

.TABLE OF PARAMETER POINTERS  
PRMPT: .WORD PRM0  
.WORD PRM1

621	001542	002372	.WORD	PRM2	
622	001544	002414	.WORD	PRM3	
623	001546	002436	.WORD	PRM4	
624	001550	002460	.WORD	PRM5	
625	001552	002502	.WORD	PRM6	
626	001554	002524	.WORD	PRM7	
627	001556	002544	.WORD	PRM10	
628	001560	002564	.WORD	PRM11	
629	001562	002606	.WORD	PRM12	
630	001564	002622	.WORD	PRM13	
631	001566	002636	.WORD	PRM14	
632	001570	002652	.WORD	PRM15	
633	001572	002666	.WORD	PRM16	
634	001574	002700	.WORD	PRM17	
635	001576	002712	.WORD	PRM20	
636	001600	002744	.WORD	PRM21	
637	001602	002760	.WORD	PRM22	
638	001604	000000	.WORD	0	; TERMINATOR

```

;TABLE OF PARAMETER UPPER LIMITS
PRMLMT: .WORD 32767 ;"R"
        .WORD 822. ;"FC"
        .WORD 822. ;"LC"
        .WORD 822. ;"FC'"
        .WORD 822. ;"LC'"
        .WORD 822. ;"IC"
        .WORD 4 ;"FT"
        .WORD 4 ;"LT"
        .WORD 4 ;"IT"
        .WORD 31. ;"FS"
        .WORD 31. ;"LS"
        .WORD 177777 ;"PAT"
    
```

```

;TABLE OF MESSAGE POINTERS
PRMMSG: .WORD MSG.R
        .WORD MSG.FC
        .WORD MSG.LC
        .WORD MSGFCP
        .WORD MSG.LCP
        .WORD MSG.IC
        .WORD MSG.FT
        .WORD MSG.LT
        .WORD MSG.IT
        .WORD MSG.FS
        .WORD MSG.LS
    
```

```

;STATUS/ERROR INDICATOR MESSAGES POINTER TABLE
;DEFAULT VALUES OF TEST PARAMETERS
DFLT: .WORD 1125,200.,410.,822.,0,0 ;RECAL/SEEK (T0)
      .WORD 3377,100.,0,128.,0,256.,0,0,0,0,0 ;SEEK/SEEK (T1)
    
```



677	001722	000000	000000				
678	001726	001177	000001	000000	.WORD	1177,1,0,410.,0,822.,1,0,0	;INCREMENT SEEK (T2)
679	001734	000632	000000	001466			
680	001742	000001	000000	000000			
681	001750	001177	000010	000000	.WORD	1177,10,0,256.,0,512.,1,0,0	;STEPPING SEEK (T3)
682	001756	000400	000000	001000			
683	001764	000001	000000	000000			
684	001772	001177	000001	000000	.WORD	1177,1,0,410.,0,822.,1,0,0	;OSCILLATING SEEK (T4)
685	002000	000632	000000	001466			
686	002006	000001	000000	000000			
687	002014	001177	000001	000000	.WORD	1177,1,0,410.,0,822.,1,0,0	;CONVERGING/DIVERGING SEEK (T5)
688	002022	000632	000000	001466			
689	002030	000001	000000	000000			
690	002036	001177	000001	000000	.WORD	1177,1,0,410.,0,822.,1,0,0	;SERVO ADDRESSING LOGIC NOISE (T6)
691	002044	000632	000000	001466			
692	002052	000001	000000	000000			
693	002060	000337	011610	000000	.WORD	337,5000.,0,410.,0,822.,0,4	;RANDOM SEEK TEST (T7)
694	002066	000632	000000	001466			
695	002074	000000	000004				
696	002100	000177	000001	000000	.WORD	177,1,0,410.,0,822.,100.,0	;SERVO SETTLE DOWN TEST (T10)
697	002106	000632	000000	001466			
698	002114	000144	000000				
699	002120	001177	000001	000000	.WORD	1177,1,0,410.,0,822.,1,0,0	;ALL SEEKS TEST (T11)
700	002126	000632	000000	001466			
701	002134	000001	000000	000000			
702	002142	001113	000001	000000	.WORD	1113,1,0,0,0,0	;ROTATIONAL SPEED TIMING TEST (T12)
703	002150	000000	000000	000000			
704	002156	000037	000001	000000	.WORD	37,1,0,410.,0,822.	;ONE CYLINDER SEEK TIMING TEST (T13)
705	002164	000632	000000	001466			
706	002172	000037	000001	000000	.WORD	37,1,0,136.,0,255.	;ACCESS TIME MEASUREMENT TEST (T14)
707	002200	000210	000000	000377			
708	002206	000037	000001	000000	.WORD	37,1,0,410.,0,822.	;MAXIMUM SEEK TIMING TEST (T15)
709	002214	000632	000000	001466			
710	002222	000113	000001	000000	.WORD	113,1,0,0,0	;SECTOR ADDRESSING TEST (T16)
711	002230	000000	000000				
712	002234	001013	000001	000000	.WORD	1013,1,0,0,0	;TRACK ADDRESSING TEST (T17)
713	002242	000000	000000				
714	002246	007777	000001	000000	.WORD	7777,1,0,410.,0,821.,64.,0,4,1,1,0,177777	;DATA TEST (T20)
715	002254	000632	000000	001466			
716	002262	000100	000000	000004			
717	002270	000001	000001	000000			
718	002276	177777					
719	002300	000037	001750	000000	.WORD	37,1000.,0,410.,0,821.	;EXERCISER (T21)
720	002306	000632	000000	001466			
721	002314	000037	011610	000000	.WORD	37,5000.,0,136.,0,255.	;RMO3 ACCESS TIME ADJUSTMENT TEST (T22)
722	002322	000210	000000	000377			

;PARAMETER TABLES

:RECAL/SEEK (T0)

PRMO:	.WORD	1125
	.WORD	200.
	.WORD	410.
	.WORD	822.

729	002330	001125
730	002332	000310
731	002334	000632
732	002336	001466

733	002340	000000	.WORD	0
734	002342	000000	.WORD	0
735				
736				
737	002344	003377	:SEEK/SEEK (T1)	
738	002346	000144	PRM1: .WORD	3377
739	002350	000000	.WORD	100.
740	002352	000200	.WORD	0
741	002354	000000	.WORD	128.
742	002356	000400	.WORD	0
743	002360	000000	.WORD	256.
744	002362	000000	.WORD	0
745	002364	000000	.WORD	0
746	002366	000000	.WORD	0
747	002370	000000	.WORD	0

748				
749			:INCREMENT SEEK (T2)	
750	002372	001177	PRM2: .WORD	1177
751	002374	000001	.WORD	1
752	002376	000000	.WORD	0
753	002400	000632	.WORD	410.
754	002402	000000	.WORD	0
755	002404	001466	.WORD	822.
756	002406	000001	.WORD	1
757	002410	000000	.WORD	0
758	002412	000000	.WORD	0

759				
760			:STEPPING SEEK (T3)	
761	002414	001177	PRM3: .WORD	1177
762	002416	000001	.WORD	1
763	002420	000000	.WORD	0
764	002422	000400	.WORD	256.
765	002424	000000	.WORD	0
766	002426	001000	.WORD	512.
767	002430	000001	.WORD	1
768	002432	000000	.WORD	0
769	002434	000000	.WORD	0

770				
771			:OSCILLATING SEEK (T4)	
772	002436	001177	PRM4: .WORD	1177
773	002440	000001	.WORD	1
774	002442	000000	.WORD	0
775	002444	000632	.WORD	410.
776	002446	000000	.WORD	0
777	002450	001466	.WORD	822.
778	002452	000001	.WORD	1
779	002454	000000	.WORD	0
780	002456	000000	.WORD	0

781				
782			:CONVERGING/DIVERGING SEEK (T5)	
783	002460	001177	PRM5: .WORD	1177
784	002462	000001	.WORD	1
785	002464	000000	.WORD	0
786	002466	000632	.WORD	410.
787	002470	000000	.WORD	0
788	002472	001466	.WORD	822.

789	002474	000001	.WORD	1
790	002476	000000	.WORD	0
791	002500	000000	.WORD	0
792				
793				
794	002502	001177	.SERVO ADDRESSING LOGIC NOISE GENERATOR (T6)	
795	002504	000001	PRM6: .WORD	1177
796	002506	000000	.WORD	1
797	002510	000632	.WORD	0
798	002512	000000	.WORD	410.
799	002514	001466	.WORD	0
800	002516	000001	.WORD	822.
801	002520	000000	.WORD	1
802	002522	000000	.WORD	0
803				
804				
805	002524	000337	.RANDOM SEEK TEST (T7)	
806	002526	011610	PRM7: .WORD	337
807	002530	000000	.WORD	5000.
808	002532	000632	.WORD	0
809	002534	000000	.WORD	410.
810	002536	001466	.WORD	0
811	002540	000000	.WORD	822.
812	002542	000004	.WORD	0
813				
814				
815	002544	000177	.SERVO SETTLE DOWN TEST (T10)	
816	002546	000001	PRM10: .WORD	177
817	002550	000000	.WORD	1
818	002552	000632	.WORD	0
819	002554	000000	.WORD	410.
820	002556	001466	.WORD	0
821	002560	000144	.WORD	822.
822	002562	000000	.WORD	100.
823				
824				
825	002564	001177	.ALL SEEKS TEST (T11)	
826	002566	000001	PRM11: .WORD	1177
827	002570	000000	.WORD	1
828	002572	000632	.WORD	0
829	002574	000000	.WORD	410.
830	002576	001466	.WORD	0
831	002600	000001	.WORD	822.
832	002602	000000	.WORD	1
833	002604	000000	.WORD	0
834				
835				
836	002606	001113	.ROTATIONAL SPEED TIMING TEST (T12)	
837	002610	000001	PRM12: .WORD	1113
838	002612	000000	.WORD	1
839	002614	000000	.WORD	0
840	002616	000000	.WORD	0
841	002620	000000	.WORD	0
842				
843				
844	002622	000037	.ONE CYLINDER SEEK TIMING TEST (T13)	
			PRM13: .WORD	37

845 002624 000001  
846 002626 000000  
847 002630 000632  
848 002632 000000  
849 002634 001466

.WORD 1  
.WORD 0  
.WORD 410.  
.WORD 0  
.WORD 822.

: ACCESS TIME MEASUREMENT TEST (T14)

851  
852 002636 000037  
853 002640 000001  
854 002642 000000  
855 002644 000210  
856 002646 000000  
857 002650 000377

PRM14: .WORD 37  
.WORD 1  
.WORD 0  
.WORD 136.  
.WORD 0  
.WORD 255.

: MAXIMUM SEEK TIMING TEST (T15)

858  
859  
860 002652 000037  
861 002654 000001  
862 002656 000000  
863 002660 000632  
864 002662 000000  
865 002664 001466

PRM15: .WORD 37  
.WORD 1  
.WORD 0  
.WORD 410.  
.WORD 0  
.WORD 822.

: SECTOR ADDRESSING TEST (T16)

866  
867  
868 002666 000113  
869 002670 000001  
870 002672 000000  
871 002674 000000  
872 002676 000000

PRM16: .WORD 113  
.WORD 1  
.WORD 0  
.WORD 0  
.WORD 0

: TRACK ADDRESSING TEST (T17)

873  
874  
875 002700 001013  
876 002702 000001  
877 002704 000000  
878 002706 000000  
879 002710 000000

PRM17: .WORD 1013  
.WORD 1  
.WORD 0  
.WORD 0  
.WORD 0

: DATA TEST (T20)

880  
881  
882 002712 007777  
883 002714 000001  
884 002716 000000  
885 002720 000632  
886 002722 000000  
887 002724 001465  
888 002726 000100  
889 002730 000000  
890 002732 000004  
891 002734 000001  
892 002736 000001  
893 002740 000000  
894 002742 177777

PRM20: .WORD 7777  
.WORD 1  
.WORD 0  
.WORD 410.  
.WORD 0  
.WORD 821.  
.WORD 64.  
.WORD 0  
.WORD 4  
.WORD 1  
.WORD 1  
.WORD 0  
PTRN15: .WORD 177777

: EXERCISER (T21)

895  
896  
897 002744 000037  
898 002746 001750  
899 002750 000000  
900 002752 000632

PRM21: .WORD 37  
.WORD 1000.  
.WORD 0  
.WORD 410.

901 002754 000000  
 902 002756 001465  
 903  
 904  
 905 002760 000037  
 906 002762 011610  
 907 002764 000000  
 908 002766 000210  
 909 002770 000000  
 910 002772 000377  
 911  
 912  
 913  
 914  
 915  
 916 002774 044236  
 917 002776 000000  
 918 003000 003142  
 919 003002 003244  
 920  
 921 003004 044236  
 922 003006 000000  
 923 003010 003131  
 924 003012 003255  
 925  
 926 003014 044270  
 927 003016 044337  
 928 003020 000000  
 929 003022 001274  
 930  
 931 003024 044354  
 932 003026 044423  
 933 003030 000000  
 934 003032 006200  
 935  
 936 003034 044440  
 937 003036 044502  
 938 003040 000000  
 939 003042 012574  
 940  
 941 003044 003104  
 942 003046 003144  
 943 003050 003204  
 944 003052 003244  
 945 003054 003304  
 946 003056 003344  
 947 003060 003404  
 948 003062 003444  
 949 003064 003504  
 950 003066 003544  
 951 003070 003604  
 952 003072 003644  
 953 003074 003704  
 954 003076 003744  
 955 003100 004004  
 956 003102 004044

.WORD 0  
 .WORD 821.  
 ;RM03 ACCESS TIME ADJUSTMENT TEST (T22)  
 PRM22: .WORD 37  
 .WORD 5000.  
 .WORD 0  
 .WORD 136.  
 .WORD 0  
 .WORD 255.  
 ;TIMING LIMITS  
 T7A: .WORD MSG7  
 .WORD 0  
 .WORD 1634. ;(16.67-2%)\*2  
 .WORD 1700. ;(16.67+2%)\*2  
 T7B: .WORD MSG7  
 .WORD 0  
 .WORD 1625. ;(16.67-2.5%)\*2  
 .WORD 1709. ;(16.67+2.5%)\*2  
 T10: .WORD MSG10A  
 .WORD MSG10B  
 .WORD 0 ;NO LOWER LIMIT  
 .WORD 700. ;(7)\*2  
 T11: .WORD MSG11A  
 .WORD MSG11B  
 .WORD 0 ;NO LOWER LIMIT  
 .WORD 3200. ;(32)\*2  
 T12: .WORD MSG12A  
 .WORD MSG12B  
 .WORD 0 ;NO LOWER LIMIT  
 .WORD 5500. ;(55)\*2  
 PAT.PT: .WORD PAT0 ;TABLE OF POINTERS WHICH POINT TO THE  
 .WORD PAT1 ;PATTERNS USED BY THE DATA TEST  
 .WORD PAT2  
 .WORD PAT3  
 .WORD PAT4  
 .WORD PAT5  
 .WORD PAT6  
 .WORD PAT7  
 .WORD PAT8  
 .WORD PAT9  
 .WORD PAT10  
 .WORD PAT11  
 .WORD PAT12  
 .WORD PAT13  
 .WORD PAT14  
 .WORD PAT15

```

957
958
959
960 003104 066666 ; PATTERNS 0 THRU 15
961 003106 066666 PAT0: .WORD 066666 ; PATTERN 0
962 003110 066666 .WORD 066666
963 003112 066666 .WORD 066666
964 003114 066666 .WORD 066666
965 003116 066666 .WORD 066666
966 003120 066666 .WORD 066666
967 003122 066666 .WORD 066666
968 003124 066666 .WORD 066666
969 003126 066666 .WORD 066666
970 003128 066666 .WORD 066666
971 003130 066666 .WORD 066666
972 003132 066666 .WORD 066666
973 003134 066666 .WORD 066666
974 003136 066666 .WORD 066666
975 003140 066666 .WORD 066666
976 003142 066666 .WORD 066666
977 003144 000001 PAT1: .WORD 000001 ; PATTERN 1
978 003146 000003 .WORD 000003
979 003150 000007 .WORD 000007
980 003152 000017 .WORD 000017
981 003154 000037 .WORD 000037
982 003156 000077 .WORD 000077
983 003160 000177 .WORD 000177
984 003162 000377 .WORD 000377
985 003164 000777 .WORD 000777
986 003166 001777 .WORD 001777
987 003170 003777 .WORD 003777
988 003172 007777 .WORD 007777
989 003174 017777 .WORD 017777
990 003176 037777 .WORD 037777
991 003200 077777 .WORD 077777
992 003202 177777 .WORD 177777
993
994 003204 177776 PAT2: .WORD 177776 ; PATTERN 2
995 003206 177774 .WORD 177774
996 003210 177770 .WORD 177770
997 003212 177760 .WORD 177760
998 003214 177740 .WORD 177740
999 003216 177700 .WORD 177700
1000 003220 177600 .WORD 177600
1001 003222 177400 .WORD 177400
1002 003224 177000 .WORD 177000
1003 003226 176000 .WORD 176000
1004 003230 174000 .WORD 174000
1005 003232 170000 .WORD 170000
1006 003234 160000 .WORD 160000
1007 003236 140000 .WORD 140000
1008 003240 100000 .WORD 100000
1009 003242 000000 .WORD 000000
1010
1011 003244 000000 PAT3: .WORD 000000 ; PATTERN 3
1012 003246 000000 .WORD 000000

```

1013	003250	000000	.WORD	000000
1014	003252	177777	.WORD	177777
1015	003254	177777	.WORD	177777
1016	003256	177777	.WORD	177777
1017	003260	000000	.WORD	000000
1018	003262	000000	.WORD	000000
1019	003264	177777	.WORD	177777
1020	003266	177777	.WORD	177777
1021	003270	000000	.WORD	000000
1022	003272	177777	.WORD	177777
1023	003274	000000	.WORD	000000
1024	003276	177777	.WORD	177777
1025	003300	000000	.WORD	000000
1026	003302	177777	.WORD	177777

1027				
1028	003304	000000	PAT4: .WORD	000000 ;PATTERN 4
1029	003306	010421	.WORD	010421
1030	003310	021042	.WORD	021042
1031	003312	031463	.WORD	031463
1032	003314	042104	.WORD	042104
1033	003316	052525	.WORD	052525
1034	003320	063146	.WORD	063146
1035	003322	073567	.WORD	073567
1036	003324	104210	.WORD	104210
1037	003326	114631	.WORD	114631
1038	003330	125252	.WORD	125252
1039	003332	135673	.WORD	135673
1040	003334	146314	.WORD	146314
1041	003336	156735	.WORD	156735
1042	003340	167356	.WORD	167356
1043	003342	177777	.WORD	177777

1044				
1045	003344	052525	PAT5: .WORD	052525 ;PATTERN 5
1046	003346	052525	.WORD	052525
1047	003350	052525	.WORD	052525
1048	003352	125252	.WORD	125252
1049	003354	125252	.WORD	125252
1050	003356	125252	.WORD	125252
1051	003360	052525	.WORD	052525
1052	003362	052525	.WORD	052525
1053	003364	125252	.WORD	125252
1054	003366	125252	.WORD	125252
1055	003370	052525	.WORD	052525
1056	003372	125252	.WORD	125252
1057	003374	052525	.WORD	052525
1058	003376	125252	.WORD	125252
1059	003400	052525	.WORD	052525
1060	003402	125252	.WORD	125252

1061				
1062	003404	007417	PAT6: .WORD	007417 ;PATTERN 6
1063	003406	007417	.WORD	007417
1064	003410	007417	.WORD	007417
1065	003412	170360	.WORD	170360
1066	003414	170360	.WORD	170360
1067	003416	170360	.WORD	170360
1068	003420	007417	.WORD	007417

1069	003422	007417	.WORD	007417
1070	003424	170360	.WORD	170360
1071	003426	170360	.WORD	170360
1072	003430	007417	.WORD	007417
1073	003432	170360	.WORD	170360
1074	003434	007417	.WORD	007417
1075	003436	170360	.WORD	170360
1076	003440	007417	.WORD	007417
1077	003442	170360	.WORD	170360

1078				
1079	003444	026455	PAT7: .WORD	026455 ;PATTERN 7
1080	003446	026455	.WORD	026455
1081	003450	026455	.WORD	026455
1082	003452	151322	.WORD	151322
1083	003454	151322	.WORD	151322
1084	003456	151322	.WORD	151322
1085	003460	026455	.WORD	026455
1086	003462	026455	.WORD	026455
1087	003464	151322	.WORD	151322
1088	003466	151322	.WORD	151322
1089	003470	026455	.WORD	026455
1090	003472	151322	.WORD	151322
1091	003474	026455	.WORD	026455
1092	003476	151322	.WORD	151322
1093	003500	026455	.WORD	026455
1094	003502	151322	.WORD	151322

1095				
1096	003504	165555	PAT8: .WORD	165555 ;PATTERN 8
1097	003506	133333	.WORD	133333
1098	003510	165555	.WORD	165555
1099	003512	133333	.WORD	133333
1100	003514	165555	.WORD	165555
1101	003516	133333	.WORD	133333
1102	003520	165555	.WORD	165555
1103	003522	133333	.WORD	133333
1104	003524	165555	.WORD	165555
1105	003526	133333	.WORD	133333
1106	003530	165555	.WORD	165555
1107	003532	133333	.WORD	133333
1108	003534	165555	.WORD	165555
1109	003536	133333	.WORD	133333
1110	003540	165555	.WORD	165555
1111	003542	133333	.WORD	133333

1112				
1113	003544	000001	PAT9: .WORD	000001 ;PATTERN 9
1114	003546	000002	.WORD	000002
1115	003550	000004	.WORD	000004
1116	003552	000010	.WORD	000010
1117	003554	000020	.WORD	000020
1118	003556	000040	.WORD	000040
1119	003560	000100	.WORD	000100
1120	003562	000200	.WORD	000200
1121	003564	000400	.WORD	000400
1122	003566	001000	.WORD	001000
1123	003570	002000	.WORD	002000
1124	003572	004000	.WORD	004000



1125	003574	010000	.WORD	010000	
1126	003576	020000	.WORD	020000	
1127	003600	040000	.WORD	040000	
1128	003602	100000	.WORD	100000	
1129					
1130	003604	177776	PAT10: .WORD	177776	;PATTERN 10
1131	003606	177775	.WORD	177775	
1132	003610	177773	.WORD	177773	
1133	003612	177767	.WORD	177767	
1134	003614	177757	.WORD	177757	
1135	003616	177737	.WORD	177737	
1136	003620	177677	.WORD	177677	
1137	003622	177577	.WORD	177577	
1138	003624	177377	.WORD	177377	
1139	003626	176777	.WORD	176777	
1140	003630	175777	.WORD	175777	
1141	003632	173777	.WORD	173777	
1142	003634	167777	.WORD	167777	
1143	003636	157777	.WORD	157777	
1144	003640	137777	.WORD	137777	
1145	003642	077777	.WORD	077777	
1146					
1147	003644	172666	PAT11: .WORD	172666	;PATTERN 11
1148	003646	155555	.WORD	155555	
1149	003650	172666	.WORD	172666	
1150	003652	155555	.WORD	155555	
1151	003654	172666	.WORD	172666	
1152	003656	155555	.WORD	155555	
1153	003660	172666	.WORD	172666	
1154	003662	155555	.WORD	155555	
1155	003664	172666	.WORD	172666	
1156	003666	155555	.WORD	155555	
1157	003670	172666	.WORD	172666	
1158	003672	155555	.WORD	155555	
1159	003674	172666	.WORD	172666	
1160	003676	155555	.WORD	155555	
1161	003700	172666	.WORD	172666	
1162	003702	155555	.WORD	155555	
1163					
1164	003704	077777	PAT12: .WORD	077777	;PATTERN 12
1165	003706	137777	.WORD	137777	
1166	003710	157777	.WORD	157777	
1167	003712	167777	.WORD	167777	
1168	003714	173777	.WORD	173777	
1169	003716	175777	.WORD	175777	
1170	003720	176777	.WORD	176777	
1171	003722	177377	.WORD	177377	
1172	003724	177577	.WORD	177577	
1173	003726	177677	.WORD	177677	
1174	003730	177737	.WORD	177737	
1175	003732	177757	.WORD	177757	
1176	003734	177767	.WORD	177767	
1177	003736	177773	.WORD	177773	
1178	003740	177775	.WORD	177775	
1179	003742	177776	.WORD	177776	
1180					

1181	003744	153333	PAT13:	.WORD	153333	;PATTERN 13
1182	003746	066667		.WORD	066667	
1183	003750	153333		.WORD	153333	
1184	003752	066667		.WORD	066667	
1185	003754	153333		.WORD	153333	
1186	003756	066667		.WORD	066667	
1187	003760	153333		.WORD	153333	
1188	003762	066667		.WORD	066667	
1189	003764	153333		.WORD	153333	
1190	003766	066667		.WORD	066667	
1191	003770	153333		.WORD	153333	
1192	003772	066667		.WORD	066667	
1193	003774	153333		.WORD	153333	
1194	003776	066667		.WORD	066667	
1195	004000	153333		.WORD	153333	
1196	004002	066667		.WORD	066667	

1197						
1198	004004	000000	PAT14:	.WORD	000000	;PATTERN 14
1199	004006	177777		.WORD	177777	
1200	004010	177777		.WORD	177777	
1201	004012	177777		.WORD	177777	
1202	004014	177777		.WORD	177777	
1203	004016	177777		.WORD	177777	
1204	004020	177777		.WORD	177777	
1205	004022	177777		.WORD	177777	
1206	004024	177777		.WORD	177777	
1207	004026	177777		.WORD	177777	
1208	004030	177777		.WORD	177777	
1209	004032	177777		.WORD	177777	
1210	004034	177777		.WORD	177777	
1211	004036	177777		.WORD	177777	
1212	004040	177777		.WORD	177777	
1213	004042	177777		.WORD	177777	

1214						
1215	004044	177777	PAT15:	.WORD	177777	;PATTERN 15
1216	004046	000000		.WORD	000000	
1217	004050	000000		.WORD	000000	
1218	004052	000000		.WORD	000000	
1219	004054	000000		.WORD	000000	
1220	004056	000000		.WORD	000000	
1221	004060	000000		.WORD	000000	
1222	004062	000000		.WORD	000000	
1223	004064	000000		.WORD	000000	
1224	004066	000000		.WORD	000000	
1225	004070	000000		.WORD	000000	
1226	004072	000000		.WORD	000000	
1227	004074	000000		.WORD	000000	
1228	004076	000000		.WORD	000000	
1229	004100	000000		.WORD	000000	
1230	004102	000000		.WORD	000000	

;DPB (DATA PARAMETER BLOCK)

1231						
1232						
1233						
1234	004104	000	DPB.A:	.BYTE	0	;(0) DRIVE NUMBER
1235	004105	000		.BYTE	0	;(1) OFFSET VALUE OR FMT22, ECI, AND HCI
1236	004106	000		.BYTE	0	;(2) COMMAND

1237	004107	000	.BYTE	0	;(3) PSEL AND A17 AND A16
1238	004110	000000	.WORD	0	;(4) WORD COUNT (MUST BE NEG.)
1239	004112	050514	.WORD	BUFFER	;(6) BUFFER ADDRESS OR
1240					REGISTER TABLE POINTER
1241	004114	000	.BYTE	0	;(10) SECTOR ADDRESS OR
1242					FIRST REG. INDEX
1243	004115	000	.BYTE	0	;(11) TRACK ADDRESS OR
1244					LAST REG. INDEX
1245	004116	000000	.WORD	0	;(12) CYLINDER ADDRESS
1246	004120	004204	.WORD	RM.REG	;(14) ERROR TABLE POINTER
1247					POINTS TO THE FIRST OF TWENTY
1248					LOCATIONS OF WHERE THE DRIVER
1249					IS TO STORE THE RH70/RMO3
1250					REGISTERS ON AN ERROR. IF LEFT
1251					ZERO REGISTERS ARE NOT SAVED.
1252	004122	000000	.WORD	0	;(16) STATUS/ERROR INDICATOR
1253					BIT15=1=>ERROR OCCURRED
1254					BIT07=1=>DONE
1255					BIT14-BIT09 AND BIT06-BIT03
1256					INDICATE TYPE OF ERROR
1257					
1258	004124	000	DPB.B: .BYTE	0	;(0) DRIVE NUMBER
1259	004125	000	.BYTE	0	;(1) OFFSET VALUE OR FMT22, ECI, AND HCI
1260	004126	000	.BYTE	0	;(2) COMMAND
1261	004127	000	.BYTE	0	;(3) PSEL AND A17 AND A16
1262	004130	177776	.WORD	-2	;(4) WORD COUNT (MUST BE NEG.)
1263	004132	050514	.WORD	BUFFER	;(6) BUFFER ADDRESS OR
1264					REGISTER TABLE POINTER
1265	004134	000	.BYTE	0	;(10) SECTOR ADDRESS OR
1266					FIRST REG. INDEX
1267	004135	000	.BYTE	0	;(11) TRACK ADDRESS OR
1268					LAST REG. INDEX
1269	004136	000000	.WORD	0	;(12) CYLINDER ADDRESS
1270	004140	004204	.WORD	RM.REG	;(14) ERROR TABLE POINTER
1271					POINTS TO THE FIRST OF TWENTY
1272					LOCATIONS OF WHERE THE DRIVER
1273					IS TO STORE THE RH70/RMO3
1274					REGISTERS ON AN ERROR. IF LEFT
1275					ZERO REGISTERS ARE NOT SAVED.
1276	004142	000000	.WORD	0	;(16) STATUS/ERROR INDICATOR
1277					BIT15=1=>ERROR OCCURRED
1278					BIT07=1=>DONE
1279					BIT14-BIT09 AND BIT06-BIT03
1280					INDICATE TYPE OF ERROR
1281					
1282	004144	000	DPB.C: .BYTE	0	;(0) DRIVE NUMBER
1283	004145	000	.BYTE	0	;(1) OFFSET VALUE OR FMT22, ECI, AND HCI
1284	004146	000	.BYTE	0	;(2) COMMAND
1285	004147	000	.BYTE	0	;(3) PSEL AND A17 AND A16
1286	004150	177776	.WORD	-2	;(4) WORD COUNT (MUST BE NEG.)
1287	004152	050514	.WORD	BUFFER	;(6) BUFFER ADDRESS OR
1288					REGISTER TABLE POINTER
1289	004154	000	.BYTE	0	;(10) SECTOR ADDRESS OR
1290					FIRST REG. INDEX
1291	004155	000	.BYTE	0	;(11) TRACK ADDRESS OR
1292					LAST REG. INDEX

1293	004156	000000	.WORD	0	;(12) CYLINDER ADDRESS
1294	004160	004204	.WORD	RM.REG	;(14) ERROR TABLE POINTER
1295					POINTS TO THE FIRST OF TWENTY
1296					LOCATIONS OF WHERE THE DRIVER
1297					IS TO STORE THE RH70/RM03
1298					REGISTERS ON AN ERROR. IF LEFT
1299					ZERO REGISTERS ARE NOT SAVED.
1300	004162	000000	.WORD	0	;(16) STATUS/ERROR INDICATOR
1301					BIT15=1=>ERROR OCCURRED
1302					BIT07=1=>DONE
1303					BIT14-BIT09 AND BIT06-BIT03
1304					INDICATE TYPE OF ERROR
1305					
1306	004164	000	DTADPB: .BYTE	0	;(0) DRIVE NUMBER
1307	004165	000	.BYTE	0	;(1) OFFSET VALUE OR FMT22, ECT, AND HCI
1308	004166	000	.BYTE	0	;(2) COMMAND
1309	004167	000	.BYTE	0	;(3) PSEL AND A17 AND A16
1310	004170	000000	.WORD	0	;(4) WORD COUNT (MUST BE NEG.)
1311	004172	050514	.WORD	BUFFER	;(6) BUFFER ADDRESS OR
1312					REGISTER TABLE POINTER
1313	004174	000	.BYTE	0	;(10) SECTOR ADDRESS OR
1314					FIRST REG. INDEX
1315	004175	000	.BYTE	0	;(11) TRACK ADDRESS OR
1316					LAST REG. INDEX
1317	004176	000000	.WORD	0	;(12) CYLINDER ADDRESS
1318	004200	004204	.WORD	RM.REG	;(14) ERROR TABLE POINTER
1319					POINTS TO THE FIRST OF TWENTY
1320					LOCATIONS OF WHERE THE DRIVER
1321					IS TO STORE THE RH70/RM03
1322					REGISTERS ON AN ERROR. IF LEFT
1323					ZERO REGISTERS ARE NOT SAVED.
1324	004202	000000	.WORD	0	;(16) STATUS/ERROR INDICATOR
1325					BIT15=1=>ERROR OCCURRED
1326					BIT07=1=>DONE
1327					BIT14-BIT09 AND BIT06-BIT03
1328					INDICATE TYPE OF ERROR
1329					
1330					
1331					
1332					
1333					;SAVE RH70/RM03 REGISTERS HERE ON ERROR
1334	004204	000000	RM.REG: .WORD	0	;RMCS1 (776700) CONTROL & STATUS #1
1335	004206	000000	.WORD	0	;RMWC (776702) WORD COUNT
1336	004210	000000	.WORD	0	;RMB A (776704) BUS ADDRESS
1337	004212	000000	.WORD	0	;RMD A (776706) DESIRED SECTOR/TRACK
1338	004214	000000	.WORD	0	;RMCS2 (776710) CONTROL & STATUS #2
1339	004216	000000	.WORD	0	;RMD S (776712) DISK STATUS
1340	004220	000000	.WORD	0	;RMR1 (776714) ERROR REG. #1
1341	004222	000000	.WORD	0	;RMA S (776716) ATTENTION SUMMARY
1342	004224	000000	.WORD	0	;RML A (776720) LOOK AHEAD
1343	004226	000000	.WORD	0	;RMD B (776722) DATA BUFFER
1344	004230	000000	.WORD	0	;RMR1 (776724) MAINTAINABILITY
1345	004232	000000	.WORD	0	;RMD T (776726) DRIVE TYPE
1346	004234	000000	.WORD	0	;RMS N (776730) SERIAL NUMBER
1347	004236	000000	.WORD	0	;RMO F (776732) OFFSET
1348	004240	000000	.WORD	0	;RMD C (776734) DESIRED CYLINDER

1349	004242	000000	.WORD	0	;RMHR (776736) CURRENT CYLINDER
1350	004244	000000	.WORD	0	;RMR2 (776740) ERROR REG #2
1351	004246	000000	.WORD	0	;RMR2 (776742) ERROR REG #3
1352	004250	000000	.WORD	0	;RMEC1 (776744) ECC POSITION
1353	004252	000000	.WORD	0	;RMEC2 (776746) ECC PATTERN

1354					
1355					
1356	004254	045531	.STATUS/ERROR	MESSAGE POINTER TABLE	
1357	004256	045573	STATBL: .WORD	MSG814	;OFFLINE OR UNSAFE DRIVE REQUESTED
1358	004260	045624	.WORD	MSG813	;UNLOAD DRIVE REQUESTED
1359	004262	045646	.WORD	MSG812	;PERSISTENT UNSAFE
1360	004264	045674	.WORD	MSG811	;PARITY ERROR OCCURRED
1361	004266	045717	.WORD	MSG810	;FATAL PARITY ERROR
1362	004270	045756	.WORD	MSG809	;SOFTWARE TIMEOUT ON THIS DRIVE
1363	004272	046020	.WORD	MSG808	;SOFTWARE TIMEOUT ON ANOTHER DRIVE
1364	004274	046064	.WORD	MSG806	;ERROR OCCURRED DURING I/O OPERATION
1365	004276	046134	.WORD	MSG805	;ERROR OCCURRED DURING NON-I/O OPERATION
1366	004300	046154	.WORD	MSG804	;UNSAFE OCCURRED
1367	004302	046224	.WORD	MSG803	;AUTOMATIC RECALIBRATE SEQUENCE OCCURRED
1368	004304	046274	.WORD	MSG802	;DRIVE HAS NOT RESPONDED TO PORT REQUEST
			.WORD	MSG801	;DRIVE HAS BECOME NONEXISTENT

.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  
;\* RH7D INTERRUPT OCCURED (RMAS = 0)  
;\* ERR PC RMAS  
;\* \$ERRPC \$REG3

EM1  
DH1  
DT1  
DF1

;\* ERROR ITEM 2  
;\* UNEXPECTED ATTENTION OCCURRED  
;\* ERR PC DRIVE RMAS RMD5 RMER1 RMMR2 RMER2  
;\* \$ERRPC \$REG1 \$REG3 RMERRS RMERRS+2 RMERRS+4 RMERRS+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

;\* ERROR ITEM 4  
;\* MASSBUS PARITY ERROR (PAR=1)  
;\* TEST ERR PC ADDRESS GDDATA BDDATA

1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383 004306  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397 004306 044662  
1398 004310 046332  
1399 004312 047706  
1400 004314 050340  
1401  
1402  
1403  
1404  
1405  
1406  
1407 004316 044725  
1408 004320 046347  
1409 004322 047712  
1410 004324 050344  
1411  
1412  
1413  
1414  
1415  
1416  
1417 004326 044763  
1418 004330 046434  
1419 004332 047730  
1420 004334 050350  
1421  
1422  
1423  
1424

1425			;	*	\$TMPD	\$ERRPC	WRT.ADR	WRT.WD	RD.WRD		
1426											
1427	004336	045020				EM4					
1428	004340	046471				DH4					
1429	004342	047740				DT4					
1430	004344	050354				DF4					
1431											
1432			;	*	ERROR ITEM 5						
1433			;	*	ADDRESS PLUG CHANGE BIT SET						
1434			;	*	ERR PC	DRIVE	RMAS	RMDS	RMER1	RMMR2	RMER2
1435			;	*	\$ERRPC	\$REG1	\$REG3	RMERRS	RMERRS+2	RMERRS+4	RMERRS+6
1436											
1437	004346	045054				EM5					
1438	004350	046347				DH2					
1439	004352	047712				DT2					
1440	004354	050344				DF2					
1441											
1442			;	*	ERROR ITEM 6 -- NOT USED						
1443											
1444	004356	000000				0					
1445	004360	000000				0					
1446	004362	000000				0					
1447	004364	000000				0					
1448											
1449			;	*	ERROR ITEM 7 -- NOT USED						
1450											
1451	004366	000000				0					
1452	004370	000000				0					
1453	004372	000000				0					
1454	004374	000000				0					
1455											
1456			;	*	ERROR ITEM 10						
1457			;	*	RH70/RMO3 FAILED TO RESPOND TO ADDRESSING						
1458			;	*	RMCS1	ERR PC					
1459			;	*	RH.ADR	\$ERRPC					
1460											
1461	004376	045110				EM10					
1462	004400	046540				DH10					
1463	004402	047772				DT10					
1464	004404	050360				DF10					
1465											
1466			;	*	ERROR ITEM 11						
1467			;	*	DRIVE SELECTED IS NOT ONLINE						
1468			;	*	DRIVE	ERR PC					
1469			;	*	\$REG2	\$ERRPC					
1470											
1471	004406	045162				EM11					
1472	004410	046557				DH11					
1473	004412	047776				DT11					
1474	004414	050364				DF11					
1475											
1476			;	*	ERROR ITEM 12						
1477			;	*	IMPROPER HEADER DATA						
1478			;	*	TEST	ERR PC	TST PC	DRIVE	CYLNR	TRACK	SECTOR
1479			;	*	\$TMPD	\$ERRPC	\$REG0	CHKDRV	CYL.DS	TRK.DS	SEC.DS
1480			;	*	GDCYL	GDTRK	GDSCTR	BDCYL	BOTRK	BOSCTR	

Line	Code	Value	Description
1481			:* CYL.DS TRK.DS SEC.DS CYL.RD TRK.RD SEC.RD
1482			:* CYLNDR, TRACK, AND SECTOR ARE DECIMAL
1483			
1484	004416	045217	EM12
1485	004420	046576	DH12
1486	004422	050002	DT12
1487	004424	050370	DF12
1488			
1489			:* ERROR ITEM 13
1490			:* DATA COMPARE FAILRE
1491			:* TEST ERR PC TST PC DRIVE CYLNDR TRACK SECTOR
1492			:* \$TMPO \$ERRPC \$FREGO CHKDRV CYL.DS TRK.DS SEC.DS
1493			:* GDDAT BDDAT WRDCNT GDADR BDADR
1494			:* \$GDDAT \$BDDAT \$REG4 \$GDADR \$BDADR
1495			:* CYLNDR, TRACK, SECTOR, AND WRDCNT ARE DECIMAL
1496			
1497	004426	045244	EM13
1498	004430	046576	DH12
1499	004432	050034	DT13
1500	004434	050400	DF13
1501			
1502			:* ERROR ITEM 14 -- FOLLOWS #13
1503			:* \$GDDAT \$BDDAT \$REG4 \$GDADR \$BDADR
1504			
1505	004436	000000	0
1506	004440	000000	0
1507	004442	050052	DT13A
1508	004444	050410	DF14
1509			
1510			:* ERROR ITEM 15
1511			:* DATA COMPARE FAILURE
1512			:* TEST ERR PC TST PC DRIVE CYLNDR TRACK SECTOR
1513			:* \$TMPO \$ERRPC \$FREGO CHKDRV CYL.DS TRK.DS SEC.DS
1514			:* GDDAT BDDAT WRDCNT GDADR BDADR
1515			:* \$GDDAT \$BDDAT \$REG4 \$GDADR \$BDADR
1516			:* CYLNDR, TRACK, SECTOR, AND WRDCNT ARE DECIMAL
1517			
1518	004446	045244	EM13
1519	004450	046576	DH12
1520	004452	050034	DT13
1521	004454	050400	DF13
1522			
1523			:* ERROR ITEM 16 -- FOLLOWS #15
1524			:* \$GDDAT \$BDDAT \$REG4 \$GDADR \$BDADR
1525			
1526	004456	000000	0
1527	004460	000000	0
1528	004462	050052	DT13A
1529	004464	050410	DF14
1530			
1531			:* ERROR ITEM 17
1532			:* DISK ERROR IN TIMING TEST
1533			:* TEST ERR PC DRIVE RMCS1 RMDS RMER1 RMMR2 RMER2
1534			:* \$TMPO \$ERRPC CHKDRV RM.REG RM.REG+12 RM.REG+14 RM.REG+40 RM.REG+42
1535			
1536	004466	045271	EM17



1537	004470	047012	DH17
1538	004472	050064	DT17
1539	004474	050414	DF17
1540			
1541			;* ERROR ITEM 20
1542			;* CLOCK (KW11-P) OVERFLOW IN TIMING TEST
1543			;* TEST ERR PC DRIVE RMCS1 RMDS RMER1 RMMR2 RMER2
1544			;* \$TMPD \$ERRPC CHKDRV RM.REG RM.REG+12 RM.REG+14 RM.REG+40 RM.REG+42
1545			
1546	004476	045323	EM20
1547	004500	047012	DH17
1548	004502	050064	DT17
1549	004504	050414	DF17
1550			
1551			;* ERROR ITEM 21
1552			;* DATA COMPARE FAILURE
1553			;* TEST ERR PC TST PC DRIVE CYLNDR TRACK
1554			;* \$TMPD \$ERRPC \$REGD CHKDRV CYL.DS TRK.DS
1555			;* GDDAT BDDAT WRDCNT SECTOR
1556			;* \$REG1 \$BDDAT \$REG4 \$REG1
1557			;* CYLINDER, TRACK, WRDCNT, AND SECTOR ARE DECIMAL
1558			
1559	004506	045244	EM13
1560	004510	047107	DH21
1561	004512	050104	DT21
1562	004514	050420	DF21
1563			
1564			;* ERROR ITEM 22--FOLLOWS #21
1565			;* \$REG1 \$BDDAT \$REG4 \$REG1
1566			
1567	004516	000000	0
1568	004520	000000	0
1569	004522	050120	DT21A
1570	004524	050430	DF22
1571			
1572			;* ERROR ITEM 23
1573			;* DISK ERROR DURING SEEK
1574			;* TEST ERR PC DRIVE CYLNDR RMCS1 RMCS2 RMDS
1575			;* \$TMPD \$ERRPC CHKDRV CYL.DS RM.REG RM.REG+10 RM.REG+12
1576			;* RMER1 RMMR2 RMER2 RMDC RMR
1577			;* RM.REG+14 RM.REG+40 RM.REG+42 RM.REG+34 RM.REG+36
1578			
1579	004526	045372	EM23
1580	004530	047224	DH23
1581	004532	050130	DT23
1582	004534	050434	DF23
1583			
1584			;* ERROR ITEM 24
1585			;* SEEK NOT COMPLETE WITHIN 120 MS
1586			;* TEST ERR PC DRIVE CYLNDR RMCS1 RMCS2 RMDS
1587			;* \$TMPD \$ERRPC CHKDRV CYL.DS RM.REG RM.REG+10 RM.REG+12
1588			;* RMER1 RMMR2 RMER2 RMDC RMR
1589			;* RM.REG+14 RM.REG+40 RM.REG+42 RM.REG+34 RM.REG+36
1590			
1591	004536	045421	EM24
1592	004540	047224	DH23

1593 004542 050130  
1594 004544 050434

DT23  
DF23

1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648

004546

004546 045461  
004550 047356  
004552 050160  
004554 050444  
  
004556 045461  
004560 047414  
004562 050170  
004564 050450

```
*****  
*****  
* 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:  
* RH70/RMO3 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  
* 5) FATAL PARITY ERROR  
* 6) SOFTWARE TIMEOUT ON THIS DRIVE  
* 7) SOFTWARE TIMEOUT ON ANOTHER DRIVE  
* 8) ERROR OCCURRED DURING I/O OPERATION  
* 9) ERROR OCCURRED DURING NON-I/O OPERATION  
* 10) UNSAFE OCCURRED  
* 11) AUTOMATIC RECALIBRATE SEQUENCE OCCURRED
```

ITEM41:

```
* ERROR ITEM 41  
* RH70/RMO3 ERROR (MESSAGE)  
* TEST ERR PC TST PC DRIVE  
* $TMPO $ERRPC $REGO CHKDRV
```

EM41  
DM41  
DT41  
DF41

```
* ERROR ITEM 42  
* RH70/RMO3 ERROR (MESSAGE)  
* TEST ERR PC TST PC DRIVE RMCS1 RMCS2 RMD5  
* $TMPO $ERRPC $REGO CHKDRV RM.REG RM.REG+10 RM.REG+12
```

EM41  
DM42  
DT42  
DF42

```
* ERROR ITEM 43  
* RH70/RMO3 ERROR (MESSAGE)  
* TEST ERR PC TST PC DRIVE RMCS1 RMCS2 RMD5  
* $TMPO $ERRPC $REGO CHKDRV RM.REG RM.REG+10 RM.REG+12  
* RMER1 RMR2 RMER2  
* RM.REG+14 RM.REG+40 RM.REG+42
```

EM41  
DM42  
DT43

1649 004574 050454

DF43

1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695

004576 045461  
004600 046576  
004602 050232  
004604 050464  
  
004606 045461  
004610 046576  
004612 050272  
004614 050500  
  
004616 045501  
004620 046576  
004622 050272  
004624 050500

```

:* ERROR ITEM 44
:* RH70/RMO3 ERROR (MESSAGE)
:* TEST ERR PC TST PC DRIVE CYLNDR TRACK SECTOR
:* STMPO $ERRPC $REGO CHKDRV CYL.DS TRK.DS SEC.DS
:* RMCS1 RMCS2 RMD5 RMHR RMDC RMDA
:* RM.REG RM.REG+10 RM.REG+12 RM.REG+36 RM.REG+34 RM.REG+06
:* RMER1 RMMR2 RMER2
:* RM.REG+14 RM.REG+40 RM.REG+42
:* CYLNDR, TRACK, AND SECTOR ARE DECIMAL

```

EM41  
DH12  
DT44  
DF44

```

:* ERROR ITEM 45
:* RH70/RMO3 ERROR (MESSAGE)
:* TEST ERR PC TST PC DRIVE CYLNDR TRACK SECTOR
:* STMPO $ERRPC $REGO CHKDRV CYL.DS TRK.DS SEC.DS
:* RMCS1 RMCS2 RMD5 RMHR RMDC RMDA
:* RM.REG RM.REG+10 RM.REG+12 RM.REG+36 RM.REG+34 RM.REG+06
:* RMER1 RMMR2 RMER2 RMWC RMBA RMDB
:* RM.REG+14 RM.REG+40 RM.REG+42 RM.REG+2 RM.REG+4 RM.REG+22
:* CYLNDR, TRACK, AND SECTOR ARE DECIMAL

```

EM41  
DH12  
DT45  
DF45

```

:* ERROR ITEM 46
:* FATAL WRITE CHECK ERROR (MESSAGE)
:* TEST ERR PC TST PC DRIVE CYLNDR TRACK SECTOR
:* STMPO $ERRPC $REGO CHKDRV CYL.DS TRK.DS SEC.DS
:* RMCS1 RMCS2 RMD5 RMHR RMDC RMDA
:* RM.REG RM.REG+10 RM.REG+12 RM.REG+36 RM.REG+34 RM.REG+06
:* RMER1 RMMR2 RMER2 RMWC RMBA RMDB
:* RM.REG+14 RM.REG+40 RM.REG+42 RM.REG+2 RM.REG+4 RM.REG+22
:* CYLNDR, TRACK, AND SECTOR ARE DECIMAL

```

EM46  
DH12  
DT45  
DF45

```

1696
1697
1698
1699
1700 004626 012737 177777 001226 START3: MOV    #-1,2#BUSADR ;GET BUSADR FLAG
1701 004634 000402          BR      STR1A
1702 004636 75037 001226 START1: CLR    2#BUSADR ;CLR BUSADR FLAG
1703 004642 005037 001224 STRT1A: CLR    2#CNTRLC ;NO CONTROL "C"
1704 004646 000411          BR      START
1705 004650 012737 177777 001226 START4: MOV    #-1,2#BUSADR ;SET BUSADR FLAG
1706 004656 000402          BR      STR2A
1707 004660 005037 001226 START2: CLR    2#BUSADR ;CLR BUSADR FLAG
1708 004664 012737 177777 001224 STRT2A: MOV    #-1,2#CNTRLC ;SET CONTROL "C" FLAG
1709 004672 000005          START: RESET
1710
1711 .SBTTL INITIALIZE THE COMMON TAGS
1712 ;;CLEAR THE COMMON TAGS (%CMTAG) AREA
1713 004674 012706 001100 MOV    #%CMTAG,P6 ;;FIRST LOCATION TO BE CLEARED
1714 004700 005026          CLR    (R6)+ ;;CLEAR MEMORY LOCATION
1715 004702 022706 001140 CMP    #SWR,R6 ;;DONE?
1716 004706 001374          BNE    #-6 ;;LOOP BACK IF NO
1717 004710 012706 001100 MOV    #STACK,SP ;;SETUP THE STACK POINTER
1718 004714 012737 023614 000020 ;;INITIALIZE A FEW VECTORS
1719 004722 012737 000340 000022 MOV    #SCOPE,2#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
1720 004730 012737 020654 000030 MOV    #340,2#IOTVEC+2 ;;LEVEL 7
1721 004736 012737 000340 000032 MOV    #ERROR,2#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
1722 004744 012737 024134 000034 MOV    #340,2#EMTVEC+2 ;;LEVEL 7
1723 004752 012737 000340 000036 MOV    #STRAP,2#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1724 004760 012737 176543 024606 MOV    #340,2#TRAPVEC+2;LEVEL 7
1725 004766 012737 123456 024610 MOV    #176543,$HINUM ;;PRIME THE RANDOM NUMBER GENERATOR
1726 004774 005037 001204          CLR    #123456,$LONUM ;;BOTH HIGH AND LOW WORDS
1727 005000 005037 001206          CLR    $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
1728 005004 112737 000001 001115 CLR    $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1729 005012 012737 005012 001106 MOV    #1,$ERMAX ;;ALLOW ONE ERROR PER TEST
1730 005020 012737 005020 001110 MOV    #,$SLPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1731 MOV    #,$SLPERR ;;SETUP THE ERROR LOOP ADDRESS
1732 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1733 ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
1734 005026 013746 000004          MOV    2#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
1735 005032 012737 005066 000004 MOV    #64,$2#ERRVEC ;;SET UP ERROR VECTOR
1736 005040 012737 177570 001140 MOV    #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
1737 005046 012737 177570 001142 MOV    #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
1738 005054 022777 177777 174056 CMP    #-1,2#SWR ;;TRY TO REFERENCE HARDWARE SWR
1739 BNE    66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1740 BR    65$ ;;AND THE HARDWARE SWR IS NOT = -1
1741 005064 000403          BR    65$ ;;BRANCH IF NO TIMEOUT
1742 005066 012716 005074          64$: MOV    #65$,(SP) ;;SET UP FOR TRAP RETURN
1743 005072 000002          RTI
1744 005074 012737 000176 001140 65$: MOV    #SWREG,SWR ;;POINT TO SOFTWARE SWR
1745 005102 012737 000174 001142 MOV    #DISPREG,DISPLAY
1746 005110 012637 000004          66$: MOV    (SP)+,2#ERRVEC ;;RESTORE ERROR VECTOR
1747 005114 012700 001160          MOV    #SREGAD,RO ;;FIRST ADDRESS
1748 005120 005020          1$: CLR    (RO)+ ;;CLEAR VARIABLE STORAGE
1749 005122 022700 001210          CMP    #SBELL,RO ;;DONE?
1750 005126 001374          BNE    1$ ;;NO--BRANCH
1751 005130 013737 001414 001150 MOV    2#TPS,2#STPS ;;SETUP THE STATUS AND BUFFER REG'S

```

1752	005136	013737	001416	001152	MOV	2#TPB,2#STPB	FOR THE TYPE ROUTINE
1753	005144	005227	177777		INC	#-1	FIRST START ?
1754	005150	001026			BNE	3\$	BR IF NOT
1755	005152	104401	050514		TYPE	TITLE	TYPE THE PROGRAM'S TITLE
1756	005156	005737	000042		TST	42	AUTO ACCEPT OR CHAIN MODE ?
1757	005162	001006			BNE	2\$	BR IF EITHER
1758	005164	122737	000011	000041	CMPB	#11,41	LOADED FROM AN RMO3 ?
1759	005172	001002			BNE	2\$	BR IF NOT
1760	005174	104401	050612		TYPE	,LOADRV	INSTRUCT THE OPERATOR TO REMOVE THE PACK
1761							ON DRIVE 0 IF DRIVE 0 IS TO BE TESTED
1762	005200	105737	000041		2\$: TSTB	2#41	LOADED FROM PAPER TAPE ?
1763	005204	001410			BEQ	3\$	BR IF NOT
1764	005206	004737	051106		JSR	PC,\$SIZE	SIZE THE MEMORY
1765	005212	023727	051202	100000	CMP	\$LSTAD,#100000	16K OR MORE ON THE SYSTEM ?
1766	005220	103002			BHIS	3\$	BR IF YES
1767	005222	104401	051010		TYPE	,NOLOAD	INFORM THE OPERATOR THAT THE 'XXDP' LOADER
1768							WILL BE OVERWRITTEN
1769	005226	004737	022352		3\$: JSR	PC,\$TKINT	TURN ON THE TTY KEYBOARD INTERRUPT
1770					.SBTTL	GET VALUE FOR SOFTWARE SWITCH REGISTER	
1771	005232	005737	000042		TST	2#42	ARE WE RUNNING UNDER XXDP/CT?
1772	005236	001006			BNE	67\$	BRANCH IF YES
1773	005240	023727	001140	000176	CMP	\$WR,\$SWREG	SOFTWARE SWITCH REG SELECTED?
1774	005246	001005			BNE	68\$	BRANCH IF NO
1775	005250	104406			GTSWR		GET SOFT-SWR SETTINGS
1776	005252	000403			BR	68\$	
1777	005254	112737	000001	001134	67\$: MOVB	#1,\$AUTOB	SET AUTO-MODE INDICATOR
1778	005262				68\$:		
1779	005262	005227	177777		INC	#-1	SEE IF FIRST START
1780	005266	001002			BNE	SRTINT	BR IF NOT
1781	005270	004737	051204		JSR	PC,GETADR	GET OR CHECK THE RH70 ADDRESS
1782	005274	104401	001215		SRTINT: TYPE	,\$CRLF	CR-LF
1783	005300	004737	025032		JSR	PC,2#LP.AVL	CHECK FOR A LINE PRINTER
1784	005304	005037	177776		CIR	2#PS	ENSURE THE PRIORITY = 0
1785	005310	012 37	000001	001104	MOV	#1,\$ICNT	SET ITERATION COUNT TO 1
1786	005316	004737	032424		JSR	PC,2#GETSWR	GO CHECK FOR CONTROL SWITCHES
1787	005322	004737	025074		JSR	PC,2#ST.CLK	INITIALIZE THE CLOCK
1788	005326	004737	035532		SETVEC: JSR	PC,RMINIT	CHECK THE DRIVE STATUS
1789	005332	012737	177777	035454	MOV	#-1,\$SAVEFG	SET THE SAVE REGISTERS FLAG
1790	005340	062727	177777	000000	ADD	#-1,#0	FIRST START ?
1791	005346	103003			BCC	11\$	BR IF YES
1792	005350	005737	001224		TST	CNTRLC	CONTROL 'C' SWITCH SET ?
1793	005354	001072			BNE	SRTDRV	CONTINUE IF YES
1794	005356	012737	000340	177776	11\$: MOV	#PR7,\$PS	SET PRIORITY TO 7
1795	005364	005004			CLR	R4	DRIVE TABLE POINTER
1796	005366	104401	001215		TYPE	,\$CRLF	CR-LF
1797	005372	104401	043734		TYPE	,SYSTAT	TYPE STATUS HEADING
1798	005376				1\$:		
1799	005376	010446			MOV	R4,-(SP)	SAVE R4 FOR TYPEOUT
1800							TYPE DRIVE NUMBER
1801	005400	104403			TYPOS		GO TYPE--OCTAL ASCII
1802	005402	002			.BYTE	2	TYPE 2 DIGIT(S)
1803	005403	000			.BYTE	0	SUPPRESS LEADING ZEROS
1804	005404	104401	044657		TYPE	,MSG.SP	SPACES
1805	005410	104401	044657		TYPE	,MSG.SP	SPACES
1806	005414	105764	035366		TSTB	DRVSTA(R4)	CHECK DRIVE'S STATUS
1807	005420	100416			BMI	4\$	BR IF UNSAFE

1808	005422	001020		BNE	5\$		; BR IF ONLINE
1809	005424	105764	035376	TSTB	DRVTP(R4)		; SEE IF OFFLINE OR NONEXISTENT
1810	005430	001404		BEQ	2\$		; BR IF NONEXISTENT
1811	005432	100006		BPL	3\$		; BR IF OFFLINE
1812	005434	104401	044030	TYPE	NOTRM		; DRIVE NOT AN RMO3
1813	005440	000430		BR	9\$		; CHECK NEXT DRIVE
1814	005442	104401	044003	2\$: TYPE	NOTPRS		; DRIVE NOT PRESENT
1815	005446	000425		BR	9\$		; CHECK NEXT DRIVE
1816	005450	104401	043762	3\$: TYPE	UNTOFF		; DRIVE OFFLINE
1817	005454	000405		BR	6\$		; PRINT DRIVE TYPE
1818	005456	104401	044020	4\$: TYPE	NOTSAF		; DRIVE UNSAFE
1819	005462	000402		BR	6\$		; PRINT DRIVE TYPE
1820	005464	104401	043773	5\$: TYPE	UNTON		; DRIVE ONLINE
1821	005470	104401	044657	6\$: TYPE	MSG.SP		; SPACES
1822	005474	012737	044042	005520	MOV	#ISRM03, 8\$	; ADDRESS OF RMO3 MESSAGE
1823	005502	132764	000004	035376	BITB	#BIT02, DRVTP(R4)	; RMO3 ?
1824	005510	001002		BNE	7\$		; TYPE IT IF YES
1825	005512	104401	044030	TYPE	NOTRM		; TYPE DRIVE NOT AN RMO3 IF NOT
1826	005516	104401		7\$: TYPE			; TYPE THE DRIVE TYPE MESSAGE
1827	005520	000000		8\$: WORD	0		; MESSAGE ADDRESS HERE
1828	005522	104401	001215	9\$: TYPE	SCRLF		; CR-LF
1829	005526	005204		INC	R4		; INCREMENT DRIVE NUMBER/TABLE POINTER
1830	005530	020427	000010	CMP	R4, #8.		; FINISHED ?
1831	005534	001320		BNE	1\$		; BR IF NOT
1832	005536	005037	177776	CLR	PS		; SET PRIORITY BACK TO 'D'
1833	005542	005737	001224	SRTDRV: TST	#CNTRLC		; CONTROL "C" START/RESTART?
1834	005546	001417		BEQ	1\$		; NO--BRANCH
1835	005550	013746	001222	MOV	SAVCSW, -(SP)		; GET THE PREVIOUS 'C.SWR' CONTENTS
1836	005554	063716	001220	ADD	C.SWR, (SP)		; SET UP TO SEE IF 'BIT00' IS DIFFERENT
1837	005560	032726	000001	BIT	#BIT00, (SP)+		; IS 'BIT00' DIFFERENT ?
1838	005564	001405		BEQ	9\$		; BR IF NOT
1839	005566	013737	001220	001222	MOV	C.SWR, SAVCSW	; STORE PRESENT 'C.SWR' VALUE
1840	005574	004737	025352	JSR	PC, LODFLT		; RESET PARAMETERS TO THEIR DEFAULT VALUES
1841	005600	004737	032654	9\$: JSR	PC, #GT.PRM		; GET PARAMETERS
1842	005604	000420		BR	4\$		
1843	005606	004737	025352	1\$: JSR	PC, LODFLT		; SETUP DEFAULT PARAMETERS
1844	005612	005037	001232	CLR	DRVSEL		; NO DRIVES SELECTED
1845	005616	005000		CLR	RO		; DETERMINE THE DRIVES THAT
1846	005620	012701	000001	MOV	#1, R1		; ARE AVAILABLE FOR TESTING
1847	005624	105760	035366	2\$: TSTB	DRVSTA(RO)		
1848	005630	003403		BLE	3\$		
1849	005632	156037	035502	001232	BISB	ATABIT(RO), #DRVSEL	
1850	005640	005200		3\$: INC	RO		
1851	005642	106301		ASLB	R1		
1852	005644	001367		BNE	2\$		
1853	005646	005037	035456	4\$: CLR	#SEEKFG		; CLEAR SEEK FLAG
1854	005652	032737	000400	001220	BIT	#SW08, #C.SWR	; DO SEEK BEFORE DATA TRANSFER?
1855	005660	001002		BNE	5\$		; YES--BRANCH
1856	005662	005137	035456	COM	#SEEKFG		; NO
1857	005666	122737	000011	000041	5\$: CMPB	#11, 41 ;LOADED	; FROM AN RMO3 ?
1858	005674	001003		BNE	10\$		; BR IF NOT
1859	005676	042737	000001	001232	BIC	#BIT00, DRVSEL	; CLEAR THE DRIVE 0 SELECTION BIT
1860	005704	104401	044047	10\$: TYPE	DRIVES		; 'DRIVES(S) TO BE TESTED'
1861	005710	005037	020570	CLR	#SENDCT		; DETERMINE PASSES TO MAKE AND
1862	005714	005000		CLR	RO		; THE DRIVES TO BE TESTED
1863	005716	013701	001232	MOV	#DRVSEL, R1		; ANY DRIVES SELECTED?

E06

MD-11-DZRMF-A RM03 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 37  
GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0069

```

1864 005722 001004          BNE      6$          ;YES--BRANCH
1865 005724 104401 044100   TYPE     ,NONE      ;'NONE'
1866
1867 005730 000137 020406   JMP      @#SEOP      ;GO TO END OF PROGRAM
1868 005734 006201          6$: ASR      R1        ;REPORT THE DRIVES TO BE TESTED
1869 005736 103011          BCC      7$
1870 005740 005237 020570   INC      @#SENDCT    ;GIVE THIS DRIVE A PASS
1871 005744 010046          MOV      RO,-(SP)    ;SAVE RO FOR TYPEOUT
1872 005746 104403          TYPOS    ;GO TYPE--OCTAL ASCII
1873 005750          001      .BYTE     1    ;TYPE 1 DIGIT(S)
1874 005751          000      .BYTE     0    ;SUPPRESS LEADING ZEROS
1875 005752 005701          TST      R1        ;MORE DRIVES?
1876 005754 001404          BEQ      8$        ;NO--BRANCH
1877 005756 104401 044105   TYPE     ,COMMA     ;
1878 005762 005200          7$: INC      RO      ;FORM DRIVE NUMBER
1879 005764 000763          BR       6$
1880 005766 013737 020570 020562 8$: MOV      @#SENDCT,@#SEOPCT
1881 005774 005737 001244   TST      @#CLKSTA    ;KWI1-P AVAILABLE
1882 006000 003006          RSTRT1   ;YES--BRANCH
1883 006002 032737 036000 001234 BIT      #36000,@#TSTNMS ;NO--ANY TIMING TESTS TO BE PERFORMED?
1884 006010 001402          BEQ      RSTRT1     ;NO--BRANCH
1885 006012 104401 044107   TYPE     ,NOCLOCK   ;'NO KWI1-P CLOCK, TIMING TESTS WILL NOT BE PERFORMED'
1886 006016 005737 001232   RSTRT1: TST     DRVSEL ;ANY DRIVES SELECTED ?
1887 006022 001002          BNE     1$         ;BR IF YES
1888 006024 000137 004660   JMP      START2     ;GET DRIVE SELECTION ENTRY
1889
1890
1891 006030 005037 001254          1$: CLR      CHKDRV   ;INIT. THE CHECK DRIVE KEY
1892 006034 012737 000001 001256 MOV      #1,DRVMSK   ;START TO CHECK DESIRED DRIVES
1893 006042 033737 001256 001232 RSTRT2: BIT     DRVMSK,DRVSEL ;IS THIS DRIVE SELECTED?
1894 006050 001010          BNE     DRVOK       ;YES--GO CHECK IF DRIVE IS READY FOR TESTING
1895 006052 012706 001100   RESTART: MOV     #STACK,SP ;SETUP THE STACK POINTER
1896 006056 005237 001254          INC     CHKDRV     ;MOVE TO NEXT DRIVE NUMBER
1897 006062 106337 001256   ASLB    DRVMSK     ;POSITION THE MASK
1898 006066 103753          BCS     RSTRT1     ;BRANCH IF THE DRIVE NUMBER NEEDS INITIALIZED
1899 006070 000764          BR      RSTRT2
1900
1901 006072 013702 001254          DRVOK: MOV     CHKDRV,R2 ;PICKUP THE DRIVE NUMBER
1902 006076 105762 035366   TSTB   DRVSTA(R2)  ;IS DESIRED DRIVE ON-LINE?
1903 006102 003005          BGT     1$         ;YES, BRANCH
1904 006104 104011          ERROR   11        ;DRIVE SELECTED IS NOT ONLINE
1905 006106 043737 001256 001232 BIC     DRVMSK,DRVSEL ;CLEAR DRIVE'S SELECTION BIT
1906 006114 000756          BR      RESTART    ;RETURN
1907 006116 010237 004104          1$: MOV     R2,@#DPB.A ;SET THE DRIVE NUMBER INTO THE DPB'S
1908 006122 010237 004124          MOV     R2,@#DPB.B
1909 006126 010237 004144          MOV     R2,@#DPB.C
1910 006132 010237 004164          MOV     R2,@#DTADPB
1911 006136 004737 025766   JSR     PC,@#LDCMD  ;LOAD COMMAND INTO DPB.B AND DPB.C
1912 006142 012737 020406 001252 MOV     @#SEOP,@#BYPASS ;IF ERROR GO TO END OF PROGRAM
1913 006150 112737 000020 004105 MOV     #FMT22/256,DPB.A+1 ;ASSUME 16 BIT FORMAT
1914 006156 032737 000001 001220 BIT     #BIT00,C.SWR ;16 BIT FORMAT REQUESTED ?
1915 006164 001402          BEQ     2$         ;BR IF YES
1916 006166 105037 004105          CLR     DPB.A+1    ;CLEAR THE 'FMT22' BIT
1917 006172 112737 000143 004106 2$: MOV     #SETFORM,DPB.A+2 ;SET THE FORMAT BIT PER DPB.A+1
1918 006200 004037 026032          JSR     RO,@#CALL.A ;GO EXECUTE THE COMMAND
1919 006204 112737 000107 004106   MOV     #RECAL,@#DPB.A+2 ;RECAL=COMMAND

```

# F06

MD-11-DZRMF-A RM03 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 38  
GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0070

1920	006212	004037	026032		JSR	RO, @CALL.A	; GO EXECUTE THE COMMAND
1921	006216	104401	044175		TYPE	TESTING	; 'TESTING DRIVE'
1922	006222	010246			MOV	R2, -(SP)	; SAVE R2 FOR TYPEOUT
1923	006224	104403			TYPOS		; GO TYPE--OCTAL ASCII
1924	006226	001			.BYTE	1	; TYPE 1 DIGIT(S)
1925	006227	000			.BYTE	0	; SUPPRESS LEADING ZEROS
1926	006230	104401	044657		TYPE	MSG.SP	; TYPE SPACES
1927	006234	104401	044217		TYPE	SERIAL	; 'SERIAL NUMBER'
1928	006240	012700	000004		MOV	#4, R0	; FOUR DIGITS TO TYPE
1929	006244	013701	004234		MOV	RM.REG+30, R1	; SERIAL NUMBER
1930	006250	005002		3\$:	CLR	R2	; ZERO
1931	006252	006101			ROL	R1	; PUT THE NEXT DIGIT
1932	006254	006102			ROL	R2	; INTO R2
1933	006256	006101			ROL	R1	
1934	006260	006102			ROL	R2	
1935	006262	006101			ROL	R1	
1936	006264	006102			ROL	R2	
1937	006266	006101			ROL	R1	
1938	006270	006102			ROL	R2	
1939	006272	062702	000060		ADD	#'0, R2	; MAKE IT ASCII
1940	006276	010227			MOV	R2, (PC)+	; SAVE IT
1941	006300	000000		4\$:	.WORD	0	
1942	006302	104401	006300		TYPE	4\$	; TYPE
1943	006306	005300			DEC	R0	; ALL DIGITS TYPED?
1944	006310	003357			BGT	3\$	; NO -- BRANCH
1945	006312	104401	001215		TYPE	\$CRLF	
1946	006316	113737	001364	001115	MOVB	ERR.CT, \$ERMAX	; SETUP MAX ERROR COUNT



1947  
1948  
1949

.SBTTL ##### TESTS #####

;/\:  
;/\:  
;/\:

\*IN THE DESCRIPTIONS OF THE BELOW TESTS THE VARIABLES USED  
\*AND THEIR DEFAULT VALUES (UNLESS SPECIFIED OTHERWISE) ARE:

*MNEMONIC	VALUE	VARIABLE
*R	1	ITERATIONS (REPEATS)
*FC	0	FIRST CYLINDER ADDRESS
*LC	410 OR 814	LAST CYLINDER ADDRESS
*IC	1	INCREMENT VALUE
*NC OF NC1	FC+IC	NEW OR MODIFIED CYLINDER ADDRESS
*NC2	LC-IC	NEW OR MODIFIED CYLINDER ADDRESS
*FT	0	FIRST TRACK ADDRESS
*LT	18	LAST TRACK ADDRESS
*IT	1	INCREMENT VALUE
*NT	FT+IT	NEW OR MODIFIED TRACK ADDRESS
*FS	0	FIRST SECTOR ADDRESS
*LS	21	LAST SECTOR ADDRESS

;/\:  
;/\:  
;/\:

1982  
1983  
1984

.SBTTL \*\*\* SEEK TESTS \*\*\*

;/\:  
;/\:  
;/\:

\*THE SEEK TESTS WILL BE EXECUTED USING IMPLIED SEEKS. THESE  
\*IMPLIED SEEKS WILL BE PERFORMED BY "READ HEADER AND  
\*DATA" COMMANDS TO TRACK "FT" SECTOR "FS" OF THE DESIRED CYLINDER.  
\*THE WORD COUNT WILL BE SET SUCH THAT ONLY THE CYLINDER AND  
\*TRACK/SECTOR WORDS OF THE HEADER ARE READ.

;/\:  
;/\:  
;/\:

2000  
2001  
2002

\*\*\*\*\*  
; \*TEST 0 RECAL/SEEK TEST

H06

MD-11-DZRMF-A RMO3 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 40  
TO RECAL/SEEK TEST

SEQ 0072

2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058

```

;* 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 ENSURE NO ERRORS OCCURRED.
;*****
;TESTO:
NOP
BIT 2#BITS+(0*2),TSTNMS ;DO THIS TEST?
BNE 64$ ;YES--BRANCH
JMP TST1 ;NO--GO TO THE NEXT TEST
MOV #0,2#STSTNM ;SET UP TEST NUMBER AND
;CLEAR THE ERROR FLAG (2#EFLG)
JSR PC,LOOPRM ;LOAD THE PARAMETERS FOR THE TEST
MOV 2#TESTO,2#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
MOV 2#STSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
MOV 2#RPT,2#TIMES ;GET THE ITERATION COUNT
MOVB 2#25.,2#SERMAX ;MAX ERRORS ALLOWED FOR TEST

;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
BIT 2#SW12,2#SWR ;CHECK FOR SWITCH 12 SET
BEQ 20$ ;IF = 0, THEN DON'T TYPE TEST NUMBER
MOV 2#STSTNM,R0 ;ELSE, TYPE THE NUMBER
BIC 2#177000,R0 ;MASK OUT THE HIGH ORDER BYTE
TYPE ,2#R0,2#R0 ;SAVE R0 FOR TYPEOUT
;TEST NUMBER
;GO TYPE--OCTAL ASCII
;TYPE 2 DIGIT(S)
;SUPPRESS LEADING ZEROS
20$:
MOVB 2#RECAL,2#DPB.A+2 ;RECAL=COMMAND
MOVB 2#FS,2#DPB.B+10 ;FS
MOVB 2#FT,2#DPB.B+11 ;FT
MOV 2#LC,2#DPB.B+12 ;LC
MOV 2#EXITO,2#BYPASS ;GO TO EXITO ON ERROR
MOV 2#TESTO,2#LPAOR ;SETUP LOOP ADDRESS
TESTO: MOV 2#STACK,SP ;SET UP STACK POINTER
JSR R0,2#CALL.A ;GO EXECUTE THE COMMAND
JSR R0,2#CALL.B ;GO EXECUTE THE COMMAND
EXITO: 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 ENSURE PROPER OPERATION.
;* "LC" WILL DEFAULT TO 128 AND "FC", "FT", "LT", "FS", AND "LS"
;* WILL DEFAULT TO 0
;*****

```

```

2059 006524
2060 006524 000240
2061 006526 033737 001426 001234
2062 006534 001002
2063 006536 000137 006740
2064 006542 012737 000001 001102
2065
2066 006550 004737 025610
2067 006554 012737 006722 001110
2068 006562 013777 001102 172352
2069 006570 013737 001506 001204
2070 006576 112737 000031 001115
2071
2072
2073 006604 032777 010000 172326
2074 006612 001413
2075 006614 013700 001102
2076 006620 042700 177000
2077 006624 104401 001215
2078 006630 010046
2079
2080 006632 104403
2081 006634 002
2082 006635 000
2083 006636 104401 001215
2084 006642
2085
2086 006642 113737 001524 004134
2087 006650 113737 001526 004154
2088 006656 113737 001516 004135
2089 006664 113737 001520 004155
2090 006672 013737 001510 004136
2091 006700 013737 001512 004156
2092 006706 012737 006736 001252
2093 006714 012737 006722 001106
2094 006722 012706 001100
2095 006726 004037 026334
2096 006732 004037 026144
2097 006736 000004
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112 006740
2113 006740 000240
2114 006742 033737 001430 001234

```

```

TST1:
NOP
BIT @#BITS+<1*2>,TSTNMS ;DO THIS TEST?
BNE 645 ;YES--BRANCH
JMP TST2 ;NO--GO TO THE NEXT TEST
645: MOV #1,@#TSTNM ;SET UP TEST NUMBER AND
;CLEAR THE ERROR FLAG (SERFLG)
JSR PC,LODPRM ;LOAD THE PARAMETERS FOR THE TEST
MOV #TST1,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
MOV $TSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
MOV @#RPT,$TIMES ;GET THE ITERATION COUNT
MOVB #25,$SERMAX ;MAX ERRORS ALLOWED FOR TEST

;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
BIT #SW12,@SWR ;CHECK FOR SWITCH 12 SET
BEQ 205 ;IF = 0, THEN DON'T TYPE TEST NUMBER
MOV $TSTNM,RD ;ELSE, TYPE THE NUMBER
BIC #177000,RD ;MASK OUT THE HIGH ORDER BYTE
TYPE $SCRLF
MOV RD,-(SP) ;SAVE RD FOR TYPEOUT
;TEST NUMBER
;GO TYPE--OCTAL ASCII
;TYPE 2 DIGIT(S)
;SUPPRESS LEADING ZEROS
205:
MOVB @#FS,@#DPB.B+10 ;FS
MOVB @#LS,@#DPB.C+10 ;LS
MOVB @#FT,@#DPB.B+11 ;FT
MOVB @#LT,@#DPB.C+11 ;LT
MOV @#FC,@#DPB.B+12 ;FC
MOV @#LC,@#DPB.C+12 ;LC
MOV #EXIT1,@#BYPASS ;GO TO EXIT1 ON ERROR
MOV #TST1,$LPADR ;SETUP LOOP ADDRESS
TEST1: MOV #STACK,SP ;SET THE STACK POINTER
JSR RD,@#CALL.C ;GO EXECUTE THE COMMAND
JSR RD,@#CALL.B ;GO EXECUTE THE COMMAND
EXIT1: SCOPE ;LOOP

;*****
;*TEST 2 INCREMENT/SEEK TEST
;*
;* THIS TEST WILL COMMAND FORWARD SEEK CYCLES TO ADVANCE THE
;* CYLINDER ADDRESS FROM "FC" TO "LC" BY THE INCREMENT "IC".
;* WHEN THE RESULTANT CYLINDER ADDRESS (NC) EXCEEDS
;* "LC" REVERSE SEEK CYCLES ARE INITIATED; STARTING
;* AT THE LAST LEGAL "NC" AND DECREMENTING BY "IC"
;* UNTIL "NC" IS LESS THAN "FC". AT THE COMPLETION OF EACH
;* SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO
;* ENSURE PROPER OPERATION.
;*****
TST2:
NOP
BIT @#BITS+<2*2>,TSTNMS ;DO THIS TEST?

```



```

2171 007214 000240      NOP
2172 007216 033737 001432 001234  BIT      2#BITS+(3*2),TSTNMS ;DO THIS TEST?
2173 007224 001002      BNE      64$ ;YES--BRANCH
2174 007226 000137 007446      JMP      TST4 ;NO--GO TO THE NEXT TEST
2175 007232 012737 000003 001102 64$: MOV      #3,2#STSTNM ;SET UP TEST NUMBER AND
2176                                ;CLEAR THE ERROR FLAG (SERFLG)
2177 007240 004737 025610      JSR      PC,LOOPRM ;LOAD THE PARAMETERS FOR THE TEST
2178 007244 012737 007362 001110  MOV      #TEST3,2#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
2179 007252 013777 001102 171662  MOV      $TSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2180 007260 013737 001506 001204  MOV      2#RPT,$TIMES ;GET THE ITERATION COUNT
2181 007266 112737 000031 001115  MOVB     #25.,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
2182
2183                                ;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
2184 007274 032777 010000 171636  BIT      #SW12,2#SWR ;CHECK FOR SWITCH 12 SET
2185 007302 001413      BEQ      20$ ;IF = 0, THEN DON'T TYPE TEST NUMBER
2186 007304 013700 001102      MOV      $TSTNM,R0 ;ELSE, TYPE THE NUMBER
2187 007310 042700 177000      BIC      #177000,R0 ;MASK OUT THE HIGH ORDER BYTE
2188 007314 104401 001215      TYPE     $,SCRLF
2189 007320 010046      MOV      R0,-(SP) ;SAVE R0 FOR TYPEOUT
2190                                ;TEST NUMBER
2191 007322 104403      TYPOS   ;GO TYPE--OCTAL ASCII
2192 007324 002 ;TYPE 2 DIGIT(S)
2193 007325 000 ;SUPPRESS LEADING ZEROS
2194 007326 104401 001215      TYPE     $,SCRLF
2195 007332 20$:
2196
2197 007332 012737 007340 001106  MOV      #1$,SLPADR ;SETUP TEST LOOP ADDRESS
2198 007340 113737 001524 004134 1$: MOVB     2#FS,2#DPB.B+10 ;FS
2199 007346 113737 001516 004135  MOVB     2#FT,2#DPB.B+11 ;FT
2200 007354 012737 007444 001252  MOV      #EXIT3,2#BYPASS ;GO TO BYPASS ON ERROR
2201 007362 013737 001510 004136 TEST3: MOV      2#FC,2#DPB.B+12 ;FC
2202 007370 012737 007370 001110  MOV      #.,$LPERR ;SETUP THE ERROR LOOP ADDRESS
2203 007376 012706 001100      MOV      #STACK,SP ;LOAD THE STACK POINTER
2204 007402 004037 026144      JSR      R0,2#CALL.B ;GO EXECUTE THE COMMAND
2205 007406 013701 001514      MOV      IC,R1 ;CYLINDER I
2206 007412 012737 007412 001110  MOV      #.,$LPERR ;SETUP THE ERROR LOOP ADDRESS
2207 007420 012706 001100      MOV      #STACK,SP ;LOAD THE STACK POINTER
2208 007424 010137 004136 1$: MOV      R1,2#DPB.B+12 ;DESIRED CYLINDER
2209 007430 004037 026144      JSR      R0,2#CALL.B ;GO EXECUTE THE COMMAND
2210 007434 006301      ASL      R1 ;MOVE TO NEXT CYLINDER
2211 007436 020137 001512      CMP      R1,2#LC ;DONE?
2212 007442 003770      BLE     1$ ;NO--LOOP
2213 007444 000004      EXIT3: SCOPE
2214
2215 ;*****
2216 ;*TEST 4 OSCILLATING SEEK TEST
2217
2218 ;* THIS TEST WILL COMMAND SEEK CYCLES FROM "FC" TO "NC" AND BACK
2219 ;* TO "FC". "NC" STARTS AT "FC" AND INCREMENTS BY "IC" UP TO CYLINDER
2220 ;* "LC", THEN IS DECREMENTED BY "IC" BACK TO CYLINDER "FC". AT THE
2221 ;* COMPLETION OF EVERY SEEK COMMAND THE PROPER INDICATORS ARE
2222 ;* EXAMINED TO ENSURE PROPER OPERATION.
2223
2224 ;*****
2225 007446 †TST4:
2226 007446 000240      NOP

```

```

2227 007450 033737 001434 001234      BIT      @#BITS+(4*2),TSTNMS ;DO THIS TEST?
2228 007456 001002                    BNE      64$                ;YES--BRANCH
2229 007460 000137 010072                    JMP      TST5                ;NO--GO TO THE NEXT TEST
2230 007464 012737 000004 001102 64$:  MOV      #4,@#TSTNM          ;SET UP TEST NUMBER AND
2231                                ;CLEAR THE ERROR FLAG (SERFLG)
2232 007472 004737 025610                    JSR      PC,LODPRM          ;LOAD THE PARAMETERS FOR THE TEST
2233 007476 012737 007630 001110          MOV      @TEST4,@#SLPERR    ;SETUP THE LOOP ON ERROR ADDRESS
2234 007504 013777 001102 171430          MOV      $TSTNM,@DISPLAY   ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2235 007512 013737 001506 001204          MOV      @#APT,$TIMES      ;GET THE ITERATION COUNT
2236 007520 112737 000031 001115          MOV      @25.,$ERMAX      ;MAX ERRORS ALLOWED FOR TEST
2237
2238                                ;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
2239 007526 032777 010000 171404          BIT      #SW12,@SWR        ;CHECK FOR SWITCH 12 SET
2240 007534 001413                    BEQ      20$                ;IF = 0, THEN DON'T TYPE TEST NUMBER
2241 007536 013700 001102                    MOV      $TSTNM,R0         ;ELSE, TYPE THE NUMBER
2242 007542 042700 177000                    BIC      #177000,R0        ;MASK OUT THE HIGH ORDER BYTE
2243 007546 104401 001215                    TYPE     $CRLF
2244 007552 010046                    MOV      R0,-(SP)         ;;SAVE R0 FOR TYPEOUT
2245                                ;;TEST NUMBER
2246 007554 104403                    TYPOS    ;GO TYPE--OCTAL ASCII
2247 007556 002                        .BYTE   2                ;;TYPE 2 DIGIT(S)
2248 007557 000                        .BYTE   0                ;;SUPPRESS LEADING ZEROS
2249 007560 104401 001215                    TYPE     $CRLF
2250 007564                                20$:
2251
2252 007564 012737 007572 001106          MOV      #1$,SLPADR        ;SETUP LOOP ADDRESS
2253 007572 113737 001524 004134 1$:  MOV      @#FS,@#DPB.B+10   ;FS
2254 007600 113737 001516 004135          MOV      @#FT,@#DPB.B+11   ;FT
2255 007606 012737 010070 001252          MOV      @EXIT4,@#BYPASS   ;GO TO EXIT4 ON ERROR
2256 007614 005002                    CLR      R2                ;CLEAR STALL SWITCH (NO STALL)
2257 007616 032737 010000 001220          BIT      #SW12,@#C.SWR    ;STALL REQUIRED?
2258 007624 001401                    BEQ      TEST4             ;NO--BRANCH
2259 007626 005102                    COM      R2                ;YES--SET SWITCH
2260 007630 013701 001510                    MOV      @#FC,R1          ;SET NC TO FC
2261 007634 005037 001336                    CLR      @#STALLO         ;START AT ZERO IF STALLS REQUIRED
2262 007640 012737 007640 001110          MOV      @#SLPERR          ;SETUP THE ERROR LOOP ADDRESS
2263 007646 012706 001100                    MOV      @#STACK,SP       ;LOAD THE STACK POINTER
2264 007652 010137 004136 1$:  MOV      R1,@#DPB.B+12     ;NC
2265 007656 004037 026144                    JSR      R0,@#CALL.B      ;GO EXECUTE THE COMMAND
2266 007662 005702                    TST     R2                ;STALL?
2267 007664 001403                    BEQ      2$                ;NO--BRANCH
2268 007666 004037 027364                    JSR      R0,@#STALL      ;YES--GO TO STALL ROUTINE
2269 007672 001336                    .WORD   STALLO            ;TIME POINTER
2270 007674 013737 001510 004136 2$:  MOV      FC,@#DPB.B+12     ;FC
2271 007702 004037 026144                    JSR      R0,@#CALL.B      ;GO EXECUTE THE COMMAND
2272 007706 005702                    TST     R2                ;STALL?
2273 007710 001413                    BEQ      3$                ;NO--BRANCH
2274 007712 004037 027364                    JSR      R0,@#STALL      ;YES--GO TO STALL ROUTINE
2275 007716 011336                    .WORD   STALLO            ;TIME POINTER
2276 007720 015237 001336                    INC      @#STALLO         ;UPDATE THE TIME
2277 007724 023737 001362 001336          CMP      @#MXSTAL,@#STALLO ;TIME TO BIG?
2278 007732 003347                    BGT     1$                ;NO--BRANCH
2279 007734 005037 001336                    CLR      @#STALLO         ;YES--START OVER AT ZERO
2280 007740 063701 001514 3$:  ADD      @#IC,R1          ;MOVE TO NEXT CYLINDER
2281 007744 020137 001512                    CMP      R1,@#LC          ;LAST CYLINDER COMPLETED?
2282 007750 003740                    BLE     1$                ;NO--BRANCH

```

```

2283 007752 013701 001512      MOV      @#LC,R1      ;SET NC TO LC
2284 007756 012737 007756 001110  MOV      @#SLPERR    ;SETUP THE ERROR LOOP ADDRESS
2285 007764 012706 001100      MOV      @#STACK,SP  ;LOAD THE STACK POINTER
2286 007770 010137 004136      MOV      R1,@#DPB.B+12 ;NC
45:   JSR      R0,@#CALL.B ;GO EXECUTE THE COMMAND
      TST      R2      ;STALL?
      BEQ      55      ;NO--BRANCH
2287 007774 004037 026144      JSR      R0,@#STALL  ;YES--GO TO STALL ROUTINE
2288 010000 005702      .WORD   STALLO      ;TIME POINTER
2289 010002 001403      JSR      R0,@#STALL  ;YES--GO TO STALL ROUTINE
2290 010004 004037 027364      .WORD   STALLO      ;TIME POINTER
2291 010010 001336      MOV      @#LC,@#DPB.B+12 ;LC
2292 010012 013737 001512 004136 55:   JSR      R0,@#CALL.B ;GO EXECUTE THE COMMAND
2293 010020 004037 026144      TST      R2      ;STALL?
2294 010024 005702      BEQ      65      ;NO--BRANCH
2295 010026 001413      JSR      R0,@#STALL  ;YES--GO TO STALL ROUTINE
2296 010030 004037 027364      .WORD   STALLO      ;TIME POINTER
2297 010034 001336      INC      @#STALLO    ;UPDATE STALL TIME
2298 010036 005237 001336      CMP      @#MXSTAL,@#STALLO ;TIME TOO BIG?
2299 010042 023737 001362 001336      BGT      45      ;NO--BRANCH
2300 010050 003347      CLR      @#STALLO    ;YES--SET STALL TIME BACK TO ZERO
2301 010052 005037 001336      SUB      @#IC,R1     ;NEXT CYLINDER
2302 010056 163701 001514      CMP      R1,@#FC    ;DONE?
2303 010062 020137 001510      BGE      45      ;NO--BRANCH
2304 010066 002340      EXIT4:  SCOPE      ;LOOP
2305 010070 000004
2306
2307 ;*****
2308 ;*TEST 5      CONVERGING/DIVERGING SEEK TEST
2309
2310 ;*      THIS TEST WILL CAUSE THE DRIVE TO EXECUTE FORWARD AND REVERSE
2311 ;*      SEEKS FROM "NC1" AND "NC2" RESPECTIVELY, "NC1" WILL BE INCREMENTED
2312 ;*      BY "IC" AND "NC2" WILL BE DECREMENTED BY "IC" UNTIL "NC1" IS
2313 ;*      GREATER THAN THE INITIAL VALUE OF "NC2" AND "NC2" IS
2314 ;*      LESS THAN THE INITIAL VALUE OF "NC1".  AT THE COMPLETION OF
2315 ;*      EACH SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO
2316 ;*      ENSURE PROPER OPERATION.  "NC1" AND "NC2" DEFAULT TO
2317 ;*      "FC" AND "LC" RESPECTIVELY.
2318
2319 ;*****
2320 †ST5:
2321 010072 000240      NOP
2322 010074 033737 001436 001234      BIT      @#BITS+(5*2),TSTNMS ;DO THIS TEST?
2323 010102 001002      BNE      645      ;YES--BRANCH
2324 010104 000137 010330      JMP      TST6      ;NO--GO TO THE NEXT TEST
2325 010110 012737 000005 001102 645:  MOV      #5,@#STSTNM ;SET UP TEST NUMBER AND
2326      ;CLEAR THE ERROR FLAG (SERFLG)
2327 010116 004737 025610      JSR      PC,LODPRM  ;LOAD THE PARAMETERS FOR THE TEST
2328 010122 012737 010240 001110      MOV      @#TEST5,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
2329 010130 013777 001102 171004      MOV      $STNM,@#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2330 010136 013737 001506 001204      MOV      @#RPT,$TIMES ;GET THE ITERATION COUNT
2331 010144 112737 000031 001115      MOV      @#25,$SERMAX ;MAX ERRORS ALLOWED FOR TEST
2332
2333 ; THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
2334 010152 032777 010000 170760      BIT      @#SW12,@#SWR ;CHECK FOR SWITCH 12 SET
2335 010160 001413      BEQ      205      ;IF = 0, THEN DON'T TYPE TEST NUMBER
2336 010162 013700 001102      MOV      $STNM,R0   ;ELSE, TYPE THE NUMBER
2337 010166 042700 177000      BIC      @#177000,R0 ;MASK OUT THE HIGH ORDER BYTE
2338 010172 104401 001215      TYPE     ,SCLF

```

```

2339 010176 010046          MOV      RO,-(SP)          ;; SAVE RO FOR TYPEOUT
2340                                ;; TEST NUMBER
2341 010200 104403          TYPOS                                ;; GO TYPE--OCTAL ASCII
2342 010202      002        .BYTE 2                                ;; TYPE 2 DIGIT(S)
2343 010203      000        .BYTE 0                                ;; SUPPRESS LEADING ZEROS
2344 010204 104401 001215          TYPE      ,SCLRF
2345 010210
2346
2347 010210 012737 010216 001106          MOV      #15,$LPAOR        ;; SETUP LOOP ADDRESS
2348 010216 113737 001524 004134 15:  MOVB    #FS,#DPB.B+10      ;; FS
2349 010224 113737 001516 004135          MOVB    #FT,#DPB.B+11      ;; FT
2350 010232 012737 010326 001252          MOV      #EXITS,#BYPASS    ;; GO TO EXITS ON ERROR
2351 010240 013701 001510          TEST5:  MOV      #FC,R1     ;; START NC1 AT FC
2352 010244 013702 001512          MOV      #LC,R2           ;; START NC2 AT LC
2353 010250 012737 010250 001110          MOV      #,$SLPERR        ;; SETUP THE ERROR LOOP ADDRESS
2354 010256 012706 001100          MOV      #STACK,SP        ;; LOAD THE STACK POINTER
2355 010262 010137 004136 15:  MOV      R1,#DPB.B+12      ;; NC1
2356 010266 004037 026144          JSR      RO,#CALL.B        ;; GO EXECUTE THE COMMAND
2357 010272 010237 004136          MOV      R2,#DPB.B+12      ;; NC2
2358 010276 004037 026144          JSR      RO,#CALL.B        ;; GO EXECUTE THE COMMAND
2359 010302 063701 001514          ADD      #IC,R1           ;; NEXT NC1
2360 010306 163702 001514          SUB      #IC,R2           ;; NEXT NC2
2361 010312 020137 001512          CMP      R1,#LC           ;; DONE?
2362 010316 003003          BGT      EXITS            ;; YES--BRANCH
2363 010320 020237 001510          CMP      R2,#FC ;?
2364 010324 002356          BGE      15               ;; NO--BRANCH
2365 010326 000004          EXIT5:  SCOPE            ;; LOOP
2366
2367 ;*****
2368 ;*TEST 6          SERVO ADDRESSING LOGIC NOISE GENERATOR
2369
2370 ;*
2371 ;* IN THIS TEST A SEEK IS DONE TO CYL "NC" THEN A SEEK TO
2372 ;* NC+4 THEN NC+1 THEN NC+3 THEN NC+2 THEN NC+5. NOW "NC" IS UPDATED
2373 ;* BY "IC" AND THE ABOVE SEQUENCE IS REPEATED UNTIL "LC" IS
2374 ;* EXCEEDED BY ANY OF THE ABOVE VALUES. THE INITIAL VALUE OF "NC"
2375 ;* IS "FC". AT THE COMPLETION OF EACH SEEK COMMAND THE
2376 ;* PROPER INDICATORS ARE EXAMINED TO ENSURE PROPER OPERATION.
2377 ;*****
2378 010330          †ST6:
2379 010330 000240          NOP
2380 010332 033737 001440 001234          BIT      #BITS+(6*2),TSTNMS ;DO THIS TEST?
2381 010340 001002          BNE      64$             ;; YES--BRANCH
2382 010342 000137 010632          JMP      TST7            ;; NO--GO TO THE NEXT TEST
2383 010346 012737 000006 001102 64$:  MOV      #6,#$TSTNM      ;; SET UP TEST NUMBER AND
2384                                ;; CLEAR THE ERROR FLAG (SERFLG)
2385 010354 004737 025610          JSR      PC,LODPRM        ;; LOAD THE PARAMETERS FOR THE TEST
2386 010360 012737 010476 001110          MOV      #TEST6,#SLPERR   ;; SETUP THE LOOP ON ERROR ADDRESS
2387 010366 013777 001102 170546          MOV      $TSTNM,$DISPLAY  ;; LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2388 010374 013737 001506 001204          MOV      #RPT,$TIMES     ;; GET THE ITERATION COUNT
2389 010402 112737 000031 001115          MOVB    #25,$SERMAX      ;; MAX ERRORS ALLOWED FOR TEST
2390
2391 ; THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
2392 010410 032777 010000 170522          BIT      #SW12,$SWR       ;; CHECK FOR SWITCH 12 SET
2393 010416 001413          BEQ      20$             ;; IF = 0, THEN DON'T TYPE TEST NUMBER
2394 010420 013700 001102          MOV      $TSTNM,RO        ;; ELSE, TYPE THE NUMBER

```



```

2395 010424 042700 177000 BIC #177000,R0 ;MASK OUT THE HIGH ORDER BYTE
2396 010430 104401 001215 TYPE $CRLF
2397 010434 010046 MOV R0,-(SP) ;;SAVE R0 FOR TYPEOUT
2398 ;;TEST NUMBER
2399 010436 104403 TYPOS ;;GO TYPE--OCTAL ASCII
2400 010440 002 .BYTE 2 ;;TYPE 2 DIGIT(S)
2401 010441 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
2402 010442 104401 001215 TYPE , $CRLF

```

20\$:

```

2405 010446 012737 010454 001106 MOV #15,$LPAOR ;SETUP LOOP ADDRESS
2406 010454 113737 001524 004134 1$: MOVB @#FS,@#DPB.B+10 ;FS
2407 010462 113737 001516 004135 MOVB @#FT,@#DPB.B+11 ;FT
2408 010470 012737 010630 001252 MOV @EXIT6,@#BYPASS ;GO TO EXIT6 ON ERROR
2409 010476 013701 001510 TEST6: MOV @#FC,R1 ;PICKUP "FC"
2410 010502 013702 001512 MOV @#LC,R2 ;FORM LAST CYLINDER THAT
2411 010506 162702 000005 SUB #5,R2 ;IS AVAILABLE FOR TESTING
2412 010512 012737 010512 001110 MOV #,$SLPERR ;SETUP THE ERROR LOOP ADDRESS
2413 010520 012706 001100 MOV @STACK,SP ;LOAD THE STACK POINTER
2414 010524 020102 1$: CMP R1,R2 ;LAST CYLINDER
2415 010526 003040 BGT EXIT6 ;YES--BRANCH
2416 010530 010137 004136 MOV R1,@#DPB.B+12 ;NC
2417 010534 004037 026144 JSR R0,@#CALL.B ;GO EXECUTE THE COMMAND
2418 010540 062737 000004 004136 ADD #4,@#DPB.B+12 ;NC+4
2419 010546 004037 026144 JSR R0,@#CALL.B ;GO EXECUTE THE COMMAND
2420 010552 162737 000003 004136 SUB #3,@#DPB.B+12 ;NC+1
2421 010560 004037 026144 JSR R0,@#CALL.B ;GO EXECUTE THE COMMAND
2422 010564 062737 000002 004136 ADD #2,@#DPB.B+12 ;NC+3
2423 010572 004037 026144 JSR R0,@#CALL.B ;GO EXECUTE THE COMMAND
2424 010576 162737 000001 004136 SUB #1,@#DPB.B+12 ;NC+2
2425 010604 004037 026144 JSR R0,@#CALL.B ;GO EXECUTE THE COMMAND
2426 010610 062737 000003 004136 ADD #3,@#DPB.B+12 ;NC+5
2427 010616 004037 026144 JSR R0,@#CALL.B ;GO EXECUTE THE COMMAND
2428 010622 063701 001514 ADD @#IC,R1
2429 010626 000736 BR 15
2430 010630 000004 EXIT6: SCOPE ;LOOP

```

\*\*\*\*\*  
; \*TEST 7 RANDOM SEEK TEST

\* THIS TEST PERFORMS RANDOM SEEK OPERATIONS BETWEEN CYLINDERS 'FC'  
\* 'LC'. AFTER EACH SEEK, THE POSITION OF THE DRIVE IS VERIFIED BY  
\* READING A SECTOR FROM THE CURRENTLY ADDRESSED CYLINDER AND TRACK.  
\* THE TRACK ADDRESS IS INCREMENTED FOR EACH SEEK SO THAT VERIFICATION  
\* OF POSITIONING OCCURS USING EACH HEAD. TRACK ADDRESSES ARE INCREMENTED  
\* BETWEEN PARAMTERS 'FT' AND 'LT'.

\*\*\*\*\*  
†ST7:

```

2443 010632 000240 NOP
2444 010632 033737 001442 001234 BIT @#BITS+(7*2),TSTNMS ;DO THIS TEST?
2445 010634 001002 BNE 64$ ;YES--BRANCH
2446 010642 000137 011250 JMP TST10 ;NO--GO TO THE NEXT TEST
2447 010644 012737 000007 001102 64$: MOV #7,@#STSTNM ;SET UP TEST NUMBER AND
2448 010650 004737 025610 JSR PC,LODPRM ;CLEAR THE ERROR FLAG ($ERFLG)
2449 ;LOAD THE PARMETERS FOR THE TEST
2450

```

```

2451 010662 012737 011012 001110      MOV      #TEST7,2#SLPERR      ;SETUP THE LOOP ON ERROR ADDRESS
2452 010670 013777 001102 170244      MOV      $TSTNM,2DISPLAY     ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2453 010676 013737 001506 001204      MOV      2#RPT,$TIMES        ;GET THE ITERATION COUNT
2454 010704 112737 000031 001115      MOVB     #25.,$ERMAX         ;MAX ERRORS ALLOWED FOR TEST
2455
2456                                     ;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
2457 010712 032777 010000 170220      BIT      #SW12,2SWR          ;CHECK FOR SWITCH 12 SET
2458 010720 001413 000000 000000      BEQ      20$                 ;IF = 0, THEN DON'T TYPE TEST NUMBER
2459 010722 013700 001102 000000      MOV      $TSTNM,R0           ;ELSE, TYPE THE NUMBER
2460 010726 042700 177000 000000      BIC      #177000,R0          ;MASK OUT THE HIGH ORDER BYTE
2461 010732 104401 001215 000000      TYPE     $CRLF
2462 010736 010046 000000 000000      MOV      R0,-(SP)           ;SAVE R0 FOR TYPEOUT
2463
2464                                     ;TEST NUMBER
2465 010740 104403 000000 000000      TYPOS
2466 010742 002 000000 000000      .BYTE   2                   ;GO TYPE--OCTAL ASCII
2467 010743 000 000000 000000      .BYTE   0                   ;TYPE 2 DIGIT(S)
2468 010744 104401 001215 000000      TYPE     , $CRLF           ;SUPPRESS LEADING ZEROS
2469
2470                                     20$:
2470 010750 113737 001516 004135      MOVB     FT,DPB.B+11         ;LOAD STARTING TRACK ADDRESS
2471 010756 112737 000105 004106      MOVB     #SEEK,2#DPB.A+2     ;SEEK=COMMAND
2472 010764 112737 000173 004126      MOVB     #READHD,DPB.B+2     ;READ HEADER & DATA COMMAND
2473 010772 013704 035514 000000      MOV      RMAOR,R4           ;UNIBUS ADDRESS OF THE RH70
2474 010776 012737 011246 001252      MOV      #EXIT7,BYPASS      ;ERROR TERMINATION ADDRESS
2475 011004 012737 011012 001106      MOV      #TEST7,$LPADR      ;SETUP THE LOOP ON TEST ADDRESS
2476 011012 012706 001100 000000      MOV      #STACK,SP          ;SETUP THE STACK POINTER
2477 011016 013737 001510 004136      MOV      FC,DPB.B+12         ;INITIAL CYLINDER ADDRESS
2478 011024 023737 001510 001512      CMP      FC,LC               ;CYLINDER LIMITS THE SAME ?
2479 011032 001422 000000 000000      BEQ      1$                  ;BR IF THEY ARE
2480 011034 004737 024510 000000      JSR      PC,$RAND            ;CYCLE THE RANDOM NUMBER GENERATOR
2481 011040 013746 024606 000000      MOV      $HNUM,-(SP)         ;USE THE HIGH RANDOM NUMBER
2482 011044 005046 000000 000000      CLR      -(SP)              ;UPPER DIVIDEND
2483 011046 013746 001512 000000      MOV      LC,-(SP)           ;FORM THE DIVISOR
2484 011052 005216 000000 000000      INC      (SP)               ;INCREMENT
2485 011054 163716 001510 000000      SUB      FC,(SP)            ;SUBTRACT THE LOWER LIMIT
2486 011060 004737 024612 000000      JSR      PC,$DIV             ;DIVIDE
2487 011064 062637 004136 000000      ADD      (SP)+,DPB.B+12     ;ADD THE REMAINDER TO THE INITIAL CYLINDER
2488 011070 005726 000000 000000      TST      (SP)+              ;DISCARD THE QUOTIENT
2489 011072 013737 004136 004116      MOV      DPB.B+12,DPB.A+12  ;COPY NEW CYLINDER ADDRESS
2490
2491                                     1$:
2491 011100 012737 011100 001110      MOV      #,$SLPERR          ;SETUP THE ERROR LOOP ADDRESS
2492 011106 012706 001100 000000      MOV      #STACK,SP          ;LOAD THE STACK POINTER
2493 011112 004037 026032 000000      JSR      R0,2#CALL.A         ;GO EXECUTE THE COMMAND
2494 011116 012737 011116 001110      MOV      #,$SLPERR          ;SETUP THE ERROR LOOP ADDRESS
2495 011124 012706 001100 000000      MOV      #STACK,SP          ;LOAD THE STACK POINTER
2496 011130 113764 004104 000010      MOVB     DPB.A,RMCS2(R4)     ;SELECT THE DRIVE
2497 011136 016446 000020 000000      MOV      RMLA(R4),-(SP)     ;GET THE LOOK AHEAD REGISTER
2498 011142 006316 000000 000000      ASL      (SP)               ;ALIGN THE SECTOR ADDRESS
2499 011144 006316 000000 000000      ASL      (SP)               ;ALIGN THE SECTOR ADDRESS
2500 011146 000316 000000 000000      SWAB     (SP)               ;PUT ADDRESS IN LOWER BYTE
2501 011150 105766 000001 000000      TSTB     1(SP)              ;IN THE 1ST 20% OF SECTOR ?
2502 011154 001401 000000 000000      BEQ      2$                  ;BR IF YES
2503 011156 105216 000000 000000      INCB     (SP)               ;INCREMENT THE SECTOR ADDRESS
2504 011160 105216 000000 000000      INCB     (SP)               ;INCREMENT THE SECTOR ADDRESS
2505 011162 112637 004174 000000      MOVB     (SP)+,DTADPB+10     ;LOAD THE DPB
2506 011166 013746 001630 000000      MOV      PRMLM+22,-(SP)     ;PUT LAST SECTOR ADDRESS ON THE STACK

```

```

2507 011172 005216 INC (SP) ; INCREMENT IT
2508 011174 122637 004174 CMPB (SP)+,DTADPB+10 ; NEW SECTOR ADDRESS TOO LARGE ?
2509 011200 103007 BHIS 4$ ; BR IF NOT
2510 011202 103403 BLO 3$ ; BR IF ADDRESS IS 2 GREATER
2511 011204 105037 004174 CLRB DTADPB+10 ; RESET TO SECTOR ADDRESS 0
2512 011210 000403 BR 4$ ; CONTINUE
2513 011212 112737 000001 004174 3$: MOVB #1,DTADPB+10 ; RESET ADDRESS TO SECTOR 1
2514 011220 4$:
2515 011220 004037 026144 JSR RD,@CALL.B ; GO EXECUTE THE COMMAND
2516 011224 105237 004135 INCB DPB.B+11 ; INCREMENT THE TRACK ADDRESS
2517 011230 123737 004135 001520 CMPB DPB.B+11,LT ; MAXIMUM ?
2518 011236 101403 BLOS EXIT7 ; BR IF NOT
2519 011240 113737 001516 004135 MOVB FT,DPB.B+11 ; RELOAD STARTING TRACK ADDRESS
2520 011246 000004 EXIT7: SCOPE ; LOOP ?

```

```

*****
*TEST 10 SERVO SETTLE DOWN TEST

```

```

;*
;* THIS TEST VERIFIES THAT THE SERVO HAS SETTLED DOWN AND THAT
;* THE DRIVE IS ON CYLINDER WHEN THE DRIVE INDICATES SEEK COMPLETE.
;* RANDOM SEEKS ARE ISSUED BETWEEN CYLINDERS 'NC1' AND 'NC1+IC'
;* ('NC1' STARTS AT VALUE 'FC'). AT THE COMPLETION OF 1000 (10) SEEKS,
;* 'NC1' IS INCREMENTED BY VALUE 'IC' AND THE SEQUENCE IS REPEATED.
;* THE TEST IS COMPLETED WHEN 'NC1' HAS BEEN INCREMENTED BEYOND 'LC'.
;*
;* WHEN THE SEEK COMPLETES, THE PROGRAM READS THE DRIVE'S LOOK-AHEAD
;* REGISTER (RMLA) TO DETERMINE THE ADDRESS OF THE SECTOR ROTATING INTO
;* POSITION. THE PROGRAM THEN ISSUES A WRITE HEADER AND DATA COMMAND
;* FOR THAT SECTOR.
;* ERRORS IN THIS TEST INDICATE THAT THE SERVO SERVO S - 1 MAY NOT BE ADJUSTED
;* CORRECTLY, THAT THE DRIVE IS MALFUNCTIONING, ( - AT A PACK WITH
;* MARGINAL SERVO TRACKS IS MOUNTED ON THE DRIVE.
;*
;* THIS TEST IS VALID ONLY IF THE OPERATION IS STARTED WITHIN A FEW
;* HUNDRED MICRO-SECONDS AFTER SEEK DONE OCCURS. THE NECESSARY
;* TIME DEPENDENT PARAMETERS OCCUR FREQUENTLY ENOUGH WITHIN THE REQUIRED
;* RANGE TO PERMIT THIS TEST TO BE EFFECTIVE.

```

```

*****
TST10:

```

```

2547 011250 NOP
2548 011250 000240 BIT #BITS+(10*2),TSTNMS ; DO THIS TEST?
2549 011252 033737 001444 001234 BNE 64$ ; YES--BRANCH
2550 011260 001002 JMP TST11 ; NO--GO TO THE NEXT TEST
2551 011262 000137 012412 001102 64$: MOV #10,@$TSTNM ; SET UP TEST NUMBER AND
2552 011266 012737 000010 001102 ; CLEAR THE ERROR FLAG (SERFLG)
2553 ;
2554 011274 004737 025610 JSR PC_LOADPRM ; LOAD THE PARAMETERS FOR THE TEST
2555 011300 012737 011514 001110 MOV #TST10,@$SLPERR ; SETUP THE LOOP ON ERROR ADDRESS
2556 011306 013777 001102 167626 MOV $TSTNM,@DISPLAY ; LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2557 011314 013737 001506 001204 MOV @RPT,$TIMES ; GET THE ITERATION COUNT
2558 011322 112737 000031 001115 MOVB #25.,$ERMAX ; MAX ERRORS ALLOWED FOR TEST
2559 ;
2560 ; THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
2561 011330 032777 010000 167602 BIT #SW12,@SWR ; CHECK FOR SWITCH 12 SET
2562 011336 001413 BEQ 20$ ; IF = 0, THEN DON'T TYPE TEST NUMBER

```

2563	011340	013700	001102		MOV	\$STSTM,RO	;ELSE, TYPE THE NUMBER
2564	011344	042700	177000		BIC	#177000,RO	;MASK OUT THE HIGH ORDER BYTE
2565	011350	104401	001215		TYPE	\$CRLF	
2566	011354	010046			MOV	RO,-(SP)	::SAVE RO FOR TYPEOUT
2567							::TEST NUMBER
2568	011356	104403			TYPOS		::GO TYPE--OCTAL ASCII
2569	011360	002			.BYTE	2	::TYPE 2 DIGIT(S)
2570	011361	000			.BYTE	0	::SUPPRESS LEADING ZEROS
2571	011362	104401	001215		TYPE	, \$CRLF	
2572	011366			20\$:			
2573							
2574	011366	012737	011374	001106	MOV	#1\$,\$LPADR	;SETUP THE LOOP ADDRESS
2575	011374						
2576	011374	112737	000105	004106	MOV	#SEEK,\$DPB.A+2	;SEEK=COMMAND
2577	011402	112737	000161	004166	MOV	#WRITE,\$DTADPB+2	;COMMAND
2578	011410	113737	001516	004175	MOV	FT,\$DTADPB+11	;TRACK ADDRESS FOR THE WRITE
2579	011416	013737	001510	004116	MOV	FC,\$DPB.A+12	;CYLINDER ADDRESS FOR THE SEEK
2580	011424	013737	001510	004176	MOV	FC,\$DTADPB+12	;CYLINDER ADDRESS FOR THE WRITE
2581	011432	013737	001510	001532	MOV	FC,NC1	;STARTING CYLINDER
2582	011440	013737	001514	001350	MOV	IC,DELTA	;CYLINDER INCREMENT VALUE
2583	011446	012737	176000	004170	MOV	#-<256.*4.>,\$DTADPB+4	;WORD COUNT
2584	011454	012737	050514	004172	MOV	#BUFFER,\$DTADPB+6	;BUFFER ADDRESS
2585	011462	005000			CLR	RO	;PATTERN POINTER (WC PATTERN)
2586	011464	004737	031360		JSR	PC,SETBUF	;LOAD THE WRITE BUFFER
2587	011470	005001			CLR	R1	;CLEAR REGISTER
2588	011472	113701	004104		MOV	DPB.A,R1	;LOAD DRIVE ADDRESS
2589	011476	013704	035514		MOV	RMADR,R4	;UNIBUS ADDRESS OF THE RH70
2590	011502	004737	043346		JSR	PC,CLIQUE	;CLEAR THE OPERATION QUEUES
2591	011506	012737	012410	001252	MOV	#EXIT10,BYPASS	;ERROR EXIT FROM TEST
2592	011514						
2593	011514	012737	011514	001110	MOV	#,\$LPERR	;SETUP THE ERROR LOOP ADDRESS
2594	011522	012706	001100		MOV	#STACK,SP	;LOAD THE STACK POINTER
2595	011526	012737	000340	177776	MOV	#PR7,\$PS	;SET PRIORITY TO 7
2596	011534	005737	001244		TST	CLKSTA	;SEE WHICH CLOCK ON SYSTEM
2597	011540	001415			BEQ	3\$	;BR IF NO CLOCK
2598	011542	100405			BMI	1\$	;BR IF KW11-L CLOCK
2599	011544	017746	167624		MOV	\$PKV,-(SP)	;SAVE THE VECTOR
2600	011550	013746	001374		MOV	PKV,-(SP)	;SAVE THE VECTOR ADDRESS
2601	011554	000404			BR	2\$	;CONTINUE
2602	011556	017746	167624		MOV	\$LKV,-(SP)	;SAVE THE 'L' CLOCK VECTOR
2603	011562	013746	001406		MOV	LKV,-(SP)	;SAVE THE VECTOR ADDRESS
2604	011566	012776	012344	000000	MOV	#T\$10B,\$(SP)	;CHANGE THE VECTOR
2605	011574	012777	030052	023714	MOV	#DORTI,\$RMVEC	;CHANGE THE RMO3 VECTOR
2606	011602	012737	000010	001344	MOV	#8,\$SEKTR	;LOAD THE SEEK TIMER
2607	011610	012764	000040	000010	MOV	#BIT05,RMCS2(R4)	;INIT THE MASSBUS
2608	011616	110164	000010		MOV	R1,RMCS2(R4)	;RESELECT THE DRIVE
2609	011622	013764	004116	000034	MOV	DPB.A+12,RMDC(R4)	;LOAD THE CYLINDER ADDRESS
2610	011630	013737	004116	001270	MOV	DPB.A+12,CYL.DS	;CYLINDER ADDRESS FOR ERROR MESSAGE
2611	011636	112764	000105	000000	MOV	#SEEK,RMCS1(R4)	;START THE SEEK
2612	011644	005037	177776		CLR	\$PS	;CLEAR THE PRIORITY
2613	011650	105764	000012		TSTB	RMDS(R4)	;HAS THE DRIVE FINISHED ?
2614	011654	100402			BMI	5\$	;BR IF IT HAS
2615	011656	000001			WAIT		;WAIT FOR THE OPERATION TO COMPLETE
2616	011660	000773			BR	4\$	;CONTINUE
2617	011662	012737	000340	177776	MOV	#PR7,\$PS	;CHANGE PRIORITY TO MAX
2618	011670	032764	040000	000012	BIT	#BIT14,RMDS(R4)	;ERROR ?

2619	011676	001412				BEQ	6\$		: BR IF NOT
2620	011700	012702	004104			MOV	#DPB,A,R2		: DPB POINTER
2621	011704	004737	042664			JSR	PC,SVRH70		: SAVE THE REGISTERS
2622	011710	104023				ERROR	23		: ERROR DURING SEEK
2623	011712	012764	000040	000010		MOV	#BIT05,RMCS2(R4)		: INIT THE MASSBUS
2624	011720	110164	000010			MOV	R1,RMCS2(R4)		: RESELECT THE DRIVE
2625	011724	012777	040320	023564	6\$:	MOV	#ISR,DRIVEC		: SETUP THE RMD3 VECTOR
2626	011732	005737	001244			TST	CLKSTA		: WHICH CLOCK
2627	011736	001405				BEQ	TST10A		: BR IF NONE
2628	011740	016676	000002	000000		MOV	2(SP),2(SP)		: RELOAD THE CLOCK VECTOR
2629	011746	062706	000004			ADD	#4,SP		: CORRECT THE STACK POINTER
2630	011752								
2631	011752	012737	011752	001110		MOV	#,SLPERR		: SETUP THE ERROR LOOP ADDRESS
2632	011760	012706	001100			MOV	#STACK,SP		: LOAD THE STACK POINTER
2633	011764	110164	000010			MOV	R1,RMCS2(R4)		: SELECT THE DRIVE
2634	011770	016446	000020			MOV	RMLA(R4),-(SP)		: GET THE LOOK AHEAD REGISTER
2635	011774	006316				ASL	(SP)		: ALIGN THE SECTOR ADDRESS
2636	011776	006316				ASL	(SP)		: ALIGN THE SECTOR ADDRESS
2637	012000	000316				SWAB	(SP)		: PUT ADDRESS IN LOWER BYTE
2638	012002	122766	000300	000001		CMPB	#300,1(SP)		: IN THE LAST 20% OR SECTOR ?
2639	012010	001001				BNE	2\$		: BR IF NOT
2640	012012	105216				INCB	(SP)		: INCREMENT THE SECTOR ADDRESS
2641	012014	105216			2\$:	INCB	(SP)		: INCREMENT THE SECTOR ADDRESS
2642	012016	112637	004174			MOV	(SP)+,DTADPB+10		: LOAD THE DPB
2643	012022	013746	001630			MOV	PRMLM+22,-(SP)		: PUT MAXIMUM SECTOR ADDRESS ON THE STACK
2644	012026	005216				INC	(SP)		: INCREMENT PAST THE MAXIMUM ADDRESS
2645	012030	122637	004174			CMPB	(SP)+,DTADPB+10		: NEW SECTOR ADDRESS TOO LARGE ?
2646	012034	101007				BHI	4\$		: BR IF NOT
2647	012036	103403				BLO	3\$		: BR IF ADDRESS IS 2 GREATER THAN MAXIMUM
2648	012040	105037	004174			CLRB	DTADPB+10		: RESET TO SECTOR ADDRESS 0
2649	012044	000403				BR	4\$		: CONTINUE
2650	012046	112737	000001	004174	3\$:	MOV	#1,DTADPB+10		: RESET ADDRESS TO SECTOR 1
2651	012054	012703	004170		4\$:	MOV	#DTADPB+4,R3		: POINTER
2652	012060	012764	000111	000000		MOV	#DRVCLR,RMCS1(R4)		: CLEAR THE DRIVE
2653	012066	012364	000002			MOV	(R3)+,RMWC(R4)		: LOAD THE WORD COUNT
2654	012072	012364	000004			MOV	(R3)+,RMBR(R4)		: LOAD THE BUFFER ADDRESS
2655	012076	012364	000006			MOV	(R3)+,RMDR(R4)		: LOAD THE TRACK/SECTOR ADDR
2656	012102	005037	004202			CLR	DTADPB+16		: RESET 'DONE' INDICATOR
2657	012106	012737	004164	035426		MOV	#DTADPB,TRNSWT		: LOAD 'TRANSFER' DPB ADDRESS
2658	012114	010137	035500			MOV	R1,DTUM		: ADDRESS OF DRIVE TRANSFERING
2659	012120	112761	000001	035356		MOV	#1,DRVACT(R1)		: SET DRIVE ACTIVE INDICATOR
2660	012126	006301				ASL	R1		: SHIFT DRIVE ADDRESS
2661	012130	012761	001750	035460		MOV	#1000.,TIMER(R1)		: SETUP THE OPERATION TIMER
2662	012136	006201				ASR	R1		: RESTORE R1
2663	012140	013764	004166	000000		MOV	DTADPB+2,RMCS1(R4)		: START THE OPERATION
2664	012146	005037	177776			CLR	2#PS		: CLEAR THE PRIORITY
2665	012152	004037	026544			JSR	RD,DRVCL1		: WAIT FOR OPERATION TO COMPLETE
2666	012156	023727	001346	001750	5\$:	CMP	SEKCNT,#1000.		: FINISHED SEEKS ?
2667	012164	001026				BNE	6\$		: BR IF NOT
2668	012166	005037	001346			CLR	SEKCNT		: CLEAR THE SEEK COUNT
2669	012172	063737	001514	001532		ADD	IC,NC1		: ADD THE INCREMENT
2670	012200	023737	001532	001512		CMP	NC1,LC		: EXCEEDED THE CYLINDER LIMIT ?
2671	012206	103100				BHIS	EXIT10		: BR IF IT HAS
2672	012210	013737	001512	001350		MOV	LC,DELTA		: GET THE NEXT 'ZONE' ADDRESS
2673	012216	163737	001532	001350		SUB	NC1,DELTA		: CHECK THE DIFFERENCE
2674	012224	023737	001514	001350		CMP	IC,DELTA		: DIFFERENCE GREATER THAN THE INCREMENT ?

```

2675 012232 101003          BHI      6$          ;BR IF IT IS
2676 012234 013737 001514 001350  MOV      IC,DELTA  ;USE THE ICREMENT PARAMETER
2677 012242 005237 001346          6$: INC      SEKCN  ;COUNT THE NEXT SEEK
2678 012246 023737 001510 001512  CMP      FC,LC    ;BEGINNING AND ENDING CYLINDERS THE SAME ?
2679 012254 001002          BNE      7$          ;BR IF NOT
2680 012256 000137 011514          JMP      TEST10    ;BR IF THEY ARE
2681 012262 013737 001532 004116  7$: MOV      NCI,DPB.A+12 ;RESET THE CYLINDER ADDRESS
2682 012270 004737 024510          JSR      PC,$RAND  ;CYCLE THE RANDOM NUMBER GENERATOR
2683 012274 013746 024606          MOV      $HNUM,-(SP) ;USE THE HIGH RANDOM NUMBER
2684 012300 005046          CLR      -(SP)     ;CLEAR THE UPPER DIVIDEND
2685 012302 013746 001350          MOV      DELTA,-(SP) ;FORM THE DIVISOR
2686 012306 005216          INC      (SP)     ;INCREMENT
2687 012310 004737 024612          JSR      PC,$DIV   ;DIVIDE
2688 012314 062637 004116          ADD      (SP)+,DPB.A+12 ;ADD THE REMAINDER TO THE INITIAL CYLINDER
2689 012320 005726          TST      (SP)+    ;DISCARD THE QUOTENT
2690 012322 023737 004116 004176  CMP      DPB.A+12,DTADPB+12 ;SAME CYLINDER SELECTED AS LAST TIME ?
2691 012330 001754          BEQ      7$          ;BR IF IT WAS
2692 012332 013737 004116 004176  MOV      DPB.A+12,DTADPB+12 ;COPY NEW CYLINDER ADDRESS
2693 012340 000137 011514          JMP      TEST10    ;CONTINUE
2694 012344 005337 001344          TST10B: DEC     SEKTR  ;DECREMENT THE SEEK TIMER
2695 012350 001016          BNE      1$          ;CONTINUE IF NOT DONE
2696 012352 012702 004104          MOV      #DPB.A,R2 ;DPB ADDRESS
2697 012356 004737 042664          JSR      PC,$VRA70 ;SAVE THE REGISTERS
2698 012362 104024          ERROR   24         ;TIMEOUT DURING SEEK
2699 012364 012764 000040 000010  MOV      #BIT05,RMC$2(R4) ;INIT THE MASSBUS
2700 012372 110164 000010          MOV      R1,RMC$2(R4) ;RESELECT THE DRIVE
2701 012376 016676 000002 000000  MOV      2(SP),2(SP) ;RESTORE THE CLOCK VECTOR ADDRESS
2702 012404 000401          BR       EXIT10    ;ABORT THE TEST
2703 012406 000002          1$: RTI          ;RETURN
2704 012410 00JJO4          EXIT10: SCOPE     ;LOOP ?
2705
2706 ;*****
2707 ;*TEST 11 ALL SEEKS TEST
2708
2709 ;* THIS TEST VERIFIES THAT THE DISK DRIVE CAN SEEK FROM EACH CYLINDER
2710 ;* TO ALL OTHER CYLINDERS.
2711 ;*
2712 ;* BEGINNING WITH CYLINDER 'FC', THE TEST SEEKS TO EACH CYLINDER
2713 ;* BETWEEN 'FC' AND 'LC' FROM CYLINDER 'FC'. THE BEGINNING CYLINDER
2714 ;* ADDRESS IS INCREMENTED AND THE TEST SEEKS BETWEEN THE NEW CYLINDER
2715 ;* ADDRESS AND ALL CYLINDERS BETWEEN 'FC' AND 'LC'. THE SEQUENCE
2716 ;* CONTINUES UNTIL ALL CYLINDERS HAVE BEEN CHECKED.
2717
2718 ;*****
2719 ;*TST11:
2720 012412 000240          NOP
2721 012414 033737 001446 001234  BIT      2#$BITS+<11*2>,$TSTNMS ;DO THIS TEST?
2722 012422 001002          BNE      64$        ;YES--BRANCH
2723 012424 000137 012670          JMP      TST12     ;NO--GO TO THE NEXT TEST
2724 012430 012737 000011 001102  64$: MOV      #11,$TSTNM ;SET UP TEST NUMBER AND
2725 ;* CLEAR THE ERROR FLAG ($ERFLG)
2726 012436 004737 025610          JSR      PC,$LOPRM ;LOAD THE PARAMETERS FOR THE TEST
2727 012442 012737 012610 001110  MOV      #TST11,$SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
2728 012450 013777 001102 166464  MOV      $TSTNM,$DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2729 012456 013737 001506 001204  MOV      2#$RPT,$TIMES ;GET THE ITERATION COUNT
2730 012464 112737 000031 001115  MOV      #25,$ERMAX ;MAX ERRORS ALLOWED FOR TEST

```

```

2731
2732
2733 012472 032777 010000 166440
2734 012500 001413
2735 012502 013700 001102
2736 012506 042700 177000
2737 012512 104401 001215
2738 012516 010046
2739
2740 012520 104403
2741 012522 002
2742 012523 000
2743 012524 104401 001215
2744 012530
2745
2746 012530 012737 012536 001106
2747 012536 113737 001524 004134
2748 012544 113737 001524 004154
2749 012552 113737 001516 004135
2750 012560 113737 001516 004155
2751 012566 013737 001510 004136
2752 012574 013737 001510 004156
2753 012602 012737 012666 001252
2754 012610 012706 001100
2755 012614
2756 012614 004037 026334
2757 012620 004037 026144
2758 012624 063737 001514 004156
2759 012632 023737 001512 004156
2760 012640 002365
2761 012642 013737 001510 004156
2762 012650 063737 001514 004136
2763 012656 023737 001512 004136
2764 012664 002353
2765 012666 000004
2766

;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
BIT #SW12, @SWR ;CHECK FOR SWITCH 12 SET
BEQ 20$ ;IF = 0, THEN DON'T TYPE TEST NUMBER
MOV STSTNM, R0 ;ELSE, TYPE THE NUMBER
BIC #177000, R0 ;MASK OUT THE HIGH ORDER BYTE
TYPE $SCLF
MOV R0, -(SP) ;;SAVE R0 FOR TYPEOUT
;;TEST NUMBER
;;GO TYPE--OCTAL ASCII
;;TYPE 2 DIGIT(S)
;;SUPPRESS LEADING ZEROS
20$:
TYPE , $SCLF

1$: MOV #1$, $LPADR ;SETUP THE LOOP ADDRESS
MOVBS FS, DPB.B+10 ;SECTOR ADDRESS
MOVBS FS, DPB.C+10 ;SECTOR ADDRESS
MOVBS FT, DPB.B+11 ;TRACK ADDRESS
MOVBS FT, DPB.C+11 ;TRACK ADDRESS
MOV FC, DPB.B+12 ;STARTING CYLINDER ADDRESS
MOV FC, DPB.C+12 ;STARTING CYLINDER ADDRESS
MOV #EXIT11, BYPASS ;TEST ABORT EXIT
MOV #STACK, SP ;SETUP THE STACK POINTER

TEST11:
1$: JSR R0, @CALL.C ;GO EXECUTE THE COMMAND
JSR R0, @CALL.B ;GO EXECUTE THE COMMAND
ADD IC, DPB.C+12 ;INCREMENT THE ENDING CYLINDER ADDRESS
CMP LC, DPB.C+12 ;CHECK IF EXCEEDING MAXIMUM
BGE 1$ ;BR IF NOT
MOV FC, DPB.C+12 ;RESET ENDING CYLINDER ADDRESS
ADD IC, DPB.B+12 ;INCREMENT THE STARTING ADDRESS
CMP LC, DPB.B+12 ;EXCEEDING MAXIMUM ?
BGE 1$ ;BR IF NOT
EXIT11: SCOPE ;LOOP ?

```

2767  
2768  
2769

.SBTTL \*\*\* TIMING TESTS \*\*\*

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

;\*THE TIMING TESTS WILL ENSURE THAT THOSE FUNCTIONS BEING  
;\*TIMED ARE WITHIN THE TOLERANCES SPECIFIED IN THE "RMO3  
;\*ENGINEERING SPECIFICATIONS".  
;\*THE SEEK TIMING WILL BE PERFORMED USING EXPLICIT SEEK  
;\*OPERATIONS. AT THE COMPLETION OF EACH OF THE TIMING  
;\*TESTS THE MINIMUM, MAXIMUM AND AVERAGE TIMES WILL BE  
;\*TYPED.

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

2787  
2788  
2789  
2790  
2791  
2792  
2793  
2794  
2795  
2796  
2797  
2798  
2799

::\*\*\*\*\*  
;\*TEST 12 ROTATIONAL SPEED TIMING TEST

;\* THIS TEST WILL START A SEARCH TO CYLINDER 0, TRACK 0, SECTOR  
;\* 0. AS SOON AS THE INTERRUPT OCCURS, THE GO BIT IS SET AGAIN  
;\* AND THE OPERATION IS TIMED. THIS PROCEDURE IS REPEATED 10  
;\* TIMES THEN THE AVERAGE TIME IS CALCULATED AND CHECKED TO  
;\* ENSURE IT IS WITHIN TOLERANCE:  
;\* 16.67 MS/REV + OR - 2% IF 60HZ  
;\* 16.67 MS/REV + OR - 2.5% IF 50HZ.

::\*\*\*\*\*  
\*TST12:

2800  
2801  
2802  
2803  
2804  
2805  
2806  
2807  
2808  
2809  
2810  
2811  
2812  
2813  
2814  
2815  
2816  
2817  
2818  
2819  
2820  
2821  
2822

012670	000240				NOP		
012670	033737	001450	001234		BIT	2#BITS+(12*2),TSTNMS	;DO THIS TEST?
012672	001002				BNE	64\$	;YES--BRANCH
012700	000137	013464			JMP	TST13	;NO--GO TO THE NEXT TEST
012702	012737	000012	001102	64\$:	MOV	#12,2#STSTNM	;SET UP TEST NUMBER AND
012714	004737	025610			JSR	PC,LOOPRM	;CLEAR THE ERROR FLAG (SERFLG)
012720	012737	012670	001110		MOV	#TST12,2#SLPERR	;LOAD THE PARAMETERS FOR THE TEST
012726	013777	001102	166206		MOV	\$TSTNM,2#DISPLAY	;SETUP THE ERROR LOOP ADDRESS
012734	013737	001506	001204		MOV	2#RPT,\$TIMES	;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
012742	112737	000031	001115		MOVB	#25, \$ERMAX	;GET THE ITERATION COUNT
							;MAX ERRORS ALLOWED FOR TEST
012750	032777	010000	166162				;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
012756	001413				BIT	#SW12,2\$WR	;CHECK FOR SWITCH 12 SET
012760	013700	001102			BEQ	20\$	;IF = 0, THEN DON'T TYPE TEST NUMBER
012764	042700	177000			MOV	\$TSTNM,RO	;ELSE, TYPE THE NUMBER
012770	104401	001215			BIC	#177000,RO	;MASK OUT THE HIGH ORDER BYTE
012774	010046				TYPE	\$CRLF	
					MOV	RO,-(SP)	;SAVE RO FOR TYPEOUT
							;TEST NUMBER
012776	104403				TYPOS		;GO TYPE--OCTAL ASCII
013000	002				.BYTE	2	;TYPE 2 DIGIT(S)



```

2823 013001 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
2824 013002 104401 001215 TYPE ,SCLF
2825 013006 20$:
2826
2827 013006 005737 001244 TST @CLKSTA ;KW11-P CLOCK?
2828 013012 003002 BGT 15 ;YES--START TEST
2829 013014 000137 013464 JMP TST13 ;NO--GO TO NEXT TEST
2830 013020 012737 013020 001106 1$: MOV #15,$LPADR ;SETUP LOOP ADDRESS
2831 013026 004037 027670 JSR RO,@SRCH00 ;DO A MASSBUS INIT & RECAL
2832 013032 000402 BR 25 ;RETURN HERE IF NO ERROR
2833 013034 000137 013462 JMP EXIT12 ;RETURN HERE IF ERROR
2834 013040 013764 001510 000034 2$: MOV @#FC,RMDC(R4) ;FC
2835 013046 013746 001524 MOV @#FS,-(SP) ;FS
2836 013052 113766 001516 000001 MOVB @#FT,1(SP) ;FT
2837 013060 012664 000006 MOV (SP)+,RMDA(R4) ;LOAD FT/FS
2838 013064 012737 013462 001206 MOV #EXIT12,$ESCAPE ;ESCAPE TO EXIT12 ON ERROR
2839 013072 005705 CLR R5 ;COUNT UP
2840 013074 012703 002774 MOV #T7A,R3 ;60HZ PARAMETERS
2841 013100 032737 000100 001220 BIT #SW06,@#C.SWR ;60 HZ?
2842 013106 001402 BEQ TEST12 ;YES--BRANCH
2843 013110 012703 003004 MOV #T7B,R3 ;NO--50 HZ PARAMETERS
2844 013114 012706 001100 TEST12: MOV #STACK,SP ;SETUP STACK
2845 013120 012701 000012 MOV #10,R1 ;TIME 10 SEARCHES
2846 013124 004737 030054 JSR PC,@STRTMR ;INITIALIZE THE TIMERS
2847 013130 012777 013336 166236 MOV #7,$@PKV ;SETUP VECTOR IN CASE OF OVERFLOW
2848 013136 012777 030052 022352 MOV #DORTI,@RMVEC ;SETUP RMO3 VECTOR
2849 013144 005077 166232 1$: CLR @PKB ;START COUNTING AT ZERO
2850 013150 012777 000131 166222 MOV #131,@PKCS ;INT.EN., COUNT UP AT 100KHZ
2851 013156 012714 000131 MOV #SEARCH,(R4) ;START A SEARCH
2852 013162 000001 WAIT ;WAIT ON INTERRUPT
2853 013164 042777 000101 166206 BIC #101,@PKCS ;STOP THE CLOCK
2854 013172 032764 040000 000012 BIT #BIT14,RMDS(R4) ;ERROR?
2855 013200 001415 BEQ 25 ;NO--BRANCH
2856 013202 104412 SAVREG ;SAVE R0-R5
2857 013204 012702 004164 MOV #DTADPB,R2 ;DPB POINTER
2858 013210 004737 042664 JSR PC,@SVRH70 ;SAVE ALL THE RH70/RMO3 REGISTERS
2859 013214 012764 000040 000010 MOV #BIT05,RMCS2(R4) ;MASSBUS CLEAR
2860 013222 013764 004164 000010 MOV @#DTADPB,RMCS2(R4) ;SELECT DRIVE
2861 013230 104413 RESREG ;RESTORE R0-R5
2862 013232 104017 ERROR 17
2863 013234 005077 166142 2$: CLR @PKB ;START THE COUNT AT ZERO
2864 013240 012714 000131 MOV #SEARCH,(R4) ;START A SEARCH
2865 013244 012777 000131 166126 MOV #131,@PKCS ;START THE CLOCK
2866 013252 000001 WAIT ;WAIT ON INTERRUPT
2867 013254 042777 000101 166116 BIC #101,@PKCS ;STOP THE CLOCK
2868 013262 032764 040000 000012 BIT #BIT14,RMDS(R4) ;IS "ERR=1"?
2869 013270 001415 BEQ 35 ;NO--BRANCH
2870 013272 104412 SAVREG ;SAVE R0-R5
2871 013274 012702 004164 MOV #DTADPB,R2 ;DPB POINTER
2872 013300 004737 042664 JSR PC,@SVRH70 ;SAVE ALL THE RH70/RMO3 REGISTERS
2873 013304 012764 000040 000010 MOV #BIT05,RMCS2(R4) ;MASSBUS CLEAR
2874 013312 013764 004164 000010 MOV @#DTADPB,RMCS2(R4) ;SELECT DRIVE
2875 013320 104413 RESREG ;RESTORE R0-R5
2876 013322 104017 ERROR 17 ;DISK ERROR OCCURRED
2877 013324 004737 030120 3$: JSR PC,@COUNT ;UPDATE THE COUNT
2878 013330 005301 DEC R1 ;DONE?

```

```

2879 013332 003304          BGT      1$          ;NO--BRANCH
2880 013334 000424          BR       8$          ;YES--GO TO THE EXIT
2891 013336 042777 000101 166034 7$: BIC     #101,@PKCS  ;STOP THE CLOCK
2932 013344 005037 177776          CLR     @#PS        ;DROP THE PRIORITY
2883 013350 012600          MOV     (SP)+,R0    ;PC OF WAIT+2
2934 013352 005726          TST     (SP)+      ;POP THE PS FROM THE STACK
2935 013354 104412          SAVREG          ;SAVE R0-R5
2886 013356 012702 004164          MOV     @DTADPB,R2  ;DPB POINTER
2997 013362 004737 042664          JSR     PC,@SVRH70 ;SAVE ALL THE RH70/RM03 REGISTERS
2938 013366 012764 000040 000010  MOV     #BIT05,RMC52(R4) ;MASSBUS CLEAR
2889 013374 013764 004164 000010  MOV     @DTADPB,RMC52(R4) ;SELECT DRIVE
2890 013402 104413          RESREG          ;RESTORE R0-R5
2891 013404 104020          ERROR    20        ;CLOCK OVERFLOWED
2892 013406
2893 013406 012764 000040 000010 8$: MOV     #BIT05,RMC52(R4) ;MASSBUS INIT.
2894 013414 013764 004164 000010  MOV     @DTADPB,RMC52(R4) ;SELECT DRIVE
2895 013422 004737 025074          JSR     PC,@ST.CLK ;INITIALIZE THE CLOCK
2896 013426 012777 040320 022062  MOV     #ISR,@RMVEC ;RESTORE RH70/RM03 INT. VECTOR
2897 013434 032737 000100 001220  BIT     #SW06,@#C.SWR ;60 HZ?
2898 013442 001004          BNE     9$          ;NO -- BRANCH
2899 013444 004037 030252          JSR     R0,@TYPTIM ;GO TYPE THE TIMES
2900 013450 002774          T7A
2901 013452 000403          BR      EXIT12     ;GO TO EXIT
2902 013454
2903 013454 004037 030252 9$: JSR     R0,@TYPTIM ;GO TYPE THE TIMES
2904 013460 003004          T7B
2905 013462 000004          EXIT12: SCOPE    ;LOOP ?
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934

```

\*\*\*\*\*  
; \*TEST 13 ONE CYLINDER SEEK TIMING TEST

; \* THIS TEST WILL COMMAND FORWARD SEEK CYCLES TO ADVANCE THE  
; \* CYLINDER BY ONE UNTIL THE INCREMENT IS GREATER THAN THE  
; \* CYLINDER 'LC', THEN REVERSE SEEK TO CYLINDER 'FC'. THE  
; \* TIME TO PERFORM EACH SEEK IS CHECKED TO ENSURE IT DOES NOT  
; \* EXCEED THE MAXIMUM TIME PERMITTED FOR A ONE CYLINDER SEEK.  
; \* THE TIME MUST BE LESS THAN 10MS.

\*\*\*\*\*  
; \*ST13:

```

2920 013464 000240          NOP
2921 013466 033737 001452 001234  BIT     @#BITS+(13*2),TSTNMS ;DO THIS TEST?
2922 013474 001002          BNE     64$        ;YES--BRANCH
2923 013476 000137 014166          JMP     TST14     ;NO--GO TO THE NEXT TEST
2924 013502 012737 000013 001102 64$: MOV     #13,@#STSTNM ;SET UP TEST NUMBER AND
2925
2926 013510 004737 025610          JSR     PC,LODPRM ;CLEAR THE ERROR FLAG (SERFLG)
2927 013514 012737 013464 001110  MOV     #TST13,@#SLPERR ;LOAD THE PARAMETERS FOR THE TEST
2928 013522 013777 001102 165412  MOV     $TSTNM,@DISPLAY ;SET UP THE ERROR LOOP ADDRESS
2929 013530 013737 001506 001204  MOV     @#RPT,$TIMES ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2930 013536 112737 000031 001115  MOV     @#25,$ERMAX ;GET THE ITERATION COUNT
2931
2932
2933 013544 032777 010000 165366  MOV     @#SW12,@SWR ;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
2934 013552 001413          BEQ     20$        ;CHECK FOR SWITCH 12 SET
; IF = 0, THEN DON'T TYPE TEST NUMBER

```

L07

MD-11-DZRMF-A RMO3 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 57  
T13 ONE CYLINDER SEEK TIMING TEST

SEQ 0089

```

2935 013554 013700 001102      MOV    $STSTNM,R0      ;ELSE, TYPE THE NUMBER
2936 013560 042700 177000      BIC    #177000,R0     ;MASK OUT THE HIGH ORDER BYTE
2937 013564 104401 001215      TYPE   $SCLF
2938 013570 010046                MOV    R0,-(SP)       ;;SAVE R0 FOR TYPEOUT
2939                                ;;TEST NUMBER
2940 013572 104403                TYPOS  ;GO TYPE--OCTAL ASCII
2941 013574      002                .BYTE 2              ;TYPE 2 DIGIT(S)
2942 013575      000                .BYTE 0              ;SUPPRESS LEADING ZEROS
2943 013576 104401 001215      TYPE   ,SCLF
2944 013602
2945
2946 013602 005737 001244      TST    @#CLKSTA       ;KW11-P CLOCK?
2947 013606 003002                BGT    1$             ;YES--START TEST
2948 013610 000137 014166      JMP    TST14          ;NO--GO TO NEXT TEST
2949 013614 012737 013614 001106 1$:  MOV    #1$, $LPADR    ;SETUP THE LOOP ADDRESS
2950 013622 004037 027670      JSR    R0,@#SRCHOO   ;DO A MASSBUS INIT. AND RECAL
2951 013626 000402                BR     2$             ;NO ERROR RETURN
2952 013630 000137 014164      JMP    EXIT13        ;ERROR RETURN--SCOPE LOOP CALL
2953 013634 012703 003014      MOV    #T10,R3       ;PARAMETER POINTER
2954 013640 012737 014164 001206 2$:  MOV    #EXIT13,$ESCAPE ;ESCAPE TO EXIT13 ON ERROR
2955 013646 012706 001100      TEST13: MOV   $STACK SP    ;SETUP STACK
2956 013652 013737 001510 004176  MOV    FC,@#DTADPB+12 ;START WITH BEGINNING CYLINDER
2957 013660 005237 004176      INC    DTADPB+12     ;INCREMENT THE BEGINNING CYLINDER
2958 013664 005005                CLR    R5             ;SET THE UP/DOWN SWITCH TO UP
2959 013666 004737 030054      JSR    PC,@#STRATMR  ;INITIALIZE THE TIMERS
2960 013672 012777 014060 165474  MOV    #7$, $PKV     ;SETUP INCASE OF OVERFLOW
2961 013700 012777 030052 021610  MOV    #DORTI,$RMVEC ;SET RMO3 VECTOR
2962 013706 005077 165470      CLR    $PKB         ;START THE COUNTER AT ZERO
2963 013712 013764 004176 000034 1$:  MOV    @#DTADPB+12,RMDC ;LOAD DESIRED CYLINDER
2964 013720 012714 000105      MOV    #SEEK,(R4)    ;START A SEEK
2965 013724 012777 000131 165446  MOV    #131,$PKCS    ;START THE CLOCK
2966 013732 000001                WAIT   ;WAIT ON INTERRUPT
2967 013734 042777 000101 165436  BIC    #101,$PKCS    ;STOP THE CLOCK
2968 013742 032764 040000 000012  BIT    #BIT14,RMDS(R4) ;ANY DISK ERRORS?
2969 013750 001415                BEQ    2$             ;NO--BRANCH
2970 013752 104412                SAVREG ;SAVE R0-R5
2971 013754 012702 004164      MOV    #DTADPB,R2    ;DPB POINTER
2972 013760 004737 042664      JSR    PC,@#SVRH70   ;SAVE ALL THE RH70/RMO3 REGISTERS
2973 013764 012764 000040 000010  MOV    #BIT05,RMCS2(R4) ;MASSBUS CLEAR
2974 013772 013764 004164 000010  MOV    @#DTADPB,RMCS2(R4) ;SELECT DRIVE
2975 014000 104413                RESREG ;RESTORE R0-R5
2976 014002 104017                ERROR 17             ;REPORT THE ERROR
2977 014004 004737 030120 2$:  JSR    PC,@#COUNT   ;COUNT THIS SEEKS TIME
2978 014010 004737 027446      JSR    PC,@#TWOMS    ;STALL FOR 2 MILLISECONDS
2979 014014 005705                TST    R5             ;UP OR DOWN?
2980 014016 001011                BNE    4$             ;DOWN--BRANCH
2981 014020 005237 004176 3$:  INC    @#DTADPB+12   ;MOVE TO NEXT CYLINDER
2982 014024 023737 004176 001512  CMP    @#DTADPB+12,LC ;OUT OF CYLINDERS?
2983 014032 002725                BLT    1$             ;NO--GO DO THE NEXT SEEK
2984 014034 012705 177777      MOV    #-1,R5        ;SET UP/DOWN SWITCH TO DOWN
2985 014040 000722                BR     1$             ;GO DO THE NEXT SEEK
2986 014042 005337 004176 4$:  DEC    @#DTADPB+12   ;MOVE TO NEXT CYLINDER
2987 014046 023727 004176 000000  CMP    @#DTADPB+12,#0 ;OUT OF CYLINDERS?
2988 014054 003314                BGT    1$             ;NO--GO DO THE NEXT SEEK
2989 014056 000424                BR     8$             ;GO TO THE EXIT
2990 014060 042777 000101 165312 7$:  BIC    #101,$PKCS    ;STOP THE CLOCK

```

```

2991 014066 005037 177776
2992 014072 012600
2993 014074 005726
2994 014076 104412
2995 014100 012702 004164
2996 014104 004737 042664
2997 014110 012764 000040 000010
2998 014116 013764 004164 000010
2999 014124 104413
3000 014126 104020
3001 014130
3002 014130 012764 000040 000010
3003 014136 013764 004164 000010
3004 014144 004737 025074
3005 014150 012777 040320 021340
3006 014156 004037 030252
3007 014162 003014
3008 014164 000004

```

```

CLR 2#PS ;DROP THE PRIORITY
MOV (SP)+,R0 ;PC OF WAIT+2
TST (SP)+ ;POP THE PS FROM THE STACK
SAVREG ;SAVE R0-R5
MOV #DTADPB,R2 ;DPB POINTER
JSR PC,2#SVRH70 ;SAVE ALL THE RH70/RM03 REGISTERS
MOV #BIT05,RMCS2(R4) ;MASSBUS CLEAR
MOV 2#DTADPB,RMCS2(R4) ;SELECT DRIVE
RESREG ;RESTORE R0-R5
ERROR 20 ;REPORT CLOCK OVERFLOW

85: MOV #BIT05,RMCS2(R4) ;MASSBUS INIT.
MOV 2#DTADPB,RMCS2(R4) ;SELECT DRIVE
JSR PC,2#ST.CLK ;INITIALIZE THE CLOCK
MOV #ISR,ARMVEC ;RESTORE RH70/RM03 INT. VECTOR
JSR R0,2#TYPTIM ;GO TYPE THE TIMES
TIO ;POINTER
EXIT13: SCOPE ;LOOP ?

```

```

3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023

```

```

*****
;TEST 14 ACCESS TIME MEASUREMENT TEST
;
; THIS TEST WILL COMMAND A FORWARD SEEK FROM CYLINDER 0 TO
; CYLINDER 'LC', THEN A REVERSEK FROM CYLINDER 'LC' TO
; CYLINDER 0. BOTH SEEKS ARE TIMED AND CHECKED TO ENSURE THEY
; ARE WITHIN THE TOLERANCE ALLOWED FOR THE ACCESS TIME MEASUREMENT.
; THIS SEQUENCE IS REPEATED 128 TIMES (FOR A TOTAL
; OF 256 SEEKS). THE AVERAGE ACCESS TIME MUST BE LESS THAN 30 MS.
; CYLINDER 'LC' DEFAULTS TO 136 (10) FOR AN RMO3/5 OR TO 255 (10)
; FOR AN RMO3.
*****

```

```

3024 014166
3025 014166 000240
3026 014170 033737 001454 001234
3027 014176 001002
3028 014200 000137 014742
3029 014204 012737 000014 001102
3030
3031 014212 004737 025610
3032 014216 012737 014166 001110
3033 014224 013777 001102 164710
3034 014232 013737 001506 001204
3035 014240 112737 000031 001115
3036
3037
3038 014246 032777 010000 164664
3039 014254 001413
3040 014256 013700 001102
3041 014262 042700 177000
3042 014266 104401 001215
3043 014272 010046
3044
3045 014274 104403
3046 014276 002

```

```

*****
†ST14:
NOP
BIT 2#BITS+(14*2),TSTNMS ;DO THIS TEST?
BNE 645 ;YES--BRANCH
JMP TST15 ;NO--GO TO THE NEXT TEST
645: MOV #14,2#STSTNM ;SET UP TEST NUMBER AND
;CLEAR THE ERROR FLAG (SERFLG)
JSR PC,LODPRM ;LOAD THE PARAMETERS FOR THE TEST
MOV #TST14,2#SLPERR ;SETUP THE ERROR LOOP ADDRESS
MOV $TSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
MOV 2#RPT,$TIMES ;GET THE ITERATION COUNT
MOVB #25.,$ERMAX ;MAX ERRORS ALLOWED FOR TEST

;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
BIT #SW12,2#SWR ;CHECK FOR SWITCH 12 SET
BEQ 205 ;IF = 0, THEN DON'T TYPE TEST NUMBER
MOV $TSTNM,R0 ;ELSE, TYPE THE NUMBER
BIC #177000,R0 ;MASK OUT THE HIGH ORDER BYTE
TYPE $R0L,F ;SAVE R0 FOR TYPEOUT
MOV R0,-(SP) ;TEST NUMBER
TYPOS ;GO TYPE--OCTAL ASCII
.BYTE 2 ;TYPE 2 DIGIT(S)

```

N07

MD-11-DZRMF-A RMO3 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 59  
T14 ACCESS TIME MEASUREMENT TEST

SEQ 0091

```

3047 014277 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
3048 014300 104401 001215 TYPE ,SCRLF
3049 014304 20$:
3050
3051 014304 005737 001244 TST @CLKSTA ;KW11-P CLOCK?
3052 014310 003002 BGT 1$ ;YES--START TEST
3053 014312 000137 014742 JMP TST15 ;NO--GO TO NEXT TEST
3054 014316 012737 014316 001106 1$: MOV #1$, $LPADR ;SET THE LOOP ADDRESS
3055 014324 004037 027670 JSR RO, @SRCHOO ;DO A MASSBUS INIT & RECAL
3056 014330 000402 BR 2$ ;RETURN HERE IF NO ERROR
3057 014332 000137 014740 JMP EXIT14 ;RETURN HERE ON ERROR
3058 014336 012703 003024 2$: MOV #T11, R3 ;PARAMETER POINTER
3059 014342 012737 014740 001206 MOV #EXIT14, $ESCAPE ;ESCAPE TO EXIT14 ON ERROR
3060 014350 012706 001100 TEST14: MOV #STACK, $P ;SETUP STACK
3061 014354 012701 000200 MOV #128, R1 ;REPEAT "0-136-0" 128 TIMES
3062 014360 004737 030054 JSR PC, @STRTRM ;INIT. THE COUNTERS
3063 014364 012777 014634 165002 MOV #7$, @PKV ;SET UP VECTOR IN CASE OF OVERFLOW
3064 014372 012777 030052 021116 MOV #DORTI, @RMVEC ;SETUP RMO3 VECTOR
3065 014401 005077 164776 1$: CLR @PKB ;START COUNT AT ZERO
3066 014404 013764 001512 000034 MOV LC, RMDC(R4) ;'MIDDLE' CYLINDER
3067 014412 012764 000105 000000 MOV #SEEK, RMCS1(R4) ;START A SEEK
3068 014420 012777 000131 164752 MOV #131, @PKCS ;START THE CLOCK
3069 014426 000001 WAIT ;WAIT ON INTERRUPT
3070 014430 042777 000101 164742 BIC #101, @PKCS ;STOP CLOCK
3071 014436 032764 040000 000012 BIT #BIT14, RMDS(R4) ;ERR=1?
3072 014444 001415 BEQ 2$ ;NO--BRANCH
3073 014446 104412 SAVREG ;SAVE RO-R5
3074 014450 012702 004164 MOV #DTADPB, R2 ;DPB POINTER
3075 014454 004737 042664 JSR PC, @SVRH70 ;SAVE ALL THE RH70/RMO3 REGISTERS
3076 014460 012764 000040 000010 MOV #BIT05, RMCS2(R4) ;MASSBUS CLEAR
3077 014466 013764 004164 000010 MOV @DTADPB, RMCS2(R4) ;SELECT DRIVE
3078 014474 104413 RESREG ;RESTORE RO-R5
3079 014476 104017 ERROR 17
3080 014500 005005 2$: CLR R5 ;SET UP/DOWN SWITCH TO UP
3081 014502 004737 030120 JSR PC, @COUNT ;UPDATE THE COUNT
3082 014506 004737 027446 JSR PC, @TWOMS ;STALL FOR 2 MILLISECONDS
3083 014512 005077 164664 CLR @PKB ;START THE COUNT AT ZERO
3084 014516 013764 001510 000034 MOV FC, RMDC(R4) ;BEGINNING CYLINDER
3085 014524 012764 000105 000000 MOV #SEEK, RMCS1(R4) ;START A SEEK
3086 014532 012777 000131 164640 MOV #131, @PKCS ;START THE CLOCK
3087 014540 000001 WAIT ;WAIT ON INTERRUPT
3088 014542 042777 000101 164630 BIC #101, @PKCS ;STOP THE CLOCK
3089 014550 032764 040000 000012 BIT #BIT14, RMDS(R4) ;ERR=1?
3090 014556 001415 BEQ 3$ ;NO--BRANCH
3091 014560 104412 SAVREG ;SAVE RO-R5
3092 014562 012702 004164 MOV #DTADPB, R2 ;DPB POINTER
3093 014566 004737 042664 JSR PC, @SVRH70 ;SAVE ALL THE RH70/RMO3 REGISTERS
3094 014572 012764 000040 000010 MOV #BIT05, RMCS2(R4) ;MASSBUS CLEAR
3095 014600 013764 004164 000010 MOV @DTADPB, RMCS2(R4) ;SELECT DRIVE
3096 014606 104413 RESREG ;RESTORE RO-R5
3097 014610 104017 ERROR 17
3098 014612 012705 177777 3$: MOV #-1, R5 ;SET UP/DOWN SWITCH TO DOWN
3099 014616 004737 030120 JSR PC, @COUNT ;UPDATE THE COUNT
3100 014622 004737 027446 JSR PC, @TWOMS ;STALL FOR 2 MILLISECONDS
3101 014626 005301 DEC R1 ;DONE?
3102 014630 003263 BGT 1$ ;NO--BRANCH

```

```

3103 014632 003424          BR      B$          ;YES--EXIT
3104 014634 042777 000101 164536 7$: BIC      #101,#PKCS ;STOP THE CLOCK
3105 014642 005037 177776          CLR      #PS        ;DROP THE PRIORITY
3106 014646 012600          MOV      (SP)+,RO    ;PC OF WAIT+2
3107 014650 005726          TST      (SP)+      ;POP THE PS FROM THE STACK
3108 014652 104412          SAVREG          ;SAVE RO-R5
3109 014654 012702 004164          MOV      #DTADPB,R2 ;DPB POINTER
3110 014660 004737 042664          JSR      PC,#SVRH70 ;SAVE ALL THE RH70/RM03 REGISTERS
3111 014664 012764 000040 000010          MOV      #BIT05,RMCS2(R4) ;MASSBUS CLEAR
3112 014672 013764 004164 000010          MOV      #DTADPB,RMCS2(R4) ;SELECT DRIVE
3113 014700 104413          RESREG          ;RESTORE RO-R5
3114 014702 104020          ERROR      20      ;CLOCK OVERFLOWED
3115 014704          ;
3116 014704 012764 000040 000010 8$: MOV      #BIT05,RMCS2(R4) ;MASSBUS INIT.
3117 014712 013764 004164 000010          MOV      #DTADPB,RMCS2(R4) ;SELECT DRIVE
3118 014720 004737 025074          JSR      PC,#ST.CLK ;INITIALIZE THE CLOCK
3119 014724 012777 040320 020564          MOV      #ISR,#RMVEC ;RESTORE RH70/RM03 INT. VECTOR
3120 014732 004037 030252          JSR      RO,#TYPTIM ;GO TYPE THE TIMES
3121 014736 003024          T11          ;POINTER
3122 014740 000004          EXIT14: SCOPE ;LOOP ?
3123
3124 ;*****
3125 ;*TEST 15          MAXIMUM SEEK TIMING TEST
3126
3127 ;*
3128 ;* THIS TEST WILL COMMAND A FORWARD SEEK FROM CYLINDER 0 TO
3129 ;* CYLINDER 'LC', THEN A REVERSE SEEK FROM CYLINDER 'LC' TO
3130 ;* CYLINDER 0. BOTH SEEKS ARE TIMED AND CHECKED TO ENSURE
3131 ;* THEY ARE WITHIN THE TOLERANCE ALLOWED FOR THE MAXIMUM SEEK
3132 ;* TIME. THIS SEQUENCE IS REPEATED 128 TIMES (FOR
3133 ;* A TOTAL OF 256 SEEKS). THE MAXIMUM SEEK TIME MUST BE LESS THAN
3134 ;* 54 MS. 'LC' DEFAULTS TO 410 (10) FOR RMO3/5'S AND TO 814 (10)
3135 ;* FOR RMO3'S.
3136 ;*****
3137 014742          †ST15:
3138 014742 000240          NOP
3139 014744 033737 001456 001234          BIT      #BITS+(15*2),TSTNMS ;DO THIS TEST?
3140 014752 001002          BNE      64$      ;YES--BRANCH
3141 014754 000137 015516          JMP      TST16    ;NO--GO TO THE NEXT TEST
3142 014760 012737 000015 001102 64$: MOV      #15,#$TSTNM ;SET UP TEST NUMBER AND
3143          ;CLEAR THE ERROR FLAG (SERFLG)
3144 014766 004737 025610          JSR      PC,LODPRM ;LOAD THE PARAMETERS FOR THE TEST
3145 014772 012737 014742 001110          MOV      #TST15,#SLPERR ;SETUP THE ERROR LOOP ADDRESS
3146 015000 013777 001102 164134          MOV      $TSTNM,#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
3147 015006 013737 001506 001204          MOV      #RPT,$TIMES ;GET THE ITERATION COUNT
3148 015014 112737 000031 001115          MOVB     #25.,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
3149
3150 ;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
3151 015022 032777 010000 164110          BIT      #SW12,$SWR ;CHECK FOR SWITCH 12 SET
3152 015030 001413          BEQ      20$      ;IF = 0, THEN DON'T TYPE TEST NUMBER
3153 015032 013700 001102          MOV      $TSTNM,RO ;ELSE, TYPE THE NUMBER
3154 015036 042700 177000          BIC      #177000,RO ;MASK OUT THE HIGH ORDER BYTE
3155 015042 104401 001215          TYPE     $CRLF
3156 015046 010046          MOV      RO,-(SP) ;SAVE RO FOR TYPEOUT
3157          ;TEST NUMBER
3158 015050 104403          TYPOS        ;GO TYPE--OCTAL ASCII

```

Address	Hex	Hex	Hex	Hex	Label	OpCode	Comment
3159	015052	002			.BYTE	2	::TYPE 2 DIGIT(S)
3160	015053	000			.BYTE	0	::SUPPRESS LEADING ZEROS
3161	015054	104401	001215		TYPE	,SCRLF	
3162	015060						
3163							
3164	015060	005737	001244		TST	@CLKSTA	:K11-P CLOCK
3165	015064	003002			BGT	15	:YES--START TEST
3166	015066	000137	015516		JMP	TST16	:NO--GO TO NEXT TEST
3167	015072	012737	015072	001106	MOV	#15, @LPAOR	:SETUP THE LOOP ADDRESS
3168	015100	004037	027670		JSR	RO, @SRCHOO	:DO A MASSBUS INIT & RECAL
3169	015104	000402			BR	25	:RETURN HERE IF NO ERROR
3170	015106	000137	015514		JMP	EXIT15	:RETURN HERE ON ERROR
3171	015112	012703	003034		MOV	@T12, R3	:PARAMETER POINTER
3172	015116	012737	015514	001206	MOV	@EXIT15, @ESCAPE	:ESCAPE TO EXIT15 ON ERROR
3173	015124	012706	001100		TEST15: MOV	@STACK, SP	:SETUP STACK
3174	015130	012701	000200		MOV	#128, R1	:REPEAT "O-'LC'-O" 128 TIMES
3175	015134	004737	030054		JSR	PC, @STRTMR	:INIT. THE TIMERS
3176	015140	012777	015410	164226	MOV	@75, @PKV	:SETUP VECTOR IN CASE OF OVERFLOW
3177	015146	012777	030052	020342	MOV	@DORTI, @RMVEC	:SETUP RMO3 VECTOR
3178	015154	005077	164222		15: CLR	@PKB	:START COUNTING FROM ZERO
3179	015160	013764	001512	000034	MOV	LC, RMO3(R4)	:MAXIMUM CYLINDER
3180	015166	012764	000105	000000	MOV	@SEEK, RMO31(R4)	:START A SEEK
3181	015174	012777	000131	164176	MOV	#131, @PKCS	:START THE CLOCK
3182	015202	000001			WAIT		:WAIT ON INTERRUPT
3183	015204	042777	000101	164166	BIC	#101, @PKCS	:STOP THE CLOCK
3184	015212	032764	040000	000012	BIT	@BIT14, RMO3(R4)	:ERR=1?
3185	015220	001415			BEQ	25	:NO--BRANCH
3186	015222	104412			SAVREG		:SAVE R0-R5
3187	015224	012702	004164		MOV	@DTADPB, R2	:DPB POINTER
3188	015230	004737	042664		JSR	PC, @SVRH70	:SAVE ALL THE R170/RMO3 REGISTERS
3189	015234	012764	000040	000010	MOV	@BIT05, RMO32(R4)	:MASSBUS CLEAR
3190	015242	013764	004164	000010	MOV	@DTADPB, RMO32(R4)	:SELECT DRIVE
3191	015250	104413			RESREG		:RESTORE R0-R5
3192	015252	104017			ERROR	17	
3193	015254	005005			25: CLR	R5	:SET THE UP/DOWN SWITCH TO UP
3194	015256	004737	030120		JSR	PC, @COUNT	:UP THE COUNT
3195	015262	004737	027446		JSR	PC, @TWOMS	:STALL FOR 2 MILLISECONDS
3196	015266	005077	164110		CLR	@PKB	:START COUNT AT ZERO
3197	015272	013764	001510	000034	MOV	FC, RMO3(R4)	:BEGINNING CYLINDER
3198	015300	012764	000105	000000	MOV	@SEEK, RMO31(R4)	:START A SEEK
3199	015306	012777	000131	164064	MOV	#131, @PKCS	:START THE CLOCK
3200	015314	000001			WAIT		:WAIT ON INTERRUPT
3201	015316	042777	000101	164054	BIC	#101, @PKCS	:STOP THE CLOCK
3202	015324	032764	040000	000012	BIT	@BIT14, RMO3(R4)	:ERR=1?
3203	015332	001415			BEQ	35	:NO--BRANCH
3204	015334	104412			SAVREG		:SAVE R0-R5
3205	015336	012702	004164		MOV	@DTADPB, R2	:DPB POINTER
3206	015342	004737	042664		JSR	PC, @SVRH70	:SAVE ALL THE R170/RMO3 REGISTERS
3207	015346	012764	000040	000010	MOV	@BIT05, RMO32(R4)	:MASSBUS CLEAR
3208	015354	013764	004164	000010	MOV	@DTADPB, RMO32(R4)	:SELECT DRIVE
3209	015362	104413			RESREG		:RESTORE R0-R5
3210	015364	104017			ERROR	17	:REPORT THE ERROR
3211	015366	012705	177777		35: MOV	#-1, R5	:SET THE UP/DOWN SWITCH TO DOWN
3212	015372	004737	030120		JSR	PC, @COUNT	:UPDATE THE COUNT
3213	015376	004737	027446		JSR	PC, @TWOMS	:STALL FOR 2 MILLISECONDS
3214	015402	005301			DEC	R1	:DONE?

3215	015404	003263				BGT	1\$		; NO--BRANCH
3216	015406	000424				BR	8\$		; YES--EXIT
3217	015410	042777	000101	163762	7\$:	BIC	#101, @PKCS		; STOP THE CLOCK
3218	015416	005037	177776			CLR	@#PS		; DROP THE PRIORITY
3219	015422	012600				MOV	(SP)+, R0		; PC OF WAIT+2
3220	015424	005726				TST	(SP)+		; POP THE PS FROM THE STACK
3221	015426	104412				SAVREG			; SAVE R0-R5
3222	015430	012702	004164			MOV	@DTADPB, R2		; DPB POINTER
3223	015434	004737	042664			JSR	PC, @SVR#70		; SAVE ALL THE RH70/RMO3 REGISTERS
3224	015440	012764	000040	000010		MOV	#BIT05, RMCS2(R4)		; MASSBUS CLEAR
3225	015446	013764	004164	000010		MOV	@DTADPB, RMCS2(R4)		; SELECT DRIVE
3226	015454	104413				RESREG			; RESTORE R0-R5
3227	015456	104020				ERROR	20		; CLOCK OVERFLOWED
3228	015460				8\$:				
3229	015460	012764	000040	000010		MOV	#BIT05, RMCS2(R4)		; MASSBUS INIT.
3230	015466	013764	004164	000010		MOV	@DTADPB, RMCS2(R4)		; SELECT DRIVE
3231	015474	004737	025074			JSR	PC, @ST.CLK		; INITIALIZE THE CLOCK
3232	015500	012777	040320	020010		MOV	#ISR, @RMVEC		; RESTORE RH70/RMO3 INT. VECTOR
3233	015506	004037	030252			JSR	R0, @TYPTIM		; GO TYPE THE TIMES
3234	015512	003034				T12			; POINTER
3235	015514	000004				EXIT15:	SCOPE		; LOOP ?



3236  
3237  
3238

.SBTTL \*\*\* ADDRESSING TESTS \*\*\*

:/\:  
;\:/\;  
:\;/\;

;\*THE ADDRESSING TESTS ENSURES PROPER OPERATION OF THE TRACK  
;\*AND SECTOR ADDRESS CIRCUITRY. BOTH ADDRESSING TESTS  
;\*WILL BE PERFORMED ON CYLINDER FC.

;/\;  
;\;/\;  
:\;/\;

3252  
3253  
3254  
3255

::\*\*\*\*\*  
;\*TEST 16 SECTOR ADDRESSING TEST

3256  
3257  
3258  
3259  
3260  
3261  
3262  
3263  
3264  
3265

;\* THIS TEST WRITES DATA INTO ALL SECTORS OF TRACK "FT". THE  
;\* DATA WILL BE 256 WORDS OF THE SECTOR ADDRESS OF THE SECTOR  
;\* BEING WRITTEN. A WRITE CHECK IS PERFORMED, THE BUFFER IS  
;\* CLEARED (TO 177400) AND THE DATA IS READ AND COMPARED. THEN SECTOR 0  
;\* IS REWRITTEN AND SECTORS 0 - 21 ARE WRITE CHECKED. THEN  
;\* SECTOR 1 IS REWRITTEN AND SECTORS 0 - 21 ARE WRITE CHECKED.  
;\* THIS REWRITE AND WRITE CHECK PROCEDURE IS CONTINUED UP THROUGH  
;\* REWRITE SECTOR 21 AND WRITE CHECK SECTORS 0-21.

3266

::\*\*\*\*\*  
†TST16:

3266	015516			
3267	015516	000240		
3268	015520	033737	001460	001234
3269	015526	001002		
3270	015530	000137	016152	
3271	015534	012737	000016	001102
3272				
3273	015542	004737	025610	
3274	015546	012737	015650	001110
3275	015554	013777	001102	163360
3276	015562	013737	001506	001204
3277	015570	113737	001364	001115
3278				
3279				
3280	015576	032777	010000	163334
3281	015604	001413		
3282	015606	013700	001102	
3283	015612	042700	177000	
3284	015616	104401	001215	
3285	015622	010046		
3286				
3287	015624	104403		
3288	015626	002		
3289	015627	000		
3290	015630	104401	001215	
3291	015634			

```

NOP
BIT @#BITS+<16*2>,TSTNMS ;DO THIS TEST?
BNE 64$ ;YES--BRANCH
JMP TST17 ;NO--GO TO THE NEXT TEST
MOV #16,@#STSTNM ;SET UP TEST NUMBER AND
;CLEAR THE ERROR FLAG (SERFLG)
JSR PC,LOOPRM ;LOAD THE PARAMETERS FOR THE TEST
MOV #TST16,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
MOV $STSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
MOV @#RPT,$TIMES ;GET THE ITERATION COUNT
MOVB ERR.CT,$ERMAX ;MAX ERRORS ALLOWED FOR TEST

;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
BIT #SW12,@SWR ;CHECK FOR SWITCH 12 SET
BEQ 20$ ;IF = 0, THEN DON'T TYPE TEST NUMBER
MOV $STSTNM,R0 ;ELSE, TYPE THE NUMBER
BIC #177000,R0 ;MASK OUT THE HIGH ORDER BYTE
TYPE $CRLF
MOV R0,-(SP) ;;SAVE R0 FOR TYPEOUT
;TEST NUMBER
;GO TYPE--OCTAL ASCII
;TYPE 2 DIGIT(S)
;SUPPRESS LEADING ZEROS

20$:
TYPE ,CRLF

```

F08

MD-11-DZRMF-A RM03 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 64  
T16 SECTOR ADDRESSING TEST

SEQ 0096

3292						
3293	015634	012737	016150	001252		MOV #EXIT16,2#BYPASS
3294	015642	012737	015650	001106		MOV #TEST16,SLPADR ; SETUP THE LOOP ADDRESS
3295	015650	012706	001100		TEST16:	MOV #STACK,SP ; SET THE STACK POINTER
3296	015654	004737	030704			JSR PC,2#FILBUF ; FILL THE BUFFER WITH SECTOR ADDRESSES
3297	015660	013737	001510	004176		MOV 2#FC,2#DTADPB+12 ; CYLINDER
3298	015666	113737	001516	004175		MOV 2#FT,2#DTADPB+11 ; TRACK
3299	015674	105037	004174			CLRB 2#DTADPB+10 ; SECTOR
3300	015700	013737	001352	004170		MOV TRCKWC,2#DTADPB+4 ; WORD COUNT
3301	015706	012737	050514	004172		MOV #BUFFER,2#DTADPB+6 ; BUFFER ADDRESS
3302	015714	012737	015714	001110		MOV #SLPERR ; SETUP THE ERROR LOOP ADDRESS
3303	015722	012706	001100			MOV #STACK,SP ; LOAD THE STACK POINTER
3304	015726	012737	000161	004166		MOV #WRITE,2#DTADPB+2 ; COMMAND=WRITE DATA
3305	015734	004037	026524			JSR RO,2#DRVCL ; START A DATA TRANSFER
3306	015740	012737	000151	004166		MOV #WRCKD,2#DTADPB+2 ; COMMAND=WRITE CHECK DATA
3307	015746	012737	015746	001110		MOV #SLPERR ; SETUP THE ERROR LOOP ADDRESS
3308	015754	012706	001100			MOV #STACK,SP ; LOAD THE STACK POINTER
3309	015760	004037	026524			JSR RO,2#DRVCL ; START A DATA TRANSFER
3310	015764	012737	015764	001110		MOV #SLPERR ; SETUP THE ERROR LOOP ADDRESS
3311	015772	012706	001100			MOV #STACK,SP ; LOAD THE STACK POINTER
3312	015776	004037	030742			JSR RO,2#CLBUF ; CLEAR BUFFER
3313	016002	012737	000171	004166		MOV #READ,2#DTADPB+2 ; COMMAND = READ
3314	016010	004037	026524			JSR RO,2#DRVCL ; START A DATA TRANSFER
3315	016014	004037	031010			JSR RO,2#CKSCTR ; CHECK THE SECTOR DATA READ
3316	016020	012700	050514			MOV #BUFFER,RO ; BUFFER ADDRESS
3317	016024	005001				CLR R1 ; FIRST SECTOR
3318	016026	012737	016026	001110		MOV #SLPERR ; SETUP THE ERROR LOOP ADDRESS
3319	016034	012706	001100			MOV #STACK,SP ; LOAD THE STACK POINTER
3320	016040	012737	000161	004166	IS:	MOV #WRITE,2#DTADPB+2 ; COMMAND=WRITE DATA
3321	016046	012737	177400	004170		MOV #SCTRWC,2#DTADPB+4 ; WORD COUNT
3322	016054	010037	004172			MOV RO,2#DTADPB+6 ; BUFFER ADDRESS
3323	016060	110137	004174			MOV 2#DTADPB+10 ; SECTOR
3324	016064	004037	026524			JSR RO,2#DRVCL ; START A DATA TRANSFER
3325	016070	012737	016070	001110		MOV #SLPERR ; SETUP THE ERROR LOOP ADDRESS
3326	016076	012706	001100			MOV #STACK,SP ; LOAD THE STACK POINTER
3327	016102	012737	000151	004166		MOV #WRCKD,2#DTADPB+2 ; COMMAND=WRITE CHECK DATA
3328	016110	013737	001352	004170		MOV TRCKWC,2#DTADPB+4 ; WORD COUNT
3329	016116	012737	050514	004172		MOV #BUFFER,2#DTADPB+6 ; BUFFER ADDRESS
3330	016124	105037	004174			CLRB 2#DTADPB+10 ; SECTOR
3331	016130	004037	026524			JSR RO,2#DRVCL ; START A DATA TRANSFER
3332	016134	062700	001000			ADD #512.,RO ; MOVE TO NEXT SECTOR
3333	016140	005201				INC R1
3334	016142	023701	001630			CMP PRMLMT+22,R1 ; DONE?
3335	016146	103334				BHIS IS ; NO--BRANCH
3336	016150	000004				EXIT16: SCOPE ; LOOP ?

3337  
3338  
3339  
3340  
3341  
3342  
3343  
3344  
3345  
3346  
3347

```

;*****
;*TEST 17 TRACK ADDRESSING TEST
;
;* THIS TEST WILL WRITE DATA IN THE FORM OF TRACK ADDRESSES
;* IN CYLINDER "FC" SECTOR "FS" OF EVERY TRACK WITH EACH TRACK
;* GETTING ITS OWN TRACK ADDRESS.
;* A WRITE CHECK IS THEN PERFORMED ON EACH TRACK TO ENSURE
;* THE DATA IS VALID. THEN TRACK 0 IS REWRITTEN AND TRACK 1
;* THROUGH TRACK 18 IS WRITE CHECKED. THEN TRACK 1 IS
;* REWRITTEN AND TRACK 2 THROUGH TRACK 18 IS WRITE CHECKED.
;*
```

```

3348 ;* THIS PROCEDURE IS CONTINUED UP THROUGH REWRITING TRACK 17
3349 ;* AND WRITE CHECKING TRACK 18.
3350
3351 :*****
3352 TST17:
3353 NOP
3354 BIT 2#BITS+(17*2),TSTNMS ;DO THIS TEST?
3355 BNE 64$ ;YES--BRANCH
3356 JMP TST20 ;NO--GO TO THE NEXT TEST
3357 64$: MOV #17,2#STSTNM ;SET UP TEST NUMBER AND
3358 ;CLEAR THE ERROR FLAG (SERFLG)
3359 JSR PC,LOOPRM ;LOAD THE PARAMETERS FOR THE TEST
3360 MOV #TST17,2#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
3361 MOV STSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
3362 MOV 2#RPT,$TIMES ;GET THE ITERATION COUNT
3363 MOVB ERR.CT,$SERMAX ;MAX ERRORS ALLOWED FOR TEST
3364
3365 ;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
3366 BIT #SW12,2#SWR ;CHECK FOR SWITCH 12 SET
3367 BEQ 20$ ;IF = 0, THEN DON'T TYPE TEST NUMBER
3368 MOV STSTNM,RO ;ELSE, TYPE THE NUMBER
3369 BIC #177000,RO ;MASK OUT THE HIGH ORDER BYTE
3370 TYPE $CRLF
3371 MOV RO,-(SP) ;;SAVE RO FOR TYPEOUT
3372 ;;TEST NUMBER
3373 TYPOS ;;GO TYPE--OCTAL ASCII
3374 .BYTE 2 ;;TYPE 2 DIGIT(S)
3375 .BYTE 0 ;;SUPPRESS LEADING ZEROS
3376 TYPE , $CRLF
3377 20$:
3378
3379 MOV #EXIT17,2#BYPASS
3380 MOV #TST17,$LPADR ;SETUP THE LOOP ADDRESS
3381 TEST17: MOV #STACK,$P ;SET THE STACK POINTER
3382 JSR PC,2#FILBUF ;FILL THE BUFFER WITH TRACK ADDRESS
3383 MOV #WRITE,2#DTADPB+2 ;COMMAND=WRITE DATA
3384 MOV 2#FC,2#DTADPB+12 ;CYLINDER
3385 MOVB 2#FS,2#DTADPB+10 ;SECTOR
3386 MOV #SCTRNC,2#DTADPB+4 ;WORD COUNT
3387 MOV #BUFFER,2#DTADPB+6 ;BUFFER ADDRESS
3388 CLR RO ;TRACK=0
3389 MOV #,$SLPERR ;SETUP THE ERROR LOOP ADDRESS
3390 MOV #STACK,$P ;LOAD THE STACK POINTER
3391 1$: MOVB RO,2#DTADPB+11 ;TRACK ADDRESS
3392 JSR RO,2#DRVCAL ;START A DATA TRANSFER
3393 ADD #256.*2.,2#DTADPB+6 ;UPDATE BUFFER ADDRESS
3394 INC RO ;UPDATE TRACK NUMBER
3395 CMP #5,RO ;OUT OF TRACKS?
3396 BGT 1$ ;NO--BRANCH
3397 MOV #BUFFER,2#DTADPB+6 ;BUFFER ADDRESS
3398 CLR RO
3399 MOV #,$SLPERR ;SETUP THE ERROR LOOP ADDRESS
3400 MOV #STACK,$P ;LOAD THE STACK POINTER
3401 MOV #WRCKD,2#DTADPB+2 ;COMMAND=WRITE CHECK
3402 2$: MOVB RO,2#DTADPB+11 ;TRACK ADDRESS
3403 JSR RO,2#DRVCAL ;START A DATA TRANSFER

```

# H08

MD-11-DZRMF-A RMO3 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 66  
T17 TRACK ADDRESSING TEST

SEQ 0098

3404	016454	062737	001000	004172	ADD	#256.*2.,@#DTADPB+6	:UPDATE BUFFER ADDRESS
3405	016462	005200			INC	RO	:UPDATE TRACK NUMBER
3406	016464	022700	000005		CMP	#5,RO	:OUT OF TRACKS?
3407	016470	003365			BGT	2\$	:NO--BRANCH
3408	016472	005000			CLR	RO	:FIRST TRACK ADDRESS
3409	016474	110037	004172		MOV	RO,@#DTADPB+11	:TRACK
3410	016500	010001			MOV	RO,R1	:FORM BUFFER ADDRESS
3411	016502	012737	050514	004172	MOV	#BUFFER,@#DTADPB+6	:FORM BUFFER ADDRESS
3412	016510	005301			DEC	R1	
3413	016512	002411			BLT	5\$	
3414	016514	062737	001000	004172	ADD	#256.*2.,@#DTADPB+6	
3415	016522	000772			BR	4\$	
3416	016524	012737	016524	001110	MOV	#.SLPERR	:SETUP THE ERROR LOOP ADDRESS
3417	016532	012706	001100		MOV	#STACK,SP	:LOAD THE STACK POINTER
3418	016536	012737	000161	004166	MOV	#WRITE,@#DTADPB+2	:COMMAND=WRITE DATA
3419	016544	004037	026524		JSR	RO,@#DRVCAL	:START A DATA TRANSFER
3420	016550	062737	001000	004172	ADD	#256.*2.,@#DTADPB+6	:UPDATE BUFFER ADDRESS
3421	016556	105237	004175		INCB	@#DTADPB+11	:MOVE TO NEXT TRACK
3422	016562	012737	016562	001110	MOV	#.SLPERR	:SETUP THE ERROR LOOP ADDRESS
3423	016570	012706	001100		MOV	#STACK,SP	:LOAD THE STACK POINTER
3424	016574	012737	000151	004166	MOV	#WRCKD,@#DTADPB+2	:COMMAND=WRITE CHECK DATA
3425	016602	004037	026524		JSR	RO,@#DRVCAL	:START A DATA TRANSFER
3426	016606	122737	000004	004175	CMPB	#4,@#DTADPB+11	:OUT OF TRACKS?
3427	016614	003355			BGT	6\$	:NO--BRANCH
3428	016616	005200			INC	RO	:NEXT TRACK TO WRITE
3429	016620	022700	000004		CMP	#4,RO	:OUT OF TRACKS?
3430	016624	003323			BGT	3\$	:NO--BRANCH
3431	016626	000004			EXIT17:	SCOPE	

3432  
3433  
3434  
3435  
3436  
3437  
3438  
3439  
3440  
3441  
3442  
3443  
3444  
3445  
3446  
3447  
3448  
3449  
3450  
3451  
3452  
3453  
3454  
3455  
3456  
3457  
3458  
3459  
3460  
3461  
3462  
3463  
3464  
3465  
3466  
3467  
3468  
3469  
3470  
3471  
3472  
3473  
3474  
3475  
3476  
3477  
3478  
3479  
3480  
3481  
3482  
3483  
3484  
3485  
3486  
3487

.SBTTL \*\*\* DATA TEST \*\*\*

\*\*\*\*\*  
\*TEST 20 DATA TEST

\* THIS TEST PERFORMS DATA STORAGE AND RETRIEVAL ON CYLINDERS  
\* "FC" THROUGH "LC" BY THE INCREMENT "IC" USING THE DATA PATTERNS  
\* SPECIFIED. THE FOLLOWING SEQUENCE OCCURS FOR EACH CYLINDER:  
\* 1. SET "NT" TO "FT" THEN REPEAT 2-4 UNTIL "NT" > "LT"  
\* 2. WRITE THEN WRITE CHECK "FS" THROUGH "LS" OF TRACK "NT"  
\* 3. READ THEN SOFTWARE COMPARE "FS" THROUGH "LS" OF TRACK "NT"  
\* 4. INCREMENT "NT" BY "IT"  
\* 5. REPEAT STEPS 1-4 FOR EACH DATA PATTERN  
\* 6. REPEAT STEPS 1-5 FOR "FC" THROUGH "LC" ADVANCING BY "IC"

\* IF A WRITE CHECK ERROR OCCURS THE ERROR IS REPORTED AND  
\* THE TRACK IN ERROR IS REWRITTEN AND CHECKED. THIS CHECK IS  
\* ACCOMPLISHED BY PERFORMING TWO(2) SUCCESSIVE ERROR FREE  
\* WRITE CHECKS. IF THE CHECK FAILS THE ERROR IS REPORTED AS  
\* FATAL AND NO READ OCCURS.  
\* FS DEFAULTS TO 1 AND LS DEFAULTS TO 0  
\* PAT DEFAULTS TO 177777 (ALL POSSIBLE PATTERNS)  
\* THE POSSIBLE PATTERNS ARE:

PAT 0	PAT 1	PAT 2	PAT 3	PAT 4	PAT 5	PAT 6	PAT 7
165555	000001	177776	000000	000000	052525	007417	026455
133333	000003	177774	000000	010421	052525	007417	026455
165555	000007	177770	000000	021042	052525	007417	026455
133333	000017	177760	177777	031463	125252	170360	151322
165555	000037	177740	177777	042104	125252	170360	151322
133333	000077	177700	177777	052525	125252	170360	151322
165555	000177	177600	000000	063146	052525	007417	026455
133333	000377	177400	000000	073567	052525	007417	026455
165555	000777	177000	177777	104210	125252	170360	151322
133333	001777	176000	177777	114631	125252	170360	151322
165555	003777	174000	000000	125252	052525	007417	026455
133333	007777	170000	177777	135673	125252	170360	151322
165555	017777	160000	000000	146314	052525	007417	026455
133333	037777	140000	177777	156735	125252	170360	151322
165555	077777	100000	000000	167356	052525	007417	026455
133333	177777	000000	177777	177777	125252	170360	151322
PAT 8	PAT 9	PAT 10	PAT 11	PAT 12	PAT 13	PAT 14	PAT 15
165555	000001	177776	172666	077777	153333	000000	177777
133333	000002	177775	155555	137777	066667	177777	000000
165555	000004	177773	172666	157777	153333	177777	000000
133333	000010	177767	155555	167777	066667	177777	000000
165555	000020	177757	172666	173777	153333	177777	000000
133333	000040	177737	155555	175777	066667	177777	000000
165555	000100	177677	172666	176777	153333	177777	000000
133333	000200	177577	155555	177377	066667	177777	000000
165555	000400	177377	172666	177577	153333	177777	000000
133333	001000	176777	155555	177677	066667	177777	000000
165555	002000	175777	172666	177737	153333	177777	000000

```

3488 ;* 133333 004000 173777 155555 177757 066667 177777 000000
3489 ;* 165555 010000 167777 172666 177767 153333 177777 000000
3490 ;* 133333 020000 157777 155555 177773 066667 177777 000000
3491 ;* 165555 040000 137777 172666 177775 153333 177777 000000
3492 ;* 133333 100000 077777 155555 177776 066667 177777 000000
3493 ;*
3494 ;*
3495 ;*

```

```

:*****
TST20:

```

```

3496 016630 NOP
3497 016630 000240 BIT BITS+(20*2-40),TSTNMS+2 ;DO THIS TEST ?
3498 016632 033737 001424 001236 BNE 64$ ;YES--BRANCH
3499 016640 001002 BNE 64$ ;NO--GO TO THE NEXT TEST
3500 016642 000137 017422 JMP TST21 ;SET UP TEST NUMBER AND
3501 016646 012737 000020 001102 64$: MOV #20,#STSTNM ;CLEAR THE ERROR FLAG (SERFLG)
3502 JSR PC,LOOPRM ;LOAD THE PARAMETERS FOR THE TEST
3503 016654 004737 025610 MOV #TST20,#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
3504 016660 012737 017070 001110 MOV STSTNM,#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
3505 016666 013777 001102 162246 MOV #RPT,$TIMES ;GET THE ITERATION COUNT
3506 016674 013737 001506 001204 MOVB ERR.CT,$SERMAX ;MAX ERRORS ALLOWED FOR TEST
3507 016702 113737 001364 001115

```

```

3508 ;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
3509 BIT #SW12,$SWR ;CHECK FOR SWITCH 12 SET
3510 016710 032777 010000 162222 BEQ 20$ ;IF = 0, THEN DON'T TYPE TEST NUMBER
3511 016716 001413 MOV $STSTNM,R0 ;ELSE, TYPE THE NUMBER
3512 016720 013700 001102 BIC #177000,R0 ;MASK OUT THE HIGH ORDER BYTE
3513 016724 042700 177000 TYPE $CRLF
3514 016730 104401 001215 MOV R0,-(SP) ;SAVE R0 FOR TYPEOUT
3515 016734 010046 ;TEST NUMBER

```

```

3516 ;GO TYPE--OCTAL ASCII
3517 016736 104403 .BYTE 2 ;TYPE 2 DIGIT(S)
3518 016740 002 .BYTE 0 ;SUPPRESS LEADING ZEROS
3519 016741 000 TYPE ,CRLF
3520 016742 104401 001215 20$:
3521 016746

```

```

3522 ;SET UP THE TRACK WORD COUNT FOR DATA TRANSFER
3523
3524
3525

```

```

3526 016746 012737 016746 001106 MOV #,$SLPADR ;SETUP THE LOOP ADDRESS
3527 016754 005000 CLR R0 ;CLEAR SWITCH
3528 016756 005004 CLR R4 ;FORM WORD COUNT IN R4
3529 016760 013701 001526 MOV #LS,R1 ;LOAD LAST SECTOR
3530 016764 163701 001524 SUB #FS,R1 ;COMPARE WITH FIRST SECTOR
3531 016770 002004 BGE 1$ ;SKIP NEXT IF FS < OR = LS
3532
3533 016772 063701 001630 ADD PRMLT+22,R1 ;ADD MAXIMUM SECTOR ADDRESS TO
3534 016776 005201 INC R1 ;MAKE THE DIFFERENCE POSITIVE
3535 017000 005100 COM R0 ;SET SWITCH FOR FS > LS
3536
3537 017002 062704 000400 1$: ADD #256.,R4 ;WORDS/SECTOR
3538 017006 005301 DEC R1 ;LS - FS SECTORS MINUS ONE
3539 017010 002374 BGE 1$ ;INCR. WORD COUNT IF NO. SECTORS STILL +
3540 017012 005404 NEG R4 ;FORM NEGATIVE WORD COUNT
3541 017014 010405 MOV R4,R5 ;COPY NORMAL WORD COUNT INTO SMALL WC
3542 017016 005700 TST R0 ;FS > LS SWITCH SET?
3543 017020 001412 BEQ 3$ ;NO, SKIP NEXT

```

```

3544
3545 017022 005005          CLR      R5          ;FORM WORD COUNT FOR FS > LS
3546 017024 013701 001630  MOV      PRMLT+22,R1 ;MAX SECTOR ADDRESS
3547 017030 163701 001524  SUB      @#FS,R1     ;SUBTRACT THE FIRST SECTOR
3548 017034 062705 000400  2$: ADD      #256.,R5 ;WORDS/SECTOR
3549 017040 005301          DEC      R1          ;NO SECTORS MINUS ONE
3550 017042 002374          BGE     2$          ;INCR. WORD COUNT IF NO. SECTORS STILL +
3551 017044 005405          NEG      R5          ;FORM NEGATIVE WORD COUNT FOR THIS CASE
3552
3553          ;SET UP FOR DATA TRANSFERS AND PATTERN VARIATIONS
3554
3555 017046 113737 001524 004174 3$: MOVVB   @#FS,@#DTADPB+10 ;SECTOR
3556 017054 012737 050514 004172  MOV      #BUFFER,@#DTADPB+6 ;DATA BUFFER
3557 017062 012737 017420 001252  MOV      #EXIT20,@#BYPASS
3558 017070 012706 001100  TEST20: MOV     #STACK,SP ;LOAD THE STACK POINTER
3559 017074 005037 001334  CLR      @#WCEFLG    ;CLEAR THE WRITE CHECK ERROR FLAG
3560 017100 013701 001510  MOV      @#FC,R1     ;PICKUP FIRST CYLINDER
3561 017104 000407          BR       2$          ;SKIP PATTERN SET-UP FIRST TIME THRU
3562
3563 017106 005720          1$: TST     (R0)+      ;MOVE TO NEXT DATA PATTERN
3564 017110 022700 000040  CMP      #16.*2.,R0 ;OUT OF PATTERNS?
3565 017114 003004          BGT     3$          ;NO, STAY ON THIS CYLINDER UNTIL DONE
3566 017116 004037 030630  JSR     R0,@#INCCYL ;MOVE TO NEXT CYLINDER
3567 017122 000536          BR       EXIT20     ;OUT OF CYLINDERS ----->
3568
3569          ;DO NEXT CYLINDER
3570
3571 017124 005000          2$: CLR      R0          ;START WITH PATTERN 0
3572 017126 036037 001424 001530 3$: BIT     BITS(R0),@#PAT ;THIS PATTERN SELECTED?
3573 017134 001764          BEQ     1$          ;NO, GO BACK AND GET ONE THAT WAS
3574 017136 013702 001516  MOV      @#FT,R2     ;FIRST TRACK
3575 017142 010137 004176  MOV      R1,@#DTADPB+12 ;LOAD THE CYLINDER
3576 017146 110237 004175  4$: MOVVB   R2,@#DTADPB+11 ;LOAD THE TRACK
3577
3578 017152 020127 001466  CMP      R1,#822.    ;CHECK FOR LAST DISK CYLINDER
3579 017156 001003          BNE     10$         ;SKIP LAST TRACK CHECK IF NOT
3580 017160 020227 000004  CMP      R2,#4      ;LAST DISK TRACK ?
3581 017164 001515          BEQ     EXIT20     ;DON'T TEST LAST TRACK AS IT HAS BAD
3582 017166          10$:          ;BLOCK INFORMATION STORED ON IT ----->
3583
3584 017166 010437 004170  MOV      R4,@#DTADPB+4 ;LOAD THE WORD COUNT
3585 017172 023701 001512  CMP      LC,R1       ;LAST DISK CYLINDER TO TEST ?
3586 017176 003005          BGT     5$          ;NO, SKIP TRACK CHECK
3587 017200 022702 000004  CMP      #4,R2       ;LAST DISK TRACK TO TEST ?
3588 017204 003002          BGT     5$          ;NO, SKIP NEXT
3589 017206 010537 004170*  MOV      R5,@#DTADPB+4 ;SHORT WORD COUNT
3590 017212 017703 161722  5$: MOV      @SWR,R3    ;INHIBIT WRITE AND
3591 017216 005103          COM     R3          ;WRITE CHECK?
3592 017220 032703 000030  BIT     #SW04!SW03,R3
3593 017224 001436          BEQ     7$          ;YES--BRANCH
3594 017226 004737 031360  JSR     PC,@#SETBUF  ;MOVE DATA PATTERN INTO THE BUFFER
3595 017232 032777 000020 161700  BIT     #SW04,@SWR  ;INHIBIT WRITE?
3596 017240 001012          BNE     6$          ;YES, DO NEXT PORTION OF TESTING
3597 017242 012737 017242 001110  MOV      #,$LPERR    ;SETUP THE ERROR LOOP ADDRESS
3598 017250 012706 001100  MOV      #STACK,SP   ;LOAD THE STACK POINTER
3599 017254 012737 000161 004166  MOV      #WRITE,@#DTADPB+2 ;COMMAND=WRITE DATA

```

```

3600
3601 ;DO A DATA TRANSFER
3602
3603 017262 004037 026524 JSR RO,@#DRVCAI ;START A DATA TRANSFER
3604 017266 032777 000010 161644 6$: BIT #SW03,@SWR ;INHIBIT WRITE CHECK?
3605 017274 001012 BNE 7$ ;YES--BRANCH
3606 017276 012737 017276 001110 MOV #,SLPERR ;SETUP THE ERROR LOOP ADDRESS
3607 017304 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
3608 017310 012737 000151 004166 MOV #WRCKD,@#DTADPB+2 ;COMMAND=WRITE CHECK DATA
3609 017316 004037 026524 JSR RO,@#DRVCAI ;START A DATA TRANSFER
3610 017322 005737 001334 7$: TST @#WCEFLG ;WRITE CHECK ERROR FLAG SET?
3611 017326 001404 BEQ 8$ ;NO--BRANCH
3612 017330 032777 000001 161602 BIT #SW00,@SWR ;PERFORM READ AFTER FATAL "WCE"
3613 017336 001424 BEQ 9$ ;NO--BRANCH
3614 017340 032777 000004 161572 8$: BIT #SW02,@SWR ;INHIBIT READ DATA AND SOFTWARE COMPARE?
3615 017346 001020 BNE 9$ ;YES--BRANCH
3616 017350 012737 017350 001110 MOV #,SLPERR ;SETUP THE ERROR LOOP ADDRESS
3617 017356 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
3618 017362 012737 000171 004166 MOV #READ,@#DTADPB+2 ;COMMAND=READ
3619 017370 004037 026524 JSR RO,@#DRVCAI ;START A DATA TRANSFER
3620 017374 032777 000002 161536 BIT #SW01,@SWR ;COMPARE THE DATA?
3621 017402 001002 BNE 9$ ;NO--BRANCH
3622 017404 004737 031450 JSR PC,@#DATCMP ;YES--DO IT
3623 017410 004037 030600 9$: JSR RO,@#INCTRK ;MOVE TO NEXT TRACK
3624 017414 000634 BR 1$ ;OUT OF TRACKS GO TO NEXT PATTERN
3625 017416 000653 BR 4$ ;LOOP
3626 017420 000004 EXIT20: SCOPE ;SCOPE LOOP

```



3627  
3628  
3629  
3630  
3631  
3632  
3633  
3634  
3635  
3636  
3637  
3638  
3639  
3640  
3641  
3642  
3643  
3644  
3645  
3646  
3647  
3648  
3649  
3650  
3651  
3652  
3653  
3654  
3655  
3656  
3657  
3658  
3659  
3660  
3661  
3662  
3663  
3664  
3665  
3666  
3667  
3668  
3669  
3670  
3671  
3672  
3673  
3674  
3675  
3676  
3677  
3678  
3679  
3680  
3681  
3682

```
.SBTTL *** EXERCISE TEST ***
;*****
;TEST 21 RANDOM ADDRESS AND RANDOM PATTERN TEST
;* STARTING AT "FC" AND GOING SEQUENTIALLY TO "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 1000.
;* 1) GENERATE A RANDOM SECTOR ADDRESS
;* 2) WRITE A RANDOM PATTERN AT THE ADDRESS
;* GENERATED IN 1.
;* 3) GENERATE A NEW RANDOM SECTOR ADDRESS
;* 4) READ THE SECTOR AT THE ADDRESS
;* GENERATED IN 3.
;* 5) DO A SOFTWARE CHECK OF THE DATA READ IN 4 AGAINST
;* THE ORIGINAL RANDOM PACK DATA.
;* 6) DO A WRITE CHECK OF THE DATA WRITTEN IN 2
;* 7) GENERATE A NEW RANDOM SECTOR 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
```

```
017422
017422 000240
017424 033737 001426 001236
017432 001002
017434 000137 020236
017440 012737 000021 001102 645:
017446 004737 025610
017452 012737 017730 001110
017460 013777 001102 161454
017466 013737 001506 001204
017474 113737 001364 001115
017502 032777 010000 161430
017510 001413
017512 013700 001102
017516 042700 177000
017522 104401 001215
017526 010046
017530 104403
017532 002
017533 000
017534 104401 001215
017540
```

```
;*****
TST21:
NOP
BIT BITS+(21*2-40),TSTNMS+2 ;DO THIS TEST ?
BNE 645 ;YES--BRANCH
JMP TST22 ;NO--GO TO THE NEXT TEST
MOV #21,#TSTNM ;SET UP TEST NUMBER AND
;CLEAR THE ERROR FLAG ($ERFLG)
JSR PC,LODPRM ;LOAD THE PARAMETERS FOR THE TEST
MOV #TST21,#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
MOV $TSTNM,$DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
MOV #RPT,$TIMES ;GET THE ITERATION COUNT
MOVB ERR.CT,$ERMAX ;MAX ERRORS ALLOWED FOR TEST

;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
BIT #SW12,$SWR ;CHECK FOR SWITCH 12 SET
BEQ 205 ;IF = 0, THEN DON'T TYPE TEST NUMBER
MOV $TSTNM,RO ;ELSE, TYPE THE NUMBER
BIC #177000,RO ;MASK OUT THE HIGH ORDER BYTE
TYPE $CRLF
MOV RO,-(SP) ;;SAVE RO FOR TYPEOUT
;;TEST NUMBER
;;GO TYPE--OCTAL ASCII
;;TYPE 2 DIGIT(5)
;;SUPPRESS LEADING ZEROS
205:
TYPE ,CRLF
```

3683	017540	012737	017540	001106		MOV	#,SLPADR	;SETUP THE LOOP ADDRESS
3684	017546	012737	020234	001252		MOV	#EXIT21,2#BYPASS	
3685	017554	012737	176543	024606		MOV	#176543,2#MINUM	;PRIME THE RANDOM NUMBER GENERATOR
3686	017562	012737	123456	024610		MOV	#123456,2#LONUM	
3687	017570	013737	001510	004176		MOV	2#FC,2#DTADPB+12	;CYLINDER
3688	017576	013737	001352	004170		MOV	TRKWC,2#DTADPB+4	;WORD COUNT FOR 32/30 SECTORS (FULL TRACK)
3689	017604	012737	050514	004172		MOV	#BUFFER,2#DTADPB+6	;BUFFER ADDRESS
3690	017612	012737	000161	004166		MOV	#WRITE,2#DTADPB+2	;COMMAND
3691	017620	032737	100000	001220		BIT	#SW15,2#C.SWR	;WRITE THE DISK PACK BEFORE TESTING?
3692	017626	001027				BNE	3\$	;NO--BEGIN TESTING
3693	017630	004037	031766			JSR	RD,2#FILRAN	;FILL DATA BUFFER WITH RANDOM DATA
3694	017634	005037	004174		15:	CLR	2#DTADPB+10	;SECTOR AND TRACK
3695	017640	012737	017640	001110		MOV	#,SLPERR	;SETUP THE ERROR LOOP ADDRESS
3696	017646	012706	001100			MOV	#STACK,SP	;LOAD THE STACK POINTER
3697	017652				25:			
3698	017652	004037	026524			JSR	RD,2#DRVCAL	;START A DATA TRANSFER
3699	017656	105237	004175			INCB	2#DTADPB+11	;NEXT TRACK
3700	017662	122737	000005	004175		CMPB	#5,2#DTADPB+11	;TIME FOR NEXT CYLINDER ?
3701	017670	003370				BGT	2\$	;NO--DO NEXT TRACK ON THIS CYL.
3702	017672	005237	004176			INC	2#DTADPB+12	;INCR CYLINDER ADDRESS
3703	017676	023737	001512	004176		CMP	2#LC,2#DTADPB+12	;OUT OF CYLINDERS?
3704	017704	002353				BGE	1\$	;NO--CONTINUE SEQUENTIAL RANDOM WRITE
3705								
3706								
3707	017706	012737	177400	004170	35:	MOV	#SCTRWC,2#DTADPB+4	;WORD COUNT
3708	017714	012737	017730	001106		MOV	#TEST21,2#SLPADR	
3709	017722	012737	017730	001110		MOV	#TEST21,2#SLPERR	
3710	017730	012706	001100		TEST21:	MOV	#STACK,SP	;SET STACK POINTER
3711								
3712	017734	004037	032242			JSR	RD,2#RANADR	;GENERATE A RANDOM ADDRESS
3713	017740	013737	004174	001340		MOV	2#DTADPB+10,2#SVADR	;SAVE THE TRACK/SECTOR
3714	017746	013737	004176	001342		MOV	2#DTADPB+12,2#SVADR+2	;SAVE THE CYLINDER
3715	017754	012737	000161	004166		MOV	#WRITE,2#DTADPB+2	;COMMAND=WRITE DATA
3716	017762	012701	050514			MOV	#BUFFER,R1	;WRITE BUFFER ADDRESS FOR "RANPAT"
3717	017766	010137	004172			MOV	R1,2#DTADPB+6	;INTO THE DATA PARAMETER BLOCK
3718	017772	004037	032206			JSR	RD,2#RANPAT	;GENERATE RANDOM 256 WORD PATTERN
3719								;AND PUT INTO THE WRITE BUFFER
3720	017776	012737	017776	001110		MOV	#,SLPERR	;SETUP THE ERROR LOOP ADDRESS
3721	020004	012706	001100			MOV	#STACK,SP	;LOAD THE STACK POINTER
3722	020010	004037	026524			JSR	RD,2#DRVCAL	;START A DATA TRANSFER
3723								
3724	020014	004037	032242			JSR	RD,2#RANADR	;GENERATE A NEW RANDOM ADDRESS
3725	020020	012737	000171	004166		MOV	#READ,2#DTADPB+2	;COMMAND=READ DATA
3726	020026	012737	051514	004172		MOV	#BUFFER+512,2#DTADPB+6	;READ BUFFER ADDRESS
3727	020034	012737	020034	001110		MOV	#,SLPERR	;SETUP THE ERROR LOOP ADDRESS
3728	020042	012706	001100			MOV	#STACK,SP	;LOAD THE STACK POINTER
3729	020046	004037	026524			JSR	RD,2#DRVCAL	;START A DATA TRANSFER
3730	020052	004037	032010			JSR	RD,2#RANCK	;SOFTWARE CHECK THE DATA
3731								
3732	020056	013737	001340	004174		MOV	2#SVADR,2#DTADPB+10	;GET ADDRESS OF WHERE THE LAST
3733	020064	013737	001342	004176		MOV	2#SVADR+2,2#DTADPB+12	;WRITE WAS PERFORMED
3734	020072	012737	000151	004166		MOV	#WRCKD,2#DTADPB+2	;COMMAND=WRITE CHECK DATA
3735	020100	012737	050514	004172		MOV	#BUFFER,2#DTADPB+6	;DATA BUFFER ADDRESS FOR HARDWARE
3736								;CHECK OF THE DATA
3737	020106	012737	020106	001110		MOV	#,SLPERR	;SETUP THE ERROR LOOP ADDRESS
3738	020114	012706	001100			MOV	#STACK,SP	;LOAD THE STACK POINTER

```

3739 020120 004037 026524 JSR RO,2#DRVCL ;START A DATA TRANSFER
3740
3741 020124 004037 032242 JSR RO,2#RANADR ;GENERATE A NEW RANDOM ADDRESS
3742 020130 012737 00C171 004166 MOV #READ,2#DTADPB+2 ;COMMAND=READ
3743 020136 012737 051514 004172 MOV #BUFFER+512.,2#DTADPB+6 ;DATA BUFFER ADDRESS
3744 020144 012737 020144 001110 MOV #SLPERR ;SETUP THE ERROR LOOP ADDRESS
3745 020152 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
3746 020156 004037 026524 JSR RO,2#DRVCL ;START A DATA TRANSFER
3747 020162 004037 032010 JSR RO,2#RANCK ;SOFTWARE CHECK THE DATA
3748
3749 020166 013737 001340 004174 MOV 2#SVADR,2#DTADPB+10 ;GET DISK ADDRESS OF THE
3750 020174 013737 001342 004176 MOV 2#SVADR+2,2#DTADPB+12 ;LAST WRITE
3751 020202 012737 000151 004166 MOV #WRCKD,2#DTADPB+2 ;COMMAND=WRITE CHECK DATA
3752 020210 012737 050514 004172 MOV #BUFFER,2#DTADPB+6 ;DATA BUFFER ADDRESS
3753 020216 012737 020216 001110 MOV #SLPERR ;SETUP THE ERROR LOOP ADDRESS
3754 020224 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
3755 020230 004037 026524 JSR RO,2#DRVCL ;START A DATA TRANSFER
3756 020234 000004 EXIT21: SCOPE ;LOOP ?
3757
3758 .SBTTL *** RMO3 ACCESS TIME ADJUSTMENT TEST ***
3759
3760 ;*****
3761 ;*TEST 22 RMO3 ACCESS TIME ADJUSTMENT TEST
3762
3763 ;* THIS TEST PERFORMS SEEKS BETWEEN CYLINDERS 0 & 136 TO ALLOW THE
3764 ;* OPERATOR TO ADJUST THE ACCESS TIME ON AN RMO3 USING THE
3765 ;* DDU. THE PROGRAM STALLS APPROXIMATELY 5 SECONDS BETWEEN SEEKS
3766 ;* SO THAT THE ACCESS TIME INDICATORS ON THE DDU MAY BE OBSERVED.
3767
3768 ;*****
3769 †ST22:
3770 020236 000240 NOP
3771 020240 033737 001430 001236 BIT BITS+(22*2-40),TSTNMS+2 ;DO THIS TEST ?
3772 020246 001002 BNE 645 ;YES--BRANCH
3773 020250 000137 020406 JMP SEOP ;NO--GO TO THE END OF THE PROGRAM
3774 020254 012737 000022 001102 645: MOV #22,2#STSTNM ;SET UP TEST NUMBER AND
3775 ;CLEAR THE ERROR FLAG (SERFLG)
3776 020262 004737 025610 JSR PC,LODPRM ;LOAD THE PARAMETERS FOR THE TEST
3777 020266 012737 020324 001110 MOV #TEST22,2#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
3778 020274 013777 001102 160640 MOV STSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
3779 020302 013737 001506 001204 MOV 2#RPT,$TIMES ;GET THE ITERATION COUNT
3780 020310 112737 000144 001115 MOVB #100,SERMAX ;MAX ERRORS ALLOWED FOR TEST
3781 020316 012737 020324 001106 MOV #TEST22,SLPADR ;SETUP THE LOOP ADDRESS
3782 020324 012706 001100 TEST22: MOV #STACK,SP ;SETUP THE STACK POINTER
3783 020330 013737 001512 004116 MOV LC,DPB.A+12 ;ENDING CYLINDER
3784 020336 112737 000105 004106 MOVB #SEEK,2#DPB.A+2 ;SEEK=COMMAND
3785 020344 004037 026032 JSR RO,2#CALL.A ;GO EXECUTE THE COMMAND
3786 020350 004037 027364 JSR RO,STALL ;STALL
3787 020354 001360 .WORD STALL3 ;ADDRESS OF STALL VALUE
3788 020356 013737 001510 004116 MOV FC,DPB.A+12 ;STARTING CYLINDER
3789 020364 112737 000105 004106 MOVB #SEEK,2#DPB.A+2 ;SEEK=COMMAND
3790 020372 004037 026032 JSR RO,2#CALL.A ;GO EXECUTE THE COMMAND
3791 020376 004037 027364 JSR RO,STALL ;STALL
3792 020402 001360 .WORD STALL3 ;ADDRESS OF STALL VALUE
3793 020404 000004 EXIT22: SCOPE ;LOOP ?
3794

```

```

3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805 020406
3806 020406 104401 020414
3807 020412 000410
3808
3809 020434
3810 020434 005737 001232
3811 020440 001434
3812 020442 104401 020450
3813 020446 000405
3814
3815 020462
3816 020462 013746 001254
3817 020466 104403
3818 020470 002
3819 020471 000
3820 020472 104401 020500
3821 020476 000412
3822
3823 020524
3824 020524 013746 001112
3825 020530 104402
3826 020532 005037 001112
3827 020536 005037 001102
3828 020542 005037 001204
3829 020546 005237 001100
3830 020552 042737 100000 001100
3831 020560 005327
3832 020562 000010
3833 020564 003027
3834 020566 012737
3835 020570 000010
3836 020572 020562
3837 020574 104401 020602
3838 020600 000407
3839
3840 020620
3841 020620 104401 020650
3842 020624 013700 000042
3843 020630 001405
3844 020632 000005
3845 020634 004710
3846 020636 000240
3847 020640 000240
3848 020642 000240
3849 020644
3850 020644 000137

```

.SBTTL END OF PASS ROUTINE

```

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

```

```

SEOP:
        TYPE      65$          ;; TYPE ASCIZ STRING
        BR        64$          ;; GET OVER THE ASCIZ
;; 65$: .ASCIZ  <CR><LF><LF>/END OF PASS/
64$:
        TST       2#DRVSEL     ;; ANY DRIVES SELECTED?
        BEQ       1$          ;; NO--BRANCH
        TYPE      67$          ;; TYPE ASCIZ STRING
        BR        66$          ;; GET OVER THE ASCIZ
;; 67$: .ASCIZ  / ON DRIVE/
66$:
        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      69$          ;; TYPE ASCIZ STRING
        BR        68$          ;; GET OVER THE ASCIZ
;; 69$: .ASCIZ  / ERRORS DETECTED=/
68$:
        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       $PASS        ;; INCREMENT THE PASS NUMBER
        BIC       #100000,$PASS ;; DON'T ALLOW A NEG. NUMBER
        DEC       (PC)+        ;; LOOP?
SEOPCT: .WORD      8.
        BGT       $DOAGN       ;; YES
        MOV       (PC)+,2(PC)+ ;; RESTORE COUNTER
SENDCT: .WORD      8.
;; 65$: .ASCIZ  <CR><LF>/END OF TEST/
64$:
        TYPE      $ENULL       ;; TYPE NULL CHARACTER
        MOV       2#42,RO      ;; GET MONITOR ADDRESS
        BEQ       $DOAGN       ;; BRANCH IF NO MONITOR
        RESET     ;; CLEAR THE WORLD
SENDAD: JSR       PC,(RO)     ;; GO TO MONITOR
        NOP       ;; SAVE ROOM
        NOP       ;; FOR
        NOP       ;; ACT11
$DOAGN: JMP       2(PC)+      ;; RETURN

```

009

MD-11-DZRMF-A RM03 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 75  
END OF PASS ROUTINE

SEQ 0107

3851	020646	006052			\$RTNAD: .WORD	RESTART	
3852	020650	377	377	000	\$ENULL: .BYTE	-1,-1,0	:::NULL CHARACTER STRING
3853	020654				.EVEN		

3854  
3855  
3856  
3857  
3858  
3859  
3860  
3861  
3862  
3863  
3864  
3865  
3866  
3867  
3868  
3869  
3870  
3871  
3872  
3873  
3874  
3875  
3876  
3877  
3878  
3879  
3880  
3881  
3882  
3883  
3884  
3885  
3886  
3887  
3888  
3889  
3890  
3891  
3892  
3893  
3894  
3895  
3896  
3897  
3898  
3899  
3900  
3901  
3902  
3903  
3904  
3905  
3906  
3907  
3908  
3909

.SBTTL \*\*\* SYSMAC SUBROUTINES \*\*\*

.SBTTL ERROR HANDLER ROUTINE

```
*****
; THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
; SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
; AND GO TO TYPERR ON ERROR
; THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
; SW15=1      HALT ON ERROR
; SW13=1      INHIBIT ERROR TYPEOUTS
; SW10=1      BELL ON ERROR
; SW09=1      LOOP ON ERROR
; CALL
; *          ERROR N          ;; ERROR=EMT AND N=ERROR ITEM NUMBER
```

SERROR:

```
CKSWR          ;; TEST FOR CHANGE IN SOFT-SWR
BIT            #SW08, @SWR      ;; SEND ERROR MESSAGE TO TTY?
BEQ            7$              ;; YES--BRANCH
TST           @#LPTAVL        ;; IS THERE A LINE PRINTER AVAILABLE?
BEQ            7$              ;; NO--BRANCH
MOV           @#LPS, @#STPS    ;; YES--SETUP STATUS
MOV           @#LPB, @#STPB    ;; AND BUFFER REG.'S FOR LINE PRINTER
7$: INCB       $ERFLG         ;; SET THE ERROR FLAG
BEQ            7$              ;; DON'T LET THE FLAG GO TO ZERO
MOV           $STNM, @DISPLAY  ;; DISPLAY TEST NUMBER AND ERROR FLAG
BIT           #BIT10, @SWR    ;; BELL ON ERROR?
BEQ            1$              ;; NO - SKIP
TYPE          $BELL           ;; RING BELL
1$: INC        $ERTTL         ;; COUNT THE NUMBER OF ERRORS
MOV           (SP), $ERRPC     ;; GET ADDRESS OF ERROR INSTRUCTION
SUB           @2, $ERRPC
MOVB         @SWR, @ITEMB     ;; STRIP AND SAVE THE ERROR ITEM CODE
BIT           #BIT13, @SWR    ;; SKIP TYPEOUT IF SET
BNE           20$             ;; SKIP TYPEOUTS
JSR          PC, TYPERR       ;; GO TO USER ERROR ROUTINE
TYPE          , $CRLF

20$:
2$: TST        @SWR           ;; HALT ON ERROR
BPL           3$              ;; SKIP IF CONTINUE
HALT          ;; HALT ON ERROR!
CKSWR        ;; TEST FOR CHANGE IN SOFT-SWR
BIT           #BIT09, @SWR    ;; LOOP ON ERROR SWITCH SET?
BEQ            4$              ;; BR IF NO
MOV           $LPERR, (SP)    ;; FUDGE RETURN FOR LOOPING
4$: TST        $ESCAPE        ;; CHECK FOR AN ESCAPE ADDRESS
BEQ            5$              ;; BR IF NONE
MOV           $ESCAPE, (SP)  ;; FUDGE RETURN ADDRESS FOR ESCAPE
5$: MOV        @#TPS, @#STPS  ;; SET STATUS AND BUFFER REG.'S
MOV           @#TPB, @#STPB  ;; FOR TTY
RTI           ;; RETURN FROM ERROR CALL
```

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

```

3910 .SBTTL TYPERR - TYPE ERROR ROUTINE
3911 ; THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE
3912 ; WHICH ERROR IS TO BE REPORTED, IT THEN OBTAINS, FROM THE "ERROR
3913 ; TABLE" ($ERRTB), AND REPORTS THE APPROPRIATE INFORMATION
3914 ; CONCERNING THE ERROR.
3915 ; CALL
3916 ;
3917 ; JSR PC, @#TYPERR
3918 ; RETURN
3919 021062 113737 001102 001176 TYPERR: MOVB @#STSTNM, @#STMPO ; SAVE THE TEST NUMBER
3920 021070 104412 SAVREG ; SAVE R0 - R5
3921 021072 162700 000004 SUB #4, R0 ; FORM TEST PC
3922 021076 010037 001162 MOV R0, @#$REG0 ; COPY R0-R5 IN $REG0-$REG5
3923 021102 010137 001164 MOV R1, @#$REG1
3924 021106 010237 001166 MOV R2, @#$REG2
3925 021112 010337 001170 MOV R3, @#$REG3
3926 021116 010437 001172 MOV R4, @#$REG4
3927 021122 010537 001174 MOV R5, @#$REG5
3928 021126 113700 001114 MOVB @#$ITEMB, R0 ; PICKUP ERROR ITEM NUMBER
3929 021132 010001 MOV R0, R1 ; AND COPY IT INTO R1
3930 021134 005300 DEC R0 ; FORM INDEX FOR ERROR TABLE
3931 021136 106300 ASLB R0
3932 021140 106300 ASLB R0
3933 021142 106300 ASLB R0
3934 021144 103002 BCC 1$ ; IS ERROR > 37?
3935 021146 062700 000240 ADD #ITEM41-$ERRTB, R0 ; YES--FORM OFFSET
3936 021152 062700 004306 1$: ADD $ERRTB, R0 ; FORM ADDRESS
3937 021156 012037 021172 MOV (R0)+, 2$ ; GET ERROR MESSAGE (EM) POINTER
3938 021162 001447 BEQ 7$ ; BRANCH IF THERE ISN'T ONE
3939 021164 104401 001215 TYPE , $CRLF ; "CARRIAGE RETURN - LINE FEED"
3940 021170 104401 TYPE
3941 021172 000000 2$: .WORD 0 ; "EM" POINTER GOES HERE
3942 021174 162701 000041 SUB #41, R1 ; SPECIAL ERROR ITEM NUMBER?
3943 021200 100440 BMI 7$ ; NO--BRANCH
3944 021202 013701 001260 MOV @#SVSTAT, R1 ; GET STATUS/ERROR INDICATOR
3945 021206 106301 ASLB R1 ; STRIP "DONE" BIT (BIT07)
3946 021210 006301 ASL R1 ; STRIP "ERROR" BIT (BIT15)
3947 021212 012702 004254 MOV #STATBL, R2 ; 1ST ADDRESS ON STATUS MESSAGE POINTERS
3948 021216 005003 CLR R3 ; CARRIAGE RETURN-LINE FEED SWITCH
3949 021220 104401 021226 TYPE , 65$ ; TYPE ASCIZ STRING
3950 021224 000402 BR 64$ ; GET OVER THE ASCIZ
3951 ;: 65$: .ASCIZ / (/
3952 021232 64$:
3953 021232 012237 021254 3$: MOV (R2)+, 5$ ; MESSAGE POINTER
3954 021236 006301 ASL R1 ; TYPE THIS MESSAGE?
3955 021240 103013 BCC 6$ ; NO--BRANCH
3956 021242 005103 COM R3 ; YES--TYPE A "CR" & "LF"?
3957 021244 001002 BNE 4$ ; NO--BRANCH
3958 021246 104401 001215 TYPE , $CRLF ; YES
3959 021252 104401 4$: TYPE
3960 021254 000000 5$: .WORD 0 ; MESSAGE POINTER GOES HERE
3961 021256 005701 TST R1 ; MORE TO TYPE?
3962 021260 001403 BEQ 6$ ; NO--BRANCH
3963 021262 104401 044657 TYPE , MSG.SP ; YES--SPACES
3964 021266 000761 BR 3$ ; LOOP
3965 021270 001360 6$: BNE 3$ ; BRANCH IF NOT FINISHED

```

```

3966 021272 104401 021300          TYPE      67$      ;;TYPE ASCIZ STRING
3967 021276 000401                BR          66$      ;;GET OVER THE ASCIZ
3968                                ;;67$: .ASCIZ  /)/
3969 021302                66$:
3970 021302 012037 021316          7$: MOV      (R0)+,B$  ;PICK UP DATA HEADER (DH) POINTER
3971 021306 001404                BEQ          9$      ;BRANCH IF NONE
3972 021310 104401 001215          TYPE      ,SCLF    ;CARRIAGE RETURN-LINE FEED
3973 021314 104401                TYPE
3974 021316 000000          8$: .WORD      0      ;"DH" POINTER GOES HERE
3975 021320 012001          9$: MOV      (R0)+,R1  ;PICKUP DATA TABLE (DT) POINTER
3976 021322 001450                BEQ          20$     ;BRANCH IF NONE
3977 021324 005005                CLR          R5      ;SET INDENT SWITCH
3978 021326 012000          MOV      (R0)+,R0  ;DATA FORMAT (DF) POINTER
3979 021330 012002          MOV      (R0)+,R2  ;NUMBER OF DH'S TO TYPE
3980 021332 001441          BEQ          17$     ;BRANCH IF DH NUMBER IS 0
3981 021334 005105          COM          R5      ;NO INDENT
3982 021336 104401 001215          TYPE      ,SCLF    ;CARRIAGE RETURN-LINE FEED
3983 021342 112003          10$: MOVB    (R0)+,R3  ;NUMBER OF DATA WORDS TO TYPE
3984 021344 112004          MOVB    (R0)+,R4  ;AND HOW TO TYPE THEM
3985 021346 006004          11$: ROR     R4      ;OCTAL OR DECIMAL?
3986 021350 103403          BCS      12$     ;DECIMAL--BRANCH
3987 021352 013146          MOV      2(R1)+,-(SP) ;SAVE 2(R1)+ FOR TYPEOUT
3988 021354 104402          TYPOC
3989 021356 000402          BR          13$
3990 021360          12$:
3991 021360 013146          MOV      2(R1)+,-(SP) ;SAVE 2(R1)+ FOR TYPEOUT
3992 021362 104405          TYPDS
3993 021364 005303          13$: DEC     R3      ;GO TYPE--DECIMAL ASCII WITH SIGN
3994 021366 001403          BEQ      14$     ;MORE NUMBERS TO TYPE?
3995 021370 104401 044657          TYPE      ,MSG.SP  ;NO--BRANCH
3996 021374 000764          BR          11$     ;YES--TYPE SEPERATORS
3997 021376 005302          14$: DEC     R2      ;LOOP
3998 021400 003421          BLE     20$     ;MORE DH'S?
3999 021402 104401 001215          TYPE      ,SCLF    ;NO--BRANCH
4000 021406 005105          COM          R5      ;YES--START A NEW LINE
4001 021410 001002          BNE     15$     ;INDENT?
4002 021412 104401 044657          TYPE      ,MSG.SP  ;NO--BRANCH
4003 021416 012037 021424          15$: MOV      (R0)+,16$ ;YES--TYPE SPACES
4004 021422 104401          TYPE
4005 021424 000000          16$: .WORD      0      ;GET NEXT DH
4006 021426 104401 001215          TYPE      ,SCLF    ;AND TYPE IT
4007 021432 005705          TST      R5      ;DH POINTER GOES HERE
4008 021434 001342          BNE     10$     ;CARRIAGE RETURN-LINE FEED
4009 021436 104401 044657          17$: TYPE      ,MSG.SP ;INDENT?
4010 021442 000737          BR          10$     ;NO--BRANCH
4011 021444 104413          20$: RESREG
4012 021446 000207          RTS      PC      ;YES--TYPE SPACES
4013                                ;LOOP
4014                                ;RESTORE R0 - R5
4015                                ;RETURN

```

.SBTTL TYPE ROUTINE

```

4016 *****
4017 *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
4018 *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
4019 *NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
4020 *NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
4021 *NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.

```



```

4022
4023
4024
4025
4026
4027
4028
4029
4030
4031 021450 105737 001157
4032 021454 100002
4033 021456 000000
4034 021460 000407
4035 021462 010046
4036 021464 017600 000002
4037 021470 112046
4038 021472 001005
4039 021474 005726
4040 021476 012600
4041 021500 062716 000002
4042 021504 000002
4043 021506 122716 000011
4044 021512 001430
4045 021514 122716 000200
4046 021520 001006
4047 021522 005726
4048 021524 104401
4049 021526 001215
4050 021530 105037 021664
4051 021534 000755
4052 021536 004737 021620
4053 021542 123726 001156
4054 021546 001350
4055 021550 013746 001154
4056
4057 021554 105366 000001
4058 021560 002770
4059 021562 004737 021620
4060 021566 105337 021664
4061 021572 000770
4062
4063
4064
4065 021574 112716 000040
4066 021600 004737 021620
4067 021604 132737 000007 021664
4068 021612 001372
4069 021614 005726
4070 021616 000724
4071 021620 105777 157324
4072 021624 100375
4073 021626 116677 000002 157316
4074 021634 122766 000015 000002
4075 021642 001003
4076 021644 105037 021664
4077 021650 000406

; *
; *CALL:
; *1) USING A TRAP INSTRUCTION
; * TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
; *OR
; * TYPE
; * MESADR
; *
$TYPE: TSTB $TPFLG ;; IS THERE A TERMINAL?
BPL 1$ ;; BF IF YES
HALT ;; HALT HERE IF NO TERMINAL
BR 3$ ;; LEAVE
1$: MOV R0,-(SP) ;; SAVE R0
MOV 2$(SP),R0 ;; GET ADDRESS OF ASCIZ STRING
2$: MOVB (R0)+,-(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
60$: MOV (SP)+,R0 ;; RESTORE R0
3$: ADD #2,(SP) ;; ADJUST RETURN PC
RTI ;; RETURN
4$: CMPB #HT,(SP) ;; BRANCH IF <HT>
BEQ 8$
CMPB #CRLF,(SP) ;; BRANCH IF NOT <CRLF>
BNE 5$
TST (SP)+ ;; POP <CR><LF> EQUIV
TYPE ;; TYPE A CR AND LF
$CRLF
CLRB $CHARCNT ;; CLEAR CHARACTER COUNT
BR 2$ ;; GET NEXT CHARACTER
5$: JSR PC,$TYPEC ;; GO TYPE THIS CHARACTER
6$: CMPB $FILLC,(SP)+ ;; IS IT TIME FOR FILLER CHARS.?
BNE 2$ ;; IF NO GO GET NEXT CHAR.
MOV $NULL,-(SP) ;; GET # OF FILLER CHARS. NEEDED
AND THE NULL CHAR.
7$: DECB 1(SP) ;; DOES A NULL NEED TO BE TYPED?
BLT 6$ ;; BR IF NO--GO POP THE NULL OFF OF STACK
JSR PC,$TYPEC ;; GO TYPE A NULL
DECB $CHARCNT ;; DO NOT COUNT AS A COUNT
BR 7$ ;; LOOP

;HORIZONTAL TAB PROCESSOR
8$: MOVB #' (SP) ;; REPLACE TAB WITH SPACE
9$: JSR PC,$TYPEC ;; TYPE A SPACE
BITB #7,$CHARCNT ;; BRANCH IF NOT AT
BNE 9$ ;; TAB STOP
TST (SP)+ ;; POP SPACE OFF STACK
BR 2$ ;; GET NEXT CHARACTER
$TYPEC: TSTB 2$TPS ;; WAIT UNTIL PRINTER IS READY
BPL $TYPEC
MOVB 2(SP),2$TPB ;; LOAD CHAR TO BE TYPED INTO DATA REG.
CMPB #CR,2(SP) ;; IS CHARACTER A CARRIAGE RETURN?
BNE 1$ ;; BRANCH IF NO
CLRB $CHARCNT ;; YES--CLEAR CHARACTER COUNT
BR $TYPEX ;; EXIT

```

```

4078 021652 122766 000012 000002 1$:  CMPB  #LF,2(SP)  ;; IS CHARACTER A LINE FEED?
4079 021660 001402  BEQ  $TYPEX  ;; BRANCH IF YES
4080 021662 105227  INCB  (PC)+  ;; COUNT THE CHARACTER
4081 021664 000000  $CHARCNT: .WORD 0  ;; CHARACTER COUNT STORAGE
4082 021666 000207  $TYPEX: RTS  PC

4083
4084
4085
4086
4087
4088
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4110 021670 017646 000000 022113 $TYPOS: MOV 2(SP),-(SP)  ;; PICKUP THE MODE
4111 021674 116637 000001  MOVB 1(SP),$OFILL  ;; LOAD ZERO FILL SWITCH
4112 021702 112637 022115  MOVB (SP)+,$OMODE+1  ;; NUMBER OF DIGITS TO TYPE
4113 021706 062716 000002  ADD #2,(SP)  ;; ADJUST RETURN ADDRESS
4114 021712 000406  BR $TYPON
4115 021714 112737 000001 022113 $TYPOC: MOVB #1,$OFILL  ;; SET THE ZERO FILL SWITCH
4116 021722 112737 000006 022115  MOVB #6,$OMODE+1  ;; SET FOR SIX(6) DIGITS
4117 021730 112737 000005 022112 $TYPON: MOVB #5,$OCNT  ;; SET THE ITERATION COUNT
4118 021736 010346  MOV R3,-(SP)  ;; SAVE R3
4119 021740 010446  MOV R4,-(SP)  ;; SAVE R4
4120 021742 010546  MOV R5,-(SP)  ;; SAVE R5
4121 021744 113704 022115  MOVB $OMODE+1,R4  ;; GET THE NUMBER OF DIGITS TO TYPE
4122 021750 005404  NEG R4
4123 021752 062704 000006  ADD #6,R4  ;; SUBTRACT IT FOR MAX. ALLOWED
4124 021756 110437 022114  MOVB R4,$OMODE  ;; SAVE IT FOR USE
4125 021762 113704 022113  MOVB $OFILL,R4  ;; GET THE ZERO FILL SWITCH
4126 021766 016605 000012  MOV 12(SP),R5  ;; PICKUP THE INPUT NUMBER
4127 021772 005003  CLR R3  ;; CLEAR THE OUTPUT WORD
4128 021774 006105 1$: ROL R5  ;; ROTATE MSB INTO "C"
4129 021776 000404  BR 3$  ;; GO DO MSB
4130 022000 006105 2$: ROL R5  ;; FORM THIS DIGIT
4131 022002 006105  ROL R5
4132 022004 006105  ROL R5
4133 022006 010503  MOV R5,R3

```

```

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

```

```

4134 022010 006103
4135 022012 105337 022114
4136 022016 100016
4137 022020 042703 177770
4138 022024 001002
4139 022026 005704
4140 022030 001403
4141 022032 005204
4142 022034 052703 000060
4143 022040 052703 000040
4144 022044 110337 022110
4145 022050 104401 022110
4146 022054 105337 022112
4147 022060 003347
4148 022062 002402
4149 022064 005204
4150 022066 000744
4151 022070 012605
4152 022072 012604
4153 022074 012603
4154 022076 016666 000002 000004
4155 022104 012616
4156 022106 000002
4157 022110 000
4158 022111 000
4159 022112 000
4160 022113 000
4161 022114 000000
4162
4163
4164
4165
4166
4167
4168
4169
4170
4171
4172
4173
4174
4175 022116
4176 022116 010046
4177 022120 010146
4178 022122 010246
4179 022124 010346
4180 022126 010546
4181 022130 012746 020200
4182 022134 016605 000020
4183 022140 100004
4184 022142 005405
4185 022144 112766 000055 000001
4186 022152 005000
4187 022154 012703 022332
4188 022160 112723 000040
4189 022164 005002
    
```

```

3$: ROL R3 ; GET LSB OF THIS DIGIT
    DECB $OMODE ; TYPE THIS DIGIT?
    BPL 7$ ; BR IF NO
    BIC #177770,R3 ; GET RID OF JUNK
    BNE 4$ ; TEST FOR 0
    TST R4 ; SUPPRESS THIS 0?
    BEQ 5$ ; BR IF YES
4$: INC R4 ; DON'T SUPPRESS ANYMORE 0'S
    BIS #'0,R3 ; MAKE THIS DIGIT ASCII
5$: BIS #'R3 ; MAKE ASCII IF NOT ALREADY
    MOVB R3,8$ ; SAVE FOR TYPING
    TYPE 8$ ; GO TYPE THIS DIGIT
7$: DECB $OCNT ; COUNT BY 1
    BGT 2$ ; BR IF MORE TO DO
    B.T 6$ ; BR IF DONE
    INC R4 ; INSURE LAST DIGIT ISN'T A BLANK
    BR 2$ ; GO DO THE LAST DIGIT
6$: MOV (SP)+,R5 ; RESTORE R5
    MOV (SP)+,R4 ; RESTORE R4
    MOV (SP)+,R3 ; RESTORE R3
    MOV 2(SP),4(SP) ; SET THE STACK FOR RETURNING
    MOV (SP)+,(SP)
    RTI ; RETURN
8$: .BYTE 0 ; STORAGE FOR ASCII DIGIT
    .BYTE 0 ; TERMINATOR FOR TYPE ROUTINE
$OCNT: .BYTE 0 ; OCTAL DIGIT COUNTER
$OFILL: .BYTE 0 ; ZERO FILL SWITCH
$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
    
```

```

$TYPDS: MOV R0,-(SP) ; PUSH R0 ON STACK
        MOV R1,-(SP) ; PUSH R1 ON STACK
        MOV R2,-(SP) ; PUSH R2 ON STACK
        MOV R3,-(SP) ; PUSH R3 ON STACK
        MOV R5,-(SP) ; PUSH R5 ON STACK
        MOV #20200,-(SP) ; SET BLANK SWITCH AND SIGN
        MOV 20(SP),R5 ; GET THE INPUT NUMBER
        BPL 1$ ; BR IF INPUT IS POS.
        NEG R5 ; MAKE THE BINARY NUMBER POS.
        MOVB #'-,1(SP) ; MAKE THE ASCII NUMBER NEG.
1$: CLR R0 ; ZERO THE CONSTANTS INDEX
    MOV #50BLK,R3 ; SETUP THE OUTPUT POINTER
    MOVB #' ,(R3)+ ; SET THE FIRST CHARACTER TO A BLANK
2$: CLR R2 ; CLEAR THE BCD NUMBER
    
```

```

4190 022166 016001 022322
4191 022172 160105
4192 022174 002402
4193 022176 005202
4194 022200 000774
4195 0 32 060105
4196 0 54 005702
4197 022206 001002
4198 0 210 105716
4199 0 212 100407
4200 0 214 106316
4201 0 216 103003
4202 0 220 116663 000001 177777
4203 0 226 052702 000060
4204 022232 052702 000040
4205 022236 110223
4206 0 240 005720
4207 0 242 020027 000010
4208 022246 002746
4209 022250 003002
4210 022252 010502
4211 022254 000764
4212 0 256 105726
4213 02 160 100003
4214 022262 116663 177777 177776
4215 022270 105013
4216 022272 012605
4217 022274 012603
4218 022276 012602
4219 022300 012601
4220 022302 012600
4221 022304 104401 022332
4222 022310 016666 000002 000004
4223 022316 012616
4224 022320 000002
4225 022322 023420
4226 022324 001750
4227 022326 000144
4228 022330 000012
4229 022332 000004
4230
4231
4232
4233
4234
4235 022342 000000
4236 022344 000000
4237 022346 000000
4238 022350 000002
4239
4240
4241
4242
4243
4244
4245

3$: MOV $DTBL(R0),R1 ;;GET THE CONSTANT
SUB R1,R5 ;;FORM THIS BCD DIGIT
BLT 4$ ;;BR IF DONE
INC R2 ;;INCREASE THE BCD DIGIT BY 1
BR 3$

4$: ADD R1,R5 ;;ADD BACK THE CONSTANT
TST R2 ;;CHECK IF BCD DIGIT=0
BNE 5$ ;;FALL THROUGH IF 0
TSTB (SP) ;;STILL DOING LEADING 0'S?
BMI 7$ ;;BR IF YES
ASLB (SP) ;;MSD?
BCC 6$ ;;BR IF NO
MOVB 1(SP),-1(R3) ;;YES--SET THE SIGN
BIS #'0,R2 ;;MAKE THE BCD DIGIT ASCII
BIS #' ,R2 ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOVB R2,(R3)+ ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST (R0)+ ;;JUST INCREMENTING
CMP R0,#10 ;;CHECK THE TABLE INDEX
BLT 2$ ;;GO DO THE NEXT DIGIT
BGT 8$ ;;GO TO EXIT
MOV R5,R2 ;;GET THE LSD
BR 6$ ;;GO CHANGE TO ASCII
TSTB (SP)+ ;;WAS THE LSD THE FIRST NON-ZERO?
BPL 9$ ;;BR IF NO
MOVB -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
CLRB (R3) ;;SET THE TERMINATOR
MOV (SP)+,R5 ;;POP STACK INTO R5
MOV (SP)+,R3 ;;POP STACK INTO R3
MOV (SP)+,R2 ;;POP STACK INTO R2
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,R0 ;;POP STACK INTO R0
TYPE $DBLK ;;NOW TYPE THE NUMBER
MOV 2(SP),4(SP) ;;ADJUST THE STACK
MOV (SP)+,(SP)
RTI ;;RETURN TO USER

$DTBL: 10000.
1000.
100.
10.

$DBLK: .BLKW 4

.SBTTL TTY INPUT ROUTINE

;*****
;ENABL LSB
$TKCNT: .WORD 0 ;;NUMBER OF ITEMS IN QUEUE
$TKQIN: .WORD 0 ;;INPUT POINTER
$TKQOUT: .WORD 0 ;;OUTPUT POINTER
$TKQSRT: .BLKB 2 ;;TTY KEYBOARD QUEUE
$TKQEND=.

;*TK INITIALIZE ROUTINE
;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
;*CALL:

```

```

4246      *      JSR      PC,STKINT
4247      *      RETURN
4248
4249 022352 005037 022342 $TKINT: CLR      STKCNT      ;; CLEAR COUNT OF ITEMS IN QUEUE
4250 022356 012737 022350 022344 MOV      #STKQSR,STKQIN ;; MOVE THE STARTING ADDRESS OF THE
4251 022364 013737 022344 022346 MOV      STKQIN,STKQOUT ;; QUEUE INTO THE INPUT & OUTPUT POINTERS.
4252 022372 012737 022422 000060 MOV      #STKSRV,#TKVEC ;; INITIALIZE THE KEYBOARD VECTOR
4253 022400 012737 000200 000062 MOV      #200,#TKVEC+2 ;; "BR" LEVEL 4
4254 022406 005777 156534 TST      #TKB      ;; CLEAR DONE FLAG
4255 022412 012777 000100 156524 MOV      #100,#TKS ;; ENABLE TTY KEYBOARD INTERRUPT
4256 022420 000207 RTS      PC      ;; RETURN TO CALLER
4257
4258
4259 *TK SERVICE ROUTINE
4260 *THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
4261 *BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
4262 *IT IN THE QUEUE.
4263 *IF THE CHARACTER IS A "CONTROL-C" (↑C) $TKINT IS CALLED AND
4264 *UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (START2)
4265 022422 117746 156520 $TKSRV: MOVB    #TKB,-(SP) ;; PICKUP THE CHARACTER
4266 022426 042716 177600 BIC      #↑C177,(SP) ;; STRIP THE JUNK
4267 022432 021627 000003 CMP      (SP),#3 ;; IS IT A CONTROL C?
4268 022436 001007 BNE      15 ;; BRANCH IF NO
4269 022440 104401 023552 TYPE     #CNTLC ;; TYPE A CONTROL-C (↑C)
4270 022444 004737 022352 JSR      PC,STKINT ;; INIT THE KEYBOARD
4271 022450 005726 TST      (SP)+ ;; CLEAN UP STACK
4272 022452 000137 004660 JMP      START2 ;; CONTROL C RESTART
4273 022456 021627 000007 15: CMP      (SP),#7 ;; IS IT A CONTROL G?
4274 022462 001004 BNE      25 ;; BRANCH IF NO
4275 022464 022737 000176 001140 CMP      #SWREG,SWR ;; IS SOFT-SWR SELECTED?
4276 022472 001500 BEQ      65 ;; GO TO SWR CHANGE
4277
4278
4279 022474 022737 000002 022342 25: CMP      #2,STKCNT ;; IS THE QUEUE FULL?
4280 022502 001004 BNE      35 ;; BRANCH IF NO
4281 022504 104401 001210 TYPE     #BELL ;; RING THE TTY BELL
4282 022510 005726 TST      (SP)+ ;; CLEAN CHARACTER OFF OF STACK
4283 022512 000451 BR       55 ;; EXIT
4284 022514 021627 000023 35: CMP      (SP),#23 ;; IS IT A CONTROL-S?
4285 022520 001021 BNE      325 ;; BRANCH IF NO
4286 022522 005077 156416 CLR      #TKS ;; DISABLE TTY KEYBOARD INTERRUPTS
4287 022526 005726 TST      (SP)+ ;; CLEAN CHAR OFF STACK
4288 022530 105777 156410 315: TSTB    #TKS ;; WAIT FOR A CHAR
4289 022534 100375 BPL      315 ;; LOOP UNTIL ITS THERE
4290 022536 117746 156404 MOVB    #TKB,-(SP) ;; GET THE CHARACTER
4291 022542 042716 177600 BIC      #↑C177,(SP) ;; MAKE IT 7-BIT ASCII
4292 022546 022627 000021 CMP      (SP)+,#21 ;; IS IT A CONTROL-Q?
4293 022552 001366 BNE      315 ;; BRANCH IF NO
4294 022554 012777 000100 156362 MOV      #100,#TKS ;; REENABLE TTY KEYBOARD INTERRUPTS
4295 022562 000002 RTI      ;; RETURN
4296 022564 005237 022342 325: INC      STKCNT ;; COUNT THIS CHARACTER
4297 022570 021627 000140 CMP      (SP),#140 ;; IS IT UPPER CASE?
4298 022574 002405 BLT      45 ;; BRANCH IF YES
4299 022576 021627 000175 CMP      (SP),#175 ;; IS IT A SPECIAL CHAR?
4300 022602 003002 BGT      45 ;; BRANCH IF YES
4301 022604 042716 000040 BIC      #40,(SP) ;; MAKE IT UPPER CASE

```



4358	023030	021627	000025	95:	CMP	(SP),#25	:: IS IT A CONTROL-U?	
4359	023034	001005			BNE	105	:: BRANCH IF NOT	
4360	023036	104401	023557		TYPE	\$CNTLU	:: YES, ECHO CONTROL-U (↑U)	
4361	023042	062706	000006	205:	ADD	#6,SP	:: IGNORE PREVIOUS INPUT	
4362	023046	000737			BR	195	:: LET'S TRY IT AGAIN	
4363								
4364								
4365	023050	021627	000015	105:	CMP	(SP),#15	:: IS IT A <CR>?	
4366	023054	001022			BNE	165	:: BRANCH IF NO	
4367	023056	005766	000004		TST	4(SP)	:: YES, IS IT THE FIRST CHAR?	
4368	023062	001403			BEQ	115	:: BRANCH IF YES	
4369	023064	016677	000002	156046	MOV	2(SP),JSWR	:: SAVE NEW SWR	
4370	023072	062706	000006		ADD	#6,SP	:: CLEAR UP STACK	
4371	023076	104401	001215	115:	TYPE	\$CRLF	:: ECHO <CR> AND <LF>	
4372	023102	123727	001135	000001	145:	CMPB	\$INTAG,#1	:: RE-ENABLE TTY KBD INTERRUPTS?
4373	023110	001003			BNE	155	:: BRANCH IF NOT	
4374	023112	012777	000100	156024	MOV	#100,JSWKS	:: RE-ENABLE TTY KBD INTERRUPTS	
4375	023120	000002			RTI		:: RETURN	
4376	023122	004737	021620	155:	JSR	PC,\$TYPEC	:: ECHO CHAR	
4377	023126	021627	000060	165:	CMP	(SP),#60	:: CHAR < 0?	
4378	023132	002420			BLT	185	:: BRANCH IF YES	
4379	023134	021627	000067		CMP	(SP),#67	:: CHAR > 7?	
4380	023140	003015			BGT	185	:: BRANCH IF YES	
4381	023142	042726	000060		BIC	#60,(SP)+	:: STRIP-OFF ASCII	
4382	023146	005766	000002		TST	2(SP)	:: IS THIS THE FIRST CHAR	
4383	023152	001403			BEQ	175	:: BRANCH IF YES	
4384	023154	006316			ASL	(SP)	:: NO, SHIFT PRESENT	
4385	023156	006316			ASL	(SP)	:: CHAR OVER TO MAKE	
4386	023160	006316			ASL	(SP)	:: ROOM FOR NEW ONE.	
4387	023162	005266	000002	175:	INC	2(SP)	:: KEEP COUNT OF CHAR	
4388	023166	056616	177776		BIS	-2(SP),(SP)	:: SET IN NEW CHAR	
4389	023172	000667			BR	75	:: GET THE NEXT ONE	
4390	023174	104401	001214	185:	TYPE	\$QUES	:: TYPE ?<CR><LF>	
4391	023200	000720			BR	205	:: SIMULATE CONTROL-U	
4392					.DSABL	LSB		

```

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

```

4403	023202	011646			\$RDCHR:	MOV	(SP),-(SP)	:: PUSH DOWN THE PC AND
4404	023204	016666	000004	000002		MOV	4(SP),2(SP)	:: THE PS
4405	023212	005066	000004			CLR	4(SP)	:: GET READY FOR A CHARACTER
4406	023216	005046				CLR	-(SP)	:: PUT NEW PS ON STACK
4407	023220	012746	023226			MOV	#645,-(SP)	:: PUT NEW PC ON STACK
4408	023224	000002				RTI		:: POP NEW PC AND PS
4409	023226				645:			
4410	023226	005737	022342		15:	TST	\$STKCNT	:: WAIT ON A CHARACTER
4411	023232	001775				BEQ	15	
4412	023234	005337	022342			DEC	\$STKCNT	:: DECREMENT THE COUNTER
4413	023240	117766	177102	000004		MOVB	\$STKQOUT,4(SP)	:: GET ONE CHARACTER

```

4414 023246 005237 022346          INC      $TKQOUT          ;; UPDATE THE POINTER
4415 023252 023727 022346 022352  CMP      $TKQOUT,#$TKQEND ;; DID IT GO OFF OF THE END?
4416 023260 001003                BNE      2$              ;; BRANCH IF NO
4417 023262 012737 022350 022346  MOV      #$TKQRT,$TKQOUT ;; RESE. THE POINTER
4418 023270 000002                RTI                      ;; RETURN
4419                                     ;; *****
4420                                     ;; THIS ROUTINE WILL INPUT A STRING FROM THE TTY
4421                                     ;; *CALL:
4422                                     ;; *
4423                                     ;; *   RDLIN
4424                                     ;; *   RETURN HERE
4425                                     ;; *
4426                                     ;; INPUT A STRING FROM THE TTY
4427                                     ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
4428                                     ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
4426 023272 010346          SRDLIN: MOV      R3,-(SP)          ;; SAVE R3
4427 023274 005046          CLR      -(SP)          ;; CLEAR THE RUBOUT KEY
4428 023276 012703 023526 1$:  MOV      #$TTYIN,R3      ;; GET ADDRESS
4429 023302 022703 023552 2$:  CMP      #$TTYIN+20.,R3  ;; BUFFER FULL?
4430 023306 101456          BLOS     4$              ;; BR IF YES
4431 023310 104410          R0CHR   ;; GO READ ONE CHARACTER FROM THE TTY
4432 023312 112613          MOVVB   (SP)+,(R3)      ;; GET CHARACTER
4433 023314 122713 000177 10$:  CMPB    #177,(R3)      ;; IS IT A RUBOUT
4434 023320 001022          BNE     5$              ;; BR IF NO
4435 023322 005716          TST    (SP)            ;; IS THIS THE FIRST RUBOUT?
4436 023324 001007          BNE     6$              ;; BR IF NO
4437 023326 112737 000134 023524 MOVVB   #' \,9$        ;; TYPE A BACK SLASH
4438 023334 104401 023524          TYPE   9$
4439 023340 012716 177777          MOV    #-1,(SP)        ;; SET THE RUBOUT KEY
4440 023344 005303          DEC    R3              ;; BACKUP BY ONE
4441 023346 020327 023526          CMP    R3,$$TTYIN     ;; STACK EMPTY?
4442 023352 103434          BLO    4$              ;; BR IF YES
4443 023354 111337 023524          MOVVB  (R3),9$        ;; SETUP TO TYPEOUT THE DELETED CHAR.
4444 023360 104401 023524          TYPE   9$
4445 023364 000746          BR     2$              ;; GO READ ANOTHER CHAR.
4446 023366 005716          5$:  TST    (SP)            ;; RUBOUT KEY SET?
4447 023370 001406          BEQ    7$              ;; BR IF NO
4448 023372 112737 000134 023524 MOVVB   #' \,9$        ;; TYPE A BACK SLASH
4449 023400 104401 023524          TYPE   9$
4450 023404 005016          CLR    (SP)            ;; CLEAR THE RUBOUT KEY
4451 023406 122713 000025 7$:  CMPB    #25,(R3)      ;; IS CHARACTER A CTRL U?
4452 023412 001003          BNE    8$              ;; BR IF NO
4453 023414 104401 023557          TYPE   ,SCNTLU        ;; TYPE A CONTROL "U"
4454 023420 000726          BR     1$              ;; GO START OVER
4455 023422 122713 000022 8$:  CMPB    #22,(R3)      ;; IS CHARACTER A "r"?
4456 023426 001011          BNE    3$              ;; BRANCH IF NO
4457 023430 105013          CLRB   (R3)            ;; CLEAR THE CHARACTER
4458 023432 104401 001215          TYPE   ,SCRLF        ;; TYPE A "CR" & "LF"
4459 023436 104401 023526          TYPE   ,TTYIN        ;; TYPE THE INPUT STRING
4460 023442 000717          BR     2$              ;; GO PICKUP ANOTHER CHARACTER
4461 023444 104401 001214 4$:  TYPE   ,SQUES        ;; TYPE A '?'
4462 023450 000712          BR     1$              ;; CLEAR THE BUFFER AND LOOP
4463 023452 111337 023524 3$:  MOVVB  (R3),9$        ;; ECHO THE CHARACTER
4464 023456 104401 023524          TYPE   9$
4465 023462 122723 000015          CMPB   #15,(R3)+      ;; CHECK FOR RETURN
4466 023466 001305          BNE    2$              ;; LOOP IF NOT RETURN
4467 023470 105053 177777          CLRB   -(R3)          ;; CLEAR RETURN (THE 15)
4468 023474 104401 001216          TYPE   ,SLF          ;; TYPE A LINE FEED
4469 023500 005726          TST    (SP)+          ;; CLEAN RUBOUT KEY FROM THE STACK

```



```

4470 023502 012603          MOV      (SP)+,R3          ;;RESTORE R3
4471 023504 011646          MOV      (SP)-(SP)        ;;ADJUST THE STACK AND PUT ADDRESS OF THE
4472 023506 016666 000004 000002  MOV      4(SP),2(SP)      ;;FIRST ASCII CHARACTER ON IT
4473 023514 012766 023526 000004  MOV      $TTYIN,4(SP)
4474 023522 000002          RTI                      ;;RETURN
4475 023524 000          9$: .BYTE 0          ;;STORAGE FOR ASCII CHAR. TO TYPE
4476 023525 000          .BYTE 0          ;;TERMINATOR
4477 023526 000024          $TTYIN: .BLKB 20.      ;;RESERVE 20. BYTES FOR TTY INPUT
4478 023552 041536 005015 000  $CNTLC: .ASCIZ /↑C/<15><12> ;;CONTROL "C"
4479 023557 136 006525 000012  $CNTLU: .ASCIZ /↑U/<15><12> ;;CONTROL "U"
4480 023564 043536 005015 000  $CNTLG: .ASCIZ /↑G/<15><12> ;;CONTROL "G"
4481 023571 015 051412 051127  $MSWR: .ASCIZ <15><12>/SWR = /
4482 023576 036440 000040          $MNEW: .ASCIZ / NEW = /
4483 023602 020040 042516 020127  .EVEN
4484 023610 020075 000
4485 023614
4486
4487 .SBTTL SCOPE HANDLER ROUTINE
4488
4489 *****
4490 ;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
4491 ;AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
4492 ;AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
4493 ;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4494 ;SW14=1 LOOP ON TEST
4495 ;SW11=1 INHIBIT ITERATIONS
4496 ;SW09=1 LOOP ON ERROR
4497 ;CALL
4498 ;* SCOPE ;;SCOPE=IOT
4499
4500 $SCOPE:
4501 023614 104407          1$: CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
4502 023616 032777 040000 155314  BIT      #BIT14,$SWR      ;;LOOP ON PRESENT TEST?
4503 023624 001076          BNE      $OVER          ;;YES IF SW14=1
4504          ;*****START OF CODE FOR THE XOR TESTER*****
4505 023626 000416          $XTSTR: BR      6$      ;;IF RUNNING ON THE "XOR" TESTER CHANGE
4506          THIS INSTRUCTION TO A "NOP" (NOP=240)
4507 023630 013746 000004          MOV      2$ERRVEC, -(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
4508 023634 012737 023654 000004          MOV      $5$,2$ERRVEC   ;;SET FOR TIMEOUT
4509 023642 005737 177060          TST      2$177060      ;;TIME OUT ON XOR?
4510 023646 012637 000004          MOV      (SP)+,2$ERRVEC ;;RESTORE THE ERROR VECTOR
4511 023652 000450          BR      $SVLAD        ;;GO TO THE NEXT TEST
4512 023654 022626          5$: CMP      (SP)+,(SP)+  ;;CLEAR THE STACK AFTER A TIME OUT
4513 023656 012637 000004          MOV      (SP)+,2$ERRVEC ;;RESTORE THE ERROR VECTOR
4514 023662 000413          BR      7$           ;;LOOP ON THE PRESENT TEST
4515 023664          6$; *****END OF CODE FOR THE XOR TESTER*****
4516 023664 105737 001103          2$: TSTB   $ERFLG      ;;HAS AN ERROR OCCURRED?
4517 023670 001421          BEQ      3$           ;;BR IF NO
4518 023672 123737 001115 001103          CMPB   $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
4519 023700 101015          BHI     3$           ;;BR IF NO
4520 023702 032777 001000 155230          BIT    #BIT09,$SWR    ;;LOOP ON ERROR?
4521 023710 001404          BEQ    4$           ;;BR IF NO
4522 023712 013737 001110 001106          7$: MOV    $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
4523 023720 000440          BR    $OVER
4524 023722 105037 001103          4$: CLRB   $ERFLG      ;;ZERO THE ERROR FLAG
4525 023726 005037 001204          CLR    $TIMES        ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE

```

```

4526 023732 000412          BR      1$          ;; ESCAPE TO THE NEXT TEST
4527 023734 032777 004000 155176 3$: BIT  #BIT11,DSWR  ;; INHIBIT ITERATIONS?
4528 023742 001006          BNE   1$          ;; BR IF YES
4529 023744 005237 001104          INC   $ICNT       ;; INCREMENT ITERATION COUNT
4530 023750 023737 001204 001104  CMP   $TIMES,$ICNT ;; CHECK THE NUMBER OF ITERATIONS MADE
4531 023756 002021          BGE   $OVER      ;; BR IF MORE ITERATION REQUIRED
4532 023760 012737 000001 001104 1$: MOV  #1,$ICNT   ;; REINITIALIZE THE ITERATION COUNTER
4533 023766 013737 024036 001204  MOV  $MXCNT,$TIMES ;; SET NUMBER OF ITERATIONS TO DO
4534 023774 105237 001102  $SVLAD: INCB $STNM   ;; COUNT TEST NUMBERS
4535 024000 011637 001106          MOV  (SP),$LPADR  ;; SAVE SCOPE LOOP ADDRESS
4536 024004 011637 001110          MOV  (SP),$LPERR  ;; SAVE ERROR LOOP ADDRESS
4537 024010 005037 001206          CLR  $ESCAPE     ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
4538 024014 112737 000001 001115  MOVB #1,$ERMAX   ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
4539 024022 013777 001102 155112 $OVER: MOV $STNM,$DISPLAY ;; DISPLAY TEST NUMBER
4540 024030 013716 001106          MOV  $LPADR,(SP) ;; FUDGE RETURN ADDRESS
4541 024034 000002          RTI          ;; FIXES PS
4542 024036 000001  $MXCNT: 1          ;; MAX. NUMBER OF ITERATIONS

```

.SBTTL SAVE AND RESTORE RO-R5 ROUTINES

```

4543
4544
4545
4546
4547
4548
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4560

```

```

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

```

```

4561 024040          $SAVREG: MOV  R0,-(SP)    ;; PUSH R0 ON STACK
4562 024040 010046          MOV  R1,-(SP)    ;; PUSH R1 ON STACK
4563 024042 010146          MOV  R2,-(SP)    ;; PUSH R2 ON STACK
4564 024044 010246          MOV  R3,-(SP)    ;; PUSH R3 ON STACK
4565 024046 010346          MOV  R4,-(SP)    ;; PUSH R4 ON STACK
4566 024050 010446          MOV  R5,-(SP)    ;; PUSH R5 ON STACK
4567 024052 010546          MOV  22(SP),-(SP) ;; SAVE PS OF MAIN FLOW
4568 024054 016646 000022  MOV  22(SP),-(SP) ;; SAVE PC OF MAIN FLOW
4569 024060 016646 000022  MOV  22(SP),-(SP) ;; SAVE PS OF CALL
4570 024064 016646 000022  MOV  22(SP),-(SP) ;; SAVE PC OF CALL
4571 024070 016646 000022          RTI
4572 024074 000002

```

```

4573
4574
4575
4576
4577
4578
4579
4580
4581

```

```

*RESTORE RO-R5
*CALL:
* RESREG
$RESREG:
MOV  (SP)+,22(SP) ;; RESTORE PC OF CALL
MOV  (SP)+,22(SP) ;; RESTORE PS OF CALL
MOV  (SP)+,22(SP) ;; RESTORE PC OF MAIN FLOW
MOV  (SP)+,22(SP) ;; RESTORE PS OF MAIN FLOW

```

```

4582 024116 012605      MOV      (SP)+,R5      ;; POP STACK INTO R5
4583 024120 012604      MOV      (SP)+,R4      ;; POP STACK INTO R4
4584 024122 012603      MOV      (SP)+,R3      ;; POP STACK INTO R3
4585 024124 012602      MOV      (SP)+,R2      ;; POP STACK INTO R2
4586 024126 012601      MOV      (SP)+,R1      ;; POP STACK INTO R1
4587 024130 012600      MOV      (SP)+,R0      ;; POP STACK INTO R0
4588 024132 000002      RTI

```

.SBTTL TRAP DECODER

```

4591
4592 ;*****
4593 ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
4594 ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
4595 ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
4596 ;*GO TO THAT ROUTINE.

```

```

4598 024134 010046      $TRAP:  MOV      RO, -(SP)      ;; SAVE RO
4599 024136 016600 000002  MOV      2(SP),RO      ;; GET TRAP ADDRESS
4600 024142 005740      TST      -(RO)         ;; BACKUP BY 2
4601 024144 111000      MOV      (RO),RO      ;; GET RIGHT BYTE OF TRAP
4602 024146 006300      ASL      RO           ;; POSITION FOR INDEXING
4603 024150 016000 024170  MOV      $TRPAD(RO),RO  ;; INDEX TO TABLE
4604 024154 000200      RTS      RO           ;; GO TO ROUTINE

```

;; THIS IS USE TO HANDLE THE "GETPRI" MACRO

```

4608
4609 024156 011646      $TRAP2: MOV      (SP), -(SP)    ;; MOVE THE PC DOWN
4610 024160 016666 000004 000C32  MOV      4(SP), 2(SP)    ;; MOVE THE PSW DOWN
4611 024166 000002      RTI                    ;; RESTORE THE PSW

```

.SBTTL TRAP TABLE

```

4613 ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
4614 ;*BY THE "TRAP" INSTRUCTION.

```

```

4615 ;
4616 ; ROUTINE
4617 ; -----
4618 ;
4619 $TRPAD: .WORD  $TRAP2
4620          $TYPE  ;; CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
4621          $TYPOC ;; CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
4622          $TYPOS ;; CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
4623          $TYPON ;; CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
4624          $TYPDS ;; CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
4625
4626          $GTSWR ;; CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING
4627
4628          $CKSWR ;; CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
4629          $RDCHR ;; CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
4630          $RDLIN ;; CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
4631          $SAVREG ;; CALL=SAVREG   TRAP+12(104412) SAVE RO-R5 ROUTINE
4632          $RESREG ;; CALL=RESREG   TRAP+13(104413) RESTORE RO-R5 ROUTINE
4633
4634
4635
4636
4637

```

.SBTTL SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE

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

F10

MD-11-DZRMF-A RMO3 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 90  
SINGLE LENGTH BINARY TO DECIMAL ASCII ROUTINE

SEQ 0122

```

4638      ;*THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
4639      ;*UNSIGNED DECIMAL ASCII NUMBER.
4640      ;*CALL
4641      ;*      MOV      NUMBER, -(SP)      ;: PUT BINARY NUMBER ON THE STACK
4642      ;*      JSR      PC, @$$S820      ;: CALL
4643      ;*      RETURN     ;: ADDRESS OF THE 1ST ASCII CHAR. IS ON THE STACK
4644
4645
4646 024220 016637 000002 024250 $$S820: MOV      2(SP), 1$      ;: SAVE BINARY NUMBER
4647 024226 012746 024250      MOV      #1$, -(SP)      ;: SET POINTER
4648 024232 004737 024254      JSR      PC, @$$SDB20      ;: CALL DOUBLE LENGTH CONVERT
4649 024236 062716 000005      ADD      #5, (SP)      ;: ONLY ALLOW FIVE CHARACTERS
4650 024242 012666 000002      MOV      (SP)+, 2(SP)      ;: PICKUP POINTER
4651 024246 000207      RTS      PC      ;: RETURN
4652 024250 000000 000000 1$:      .WORD      0, 0
4653
4654      .SBTTL  DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
4655
4656      ;*****
4657      ;*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
4658      ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
4659      ;*POSITIVE.
4660      ;*CALL
4661      ;*      MOV      #PNTR, -(SP)      ;: POINTER TO LOW WORD OF BINARY NUMBER
4662      ;*      JSR      PC, @$$SDB20      ;:
4663      ;*      RETURN     ;: THE FIRST ADDRESS OF ASCII
4664      ;: IS ON THE STACK
4665
4666
4667 024254 104412      $SDB20: SAVREG      ;: SAVE REGISTERS
4668 024256 016602 000002      MOV      2(SP), R2      ;: PICKUP THE DATA POINTER
4669 024262 012700 024434      MOV      $$SDECLV, R0      ;: GET ADDRESS OF "$DECLV" STRING
4670 024266 010066 000002      MOV      R0, 2(SP)      ;: PUT ADDRESS OF ASCII STRING ON STACK
4671 024272 012201      MOV      (R2)+, R1      ;: PICKUP THE BINARY NUMBER
4672 024274 012202      MOV      (R2)+, R2
4673 024276 012737 000012 024352      MOV      #10, 4$      ;: SET UP TO DO 10 CONVERSIONS
4674 024304 012704 024364      MOV      $STNPNR, R4      ;: ADDRESS OF TEN POWER
4675 024310 012705 024365      MOV      $STNPNR+2, R5
4676 024314 005003 1$:      CLR      R3      ;: CLEAR PARTIAL
4677 024316 161401 2$:      SUB      (R4), R1      ;: SUBTRACT TEN POWER
4678 024320 005602      SBC      R2
4679 024322 161502      SUB      (R5), R2
4680 024324 002402      BLT      3$      ;: BR IF TEN POWER TOO LARGE
4681 024326 005203      INC      R3      ;: ADD 1 TO PARTIAL
4682 024330 000772      BR      2$      ;: LOOP
4683 024332 062401 3$:      ADD      (R4)+, R1      ;: RESTORE SUBTRACTED VALUE
4684 024334 005502      ADC      R2
4685 024336 062402      ADD      (R4)+, R2
4686 024340 022525      CMP      (R5)+, (R5)+      ;: MOVE TO NEXT TEN POWER
4687 024342 052703 000060      BIS      #'0, R3      ;: CHANGE PARTIAL TO ASCII
4688 024346 110320      MOV      R3, (R0)+      ;: SAVE IT
4689 024350 005327      DEC      (PC)+      ;: DONE?
4690 024352 000000 4$:      .WORD      0
4691 024354 001357      BNE      1$      ;: BR IF NO
4692 024356 105020      CLRB      (R0)+      ;: TERMINATOR
4693 024360 104413      RESREG      ;: RESTORE REGISTERS

```

G10

MD-11-DZRMF-A RMO3 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 91  
DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

SEQ 0123

4694	024362	000207		RTS	PC	;;RETURN
4695	024364	145000		\$TNPWR: 145000		;;1.OE09
4696	024366	035632		35632		
4697	024370	160400		160400		;;1.OE08
4698	024372	002765		2765		
4699	024374	113200		113200		;;1.OE07
4700	024376	000230		230		
4701	024400	041100		041100		;;1.OE06
4702	024402	000017		17		
4703	024404	103240		103240		;;1.OE05
4704	024406	000001		1		
4705	024410	023420		23420		;;1.OE04
4706	024412	000000		0		
4707	024414	001750		1750		;;1.OE03
4708	024416	000000		0		
4709	024420	000144		144		;;1.OE02
4710	024422	000000		0		
4711	024424	000012		12		;;1.OE01
4712	024426	000000		0		
4713	024430	000001		1		;;1.OE00
4714	024432	000000		0		
4715	024434	000014		\$DECVL: .BLKB 12.		;;RESERVE STORAGE FOR ASCIZ STRING
4716						
4717				.SBTTL TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS		
4718						
4719				;;*****		
4720				;;*THIS ROUTINE IS USED TO TYPE AN ASCIZ NUMBER SUPPRESSING THE		
4721				;;*LEADING NUMBERS.		
4722				;;*CALL		
4723				;;* MOV #NUMADR, -(SP) ; ;FIRST ADDRESS OF ASCIZ STRING		
4724				;;* JSR PC, @#SSUPRS		
4725						
4726						
4727	024450	010046		\$SUPRS: MOV RO, -(SP)		;;SAVE RO
4728	024452	016600	000004	MOV 4(SP), RO		;;PICKUP THE POINTER
4729	024456	105710		1\$: TSTB (RO)		;;TERMINATEOR?
4730	024460	001403		BEQ 2\$		;;BR IF YES
4731	024462	122720	000060	CMPB #'0, (RO)+		;;IS THIS AN ASCII "0" ?
4732	024466	001773		BEQ 1\$		;;BR IF YES
4733	024470	005300		2\$: DEC RO		;;BACKUP BY "1"
4734	024472	010037	024500	MOV RO, 3\$		;;SAVE FOR TYPING
4735	024476	104401		TYPE		;;GO TYPE
4736	024500	000000		3\$: .WORD 0		;;ASCIZ POINTER GOES HERE
4737	024502	012600		MOV (SP)+, RO		;;RESTORE RO
4738	024504	012616		MOV (SP)+, (SP)		;;RESTORE THE STACK
4739	024506	000207		RTS	PC	;;RETURN
4740						
4741				.SBTTL RANDOM NUMBER GENERATOR ROUT E		
4742						
4743				;;*****		
4744				;;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR		
4745				;;*WITH A RANGE OF 0 TO 2(+33)-1.		
4746				;;*CALL:		
4747				;;* JSR PC, \$RAND ; ;CALL THE ROUTINE		
4748				;;* RETURN ; ;RETURN HERE THE RANDOM		
4749				;;* ; ;NUMBER WILL BE IN		

# H10

MD-11-DZRMF-A RMO3 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 92  
RANDOM NUMBER GENERATOR ROUTINE

SEQ 0124

```

4750
4751
4752 024510
4753 024510 010046
4754 024512 010146
4755 024514 010246
4756 024516 013700 024610
4757 024522 013701 024606
4758 024526 012702 177771
4759 024532 006300
4760 024534 006101
4761 024536 005202
4762 024540 001374
4763 024542 063700 024610
4764 024546 001501
4765 024550 063701 024606
4766 024554 062700 001057
4767 024560 005501
4768 024562 062701 047401
4769 024566 010037 024610
4770 024572 010137 024606
4771 024576 012602
4772 024600 012601
4773 024602 012600
4774 024604 000207
4775 024606 176543
4776 024610 123456
4777
4778
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803 024612
4804 024612 104400
4805 024614 042716 000017
    
```

```

;*                                     ;;SHINUM,SLONUM
SRAND:
MOV RO,-(SP)                          ;; PUSH RO ON STACK
MOV R1,-(SP)                          ;; PUSH R1 ON STACK
MOV R2,-(SP)                          ;; PUSH R2 ON STACK
MOV SLONUM,RO                          ;; SET RO WITH LOW
MOV SHINUM,R1                          ;; SET R1 WITH HIGH
MOV #-7,R2                             ;; SET SHIFT COUNT
1$: ASL RO                             ;; SHIFT RO LEFT AND
ROL R1                                 ;; ROTATE CARRY INTO R1 AND
INC R2                                 ;; CHECK FOR DONE
BNE 1$                                ;; CONTINUE SHIFT LOOP
ADD SLONUM,RO                          ;; ADD NUMBER TO MAKE X 129
ADC R1                                 ;; PROPAGATE CARRY
ADD SHINUM,R1                          ;; ADD NUMBER TO MAKE X 129
ADD #1057,RO                           ;; ADD LOW CONSTANT
ADC R1                                 ;; PROPAGATE CARRY
ADD #47401,R1                          ;; ADD HIGH CONSTANT
MOV RO,SLONUM                          ;; SAVE RO
MOV R1,SHINUM                          ;; SAVE R1
MOV (SP)+,R2                           ;; POP STACK INTO R2
MOV (SP)+,R1                           ;; POP STACK INTO R1
MOV (SP)+,RO                           ;; POP STACK INTO RO
RTS PC                                 ;; RETURN
SHINUM: .WORD 176543
SLONUM: .WORD 123456

.SBTTL INTEGER DIVIDE ROUTINE
*****
*THIS ROUTINE WILL DIVIDE A 32-BIT TWO'S COMPLEMENT INTEGER
*DIVIDEND BY A 16-BIT TWO'S COMPLEMENT INTEGER DIVISOR GIVING
*A 16-BIT TWO'S COMPLEMENT INTEGER QUOTIENT AND A 16-BIT REMAINDER.
*DIVISION WILL BE PERFORMED SO THAT THE REMAINDER IS OF THE
*SAME SIGN AS THE DIVIDEND.
*CALL:
*   MOV LOW DIVIDEND,-(SP) ;; THE HIGH DIVIDEND MUST BE < 1/2
*   MOV HIGH DIVIDEND,-(SP); AS LARGE AS THE DIVISOR
*   MOV DIVISOR,-(SP)
*   JSR PC,$DIV
*   RETURN ;; QUOTIENT & REMAINDER ARE ON THE STACK
*   "V"=0 IMPLIES NO ERROR
*   "V"=1 IMPLIES ERROR OCCURRED
*   "C"=0 DIVIDE OVERFLOW OCCURRED
*   "C"=1 ATTEMPTED TO DIVIDE BY ZERO

*   STACK NO ERROR OVERFLOW DIVIDE BY ZERO
*   -----
*   TOP REMAINDER ALL ZEROS ALL ONES
*   +2 QUOTIENT ALL ZEROS ALL ONES
SDIV:
TRAP ;; PUSH OLD PSW AND PC ON STACK
BIC #17,(SP) ;; STRIP AWAY CONDITION CODES
    
```

4806	024620	010046			MOV	RO,-(SP)	::	PUSH RO ON STACK
4807	024622	010146			MOV	R1,-(SP)	::	PUSH R1 ON STACK
4808	024624	010246			MOV	R2,-(SP)	::	PUSH R2 ON STACK
4809	024626	010346			MOV	R3,-(SP)	::	PUSH R3 ON STACK
4810	024630	005046			CLR	-(SP)	::	SAVE A PLACE FOR SIGNS
4811	024632	012746	000021		MOV	#17,-(SP)	::	SETUP THE ITERATION COUNTER
4812	024636	016601	000024		MOV	24(SP),R1	::	PICKUP THE DIVIDEND
4813	024642	016600	000022		MOV	22(SP),RO		
4814	024646	100005			BPL	1\$	::	CHECK THE SIGN
4815	024650	105366	000003		DECB	3(SP)	::	KEEP TRACK OF THE SIGN
4816	024654	005400			NEG	RO	::	AND NEGATE THE ORIGINAL
4817	024656	005401			NEG	R1	::	NUMBER
4818	024660	005600			SBC	RO		
4819	024662	016602	000020	1\$:	MOV	20(SP),R2	::	PICKUP THE DIVISOR
4820	024666	002407			BLT	2\$	::	CHECK THE SIGN
4821	024670	003011			BGT	3\$	::	DIVISOR OF 0 IS A NO-NO
4822	024672	052766	000003	000014	BIS	#3,14(SP)	::	SET "V" & "C"
4823	024700	012700	177777		MOV	#-1,RO	::	SET REMAINDER TO ALL ONES
4824	024704	000424			BR	7\$	::	EXIT
4825	024706	005266	000002	2\$:	INC	2(SP)	::	KEEP TRACK OF DIVISORS SIGN
4826	024712	000401			BR	4\$		
4827	024714	005402		3\$:	NEG	R2	::	NEGATE THE ORIGINAL NUMBER
4828	024716	000241		4\$:	CLC		::	CLEAR "C"
4829	024720	000405			BR	6\$	::	START FORMING QUOTIENT
4830	024722	006100		5\$:	ROL	RO	::	POSITION MSB'S
4831	024724	010003			MOV	RO,R3	::	COPY
4832	024726	060203			ADD	R2,R3	::	COMPARE DIVIDEND & DIVISOR
4833	024730	103001			BCC	6\$	::	BR IF DIVIDEND > DIVISOR
4834	024732	010300			MOV	R3,RO	::	REMAINDER AFTER THIS LOOP
4835	024734	006101		6\$:	ROL	R1	::	QUOTIENT BIT ENTERS HERE
4836	024736	005316			DEC	(SP)	::	DONE?
4837	024740	001370			BNE	5\$	::	BR IF NO
4838	024742	005701			TST	R1	::	OVERFLOW?
4839	024744	100005			BPL	8\$	::	BR IF NO
4840	024746	052766	000002	000014	BIS	#2,14(SP)	::	SET "V" IN RETURN STATUS WORD
4841	024754	005000			CLR	RO	::	SET REMAINDER TO ALL ZEROS
4842	024756	010001		7\$:	MOV	RO,R1	::	COPY REMAINDER INTO QUOTIENT
4843	024760	005726		8\$:	TST	(SP)+	::	CLEAR COUNTER FROM STACK
4844	024762	005716			TST	(SP)	::	REMAINDER SIGN CORRECTION NEEDED?
4845	024764	002004			BGE	9\$	::	BR IF NO
4846	024766	005400			NEG	RO	::	NEGATE REMAINDER
4847	024770	105066	000001		CLRB	1(SP)	::	CLEAR SIGN
4848	024774	005316			DEC	(SP)	::	BUT DON'T FORGET QUOTIENT
4849	024776	005726		9\$:	TST	(SP)+	::	QUOTIENT SIGN CORRECTION NEEDED?
4850	025000	001401			BEQ	10\$	::	BR IF NO
4851	025002	005401			NEG	R1	::	NEGATE QUOTIENT
4852	025004	010166	000020	10\$:	MOV	R1,20(SP)	::	RETURN QUOTIENT AND
4853	025010	010066	000016		MOV	RO,16(SP)	::	REMAINDER TO USER
4854	025014	012603			MOV	(SP)+,R3	::	POP STACK INTO R3
4855	025016	012602			MOV	(SP)+,R2	::	POP STACK INTO R2
4856	025020	012601			MOV	(SP)+,R1	::	POP STACK INTO R1
4857	025022	012600			MOV	(SP)+,RO	::	POP STACK INTO RO
4858	025024	012666	000002		MOV	(SP)+,2(SP)	::	SETUP TO RETURN CONDITION CODES
4859	025030	000002			RTI		::	RETURN

.SBTTL \*\*\* PROGRAM SUBROUTINES \*\*\*

```

4862
4863 ;SET "LPTAVL" TO THE PROPER STATE.
4864 ; LPTAVL = 0 IF NO LINE PRINTER AVAILABLE
4865 ; LPTAVL = 1 IF LINE PRINTER IS AVAILABLE
4866 ;CALL
4867 ;       JSR      PC,@#LP.AVL
4868 ;       RETURN
4869
4870 025032 005037 001230          LP.AVL: CLR      @#LPTAVL      ;START WITH NO PRINTER AVAILABE
4871 025036 012737 025052 000004  MOV      #1,@#ERRVEC    ;SETUP THE TIMEOUT VECTOR
4872 025044 005037 000006      CLR      @#ERRVEC+2
4873 025050 005777 154344      TST      @#LPS          ;IS THERE A LINE PRINTER?
4874 025054 005237 001230      INC      @#LPTAVL      ;YES--SET AVAILABLE SWITCH
4875 025060 000401              BR       2$
4876 025062 022626              1$:  CMP      (SP)+,(SP)+    ;NO--POP STACK
4877 025064 012737 000006 000004 2$:  MOV      @#ERRVEC+2,@#ERRVEC ;RESTORE TIMEOUT VECTOR
4878 025072 000207              RTS      PC              ;RETURN
4879
4880 ;THIS ROUTINE WILL DETERMINE IF THERE IS A CLOCK ON THE SYSTEM
4881 ;AND IF THERE IS IT WILL SETUP THE VECTOR AND START THE CLOCK
4882 ;"CLKSTA" WILL INDICATE THE CLOCK TYPE
4883 ; 0= NO CLOCK
4884 ;+1= KW11-P
4885 ;-1= KW11-L
4886 ;THIS ROUTINE WILL ALSO SETUP "TICKMS" (TIME
4887 ;PER CLOCK TICK IN MILLISECONDS) AND "TICKUS"
4888 ;(TIME PER CLOCK TICK IN MICROSECONDS) AS
4889 ;PER SW00.
4890 ;SW00=0 -- 60HZ
4891 ;SW00=1 -- 50HZ
4892 ;CALL
4893 ;       JSR      PC,@#ST.CLK
4894 ;       RETURN
4895
4896 025074 010146              ST.CLK: MOV      R1,-(SP)    ;SAVE R1
4897 025076 012701 000006      MOV      @#ERRVEC+2,R1  ;SAVE AND SETUP TIMEOUT VECTOR
4898 025102 011146              MOV      (R1),-(SP)
4899 025104 005011              CLR      (R1)          ;LEVEL 0
4900 025106 014146              MOV      -(R1),-(SP)
4901 025110 012711 025140      MOV      #1,(R1)      ;GO TO 1$ ON TIMEOUT
4902 025114 005037 001244      CLR      CLKSTA      ;SET CLOCK STATUS TO NO CLOCK
4903 025120 005777 154254      TST      @PKCS       ;IS THERE A KW11-P?
4904 025124 012737 000001 001244  MOV      #1,CLKSTA    ;YES--SET STATUS TO KW11-P
4905 025132 004737 025242      JSR      PC,ST.PCLK  ;START THE KW11-P
4906 025136 000414              BR       3$           ;GO TO EXIT
4907 025140 022626              1$:  CMP      (SP)+,(SP)+  ;CLEAN UP THE STACK
4908 025142 012711 025166      MOV      #2,(R1)     ;IF TIMEOUT GO TO 2$
4909 025146 005777 154240      TST      @LKS        ;IS THERE A KW11-L?
4910 025152 012737 177777 001244  MOV      #-1,CLKSTA  ;YES-- SET STATUS TO KW11-L
4911 025160 004737 025304      JSR      PC,ST.LCLK  ;START THE KW11-L
4912 025164 000401              BR       3$           ;EXIT
4913 025166 022626              2$:  CMP      (SP)+,(SP)+  ;CLEAN UP THE STACK
4914 025170 012621 001244      3$:  MOV      (SP)+,(R1)+ ;RESTORE THE TIMEOUT VECTOR
4915 025172 012621              MOV      (SP)+,(R1)+
4916 025174 012601              MOV      (SP)+,R1    ;RESTORE R1
4917 025176 032737 000100 001220  BIT      @SW06,@#C.SWR ;50HZ OR 60HZ?

```



```

4918 025204 001407          BEQ      4$          ;BRANCH IF 60
4919 025206 012737 000020 001246    MOV      #20,2,TICKMS ;SETUP TIME PER
4920 025214 012737 047040 001250    MOV      #20000.,2,TICKUS ;TICK FOR 50HZ
4921 025222 000406          BR       5$
4922 025224 012737 000016 001246    4$: MOV    #16,2,TICKMS ;SETUP TIME PER
4923 025232 012737 040432 001250    MOV    #16666.,2,TICKUS ;TICK FOR 60HZ
4924 025240 000207          5$: RTS      PC          ;RETURN
4925
4926 025242          ST.PCLK:
4927 025242 032737 000040 001220    BIT      #SW05,2,C.SWR ;ALLOW SOFTWARE TIMEOUTS?
4928 025250 001014          BNE      1$          ;NO--BRANCH
4929 025252 012777 025340 154114    MOV      #SRVCLK,2PKV ;SETUP THE KW11-P VECTOR
4930 025260 012777 000300 154110    MOV      #300,2PKV+2
4931 025266 012777 000001 154106    MOV      #1,2PKB ;COUNT ONE TICK
4932 025274 012777 000115 154076    MOV      #115,2PKCS ;"INT.EN." COUNT DOWN" "MODE 1 (REPEAT)",
4933                                     ;"LINE FREQ", AND "RUN"
4934 025302 000207          1$: RTS      PC          ;RETURN
4935
4936 025304          ST.LCLK:
4937 025304 032737 000040 001220    BIT      #SW05,2,C.SWR ;ALLOW SOFTWARE TIMEOUTS?
4938 025312 001011          BNE      1$          ;NO--BRANCH
4939 025314 012777 025340 154064    MOV      #SRVCLK,2LKV ;SETUP THE KW11-L VECTOR
4940 025322 012777 000300 154060    MOV      #300,2LKV+2
4941 025330 012777 000100 154054    MOV      #100,2LKS ;START THE KW11-L
4942 025336 000207          1$: RTS      PC          ;RETURN
4943
4944 025340 013746 001246          SRVCLK: MOV    2,TICKMS,-(SP) ;TIME PER TICK IN MILLISECONDS
4945 025344 004737 041704          JSR      PC,2,RPTMR ;COUNT THE ELAPSED TIME
4946 025350 000002          RTI      ;RETURN AFTER INTERRUPT
4947
4948                                     ; THIS ROUTINE SETS UP DEFAULT PARAMETER VALUES WHEN THE PROGRAM IS
4949                                     ; STARTED OR WHEN THE VALUE OF BIT00 IN 'C.SWR' IS CHANGED.
4950                                     ; CALL
4951                                     ; JSR      PC,LODFLT
4952                                     ; RETURN
4953
4954 025352          LODFLT:
4955 025352 010046          MOV      R0,-(SP) ;: PUSH R0 ON STACK
4956 025354 010146          MOV      R1,-(SP) ;: PUSH R1 ON STACK
4957 025356 010246          MOV      R2,-(SP) ;: PUSH R2 ON STACK
4958 025360 010346          MOV      R3,-(SP) ;: PUSH R3 ON STACK
4959 025362 012737 176777 001234    MOV      #176777,TSTNMS ;SELECT TESTS 0-10, 12-17
4960 025370 012737 000003 001236    MOV      #3,TSTNMS+2 ;SELECT TESTS 20 & 21
4961 025376 012700 001664          MOV      #DFLT,R0 ;DEFAULT PARAMETERS POINTER
4962 025402 012701 002330          MOV      #PRMO,R1 ;TABLE POINTER
4963 025406 010102          MOV      R1,R2 ;STOP ADDRESS
4964 025410 012021          1$: MOV    (R0)+,(R1)+ ;MOVE DEFAULT PARAMETERS INTO
4965 025412 020002          CMP     R0,R2 ;RUN TIME TABLES ** DONE?
4966 025414 103775          BLO     1$          ;NO--BRANCH
4967 025416 012700 003504          MOV      #PAT8,R0 ;PATO DEFAULTS TO PATTERN 8
4968 025422 012701 003104          MOV      #PATO,R1
4969 025426 012021          2$: MOV    (R0)+,(R1)+
4970 025430 020027 003544          CMP     R0,#PAT9
4971 025434 103774          BLO     2$
4972 025436 032737 000001 001220    BIT      #BIT00,C.SWR ;: 16 BIT MODE ?
4973 025444 001012          BNE     3$          ;BR IF 18 BIT MODE

```

```

4974 025446 012737 000037 001630      MOV      #31.,PRMLMT+22 ;SET 'FS' LIMIT TO 31.
4975 025454 012737 000037 001632      MOV      #31.,PRMLMT+24 ;SET 'LS' LIMIT TO 31.
4976 025462 012737 160000 001352      MOV      #-(<256.*32.>),TRCKWC ;WORD COUNT FOR A 16 BIT TRACK
4977 025470 000411                BR        4$ ;CONTINUE
4978 025472 012737 000035 001630 3$:  MOV      #29.,PRMLMT+22 ;SET 'FS' LIMIT TO 29.
4979 025500 012737 000035 001632      MOV      #29.,PRMLMT+24 ;SET 'LS' LIMIT TO 29.
4980 025506 012737 161000 001352      MOV      #-(<256.*30.>),TRCKWC ;WORD COUNT FOR AN 18 BIT TRACK
4981 025514 012701 001536                4$:  MOV      #PRMPT,R1 ;ADDRESS OF PARAMETER POINTER TABLE
4982 025520 005711                5$:  TST      (R1) ;END OF THE TABLE ?
4983 025522 001425                BEQ      8$ ;BR IF END
4984 025524 032731 002000                BIT      #BIT10,2(R1)+ ;'LS' SELECTED ?
4985 025530 001773                BEQ      5$ ;BR IF NOT
4986 025532 016102 177776                MOV      -2(R1),R2 ;PARAMETER TABLE ADDRESS
4987 025536 011246                MOV      (R2),-(SP) ;PARAMETER ALLOCATION BITS
4988 025540 012703 000013                MOV      #11.,R3 ;NUMBER OF PARAMETERS (MAXIMUM) BEFORE 'LS'
4989 025544 006216                6$:  ASR      (SP) ;COUNT THE PARAMETER
4990 025546 103002                BCC      7$ ;BR IF NOT USED
4991 025550 062702 000002                ADD      #2,R2 ;INCREMENT THE PARAMETER TABLE ADDRESS
4992 025554 005303                7$:  DEC      R3 ;COUNT THE PARAMETER
4993 025556 001372                BNE      6$ ;BR IF NOT THERE YET
4994 025560 005726                TST      (SP)+ ;CORRECT THE STACK POINTER
4995 025562 021237 001630                CMP      (R2),PRMLMT+22 ;IS 'LS' TOO LARGE FOR THE MODE SELECTED ?
4996 025566 101754                BLOS    5$ ;BR IF NOT
4997 025570 013712 001630                MOV      PRMLMT+22,(R2) ;RESET VALUE FOR MODE USED
4998 025574 000751                BR        5$ ;CONTINUE
4999 025576                8$:
5000 025576 012603                MOV      (SP)+,R3 ;: POP STACK INTO R3
5001 025600 012602                MOV      (SP)+,R2 ;: POP STACK INTO R2
5002 025602 012601                MOV      (SP)+,R1 ;: POP STACK INTO R1
5003 025604 012600                MOV      (SP)+,R0 ;: POP STACK INTO R0
5004 025606 000207                RTS      PC ;RETURN
5005
5006 ;THIS ROUTINE FILLS THE PARAMETER TABLE THE CURRENT TEST.
5007 ;CALL
5008 ;
5009 ;
5010 ;
5011 ;
5012 025610                LODPRM:
5013 025610 010146                MOV      R1,-(SP) ;: PUSH R1 ON STACK
5014 025612 010246                MOV      R2,-(SP) ;: PUSH R2 ON STACK
5015 025614 010346                MOV      R3,-(SP) ;: PUSH R3 ON STACK
5016 025616 010446                MOV      R4,-(SP) ;: PUSH R4 ON STACK
5017 025620 005004                CLR      R4 ;CLEAR R4
5018 025622 113704 001102                MOV      #STSTNM,R4 ;GET THE TEST NUMBER
5019 025626 006304                ASL      R4 ;SETUP TO ADDRESS WORDS
5020 025630 016401 001536                MOV      PRMPT(R4),R1 ;GET THE TEST'S PARAMETER TABLE ADDRESS
5021 025634 012702 001504                MOV      #PRM,R2 ;PARAMETER EXECUTION TABLE
5022 025640 005003                CLR      R3 ;R3 IS USED AS A COUNTER
5023 025642 013704 001254                MOV      CHKDRV,R4 ;DRIVE'S ADDRESS
5024 025646 012122                MOV      (R1)+,(R2)+ ;PARAMETER SPECIFIER
5025 025650 006237 001504                1$:  ASR      PRM ;THIS PARAMETER USED IN THE TEST ?
5026 025654 103002                BCC      2$ ;BR IF NOT
5027 025656 012122                MOV      (R1)+,(R2)+ ;LOAD THE VALUE
5028 025660 000401                BR        3$ ;CONTINUE
5029 025662 005C72                2$:  CLR      (R2)+ ;CLEAR THE UNUSED PARAMETER LOCATION

```

```

5030 025664 005203 000014 3$: INC R3 ;COUNT THE POSITION IN THE OUTPUT TABLE
5031 025666 020327 000014 CMP R3,#12. ;FINISHED?
5032 025672 001430 000003 BEQ 6$ ;BR IF YES
5033 025674 020327 000003 CMP R3,#3 ;DOING THE CYLINDER ADDRESSES?
5034 025700 001363 000004 BNE 1$ ;BR IF NOT
5035 025702 132764 000004 035376 BITB #BIT02,DRV TYP(R4) ;RMO3?
5036 025710 001016 000002 BNE 5$ ;IF IT IS, OVERLAY FC & LC WITH FC' & LC'
5037 025712 062703 000002 ADD #2,R3 ;COUNT THE BYPASSED PARAMETERS (FC' & LC')
5038 025716 006237 001504 ASR PRM ;SHIFT THE COUNTER
5039 025722 103002 000002 BCC 4$ ;BR IF FC' IS NOT USED
5040 025724 062701 000002 ADD #2,R1 ;MOVE THE INPUT POINTER
5041 025730 006237 001504 4$: ASR PRM ;COUNT THE PARAMETER
5042 025734 103345 000002 BCC 1$ ;BR IF LC' NOT USED
5043 025736 062701 000002 ADD #2,R1 ;MOVE THE INPUT PINTER
5044 025742 000742 000004 BR 1$ ;KEEP GOING
5045 025744 000741 000004 BR 1$ ;KEEP GOING
5046 025746 162702 000004 5$: SUB #4,R2 ;BACKUP THE OUTPUT POINTER
5047 025752 000736 000004 BR 1$ ;KEEP GOING
5048 025754 012604 000002 6$: MOV (SP)+,R4 ;POP STACK INTO R4
5049 025754 012603 000002 MOV (SP)+,R3 ;POP STACK INTO R3
5050 025760 012602 000002 MOV (SP)+,R2 ;POP STACK INTO R2
5051 025762 012601 000002 MOV (SP)+,R1 ;POP STACK INTO R1
5052 025764 000207 000002 RTS PC ;RETURN
5053
5054
5055 ;THIS ROUTINE LOADS A READ HEADER AND DATA COMMAND OR A SEEK COMMAND
5056 ;INTO DPB.B+2 AND DPB.C+2, DEPENDING ON THE STATE OF "CONTROL SWITCH"
5057 ;BIT07.
5058 ;CALL
5059 ; JSR PC,#LDCMD
5060 ; RETURN
5061
5062 025766 032737 000200 001220 LDCMD: BIT #SW07,#C.SWR ;DO EXPLICIT SEEKS?
5063 025774 001007 000173 004126 BNE 1$ ;YES--BRANCH
5064 025776 012737 000173 004126 MOV #READHD,#DPB.B+2 ;NO--SET UP FOR READ HEADER AND
5065 026004 012737 000173 004146 MOV #READHD,#DPB.C+2 ;DATA COMMAND
5066 026012 000406 000105 004126 BR 2$
5067 026014 012737 000105 004126 1$: MOV #SEEK,#DPB.B+2 ;SETUP FOR SEEK COMMAND
5068 026022 012737 000105 004146 MOV #SEEK,#DPB.C+2
5069 026030 000207 000105 004146 2$: RTS PC
5070
5071 ;THIS ROUTINE WILL CALL THE RMO3 DRIVER AND THEN WAIT ON THE FUNCTION
5072 ;TO COMPLETE. IF AN ERROR OCCURS IT IS REPORTED.
5073 ;CALL
5074 ; FILL "DPB" WITH COMMAND INFORMATION
5075 ; JSR RO,#CALL.A
5076 ; RETURN
5077
5078 026032 005037 001206 036236 CALL.A: CLR #ESCAPE ;NO ESCAPE ADDRESS
5079 026036 004037 004104 036236 JSR RO,#RMO3 ;CALL RMO3 DRIVER
5080 026042 004104 000772 004122 DPB.A
5081 026044 000772 000772 004122 BR CALL.A
5082 026046 005737 004122 1$: TST #DPB.A+16 ;DONE?
5083 026052 001775 000772 004122 BEQ 1$ ;NO--LOOP
5084 026054 100032 000772 004122 BPL 3$ ;BRANCH IF NO ERROR
5085 026056 012737 026132 001206 MOV #2$,ESCAPE ;ESCAPE TO 2$ ON ERROR

```

# N10

MD-11-DZRMF-A RMO3 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 98  
\*\*\* PROGRAM SUBROUTINES \*\*\*

SEQ 0130

5086	026064	013737	004116	001270		MOV	@#DPB.A+12,@#CYL.DS ;CYLINDER	
5087	026072	113737	004115	001274		MOVB	@#DPB.A+11,@#TRK.DS ;TRACK	
5088	026100	113737	004114	001272		MOVB	@#DPB.A+10,@#SEC.DS ;SECTOR	
5089	026106	012746	004122			MOV	@#DPB.A+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS	
5090	026112	004737	027246			JSR	PC,@#ERINDEX ;FORM DISPATCH INDEX	
5091	026116	062607				ADD	(SP)+,PC ;REPORT PROPER ERROR	
5092	026120	104041				ERROR	41 ;	
5093	026122	104042				ERROR	42 ;PARITY ERROR	
5094	026124	104043				ERROR	43 ;UNSAFE ERROR	
5095	026126	104044				ERROR	44 ;NON-I/O ERROR	
5096	026130	104045				ERROR	45 ;I/O ERROR	
5097	026132	013746	004122		2\$:	MOV	DPB.A+16,-(SP) ;STATUS WORD	
5098	026136	004737	027206			JSR	PC,LOP.CK ;SEE IF LOOP, ABORT, OR CONTINUE	
5099	026142	000200			3\$:	RTS	RO ;RETURN	
5100								
5101								
5102								
5103								
5104								
5105								
5106								
5107								
5108								
5109	026144	005037	001206			CALL.B:	CLR @#\$ESCAPE ;NO ESCAPE ADDRESS	
5110	026150	004037	036236			JSR	RO,@#RMO3 ;CALL RMO3 DRIVER	
5111	026154	004124				DPB.B		
5112	026156	000772				BR	CALL.B	
5113	026160	005737	004142		1\$:	TST	DPB.B+16 ;DONE?	
5114	026164	001775				BEQ	1\$ ;NO--BRANCH	
5115	026166	100042				BPL	4\$ ;BRANCH IF NO ERROR	
5116	026170	012737	026262	001206		MOV	@#3\$,\$ESCAPE ;ESCAPE TO 3\$ ON ERROR	
5117	026176	013737	004136	001270		MOV	@#DPB.B+12,@#CYL.DS ;CYLINDER	
5118	026204	113737	004135	001274		MOVB	@#DPB.B+11,@#TRK.DS ;TRACK	
5119	026212	113737	004134	001272		MOVB	@#DPB.B+10,@#SEC.DS ;SECTOR	
5120	026220	012746	004142			MOV	@#DPB.B+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS	
5121	026224	004737	027246			JSR	PC,@#ERINDEX ;FORM DISPATCH INDEX	
5122	026230	062607				ADD	(SP)+,PC ;REPORT PROPER ERROR	
5123	026232	104041				ERROR	41 ;	
5124	026234	104042				ERROR	42 ;PARITY ERROR	
5125	026236	104043				ERROR	43 ;UNSAFE ERROR	
5126	026240	104044				ERROR	44 ;NON-I/O ERROR	
5127	026242	005737	004220			TST	RM.REG+RMER1 ;DRIVE ERROR ?	
5128	026246	001404				BEQ	2\$ ;BR IF NOT	
5129	026250	032737	177677	004220		BIT	@#1C100,RM.REG+RMER1 ;SEE IF ONLY 'HCE' SET	
5130	026256	001406				BEQ	4\$ ;BR IF IT IS	
5131	026260	104045			2\$:	ERROR	45 ;I/O ERROR	
5132	026262	013746	004142		3\$:	MOV	DPB.B+16,-(SP) ;STATUS WORD	
5133	026266	004737	027206			JSR	PC,LOP.CK ;SEE IF LOOP, ABORT, OR CONTINUE	
5134	026272	000410				BR	5\$ ;CHECK FOR STALL	
5135	026274	123727	004126	000173	4\$:	CMPB	@#DPB.B+2,@#READHD ;DOING IMPLIED SEEKS?	
5136	026302	001004				BNE	5\$ ;NO--BRANCH	
5137	026304	004037	027526			JSR	RO,@#VERIFY ;YES--GO CHECK THE DATA	
5138	026310	004134				DPB.B+10		
5139	026312	000407				BR	6\$ ;ERROR DURING VERIFY	
5140	026314	032737	040000	001220	5\$:	BIT	@#SW14,@#C.SWR ;STALL?	
5141	026322	001403				BEQ	6\$ ;NO--BRANCH	

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,@#CALL.B
    RETURN
```

```

5142 026324 004037 027364      JSR    RO,@#STALL      ;YES--CALL STALL ROUTINE
5143 026330 001354              .WORD  STALL1          ;STALL TIME POINTER
5144 026332 000200      6$:   RTS    RO          ;RETURN
5145
5146      ;THIS ROUTINE IS THE SAME AS "CALL.B" EXCEPT FOR THE DPB USED.
5147      ;CALL
5148      ;
5149      ;   FILL DPB
5150      ;   JSR    RO,@#CALL.C
5151      ;   RETURN
5152 026334 005037 001206      CALL.C: CLR    @#SESCAPE  ;NO ESCAPE ADDRESS
5153 026340 004037 036236      JSR    RO,@#RMO3      ;CALL RMO3 DRIVER
5154 026344 004144      DPB.C
5155 026346 000772      BR     CALL.C
5156 026350 005737 004162      1$:   TST    @#DPB.C+16  ;DONE?
5157 026354 001775      BEQ    1$             ;NO--LOOP
5158 026356 100042      BPL    4$             ;YES--BRANCH IF NO ERROR
5159 026360 012737 026452 001206      MOV    #3$,SESCAPE   ;ESCAPE TO 3$ ON ERROR
5160 026366 013737 004156 001270      MOV    @#DPB.C+12,@#CYL.DS ;CYLINDER
5161 026374 113737 004155 001274      MOVB   @#DPB.C+11,@#TRK.DS ;TRACK
5162 026402 113737 004154 001272      MOVB   @#DPB.C+10,@#SEC.DS ;SECTOR
5163 026410 012746 004162      MOV    @#DPB.C+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
5164 026414 004737 027246      JSR    PC,@#ERINDEX  ;FORM DISPATCH INDEX
5165 026420 062607      ADD    (SP)+,PC      ;REPORT PROPER ERROR
5166 026422 104041      ERROR  41
5167 026424 104042      ERROR  42             ;PARITY ERROR
5168 026426 104043      ERROR  43             ;UNSAFE ERROR
5169 026430 104044      ERROR  44             ;NON-I/O ERROR
5170 026432 005737 004220      TST    RM.REG+RMER1  ;DRIVE ERROR ?
5171 026436 001404      BEQ    2$             ;BR IF NOT
5172 026440 032737 177677 004220      BIT    #+C100,RM.REG+RMER1 ;SEE IF ONLY 'HCE' SET
5173 026446 001406      BEQ    4$             ;BR IF IT IS
5174 026450 104045      2$:   ERROR  45             ;I/O ERROR
5175 026452 013746 004162      3$:   MOV    DPB.C+16,-(SP) ;STATUS WORD
5176 026456 004737 027206      JSR    PC,LOP.CK     ;SEE IF LOOP, ABORT, OR CONTINUE
5177 026462 000410      BR     5$
5178 026464 123727 004146 000173      4$:   CMPB   @#DPB.C+2,@#READHD ;DOING IMPLIED SEEK?
5179 026472 001004      BNE    5$             ;NO--EXIT
5180 026474 004037 027526      JSR    RO,@#VERIFY   ;YES--CHECK THE DATA
5181 026500 004154      DPB.C+10
5182 026502 000407      BR     6$             ;ERROR DURING VERIFY
5183 026504 032737 040000 001220      5$:   BIT    #SW14,@#C.SWR  ;STALL?
5184 026512 001403      BEQ    6$             ;NO--BRANCH
5185 026514 004037 027364      JSR    RO,@#STALL    ;YES--CALL STALL ROUTINE
5186 026520 001354              .WORD  STALL1          ;STALL TIME POINTER
5187 026522 000200      6$:   RTS    RO
5188
5189
5190      ;THIS ROUTINE IS THE SAME AS "CALL.A" EXCEPT FOR THE DPB USED AND
5191      ;ON AN ERROR LOCATION "ERR.CT" IS EXAMINED. IF ERR.CT IS EQUAL TO
5192      ;SERFLG EXIT IS TO THE NEXT TEST.
5193      ;CALL
5194      ;
5195      ;   FILL DPB
5196      ;   JSR    RO,@#DRVCAL
5197      ;   RETURN

```

5198	026524	005037	001206			DRVCAL:	CLR	2#SESCAPE	: NO ESCAPE ADDRESS
5199	026530	005037	001334				CLR	2#WCEFLG	: CLEAR WRITE CHECK ERROR FLAG
5200	026534	004037	036236				JSR	RO,2#RMO3	: CALL RMO3 DRIVER
5201	026540	004164					DTADPB		
5202	026542	000770					BR	DRVCAL	
5203	026544	005737	004202			DRVCL1:	TST	2#DTADPB+16	: DONE
5204	026550	001775					BEQ	DRVCL1	: NO--LOOP
5205	026552	100402					BMI	1\$	: BR IF ERRORS
5206	026554	000137	027166				JMP	10\$	: NO ERRORS
5207	026560					1\$:			
5208	026560	012737	026634	001206			MOV	2\$SESCAPE	: ESCAPE TO 2\$ ON ERROR
5209	026566	013737	004176	001270			MOV	2#DTADPB+12,2#CYL.DS	: CYLINDER
5210	026574	113737	004175	001274			MOVB	2#DTADPB+11,2#TRK.DS	: TRACK
5211	026602	113737	004174	001272			MOVB	2#DTADPB+10,2#SEC.DS	: SECTOR
5212	026610	012746	004202				MOV	2#DTADPB+16,-(SP)	: STATUS/ERROR INDICATOR ADDRESS
5213	026614	004737	027246				JSR	PC,2#ERINDX	: FORM DISPATCH INDEX
5214	026620	062607					ADD	(SP)+,PC	: REPORT PROPER ERROR
5215	026622	104041					ERROR	41	
5216	026624	104042					ERROR	42	: PARITY ERROR
5217	026626	104043					ERROR	43	: UNSAFE ERROR
5218	026630	104044					ERROR	44	: NON-I/O ERROR
5219	026632	104045					ERROR	45	: I/O ERROR
5220	026634	122737	000020	001102	2\$:		CMPB	20,2#STSTNM	: TEST 20?
5221	026642	001137					BNE	8\$	: NO--BRANCH
5222	026644	013746	004202				MOV	DTADPB+16,-(SP)	: STATUS WORD
5223	026650	004737	027206				JSR	PC,LOP.CK	: SEE IF LOOP ABORT, OR CONTINUE
5224	026654	122737	000151	004166			CMPB	2#WRCKD,2#DTADPB+2	: DOING A WRITE CHECK?
5225	026662	001133					BNE	12\$	: NO--BRANCH
5226	026664	032737	040000	004214			BIT	2#BIT14,2#RM.REG+10	: IS "WCE"=1?
5227	026672	001527					BEQ	12\$	: NO--BRANCH
5228	026674	032777	000020	152236			BIT	2#SW04,2#SWR	: INHIBIT WRITES?
5229	026702	001123					BNE	12\$	: YES--BRANCH
5230	026704	112737	000161	004166			MOVB	2#WRITE,2#DTADPB+2	: SETUP FOR A WRITE
5231	026712	005037	001206				CLR	2#SESCAPE	: NO ESCAPE ADDRESS
5232	026716	004037	036236				JSR	RO,2#RMO3	: DO THE WRITE
5233	026722	004164					DTADPB		
5234	026724	000240					NOP		
5235	026726	005737	004202		3\$:		TST	2#DTADPB+16	: DONE?
5236	026732	001775					BEQ	3\$	: NO--LOOP
5237	026734	100026					BPL	4\$	: YES--BRANCH IF NO ERROR
5238	026736	012737	027142	001206			MOV	8\$SESCAPE	: ESCAPE TO 8\$ ON ERROR
5239	026744	013737	004176	001270			MOV	2#DTADPB+12,2#CYL.DS	: CYLINDER
5240	026752	113737	004175	001274			MOVB	2#DTADPB+11,2#TRK.DS	: TRACK
5241	026760	113737	004174	001272			MOVB	2#DTADPB+10,2#SEC.DS	: SECTOR
5242	026766	012746	004202				MOV	2#DTADPB+16,-(SP)	: STATUS/ERROR INDICATOR ADDRESS
5243	026772	004737	027246				JSR	PC,2#ERINDX	: FORM DISPATCH INDEX
5244	026776	062607					ADD	(SP)+,PC	: REPORT PROPER ERROR
5245	027000	104041					ERROR	41	
5246	027002	104042					ERROR	42	: PARITY ERROR
5247	027004	104043					ERROR	43	: UNSAFE ERROR
5248	027006	104044					ERROR	44	: NON-I/O ERROR
5249	027010	104045					ERROR	45	: I/O ERROR
5250	027012	112737	000151	004166	4\$:		MOVB	2#WRCKD,2#DTADPB+2	: COMMAND=WRITE CHECK DATA
5251	027020	004037	036236				JSR	RO,2#RMO3	: DO THE WRITE CHECK
5252	027024	004164					DTADPB		
5253	027026	000240					NOP		

```

5254 027030 005737 004202 5$: TST @#DTADPB+16 ;DONE?
5255 027034 001775 BEQ 5$ ;NO--LOOP
5256 027036 100410 BMI 7$ ;YES--BRANCH IF ERROR
5257 027040 004037 036236 JSR RO,@#RMO3 ;DO A 2ND WRITE CHECK
5258 027044 004164 DTADPB
5259 027046 000240 NOP
5260 027050 005737 004202 6$: TST @#DTADPB+16 ;DONE?
5261 027054 001775 BEQ 6$ ;NO--LOOP
5262 027056 100043 BPL 10$ ;YES--BRANCH IF NO ERROR
5263 027060 012737 000001 001334 7$: MOV @#1,@#WCEFLG ;SET THE WRITE CHECK ERROR FLAG
5264 027066 012737 027142 001206 @#8$,@#ESCAPE ;ESCAPE TO 8$ ON ERROR
5265 027074 013737 004176 001270 MOV @#DTADPB+12,@#CYL.DS ;CYLINDER
5266 027102 113737 004175 001274 MOV @#DTADPB+11,@#TRK.DS ;TRACK
5267 027110 113737 004174 001272 MOV @#DTADPB+10,@#SEC.DS ;SECTOR
5268 027116 012746 004202 MOV @#DTADPB+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
5269 027122 004737 027246 JSR PC,@#ERINDX ;FORM DISPATCH INDEX
5270 027126 062607 ADD (SP)+,PC ;REPORT PROPER ERROR
5271 027130 104041 ERROR 41
5272 027132 104042 ERROR 42 ;PARITY ERROR
5273 027134 104043 ERROR 43 ;UNSAFE ERROR
5274 027136 104044 ERROR 44 ;NON-I/O ERROR
5275 027140 104046 ERROR 46 ;FATAL WRITE CHECK
5276 027142 013746 004202 8$: MOV DTADPB+16,-(SP) ;STATUS WORD
5277 027146 004737 027206 JSR PC,LOP.CK ;SEE IF LOOP, ABORT, OR CONTINUE
5278 027152 123737 001364 001103 12$: CMPB @#ERR.CT,@#SERFLG ;GO TO NEXT TEST?
5279 027160 101002 BHI 10$ ;NO--BRANCH
5280 027162 013700 001252 9$: MOV @#BYPASS,RO ;YES--GET EXIT ADDRESS
5281 027166 032737 040000 001220 10$: BIT #SW14,@#C.SWR ;STALL?
5282 027174 001403 BEQ 11$ ;NO--BRANCH
5283 027176 004037 027364 JSR RO,@#STALL ;YES--CALL STALL ROUTINE
5284 027202 001356 .WORD STALL2 ;STALL TIME POINTER
5285 027204 000200 11$: RTS RO
5286
5287 ;THIS SUBROUTINE CHECK FOR LOOP, ABORT, OR CONTINUE SWITCHES AFTER
5288 ;ERRORS 41, 42, 43, 44, 45, AND 46.
5289 ;CALL
5290 ; MOV DTA+16,-(SP) ;STATUS WORD FROM DPB IN USE
5291 ; JSR PC,LOP.CK
5292 ;
5293 ; RETURN
5294 027206 032777 001000 151724 LOP.CK: BIT #SW9,@#SWR ;LOOP ON ERROR
5295 027214 001402 BEQ 1$ ;BR IF NOT
5296 027216 000177 151666 JMP @#SLPERR ;START AT THE LOOP ADDRESS
5297 027222 005037 001206 1$: CLR @#ESCAPE ;CLEAR ERROR ESCAPE FLAG
5298 027226 032766 072006 000002 BIT @#BIT14!@#BIT13!@#BIT12!@#BIT10!@#BIT02!@#BIT01,2(SP) ;CHECK ERROR TYPE
5299 027234 001402 BEQ 2$ ;BR IF DRIVE NOT OFFLINE, UNLOADED, OR
5300 ;PERSISTENT UNSAFE OR FATAL MASSBUS PARITY
5301 027236 000137 020406 2$: JMP @#EOP ;TERMINATE DRIVE
5302 027242 012616 MOV (SP)+,(SP) ;ADJUST RETURN ADDRESS
5303 027244 000207 RTS PC
5304
5305 ;THIS ROUTINE FORMS AN INDEX THAT WILL BE USED TO DISPATCH
5306 ;TO THE PROPER ERROR CALL. THE INDEX IS FORMED BY EXAMINING
5307 ;THE STATUS/ERROR INDICATOR OF THE APPLICABLE DPB.
5308 ;INDEX STATUS/ERROR
5309 -----

```

```

5310      0 BIT14!BIT13!BIT08!BIT01
5311      2 BIT11!BIT10!BIT02
5312      4 BIT12!BIT04
5313      6 BIT05!BIT03!<BIT09 & COMMAND=NON-I/O>
5314      10 BIT06!<BIT09 & COMMAND=I/O>
5315      :CALL
5316      JSR      #DPB+16,-(SP)      ;ADDRESS OF STATUS/ERROR INDICATOR
5317      JSR      PC,@#ERINDX      ;FORM INDEX
5318      RETURN      ;INDEX IS ON THE STACK
5319
5320      ERINDX: MOV      RO,-(SP)      ;SAVE RO
5321      MOV      RI,-(SP)      ;SAVE RI
5322      MOV      6(SP),RO      ;GET STATUS/ERROR INDICATOR POINTER
5323      MOV      (RO),@#SVSTAT      ;SAVE THE STATUS/ERROR INDICATOR
5324      CLR      RI      ;START INDEX AT ZERO
5325      BIT      (PC)+,(RO)      ;FORM INDEX OF 0?
5326      .WORD   BIT13!BIT08!BIT01
5327      BNE      5$      ;YES--BRANCH
5328      BIT      (PC)+,(RO)      ;FORM PARITY ERROR OR PORT REQUEST INDEX (2)?
5329      .WORD   BIT11!BIT10!BIT02
5330      BNE      4$      ;YES--BRANCH
5331      BIT      (PC)+,(RO)      ;FORM UNSAFE INDEX (4)?
5332      .WORD   BIT14!BIT12!BIT04
5333      BNE      3$      ;YES--BRANCH
5334      BIT      (PC)+,(RO)      ;FORM NON-I/O ERROR INDEX (6)?
5335      .WORD   BIT05!BIT03
5336      BNE      2$      ;YES--BRANCH
5337      BIT      (PC)+,(RO)      ;FORM I/O ERROR INDEX (10)?
5338      .WORD   BIT06
5339      BNE      1$      ;YES--BRANCH
5340      BIT      (PC)+,(RO)      ;SOFTWARE TIMEOUT?
5341      .WORD   BIT09
5342      BEQ      5$      ;NO--FORM INDEX OF 0
5343      CMPB    #150,-16(RO)      ;YES--I/O?
5344      BGT      2$      ;NO--BRANCH
5345      1$: INC      RI      ;INDEX=10---ERROR=45 OR 46
5346      2$: INC      RI      ;INDEX=6---ERROR=44
5347      3$: INC      RI      ;INDEX=4---ERROR=43
5348      4$: INC      RI      ;INDEX=2---ERROR=42
5349      5$: ASL     RI      ;INDEX=0---ERROR=41
5350      MOV      RI,6(SP)      ;RETURN INDEX TO USER
5351      MOV      (SP)+,RI      ;RESTORE RI
5352      MOV      (SP)+,RO      ;RESTORE RO
5353      RTS      PC      ;RETURN FROM CALL
5354
5355      ;THIS ROUTINE WILL PROVIDE A STALL IN MILLISECONDS FOR A SPECIFIC
5356      ;AMOUNT OF TIME IF BIT13 OF C.SWR = 0 OR A RANDOM AMOUNT OF TIME
5357      ;IF BIT 13 OF C.SWR = 1.
5358      ;STALL1 CONTAINS SPECIFIED TIME FOR TESTS 0 - 7, AND STALL2
5359      ;CONTAINS THE TIME FOR TESTS 16-21.
5360      :CALL
5361      JSR      RO,@#STALL      ;WHERE TO FIND THE STALL TIME
5362      TIME POINTER
5363
5364      STALL: MOV      @ (RO)+,-(SP)      ;PICKUP STALL TIME
5365      BIT      #SW13,@#C.SWR      ;USE A RANDOM TIME?

```



```

5366 027374 001406          BEQ      1$          ;NO--BRANCH
5367 027376 004737 024510    JSR      PC,2$SRAND ;YES--FORM RANDOM NUMBER
5368 027402 013716 024610    MOV      2$,$LONUM,(SP) ;AND USE IT FOR THE STALL TIME
5369 027406 042716 177700    BIC      #1C77,(SP) ;BUT NEVER > 64 MILLISECOND
5370 027412 005046          1$: CLR      -(SP) ;CLEAR TEMP. LOCATION
5371 027414 162766 000001 000002 2$: SUB      #1,2(SP) ;MORE STALL REQUIRED?
5372 027422 103407          BLO      4$          ;NO--BRANCH
5373 027424 012716 000144    MOV      #100.,(SP) ;STALL FOR ABOUT 1 MILLISECOND
5374 027430 005700          3$: TST      R0 ;NOP TO KILL TIME
5375 027432 005366 000000    DEC      0(SP) ;COUNT
5376 027436 001374          BNE      3$          ;LOOP IF MORE COUNTS NEEDED
5377 027440 000765          BR       2$
5378 027442 022626          4$: CMP      (SP)+,(SP)+ ;CLEAN OFF THE STACK
5379 027444 000200          RTS      R0 ;EXIT

```

```

5380
5381
5382 ;ROUTINE TO PROVIDE A 2 MS STALL AFTER A SEEK OPERATION IN THE SEEK TIMING
5383 ;TESTS. THIS STALL IS REQUIRED TO COMPENSATE FOR THE 'ACCESS READY' DELAY
5384 ;IN THE RMO3. THIS STALL TIME IS NOT INCLUDED IN THE CALCULATED SEEK TIMES.
5385 ;CALL
5386 ;
5387 ;
5388 ;

```

```

5389 027446 013746 177776          TWOMS: MOV      2$PS,-(SP) ;SAVE THE PRESENT PROCESSOR STATUS
5390 027452 012737 000240 177776    MOV      #(<5*32.)>,2$PS ;SET THE PROCESSOR PRIORITY TO 5
5391 027460 017746 151710          MOV      2$PKV,-(SP) ;SAVE THE OLD CLOCK VECTOR ADDRESS
5392 027464 012777 027510 151702    MOV      #1$,2$PKV ;SETUP NEW VECTOR ADDRESS
5393 027472 012777 000310 151702    MOV      #200.,2$PKB ;LOAD THE CLOCK BUFFER
5394 027500 012777 000101 151672    MOV      #101,2$PKCS ;START THE CLOCK
5395 027506 000001          WAIT ;WAIT FOR 2 MS
5396 027510 062706 000004          1$: ADD      #4,SP ;INCREMENT STACK FOR RETURN
5397 027514 012677 151654          MOV      (SP)+,2$PKV ;RESTORE OLD CLOCK VECTOR
5398 027520 012637 177776          MOV      (SP)+,2$PS ;RESTORE THE OLD PROCESSOR STATUS
5399 027524 000207          RTS      PC ;RETURN

```

```

5400
5401 ;ROUTINE TO SOFTWARE COMPARE HEADER ON IMPLIED SEEKS
5402 ;CALL
5403 ;
5404 ;
5405 ;
5406 ;

```

```

5407 027526 010146          VERIFY: MOV      R1,-(SP) ;SAVE R1
5408 027530 012001          MOV      (R0)+,R1 ;GET ADDRESS OF DPB+10
5409 027532 042737 150000 050514    BIC      #150000,2$BUFFER ;STRIP FORMAT AND BAD SECTOR BITS FROM CYLINDER NUMBER
5410 027540 023761 050514 000002    CMP      2$BUFFER,2(R1) ;CYLINDER NUMBER OK?
5411 027546 001003          BNE      1$          ;NO--BRANCH
5412 027550 023711 050516          CMP      2$BUFFER+2,(R1) ;YES--HOW ABOUT TRACK/SECTOR?
5413 027554 001441          BEQ      3$          ;BRANCH IF GOOD
5414 027556 013737 050514 001262          1$: MOV      2$BUFFER,2$CYL.RD ;SAVE THE EXPECTED AND THE
5415 027564 113737 050517 001264    MOV      2$BUFFER+3,2$TRK.RD ;RECEIVED CYLINDER, TRACK,
5416 027572 113737 050516 001266    MOV      2$BUFFER+2,2$SEC.RD ;AND SECTOR
5417 027600 112137 001272          MOV      (R1)+,2$SEC.DS
5418 027604 112137 001274          MOV      (R1)+,2$TRK.DS
5419 027610 011137 001270          MOV      (R1),2$CYL.DS
5420 027614 012737 027626 001206    MOV      #2$,2$ESCAPE ;ESCAPE TO 2$ ON ERROR
5421 027622 005740          TST      -(R0) ;MAKE IT TEST PC+4

```

```

5422 027624 104012          ERROR 12          ;REPORT THE ERROR
5423 027626 012737 000107 004106 2$:  MOV  #RECAL,DPB,A+2 ;LOAD RECALIBRATE ORDER CODE
5424 027634 004037 026032          JSR  RO,#CALL.A    ;GO EXECUTE THE COMMAND
5425 027640 005037 001206          CLR  #ESCAPE      ;CLEAR ERROR ESCAPE FLAG
5426 027644 032777 001000 151266  BIT  #SW9,#SWR    ;LOOP ON ERROR ?
5427 027652 001404          BEQ  4$          ;BR IF NOT
5428 027654 000177 151230          JMP  #SLPERR     ;RETURN TO ERROR LOOP ADDRESS
5429 027660 062700 000002          3$:  ADD  #2,RO    ;INCREMENT RETURN ADDRESS
5430 027664 012601          4$:  MOV  (SP)+,R1  ;RESTORE R1
5431 027666 000200          RTS  RO         ;EXIT
5432
5433          ;THIS ROUTINE WILL PERFORM A "MASSBUS" INIT. FOLLOWED BY
5434          ;A "RECALIBRATE" ON THE DRIVE UNDER TEST.
5435          ;NOTE: THIS ROUTINE DESTROYS R1 AND R4
5436          ;CALL
5437          ;      JSR  RO,SRCHOO ;DO A MASSBUS INIT. AND RECAL
5438          ;      RETURN1 ;RETURN HERE IF NO ERROR
5439          ;      RETURN2 ;RETURN HERE ON ERROR
5440
5441          SRCHOO: CLR  R1          ;INCASE OF ERROR (TYPTIM)
5442          027672 005037 177776          CLR  #PS
5443          027676 012777 040320 005612  MOV  #ISR,#RMEC ;SETUP INTERRUPT VECTOR
5444          027704 013704 035514          MOV  #RMAOR,R4  ;PICKUP ADDRESS OF RMCS1
5445          027710 012764 000040 000010  MOV  #BIT05,RMCS2(R4) ;MASSBUS INIT.
5446          027716 005037 004174          CLR  #DTADPB+10 ;TRACK=0; SECTOR=0
5447          027722 005037 004176          CLR  #DTADPB+12 ;CYLINDER =0
5448          027726 012737 000107 004166  MOV  #RECAL,#DTADPB+2 ;COMMAND = RECALIBRATE
5449          027734 005037 001206          CLR  #SESCAPE   ;NO ESCAPE ADDRESS
5450          027740 004037 036236          JSR  RO,#RMO3   ;CALL THE DRIVER
5451          027744 004164          DTADPB ;DPB POINTER
5452          027746 000440          BR   4$        ;QUEUE IS FULL
5453          027750 005737 004202          1$:  TST  DTADPB+16 ;WAIT ON DONE
5454          027754 001775          BEQ  1$
5455          027756 100030          BPL  3$        ;TAKE NORMAL EXIT IF NO ERROR
5456          027760 012737 030034 001206  MOV  #2$,#SESCAPE ;ESCAPE TO 2$ ON ERROR
5457          027766 013737 004176 001270  MOV  #DTADPB+12,#CYL.DS ;CYLINDER
5458          027774 113737 004175 001274  MOVB #DTADPB+11,#TRK.DS ;TRACK
5459          030002 113737 004174 001272  MOVB #DTADPB+10,#SEC.DS ;SECTOR
5460          030010 012746 004202          MOV  #DTADPB+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
5461          030014 004737 027246          JSR  PC,#ERINDX ;FORM DISPATCH INDEX
5462          030020 062607          ADD  (SP)+,PC  ;REPORT PROPER ERROR
5463          030022 104041          ERROR 41
5464          030024 104042          ERROR 42          ;PARITY ERROR
5465          030026 104043          ERROR 43          ;UNSAFE ERROR
5466          030030 104044          ERROR 44          ;NON-I/O ERROR
5467          030032 104045          ERROR 45          ;I/O ERROR
5468          030034 005720          2$:  TST  (RO)+    ;ADJUST FOR ERROR EXIT
5469          030036 000404          BR   4$        ;GO TO THE EXIT
5470          030040 005064 000006          3$:  CLR  RMDA(R4) ;TRACK AND SECTOR = 0
5471          030044 005064 000034          CLR  RMDC(R4)  ;CYLINDER = 0
5472          030050 000200          4$:  RTS  RO         ;RETURN
5473
5474          ;THIS IS AN RTI WHICH IS USED BY THE TIMING TESTS & THE SERVO SETTLE DOWN TEST
5475
5476          DORTI: RTI
5477          ;RETURN FROM INTERRUPT

```

```

5478 ; THIS ROUTINE WILL INITIALIZE THE TIMERS USED BY THE "TIMING ROUTI ES
5479 ; CALL
5480 ; JSR PC,@#STRTMR
5481 ; RETURN
5482 ;
5483 030054 104412 STRTMR: SAVREG ; SAVE R0-R5
5484 030056 012700 001276 MOV #TIM.UP,R0 ; START AT TIM.UP (MINIMUM)
5485 030062 012701 001332 MOV #TIM.PT,R1 ; STOP AT TIM.PT
5486 030066 005020 1$: CLR (R0)+ ; CLEAR
5487 030070 020001 CMP R0,R1 ; DONE?
5488 030072 103775 BLO 1$ ; NO--BRANCH
5489 030074 012710 050514 MOV #BUFFER,(R0) ; SETUP POINTER
5490 030100 012737 077777 001276 MOV #↑CBIT15,@#TIM.UP ; SET MINIMUM TIME TO MAXIMUM
5491 030106 012737 077777 001314 MOV #↑CBIT15,@#TIM.DN ; POSITIVE NUMBER
5492 030114 104413 RESREG ; RESTORE R0-R5
5493 030116 000207 RTS PC ; RETURN
5494
5495 ; THIS ROUTINE WILL ADD THE ELAPSED TIME TO THE AVERAGE COUNTER AND
5496 ; MAINTAIN THE MINIMUM AND MAXIMUM TIMES.
5497 ; NOTE: THIS ROUTINE DESTROYS R2
5498 ; CALL
5499 ; MOV #TP,R3 ; PARAMETER POINTER
5500 ; MOV FLAG,R5 ; FLAG=0=COUNT UP
5501 ; ; FLAG=-1=COUNT DOWN
5502 ;
5503 ; JSR PC,@#COUNT
5504 ; RETURN
5505 030120 012702 001276 COUNT: MOV #TIM.UP,R2 ; PICKUP THE "UP" POINTER
5506 030124 005705 TST R5 ; USE IT?
5507 030126 001402 BEQ 1$ ; YES--BRANCH
5508 030130 012702 001314 MOV #TIM.DN,R2 ; NO--PICKUP "DOWN" POINTER
5509 030134 027722 151244 1$: CMP #PKC,(R2)+ ; LESS THAN PREVIOUS LOW?
5510 030140 002003 BGE 2$ ; NO--BRANCH
5511 030142 017762 151236 177776 MOV #PKC,-2(R2) ; YES--SAVE IT
5512 030150 027763 151230 000004 2$: CMP #PKC,4(R3) ; LESS THAN THE LOW LIMIT?
5513 030156 002001 BGE 3$ ; NO--BRANCH
5514 030160 005212 INC (R2) ; YES--COUNT IT
5515 030162 005722 3$: TST (R2)+ ; ADVANCE THE POINTER
5516 030164 027722 151214 CMP #PKC,(R2)+ ; GREATER THAN PREVIOUS HIGH?
5517 030170 003403 BLE 4$ ; NO--BRANCH
5518 030172 017762 151206 177776 MOV #PKC,-2(R2) ; YES--SAVE IT
5519 030200 027763 151200 000006 4$: CMP #PKC,6(R3) ; GREATER THAN THE HIGH LIMIT?
5520 030206 003401 BLE 5$ ; NO--BRANCH
5521 030210 005212 INC (R2) ; YES--COUNT IT
5522 030212 005722 5$: TST (R2)+ ; ADVANCE THE POINTER
5523 030214 067722 151164 ADD #PKC,(R2)+ ; ADD THIS COUNT TO THE TOTAL
5524 030220 005522 ADC (R2)+
5525 030222 005212 INC (R2) ; COUNT THIS READING
5526 030224 022737 057004 001332 CMP #BUFFER+(4*814),@#TIM.PT ; SAVE THIS COUNT?
5527 030232 101406 BLOS 6$ ; NO--BRANCH
5528 030234 017777 151144 151070 MOV #PKC,@TIM.PT ; YES--WELL SAVE IT THEN
5529 030242 062737 000002 001332 ADD #2,@#TIM.PT ; ADVANCE THE POINTER
5530 030250 000207 6$: RTS PC ; RETURN
5531
5532 ; THIS ROUTINE IS USED TO TYPE THE MINIMUM,
5533 ; MAXIMUM, AND AVERAGE TIMES FOR THE TIMING TESTS

```

```

5534 ;IT WILL ALSO CHECK THE TIMES TO ENSURE
5535 ;THEY ARE WITHIN TOLERANCE AND IF NOT FLAG THE BAD TIMES.
5536 ;NOTE: THIS ROUTINE DESTROYS R2-R5
5537 ;CALL
5538 ;JSR RO,#TYPTIM ;GO REPORT THE TIMES
5539 ;TABLE ;POINT TO THE PROPER TABLE
5540 ;RETURN
5541
5542 ;TABLE:MSGADR1 ;ADDRESS OF ASCIZ MESSAGE NUMBER 1
5543 ;MSGADR2 ;ADDRESS OF ASCIZ MESSAGE NUMBER 2
5544 ;MIN.ALLOWED ;MINIMUM TIME ALLOWED
5545 ;MAX.ALLOWED ;MAXIMUM TIME ALLOWED
5546
5547 030252 012002 TYPT M: MOV (R0)+,R2 ;PICKUP THE TABLE POINTER
5548 030254 032777 000100 150656 BIT #SW06,#SWR ;INHIBIT TIME REPORTS?
5549 030262 001145 BNE 7$ ;YES--BRANCH
5550 030264 012237 030304 MOV (R2)+,2$ ;ADDRESS OF MESSAGE NUMBER 1
5551 030270 012205 MOV (R2)+,R5 ;ADDRESS OF MESSAGE NUMBER 2
5552 030272 012203 MOV (R2)+,R3 ;PICKUP THE LOW LIMIT
5553 030274 011202 MOV (R2),R2 ;AND THE HIGH LIMIT
5554 030276 012704 001276 MOV #TIM.UP,R4 ;PARAMETER POINTER
5555 030302 104401 1$: TYPE ;TYPE THE MESSAGE
5556 030304 000000 2$: .WORD 0 ;ASCIZ MESSAGE POINTER GOES HERE
5557 030306 005764 000014 TST 14(R4) ;DID ANY COUNTS OCCUR?
5558 030312 001527 BEQ 6$ ;NO--BRANCH
5559 030314 104401 044517 TYPE ,MSGMIN ;"MIN="
5560 030320 012446 MOV (R4)+,-(SP) ;PUT (R4)+ ON THE STACK
5561 030322 004737 024220 JSR PC,#$S82D ;CHANGE (R4)+ TO DECIMAL ASCIZ
5562 030326 004737 024450 JSR PC,#$SUPRS ;TYPE WITHOUT LEADING ZEROS
5563 030332 104401 044544 TYPE ,MSGOUS ;"0 US"
5564 030336 005724 TST (R4)+ ;ANY SEEKS BELOW THE LOW LIMIT
5565 030340 001421 BEQ 3$ ;NO--BRANCH
5566 030342 104401 044657 TYPE ,MSG.SP ;" "
5567 030346 016446 MOV -2(R4),-(SP) ;PUT -2(R4) ON THE STACK
5568 030352 004737 024220 JSR PC,#$S82D ;CHANGE -2(R4) TO DECIMAL ASCIZ
5569 030356 004737 024450 JSR PC,#$SUPRS ;TYPE WITHOUT LEADING ZEROS
5570 030362 104401 044551 TYPE ,M$BELOW ;"BELOW THE MINIMUM OF"
5571 030366 010346 MOV R3,-(SP) ;PUT R3 ON THE STACK
5572 030370 004737 024220 JSR PC,#$S82D ;CHANGE R3 TO DECIMAL ASCIZ
5573 030374 004737 024450 JSR PC,#$SUPRS ;TYPE WITHOUT LEADING ZEROS
5574 030400 104401 044544 TYPE ,MSGOUS
5575 030404 104401 044526 3$: TYPE ,M$AX ;"MAX="
5576 030410 012446 MOV (R4)+,-(SP) ;PUT (R4)+ ON THE STACK
5577 030412 004737 024220 JSR PC,#$S82D ;CHANGE (R4)+ TO DECIMAL ASCIZ
5578 030416 004737 024450 JSR PC,#$SUPRS ;TYPE WITHOUT LEADING ZEROS
5579 030422 104401 044544 TYPE ,MSGOUS
5580 030426 005724 TST (R4)+ ;ANY SEEKS ABOVE THE HIGH LIMIT
5581 030430 000421 BR 4$ ;NO--BRANCH
5582 030432 104401 044657 TYPE ,MSG.SP ;YES--REPORT HOW MANY
5583 030436 016446 MOV -2(R4),-(SP) ;PUT -2(R4) ON THE STACK
5584 030442 004737 024220 JSR PC,#$S82D ;CHANGE -2(R4) TO DECIMAL ASCIZ
5585 030446 004737 024450 JSR PC,#$SUPRS ;TYPE WITHOUT LEADING ZEROS
5586 030452 104401 044600 TYPE ,M$ABOVE ;"ABOVE THE MAXIMUM OF"
5587 030456 010246 MOV R2,-(SP) ;PUT R2 ON THE STACK
5588 030460 004737 024220 JSR PC,#$S82D ;CHANGE R2 TO DECIMAL ASCIZ
5589 030464 004737 024450 JSR PC,#$SUPRS ;TYPE WITHOUT LEADING ZEROS

```

5590	030470	104401	044544				
5591	030474	104401	044535	4\$:	TYPE	,MSGOUS	
5592	030500	012446			TYPE	,MSGAVG	; "AVG="
5593	030502	012446			MOV	(R4)+, -(SP)	; FORM THE AVERAGE
5594	030504	012446			MOV	(R4)+, -(SP)	
5595	030506	004737	024612		MOV	(R4)+, -(SP)	
5596	030512	006126			JSR	PC, @#\$DIV	
5597	030514	100001			ROL	(SP)+	; IS THE REMAINDER OVER HALF?
5598	030516	005216			BPL	5\$	; NO--BRANCH
5599	030520			5\$:	INC	(SP)	; YES--ROUND UP
5600	030520	004737	024220		JSR	PC, @#\$S82D	; CHANGE TO DECIMAL ASCIZ
5601	030524	004737	024450		JSR	PC, @#\$SUPRS	; TYPE WITHOUT LEADING ZEROS
5602	030530	104401	044544		TYPE	,MSGOUS	
5603	030534	104401	044657		TYPE	,MSG.SP	
5604	030540	016446	177776		MOV	-2(R4), -(SP)	; PUT -2(R4) ON THE STACK
5605	030544	004737	024220		JSR	PC, @#\$S82D	; CHANGE -2(R4) TO DECIMAL ASCIZ
5606	030550	004737	024450		JSR	PC, @#\$SUPRS	; TYPE WITHOUT LEADING ZEROS
5607	030554	104401	044627		TYPE	,MSGNUM	; "SEEKS TIMED"
5608	030560	010537	030304		MOV	R5, 2\$	; NEXT MESSAGE POINTER
5609	030564	001404			BEQ	7\$	; IF NONE EXIT
5610	030566	005705			CLR	R5	; NO MORE THAN 2
5611	030570	000644			BR	1\$	
5612	030572	104401	044644	6\$:	TYPE	,MSGNON	
5613	030576	000200		7\$:	RTS	R0	; EXIT
5614							
5615							
5616							
5617							
5618							
5619							
5620							
5621							
5622	030600	020237	001520		JSR	R0, @#INCTRK	
5623	030604	001410			RETURN1		; TRACK NUMBER GREATER THAN LT15
5624	030606	063702	001522		RETURN2		; TRACK NUMBER INCREMENTED
5625	030612	020237	001520				
5626	030616	003402					
5627	030620	013702	001520		INCTRK: CMP	R2, @#LT	; LAST TRACK COMPLETED?
5628	030624	005720			BEQ	2\$	; YES--EXIT
5629	030626	000200			ADD	@#IT, R2	; NO--UPDATE TRACK
5630					CMP	R2, @#LT	; TRACK TO BIG?
5631					BLE	1\$	; NO--EXIT
5632					MOV	@#LT, R2	; YES--SET TRACK TO LAST TRACK
5633				1\$:	TST	(R0)+	; ADJUST FOR RETURN 2
5634				2\$:	RTS	R0	; RETURN
5635							
5636							
5637							
5638							
5639	030630	020137	001512				
5640	030634	001410					
5641	030636	063701	001514				
5642	030642	020137	001512				
5643	030646	003402					
5644	030650	013701	001512				
5645	030654	005720					

; THIS SUBROUTINE WILL INCREMENT THE TRACK  
NUMBER (R2) BY THE AMOUNT SPECIFIED BY 'IT'.  
CALL

JSR R0, @#INCTRK  
RETURN1 ; TRACK NUMBER GREATER THAN LT15  
RETURN2 ; TRACK NUMBER INCREMENTED

INCTRK: CMP R2, @#LT ; LAST TRACK COMPLETED?  
BEQ 2\$ ; YES--EXIT  
ADD @#IT, R2 ; NO--UPDATE TRACK  
CMP R2, @#LT ; TRACK TO BIG?  
BLE 1\$ ; NO--EXIT  
MOV @#LT, R2 ; YES--SET TRACK TO LAST TRACK  
1\$: TST (R0)+ ; ADJUST FOR RETURN 2  
2\$: RTS R0 ; RETURN

; THIS SUBROUTINE WILL INCREMENT THE CYLINDER  
NUMBER (R1) BY THE AMOUNT SPECIFIED BY 'IC'.  
CALL

JSR R0, @#INCCYL  
RETURN1 ; CYLINDER NUMBER GREATER THAN LC15  
RETURN2 ; CYLINDER NUMBER INCREMENTED

INCCYL: CMP R1, @#LC ; LAST CYLINDER COMPLETED?  
BEQ 2\$ ; YES--EXIT  
ADD @#IC, R1 ; NO--UPDATE CYLINDER  
CMP R1, @#LC ; CYLINDER TO BIG?  
BLE 1\$ ; NO--EXIT  
MOV @#LC, R1 ; YES--SET CYLINDER TO LAST CYLINDER  
1\$: TST (R0)+ ; ADJUST FOR RETURN 2

```

5646 030656 000200 2$: RTS RO ;RETURN
5647
5648 ;THIS ROUTINE DECREMENTS THE SECTOR ADDRESS.
5649 ;CALL
5650 ; CLR -(SP) ;CLEAR THE STACK
5651 ; JSR PC,DECSEC ;SUBROUTINE ENTRY
5652 ;
5653 ;
5654 030660 113766 004212 000002 DECSEC: MOVB RM.REG+RMDA,2(SP) ;PUT THE SECTOR ADDRESS ON THE STACK
5655 030666 005366 000002 DEC 2(SP) ;DECREMENT THE ADDRESS
5656 030672 100003 BPL 1$ ;BR IF NOT CORRECTION NEEDED
5657 030674 013766 001634 000002 MCV PRMLMT+22.,2(SP) ;OVERFLOW OCCURED, FORCE TO MAXIMUM ADDRESS
5658 030702 000207 1$: RTS PC ;RETURN
5659
5660 ;THIS SUBROUTINE IS USED TO FILL THE DATA BUFFER
5661 ;WITH ADDRESSES FROM 0 TO 21 WITH EACH ADDRESS
5662 ;BEING STORED IN 256 CONSECUTIVE LOCATIONS
5663 ;CALL
5664 ; JSR PC,2$FILBUF
5665 ;
5666 ;
5667 030704 104412 FILBUF: SAVREG ;SAVE RO - R5
5668 030706 005000 CLR RO ;FIRST DISK ADDRESS
5669 030710 012701 050514 MOV #BUFFER,R1 ;START FILLING HERE
5670 030714 012702 000400 1$: MOV #256.,R2 ;DO 256 WORDS
5671 030720 010021 2$: MOV RO,(R1) ;STORE
5672 030722 005302 DEC R2 ;MORE?
5673 030724 003375 BGT 2$ ;YES--BRANCH
5674 030726 005200 INC RO ;NO--UPDATE DISK ADDRESS
5675 030730 023700 001630 CMP PRMLMT+22,RO ;DONE?
5676 030734 103367 BHIS 1$ ;NO--BRANCH
5677 030736 104413 RESREG ;RESTORE RO - R5
5678 030740 000207 RTS PC ;RETURN
5679
5680 ;THIS ROUTINE WILL CLEAR THE BUFFER BY
5681 ;SETTING EACH WORD TO "177400".
5682 ;CALL
5683 ; JSR RO,2$CLRBUF
5684 ;
5685 ;
5686 030742 104412 CLRBUF: SAVREG ;SAVE RO - R5
5687 030744 012701 177400 MOV #177400,R1 ;WORD TO FILL BUFFER WITH
5688 030750 012702 050514 MOV #BUFFER,R2 ;FIRST ADDRESS OF BUFFER
5689 030754 012703 076514 MOV #BUFFER+(512.*22.),R3 ;LAST ADDRESS+2 OF BUFFER
5690 030760 010122 1$: MOV R1,(R2)+ ;FILL WORDS 1,9,...249,...5625
5691 030762 010122 MOV R1,(R2)+ ;FILL WORDS 2,10,...250,...5626
5692 030764 010122 MOV R1,(R2)+ ;FILL WORDS 3,11,...251,...5627
5693 030766 010122 MOV R1,(R2)+ ;FILL WORDS 4,12,...252,...5628
5694 030770 010122 MOV R1,(R2)+ ;FILL WORDS 5,13,...253,...5629
5695 030772 010122 MOV R1,(R2)+ ;FILL WORDS 6,14,...254,...5630
5696 030774 010122 MOV R1,(R2)+ ;FILL WORDS 7,15,...255,...5631
5697 030776 010122 MOV R1,(R2)+ ;FILL WORDS 8,16,...256,...5632
5698 031000 020203 CMP R2,R3 ;DONE?
5699 031002 103766 BLO 1$ ;NO--BRANCH
5700 031004 104413 RESREG ;RESTORE RO - R5
5701 031006 000200 RTS RO ;RETURN FROM CALL

```

```

5702
5703
5704
5705
5706
5707
5708
5709
5710 031010 104412
5711 031012 162706 000004
5712 031016 005001
5713 031020 012716 050514
5714 031024 005066 000002
5715 031030 012702 000020
5716 031034 011603
5717 031036
5718 031036 020123
5719 031040 001063
5720 031042 020123
5721 031044 001061
5722 031046 020123
5723 031050 001057
5724 031052 020123
5725 031054 001055
5726 031056 020123
5727 031060 001053
5728 031062 020123
5729 031064 001051
5730 031066 020123
5731 031070 001047
5732 031072 020123
5733 031074 001045
5734 031076 020123
5735 031100 001043
5736 031102 020123
5737 031104 001041
5738 031106 020123
5739 031110 001037
5740 031112 020123
5741 031114 001035
5742 031116 020123
5743 031120 001033
5744 031122 020123
5745 031124 001031
5746 031126 020123
5747 031130 001027
5748 031132 020123
5749 031134 001025
5750 031136 005302
5751 031140 001336
5752 031142 062716 001000
5753 031146 005201
5754 031150 023701 001630
5755 031154 103325
5756 031156 005766 000002
5757 031162 001406

; THIS ROUTINE IS USED TO CHECK THE DATA BUFFER
; FOR ADDRESSES 0 THROUGH 21 WITH EACH ADDRESS
; BEING STORED IN 256 CONSECUTIVE LOCATIONS
CALL
; JSR RD,2#CKSCTR
; RETURN
CKSCTR: SAVREG ; SAVE RD - R5
SUB #4,SP ; RESERVE TEMP. STORAGE AREA
CLR R1 ; FIRST SECTOR
MOV #BUFFER,(SP) ; FIRST ADDRESS OF DATA BUFFER
CLR 2(SP) ; NO ERRORS
1$: MOV #16,R2 ; LOOP COUNT (16*16=256)
MOV (SP),R3 ; GET 1ST ADDRESS OF THIS SECTORS DATA
2$: CMP R1,(R3)+ ; WORD 1
BNE 7$ ; BRANCH IF BAD
CMP R1,(R3)+ ; WORD 2
BNE 7$ ; BRANCH IF BAD
CMP R1,(R3)+ ; WORD 3
BNE 7$ ; BRANCH IF BAD
CMP R1,(R3)+ ; WORD 4
BNE 7$ ; BRANCH IF BAD
CMP R1,(R3)+ ; WORD 5
BNE 7$ ; BRANCH IF BAD
CMP R1,(R3)+ ; WORD 6
BNE 7$ ; BRANCH IF BAD
CMP R1,(R3)+ ; WORD 7
BNE 7$ ; BRANCH IF BAD
CMP R1,(R3)+ ; WORD 8
BNE 7$ ; BRANCH IF BAD
CMP R1,(R3)+ ; WORD 9
BNE 7$ ; BRANCH IF BAD
CMP R1,(R3)+ ; WORD 10
BNE 7$ ; BRANCH IF BAD
CMP R1,(R3)+ ; WORD 11
BNE 7$ ; BRANCH IF BAD
CMP R1,(R3)+ ; WORD 12
BNE 7$ ; BRANCH IF BAD
CMP R1,(R3)+ ; WORD 13
BNE 7$ ; BRANCH IF BAD
CMP R1,(R3)+ ; WORD 14
BNE 7$ ; BRANCH IF BAD
CMP R1,(R3)+ ; WORD 15
BNE 7$ ; BRANCH IF BAD
CMP R1,(R3)+ ; WORD 16
BNE 7$ ; BRANCH IF BAD
DEC R2 ; FINISHED WITH THIS SECTORS DATA?
BNE 2$ ; NO--BRANCH
3$: ADD #512,(SP) ; YES--FIRST ADDRESS OF NEXT SECTOR
INC R1 ; MOVE TO NEXT SECTOR
CMP PRMLMT+22,R1 ; DONE?
BHS 1$ ; NO--BRANCH
4$: TST 2(SP) ; ERROR OCCUR?
BEQ 6$ ; NO--BRANCH

```

```

5758 031164 123737 001364 001103      CMPB    2#ERR.CT,2#SERFLG ;MAX. ERROR OCCURRED?
5759 031172 101002                    BHI     6$                ;NO--BRANCH
5760 031174 013700 001252      5$:    MOV     2#BYPASS,R0    ;TAKE ERROR EXIT
5761 031200 062706 000004      6$:    ADD     #4,SP         ;FREE TEMP. AREA
5762 031204 104413                    RESREG                    ;RESTORE R0 - R5
5763 031206 000200                    RTS     R0                ;RETURN FROM CALL
5764 031210 010304      7$:    MOV     R3,R4         ;FORM WORD NUMBER AND
5765 031212 161604                    SUB     (SP),R4          ;ADDRESS TO CONTINUE FROM
5766 031214 010405                    MOV     R4,R5
5767 031216 006204                    ASR     R4                ;WORD NUMBER
5768 031220 042705 177740                    BIC     #1C37,R5
5769 031224 001002                    BNE     8$                ;BRANCH IF NOT A MULTIPLE OF 16
5770 031226 012705 000040                    MOV     #40,R5          ;SET TO WORD 16
5771 031232 006305      8$:    ASL     R5
5772 031234 062705 031036                    ADD     #2$,R5          ;ADDRESS
5773 031240 016337 177776 001126                    MOV     -2(R3),2#$BDDAT ;SAVE BAD DATA
5774 031246 005766 000002                    TST     2(SP)           ;FIRST ERROR?
5775 031252 001015                    BNE     10$              ;NO--BRANCH
5776 031254 013737 004176 001270                    MOV     2#DTADPB+12,2#CYL.DS ;CYLINDER NUMBER
5777 031262 113737 004175 001274                    MOV     2#DTADPB+11,2#TRK.DS ;TRACK NUMBER
5778 031270 012737 001300 001206                    MOV     #9$,SESCAPE    ;ESCAPE TO 9$ ON ERROR
5779 031276 104021                    ERROR   21              ;REPORT THE ERROR
5780 031300 105166 000002      9$:    COMB   2(SP)         ;SET ERROR SWITCH
5781 031304 000404                    BR      11$
5782 031306                    10$:
5783 031306 012737 031316 001206                    MOV     #11$,SESCAPE   ;ESCAPE TO 11$ ON ERROR
5784 031314 104022                    ERROR   22              ;REPORT THE ERROR
5785 031316 032777 001000 147614      11$:   BIT     #SW09,2$SWR    ;LOOP ON ERROR?
5786 031324 001323                    BNE     5$                ;YES
5787 031326 032777 000002 147604                    BIT     #SW01,2$SWR    ;STOP DATA COMPARE?
5788 031334 001310                    BNE     4$                ;YES--BRANCH
5789 031336 123737 001364 001103      CMPB    2#ERR.CT,2#SERFLG ;MAX. ERRORS?
5790 031344 101713                    BLOS   5$                ;YES--BRANCH
5791 031346 032777 000040 147564                    BIT     #SW05,2$SWR    ;REPORT ONLY 1ST ERROR PER SECTOR?
5792 031354 001272                    BNE     3$                ;YES--BRANCH
5793 031356 000115                    JMP     (R5)
5794
5795 ; THIS ROUTINE WILL MOVE THE 16 WORDS OF THE
5796 ; DESIRED PATTERN INTO THE DATA BUFFER.
5797 ; CALL
5798 ;
5799 ;
5800 ;
5801 031360 104412      SETBUF: SAVREG                    ;SAVE R0 - R5
5802 031362 012701 050514                    MOV     #BUFFER,R1     ;FIRST ADDRESS
5803 031366 013702 004170                    MOV     2#DTADPB+4,R2  ;WORD COUNT
5804 031372 016003 003044      1$:    MOV     PAT.PT(R0),R3 ;PICKUP PATTERN POINTER
5805 031376 012321                    MOV     (R3)+,(R1)+    ;MOVE WORD 1 INTO DATA BUFFER
5806 031400 012321                    MOV     (R3)+,(R1)+    ;MOVE WORD 2 INTO DATA BUFFER
5807 031402 012321                    MOV     (R3)+,(R1)+    ;MOVE WORD 3 INTO DATA BUFFER
5808 031404 012321                    MOV     (R3)+,(R1)+    ;MOVE WORD 4 INTO DATA BUFFER
5809 031406 012321                    MOV     (R3)+,(R1)+    ;MOVE WORD 5 INTO DATA BUFFER
5810 031410 012321                    MOV     (R3)+,(R1)+    ;MOVE WORD 6 INTO DATA BUFFER
5811 031412 012321                    MOV     (R3)+,(R1)+    ;MOVE WORD 7 INTO DATA BUFFER
5812 031414 012321                    MOV     (R3)+,(R1)+    ;MOVE WORD 8 INTO DATA BUFFER
5813 031416 012321                    MOV     (R3)+,(R1)+    ;MOVE WORD 9 INTO DATA BUFFER

```



```

5814 031420 012321      MOV      (R3)+,(R1)+      ;MOVE WORD 10 INTO DATA BUFFER
5815 031422 012321      MOV      (R3)+,(R1)+      ;MOVE WORD 11 INTO DATA BUFFER
5816 031424 012321      MOV      (R3)+,(R1)+      ;MOVE WORD 12 INTO DATA BUFFER
5817 031426 012321      MOV      (R3)+,(R1)+      ;MOVE WORD 13 INTO DATA BUFFER
5818 031430 012321      MOV      (R3)+,(R1)+      ;MOVE WORD 14 INTO DATA BUFFER
5819 031432 012321      MOV      (R3)+,(R1)+      ;MOVE WORD 15 INTO DATA BUFFER
5820 031434 012321      MOV      (R3)+,(R1)+      ;MOVE WORD 16 INTO DATA BUFFER
5821 031436 062702 000020  ADD      0,6.,R2 ;DONE?
5822 031442 001353      BNE      1$
5823 031444 104413      RESREG
5824 031446 000207      RTS      PC              ;RETURN

```

THIS ROUTINE COMPARES A 16 WORD DATA PATTERN  
AGAINST THE DATA BUFFER

```

CALL
      MOV      #NX,R0      ;PATTERN NUMBER INDEX TO R0
      JSR      PC,#DATCMP
      RETURN

```

```

DATCMP: SAVREG      ;SAVE R0 - R5
      MOV      #BUFFER,R1  ;FIRST ADDRESS OF BUFFER
      MOV      #DTADPB+4,R2 ;WORD COUNT
      CLR      -(SP)       ;NO ERROR
1$:     MOV      PAT.PT(R0),R3 ;PATTERN POINTER
2$:

```

```

5839 031470 162321      SUB      (R3)+,(R1)+      ;CHECK WORD 1
5840 031472 001044      BNE      4$              ;BRANCH IF DIFFERENT
5841 031474 162321      SUB      (R3)+,(R1)+      ;CHECK WORD 2
5842 031476 001042      BNE      4$              ;BRANCH IF DIFFERENT
5843 031500 162321      SUB      (R3)+,(R1)+      ;CHECK WORD 3
5844 031502 001040      BNE      4$              ;BRANCH IF DIFFERENT
5845 031504 162321      SUB      (R3)+,(R1)+      ;CHECK WORD 4
5846 031506 001036      BNE      4$              ;BRANCH IF DIFFERENT
5847 031510 162321      SUB      (R3)+,(R1)+      ;CHECK WORD 5
5848 031512 001034      BNE      4$              ;BRANCH IF DIFFERENT
5849 031514 162321      SUB      (R3)+,(R1)+      ;CHECK WORD 6
5850 031516 001032      BNE      4$              ;BRANCH IF DIFFERENT
5851 031520 162321      SUB      (R3)+,(R1)+      ;CHECK WORD 7
5852 031522 001030      BNE      4$              ;BRANCH IF DIFFERENT
5853 031524 162321      SUB      (R3)+,(R1)+      ;CHECK WORD 8
5854 031526 001026      BNE      4$              ;BRANCH IF DIFFERENT
5855 031530 162321      SUB      (R3)+,(R1)+      ;CHECK WORD 9
5856 031532 001024      BNE      4$              ;BRANCH IF DIFFERENT
5857 031534 162321      SUB      (R3)+,(R1)+      ;CHECK WORD 10
5858 031536 001022      BNE      4$              ;BRANCH IF DIFFERENT
5859 031540 162321      SUB      (R3)+,(R1)+      ;CHECK WORD 11
5860 031542 001020      BNE      4$              ;BRANCH IF DIFFERENT
5861 031544 162321      SUB      (R3)+,(R1)+      ;CHECK WORD 12
5862 031546 001016      BNE      4$              ;BRANCH IF DIFFERENT
5863 031550 162321      SUB      (R3)+,(R1)+      ;CHECK WORD 13
5864 031552 001014      BNE      4$              ;BRANCH IF DIFFERENT
5865 031554 162321      SUB      (R3)+,(R1)+      ;CHECK WORD 14
5866 031556 001012      BNE      4$              ;BRANCH IF DIFFERENT
5867 031560 162321      SUB      (R3)+,(R1)+      ;CHECK WORD 15
5868 031562 001010      BNE      4$              ;BRANCH IF DIFFERENT
5869 031564 162321      SUB      (R3)+,(R1)+      ;CHECK WORD 16

```

```

5870 031566 001006          BNE      4$          ; BRANCH IF DIFFERENT
5871 031570 062702 000020  ADD      #16.,R2      ; DONE ?
5872 031574 001333          BNE      1$          ; NO--BRANCH
5873 031576 005726          3$: TST      (SP)+      ; YES -- CLEAN UP STACK
5874 031600 104413          RESREG          ; RESTORE R0 - R5
5875 031602 000207          RTS           PC
5876 031604 010104          4$: MOV      R1,R4      ; FORM THE WORD NUMBER
5877 031606 162704 050514  SUB      #BUFFER,R4
5878 031612 006204          ASR      R4          ; WORD NUMBER
5879 031614 010305          MOV      R3,R5      ; FORM ADDRESS TO CONTINUE FROM
5880 031616 166005 003044  SUB      PAT.PT(R0),R5
5881 031622 006305          ASL      R5
5882 031624 062705 031470  ADD      #2$,R5      ; ADDRESS
5883 031630 064341          ADD      -(R3),-(R1) ; RECONSTRUCT THE BAD WORD
5884 031632 010137 001122  MOV      R1,@#$BDAOR ; SAVE THE ERROR INFORMATION
5885 031636 010337 001120  MOV      R3,@#$GDAOR
5886 031642 012137 001126  MOV      (R1)+,@#$BDDAT
5887 031646 012337 001124  MOV      (R3)+,@#$GDDAT
5888 031652 005716          TST      (SP)      ; 1ST DATA COMPARE ERROR?
5889 031654 001023          BNE      6$          ; NO--BRANCH
5890 031656 013737 004176 001270  MOV      @#DTADPB+12,@#CYL.DS ; CYLINDER
5891 031664 113737 004175 001274  MOV      @#DTADPB+11,@#TRK.DS ; TRACK
5892 031672 113737 004174 001272  MOV      @#DTADPB+10,@#SEC.DS ; SECTOR
5893 031700 016600 000016  MOV      16(SP),R0 ; GET TEST PC+4
5894 031704 012737 031714 001206  MOV      #5$, $ESCAPE ; ESCAPE TO 5$ ON ERROR
5895 031712 104013          ERROR      13      ; REPORT THE ERROR
5896 031714 016600 000014  5$: MOV      14(SP),R0 ; PATTERN NUMBER INDEX
5897 031720 105116          COMB      (SP)      ; SET THE ERROR SWITCH
5898 031722 000404          BR       7$
5899 031724          6$: MOV      #7$, $ESCAPE ; ESCAPE TO 7$ ON ERROR
5900 031724 012737 031734 001206  ERROR      14      ; REPORT THE ERROR
5901 031732 104014          7$: BIT      #SW01,$SWR ; STOP DATA COMPARE?
5902 031734 032777 000002 147176  BNE      3$          ; YES--EXIT
5903 031742 001315          CMPB     @#ERR.CT,@#SERFLG ; MAX. ERRORS?
5904 031744 123737 001364 001103  BHI      8$          ; NO--BRANCH
5905 031752 101004          MOV      @#BYPASS,16(SP) ; YES--ERROR EXIT
5906 031754 013766 001252 000016  BR       3$
5907 031762 000705          8$: JMP      (R5)      ; NO--CONTINUE AT NEXT WORD
5908 031764 000115
5909
5910 ; THIS ROUTINE WILL FILL THE DATA BUFFER (256*22 WORDS) WITH
5911 ; A RANDOM PATTERN. THE FIRST TWO WORDS OF EVERY 256 WILL
5912 ; BE THE BASE OF THE RANDOM NUMBER GENERATOR FOR THE
5913 ; NEXT 254 WORDS.
5914 ; NOTE: THIS ROUTINE DESTROYS R1 AND R2
5915 ; CALL
5916 ; JSR      RO,@#FILRAN
5917 ; RETURN
5918
5919 FILRAN: MOV      #BUFFER,R1
5920 031772 013702 001630          MOV      PRMLT+22,R2 ; MAXIMUM NUMBER OF SECTORS
5921 031776 004037 032206  1$: JSR      RO,@#RANPAT
5922 032002 005302          DEC      R2
5923 032004 100374          BPL      1$
5924 032006 000200          RTS      R0
5925

```

```

5926 ; THIS ROUTINE USES THE FIRST TWO WORDS OF THE
5927 ; READ BUFFER TO GENERATED A RANDOM PATTERN. THEN
5928 ; THE READ BUFFER IS COMPARED TO THE PATTERN GENERATED.
5929 ; NOTE: THIS ROUTINE DESTROYS R1-R4
5930 ; CALL
5931 ; JSR RO, @#RANCK
5932 ; RETURN
5933
5934 032010 013746 024606 RANCK: MOV @#SHINUM, -(SP) ; SAVE THE PRESENT RANDOM NUMBER
5935 032014 013746 024610 MOV @#SLONUM, -(SP) ; READ BUFFER ADDRESS
5936 032020 012702 051514 MOV @BUFFER+512., R2 ; RANDOM PATTERN ADDRESS
5937 032024 012701 052514 MOV @BUFFER+1024., R1 ; COPY IT INTO R3 FOR LATER USE
5938 032030 010103 MOV R1, R3 ; PRIME THE RANDOM NUMBER GENERATOR
5939 032032 011237 024610 MOV (R2), @#SLONUM
5940 032036 016237 000002 024606 MOV 2(R2), @#SHINUM
5941 032044 004037 032206 JSR RO, @#RANPAT ; GENERATE A RANDOM PATTERN
5942 032050 012637 024610 MOV (SP)+, @#SLONUM ; RESTORE PRESENT RANDOM NUMBER
5943 032054 012637 024606 MOV (SP)+, @#SHINUM
5944 032060 005046 CLR -(SP) ; NO ERRORS
5945 032062 162322 15: SUB (R3)+, (R2)+ ; ARE THESE TWO WORDS DIFFERENT?
5946 032064 001441 BEQ 45 ; NO--BRANCH
5947 032066 012737 032140 001206 MOV #3$, $ESCAPE ; ESCAPE TO 3$ ON ERROR
5948 032074 064342 ADD -(R3), -(R2) ; RECREATE THE BAD WORD
5949 032076 010237 001122 MOV R2, @#$BADDR ; ADDRESS OF BAD DATA
5950 032102 010337 001120 MOV R3, @#$GADDR ; ADDRESS OF GOOD DATA
5951 032106 012237 001126 MOV (R2)+, @#$BDDAT ; BAD DATA
5952 032112 012337 001124 MOV (R3)+, @#$GDDAT ; GOOD DATA
5953 032116 010204 MOV R2, R4 ; FORM WORD NUMBER (1 TO 256)
5954 032120 162704 051514 SUB @BUFFER+512., R4
5955 032124 006204 ASR R4
5956 032126 005716 TST (SP) ; FIRST ERROR
5957 032130 001002 BNF 25 ; NO--BRANCH
5958 032132 105116 COMB (SP) ; YES--SET ERROR SWITCH
5959 032134 104015 ERROR 15 ; REPORT THE ERROR
5960 032136 104016 ERROR 16 ; REPORT THE ERROR
5961 032140 032777 001000 146772 25: BIT #SW09, @SWR ; LOOP ON ERROR?
5962 032146 001012 BNE 55 ; YES--BRANCH
5963 032150 123737 001364 001103 CMPB @#ERR.CT, @#SERFLG ; MAX. ERRORS OCCURRED?
5964 032156 101406 BLOS 55 ; YES--BRANCH
5965 032160 032777 000002 146752 BIT #SW01, @SWR ; STOP COMPARING?
5966 032166 001002 BNE 55 ; YES--BRANCH
5967 032170 020103 45: CMP R1, R3 ; ALL DATA BEEN COMPARED?
5968 032172 101333 BHI 15 ; NO--BRANCH
5969 032174 005726 55: TST (SP)+ ; ERROR OCCUR?
5970 032176 001402 BEQ 65 ; NO--BRANCH
5971 032200 013700 001252 MOV @#BYPASS, RO ; TAKE ERROR EXIT
5972 032204 000200 65: RTS RO ; EXIT
5973
5974 ; THIS ROUTINE FILLS A 256 WORD BUFFER WITH A RANDOM
5975 ; PATTERN OF WHICH THE FIRST TWO WORDS ARE THE BASE
5976 ; OF THE PATTERN.
5977 ; CALL
5978 ; MOV #ADR, R1 ; ADDRESS OF THE BUFFER
5979 ; JSR RO, @#RANPAT
5980 ; RETURN
5981

```

```

5982 032206 010246
5983 032210 012702 000200
5984 032214 000402
5985 032216 004737 024510
5986 032222 013721 024610
5987 032226 013721 024606
5988 032232 005302
5989 032234 003370
5990 032236 012602
5991 032240 000200
5992
5993
5994
5995
5996
5997
5998
5999
6000 032242 004737 024510
6001 032246 113701 024610
6002 032252 113702 024611
6003 032256 013703 024606
6004 032262 105701
6005 032264 002403
6006 032266 123701 001630
6007 032272 103003
6008 032274 000241
6009 032276 106001
6010 032300 000772
6011 032302 105702
6012 032304 002403
6013 032306 122702 000005
6014 032312 003003
6015 032314 000241
6016 032316 106002
6017 032320 000772
6018 032322 023703 001510
6019 032326 003413
6020 032330 000241
6021 032332 006003
6022 032334 005503
6023 032336 001371
6024 032340 010103
6025 032342 000303
6026 032344 060203
6027 032346 005203
6028 032350 003364
6029 032352 005403
6030 032354 000762
6031 032356 023703 001512
6032 032362 002003
6033 032364 000241
6034 032366 006003
6035 032370 000772
6036 032372 023703 001510
6037 032376 003403
    
```

```

RANPAT: MOV R2, -(SP) ;SAVE R2
        MOV #256./2., R2 ;GENERATE 256 WORDS
        BR 2$
1$: JSR PC, @SRAND ;GENERATE A RANDOM NUMBER
2$: MOV @SLONUM, (R1)+ ;PUT LOW WORD IN BUFFER
    MOV @SHINUM, (R1)+ ;PUT HIGH WORD IN BUFFER
    DEC R2 ;DONE?
    BGT 1$ ;NO--BRANCH
    MOV (SP)+, R2 ;RESTORE R2
    RTS R0 ;EXIT

; THIS ROUTINE GENERATES RANDOM CYLINDER, TRACK, AND SECTOR
; ADDRESSES AND SAVES THEM IN THE DPB (DTADPB+10, 11 & DTADPB+12).
; NOTE: THIS ROUTINE DESTROYS R1-R3
; CALL
; JSR RD, @RANADR
; RETURN

RANADR: JSR PC, @SRAND ;GENERATE A RANDOM NUMBER
        MOV @SLONUM, R1 ;FORM SECTOR IN R1
        MOV @SLONUM+1, R2 ;FORM TRACK IN R2
        MOV @SHINUM, R3 ;FORM CYLINDER IN R3
        TSTB R1 ;ENSURE THE SECTOR IS BETWEEN 0 AND 31
        JLT 2$
1$: CMPB PRMLT+22, R1 ;CHECK MAXIMUM SECTOR ADDRESS
    BHIS 3$
2$: CLC
    RORB R1
    BR 1$
3$: TSTB R2 ;ENSURE THE TRACK IS BETWEEN 0 AND 4
    BLT 5$
4$: CMPB #5, R2
    BGT 6$
5$: CLC
    RORB R2
    BR 4$
6$: CMP @FC, R3 ;ENSURE THE CYLINDER IS BETWEEN FC AND LC
    BLE 7$
    CLC
    ROR R3
    ADC R3
    BNE 6$
    MOV R1, R3
    SWAB R3
    ADD R2, R3
    INC R3
    BGT 6$
    NEG R3
    BR 6$
7$: CMP @LC, R3
    BGE 8$
    CLC
    ROR R3
    BR 7$
8$: CMP @FC, R3
    BLE 9$
    
```

```

6038 032400 005203          INC      R3
6039 032402 000303          SWAB    R3
6040 032404 000764          BR      7$
6041 032406 110137 004174    9$:     MOVB   R1, @DTADPB+10 ;SAVE SECTOR ADDRESS
6042 032412 110237 004175    MOVB   R2, @DTADPB+11 ;SAVE TRACK ADDRESS
6043 032416 010337 004176    MOV    R3, @DTADPB+12 ;SAVE CYLINDER ADDRESS
6044 032422 000200          RTS     R0 ;RETURN
6045
6046 ; THIS ROUTINE IS USED TO INPUT THE "CONTROL SWITCHES".
6047 ; IF SWR<07>=1 THE PRESENT SETTING WILL BE TYPED AND THE NEW
6048 ; SETTING IS READ AND STORED.
6049 ; NOTE: THIS ROUTINE DESTROYS R3 AND R4
6050 ; CALL
6051          JSR     PC, @GETSWR
6052          RETURN ; (C.SWR)=DESIRED CONTROL SWITCHES
6053
6054 032424 032777 000200 146506 GETSWR: BIT     #SW07, @SWR ;READ CONTROL SWITCHES?
6055 032432 001430          BEQ     2$ ;NO--BRANCH
6056 032434 104401 032442          TYPE   65$ ;TYPE ASCIZ STRING
6057 032440 000410          BR     64$ ;GET OVER THE ASCIZ
6058          ; 65$: .ASCIZ <CR><LF>/SET SWR<07>=0/
6059 032462 012703 043704    64$:   1$:     MOV    #MSG.CS, R3 ;"CONTROL SWITCHES="
6060 032462 013704 001220    MOV    @C.SWR, R4 ;PRESENT CONTROL SWITCH SETTINGS
6061 032466 004037 032516    JSR   R0, @GETNUM ;GET THE NEW SWITCH SETTINGS
6062 032472 000771          BR     1$ ;COMMA
6063 032476 000240          NOP    ;PERIOD
6064 032500 013737 001220 001222    MOV    C.SWR, SAVCSW ;SAVE PREVIOUS VALUE
6065 032502 010437 001220    MOV    R4, @C.SWR ;DOUBLE PERIOD--SAVE NEW SWITCH SETTING
6066 032510 000207          RTS     PC ;RETURN FROM CALL
6067 032514 000207
6068
6069 ; THIS ROUTINE WILL TYPE AN ASCIZ MESSAGE AND THEN
6070 ; INPUT AN ASCIZ STRING AND CHANGE THE STRING TO OCTAL
6071 ; IF REQUIRED.
6072 ; NOTE: THIS ROUTINE DESTROYS R1
6073 ; CALL
6074          MOV    #ADR, R3 ;ADDRESS OF ASCIZ MESSAGE
6075          MOV    #NUM, R4 ;OCTAL NUMBER
6076          JSR   R0, @GETNUM
6077          RETURN1 ;INPUT TERMINATED WITH A COMMA
6078          RETURN2 ;WITH A PERIOD
6079          RETURN3 ;WITH A DOUBLE PERIOD
6080          ; R4=INPUT NUMBER AND
6081          ; R2=R4*32 FOR ALL
6082          ; THREE RETURNS
6083
6084 032516 010337 032524    GETNUM: MOV    R3, 2$ ;SAVE MESSAGE POINTER
6085 032522 104401          1$:     TYPE   ;TYPE THE MESSAGE
6086 032524 000000          2$:     .WORD 0 ;MESSAGE POINTER GOES HERE
6087 032526 010446          MOV    R4, -(SP) ;SAVE R4 FOR TYPEOUT
6088 032530 104402          TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
6089 032532 104401 043730          TYPE   , SLASH ; /
6090 032536 104411          RDLIN ;READ AN ASCIZ STRING
6091 032540 012601          MOV    (SP)+, R1 ;ADDRESS OF FIRST CHARACTER
6092 032542 004037 034766    JSR   R0, @CK.CHR ;CHECK ONE CHARACTER
6093 032546 032522          1$:     ;ILLEGAL CHARACTER

```

```

6094 032550 032522          1$          ;CARRIAGE RETURN
6095 032552 032564          3$          ;"/
6096 032554 032610          7$          ;"
6097 032556 032616          8$          ;"
6098 032560 032562          11$         ;DIGIT 0-9
6099 032562 005301          35: DEC R1          ;DECREMENT THE INPUT POINTER
6100 032564                35:          ;
6101 032564 004037 035226    JSR R0,#CK.NUM      ;CHECK THE NUMBER
6102 032570 032522          1$          ;ILLEGAL INPUT
6103 032572 032604          6$          ;TERMINATED WITH A "," OR "CR"
6104 032574 032602          5$          ;TERMINATED WITH A "."
6105 032576 032600          4$          ;TERMINATED WITH A ".."
6106 032600 005720          45: TST (R0)+      ;DOUBLE PERIOD
6107 032602 005720          55: TST (R0)+      ;SINGLE PERIOD
6108 032604 010204          65: MOV R2,R4      ;COMMA--SAVE INPUT NUMBER
6109 032606 000414          BR 10$        ;GO TO EXIT
6110 032610 105711          75: TSTB (R1)      ;TERMINATOR AFTER A COMMA?
6111 032612 001343          BNE 1$        ;NO--LOOP
6112 032614 000411          BR 10$        ;YES--EXIT
6113 032616 105711          85: TSTB (R1)      ;TERMINATOR AFTER A PERIOD?
6114 032620 001406          BEQ 9$          ;YES--EXIT
6115 032622 122721 000056    CMPB #'.,(R1)+      ;NO--DOUBLE PERIOD?
6116 032626 001335          BNE 1$        ;NO--LOOP
6117 032630 105711          TSTB (R1)      ;YES--TERMINATOR?
6118 032632 001333          BNE 1$        ;NO--LOOP
6119 032634 005720          TST (R0)+      ;DOUBLE PERIOD
6120 032636 005720          95: TST (R0)+      ;PERIOD
6121 032640 010402          10$: MOV R4,R2      ;COMMA--POSITION THE
6122 032642 000302          SWAB R2        ;NUMBER IN CASE IT
6123 032644 006202          ASR R2        ;IS THE PRIORITY LEVEL
6124 032646 006202          ASR R2
6125 032650 006202          ASR R2
6126 032652 000200          RTS R0        ;EXIT
6127
6128 ;THIS ROUTINE IS USED TO CHANGE OR MODIFY
6129 ;THE TEST PARAMETERS. IT GIVES THE OPERATOR
6130 ;THE CAPABILITY OF SPECIFYING WHICH DRIVES TO TEST, WHICH
6131 ;TESTS TO RUN AND HOW MANY TIMES TO
6132 ;REPEAT EACH TEST
6133
6134 032654 104412          GT.PRM: SAVREG      ;SAVE R0 - R5
6135 032656 005037 001232    GT.PRI: CLR DRVSEL  ;NO DRIVE SELECTED
6136 032662 104401 032670    TYPE 65$         ;TYPE ASCIZ STRING
6137 032666 000406          BR 64$         ;GET OVER THE ASCIZ
6138
6139 032704                ;:65$: .ASCIZ <CR><LF>/DRIVE(S)=/
6140 032704 104411          64$:          ;
6141 032706 012601          RDLIN          ;READ TTY
6142 032710 004037 034766    MOV (SP)+,R1      ;ADDRESS OF ASCIZ STRING
6143 032714 032656          JSR R0,#CK.CHR    ;CHECK ONE CHARACTER
6144 032716 032656          GT.PRI          ;ILLEGAL CHARACTER
6145 032720 032656          GT.PRI          ;CARRIAGE RETURN
6146 032722 032656          GT.PRI          ;"/
6147 032724 032656          GT.PRI          ;"
6148 032726 032730          1$          ;DIGIT 0-9
6149 032730 005301          15: DEC R1

```

```

6150 032732          2$:
6151 032732 012702 000007      MOV      #7,R2      ;UPPER LIMIT OF INPUT
6152 032736 004037 035042      JSR      RO,#CK.DIG ;CHECK THE DIGIT(S)
6153 032742 032656          GT.PR1      ;ILLEGAL INPUT
6154 032744 032656          GT.PR1      ;INPUT TO LARGE
6155 032746 032754          3$:          ;TERMINATED WITH A "." OR "CR"
6156 032750 033000          4$:          ;TERMINATED WITH A ":"
6157 032752 033000          4$:          ;TERMINATED WITH A "-"
6158 032754 156237 035502 001232 3$: BISB     ATABIT(R2),DRVSEL ;SET THE DRIVE SELECTED BIT
6159 032762 105741          TSTB     -(R1)      ;WAS THE LINE TERMINATED?
6160 032764 001362          BNE     2$        ;NO-GET THE NEXT DRIVE
6161 032766 005037 001234      CLR     @TSTNMS    ;DESELECT ALL TESTS
6162 032772 005037 001236      CLR     TSTNMS+2
6163 032776 000405          BR      GTTST1    ;YES--SELECT TEST
6164 033000 156237 035502 001232 4$: BISB     ATABIT(R2),DRVSEL ;SET THE SELECTED DRIVE BITS
6165 033006 104413          GT.PR2: RESREG    ;RESTORE RO - R5
6166 033010 000207          RTS      PC      ;EXIT
6167
6168 033012          GTTST1:
6169 033012 104401 033020      TYPE     65$      ;:TYPE ASCIZ STRING
6170 033016 000403          BR      64$      ;:GET OVER THE ASCIZ
6171
6172 033026          ;:65$: .ASCIZ /TEST=/
6173 033026 104411          ;:64$:
6174 033030 012601          RDLIN     ;READ AN ASCIZ STRING
6175 033032 122711 000056      MOV      (SP)+,R1 ;POINTER TO R1
6176 033036 001007          CMPB     #'.,(R1) ;DOUBLE PERIOD?
6177 033040 122761 000056 000001 BNE     1$        ;NO--BRANCH
6178 033046 001003          CMPB     #'.,1(R1)
6179 033050 105761 000002          BNE     1$
6180 033054 001754          TSTB     2(R1)    ;"CR"?
6181 033056 005037 001234          BEQ     GT.PR2    ;YES--EXIT
6182 033062 005037 001236          1$:      CLR     TSTNMS    ;NO TEST SELECTED
6183 033066 005037 001240          CLR     TSTNMS+2
6184 033072 005037 001242          CLR     OPNFLG   ;NO TESTS TO BE OPENED
6185 033076 121127 000123          CLR     OPNFLG+2
6186 033102 001004          GTTST2: CMPB     (R1),#'S ;ALL SEEK TESTS?
6187 033104 052737 000777 001234 BNE     1$        ;NO--BRANCH
6188 033112 000552          BIS     #777,TSTNMS ;YES--SELECT TESTS 0-10
6189 033114 121127 000124          BR      GTTST3
6190 033120 001004          1$:      CMPB     (R1),#'T ;ALL TIMING TESTS?
6191 033122 052737 036000 001234 BNE     2$        ;NO--BRANCH
6192 033130 000543          BIS     #36000,TSTNMS ;YES--SELECT TESTS 12-15
6193 033132 121127 000101          BR      GTTST3
6194 033136 001004          2$:      CMPB     (R1),#'A ;ALL ADDRESSING TESTS?
6195 033140 052737 140000 001234 BNE     3$        ;NO--BRANCH
6196 033146 000534          BIS     #140000,TSTNMS ;YES--SELECT TESTS 16 & 17
6197 033150 121127 000104          BR      GTTST3
6198 033154 001004          3$:      CMPB     (R1),#'D ;DATA TEST?
6199 033156 052737 000001 001236 BNE     4$        ;NO--BRANCH
6200 033164 000525          BIS     #1,TSTNMS+2 ;YES--SELECT TEST 20
6201 033166 121127 000105          BR      GTTST3
6202 033172 001004          4$:      CMPB     (R1),#'E ;EXERCISER TEST?
6203 033174 052737 000002 001236 BNE     5$        ;NO--BRANCH
6204 033202 000516          BIS     #2,TSTNMS+2 ;YES--SELECT TEST 21
6205 033204 004037 034712          BR      GTTST3
5$:      JSR      RO,#CK.OCT ;OCTAL DIGIT?

```

```

6206 033210 000514 BR GTTST4 ;NO--BRANCH
6207 033212 010205 MOV R2,R5 ;YES--SAVE IT
6208 033214 005201 INC R1 ;MOVE TO NEXT CHARACTER
6209 033216 004037 034712 JSR R0,@#CK.OCT ;OCTAL DIGIT
6210 033222 000405 BR 6$ ;NO--BRANCH
6211 033224 005201 INC R1 ;MOVE TO NEXT CHARACTER
6212 033226 006305 ASL R5 ;SCALE HIGH DIGIT
6213 033230 006305 ASL R5
6214 033232 006305 ASL R5
6215 033234 060502 ADD R5,R2 ;COMBINE HIGH & LOW DIGITS
6216 033236 020227 000022 6$: CMP R2,#$TN-1 ;VALID TEST NUMBER?
6217 033242 003263 BGT GTTST1 ;NO--BRANCH
6218 033244 010237 033436 MOV R2,9$ ;SAVE THE TEST NUMBER
6219 033250 010204 MOV R2,R4 ;CONVERT TEST NUMBER INTO AN INDEX
6220 033252 042704 000017 BIC #17,R4 ;CLEAR UNWANTED BITS
6221 033256 006204 ASR R4 ;SHIFT THE BITS
6222 033260 006204 ASR R4 ;SHIFT THE BITS
6223 033262 006204 ASR R4 ;SHIFT THE BITS
6224 033264 006302 ASL R2
6225 033266 056264 001424 001234 BIS BITS(R2),TSTNMS(R4) ;SELECT TEST
6226 033274 121127 000055 CMPB (R1),#'- ;TEST STRING?
6227 033300 001060 BNE GTTST4 ;NO--BRANCH
6228 033302 005201 INC R1 ;YES--MOVE TO NEXT CHARACTER
6229 033304 004037 034712 JSR R0,@#CK.OCT ;OCTAL DIGIT?
6230 033310 000640 BR GTTST1 ;NO--BRANCH
6231 033312 010205 MOV R2,R5 ;YES--SAVE IT
6232 033314 005201 INC R1 ;MOVE TO NEXT CHARACTER
6233 033316 004037 034712 JSR R0,@#CK.OCT ;OCTAL DIGIT?
6234 033322 000405 BR 7$ ;NO--BRANCH
6235 033324 005201 INC R1 ;YES--MOVE TO NEXT CHARACTER
6236 033326 006305 ASL R5 ;SCALE HIGH DIGIT
6237 033330 006305 ASL R5
6238 033332 006305 ASL R5
6239 033334 060502 ADD R5,R2 ;COMBINE HIGH & LOW DIGIT
6240 033336 020227 000022 7$: CMP R2,#$TN-1 ;VALID TEST NUMBER?
6241 033342 003223 BGT GTTST1 ;NO--BRANCH
6242 033344 023702 033436 CMP 9$,R2 ;IS THE FIRST NUMBER OF THE
;STRING SMALLER THAN THE LAST?
6243 033350 002220 BGE GTTST1 ;NO--BRANCH
6244 033352 010240 MOV R2,-(SP) ;SAVE ENDING TEST NUMBER
6245 033354 013702 033436 MOV 9$,R2 ;GET STARTING TEST NUMBER
6246 033360 012637 033436 MOV (SP)+,9$ ;STORE ENDING TEST NUMBER
6247 033364 006337 033436 ASL 9$ ;SHIFT ENDING TEST NUMBER
6248 033370 006302 ASL R2 ;SHIFT TEST NUMBER
6249 033372 010204 8$: MOV R2,R4 ;COPY TEST NUMBER INTO R4
6250 033374 042704 000037 BIC #37,R4 ;CLEAR LOWER BITS
6251 033400 006204 ASR R4 ;SHIFT THE TEST NUMBER
6252 033402 006204 ASR R4 ;SHIFT THE TEST NUMBER
6253 033404 006204 ASR R4 ;SHIFT THE TEST NUMBER
6254 033406 006204 ASR R4 ;SHIFT THE TEST NUMBER
6255 033410 056264 001424 001234 BIS BITS(R2),TSTNMS(R4) ;SELECT THE TEST
6256 033416 062702 000002 ADD #2,R2 ;INCREMENT THE TEST NUMBER
6257 033422 020237 033436 CMP R2,9$ ;SEE IF FINISHED
6258 033426 101761 BLOS 8$ ;BR IF NOT
6259 033430 162702 000002 SUB #2,R2 ;CORRECT TEST NUMBER
6260 033434 000402 BR GTTST4 ;CONTINUE

```



```

6262 033436 000000
6263 033440 005201
6264 033442 121127 000056
6265 033446 001441
6266 033450 005737 001234
6267 033454 001005
6268 033456 005737 001236
6269 033462 001002
6270 033464 000137 033012
6271 033470 121127 000057
6272 033474 001004
6273 033476 056264 001424 001240
6274 033504 000405
6275 033506 121127 000054
6276 033512 001402
6277 033514 000137 033012
6278 033520 005201
6279 033522 105711
6280 033524 001402
6281 033526 000137 033076
6282 033532 005737 001240
6283 033536 001042
6284 033540 005737 001242
6285 033544 001037
6286 033546 000137 033012
6287 033552 005201
6288 033554 121127 000056
6289 033560 001414
6290 033562 105711
6291 033564 001402
6292 033566 000137 033012
6293 033572 005737 001240
6294 033576 001022
6295 033600 005737 001242
6296 033604 001017
6297 033606 000137 033006
6298 033612 005201
6299 033614 105711
6300 033616 001402
6301 033620 000137 033012
6302 033624 005737 001240
6303 033630 001005
6304 033632 005737 001242
6305 033636 001002
6306 033640 000137 033006
6307
6308
6309
6310 033644 104412
6311 033646 005027
6312 033650 000000
6313 033652 000411
6314 033654 005237 033650
6315 033660 022737 000022 033650
6316 033666 002003
6317 033670 104413

```

```

9$: .WORD 0 ;STORE TEST NUMBER HERE
GTTST3: INC R1 ;MOVE TO NEXT CHARACTER
GTTST4: CMPB (R1),#' ;"PERIOD"?
;YES--BRANCH
BEQ GTTST5 ;ANY TEST SELECTED THIS CYCLE?
TST TSTNMS ;BR IF YES
BNE 1$ ;ANY TEST SELECTED THIS CYCLE ?
TST TSTNMS+2 ;BR IF YES
BNE 1$ ;NO
JMP GTTST1 ;"OPEN"?
1$: CMPB (R1),#'/ ;NO--BRANCH
BNE 2$ ;YES--SET BITS FOR TEST TO OPEN
BIS BITS(R2), OPNFLG(R4)
BR 3$
2$: CMPB (R1),#', ;"COMMA"?
BEQ 3$ ;BR IF YES
JMP GTTST1 ;NO
3$: INC R1 ;MOVE TO NEXT CHARACTER
TSTB (R1) ;"CR"?
BEQ 4$ ;BR IF 'CR'
JMP GTTST2 ;NO--GO GET NEXT CHARACTER
4$: TST OPNFLG ;ANY TESTS TO OPEN ?
BNE OPNTST ;BR IF YES
TST OPNFLG+2 ;ANY TESTS TO OPEN ?
BNE OPNTST ;BR IF YES
JMP GTTST1 ;NO--START AGAIN
GTTST5: INC R1 ;MOVE TO NEXT CHARACTER
CMPB (R1),#' ;"PERIOD"?
BEQ GTTST6 ;YES--BRANCH
TSTB (R1) ;"CR"?
BEQ 1$ ;YES--BRANCH
JMP GTTST1 ;ANY TESTS TO OPEN ?
1$: TST OPNFLG ;BR IF YES
BNE OPNTST ;ANY TESTS TO OPEN ?
TST OPNFLG+2 ;BR IF YES
BNE OPNTST ;NO--GO START TESTING
JMP GT.PR2 ;MOVE TO NEXT CHARACTER
GTTST6: INC R1 ;"CR"?
TSTB (R1) ;YES--BRANCH
BEQ 1$ ;NO--GO ASK FOR TEST
JMP GTTST1 ;ANY TESTS TO OPEN ?
1$: TST OPNFLG ;BR IF YES
BNE OPNTST ;ANY TESTS TO OPEN ?
TST OPNFLG+2 ;BR IF YES
BNE OPNTST ;NO--GO START TESTING
JMP GT.PR2
;OPEN THE SELECTED TEST FOR CHANGES
OPNTST: SAVREG ;SAVE R0 - R5
CLR (PC)+ ;START WITH TEST 0
OPN.CT: .WORD 0 ;COUNT STORED HERE
BR OPN.2 ;SKIP THE INCREMENT
OPN.1: INC OPN.CT ;MOVE TO THE NEXT TEST
CMP #STN-1,OPN.CT ;TEST NUMBER TOO BIG?
BGE OPN.2 ;NO--OPEN THE NEXT TEST
JMP RESREG ;RESTORE R0 - R5

```

```

6318 033672 000137 033012      JMP      GTTST1          ;YES--GO ASK FOR MORE TESTS
6319 033676 013705 033650      OPN.2:  MOV      OPN.CT,R5 ;SETUP TO USE THE
6320 033702 006305                ASL      R5             ;TEST NUMBER AS AN INDEX
6321 033704 013703 033650      MOV      OPN.CT,R3     ;GET INDEX
6322 033710 042703 000017      BIC      #17,R3        ;CLEAR LOWER TEST BITS
6323 033714 006203                ASR      R3             ;SHIFT TEST NUMBER
6324 033716 006203                ASR      R3             ;SHIFT TEST NUMBER
6325 033720 006203                ASR      R3             ;SHIFT TEST NUMBER
6326 033722 036563 001424 001240  BIT      BITS(R5),OPNFLG(R3) ;OPEN THIS TEST?
6327 033730 001751                BEQ      OPN.1         ;NO--MOVE TO NEXT TEST
6328 033732 104401 033740      TYPE    ,65$          ;TYPE ASCIZ STRING
6329 033736 000404                BR       64$           ;GET OVER THE ASCIZ
6330
6331 033750                ;:65$: .ASCIZ / TEST /
6332 033750 013746 033650      ;64$:  MOV      OPN.CT,-(SP) ;SAVE OPN.CT FOR TYPEOUT
6333
6334 033754 104403                TYPOS   ;TEST NUMBER
6335 033756 002                .BYTE  2              ;GO TYPE--OCTAL ASCII
6336 033757 000                .BYTE  0              ;TYPE 2 DIGIT(S)
6337 033760 104401 001215      TYPE    ,SCRLF        ;SUPPRESS LEADING ZEROS
6338 033764 016500 001536      MOV      PRMPT(R5),RO  ;TYPE "CR" & "LF"
6339 033770 011046                MOV      (RO),-(SP)   ;PICKUP PARAMETER POINTER
6340 033772 012702 001504      MOV      #PRM,R2      ;SAVE THE VARIABLE INDICATOR
6341 033776 000405                BR       2$           ;FIRST ADDRESS OF TABLE
6342 034000 006216      1$:    ASR      (SP)     ;CHECK FOR A VARIABLE
6343 034002 103403                BCS     2$           ;GO MOVE THIS ONE
6344 034004 001404                BEQ     OPNPRM       ;DONE
6345 034006 005722                TST    (R2)+         ;BUMP THE POINTER
6346 034010 000773                BR     1$           ;
6347 034012 012022      2$:    MOV      (RO)+,(R2)+ ;MOVE THIS VARIABLE INTO THE
6348 034014 000771                BR     1$           ;COMMON AREA
6349 034016 013716 001504      OPNPRM: MOV     @#PRM,(SP) ;GET THE VARIABLE INDICATOR
6350 034022 005004                CLR    R4            ;ZERO THE INDEX
6351 034024 006216      1$:    ASR      (SP)     ;CHECK FOR A VARIABLE
6352 034026 103403                BCS     3$           ;GO GET IT
6353 034030 001772                BEQ     OPNPRM       ;OUT OF VARIABLES
6354 034032 005724      2$:    TST    (R4)+         ;UPDATE THE INDEX
6355 034034 000773                BR     1$           ;
6356 034036 005764 001606      3$:    TST    PRMLMT(R4)  ;IS THE MAX. MAGNITUDE NEG?
6357 034042 100466                BMI    OPNPAT        ;YES--THEN IT IS THE PATTERN
6358 034044 104401 044657      TYPE    ,MSG.SP      ;TYPE SPACES
6359 034050 016437 001636 034060  MOV     PRMMSG(R4),4$ ;TYPE THE NAME OF THIS VARIABLE
6360 034056 104401
6361 034060 000000      4$:    .WORD  0
6362 034062 104401 043702      TYPE    ,MSG.EQ     ;TYPE "="
6363 034066 016446 001506      MOV     RPT(R4),-(SP) ;PUT RPT(P4) ON THE STACK
6364 034072 004737 024220      JSR     PC,@#SSB2D   ;CHANGE RPT(R4) TO DECIMAL ASCIZ
6365 034076 004737 024450      JSR     PC,@#SSUPRS  ;TYPE WITHOUT LEADING ZEROS
6366 034102 104401 043730      TYPE    ,SLASH     ;' / '
6367 034106 104411
6368 034110 012601                RDLIN  (SP)+,R1      ;READ AN ASCIZ STRING
6369 034112 004037 034766      JSR     RO,@#CK.CHR  ;CHECK ONE CHARACTER
6370 034116 034036                3$          ;ILLEGAL CHARACTER
6371 034120 034032                2$          ;CARRIAGE RETURN
6372 034122 034170                8$          ;"'"
6373 034124 034132                5$          ;" ,

```

6374	034126	034140		6S				
6375	034130	034166		7S				: DIGIT 0-9
6376	034132	105711		5S:	TSTB	(R1)		: "CR"?
6377	034134	001340			BNE	3S		: NO--STAY ON THIS VARIABLE
6378	034136	000735			BR	2S		: YES--MOVE TO NEXT VARIABLE
6379	034140	105711		6S:	TSTB	(R1)		: IS THERE A "CR" AFTER THE PERIOD?
6380	034142	001002			BNE	64S		: NO
6381	034144	000137	034560		JMP	OPN.N2		: YES--GO CLOSE THIS TEST
6382	034150	122721	000056	64S:	CMPB	#'.,(R1)+		: DOUBLE PERIOD?
6383	034154	001330			BNE	3S		: NO--GO ASK FOR THIS VARIABLE
6384	034156	105711			TSTB	(R1)		: YES--IS A "CR" AFTER THE DOUBLE PERIOD?
6385	034160	001326			BNE	3S		: NO--ASK FOR THIS VARIABLE AGAIN
6386	034162	000137	034576		JMP	OPN.X2		: YES--CLOSE ALL TEST
6387	034166	005301		7S:	DEC	R1		: BACK THE POINTER UP BY ONE
6388	034170			8S:				
6389	034170	016402	001606		MOV	PRMLT(R4),R2		: UPPER LIMIT OF INPUT
6390	034174	004037	035042		JSR	RO,@#CK.DIG		: CHECK THE DIGIT(S)
6391	034200	034036			3S			: ILLEGAL INPUT
6392	034202	034036			3S			: INPUT TOO LARGE
6393	034204	034212			9S			: TERMINATED WITH A ",'" OR "CR"
6394	034206	034554			OPN.N1			: TERMINATED WITH A ":'"
6395	034210	034572			OPN.X1			: TERMINATED WITH A ":'."
6396	034212	010264	001506	9S:	MOV	R2,RPT(R4)		: SAVE THIS VARIABLE
6397	034216	000705			BR	2S		: MOVE TO NEXT VARIABLE
6398	034220	104401	044657	OPNPAT:	TYPE	,MSG.SP		: TYPE SPACES
6399	034224	104401	043676		TYPE	,MSG.PAT		: TYPE "PAT"
6400	034230	104401	043702		TYPE	,MSG.EQ		: TYPE "="
6401	034234	016446	001506		MOV	RPT(R4),-(SP)		: SAVE RPT(R4) FOR TYPEOUT
6402	034240	104402			TYPOC			: GO TYPE--OCTAL ASCII(ALL DIGITS)
6403	034242	104401	044660		TYPE	,MSG.SP+1		: TYPE ONE SPACE
6404	034246	104411			RDLIN			: READ ASCII STRING
6405	034250	012601			MOV	(SP)+,R1		: PICKUP POINTER
6406	034252	004037	034766		JSR	RO,@#CK.CHR		: CHECK ONE CHARACTER
6407	034256	034220			OPNPAT			: ILLEGAL CHARACTER
6408	034260	034016			OPNPRM			: CARRIAGE RETURN
6409	034262	034314			3S			: "/"
6410	034264	034016			OPNPRM			: ":'"
6411	034266	034272			1S			: ":'"
6412	034270	034312			2S			: DIGIT 0-9
6413	034272	105711		1S:	TSTB	(R1)		: "CR" AFTER THE PERIOD?
6414	034274	001531			BEQ	OPN.N2		: YES--GO CLOSE THIS TEST
6415	034276	122721	000056		CMPB	#'.,(R1)+		: NO--PERIOD?
6416	034302	001346			BNE	OPNPAT		: NO--LOOP
6417	034304	105711			TSTB	(R1)		: "CR" AFTER A DOUBLE PERIOD?
6418	034306	001533			BEQ	OPN.X2		: YES--GO START TESTING
6419	034310	000743			BR	OPNPAT		: NO--LOOP
6420	034312	005301		2S:	DEC	R1		: BACKUP THE ASCII POINTER
6421	034314			3S:				
6422	034314	004037	035226		JSR	RO,@#CK.NUM		: CHECK THE NUMBER
6423	034320	034220			OPNPAT			: ILLEGAL INPUT
6424	034322	034330			4S			: TERMINATED WITH A ",'" OR "CR"
6425	034324	034554			OPN.N1			: TERMINATED WITH A ":'"
6426	034326	034572			OPN.X1			: TERMINATED WITH A ":'."
6427	034330	010264	001506	4S:	MOV	R2,RPT(R4)		: SAVE THE INPUT NUMBER
6428	034334	006002			ROR	R2		: OPEN PATTERN 0?
6429	034336	103227			BCC	OPNPRM		: NO--START AT BEGINNING OF PARAMETER TABLE

6430	034340	104412			SAVREG			;SAVE R0 - R5
6431	034342	005000			OPNWD5: CLR	R0		;START WITH WORD 0
6432	034344	012704	003104		MOV	#PATO,R4		
6433	034350				1\$:			
6434	034350	104401	034356		TYPE	65\$		::TYPE ASCIZ STRING
6435	034354	000403			ER	64\$		::GET OVER THE ASCIZ
6436					::65\$:	.ASCIZ	/ WD/	
6437	034364				64\$:			
6438	034364	010046			MOV	R0,-(SP)		;PUT R0 ON THE STACK
6439	034366	004737	024220		JSR	PC,@#SSB2D		;CHANGE R0 TO DECIMAL ASCIZ
6440	034372	004737	024450		JSR	PC,@#SSUPRS		;TYPE WITHOUT LEADING ZEROS
6441	034376	104401	043702		TYPE	MSG.EQ		;TYPE "="
6442	034402	011446			MOV	(R4),-(SP)		;SAVE (R4) FOR TYPEOUT
6443	034404	104402			TYP0C			;GO TYPE--OCTAL ASCII(ALL DIGITS)
6444	034406	104411			RDLIN			;READ ASCIZ STRING
6445	034410	012601			MOV	(SP)+,R1		;PICKUP THE POINTER
6446	034412	004037	034766		JSR	R0,@#CK.CHR		;CHECK ONE CHARACTER
6447	034416	034350			1\$			;ILLEGAL CHARACTER
6448	034420	034452			4\$			;CARRIAGE RETURN
6449	034422	034434			2\$			"/"
6450	034424	034452			4\$			"/"
6451	034426	034466			5\$			"/"
6452	034430	034432			10\$			;DIGIT 0-9
6453	034432	005301			10\$:	DEC	R1	;BACKUP THE ASCII POINTER
6454	034434				2\$:			
6455	034434	004037	035226		JSR	R0,@#CK.NUM		;CHECK THE NUMBER
6456	034440	034350			1\$			;ILLEGAL INPUT
6457	034442	034450			3\$			;TERMINATED WITH A "," OR "CR"
6458	034444	034506			6\$			;TERMINATED WITH A "."
6459	034446	034520			8\$			;TERMINATED WITH A "..."
6460	034450	010214			3\$:	MOV	R2,(R4)	;SAVE THE INPUT
6461	034452	005724			4\$:	TST	(R4)+	;MOVE TO NEXT WORD
6462	034454	005200			INC	R0		;INCREMENT THE COUNT
6463	034456	022700	000020		CMP	#16.,R0		;COUNT TO LARGE?
6464	034462	003332			BGT	1\$		;NO--BRANCH
6465	034464	000726			BR	OPNWD5		;YES--BRANCH
6466	034466	105711			5\$:	TSTB	(R1)	; "CR" AFTER THE PERIOD?
6467	034470	001407			BEQ	7\$		;YES--GO CLOSE THIS TEST
6468	034472	122721	000056		CMPB	#'.,(R1)+		;NO--PERIOD?
6469	034476	001324			BNE	1\$		;NO--BRANCH ILLEGAL INPUT STRING
6470	034500	105711			TSTB	(R1)		; "CR" AFTER THE "PERIOD-PERIOD"?
6471	034502	001407			BEQ	9\$		;YES--GO START TESTING
6472	034504	000721			BR	1\$		;NO--LOOP
6473	034506	010224			6\$:	MOV	R2,(R4)+	;SAVE THE INPUT
6474	034510	004737	034532		7\$:	JSR	PC,@#CLSWDS	;CLOSE THE DATA PATTERN
6475	034514	104413			RESREG			;RESTORE R0 - R5
6476	034516	000420			BR	OPN.N2		;MOVE TO NEXT TEST
6477	034520	010224			8\$:	MOV	R2,(R4)+	;SAVE THE INPUT
6478	034522	004737	034532		9\$:	JSR	PC,@#CLSWDS	;CLOSE THE DATA PATTERN
6479	034526	104413			RESREG			;RESTORE R0 - R5
6480	034530	000422			BR	OPN.X2		;START TESTING
6481	034532	012701	003104		CLSWDS:	MOV	#PATO,R1	;FIRST ADDRESS OF DATA PATTERN
6482	034536	005200			1\$:	INC	R0	;COUNT THE LAST WORD THAT WAS STORED
6483	034540	022700	000017		CMP	#15.,R0		;END OF TABLE
6484	034544	002402			BLT	2\$		;YES--EXIT
6485	034546	012124			MOV	(R1)+,(R4)+		;COPY

```

6486 034550 000772
6487 034552 000207
6488 034554 010264 001506
6489 034556 005726
6490 034558 004737 034632
6491 034560 000137 033654
6492 034572 010264 001506
6493 034576 005726
6494 034600 004737 034632
6495 034604 005726
6496 034606 020527 000034
6497 034612 002403
6498 034614 104413
6499 034616 000137 033006
6500 034622 036503 001424
6501 034626 001364
6502 034630 000765
6503 034632 104412
6504 034634 012700 001504
6505 034640 016501 001536
6506 034644 012002
6507 034646 012103
6508 034650 012704 000001
6509 034654 030402
6510 034656 001403
6511 034660 030403
6512 034662 001404
6513 034664 011011
6514 034666 030403
6515 034670 001401
6516 034672 005721
6517 034674 005720
6518 034676 006304
6519 034700 032704 002000
6520 034704 001763
6521 034706 104413
6522 034710 000207
6523
6524
6525
6526
6527
6528
6529
6530
6531
6532 034712 121127 000060
6533 034716 103407
6534 034720 121127 000067
6535 034724 101004
6536 034726 111102
6537 034730 042702 177770
6538 034734 005720
6539 034736 000200
6540
6541
    
```

```

2$: BR 1$ ; LOOP
RTS PC ; RETURN
OPN.N1: MOV R2,RPT(R4) ; SAVE THIS VARIABLE
OPN.N2: TST (SP)+ ; CLEAN OFF THE STACK
JSR PC,CLOSE ; CLOSE THIS TEST
JMP OPN.1 ; GO OPEN THE NEXT TEST
OPN.X1: MOV R2,RPT(R4) ; SAVE THIS VARIABLE
OPN.X2: TST (SP)+ ; CLEAN OFF THE STACK
1$: JSR PC,CLOSE ; CLOSE THIS TEST
2$: TST (R5)+ ; UPDATE THE INDEX
CMP R5,#16*2 ; INDEX TO BIG?
BLT 3$ ; NO--BRANCH
RESREG ; RESTORE R0 - R5
JMP GT.PR2 ; GO TO EXIT
3$: BIT BITS(R5),R3 ; IS THIS TEST OPEN FOR CHANGE?
BNE 1$ ; YES--GO CLOSE IT
BR 2$ ; NO--MOVE TO NEXT TEST
CLOSE: SAVREG ; SAVE R0 - R5
MOV #PRM,R0 ; "FROM" ADDRESS
MOV PRMPT(R5),R1 ; "TO" ADDRESS
MOV (R0)+,R2 ; "FROM" INDICATOR
MOV (R1)+,R3 ; "TO" INDICATOR
MOV #1,R4 ; TEST BIT START A "RPT"
1$: BIT R4,R2 ; PARAMETER TO BE MOVED?
BEQ 2$ ; NO--BRANCH
BIT R4,R3 ; A PLACE TO PUT IT?
BEQ 3$ ; NO--BRANCH
MOV (R0),(R1) ; YES--MOVE "FROM" TO "TO"
2$: BIT R4,R3 ; "TO" PARAMETER?
BEQ 3$ ; NO--BRANCH
TST (R1)+ ; YES--UPDATE THE POINTER
3$: TST (R0)+ ; UPDATE FROM POINTER
ASL R4 ; POSITION THE TEST BIT
BIT #BIT10,R4 ; DONE?
BEQ 1$ ; NO--BRANCH
RESREG ; RESTORE R0 - R5
RTS PC ; RETURN
; THIS ROUTINE IS USED TO CHECK IF AN
; ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
CALL
MOV #ADR,R1 ; ADDRESS OF ASCII CHARACTER
JSR R0,#CK.OCT ; CHECK THE CHARACTER
RETURN1 ; CHARACTER IS NOT BETWEEN 0-7
RETURN2 ; CHARACTER IS IN R2 AS A
; OCTAL DIGIT
CK.OCT: CMPB (R1),#0 ; LESS THAN ZERO?
BLO 1$ ; YES -- BRANCH
CMPB (R1),#7 ; GREATER THAN SEVEN?
BHI 1$ ; YES -- BRANCH
MOVB (R1),R2 ; GET THE CHARACTER
BIC #1C7,R2 ; STRIP AWAY THE ASCII
TST (R0)+ ; ADJUST FOR RETURN
1$: RTS R0 ; RETURN
; THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
    
```

6542  
6543  
6544  
6545  
6546  
6547  
6548  
6549  
6550 034740 121127 000060  
6551 034744 103407  
6552 034746 121127 000071  
6553 034752 101004  
6554 034754 111102  
6555 034756 042702 000060  
6556 034762 005720  
6557 034764 000200  
6558  
6559  
6560  
6561  
6562  
6563  
6564  
6565  
6566  
6567  
6568  
6569  
6570  
6571  
6572 034766 105711  
6573 034770 001420  
6574 034772 121127 000057  
6575 034776 001414  
6576 035000 121127 000054  
6577 035004 001410  
6578 035006 121127 000056  
6579 035012 001404  
6580 035014 004037 034740  
6581 035020 000406  
6582 035022 005720  
6583 035024 005720  
6584 035026 005720  
6585 035030 005720  
6586 035032 005720  
6587 035034 005201  
6588 035036 011000  
6589 035040 000200  
6590  
6591  
6592  
6593  
6594  
6595  
6596  
6597

```

:AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
:CALL
:      MOV      #ADR,R1      ;ADDRESS OF ASCII CHARACTER
:      JSR      RD, @CK.DEC  ;CHECK THE CHARACTER
:      RETURN1  ;NOT BETWEEN 0 AND 9
:      RETURN2  ;BETWEEN 0 AND 9
:      ;R2 = DIGIT
    
```

```

CK.DEC: CMPB    (R1), #'0    ;LESS THAN ZERO?
        BLO    1$          ;YES -- BRANCH
        CMPB    (R1), #'9    ;GREATER THAN NINE?
        BHI    1$          ;YES -- BRANCH
        MOVB   (R1), R2     ;GET THE CHARACTER
        BIC    #'0, R2     ;STRIP AWAY THE ASCII
        TST    (R0)+       ;ADJUST FOR RETURN
1$:     RTS      RD         ;RETURN
    
```

:THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO  
:DETERMINE WHAT IT IS.

```

:CALL
:      MOV      #ADR,R1      ;ADDRESS OF ASCII CHARACTER
:      JSR      RD, @CK.CHR  ;CHECK CHARACTER
:      RETURN  ADR1         ;UNKNOWN CHARACTER
:      RETURN  ADR2         ;CARRIAGE RETURN * (R1)=ADR+1
:      RETURN  ADR3         ;SLASH * (R1)=ADR+1
:      RETURN  ADR4         ;COMMA * (R1)=ADR+1
:      RETURN  ADR5         ;PERIOD * (R1)=ADR+1
:      RETURN  ADR6         ;DIGIT BETWEEN 0 AND 9.
:      ;R2 = DIGIT * (R1)=ADR+1
    
```

```

CK.CHR: TSTB    (R1)        ;"CARRIAGE RETURN"?
        BEQ    4$          ;YES -- BRANCH
        CMPB    (R1), #'/'  ;"SLASH"?
        BEQ    3$          ;YES -- BRANCH
        CMPB    (R1), #','  ;"COMMA"?
        BEQ    2$          ;YES -- BRANCH
        CMPB    (R1), #'.'  ;"PERIOD"?
        BEQ    1$          ;YES -- BRANCH
        JSR      RD, @CK.DEC ;"DIGIT"?
        BR     5$          ;NO -- BRANCH
        TST    (R0)+       ;DIGIT BETWEEN 0-9
1$:     TST    (R0)+       ;PERIOD
2$:     TST    (R0)+       ;COMMA
3$:     TST    (R0)+       ;SLASH
4$:     TST    (R0)+       ;CARRIAGE RETURN
        INC    R1          ;MOVE POINTER TO NEXT CHARACTER
5$:     MOV    (R0), R0    ;UNKNOWN CHARACTER
        RTS      RD         ;RETURN
    
```

:THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL  
:CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.

```

:CALL
:      MOV      #ADR,R1      ;ADDRESS OF ASCII STRING
:      MOV      #NUM,R2     ;MAX. MAGNITUDE OF INPUT NUMBER
:      JSR      RD, @CK.DIG  ;CHECK DIGITS
:      RETURN  ADR1         ;ILLEGAL CHARACTER -- R2=?
    
```

6598			:	RETURN	ADR2	:	INPUT NUMBER TO LARGE -- R2=?	
6599			:	RETURN	ADR3	:	"COMMA" -- R2 = NUMBER	
6600			:	RETURN	ADR4	:	"PERIOD" -- R2 = NUMBER	
6601			:	RETURN	ADR5	:	"PERIOD-PERIOD" -- R2 = NUMBER	
6602			:					
6603	035042	010446		CK.DIG: MOV	R4,-(SP)	:	SAVE R4	
6604	035044	010346		MOV	R3,-(SP)	:	SAVE R3	
6605	035046	010246		MOV	R2,-(SP)	:	SAVE THE MAX. SIZE ON THE STACK	
6606	035050	005002		CLR	R2	:	START WITH 0	
6607	035052	005003		CLR	R3			
6608	035054	005004		CLR	R4			
6609	035056	004037	034766	JSR	RO,@#CK.CHR	:	CHECK ONE CHARACTER	
6610	035062	035212		B\$		:	ILLEGAL CHARACTER	
6611	035064	035212		B\$		:	CARRIAGE RETURN	
6612	035066	035212		B\$		:	"/	
6613	035070	035212		B\$		:	,"	
6614	035072	035212		B\$		:	,"	
6615	035074	035076		1\$		:	DIGIT 0-9	
6616	035076	006303	1\$:	ASL	R3	:	2	
6617	035100	010346		MOV	R3,-(SP)	:	SAVE #2	
6618	035102	006303		ASL	R3	:	4	
6619	035104	006303		ASL	R3	:	8	
6620	035106	062603		ADD	(SP)+,R3	:	(*8)+(*2)=*10.	
6621	035110	060203		ADD	R2,R3	:	UPDATE THE INPUT NUMBER	
6622	035112	004037	034766	JSR	RO,@#CK.CHR	:	CHECK ONE CHARACTER	
6623	035116	035212		B\$		:	ILLEGAL CHARACTER	
6624	035120	035132		9\$		:	CARRIAGE RETURN	
6625	035122	035212		8\$		:	"/	
6626	035124	035140		3\$		:	,"	
6627	035126	035136		2\$		:	,"	
6628	035130	035076		1\$		:	DIGIT 0-9	
6629	035132	005301	9\$:	DEC	R1	:	BACKUP THE CHARACTER POINTER	
6630	035134	000401		BR	3\$	:	CONTINUE	
6631	035136	005724		2\$:	(R4)+	:	"PERIOD"	
6632	035140	005724		3\$:	(R4)+	:	"COMMA" OR "CR"	
6633	035142	004037	034766	JSR	RO,@#CK.CHR	:	CHECK ONE CHARACTER	
6634	035146	035212		B\$		:	ILLEGAL CHARACTER	
6635	035150	035202		6\$		:	CARRIAGE RETURN	
6636	035152	035212		8\$		:	"/	
6637	035154	035212		8\$		:	,"	
6638	035156	035162		4\$		:	,"	
6639	035160	035172		5\$		:	DIGIT 0-9	
6640	035162	005724	4\$:	TST	(R4)+	:	"PERIOD-PERIOD"	
6641	035164	105711		TSTB	(R1)	:	"CR"?	
6642	035166	001405		BEG	6\$	:	YES--BRANCH	
6643	035170	000410		BR	8\$			
6644	035172	126127	177776 000054	5\$:	CMPB	-2(R1),#'	:	WAS CHARACTER BEFORE THE DIGIT A COMMA?
6645	035200	001004		BNE	8\$	:	NO--EXIT	
6646	035202	020316	6\$:	CMP	R3,(SP)	:	INPUT TO LARGE?	
6647	035204	101001		BHI	7\$	:	YES -- BRANCH	
6648	035206	060400		ADD	R4,RO	:	ADJUST RETURN ADDRESS	
6649	035210	005720	7\$:	TST	(R0)+			
6650	035212	010302	8\$:	MOV	R3,R2	:	NUMBER TO R2	
6651	035214	005726		TST	(SP)+	:	CLEAN MAX. SIZE OFF OF STACK	
6652	035216	012603		MOV	(SP)+,R3	:	RESTORE R3	
6653	035220	012604		MOV	(SP)+,R4	:	RESTORE R4	

```

6654 035222 011000      MOV      (R0),R0      ;GET RETURN ADDRESS
6655 035224 000200      RTS       RO         ;RETURN
6656
6657
6658
6659
6660
6661
6662
6663
6664
6665
6666
6667 035226 010346      CK.NUM: MOV      R3,-(SP) ;SAVE R3
6668 035230 005003      CLR      R3         ;START NUMBER AT ZERO
6669 035232 004037 034712  JSR      RO,@CK.OCT ;OCTAL DIGIT?
6670 035236 000440      BR       6$        ;NO--BRANCH
6671 035240 005201 1$:      INC      R1         ;MOVE TO NEXT CHARACTER
6672 035242 006303      ASL      R3         ;FOR THE OCTAL NUMBER IN R3
6673 035244 103735      BCS     6$        ;DON'T LET IT GET TO BIG
6674 035246 006303      ASL      R3
6675 035250 103433      BCS     6$
6676 035252 006303      ASL      R3
6677 035254 103431      BCS     6$
6678 035256 060203      ADD     R2,R3
6679 035260 004037 034712  JSR      RO,@CK.OCT ;IS THIS AN OCTAL DIGIT?
6680 035264 000401      BR       2$        ;NO--FIND OUT WHAT IT IS
6681 035266 010764      BR       1$        ;YES--MAKE IT PART OF THE NUMBER
6682 035270 010302 2$:      MOV     R3,R2     ;SAVE THE OCTAL NUMBER
6683 035272 005003      CLR     R3         ;START WITH ZERO INDEX
6684 035274 004037 034766  JSR      RO,@CK.CHR ;CHECK ONE CHARACTER
6685 035300 035340      6$      ;ILLEGAL CHARACTER
6686 035302 035330      5$      ;CARRIAGE RETURN
6687 035304 035340      6$      ;"/"
6688 035306 035330      5$      ;" "
6689 035310 035314      3$      ;"'"
6690 035312 035340      6$      ;DIGIT 0-9
6691 035314 005723 3$:      TST     (R3)+     ;"PERIOD"
6692 035316 121127 000056  CMPB    (R1),#'.   ;"PERIOD-PERIOD"?
6693 035322 001002      BNE     5$        ;NO--BRANCH
6694 035324 005201      INC     R1         ;YES--ADVANCE THE POINTER
6695 035326 005723 4$:      TST     (R3)+     ;"PERIOD-PERIOD"
6696 035330 005723 5$:      TST     (R3)+     ;"COMMA"
6697 035332 105711      TSTB   (R1)       ;"CR"?
6698 035334 001001      BNE     6$        ;NO--BRANCH
6699 035336 060300      ADD     R3,RO     ;YES--SAVE THE OCTAL NUMBER
6700 035340 012603 6$:      MOV     (SP)+,R3  ;RESTORE R3
6701 035342 011000      MOV     (R0),RO   ;PICKUP EXIT ADDRESS
6702 035344 000200      RTS       RO         ;RETURN

```



```

6703 ;*****
6704
6705
6706 .SBTTL SINGLE/DUAL PORT RH70/RMO3 DRIVER (REV 1.0)
6707
6708 ;COPYRIGHT (C) 1976
6709 ;DIGITAL EQUIPMENT CORP.
6710 ;MAYNARD, MA 01754
6711 ;AUTHOR(S): JIM LACEY/CHUCK HESS/C. CHEN
6712
6713 ;*****
6714
6715 ;STORAGE FOR RMO3, RMER1, RMER2, AND RMMR2 ON AN ERROR "2"
6716 ;RMERRS = RMO3
6717 ;RMERRS+2 = RMER1
6718 ;RMERRS+4 = RMER2
6719 ;RMERRS+6 = RMMR2
6720
6721 035346 000000 000000 000000 RMERRS: .WORD 0,0,0,0
6722 035354 000000
6723
6724 ;TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
6725 ;DRVACT=0 IF DRIVE IS IDLE
6726 ;DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
6727 ;DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION
6728
6729 035356 000 DRVACT: .BYTE 0 ;DRIVE 0
6730 035357 000 .BYTE 0 ;DRIVE 1
6731 035360 000 .BYTE 0 ;DRIVE 2
6732 035361 000 .BYTE 0 ;DRIVE 3
6733 035362 000 .BYTE 0 ;DRIVE 4
6734 035363 000 .BYTE 0 ;DRIVE 5
6735 035364 000 .BYTE 0 ;DRIVE 6
6736 035365 000 .BYTE 0 ;DRIVE 7
6737
6738 ;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
6739 ;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXISTENT
6740 ;DRVSTA>0 IF DRIVE IS ONLINE
6741 ;DRVSTA<0 IF DRIVE IS UNSAFE
6742
6743 035366 000 DRVSTA: .BYTE 0 ;DRIVE 0
6744 035367 000 .BYTE 0 ;DRIVE 1
6745 035370 000 .BYTE 0 ;DRIVE 2
6746 035371 000 .BYTE 0 ;DRIVE 3
6747 035372 000 .BYTE 0 ;DRIVE 4
6748 035373 000 .BYTE 0 ;DRIVE 5
6749 035374 000 .BYTE 0 ;DRIVE 6
6750 035375 000 .BYTE 0 ;DRIVE 7
6751
6752 ;TABLE OF DRIVE TYPES (DRV TYP=8 BYTES)
6753 ;DRV TYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
6754 ;DRV TYP=4 IF DRIVE IS RMO3
6755 ;DRV TYP=-1 IF NOT RMO3
6756
6757 035376 000 DRV TYP: .BYTE 0 ;DRIVE 0
6758 035377 000 .BYTE 0 ;DRIVE 1

```

# E13

MD-11-DZRMF-A RM03 EXTENDED DRIVE TEST  
 DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 128  
 SINGLE/DUAL PORT RH70/RM03 DRIVER (REV 1.0)

SEQ 0160

6759 035400 000  
 6760 035401 000  
 6761 035402 000  
 6762 035403 000  
 6763 035404 000  
 6764 035405 000

```
.BYTE 0 ;DRIVE 2
.BYTE 0 ;DRIVE 3
.BYTE 0 ;DRIVE 4
.BYTE 0 ;DRIVE 5
.BYTE 0 ;DRIVE 6
.BYTE 0 ;DRIVE 7
```

```
;TABLE OF DUAL PORT INITIALIZATION INDICATORS
;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
;DPINT<0 IF INITIALIZATION IS IN PROGRESS
```

6770 035406 000  
 6771 035407 000  
 6772 035410 000  
 6773 035411 000  
 6774 035412 000  
 6775 035413 000  
 6776 035414 000  
 6777 035415 000

```
DPINT: .BYTE 0 ;DRIVE 0
        .BYTE 0 ;DRIVE 1
        .BYTE 0 ;DRIVE 2
        .BYTE 0 ;DRIVE 3
        .BYTE 0 ;DRIVE 4
        .BYTE 0 ;DRIVE 5
        .BYTE 0 ;DRIVE 6
        .BYTE 0 ;DRIVE 7
```

```
;TABLE OF PENDING DUAL PORT REQUESTS
;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE
```

6783 035416 000  
 6784 035417 000  
 6785 035420 000  
 6786 035421 000  
 6787 035422 000  
 6788 035423 000  
 6789 035424 000  
 6790 035425 000

```
DPRQS: .BYTE 0 ;DRIVE 0
        .BYTE 0 ;DRIVE 1
        .BYTE 0 ;DRIVE 2
        .BYTE 0 ;DRIVE 3
        .BYTE 0 ;DRIVE 4
        .BYTE 0 ;DRIVE 5
        .BYTE 0 ;DRIVE 6
        .BYTE 0 ;DRIVE 7
```

```
;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
;"DPB" OF THE I/O OPERATION.
```

6796 035426 000000

```
TRNSWT: .WORD 0
```

```
;SEARCH WAIT KEYS (SRCHWT=1 WORD)
;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
```

6804 035430 000000

```
SRCHWT: .WORD 0
```

```
;RM03 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
;ACTDRV=0 IF DRIVER IS INACTIVE
;ACTDRV>0 IF DRIVER IS ACTIVE
```

6810 035432 000

```
ACTDRV: .BYTE 0
```

```
;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
```

6811  
 6812  
 6813  
 6814

```

6815
6816 035433 000
6817
6818
6819
6820
6821
6822
6823 035434 000
6824 035435 000
6825 035436 000
6826 035437 000
6827 035440 000
6828 035441 000
6829 035442 000
6830 035443 000
6831
6832
6833
6834
6835 035444 000
6836 035445 000
6837 035446 000
6838 035447 000
6839 035450 000
6840 035451 000
6841 035452 000
6842 035453 000
6843
6844
6845
6846
6847
6848
6849
6850 035454 000000
6851
6852
6853
6854
6855
6856
6857
6858 035456 177777
6859
6860
6861
6862
6863 035460 177777
6864 035462 177777
6865 035464 177777
6866 035466 177777
6867 035470 177777
6868 035472 177777
6869 035474 177777
6870 035476 177777
    
```

```

ACTSTR: .BYTE 0

;UNLOAD FLAG (ULDFLG=8 BYTES)
;ULDFLG=0 IF NO UNLOAD COMMAND
;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE

ULDFLG: .BYTE 0 ;DRIVE 0
        .BYTE 0 ;DRIVE 1
        .BYTE 0 ;DRIVE 2
        .BYTE 0 ;DRIVE 3
        .BYTE 0 ;DRIVE 4
        .BYTE 0 ;DRIVE 5
        .BYTE 0 ;DRIVE 6
        .BYTE 0 ;DRIVE 7

;LOOK AHEAD COUNT (LACNT=8 BYTES)
;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED

LACNT: .BYTE 0 ;DRIVE 0
        .BYTE 0 ;DRIVE 1
        .BYTE 0 ;DRIVE 2
        .BYTE 0 ;DRIVE 3
        .BYTE 0 ;DRIVE 4
        .BYTE 0 ;DRIVE 5
        .BYTE 0 ;DRIVE 6
        .BYTE 0 ;DRIVE 7

;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
;SAVEFG <0 IF SAVE THE RH70/RMO3 REGISTERS WHEN THE
;OPERATION IS COMPLETED AS PER (DPB+14).
;SAVEFG=0 IF SAVE THE RH70/RMO3 REGISTERS, AS PER
;(DPB+14), AFTER AN ERROR.

SAVEFG: .WORD 0

;SEEK FLAG (SEEKFG=1 WORD)
;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
;FOR A DATA TRANSFER START A SEARCH COMMAND
;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
;DISREGARD THE WINDOW

SEEKFG: .WORD -1

;TIMEOUT TABLE (TIMER=8 WORDS)
;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION

TIMER: .WORD -1 ;DRIVE 0
        .WORD -1 ;DRIVE 1
        .WORD -1 ;DRIVE 2
        .WORD -1 ;DRIVE 3
        .WORD -1 ;DRIVE 4
        .WORD -1 ;DRIVE 5
        .WORD -1 ;DRIVE 6
        .WORD -1 ;DRIVE 7
    
```

```

6871
6872 ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
6873 ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
6874 ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
6875
6876 035500 177777 DTUW: .WORD -1
6877
6878 ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
6879 ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
6880 ;ATTENTION BIT
6881
6882 035502 001 ATABIT: .BYTE 1 ;DRIVE 0
6883 035503 002 .BYTE 2 ;DRIVE 1
6884 035504 004 .BYTE 4 ;DRIVE 2
6885 035505 010 .BYTE 10 ;DRIVE 3
6886 035506 020 .BYTE 20 ;DRIVE 4
6887 035507 040 .BYTE 40 ;DRIVE 5
6888 035510 100 .BYTE 100 ;DRIVE 6
6889 035511 200 .BYTE 200 ;DRIVE 7
6890
6891 ;FSRMO3 TO RH70 "MASSBUS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE
6892 ;CALLING IT FATAL (MCPEMX=1 WORD)
6893
6894 035512 000003 MCPEMX: .WORD 3
6895
6896 ;STORAGE FOR RMA DR (THE FIRST ADDRESS (776700) OF THE RH70/RMO3),
6897 ;RMVEC (THE VECTOR ADDRESS (254)), AND RMVEC+2 (THE BR LEVEL (5)).
6898
6899 035514 176700 RMA DR: .WORD 176700
6900 035516 000254 000240 RMVEC: .WORD 254,5*32.
6901
6902 ;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)
6903
6904 035522 000004 MXLACT: .WORD 4
6905 ;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)
6906
6907 035524 001000 MXDLTA: .WORD 8.*64.
6908 ;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)
6909
6910 035526 000200 MNDLTA: .WORD 2*64.
6911 ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)
6912
6913 035530 000005 MXWNDW: .WORD 5
6914
6915 ;DEFINITIONS OF THE RH70/RMO3 ADDRESS INDEXES
6916
6917 000000 RMCS1=0 ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
6918 000002 RMWC=2 ;WORD COUNT REGISTER (NOT A DRIVE REG)
6919 000004 RMBA=4 ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
6920 000006 RMDA=6 ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
6921 000010 RMCS2=10 ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
6922 000012 RMD5=12 ;DRIVE STATUS REGISTER (DRIVE REG 01)
6923 000014 RMER1=14 ;ERROR REGISTER #1 (DRIVE REG. 02)
6924 000016 RMAS=16 ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
6925 000020 RMLA=20 ;LOOK AHEAD REGISTER (DRIVE REG. 07)
6926 000022 RMDB=22 ;DATA BUFFER REGISTER (NOT A DRIVE REG.)

```

H13

MD-11-DZRMF-A RMD3 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 131  
SINGLE/DUAL PORT RH70/RMD3 DRIVER (REV 1.0)

SEQ 0163

6927	000024	RMMR1=24	; MAINTAINABILITY REGISTER (DRIVE REG. 03)
6928	000026	RMDT=26	; DRIVE TYPE REGISTER (DRIVE REG. 06)
6929	000030	RMSN=30	; SERIAL NUMBER REGISTER (DRIVE REG. 10)
6930	000032	RMOF=32	; OFFSET REGISTER (DRIVE REG. 11)
6931	000034	RMDC=34	; DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
6932	000036	RMMR=36	; DUMMY ADDRESS REGISTER (DRIVE REG. 13)
6933	000040	RMMR2=40	; MAINTENANCE REGISTER #2
6934	000042	RMER2=42	; ERROR REGISTER #2 (DRIVE REG. 15)
6935	000044	RMEC1=44	; ECC POSITION REGISTER (DRIVE REG. 16)
6936	000046	RMEC2=46	; ECC PATTERN REGISTER (DRIVE REG. 17)

```

;RH70/RMD3 DRIVER INITIALIZATION CODE
;THIS ROUTINE WILL DETERMINE WHICH RMD3 DRIVES ARE
;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
;TO THE PROPER STATE FOR EACH DRIVE.
;NOTE: THIS ROUTINE CALLS DRVINT

```

;CALL

```

JSR PC,RMINIT
RETURN

```

;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED

6951	035532	104412	RMINIT: SAVREG	;SAVE R0 - R5
6952	035534	013746	MOV @#PS, -(SP)	;SAVE THE PRESENT PROCESSOR STATUS
6953	035540	012737	MOV #<5*32.>, @#PS	;CHANGE THE PRIORITY TO 5
6954	035546	004737	JSR PC, CLRQUE	;CLEAR ALL REQUEST QUEUES
6955	035552	012701	MOV #RMERRS, R1	;FIRST ADDRESS TO BE CLEARED
6956	035556	012702	MOV #SEEKFG, R2	;LAST ADDRESS TO BE CLEARED
6957	035562	005021	1\$: CLR (R1)+	;CLEAR
6958	035564	020102	CMP R1, R2	;ARE WE DONE?
6959	035566	103775	BLO 1\$	;BRANCH IF NO
6960	035570	012702	MOV #DTUW, R2	;LAST ADDRESS
6961	035574	012721	2\$: MOV #-1, (R1)+	;INITIALIZE
6962	035600	020102	CMP R1, R2	;DONE?
6963	035602	101774	BLOS 2\$	;LOOP IF NO
6964	035604	005037	CLR DRVSTA	;SET ALL DRIVES TO OFFLINE
6965	035610	005037	CLR DRVSTA+2	
6966	035614	005037	CLR DRVSTA+4	
6967	035620	005037	CLR DRVSTA+6	
6968	035624	013703	MOV RMVEC, R3	;SETUP THE RH70/RMD3 VECTOR
6969	035630	012723	MOV #ISR, (R3)+	
6970	035634	013713	MOV RMVEC+2, (R3)	
6971	035640	013704	MOV RMADR, R4	;FIRST ADDRESS OF RH70/RMD3
6972	035644	012764	MOV #BIT05, RMCS2(R4)	;MASSBUS INIT
6973	035652	005001	CLR R1	;START WITH DRIVE 0
6974	035654	004037	3\$: JSR R0, DRVINT	;INIT THE DRIVE
6975	035660	000401	BR 4\$	; 'DVA' NOT SET OR PARITY ERROR
6976	035662	000402	BR 5\$	;NORMAL RETURN
6977	035664	105061	4\$: CLR DRVSTA(R1)	;SET DRIVE STATUS TO OFFLINE
6978	035670	005201	5\$: INC R1	;GO TO NEXT DRIVE
6979	035672	042701	BIC #1C7, R1	;MASK OUT UNUSED BITS
6980	035676	001366	BNE 3\$	;BR IF MORE DRIVES TO GO
6981	035700	012701	MOV #7, R1	;START WITH DRIVE 7
6982	035704	005037	CLR @#PS	;CLEAR THE PROCESSOR STATUS

6983	035710	105761	035406	6\$:	TSTB	DPINT(R1)	;WAITING FOR DRIVE TO SWITCH PORTS ?
6984	035714	001405			BEQ	8\$	;BR NOT WAITING
6985	035716	004737	043002		JSR	PC,SET.IE	;SET INTERRUPT
6986	035722	105761	035406	7\$:	TSTB	DPINT(R1)	;DRIVE SWITCHED PORTS ?
6987	035726	001375			BNE	7\$	;BR IF NOT
6988	035730	005301		8\$:	DEC	R1	;GO TO THE NEXT DRIVE
6989	035732	100366			BPL	6\$	;CHECK NEXT DRIVE
6990	035734	012637	177776		MOV	(SP)+,2#PS	;RESTORE THE PROCESSOR STATUS
6991	035740	104413			RESREG		;RESTORE R0 - R5
6992	035742	000207			RTS	PC	;BYE-BYE
6993							
6994							
6995							
6996							
6997							
6998							
6999							
7000							
7001							
7002							
7003							
7004							
7005							
7006							
7007							
7008	035744	010546					
7009	035746	105061	035366				
7010	035752	105061	035376				
7011	035756	105061	035434				
7012	035762	010164	000010				
7013	035766	112764	000111	000000			
7014	035774	032764	010000	000010			
7015	036002	001403					
7016	036004	004737	043002				
7017	036010	000476					
7018	036012	105061	035366				
7019	036016	032764	004000	000000			
7020	036024	001750					
7021	036026	004037	042312				
7022	036032	000026					
7023	036034	036232					
7024	036036	012605					
7025	036040	112761	000004	035376			
7026	036046	022705	020024				
7027	036052	001407					
7028	036054	022705	024024				
7029	036060	001404					
7030	036062	112761	177777	035376			
7031	036070	000446					
7032	036072	012746	000121				
7033	036076	004037	042472				
7034	036102	000000					
7035	036104	036232					
7036	036106	012746	010000				
7037	036112	004037	042472				
7038	036116	000032					

  

```

;DRIVE INITIALIZATION ROUTINE
;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
;AN RMO3. 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      RMADR,R4        ;UNIBUS ADDRESS OF RH70/RMO3 (RMCS1)
JSR      RO,DRVINT       ;CALLED BY A JSR
RETURN1  ;ERROR OCCURRED (PARITY)
RETURN2  ;NORMAL RETURN

DRVINT:  MOV      R5,-(SP) ;SAVE R5
DULP:    CLRB     DRVSTA(R1) ;START DRIVE STATUS AS OFFLINE
          CLRB     DRVSTYP(R1) ;CLEAR THE DRIVE TYPE INDICATOR
          CLRB     ULDFLG(R1) ;CLEAR THE UNLOAD FLAG
          MOV      R1,RMCS2(R4) ;SELECT A DRIVE
          MOVB    #111,RMCS1(R4) ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
          BIT     #BIT12,RMCS2(R4) ;NONEXISTENT DRIVE?
          BEQ     1$
          JSR     PC,SET.IE ;GO SET "IE" WITHOUT A "TRE"
          BR     6$ ;LEAVE THIS ROUTINE
1$:      CLRB     DRVSTA(R1) ;SET DRIVE STATUS TO OFFLINE
          BIT     #BIT11,RMCS1(R4) ;SEE IF DRIVE AVAILABLE
          BEQ     DULP ;BR IF DRIVE NOT AVAILABLE
          JSR     RO,RO.RM ;READ THE DRIVE TYPE REG.
          RMDT   8$
          MOV     (SP)+,R5 ;ERROR RETURN ADDRESS
          MOVB   #4,DRVSTYP(R1) ;PUT DRIVE TYPE IN R5
          MOVB   #20024,R5 ;SET RMO3 INDICATOR
          CMP    #24024,R5 ;SINGLE PORT RMO3 ?
          BEQ    2$ ;BR IF YES
          CMP    #24024,R5 ;DUAL PORT RMO3 ?
          BEQ    2$ ;BR IF YES
          MOVB   #-1,DRVSTYP(R1) ;SET INDICATOR TO 'OTHER'
          BR     6$ ;EXIT
2$:      MOV     #121,-(SP) ;DO A "READ-IN PRESET"
          JSR     RO,WRT.RM
          RMCS1 8$
          MOV     #BIT12,-(SP) ;SET FMT22=1
          JSR     RO,WRT.RM
          RMOF
    
```

```

7039 036120 036232          BS
7040 036122 004037 042312 JSR      RO,RO.RM      ;READ RMO3
7041 036126 000012          RMO3
7042 036130 036232          BS
7043 036132 012605          MOV      (SP)+,R5      ;AND SAVE IT IN R5
7044 036134 100015          BPL      4$            ;BRANCH IF ATA=0
7045 036136 116164 035502 000016 MOVB     ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
7046 036144 004037 042312 JSR      RO,RO.RM      ;FIND OUT WHY ATA=1
7047 036150 000014          RMER1
7048 036152 036232          BS
7049 036154 006126          ROL      (SP)+        ;IS IT UNSAFE?
7050 036156 100004          BPL      4$            ;BR IF NOT
7051 036160 112761 177777 035366 MOVB     #-1,DRVSTA(R1) ;SET UNSAFE INDICATOR
7052 036166 000407          BR       6$            ;EXIT
7053 036170 005105          4$:     COM      R5      ;CHECK MOL, DPR, DRY, AND VV
7054 036172 042705 167077 BIC      #↑C<BIT12!BIT08!BIT07!BIT06>,R5
7055 036176 001003          BNE     6$            ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
7056 036200 112761 000001 035366 MOVB     #1,DRVSTA(R1) ;SET DRIVE STATUS TO ONLINE
7057 036206 005720          6$:     TST      (RO)+    ;STEP OVER THE ERROR RETURN
7058 036210 000410          BR       8$            ;EXIT
7059 036212 006301          7$:     ASL      R1      ;CHANGE INDEX TO ADDRESS WORDS
7060 036214 012761 003720 035460 MOV      #2000.,TIMER(R1) ;START 2 SEC TIMER
7061 036222 006201          ASR      R1            ;RESTORE R1
7062 036224 112761 177777 035406 MOVB     #-1,DPINT(R1) ;SET PORT INITIALIZE INDICATOR
7063 036232 012605          8$:     MOV      (SP)+,R5 ;RESTORE R5
7064 036234 000200          RTS      RO            ;EXIT
7065
7066          ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
7067          ;CALL
7068          ;
7069          ;
7070          ;
7071          ;
7072          ;
7073          ;
7074          ;
7075          ;
7076 036236 013746 177776          RMO3:  MOV      @#PS,-(SP) ;SAVE THE CALLING STATUS
7077 036242 013737 035520 177776 MOV      RMVEC+2,@#PS ;DON'T ALLOW ANY RMO3 INTERRUPTS
7078 036250 112737 000001 035432 MOVB     #1,ACTDRV ;SET "ACTIVE DRIVER" FLAG
7079 036256 104412          SAVREG ;SAVE RO - R5
7080 036260 011002          MOV      (RO),R2      ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
7081 036262 005062 000016 CLR      16(R2)        ;CLEAR THE STATUS/ERROR INDICATOR
7082 036266 111201          MOVB     (R2),R1      ;PICKUP THE DRIVE NUMBER
7083 036270 013704 035514 MOV      RMADR,R4      ;UNIBUS ADDRESS OF RMO3
7084 036274 105761 035366 TSTB     DRVSTA(R1)   ;CHECK DRIVES STATUS
7085 036300 003014          BGT      1$            ;BRANCH IF ONLINE
7086 036302 105761 035434 TSTB     ULDFLG(R1)   ;UNLOAD COMMAND IN QUEUE?
7087 036306 001036          BNE     3$            ;BRANCH IF YES
7088 036310 105761 035406 TSTB     DPINT(R1)    ;TRYING TO INIT THE DRIVE
7089 036314 001042          BNE     5$            ;BR IF YES
7090 036316 004037 035744 JSR      RO,DRVINT     ;GO INIT. THE DRIVE
7091 036322 000434          BR       4$            ;ERROR RETURN
7092 036324 105761 035366 TSTB     DRVSTA(R1)   ;IS DRIVE STATUS ONLINE?
7093 036330 003445          BLE     6$            ;BR IF NOT
7094 036332 105761 035416          1$:     TSTB     DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?

```

```

7095 036336 001031      BNE      5$      ;BR IF YES
7096 036340 010164 000010  MOV      R1,RMCS2(R4) ;SELECT THE DRIVE
7097 036344 004037 043444  JSR      RO,DRVQUE ;PUT THIS REQUEST IN QUEUE
7098 036350 000460      BR       9$      ;QUEUE IS FULL
7099 036352 122762 000103 000002  CMPB    #103,2(R2) ;IS THIS REQ. FOR AN UNLOAD?
7100 036360 001003      BNE      2$      ;BR IF NO
7101 036362 112761 177777 035434  MOVB    #-1,ULDFLG(R1) ;SET THE "UNLOAD IN QUEUE" FLAG
7102 036370 105761 035356 2$:     TSTB    DRVACT(R1) ;IS THIS DRIVE ACTIVE?
7103 036374 001043      BNE      8$      ;BR IF .CS
7104 036376 004737 036530  JSR      PC,OPT ;CALL THE OPTIMIZER
7105 036402 000440      BR       8$
7106 036404 012762 120000 000016 3$:     MOV     #BIT15!BIT13,16(R2) ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
7107 036412 000434      BR       8$      ;EXIT
7108 036414 004737 037610 4$:     JSR      PC,C17 ;GO HANDLE THE PARITY ERROR
7109 036420 000431      BR       8$
7110 036422 004037 043444 5$:     JSR      RO,DRVQUE ;PUT REQUEST IN QUEUE
7111 036426 000431      BR       9$      ;QUEUE IS FULL
7112 036430 032714 000100  BIT     #BIT06,(R4) ;IE BIT SET?
7113 036434 001023      BNE      8$      ;YES
7114 036436 004737 043002  JSR      PC,SET.IE ;SET THE INTERRUPT
7115 036442 000420      BR       8$      ;RETURN
7116 036444 105761 035366 6$:     TSTB    DRVSTA(R1) ;SEE IF DRIVE OFFLINE OR UNSAFE
7117 036450 002412      BLT     7$      ;BR IF UNSAFE
7118 036452 012762 140000 000016  MOV     #BIT15!BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
7119 036460 105761 035376  TSTB    DRVTP(R1) ;SEE IF OFFLINE OR NONEXISTENT
7120 036464 001007      BNE      8$      ;BR IF OFFLINE
7121 036466 012762 100002 000016  MOV     #BIT15!BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
7122 036474 000403      BR       8$      ;GO TO EXIT
7123 036476 012762 110000 000016 7$:     MOV     #BIT15!BIT12,16(R2) ;DRIVE IS UNSAFE
7124 036504 104413 8$:     RESREG ;RESTORE R0 - R5
7125 036506 005720      TST     (R0)+ ;SETUP FOR NORMAL RETURN
7126 036510 000401      BR     10$     ;FINISH UP, THEN EXIT
7127 036512 104413 9$:     RESREG ;RESTORE R0 - R5
7128 036514 005720 10$:    TST     (R0)+ ;CORRECT THE RETURN ADDRESS
7129 036516 105037 035432  CLRB    ACTDRV ;CLEAR "ACTIVE DRIVER" FLAG
7130 036522 012637 177776  MOV     (SP)+,2#PS ;RETURN "PS" TO USER LEVEL
7131 036526 000200      RTS     R0 ;RETURN TO CALLER
7132
7133 ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
7134 ;CALL
7135 ;
7136 ;
7137 ;
7138 ;
7139 OPT:   SAVREG ;SAVE R0 - R5
7140 MOV     2#PS, -(SP) ;SAVE PROC. STATUS
7141 BICB    ATABIT(R1),SRCHWT ;CLEAR LA SEACH FLAG
7142 CLRB    DPRQS(R1) ;RESET THE PORT REQ FLAG ****
7143 JSR      PC,GETREQ ;GET "DPB" POINTER OF REQUEST
7144 TST     R2 ;IS THERE A REQUEST IN QUEUE?
7145 BEQ     7$ ;NO--BRANCH TO EXIT
7146 MOV     R1,RMCS2(R4) ;LOAD THE DRIVE ADDRESS *****
7147 MOV     #111,RMCS1(R4) ;CLEAR THE DRIVE
7148 BIT     #BIT11,RMCS1(R4) ;DVA SET?
7149 BEQ     5$ ;TO PROT REQUEST, IF NOT
7150 TSTB    DRVSTA(R1) ;IS DRIVE ONLINE?

```



7151	036606	003014			BGT	1\$		;YES--BRANCH
7152	036610	004737	043542		JSR	PC,POPQUE		;NO--REMOVE REQUEST FROM QUEUE
7153	036614	012762	140000	000016	MOV	#BIT15!BIT14,16(R2)		;SET OFFLINE STATUS/ERROR INDICATOR
7154	036622	105761	035366		TSTB	DRVSTA(R1)		;IS DRIVE UNSAFE ?
7155	036626	100053			BPL	8\$		;BR TO EXIT IF NOT
7156	036630	012762	110000	000016	MOV	#BIT15!BIT12,16(R2)		;SET UNSAFE STATUS/ERROR INDICATOR
7157	036636	000447			BR	8\$		;BRANCH TO EXIT
7158	036640				1\$:			
7159					MOV	#111,-(SP)		;LOAD COMMAND ONTO THE STACK
7160					JSR	RO,WRT.RM		;LOAD THE REGISTER
7161					RMCS1			;REGISTER INCREMENT
7162					6\$			;ERROR RETURN ADDRESS
7163					BIT	#BIT11,(R4)		;DRIVE AVAILABLE ?
7164					BEG	9\$		;BR IF NOT
7165	036640	122762	000150	000002	CMPB	#150,2(R2)		;IS THE REQUEST FOR I/O?
7166	036646	002403			BLT	2\$		;YES--BRANCH
7167	036650	004737	037174		JSR	PC,C14		;CALL THE COMMAND INITIATOR
7168	036654	000440			BR	8\$		;BRANCH TO EXIT
7169	036656	005737	035500		2\$:	TST	DTUW	;DATA TRANSFER UNDERWAY?
7170	036662	002012			BGE	4\$		;YES--GO START A SEARCH
7171	036664	005737	035456		TST	SEEKFG		;DO IMPLIED SEEKS?
7172	036670	100404			BMI	2\$		;YES---BRANCH
7173	036672	004037	040144		JSR	RO,LA		;NO--DO LOOK AHEAD
7174	036676	000427			BR	8\$		;RETURN HERE ON A PARITY ERROR
7175	036700	000403			BR	4\$		;GO START A SEARCH
7176	036702	004737	036766		3\$:	JSR	PC,C11	;START A DATA TRANSFER
7177	036706	000423			BR	8\$		
7178	036710	004737	037074		4\$:	JSR	PC,C13	;START A SEARCH
7179	036714	000420			BR	8\$		;GO TO THE EXIT
7180	036716	112761	177777	035416	5\$:	MOV	#-1,DPRQS(R1)	;SET PORT REQUEST INDICATOR
7181	036724	010103			MOV	R1,R3		;SET UP TO ADDRESS WORDS
7182	036726	006303			ASL	R3		;CONVERT TO WORD INDEX
7183	036730	012763	023420	035460	MOV	#10000.,TIMER(R3)		;START 10 SEC TIMER
7184	036736	000402			BR	7\$		;EXIT
7185	036740	004737	037610		6\$:	JSR	PC,C17	;PROCESS THE PARITY ERROR
7186	036744	032714	000100		7\$:	BIT	#BIT06,(R4)	;SEE IF 'IE' ALREADY SET
7187	036750	001002			BNE	8\$		;BR IF SET
7188	036752	004737	043002		JSR	PC,SET.IE		;SET "IE" WITHOUT A "TRE"
7189	036756	012637	177776		8\$:	MOV	(SP)+,2#PS	;RESTORE PROC. STATUS
7190	036762	104413			RESREG			;RESTORE RO - R5
7191	036764	000207			RTS	PC		
7192								
7193								;COMMAND INITIATOR
7194								
7195								;CALL
7196					MOV	#DRVNUM,R1		;DRIVE NUMBER
7197					MOV	#DPB,R2		;ADDRESS OF DPB
7198					JSR	PC,C1?		;C1?= C11,C13, OR C14
7199								;WHERE:
7200								;C11=DATA TRANSFER
7201								;C12=SEARCH REQUESTED BY DATA XFER
7202								;C14=NOT DATA TRANSFER
7203								
7204	036766	004737	043542		C11:	JSR	PC,POPQUE	;REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
7205	036772	010237	035426		MOV	R2,TRNSWT		;PUT REQ. IN TRANSFER WAIT QUEUE
7206	036776	010203			MOV	R2,R3		;DPB ADDRESS TO R3

7207	037000	013704	035514		MOV	RMAOR R4	; RMCS1 ADDRESS
7208	037004	010164	000010		MOV	R1, RMCS2(R4)	; SELECT DRIVE
7209	037010	062703	000004		ADD	#4, R3	; DESIRED WORD COUNT
7210	037014	062704	000002		ADD	#2, R4	; RMAC ADDRESS
7211	037020	012324			MOV	(R3)+, (R4)+	; LOAD WORD COUNT
7212	037022	012324			MOV	(R3)+, (R4)+	; LOAD BUFFER ADDRESS
7213	037024	012346			MOV	(R3)+, -(SP)	; LOAD SECTOR AND TRACK
7214	037026	004037	042472		JSR	RO, WRT.RM	; CALL THE LOAD(WRITE) ROUTINE
7215	037032	000006			RMDA		; INDEX OF REGISTER TO LOAD
7216	037034	037610			CI7		; ERROR RETURN ADDRESS
7217	037036	012346			MOV	(R3)+, -(SP)	; LOAD CYLINDER ADDRESS
7218	037040	004037	042472		JSR	RO, WRT.RM	
7219	037044	000034			RMDC		
7220	037046	037610			CI7		
7221	037050	016246	000002		MOV	2(R2), -(SP)	; LOAD "COMMAND+GO", "A17&A16", AND "PSEL"
7222	037054	004037	042472		JSR	RO, WRT.RM	
7223	037060	000000			RMCS1		
7224	037062	037610			CI7		
7225	037064	010137	035500		MOV	R1, DTUM	; SET "DATA TRANSFER UNDERWAY"
7226	037070	000137	037552		JMP	CI5	
7227	037074	013704	035514		MOV	RMAOR R4	; RMCS1 ADDRESS
7228	037100	010164	000010		MOV	R1, RMCS2(R4)	; SELECT DRIVE
7229	037104	016246	000012		MOV	12(R2), -(SP)	; DESIRED CYLINDER ADDRESS
7230	037110	004037	042472		JSR	RO, WRT.RM	
7231	037114	000034			RMDC		
7232	037116	037610			CI7		
7233	037120	116203	000010		MOV	10(R2), R3	; PICKUP SECTOR ADDRESS
7234					SUB	MXWINDOW, R3	; BACKUP BY MAX. SEARCH FOR I/O WINDOW
7235					BGE	1\$	
7236					ADD	#32, R3	
7237	037124	010346			MOV	R3, -(SP)	; COMBINE THE ADJUSTED SECTOR WITH
7238	037126	042716	177740		BIC	#177740, (SP)	; CHOP OF ALL EXEMTED BITS, IF ANY
7239	037132	116266	000011	000001	MOV	11(R2), 1(SP)	; THE DESIRED TRACK
7240	037140	004037	042472		JSR	RO, WRT.RM	; LOAD DESIRED TRACK & SECTOR
7241	037144	000006			RMDA		
7242	037146	037610			CI7		
7243	037150	012746	000131		MOV	#131, -(SP)	; START A SEARCH
7244	037154	004037	042472		JSR	RO, WRT.RM	
7245	037160	000000			RMCS1		
7246	037162	037610			CI7		
7247	037164	156137	035502	035430	BISB	ATABIT(R1), SRCHWT	; SET "SEARCH WAIT" KEY
7248	037172	000567			BR	CI5	
7249	037174	013704	035514		MOV	RMAOR R4	; RMCS1 ADDRESS
7250	037200	010164	000010		MOV	R1, RMCS2(R4)	; SELECT DRIVE
7251	037204	16203	000002		MOV	2(R2), R3	; PICKUP THE REQUESTED COMMAND
7252	037210	122703	000131		CMPB	#131, R3	; IS IT A SEARCH COMMAND?
7253	037214	001007			BNE	1\$	; BRANCH IF NO
7254	037216	016246	000010		MOV	10(R2), -(SP)	; LOAD DESIRED TRACK & SECTOR
7255	037222	004037	042472		JSR	RO, WRT.RM	
7256	037226	000006			RMDA		
7257	037230	037610			CI7		
7258	037232	000403			BR	2\$	; GO LOAD CYLINDER
7259	037234	122703	000105		CMPB	#105, R3	; IS IT A SEEK COMMAND
7260	037240	001007			BNE	3\$	; BRANCH IF NO
7261	037242	016246	000012		MOV	12(R2), -(SP)	; LOAD DESIRED CYLINDER
7262	037246	004037	042472		JSR	RO, WRT.RM	

7263	037252	000034			RMOC		
7264	037254	037610			CI7		
7265	037256	000546			BR	CI6	
7266	037260	122703	000115	3\$:	CMPB	#115,R3	; IS IT AN "OFFSET" COMMAND?
7267	037264	001013			BNE	4\$	; BR IF NO
7268	037266	004037	042312		JSR	RO,RO.RM	; MERGE THE OFFSET VALUE INTO RMOF
7269	037272	000032			RMOF		; BUT DON'T CHANGE THE UPPER
7270	037274	037610			CI7		
7271	037276	116216	000001		MOVB	1(R2),(SP)	; BYTE WHEN LOADING THE
7272	037302	004037	042472		JSR	RO,WRT.RM	; REGISTER (RMOF)
7273	037306	000032			RMOF		
7274	037310	037610			CI7		
7275	037312	000530			BR	CI6	; GO START THE COMMAND
7276	037314	122703	000107	4\$:	CMPB	#107,R3	; IS IT A "RECALIBRATE" COMMAND?
7277	037320	001525			BEQ	CI6	; BRANCH IF YES
7278	037322	122703	000117		CMPB	#117,R3	; IS IT A RETURN TO CENTER?
7279	037326	001522			BEQ	CI6	; BRANCH IF YES
7280	037330	122703	000103		CMPB	#103,R3	; IS IT AN "UNLOAD" COMMAND?
7281	037334	001016			BNE	5\$	; BRANCH IF NO
7282	037336	112761	000001	035356	MOVB	#1,DRVACT(R1)	; SET THE DRIVE ACTIVE INDICATOR
7283	037344	105061	035366		CLRB	DRVSTA(R1)	; PUT DRIVE STATUS TO OFFLINE
7284	037350	112761	000001	035434	MOVB	#1,ULDFLG(R1)	; SET "UNLOAD IN PROGRESS" FLAG
7285	037356	010346			MOV	R3,-(SP)	; START THE "UNLOAD" COMMAND
7286	037360	004037	042472		JSR	RO,WRT.RM	
7287	037364	000000			RMCS1		
7288	037366	037610			CI7		
7289	037370	000207			RTS	PC	; RETURN TO USER
7290	037372	122703	000143	5\$:	CMPB	#143,R3	; IS IT A "SET FORMAT" COMMAND?
7291	037376	001014			BNE	6\$	; BRANCH IF NO
7292	037400	004037	042312		JSR	RO,RO.RM	; READ THE OFFSET REGISTER
7293	037404	000032			RMOF		
7294	037406	037610			CI7		
7295	037410	116266	000001	000001	MOVB	1(R2),1(SP)	; COMBINE "FMT22" "ECI" AND "HCI"
7296	037416	004037	042472		JSR	RO,WRT.RM	; LOAD "FMT22", "ECI", AND/OR "HCI".
7297	037422	000032			RMOF		
7298	037424	037610			CI7		
7299	037426	000436			BR	12\$	
7300	037430	122703	000141	6\$:	CMPB	#141,R3	; IS IT A "GET REGISTER" COMMAND?
7301	037434	001023			BNE	10\$	; BRANCH IF NO
7302	037436	016203	000006	7\$:	MOV	6(R2),R3	; POINTS TO 1ST ADDRESS OF WHERE
7303							; TO PUT THE REGISTER(S)
7304	037442	116237	000010	037460	MOVB	10(R2),9\$	; INIT. THE INDEX FOR THE FIRST REG.
7305	037450	116205	000011		MOVB	11(R2),R5	; INDEX OF LAST REG. TO MOVE
7306	037454	004037	042312	8\$:	JSR	RO,RO.RM	; READ RH70/RMO3 REGISTER
7307	037460	000000		9\$:	RMCS1		; INDEX OF REG. TO READ
7308	037462	037610			CI7		
7309	037464	012623			MOV	(SP)+,(R3)+	; GET THE CONTENTS OF RH70//RMO3 REG.
7310	037466	023705	037460		CMP	9\$,R5	; LAST REG. BEEN READ?
7311	037472	001414			BEQ	12\$	; GET OUT IF YES
7312	037474	062737	000002	037460	ADD	#2,9\$	; INCREASE THE INDEX BY 2
7313	037502	000764			BR	8\$	; LOOP--MORE TO READ
7314	037504	122703	000145	10\$:	CMPB	#145,R3	; IS IT A "SELECT DRIVE" COMMAND?
7315	037510	001405			BEQ	12\$	; BRANCH IF YES
7316	037512	010346		11\$:	MOV	R3,-(SP)	; LOAD THE COMMAND
7317	037514	004037	042472		JSR	RO,WRT.RM	
7318	037520	000000			RMCS1		

7319	037522	037610				CI7		
7320	037524	004737	043542		12\$:	JSR	PC,POPQUE	; REMOVE REQ. FROM QUEUE
7321	037530	052762	000200	000016		BIS	#BIT07,16(R2)	; SET THE "DONE" BIT
7322	037536	005737	035454			TST	SAVEFG	; SAVE THE RH70/RMO3 REGISTERS?
7323	037542	100002				BPL	13\$	; BRANCH IF NO
7324	037544	004737	042664			JSR	PC,SVRH70	; YES--GO SAVE THE REGISTERS
7325	037550	000207			13\$:	RTS	PC	; RETURN TO USER
7326	037552	006301			CI5:	ASL	R1	
7327	037554	012761	001750	035460		MOV	#1000.,TIMER(R1)	; SET A ONE SECOND TIMER
7328	037562	006201				ASR	R1	
7329	037564	112761	000001	035356		MOV	#1,DRVACT(R1)	; SET THE DRIVE ACTIVE
7330	037572	000207				RTS	PC	; RETURN TO THE USER
7331	037574	010346			CI6:	MOV	R3,-(SP)	; LOAD THE COMMAND
7332	037576	004037	042472			JSR	RD,WRT.RM	
7333	037602	000000				RMCS1		
7334	037604	037610				CI7		
7335	037606	000761				BR	CI5	
7336	037610	032764	010000	000010	CI7:	BIT	#BIT12,RMCS2(R4)	; DRIVE NON-EXISTENT ?
7337	037616	001034				BNE	CI8	; BR IF YES
7338	037620	005702			1\$:	TST	R2	; ANYTHING IN QUEUE ?
7339	037622	001405				BEQ	CI7B	; BR IF NOT
7340	037624	012762	104000	000016		MOV	#BIT15:BIT11,16(R2)	; SET "PARITY" ERROR INDICATOR
7341	037632	004737	042664			JSR	PC,SVRH70	; GO SAVE THE RH70/RMO3 REGISTERS
7342	037636	012746	000111		CI7B:	MOV	#111,-(SP)	; DO A "DRIVE CLEAR"
7343	037642	004037	042472			JSR	RD,WRT.RM	
7344	037646	000000				RMCS1		
7345	037650	037710				CI8		
7346	037652	004737	043424			JSR	PC,EMPTYQ	; EMPTY THE QUEUE
7347	037656	105061	035434			CLRB	ULDFLG(R1)	; CLEAR THE UNLOAD IN QUEUE FLAG
7348	037662	105061	035356			CLRB	DRVACT(R1)	; DRIVE IS IDLE
7349	037666	020137	035500			CMP	R1,DTUM	; IF THIS DRIVE HAD AN I/O REQUEST
7350	037672	001005				BNE	1\$	; IN PROGRESS CLEAR ALL OF THE FLAGS
7351	037674	005037	035426			CLR	TRNSWT	
7352	037700	012737	177777	035500		MOV	#-1,DTUM	
7353	037706	000207			1\$:	RTS	PC	
7354	037710	04412			CI8:	SAVREG		; SAVE R0 - R5
7355	037712	032764	010000	000010		BIT	#BIT12,RMCS2(R4)	; IS 'NED' SET ?
7356	037720	001002				BNE	1\$	; BR IF YES
7357	037722	005001				CLR	R1	
7358	037724	005003				CLR	R3	
7359	037726	105761	035356		1\$:	TSTB	DRVACT(R1)	; DRIVE ACTIVE?
7360	037732	001443				BEQ	5\$	; BRANCH IF NO
7361	037734	013702	035426			MOV	TRNSWT,R2	; GET THE "TRANSFER WAIT" QUEUE
7362	037740	020137	035500			CMP	R1,DTUM	; DID THIS DRIVE HAVE AN I/O IN PROGRESS?
7363	037744	001402				BEQ	2\$	; BRANCH IF YES
7364	037746	004737	043520			JSR	PC,GETREQ	; GET THE DPB POINTER
7365	037752	005702			2\$:	TST	R2	; QUEUE ENTRY FOR DRIVE ?
7366	037754	001415				BEQ	4\$	; BR IF NOT
7367	037756	032764	010000	000010		BIT	#BIT12,RMCS2(R4)	; 'NED' SET ?
7368	037764	001404				BEQ	3\$	; BR IF NOT
7369	037766	012762	100002	000016		MOV	#BIT15:BIT01,16(R2)	; SET 'DRIVE NON-EXISTENT' INDICATOR
7370	037774	000405				BR	4\$	; CONTINUE
7371	037776	012762	102000	000016	3\$:	MOV	#BIT15:BIT10,16(R2)	; SET "NON-CLEARABLE PARITY" ERROR INDICATOR
7372	040004	004737	042664			JSR	PC,SVRH70	; SAVE RH70/RMO3 REGISTERS
7373	040010	012763	177777	035460	4\$:	MOV	#-1,TIMER(R3)	; STOP THE TIMER
7374	040016	105061	035356			CLRB	DRVACT(R1)	; SET "DRIVE ACTIVE" TO IDLE

```

7375 040022 020137 035500      CMP      R1,DTUM      ;IS THIS DRIVE SETUP FOR A TRANSFER
7376 040026 001005                BNE      5$           ;BR IF NOT
7377 040030 012737 177777 035500      MOV      #-1,DTUM     ;RESET THE INDICATOR
7378 040036 005037 035426                CLR      TRNSWT       ;CLEAR THE TRANSFER QUEUE
7379 040042 105061 035434                CLR      ULDFLG(R1)   ;CLEAR UNLOAD FLAG
7380 040046 032764 010000 000010 5$:      BIT      #BIT12,RMCS2(R4) ;'NED' SET ?
7381 040054 001021                BNE      6$           ;BR IF YES
7382 040056 005201                INC      R1           ;MOVE TO THE NEXT DRIVE
7383 040060 062703 000002                ADD      #2,R3
7384 040064 042701 177770                BIC      #1C7,R1
7385 040070 001316                BNE      1$           ;BRANCH IF MORE DRIVES
7386 040072 012737 177777 035500      MOV      #-1,DTUM     ;NO DATA TRANSFERS UNDERWAY
7387 040100 005037 035426                CLR      TRNSWT       ;CLEAR THE 'TRANSFER WAIT' QUEUE
7388 040104 004737 043346                JSR      PC,CLRQUE    ;CLEAR ALL OF THE REQUEST QUEUES
7389 040110 012764 000040 000010      MOV      #BIT05,RMCS2(R4) ;DO A MASSBUS INIT.
7390 040116 000406                BR       7$           ;CONTINUE
7391 040120 004737 043424                JSR      PC,EMPTYQ    ;CLEAR THE DRIVE'S QUEUE
7392 040124 105061 035366                CLR      DRVSTA(R1)   ;SET DRIVE TO OFFLINE
7393 040130 105061 035376                CLR      DRVTYP(R1)  ;CLEAR THE DRIVE TYPE INDICATOR
7394 040134 004737 043002                JSR      PC,SET.IE    ;SET "IE" WITHOUT "TRE"
7395 040140 104413                RESREG                ;RESTORE R0 - R5
7396 040142 000207                RTS      PC           ;RETURN
7397
7398 ;LOOK AHEAD ROUTINE
7399
7400 ;CALL
7401 ;
7402 ;
7403 ;
7404 ;
7405 ;
7406 ;
7407 ;
7408 040144 013704 035514      LA:      MOV      RMADR,R4      ;GET RMCS1'S ADDRESS
7409 040150 010164 000010      MOV      R1,RMCS2(R4) ;SELECT DRIVE
7410 040154 004037 042312      JSR      R0,RD.RM      ;READ DRIVE STATUS
7411 040160 000012                RMD5
7412 040162 040312                4$
7413 040164 042716 157577      BIC      #1C020200,(SP) ;ON CYLINDER ?
7414 040170 022726 000200      CMP      #200,(SP)+    ;PIP=0,DRY=1?
7415 040174 001044                BNE      3$           ;NO
7416 040176 105261 035444      INCB     LACNT(R1)     ;INCREMENT THE LOOK AHEAD COUNT
7417 040202 126137 035444 035522      CMPB    LACNT(R1),MXLACT ;EXCEED MAX?
7418 040210 003033                BGT      2$           ;BRANCH IF YES
7419 040212 116203 000010      MOV      10(R2),R3    ;GET DESIRED SECTOR ADDRESS AND
7420 040216 000303                SWAB    R3            ;MULT. BY 64--ALIGN WITH
7421 040220 006203                ASR     R3            ;LOOK AHEAD REGISTER
7422 040222 006203                ASR     R3
7423 040224 012737 000340 177776      MOV      #340,2#PS    ;PRIORITY LEVEL "7"
7424 040232 004037 042312      JSR      R0,RD.RM      ;READ LOOK AHEAD REGISTER
7425 040236 000020                RMLA
7426 040240 040312                4$
7427 040242 021664 000020      CMP      (SP),RMLA(R4) ;CORRECT LA NUMBER ?
7428 040246 001402                BEQ     7$           ;YES
7429 040250 005726                TST    (SP)+         ;NO,CLEAR STACK
7430 040252 000415                BR      3$

```

7431	040254	162603			7\$:	SUB	(SP)+,R3	;CALCULATE THE DELTA
7432	040256	002002				BGE	1\$	
7433	040260	062703	004000			ADD	#(32.*64.),R3	;MAKE THE DELTA POSITIVE
7434	040264	023703	035524		1\$:	CMP	MXDLTA,R3	;CHECK THE DELTA TO SEE
7435	040270	002406				BLT	3\$	;IF IT IS WITHIN THE
7436	040272	023703	035526			CMP	MNDLTA,R3	;WINDOW---IF YES, ZERO
7437	040276	002003				BGE	3\$	;THE LOOK AHEAD COUNT
7438	040300	105061	035444		2\$:	CLRB	LACNT(R1)	;AND TAKE THE I/O EXIT
7439	040304	005720				TST	(R0)+	
7440	040306	005720			3\$:	TST	(R0)+	;ADJUST THE RETURN ADDRESS
7441	040310	000402				BR	5\$	;EXIT
7442	040312	004737	037610		4\$:	JSR	PC,C17	;PROCESS THE ERROR
7443	040316	000200			5\$:	RTS	RO	;RETURN
7444								
7445								;INTERRUPT SERVICE ROUTINE
7446								
7447	040320	112737	000001	035432	ISR:	MOVB	#1,ACTDRV	;SET "ACTIVE DRIVER" FLAG
7448	040326	104412				SAVREG		;SAVE R0 - R5
7449	040330	013704	035514			MOV	RMADR,R4	;ADDRESS OF RHSCSI
7450	040334	013701	035500			MOV	DTUW,R1	;GET "DATA TRANSFER UNDERWAY" INDICATOR
7451	040340	002403				BLT	1\$	;BRANCH IF NO DATA TRANSFER UNDERWAY
7452	040342	004737	040364			JSR	PC,TD	;CALL TRANSFER DONE
7453	040346	000402				BR	2\$	;EXIT
7454	040350	004737	040534		1\$:	JSR	PC,SC	;CALL SPECIAL CONDITIONS
7455	040354	104413			2\$:	RESREG		;RESTORE R0 - R5
7456	040356	105037	035432			CLRB	ACTDRV	;CLEAR "ACTIVE DRIVER" FLAG
7457	040362	000002				RTI		;RETURN
7458								
7459								;TRANSFER DONE ROUTINE
7460								
7461	040364	105061	035356		TD:	CLRB	DRVACT(R1)	;SET DRIVE ACTIVE INDICATOR TO IDLE
7462	040370	012737	177777	035500		MOV	#-1,DTUW	;NO DATA TRANSFERS UNDERWAY
7463	040376	006301				ASL	R1	
7464	040400	012761	177777	035460		MOV	#-1,TIMER(R1)	;CANCEL TIMEOUT
7465	040406	006201				ASR	R1	
7466	040410	013702	035426			MOV	TRNSWT,R2	;GET "DPB" ADDRESS FROM THE
7467	040414	005037	035426			CLR	TRNSWT	;TRANSFER WAIT QUEUE--CLEAR QUEUE
7468	040420	052762	000200	000016		BIS	#BIT07,16(R2)	;SET DONE
7469	040426	010164	000010			MOV	R1,RMC52(R4)	;SELECT THE DRIVE
7470	040432	004037	042312			JSR	RO,RD.RM	;TRANSFER ERROR(TRE=1)?
7471	040436	000000				RMCS1		
7472	040440	037610				CI7		
7473	040442	006126				ROL	(SP)+	
7474	040444	100417				BMI	3\$	;BR IF YES
7475	040446	005737	035454			TST	SAVEFG	;SAVE THE RH70/RMO3 REGISTERS?
7476	040452	100002				BPL	1\$	;BRANCH IF NO
7477	040454	004737	042664			JSR	PC,SVRH70	;YES--SAVE THE REGISTERS
7478	040460				1\$:			
7479	040460	004737	043520			JSR	PC,GETREQ	;GET DPB POINTER
7480	040464	005702				TST	R2	;ENTRY FOR DRIVE ?
7481	040466	001403				BEQ	2\$	;BR IF NOT
7482	040470	004737	036530			JSR	PC,OPT	;CALL OPTIMIZER
7483	040474	000417				BR	SC	;CHECK OTHER DRIVES
7484								;THE RELEASE DRIVE COMMAND IS FORECD TO ENTER FOR DUAL PORT OPERATION
7485	040476	012714	000113		2\$:	MOV	#113,(R4)	;RELEASE THE DRIVE
7486	040502	000414				BR	SC	;CHECK FOR OTHER DRIVES

```

7487 040504 052762 100100 000016 3$: BIS #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
7488 040512 004737 043424 JSR PC,EMPTYQ ;EMPTY THE "DRIVE'S WAIT" QUEUE
7489 040516 004737 042664 JSR PC,SVRH70 ;SAVE THE RH70/RMD3 REGISTERS
7490 040522 012714 040111 MOV #40111,(R4) ;ISSUE A "DRIVE CLEAR"
7491 040526 012714 000113 MOV #113,(R4) ;ISSUE A RELEASE TO THE DRIVE
7492 040532 000400 BR SC ;CHECK FOR OTHER DRIVES
7493
7494
7495
7496 ;SPECIAL CONDITION ROUTINE
7497
7498 040534 116403 000016 SC: MOVB RMAS(R4),R3 ;READ "RMAS"
7499 040540 001014 BNE 2$ ;BRANCH IF ANY 'ATA' BITS SET
7500 040542 004037 042312 JSR RD,RD.RM ;READ CONTROL AND STATUS REGISTER
7501 040546 000000 RMCSI
7502 040550 037710 CIB
7503 040552 106126 ROLB (SP)+ ;IS "IE"=1?
7504 040554 100405 BMI 1$ ;YES, NO DRIVES TO CHECK
7505 040556 004037 043610 JSR RD,ES.SAV ;SAVE THE ADDRESS IN 'SESCAPE'
7506 040562 104001 ERROR 1 ;REPORT AN ILLEGAL INTERRUPT
7507 040564 004737 043002 JSR PC,SET.IE ;SET INTERRUPT ENABLE
7508 040570 000207 1$: RTS PC ;RETURN
7509 040572 005046 2$: CLR -(SP) ;PROCESS ALL DRIVES THAT HAVE
7510 040574 110316 MOVB R3,(SP) ;AN "ATA"=1
7511 040576 012703 000001 MOV #1,R3
7512 040602 005001 CLR R1
7513 040604 030316 SC3: BIT R3,(SP) ;ATA=1?
7514 040606 001005 BNE SC5 ;YES--BRANCH
7515 040610 005201 SC4: INC R1 ;MOVE TO THE NEXT DRIVE
7516 040612 106303 ASLB R3
7517 040614 001373 BNE SC3 ;BRANCH IF MORE TO CHECK?
7518 040616 005726 TST (SP)+ ;CLEAN OFF THE STACK
7519 040620 000207 RTS PC ;RETURN TO USER
7520 040622 105761 035406 SC5: TSTB DPINT(R1) ;INITIALIZING THE DRIVE ?
7521 040626 001402 BEQ 1$ ;BR IF NOT
7522 040630 000137 041546 JMP SC13 ;PROCESS THE DRIVE
7523 040634 105761 035416 1$: TSTB DPRQS(R1) ;PORT REQUEST OUTSTANDING ?
7524 040640 001402 BEQ 2$ ;BR IF NOT
7525 040642 000137 041546 JMP SC13 ;START THE OUTSTANDING COMMAND
7526 040646 105761 035366 2$: TSTB DRVSTA(R1) ;CHECK THE DRIVE STATUS
7527 040652 003025 BGT 5$ ;BRANCH IF ONLINE
7528 040654 105761 035434 TSTB ULDFLG(R1) ;UNLOAD IN PROGRESS?
7529 040660 003422 BLE 5$ ;BRANCH IF NOT
7530 040662 004737 043520 JSR PC,GETREQ ;GET DPB POINTER
7531 040666 004737 042664 JSR PC,SVRH70 ;SAVE THE RH70/RMD3 REGISTERS
7532 040672 004737 041476 JSR PC,SC12 ;SAVE RMD5, RMR1, RMR2, AND RMMR2
7533 ;ALSO DO A DRIVE INIT (DRVINT)
7534 040676 105761 035366 TSTB DRVSTA(R1) ;DID DRIVE COME ONLINE?
7535 040702 003416 BLE 6$ ;NO---BRANCH
7536 040704 032737 040000 035346 BIT #BIT14,RMERRS ;WAS THERE AN ERROR?
7537 040712 001002 BNE 3$ ;BR IF ERROR
7538 040714 000137 041366 JMP SC11 ;NO ERROR
7539 040720 013705 035350 3$: MOV RMERRS+2,R5 ;YES -- PICKUP RMR1 AND
7540 040724 000502 BR SC6A ;GO PROCESS THE ERROR
7541 040726 105761 035356 5$: TSTB DRVACT(R1) ;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
7542 040732 001033 BNE SC6 ;BR IF EITHER

```

F14

MD-11-DZRMF-A RMO3 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 142  
SINGLE/DUAL PORT RH70/RMO3 DRIVER (REV 1.0)

SEQ 0174

7543	040734	004737	041476		JSR	PC,SC12	;SAVE RMD5, RMER1, RMER2, AND RMMR2
7544							;ALSO DO A DRVINT
7545	040740	105761	035406	6\$:	TSTB	DPINT(R1)	;TRYING TO INIT THE DRIVE ?
7546	040744	001321			BNE	SC4	;BR IF YES, CHECK ON MORE DRIVES
7547	040746	105761	035366		TSTB	DRVSTA(R1)	;CHECK ON DRIVE'S STATUS
7548	040752	100412			BMI	7\$	;BR IF UNSAFE
7549	040754	032737	020000	035352	BIT	#BIT13,RMERR5+4	;ADDRESS PLUG CHANGED ?
7550	040762	001013			BNE	8\$	;BR IF YES
7551	040764	012746	000113		MOV	#113,-(SP)	;RELEASE COMMAND
7552	040770	004037	042472		JSR	RD,WRT.RM	;WRITE THE COMMAND INTO RMCS1
7553	040774	000000			RMCS1		;REGISTER INDEX
7554	040776	041336			SC8		;PARITY EXIT ADDRESS
7555	041000	011605		7\$:	MOV	(SP),R5	;PICKUP (RMAS) BEFORE THE ERROR CALL
7556	041002	004037	043610		JSR	RD,ES.SAV	;SAVE THE ADDRESS IN 'SESCAPE'
7557	041006	104002			ERROR	2	;REPORT THE UNEXPECTED ATTENTION
7558	041010	000677			BR	SC4	;GO CHECK FOR MORE ATA'S
7559	041012			8\$:			
7560	041012	004037	043610		JSR	RD,ES.SAV	;SAVE THE ADDRESS IN 'SESCAPE'
7561	041016	104005			ERROR	5	;REPORT THE ADDRESS PLUG CHANGE
7562	041020	000673			BR	SC4	;CHECK FOR MORE DRIVES
7563	041022	006301		SC6:	ASL	R1	;SETUP TO ADDRESS WORDS
7564	041024	012761	177777	035460	MOV	#-1,TIMER(R1)	;STOP THE TIMER
7565	041032	006201			ASR	R1	;RESTORE THE DRIVE ADDRESS
7566	041034	004737	043520		JSR	PC,GETREQ	;GET THE DPB POINTER FROM THE QUEUE
7567	041040	010164	000010		MOV	R1,RMCS2(R4)	;SELECT DRIVE
7568	041044	004037	042312		JSR	RD,RD.RM	;READ THE RMO3'S STATUS REG.
7569	041050	000012			RMD5		
7570	041052	041336			SC8		
7571	041054	011605			MOV	(SP),R5	;AND PUT IT IN R5
7572	041056	006126			ROL	(SP)+	;WAS THERE AN ERROR?
7573	041060	100407			BMI	1\$	;BR IF ERROR
7574	041062	105761	035356		TSTB	DRVACT(R1)	;CHECK DRIVE'S STATE
7575	041066	003137			BGT	SC11	;BR IF DRIVE ACTIVE WITH ORDER
7576	041070	052762	100210	000016	BIS	#BIT15!BIT07!BIT03,16(R2)	;INFORM USER OF ERROR RECOVER COMPLETION
7577	041076	000470			BR	SC7	
7578	041100	004037	042312	1\$:	JSR	RD,RD.RM	;READ ERROR REGISTER #1
7579	041104	000014			RMER1		
7580	041106	041336			SC8		
7581	041110	012605			MOV	(SP)+,R5	;AND SAVE IT IN R5
7582	041112	004737	042664		JSR	PC,SVRH70	;SAVE RH70/RMO3 REGISTERS
7583	041116	012746	000111		MOV	#111,-(SP)	;ISSUE A DRIVE CLEAR
7584	041122	004037	042472		JSR	RD,WRT.RM	
7585	041126	000000			RMCS1		
7586	041130	041336			SC8		
7587	041132	006105		SC6A:	ROL	R5	;WAS "UNSAFE" CONDITION =1?
7588	041134	100406			BMI	1\$	;BRANCH IF YES
7589	041136	005702			TST	R2	;ANYTHING IN QUEUE ?
7590	041140	001447			BEQ	SC7	;BR IF NOT
7591	041142	052762	100240	000016	BIS	#BIT15!BIT07!BIT05,16(R2)	;INFORM USER OF ERROR
7592	041150	000443			BR	SC7	
7593	041152	004037	042312	1\$:	JSR	RD,RD.RM	;READ DRIVE STATUS REG. #1
7594	041156	000012			RMD5		
7595	041160	041336			SC8		
7596	041162	011605			MOV	(SP),R5	;SAVE RMD5 IN R5
7597	041164	006126			ROL	(SP)+	; "ERR"=1?
7598	041166	100011			BPL	2\$	;BR IF NO--UNSAFE CLEARED



G14

MD-11-DZRMF-A RM03 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 143  
SINGLE/DUAL PORT RH70/RM03 DRIVER (REV 1.0)

SEQ 0175

7599	041170	112751	177777	035366		MOVB	#-1,DRVSTA(R1)	;DRIVE IS UNSAFE
7600	041176	004737	042664			JSR	PC,SVRH70	;SAVE RH70/RM03 REGISTERS
7601	041202	052762	110000	000016		BIS	#BIT15:BIT12,16(R2)	;INFORM USER OF UNSAFE ERROR
7602	041210	000423				BR	SC7	
7603	041212	032705	010000		2\$:	BIT	#BIT12,R5	; "MOL" = 1 ?
7604	041216	001015				BNE	3\$	;BR IF YES
7605	041220	112761	177777	03535		MOVB	#-1,DRVACT(R1)	;ACTIVE ERROR RECOVER
7606	041226	112761	000001	035366		MOVB	#1,DRVSTA(R1)	;ONLINE
7607	041234	006301				ASL	R1	
7608	041236	012761	072460	035460		MOV	#30000.,TIMER(R1)	;START 30 SECOND TIMER
7609	041244	006201				ASR	R1	
7610	041246	00137	040610			JMP	SC4	
7611	041252	052762	100220	000016	3\$:	BIS	#BIT15:BIT07:BIT04,16(R2)	;INFORM USER OF ERROR
7612	041260	105061	00356		SC7:	CLRB	DRVACT(R1)	;DRIVE IS IDLE
7613	041264	004737	043424			JSR	PC,EMPTYQ	;DUMP THE QUEUE
7614	041270	105761	035434			TSTB	ULDFLG(R1)	;UNLOAD IN PROGRESS OR QUEUE?
7615	041274	003002				BGT	1\$	;BR IF NOT
7616	041276	105061	035434			CLRB	ULDFLG(R1)	;CLEAR UNLOAD FLAG
7617	041302	116164	035502	000016	1\$:	MOVB	ATABIT(R1),RMAS(R4)	;CLEAR ATTENTION BIT
7618	041310	105761	035366			TSTB	DRVSTA(R1)	;IS THE DRIVE UNSAFE ?
7619	041314	100406				BMI	2\$	;BR IF IT IS
7620	041316	012746	000113			MOV	#113,-(SP)	;RELEASE COMMAND
7621	041322	004037	042472			JSR	RD,WRT.RM	;WRITE THE COMMAND INTO RPCS1
7622	041326	000000				RMCS1		;REGISTER INDEX
7623	041330	041336				SC8		;PARITY EXIT ADDRESS
7624	041332	000137	040610		2\$:	JMP	SC4	;CHECK FOR MORE DRIVES
7625	041336	105761	035356		SC8:	TSTB	DRVACT(R1)	;IS DRIVE IDLE?
7626	041342	001405				BEQ	1\$	;YES--BRANCH
7627	041344	004737	043520			JSR	PC,GETREQ	;GET DPB POINTER
7628	041350	004737	037610			JSP	PC,C17	;PROCESS THE PARITY ERROR
7629	041354	000402				BR	2\$	;CONTINUE
7630	041356	004737	037636		1\$:	JSR	PC,C17B	;PROCESS THE UNCORRECTABLE PARITY ERROR
7631	041362	000137	040610		2\$:	JMP	SC4	;CHECK MORE DRIVES
7632	041366	105761	035434		SC11:	TSTB	ULDFLG(R1)	; "UNLOAD IN PROGRESS" ?
7633	041372	003402				BLE	1\$	;BRANCH IF NO
7634	041374	105061	035434			CLRB	ULDFLG(R1)	;CLEAR UNLOAD FLAG
7635	041400	105061	035356		1\$:	CLRB	DRVACT(R1)	;SET DRIVE IDLE
7636	041404	136137	035502	035430		BITB	ATABIT(R1),SRCHWT	;DOING A SEARCH OPERATION FOR
7637								;AN I/O COMMAND?
7638	041412	001012				BNE	2\$	;BRANCH IF YES
7639	041414	004737	043542			JSR	PC,POPQUE	;REMOVE REQUEST FROM QUEUE
7640	041420	052762	000200	000016		BIS	#BIT07,16(R2)	;SET "DONE" BIT
7641	041426	005737	035454			TST	SAVEFG	;SAVE THE REGISTERS?
7642	041432	100002				BPL	2\$	;BRANCH IF NO
7643	041434	004737	042664			JSR	PC,SVRH70	;YES--SAVE ALL OF THE RH70/RM03 REG'S
7644	041440	116164	035502	000016	2\$:	MOVB	ATABIT(R1),RMAS(R4)	;CLEAR ATTENTION BIT
7645	041446	146137	035502	035430		BICB	ATABIT(R1),SRCHWT	;CLEAR IMPLIED SEEK SET
7646	041454	006301				ASL	R1	;WORD INDEX
7647	041456	012761	177777	035460		MOV	#-1,TIMER(R1)	;STOP CLOCK
7648	041464	006201				ASR	R1	;RESTORE R1
7649	041466	004737	036530			JSR	PC,OPT	;START A REQUEST
7650	041472	000137	040610			JMP	SC4	;CHECK FOR MORE DRIVES
7651	041476	010164	000010		SC12:	MOV	R1,RMCS2(R4)	;SELECT DRIVE
7652	041502	016437	000012	035376		MOV	RMDS(R4),RMERRS	;SAVE THE FOUR REGISTERS THAT
7653	041510	016437	000014	035350		MOV	RMER1(R4),RMERRS+2	;WILL TELL US SOMETHING
7654	041516	016437	000042	035352		MOV	RMER2(R4),RMERRS+4	

# H14

MD-11-DZRMF-A RMO3 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 144  
SINGLE/DUAL PORT RH70/RMO3 DRIVER (REV 1.0)

SEQ 0176

7655	041524	016437	000040	035354		MOV	RMMR2(R4),RMERRS+6	
7656	041532	004037	035744			JSR	RO,DRVINT	;INIT. THE STATE OF THE DRIVE
7657	041536	000401				BR	1\$	;TAKE ERROR EXIT
7658	041540	000207				RTS	PC	;RETURN
7659	041542	005726			1\$:	TST	(SP)+	;POP PC OFF OF THE STACK
7660	041544	000674				BR	SC8	;PROCESS THE PARITY ERROR
7661	041546	006301			SC13:	ASL	R1	;SETUP TO ADDRESS WORDS
7662	041550	012761	177777	035460		MOV	#-1,TIMER(R1)	;STOP THE TIMER
7663	041556	006201				ASR	R1	
7664	041560	010164	000010			MOV	R1,RMCS2(R4)	;SELECT THE DRIVE
7665	041564	116164	035502	000016		MOV	ATABIT(R1),RMAS(R4)	;CLEAR THE ATTENTION BIT
7666	041572	105761	035406		1\$:	TSTB	DPINT(R1)	;INITIALIZING THE DRIVE ?
7667	041576	001424				BEQ	2\$	;BR IF NOT
7668	041600	105061	035406			CLRB	DPINT(R1)	;CLEAR THE INIT INDICATOR
7669	041604	004037	035744			JSR	RO,DRVINT	;GO INIT THE DRIVE
7670	041610	000240				NOP		;DUMMY PARITY ERROR RETURN
7671	041612	105761	035366			TSTB	DRVSTA(R1)	;DRIVE ONLINE ?
7672	041616	003014				BGT	2\$	;BR IF YES -- START ORDER
7673	041620	005702				TST	R2	;QUEUE ENTRY FOR THE DRIVE
7674	041622	001426				BEQ	3\$	;BR IF NOT
7675	041624	004737	043520			JSR	PC,GETREQ	;GET DPB ADDRESS
7676	041630	052762	140000	000016		BIS	#BIT15:BIT14,16(R2)	;INFORM USER THAT DRIVE OFFLINE
7677	041636	004737	042664			JSR	PC,SVRH70	;SAVE THE REGISTERS
7678	041642	004737	043424			JSR	PC,EMPTYQ	;EMPTY THE REQUEST QUEUE
7679	041646	000414				BR	3\$	
7680	041650	032764	004000	000000	2\$:	BIT	#BIT11,RMCS1(R4)	;DVA SET ?
7681	041656	001006				BNE	4\$	;SET THEN CALL OPT
7682	041660	006301				ASL	R1	
7683	041662	012761	023420	035460		MOV	#10000.,TIMER(R1)	
7684	041670	006201				ASR	R1	
7685	041672	000402				BR	3\$	
7686	041674	004737	036530		4\$:	JSR	PC,OPT	;START THE PENDING REQUEST
7687	041700	000137	040610		3\$:	JMP	SC4	;PROCESS OTHER DRIVES
7688								
7689								
7690								
7691								
7692								
7693								
7694	041704	005737	035432			RPTMR:	TST	ACTDRV
7695	041710	001027					BNE	4\$
7696	041712	112737	000001	035433			MOV	#1,ACTSTR
7697	041720	104412					SAVREG	
7698	041722	005001					CLR	R1
7699	041724	005003					CLR	R3
7700	041726	005763	035460		1\$:	TST	TIMER(R3)	;IS THE TIMER RUNNING?
7701	041732	002406					BLT	2\$
7702	041734	166663	000002	035460			SUB	2(SP),TIMER(R3)
7703	041742	003002					BGT	2\$
7704	041744	004737	041774				JSR	PC,STO
7705	041750	005201			2\$:	INC	R1	;MOVE TO NEXT DRIVE
7706	041752	005723					TST	(R3)+
7707	041754	022701	000010				CMP	#8.,R1
7708	041760	003362					BGT	1\$
7709	041762	104413			3\$:	RESREG		;RESTORE R0 - R5
7710	041764	105037	035433				CLRB	ACTSTR
								;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG

7711	041770	012616		4\$:	MOV	(SP)+,(SP)		;ADJUST THE STACK
7712	041772	000207			RTS	PC		;RETURN
7713								
7714								;SOFTWARE TIMEOUT ROUTINE
7715								
7716								;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
7717								OR GREATER
7718								
7719								
7720					CALL:	STO		
7721						MOV	#DRVNUM,R1	;DRIVE NUMBER
7722						JSR	PC,STO	;CALL
7723						RETURN		
7724	041774	010146		STO:	MOV	R1,-(SP)		;SAVE R1
7725	041776	010246			MOV	R2,-(SP)		;SAVE R2
7726	042000	010346			MOV	R3,-(SP)		;SAVE R3
7727	042002	010446			MOV	R4,-(SP)		;SAVE R4
7728	042004	013704	035514		MOV	RADDR,R4		;GET ADDRESS OF "RMCS1"
7729	042010	010164	000010		MOV	R1,RMCS2(R4)		;SELECT THE DRIVE
7730	042014	004037	042312		JSR	RD,RD.RM		;READ "DRIVE STATUS REG"
7731	042020	000012			RMO\$			
7732	042022	042174			STO\$			
7733	042024	105726			TSTB	(SP)+		;IS "DRY"=1?
7734	042026	100434			BMI	STO2		;BR IF YES
7735	042030	105761	035406	STO1:	TSTB	DPINT(R1)		;TRYING TO INITIALIZE THE DRIVE ?
7736	042034	001031			BNE	STO2		;BR IF YES
7737	042036	105761	035416		TSTB	DPRQS(R1)		;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
7738	042042	001026			BNE	STO2		;BR IF YES
7739	042044	013702	035426		MOV	TRNSW,R2		;PICKUP TRANSFER WAIT QUEUE
7740	042050	020137	035500		CMP	R1,DTUW		;TRANSFER UNDERWAY ON THIS DRIVE?
7741	042054	001404			BEQ	1\$		;BRANCH IF YES
7742	042056	000137	042300		JMP	STO9		;IF NOT DON'T BOTHER DRIVES
7743	042062	004737	043520		JSR	PC,GETREQ		;GET DPB ADDRESS
7744	042066	052762	101000	000016	1\$:	BIS	#BIT15:BIT09,16(R2)	;SET THE ERROR FLAGS
7745	042074	004737	042664		JSR	PC,SVRH70		;SAVE RH70/RMO3 REGISTERS
7746	042100	012764	000040	000010	MOV	#BIT05,RMCS2(R4)		; "INIT" THE MASS BUS
7747	042106	105061	035356		CLRB	DRVACT(R1)		;DRIVE IS IDLE
7748	042112	105061	035434		CLRB	ULDFLG(R1)		;CLEAR THE UNLOAD FLAG
7749	042116	000470			BR	STO9		;DON'T BOTHER OTHER DRIVES
7750	042120	116405	000016	STO2:	MOVB	RMA\$ (R4),R5		;READ ATTENTION REG
7751	042124	136105	035502		BITB	ATABIT(R1),R5		;IS ATTENTION FOR THIS DRIVE UP ?
7752	042130	001007			BNE	STO3		;YES--BRANCH
7753	042132	105761	035406		TSTB	DPINT(R1)		;TRYING TO INITIALIZE THE DRIVE ?
7754	042136	001021			BNE	STO6		;BR IF YES - DRIVE NOT ONLINE
7755	042140	105761	035416		TSTB	DPRQS(R1)		;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
7756	042144	001035			BNE	STO7		;BR IF YES - NO RESPONSE TO REQUEST
7757	042146	000454			BR	STO9		;OTHER WISE EXIT
7758	042150	105761	035406	STO3:	TSTB	DPINT(R1)		;INITIALIZING THE DRIVE ?
7759	042154	001003			BNE	1\$		;BR IF INIT PENDING
7760	042156	105761	035416		TSTB	DPRQS(R1)		;PORT REQUEST PENDING ?
7761	042162	001446			BEQ	STO9		;BR IF NOT
7762	042164	012763	177777	035460	1\$:	MOV	#-1,TIMER(R3)	;STOP THE TIMER
7763	042172	000442			BR	STO9		;EXIT
7764	042174	004737	037710	STO5:	JSR	PC,CIB		;GO HANDLE THE PARITY ERROR
7765	042200	000437			BR	STO9		
7766	042202	105061	035406	STO6:	CLRB	DPINT(R1)		;CLEAR THE INITIALIZE INDICATOR

```

7767 042206 105061 035366          CLRB   DRVSTA(R1)      ;SET UNIT OFFLINE
7768 042212 012763 177777 035460  MOV    #-1,TIMER(R3) ;STOP THE TIMER
7769 042220 004737 043520          JSR    PC,GETREQ      ;GET THE DPB ADDRESS
7770 042224 005702                    TST    R2              ;REQUEST IN QUEUE ?
7771 042226 001424                    BEQ    ST09            ;BR IF NOT
7772 042230 052762 140000 000016  BIS    #BIT15:BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
7773 042236 000414                    BR     ST08            ;FINISH
7774 042240 012763 177777 035460  ST07: MOV    #-1,TIMER(R3) ;STOP THE TIMER
7775 042246 105061 035416          CLRB   DPRQS(R1)      ;CLEAR PORT REQUEST INDICATOR
7776 042252 004737 043520          JSR    PC,GETREQ      ;GET DPB ADDRESS
7777 042256 005702                    TST    R2              ;QUEUE ENTRY FOR DRIVE ?
7778 042260 001407                    BEQ    ST09            ;BR IF NONE
7779 042262 012762 100004 000016  MOV    #BIT15:BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
7780 042270 004737 043424          ST08: JSR    PC,EMPTYQ  ;CLEAR THE QUEUE FOR THE DRIVE
7781 042274 004737 042664          ST09: JSR    PC,SVRH70 ;SAVE THE REGISTERS
7782 042300 012604                    MOV    (SP)+,R4        ;RESTORE R4
7783 042302 012603                    MOV    (SP)+,R3        ;RESTORE R3
7784 042304 012602                    MOV    (SP)+,R2        ;RESTORE R2
7785 042306 012601                    MOV    (SP)+,R1        ;RESTORE R1
7786 042310 000207                    RTS     PC              ;RETURN
7787
7788 ;ROUTINE TO READ A RH70/RMO3 REGISTER
7789 ;CALL
7790 ;
7791 ;JSR    RD, RD.RM      ;GO READ A REGISTER
7792 ;INDEX  ;REG. INDEX FROM BASE
7793 ;ERRADR ;ERROR ADDRESS--PROCESS ERROR STARTING
7794 ;        ;AT THIS ADDRESS
7795 ;        ;CONTENTS OF REG. IS ON THE STACK
7796 ;
7797 042312 013737 035512 042460  RD.RM: MOV    MCPMX,RD.RM2 ;MAX. RETRYS ALLOWED
7798 042320 011646                    MOV    (SP)-,(SP)      ;SAVE RD FOR RETURN
7799 042322 013737 035514 042336  MOV    RMAADR,RD.ADR   ;FORM THE DESIRED ADDRESS
7800 042330 062037 042336          ADD    (RD)+,RD.ADR    ;USING THE BASE AND THE INDEX
7801 042334 013727          RD.RM1: MOV    @((PC)+,(PC)+ ;READ THE DESIRED REGISTER OF THE RMO3
7802 042336 000000          RD.ADR: .WORD 0        ;ADDRESS IS FORMED HERE
7803 042340 000000          RD.WRD: .WORD 0        ;REG. CONTENTS PUT HERE
7804 042342 013766 042340 000002  MOV    RD.WRD,2(SP)    ;RETURN IT TO THE USER
7805 042350 013746 035514          MOV    RMAADR,-(SP)   ;PUT THE ADDRESS ON THE STACK
7806 042354 062716 000010          ADD    #RMCS2,(SP)    ;FORM THE ADDRESS OF RMCS2
7807 042360 032736 010000          BIT    #BIT12,@(SP)+  ;CHECK THE 'NED' BIT
7808 042364 001037          BNE    RD.RM3         ;BR IF DRIVE NON-EXISTENT
7809 042366 017746 173122          MOV    @RMAADR,-(SP)  ;READ RMCS1
7810 042372 032716 020000          BIT    #BIT13,(SP)    ;DID MCPE SET?
7811 042376 001002          BNE    1$            ;BRANCH IF YES
7812 042400 022620          CMP    (SP)+,(RD)+    ;ADJUST FOR RETURN
7813 042402 000432          BR     RD.RM4         ;EXIT
7814 042404          1$:
7815 042404 004037 043610          JSR    RD,ES.SAV      ;SAVE THE ADDRESS IN 'SESCAPE'
7816 042410 104003          ERROR 3              ;REPORT "MCPE" ERROR
7817 042412 005737 035500          TST    DTUW           ;DATA TRANSFER UNDERWAY?
7818 042416 100405          BMI    2$            ;NO--BRANCH
7819 042420 032716 040000          BIT    #BIT14,(SP)    ;NO--"TRE"=1?
7820 042424 001402          BEQ    2$            ;NO--BRANCH
7821 042426 005726          TST    (SP)+         ;YES--CLEAN OFF THE STACK AND
7822 042430 000415          BR     RD.RM3         ;TAKE THE FATAL ERROR EXIT

```

7823	042432	052716	040000		2\$:	BIS	#BIT14, (SP)	;CLEAR "MCPE" BY SENDING A "1" TO "TRE"
7824	042436	000316				SWAB	(SP)	;POSITION BEFORE WRITING
7825	042440	013737	035514	042454		MOV	RMADR, 3\$	;FORM ADDRESS OF HIGH BYTE
7826	042446	005237	042454			INC	3\$	
7827	042452	112637				MOVB	(SP)+, 2(PC)+	;WRITE THE HIGH BYTE OF RMCS1
7828	042454	000000			3\$:	.WORD	0	;ADDRESS STORAGE
7829	042456	005327				DEC	(PC)+	;EXCEEDED MAX. RETRYS
7830	042460	000003			RD.RM2:	.WORD	3	
7831	042462	002324				BGE	RD.RM1	;BRANCH IF NO
7832	042464	011000			RD.RM3:	MOV	(RO), RO	;FATAL ERROR EXIT
7833	042466	012616				MOV	(SP)↑, (SP)	
7834	042470	000200			RD.RM4:	RTS	RO	
7835								
7836								
7837								
7838								
7839								
7840								
7841								
7842								
7843								
7844								
7845	042472	013737	035512	042650	WRT.RM:	MOV	MCPEMX, WRT.R2	;MAX RETRYS ALLOWED
7846	042500	016637	000002	042560		MOV	2(SP), WRT.WD	;SAVE THE WORD TO WRITE
7847	042506	012616				MOV	(SP)+, (SP)	;ADJUST THE STACK
7848	042510	012037	042562			MOV	(RO)+, WRT.AD	;GET INDEX OF REGISTER TO BE WRITTEN
7849	042514	001015				BNE	1\$	;BRANCH IF NOT RMCS1
7850	042516	122737	000150	042560		CMPB	#150, WRT.WD	;IS THE COMMAND FOR DATA TRANSFERS?
7851	042524	002411				BLT	1\$	;YES--DON'T GET THE OLD A16 & A17, & PSEL
7852	042526	004037	042312			JSR	RO, RD.RM	;NO---COMBINE A16&A17, & PSEL WITH
7853	042532	000000				RMCS1		;THE COMMAND BEFORE SENDING IT TO
7854	042534	042654				WRT.R3		;THE RH70/RMO3
7855	042536	000316				SWAB	(SP)	
7856	042540	042716	177770			BIC	#1C7, (SP)	
7857	042544	112637	042561			MOVB	(SP)↑, WRT.WD+1	
7858	042550	063737	035514	042562	1\$:	ADD	RMADR, WRT.AD	;FORM THE ADDRESS OF THE DISK REG.
7859	042556	012737			WRT.R1:	MOV	(PC)+, 2(PC)+	;LOAD THE DESIRED REG.
7860	042560	000000			WRT.WD:	.WORD	0	;WORD TO WRITE GOES HERE
7861	042562	000000			WRT.AD:	.WORD	0	;ADDRESS IS FORMED HERE
7862	042564	013746	035514			MOV	RMADR, -(SP)	;PUT THE ADDRESS ON THE STACK
7863	042570	062716	000010			ADD	#RMCS2, (SP)	;FORM THE ADDRESS OF RMCS2
7864	042574	032736	010000			BIT	#BIT12, 2(SP)+	;CHECK THE 'NED' BIT
7865	042600	001025				BNE	WRT.R3	;BR IF DRIVE NON-EXISTENT
7866	042602	074037	042312			JSR	RO, RD.RM	;CHECK FOR PARITY ERROR ON WRITE
7867	042606	000014				RMER1		
7868	042610	042654				WRT.R3		
7869	042612	032726	000010			BIT	#BIT03, (SP)+	
7870	042616	001420				BEQ	WRT.R4	;BRANCH IF "PAR=0"
7871	042620	016037	177776	042632		MOV	-2(RO), 1\$	;PICKUP THE INDEX
7872	042626	004037	042312			JSR	RO, RD.RM	;READ THE REG.
7873	042632	000000			1\$:	.WORD	0	;REG. INDEX
7874	042634	042654				WRT.R3		;RETURN TO THIS ADDRESS ON ERROR
7875	042636	004037	043610			JSR	RO, ES.SAV	;SAVE THE ADDRESS IN 'ESCAPE'
7876	042642	104004				ERROR	4	;REPORT THE PARITY ON WRITE ERROR
7877	042644	005726				TST	(SP)+	;CLEAR OFF THE STACK
7878	042646	005327				DEC	(PC)+	;DECREMENT THE ERROR COUNT

;ROUTINE TO WRITE A REGISTER

;CALL

```

:      MOV    DATA, -(SP)
:      JSR    RO, WRT.RM
:      INDEX
:      ERRADR
:      RETURN

```

```

;DATA TO BE LOADED ON THE STACK
;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
;INDEX OF THE REGISTER TO BE LOADED
;ADDRESS TO RETURN TO ON AN ERROR
;ERROR FREE RETURN

```

```

;MAX RETRYS ALLOWED
;SAVE THE WORD TO WRITE
;ADJUST THE STACK
;GET INDEX OF REGISTER TO BE WRITTEN
;BRANCH IF NOT RMCS1
;IS THE COMMAND FOR DATA TRANSFERS?
;YES--DON'T GET THE OLD A16 & A17, & PSEL
;NO---COMBINE A16&A17, & PSEL WITH
;THE COMMAND BEFORE SENDING IT TO
;THE RH70/RMO3

```

```

;FORM THE ADDRESS OF THE DISK REG.
;LOAD THE DESIRED REG.
;WORD TO WRITE GOES HERE
;ADDRESS IS FORMED HERE
;PUT THE ADDRESS ON THE STACK
;FORM THE ADDRESS OF RMCS2
;CHECK THE 'NED' BIT
;BR IF DRIVE NON-EXISTENT
;CHECK FOR PARITY ERROR ON WRITE

```

```

;BRANCH IF "PAR=0"
;PICKUP THE INDEX
;READ THE REG.
;REG. INDEX
;RETURN TO THIS ADDRESS ON ERROR
;SAVE THE ADDRESS IN 'ESCAPE'
;REPORT THE PARITY ON WRITE ERROR
;CLEAR OFF THE STACK
;DECREMENT THE ERROR COUNT

```

```

7879 042650 000003      WRT.R2: .WORD 3          ;RETRY COUNTER
7880 042652 002341      BGE WRT.R1             ;TRY AGAIN IF NOT FINISHED
7881 042654 011000      WRT.R3: MOV (R0),R0    ;TAKE THE "PARITY ON WRITE" ERROR EXIT
7882 042656 000401      BR WRT.R5             ;EXIT
7883 042660 005720      WRT.R4: TST (R0)+     ;ADJUST FOR ERROR FREE EXIT
7884 042662 000200      WRT.R5: RTS R0
7885
7886 ;ROUTINE TO SAVE THE RH70/RP04/5/RMO3 REGISTERS AS PER DPB+14
7887 ;CALL
7888 ;
7889 ;   MOV #DPBNUM,R2      ;DPB POINTER TO R2
7890 ;   JSR PC,SVRH70      ;SAVE THE DRIVES REG'S
7891 ;
7892 SVRH70: SAVREG        ;SAVE R0 - R5
7893 TST R2                ;QUEUE ENTRY FOR THE DRIVE ?
7894 BEQ 6$                ;BR IF NONE
7895 MOV RMAADR,R4         ;
7896 MOVB (R2),RMCS2(R4)  ;SELECT DRIVE
7897 MOV 14(R2),R3        ;GET THE ERROR TABLE POINTER
7898 BEQ 6$                ;EXIT IF NO ADDRESS
7899 CLR 3$                ;COUNTER & POINTER
7900 CMP 3$,#RMOB        ;REACHED THE BUFFER REGISTER ?
7901 BNE 2$                ;BR IF NOT
7902 BIT #BIT07,RMCS2(R4) ;'OR' SET ?
7903 BNE 2$                ;BR IF SET
7904 CLR (R3)+            ;STORE RMOB AS ZEROES
7905 BR 4$                ;CONTINUE
7906 JSR RD,RD.RM        ;READ THE SELECTED REGISTER
7907 .WORD 0              ;REGISTER INDEX
7908 5$                    ;ERROR RETURN ADDRESS
7909 MOV (SP)+,(R3)+      ;STORE THE REGISTER CONTENTS
7910 CMP 3$,#RMEC2       ;REACHED THE END ?
7911 BEQ 6$                ;BR IF YES
7912 ADD #2,3$           ;INCREMENT THE REGISTER INDEX
7913 BR 1$                ;CONTINUE READING THE REGISTERS
7914 JSR PC,C17          ;PROCESS THE UNCORRECTABLE PARITY ERROR
7915 RESREG              ;RESTORE R0 - R5
7916 RTS PC              ;RETURN
7917
7918 ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
7919 ;CALL
7920 ;   MOV #DRVNUM,R1     ;DRIVE NUMBER TO R1
7921 ;   JSR PC,SET.IE     ;SET "IE"
7922 ;   RETURN
7923 ;
7924 SET.IE: MOV R4, -(SP)  ;SAVE R4
7925 MOV RMAADR,R4        ;PICKUP ADDRESS OF RMCS1
7926 MOV R1,RMCS2(R4)    ;SELECT DRIVE
7927 MOV (R4),-(SP)      ;READ RMCS1
7928 BIS #BIT14,(SP)     ;SET THE "TRE" BIT OF THE WORD READ
7929 SWAB (SP)           ;ADJUST FOR DATO
7930 MOVB #BIT06,(R4)    ;SET "IE"
7931 BIT #BIT12,RMCS2(R4) ;IS "NED"=1?
7932 BNE 1$              ;YES--CLEAR "TRE"
7933 TST (SP)+            ;CLEAN OFF THE STACK
7934 BR 2$

```

```

7935 043044 112664 000001
7936 043050 012604
7937 043052 000207
7938
7939
7940 043054 000
7941 043055 000
7942 043056 000
7943 043057 000
7944 043060 000
7945 043061 000
7946 043062 000
7947 043063 000
7948
7949
7950
7951 043064 043146
7952 043066 043166
7953 043070 043206
7954 043072 043226
7955 043074 043246
7956 043076 043266
7957 043100 043306
7958 043102 043326
7959
7960
7961
7962 043104 043146
7963 043106 043166
7964 043110 043206
7965 043112 043226
7966 043114 043246
7967 043116 043266
7968 043120 043306
7969 043122 043326
7970
7971 043124 043146
7972 043126 043166
7973 043130 043206
7974 043132 043226
7975 043134 043246
7976 043136 043266
7977 043140 043306
7978 043142 043326
7979 043144 043346
7980
7981
7982
7983 043146 000010
7984 043166 000010
7985 043206 000010
7986 043226 000010
7987 043246 000010
7988 043266 000010
7989 043306 000010
7990 043326 000010

```

```

1$:      MOVB   (SP)+,1(R4)      ;CLEAR "TRE"
2$:      MOV    (SP)+,R4         ;RESTORE R4
        RTS     PC               ;RETURN TO CALLER

; QUEUE COUNT
Q CNT:   .BYTE  0                ;DRIVE 0
        .BYTE  0                ;DRIVE 1
        .BYTE  0                ;DRIVE 2
        .BYTE  0                ;DRIVE 3
        .BYTE  0                ;DRIVE 4
        .BYTE  0                ;DRIVE 5
        .BYTE  0                ;DRIVE 6
        .BYTE  0                ;DRIVE 7

; QUEUE INPUT POINTERS
Q INPT:  .WORD  QDRV0            ;DRIVE 0
        .WORD  QDRV1            ;DRIVE 1
        .WORD  QDRV2            ;DRIVE 2
        .WORD  QDRV3            ;DRIVE 3
        .WORD  QDRV4            ;DRIVE 4
        .WORD  QDRV5            ;DRIVE 5
        .WORD  QDRV6            ;DRIVE 6
        .WORD  QDRV7            ;DRIVE 7

; QUEUE OUTPUT POINTERS
Q OUTPT: .WORD  QDRV0            ;DRIVE 0
        .WORD  QDRV1            ;DRIVE 1
        .WORD  QDRV2            ;DRIVE 2
        .WORD  QDRV3            ;DRIVE 3
        .WORD  QDRV4            ;DRIVE 4
        .WORD  QDRV5            ;DRIVE 5
        .WORD  QDRV6            ;DRIVE 6
        .WORD  QDRV7            ;DRIVE 7

Q START: .WORD  QDRV0            ;DRIVE 0 START ADDRESS
Q STOP:  .WORD  QDRV1            ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
        .WORD  QDRV2            ;STOP DRIVE 1--START DRIVE 2
        .WORD  QDRV3            ;STOP DRIVE 2--START DRIVE 3
        .WORD  QDRV4            ;STOP DRIVE 3--START DRIVE 4
        .WORD  QDRV5            ;STOP DRIVE 4--START DRIVE 5
        .WORD  QDRV6            ;STOP DRIVE 5--START DRIVE 6
        .WORD  QDRV7            ;STOP DRIVE 6--START DRIVE 7
        .WORD  QTERM           ;STOP DRIVE 7

; DRIVE REQUEST QUEUES
Q DRV0:  .BLKW  10
Q DRV1:  .BLKW  10
Q DRV2:  .BLKW  10
Q DRV3:  .BLKW  10
Q DRV4:  .BLKW  10
Q DRV5:  .BLKW  10
Q DRV6:  .BLKW  10
Q DRV7:  .BLKW  10

```

```

7991          043346          QTERM=.
7992
7993          ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
7994          ;CALL
7995          ;
7996          ;      JSR      PC,CLRQUE
7997          ;
7998 043346 104412          CLRQUE: SAVREG          ;SAVE R0 - R5
7999 043350 012702 043054  MOV      #QCNT,R2          ;ZERO THE QUEUE COUNTS
8000 043354 005022          CLR      (R2)+          ;DRIVES 0 & 1
8001 043356 005022          CLR      (R2)+          ;DRIVES 2 & 3
8002 043360 005022          CLR      (R2)+          ;DRIVES 4 & 5
8003 043362 005022          CLR      (R2)+          ;DRIVES 6 & 7
8004 043364 012703 000010  MOV      #8,R3          ;MOVE THE STARTING
8005 043370 012701 043124  MOV      #QSTART,R1      ;ADDRESS OF THE QUEUE INTO
8006 043374 012122          1$: MOV      (R1)+,(R2)+    ;THE QUEUE INPUT POINTER
8007 043376 005303          DEC      R3
8008 043400 001375          BNE     1$
8009 043402 012703 000010  MOV      #8,R3          ;MOVE THE STARTING ADDRESS
8010 043406 012701 043124  MOV      #QSTART,R1      ;OF THE QUEUE INTO THE
8011 043412 012122          2$: MOV      (R1)+,(R2)+    ;QUEUE OUTPUT POINTER
8012 043414 005303          DEC      R3
8013 043416 001375          BNE     2$
8014 043420 104413          RESREG
8015 043422 000207          RTS      PC          ;RESTORE R0 - R5
8016
8017          ;EMPTY THE QUEUE SPECIFIED BY R1
8018
8019          ;CALL
8020          ;      MOV      DRVNUM,R1          ;DRIVE NUMBER TO R1
8021          ;      JSR      PC,EMPTYQ
8022          ;
8023 043424 105061 043054  EMPTYQ: CLRB   QCNT(R1)          ;CLEAR NUMBER OF ITEMS IN QUEUE
8024 043430 006301          ASL      R1
8025 043432 016161 043064 043104  MOV      QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
8026 043440 006201          ASR      R1
8027 043442 000207          RTS      PC
8028
8029          ;ROUTINE TO PUT A REQUEST IN QUEUE
8030
8031          ;CALL
8032          ;      MOV      #DRVNUM,R1          ;DRIVE NUMBER
8033          ;      MOV      #DPB,R2          ;ADDRESS OF PARAMETER BLOCK
8034          ;      JSR      RO,DRVQUE          ;GO PUT REQUEST IN QUEUE
8035          ;      RETURN1          ;RETURN HERE IF QUEUE IS FULL
8036          ;      RETURN2          ;RETURN HERE IF REQUEST IS IN QUEUE
8037          ;
8038 043444 122761 000010 043054  DRVQUE: CMPB   #10,QCNT(R1)      ;IS QUEUE FULL?
8039 043452 001421          BEQ     2$          ;BR IF YES-TAKE RETURN1
8040 043454 105261 043054          INCB   QCNT(R1)          ;INCREMENT QUEUE COUNT
8041 043460 006301          ASL      R1
8042 043462 010271 043064          MOV      R2,QINPT(R1)    ;PUT THIS REQUEST IN QUEUE
8043 043466 062761 000002 043064  ADD      #2,QINPT(R1)    ;UPDATE THE QUEUE POINTER
8044 043474 026161 043064 043126  CMP      QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
8045 043502 001003          BNE     1$          ;BRANCH IF NO
8046 043504 016161 043124 043064  MOV      QSTART(R1),QINPT(R1) ;YES--RESET POINTER

```



# B15

MD-11-DZRMF-A RMO3 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 151  
SINGLE/DUAL PORT RH70/RMO3 DRIVER (REV 1.0)

SEQ 0183

```

8047 043512 006201
8048 043514 005720
8049 043516 000200
8050
8051
8052
8053
8054
8055
8056
8057
8058
8059 043520 005002
8060 043522 105761 043054
8061 043526 001404
8062 043530 006301
8063 043532 017102 043104
8064 043536 006201
8065 043540 000207
8066
8067
8068
8069
8070
8071
8072
8073
8074 043542 105361 043054
8075 043546 006301
8076 043550 017102 043104
8077 043554 005071 043104
8078 043560 062761 000002 043104
8079 043566 026161 043104 043126
8080 043574 001003
8081 043576 016161 043124 043104
8082 043604 006201
8083 043606 000207
8084
8085
8086
8087
8088
8089
8090
8091
8092
8093 043610 012037 043624
8094 043614 013746 001206
8095 043620 005037 001206
8096 043624 000000
8097 043626 012637 001206
8098 043632 000200
8099
8100
8101
8102
    
```

```

1S:   ASR   R1
      TST   (R0)+ ;TAKE RETURN 2
2S:   RTS   R0    ;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

GETREQ: CLR   R2
        TSTB  QCNT(R1) ;IS THERE ANY REQUEST IN QUEUE?
        BEQ   2S      ;NO---BRANCH
1S:     ASL   R1
        MOV   @QOUTPT(R1),R2 ;PICKUP "DPB" POINTER FOR THIS DRIVE
2S:     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

POPQUE: DECB  QCNT(R1) ;DECREMENT QUEUE COUNT
        ASL   R1
        MOV   @QOUTPT(R1),R2 ;GET THE "DPB" POINTER
        CLR   @QOUTPT(R1) ;REMOVE DPB ADDRESS FROM THE QUEUE
        ADD   #2,@QOUTPT(R1) ;UPDATE THE QUEUE POINTER
        CMP   @QOUTPT(R1),@STOP(R1) ;TIME TO RESET THE POINTER?
        BNE   1S      ;NO--BRANCH TO EXIT
        MOV   @START(R1),@QOUTPT(R1) ;YES -RESET THE POINTER
1S:     ASR   R1
        RTS   PC      ;RETURN TO USER

;ROUTINE TO SAVE THE CONTENTS OF 'SESCAPE' WHEN THE DRIVER
;REPORTS AN ERROR DIRECTLY.
:CALL
:      JSR   RO,ES.SAV ;THE ERROR CALL
:      ERROR N ;THE RETURN IS PAST THE ERROR CALL
:      RETURN

ES.SAV: MOV   (R0)+,1S ;GET THE ERROR CALL
        MOV   SESCOPE,-(SP) ;SAVE THE ADDRESS IN 'SESCAPE'
        CLR   SESCOPE ;CLEAR THE ESCAPE RETURN
1S:     .WORD 0 ;THE ERROR CALL IS MOVED HERE
        MOV   (SP)+,SESCAPE ;RESTORE THE ESCAPE ADDRESS
        RTS   RO ;RETURN

.SBTTL ASCIZ MESSAGES
    
```

8103	043634	000122			MSG.R: .ASCIZ /R/
8104	043636	041506	000		MSG.FC: .ASCIZ /FC/
8105	043641	114	000103		MSG.LC: .ASCIZ /LC/
8106	043644	041506	000047		MSGFCP: .ASCIZ /FC'/
8107	043650	041514	000047		MSG.LCP: .ASCIZ /LC'/
8108	043654	041511	000		MSG.IC: .ASCIZ /IC/
8109	043657	106	000124		MSG.FT: .ASCIZ /FT/
8110	043662	052114	000		MSG.LT: .ASCIZ /LT/
8111	043665	111	000124		MSG.IT: .ASCIZ /IT/
8112	043670	051506	000		MSG.FS: .ASCIZ /FS/
8113	043673	114	000123		MSG.LS: .ASCIZ /LS/
8114	043676	040520	000124		MSG.PAT: .ASCIZ /PAT/
8115	043702	000075			MSG.EQ: .ASCIZ /=/
8116	043704	005015	047503	052116	MSG.CS: .ASCIZ <CR><LF>/CONTROL SWITCHES=/
8117	043712	047522	020114	053523	
8118	043720	052111	044103	051505	
8119	043726	000075			
8120					
8121	043730	027440	000040		SLASH: .ASCIZ @ / @
8122	043734	047125	052111	051440	SYSTAT: .ASCIZ /UNIT STATUS: /<CR><LF><LF>
8123	043742	040524	052524	035123	
8124	043750	005015	000012		
8125	043754	051104	053111	000105	UNTMSG: .ASCIZ /DRIVE/
8126	043762	047440	043106	044514	UNTOFF: .ASCIZ / OFFLINE/
8127	043770	042516	000		
8128	043773	040	047117	044514	UNTON: .ASCIZ / ONLINE/
8129	044000	042516	000		
8130	044003	040	047516	020124	NOTPRS: .ASCIZ / NOT PRESENT/
8131	044010	051120	051505	047105	
8132	044016	000124			
8133	044020	052440	051516	043101	NOTSAF: .ASCIZ / UNSAFE/
8134	044026	000105			
8135	044030	047040	052117	051040	NOTRM: .ASCIZ @ NOT RMO3@
8136	044036	030115	000063		
8137	044042	046522	031460	000	ISRMO3: .ASCIZ /RMO3/
8138	044047	015	042012	044522	DRIVES: .ASCIZ <CR><LF>/DRIVE(S) TO BE TESTED /
8139	044054	042526	051450	020051	
8140	044062	047524	041040	020105	
8141	044070	042524	052123	042105	
8142	044076	000040			
8143	044100	047516	042516	000	NONE: .ASCIZ /NONE/
8144	044105	054	000		COMMA: .ASCIZ /,/
8145	044107	015	047012	020117	NOCLOK: .ASCIZ <CR><LF>/NO KW11-P CLOCK, TIMING TESTS WILL NOT BE PERFORMED/
8146	044114	053513	030461	050055	
8147	044122	041440	047514	045503	
8148	044130	020054	044524	044515	
8149	044136	043516	052040	051505	
8150	044144	051524	053440	046111	
8151	044152	020114	047516	020124	
8152	044160	042502	050040	051105	
8153	044166	047506	046522	042105	
8154	044174	000			
8155	044175	015	005012	042524	TESTNG: .ASCIZ <CR><LF><LF>/TESTING DRIVE /
8156	044202	052123	047111	020107	
8157	044210	051104	053111	020105	
8158	044216	000			

8159	044217	123	051105	040511	SERIAL: .ASCIZ /SERIAL NUMBER /
8160	044224	020114	052516	041115	
8161	044232	051105	000040		
8162					
8163	044236	005015	051012	052117	MSG7: .ASCIZ <CR><LF><LF>/ROTATIONAL SPEED TIMES/
8164	044244	052101	047511	040516	
8165	044252	020114	050123	042505	
8166	044260	020104	044524	042515	
8167	044266	000123			
8168	044270	005015	047412	042516	MSG10A: .ASCIZ <CR><LF><LF>/ONE CYLINDER SEEK TIMES/<CR><LF>/ * FORWARD/
8169	044276	041440	046131	047111	
8170	044304	042504	020122	042523	
8171	044312	045505	052040	046511	
8172	044320	051505	005015	025040	
8173	044326	043040	051117	040527	
8174	044334	042122	000		
8175	044337	015	020012	020052	MSG10B: .ASCIZ <CR><LF>/ * REVERSE/
8176	044344	042522	042526	051522	
8177	044352	000105			
8178	044354	005015	040412	041503	MSG11A: .ASCIZ <CR><LF><LF>/ACCESS TIME MEASUREMENT/<CR><LF>/ * FORWARD/
8179	044362	051505	020123	044524	
8180	044370	042515	046440	040505	
8181	044376	052523	042522	042515	
8182	044404	052116	005015	025040	
8183	044412	043040	051117	040527	
8184	044420	042122	000		
8185	044423	015	020012	020052	MSG11B: .ASCIZ <CR><LF>/ * REVERSE/
8186	044430	042522	042526	051522	
8187	044436	000105			
8188	044440	005015	046412	054101	MSG12A: .ASCIZ <CR><LF><LF>/MAXIMUM SEEK TIMES/<CR><LF>/ * FORWARD/
8189	044446	046511	046525	051440	
8190	044454	042505	020113	044524	
8191	044462	042515	006523	020012	
8192	044470	020052	047506	053522	
8193	044476	051101	000104		
8194	044502	005015	025040	051040	MSG12B: .ASCIZ <CR><LF>/ * REVERSE/
8195	044510	053105	051105	042523	
8196	044516	000			
8197					
8198	044517	015	046412	047111	MSGMIN: .ASCIZ <CR><LF>/MIN= /
8199	044524	000075			
8200	044526	005015	040515	036530	MSGMAX: .ASCIZ <CR><LF>/MAX= /
8201	044534	000			
8202	044535	015	040412	043526	MSGAVG: .ASCIZ <CR><LF>/AVG= /
8203	044542	000075			
8204	044544	020060	051525	000	MSGOUS: .ASCIZ /O US/
8205	044551	040	042502	047514	MBELOW: .ASCIZ / BELOW THE MINIMUM OF /
8206	044556	020127	044124	020105	
8207	044564	044515	044516	052515	
8208	044572	020115	043117	000040	
8209	044600	040440	047502	042526	MABOVE: .ASCIZ / ABOVE THE MAXIMUM OF /
8210	044606	052040	042510	046440	
8211	044614	054101	046511	046525	
8212	044622	047440	020106	000	
8213	044627	040	042523	045505	MSGNUM: .ASCIZ / SEEKS TIMED/
8214	044634	020123	044524	042515	

8215	044642	000104				
8216	044644	047040	052117	052040	MSGNON: .ASCIZ / NOT TIMED/	
8217	044652	046511	042105	000		
8218	044657	040	000040		MSG.SP: .ASCIZ / / ;TWO (2) SPACES	
8219						
8220					.SBTTL ERROR HEADER (EM) MESSAGES	
8221						
8222	044662	044122	030067	044440	EM1: .ASCIZ /RH70 INTERRUPT OCCURRED (RMAS = 0)/	
8223	044670	052116	051105	052522		
8224	044676	052120	047440	041503		
8225	044704	051125	042522	020104		
8226	044712	051050	040515	020123		
8227	044720	020075	024460	000		
8228	044725	125	042516	050130	EM2: .ASCIZ /UNEXPECTED ATTENTION OCCURRED/	
8229	044732	041505	042524	020104		
8230	044740	052101	042524	052116		
8231	044746	047511	020116	041517		
8232	044754	052503	051122	042105		
8233	044762	000				
8234	044763	115	051501	041123	EM3: .ASCIZ /MASSBUS PARITY ERROR(MCPE=1)/	
8235	044770	051525	050040	051101		
8236	044776	052111	020131	051105		
8237	045004	047522	024122	041515		
8238	045012	042520	030475	000051		
8239	045020	040515	051523	052502	EM4: .ASCIZ /MASSBUS PARITY ERROR(PAR=1)/	
8240	045026	020123	040520	044522		
8241	045034	054524	042440	051122		
8242	045042	051117	050050	051101		
8243	045050	030475	000051			
8244	045054	042101	051104	051505	EM5: .ASCIZ /ADDRESS PLUG CHANGE BIT SET/	
8245	045062	020123	046120	043525		
8246	045070	041440	040510	043516		
8247	045076	020105	044502	020124		
8248	045104	042523	000124			
8249	045110	044122	030067	051057	EM10: .ASCIZ "RH70/RMO3 FAILED TO RESPOND TO ADDRESSING"	
8250	045116	030115	020063	040506		
8251	045124	046111	042105	052040		
8252	045132	020117	042522	050123		
8253	045140	047117	020104	047524		
8254	045146	040440	042104	042522		
8255	045154	051523	047111	000107		
8256	045162	051104	053111	020105	EM11: .ASCIZ /DRIVE SELECTED IS NOT ONLINE/	
8257	045170	042523	042514	052103		
8258	045176	042105	044440	020123		
8259	045204	047516	020124	07117		
8260	045212	044514	042516	000		
8261	045217	111	050115	047522	EM12: .ASCIZ /IMPROPER HEADER DATA/	
8262	045224	042520	020122	042510		
8263	045232	042101	051105	042040		
8264	045240	052101	000101			
8265	045244	040504	040524	041440	EM13: .ASCIZ /DATA COMPARE FAILURE/	
8266	045252	046517	040520	042522		
8267	045260	043040	044501	052514		
8268	045266	042522	000			
8269	045271	104	051511	020113	EM17: .ASCIZ /DISK ERROR IN TIMING TEST/	
8270	045276	051105	047522	020122		

8271	045304	047111	052040	046511	
8272	045312	047111	020107	042524	
8273	045320	052123	000		
8274	045323	103	047514	045503	EM20: .ASCIZ /CLOCK (KW11-P) OVERFLOW IN TIMING TEST/
8275	045330	024040	053513	030461	
8276	045337	050055	020051	053117	
8277	045344	051105	046106	053517	
8278	045352	044440	020116	044524	
8279	045360	044515	043516	052040	
8280	045366	051505	000124		
8281	045372	044504	045523	042440	EM23: .ASCIZ /DISK ERROR DURING SEEK/
8282	045400	051122	051117	042040	
8283	045406	051125	047111	020107	
8284	045414	042523	045505	000	
8285	045421	123	042505	020113	EM24: .ASCIZ /SEEK NOT COMPLETE WITHIN 120 MS/
8286	045426	047516	020124	047503	
8287	045434	050115	042514	042524	
8288	045442	053440	052111	044510	
8289	045450	020116	031061	020060	
8290	045456	051515	000		
8291	045461	122	033510	027460	EM41: .ASCIZ "RH70/RMO3 ERROR"
8292	045466	046522	031460	042440	
8293	045474	051122	051117	000	
8294	045501	106	052101	046101	EM46: .ASCIZ /FATAL WRITE CHECK ERROR/
8295	045506	053440	044522	042524	
8296	045514	041440	042510	045503	
8297	045522	042440	051122	051117	
8298	045530	000			
8299					
8300					.SBTTL STATUS/ERROR INDICATOR MESSAGES
8301					
8302	045531	117	043106	044514	MSG814: .ASCIZ /OFFLINE OR UNSAFE DRIVE REQUESTED/
8303	045536	042516	047440	020122	
8304	045544	047125	040523	042506	
8305	045552	042040	044522	042526	
8306	045560	051040	050505	42525	
8307	045566	052123	042105	000	
8308	045573	125	046116	040517	MSG813: .ASCIZ /UNLOADED DRIVE REQUESTED/
8309	045600	042504	020104	051104	
8310	045606	053111	020105	042522	
8311	045614	052521	051505	042524	
8312	045622	000104			
8313	045624	042520	051522	051511	MSG812: .ASCIZ /PERSISTENT UNSAFE/
8314	045632	042524	052116	052440	
8315	045640	051516	043101	000105	
8316	045646	040520	044522	054524	MSG811: .ASCIZ /PARITY ERROR OCCURRED/
8317	045654	042440	051122	051117	
8318	045662	047440	041503	051125	
8319	045670	042522	000104		
8320	045674	040506	040524	020114	MSG810: .ASCIZ /FATAL PARITY ERROR/
8321	045702	040520	044522	054524	
8322	045710	042440	051122	051117	
8323	045716	000			
8324	045717	123	043117	053524	MSG809: .ASCIZ /SOFTWARE TIMEOUT ON THIS DRIVE/
8325	045724	051101	020105	044524	
8326	045732	042515	052517	020124	

8327	045740	047117	052040	044510	
8328	045746	020123	051104	053111	
8329	045754	000105			
8330	045756	047523	052106	040527	MSGB08: .ASCIZ /SOFTWARE TIMEOUT ON ANOTHER DRIVE/
8331	045764	042522	052040	046511	
8332	045772	047505	052125	047440	
8333	046000	020116	047101	052117	
8334	046006	042510	020122	051104	
8335	046014	053111	000105		
8336	046020	051105	047522	020122	MSGB06: .ASCIZ "ERROR OCCURRED DURING I/O OPERATION"
8337	046026	041517	052503	051122	
8338	046034	042105	042040	051125	
8339	046042	047111	020107	027511	
8340	046050	020117	050117	051105	
8341	046056	052101	047511	000116	
8342	046064	051105	047522	020122	MSGB05: .ASCIZ "ERROR OCCURRED DURING NON-I/O OPERATION"
8343	046072	041517	052503	051122	
8344	046100	042105	040040	051125	
8345	046106	047111	020107	047516	
8346	046114	026516	027511	020117	
8347	046122	050117	051105	052101	
8348	046130	047511	000116		
8349	046134	047125	040523	042506	MSGB04: .ASCIZ /UNSAFE OCCURRED/
8350	046142	047440	040033	051125	
8351	046150	042522	000104		
8352	046154	052501	047524	040515	MSGB03: .ASCIZ /AUTOMATIC RECALIBRATE SEQUENCE OCCURRED/
8353	046162	044524	020103	042522	
8354	046170	040503	044514	051102	
8355	046176	052101	020105	042523	
8356	046204	052521	047105	042503	
8357	046212	047440	041503	051125	
8358	046220	042522	000104		
8359	046224	051104	053111	020105	MSGB02: .ASCIZ /DRIVE HAS NOT RESPONDED TO PORT REQUEST/
8360	046232	040510	020123	047516	
8361	046240	020124	042522	050123	
8362	046246	047117	042504	020104	
8363	046254	047524	050040	051117	
8364	046262	020124	042522	052521	
8365	046270	051505	000124		
8366	046274	051104	053111	020105	MSGB01: .ASCIZ /DRIVE HAS BECOME NON-EXISTENT/
8367	046302	040510	020123	042502	
8368	046310	047503	042515	047040	
8369	046316	047117	042455	044530	
8370	046324	052123	047105	000124	
8371					
8372					.SBTTL DATA HEADER (DT) MESSAGES
8373					
8374	046332	051105	020122	041520	DH1: .ASCIZ /ERR PC RMAS/
8375	046340	020040	046522	051501	
8376	046346	000			
8377	046347	105	051122	050040	DH2: .ASCIZ /ERR PC DRIVE RMAS RMD5 RMER1 RMMR2 RMER2/
8378	046354	020103	042040	044522	
8379	046362	042526	020040	051040	
8380	046370	040515	020123	020040	
8381	046376	051040	042115	020123	
8382	046404	020040	046522	051105	



Line No	Time	Code	Time	Code	Msg	Time	Code	Time	Code	Msg	
8439	047106	000									
8440	047107	124	051505	020124	DH21: .ASCIZ /TEST	ERR PC	TST PC	DRIVE	CYLNR	TRACK/	
8441	047114	020040	042440	051122							
8442	047122	050040	020103	052040							
8443	047130	052123	050040	020103							
8444	047136	042040	044522	042526							
8445	047144	020040	041440	046131							
8446	047152	042116	020122	052040							
8447	047160	040522	045503	000							
8448	047165	107	042104	052101	DH21A: .ASCIZ /GDDAT	BDDAT	WRDNT	SECTUR/			
8449	047172	020040	041040	042104							
8450	047200	052101	020040	053440							
8451	047206	042122	047103	020124							
8452	047214	051440	041505	047524							
8453	047222	000122									
8454	047224	042524	052123	020040	DH23: .ASCIZ /TEST	ERR PC	DRIVE	CYLNR	RMCS1	RMCS2	RMDS/
8455	047232	020040	051105	020122							
8456	047240	041520	020040	051104							
8457	047246	053111	020105	020040							
8458	047254	054503	047114	051104							
8459	047262	020040	046522	051503							
8460	047270	020061	020040	046522							
8461	047276	051503	020062	020040							
8462	047304	046522	051504	000							
8463	047311	122	042515	030522	DH23A: .ASCIZ /RMER1	RMMR2	RMER2	RMDC	RMHR/		
8464	047316	020040	051040	046515							
8465	047324	031122	020040	051040							
8466	047332	042515	031122	020040							
8467	047340	051040	042115	020103							
8468	047346	020040	051040	044115							
8469	047354	000122									
8470	047356	042524	052123	020040	DH41: .ASCIZ /TEST	ERR PC	TST PC	DRIVE/			
8471	047364	020040	051105	020122							
8472	047372	041520	020040	051524							
8473	047400	020124	041520	020040							
8474	047406	051104	053111	000105							
8475	047414	042524	052123	020040	DH42: .ASCIZ /TEST	ERR PC	TST PC	DRIVE	RMCS1	RMCS2	RMDS/
8476	047422	020040	051105	020122							
8477	047430	041520	020040	051524							
8478	047436	020124	041520	020040							
8479	047444	051104	053111	020105							
8480	047452	020040	046522	051503							
8481	047460	020061	020040	046522							
8482	047466	051503	020062	020040							
8483	047474	046522	051504	000							
8484	047501	122	042515	030522	DH43A: .ASCIZ /RMER1	RMMR2	RMER2/				
8485	047506	020040	051040	046515							
8486	047514	031122	020040	051040							
8487	047522	042515	031122	000							
8488	047527	122	041515	030523	DH44A: .ASCIZ /RMCS1	RMCS2	RMDS	RMHR	RMDC	RMDA/	
8489	047534	020040	051040	041515							
8490	047542	031123	020040	051040							
8491	047550	042115	020123	020040							
8492	047556	046522	051110	020040							
8493	047564	020040	046522	041504							
8494	047572	020040	020040	046522							



8495	047600	040504	000
8496	047603	122	042515
8497	047610	020040	051040
8498	047616	031122	020040
8499	047624	042515	031122
8500	047631	122	042515
8501	047636	020040	051040
8502	047644	031122	020040
8503	047652	042515	031122
8504	047660	051040	053515
8505	047666	020040	051040
8506	047674	020101	020040
8507	047702	042115	000102

DH44B: .ASCIZ /RMER1 RMMR2 RMER2/

DH45A: .ASCIZ /RMER1 RMMR2 RMER2 RMWC RMBA RMOB/

.EVEN

.SBTTL DATA TABLE (DT)

8514	047706	001116	001170
8515	047712	001116	001164
8516	047720	035346	035350
8517	047726	035354	
8518	047730	001176	001116
8519	047736	042340	
8520	047740	001176	001116
8521	047746	042560	042340
8522	047752	001176	001116
8523	047760	001174	035346
8524	047766	035352	035354
8525	047772	001366	001116
8526	047776	001166	001116
8527	050002	001176	001116
8528	050010	001254	001270
8529	050016	001272	
8530	050020	001270	001274
8531	050026	001262	001264
8532	050034	001176	001116
8533	050042	001254	001270
8534	050050	001272	
8535	050052	001124	001126
8536	050060	001120	001122
8537	050064	001176	001116
8538	050072	004204	004216
8539	050100	004244	004246
8540	050104	001176	001116
8541	050112	001254	001270
8542	050120	001164	001126
8543	050126	001164	
8544	050130	001176	001116
8545	050136	001270	004204
8546	050144	004216	
8547	050146	004220	004244
8548	050154	004240	004242
8549	050160	001176	001116
8550	050166	001254	

DT1: .WORD \$ERRPC,\$REG3  
DT2: .WORD \$ERRPC,\$REG1,\$REG3, RMERRS, RMERRS+2, RMERRS+4, RMERRS+6

DT3: .WORD \$TMPO,\$ERRPC, RD.ADR, RD.WRD

DT4: .WORD \$TMPO,\$ERRPC, WRT.ADR, WRT.WD, RD.WRD

DT5: .WORD \$TMPO,\$ERRPC,\$REG1,\$REG5, RMERRS, RMERRS+2, RMERRS+4, RMERRS+6

DT10: .WORD RH.ADR,\$ERRPC  
DT11: .WORD \$REG2,\$ERRPC  
DT12: .WORD \$TMPO,\$ERRPC,\$REG0,CHKDRV,CYL.DS,TRK.DS,SEC.DS

DT12A: .WORD CYL.DS,TRK.DS,SEC.DS,CYL.RD,TRK.RD,SEC.RD

DT13: .WORD \$TMPO,\$ERRPC,\$REG0,CHKDRV,CYL.DS,TRK.DS,SEC.DS

DT13A: .WORD \$GDDAT,\$BDDAT,\$REG4,\$GDADR,\$BDAOR

DT17: .WORD \$TMPO,\$ERRPC,CHKDRV, RM.REG, RM.REG+12, RM.REG+14, RM.REG+40, RM.REG+42

DT21: .WORD \$TMPO,\$ERRPC,\$REG0,CHKDRV,CYL.DS,TRK.DS

DT21A: .WORD \$REG1,\$BDDAT,\$REG4,\$REG1

DT23: .WORD \$TMPO,\$ERRPC,CHKDRV,CYL.DS, RM.REG, RM.REG+10, RM.REG+12

DT23A: .WORD RM.REG+14, RM.REG+40, RM.REG+42, RM.REG+34, RM.REG+36

DT41: .WORD \$TMPO,\$ERRPC,\$REG0,CHKDRV

8551	050170	001176	001116	001162	DT42:	.WORD	\$TMPO,\$ERRPC,\$REGO,CHKDRV, RM.REG, RM.REG+10, RM.REG+12
8552	050176	001254	004204	004214			
8553	050204	004216					
8554	050206	001176	001116	001162	DT43:	.WORD	\$TMPO,\$ERRPC,\$REGO,CHKDRV, RM.REG, RM.REG+10, RM.REG+12
8555	050214	001254	004204	004214			
8556	050222	004216					
8557	050224	004220	004244	004246	DT43A:	.WORD	RM.REG+14, RM.REG+40, RM.REG+42
8558	050232	001176	001116	001162	DT44:	.WORD	\$TMPO,\$ERRPC,\$REGO,CHKDRV,CYL.DS, TRK.DS, SEC.DS
8559	050240	001254	001270	001274			
8560	050246	001272					
8561	050250	004204	004214	004216	DT44A:	.WORD	RM.REG, RM.REG+10, RM.REG+12, RM.REG+36, RM.REG+34, RM.REG+06
8562	050256	004242	004240	004212			
8563	050264	004220	004244	004246	DT44B:	.WORD	RM.REG+14, RM.REG+40, RM.REG+42
8564	050272	001176	001116	001162	DT45:	.WORD	\$TMPO,\$ERRPC,\$REGO,CHKDRV,CYL.DS, TRK.DS, SEC.DS
8565	050300	001254	001270	001274			
8566	050306	001272					
8567	050310	004204	004214	004216	DT45A:	.WORD	RM.REG, RM.REG+10, RM.REG+12, RM.REG+36, RM.REG+34, RM.REG+06
8568	050316	004242	004240	004212			
8569	050324	004220	004244	004246	DT45B:	.WORD	RM.REG+14, RM.REG+40, RM.REG+42, RM.REG+2, RM.REG+4, RM.REG+22
8570	050332	004206	004210	004226			

.SBTTL DATA FORMAT (DF) TABLE

8571							
8572							
8573							
8574	050340	000001			DF1:	.WORD 1	;NUMBER OF DATA HEADERS ;NUMBER OF WORDS IN DATA TABLE ;ALL 3 NUMBERS ARE OCTAL
8575	050342	002			.BYTE 2		
8576	050343	000			.BYTE 0		
8577							
8578	050344	000001			DF2:	.WORD 1	
8579	050346	007			.BYTE 7		
8580	050347	000			.BYTE 0		
8581							
8582	050350	000001			DF3:	.WORD 1	
8583	050352	004			.BYTE 4		
8584	050353	000			.BYTE 0		
8585							
8586	050354	000001			DF4:	.WORD 1	
8587	050356	005			.BYTE 5		
8588	050357	000			.BYTE 0		
8589							
8590	050360	000001			DF10:	.WORD 1	
8591	050362	002			.BYTE 2		
8592	050363	000			.BYTE 0		
8593							
8594	050364	000001			DF11:	.WORD 1	
8595	050366	002			.BYTE 2		
8596	050367	000			.BYTE 0		
8597							
8598	050370	000002			DF12:	.WORD 2	;2 DH'S TO BE TYPED ;7 DATA WORDS FOLLOW THE 1ST DH ;WORDS 1-4 ARE OCTAL 5-7 ARE DECIMAL ;ADDRESS OF 2ND DH ;6 DATA WORDS FOLLOW THE 2ND DH ;ALL WORDS ARE OCTAL
8599	050372	007			.BYTE 7		
8600	050373	160			.BYTE 160		
8601	050374	046665			.WORD DH12A		
8602	050376	006			.BYTE 6		
8603	050377	000			.BYTE 0		
8604							
8605	050400	000002			DF13:	.WORD 2	
8606	050402	007			.BYTE 7		

8607	050403	160		.BYTE	160	
8608	050404	046744		.WORD	DH13A	
8609	050406	005		.BYTE	5	
8610	050407	004		.BYTE	4	;WORD 3 IS DECIMAL
8611						
8612	050410	000000	DF14:	.WORD	0	
8613	050412	005		.BYTE	5	
8614	050413	004		.BYTE	4	;WORD 3 IS DECIMAL
8615						
8616	050414	000001	DF17:	.WORD	1	
8617	050416	010		.BYTE	↑08	
8618	050417	000		.BYTE	0	
8619						
8620	050420	000002	DF21:	.WORD	2	
8621	050422	006		.BYTE	6	
8622	050423	060		.BYTE	60	
8623	050424	047165		.WORD	DH21A	
8624	050426	004		.BYTE	4	
8625	050427	014		.BYTE	14	
8626						
8627	050430	000000	DF22:	.WORD	0	
8628	050432	004		.BYTE	4	
8629	050433	014		.BYTE	14	
8630						
8631	050434	000002	DF23:	.WORD	2	
8632	050436	007		.BYTE	7	
8633	050437	010		.BYTE	10	;WORD 4 IS DECIMAL
8634	050440	047311		.WORD	DH23A	
8635	050442	005		.BYTE	5	
8636	050443	000		.BYTE	0	
8637						
8638						
8639	050444	000001	DF41:	.WORD	1	
8640	050446	004		.BYTE	4	
8641	050447	000		.BYTE	0	
8642						
8643	050450	000001	DF42:	.WORD	1	
8644	050452	007		.BYTE	7	
8645	050453	000		.BYTE	0	
8646						
8647	050454	000002	DF43:	.WORD	2	
8648	050456	007		.BYTE	7	
8649	050457	000		.BYTE	0	
8650	050460	047501		.WORD	DH43A	
8651	050462	003		.BYTE	3	
8652	050463	000		.BYTE	0	
8653						
8654	050464	000003	DF44:	.WORD	3	
8655	050466	007		.BYTE	7	
8656	050467	160		.BYTE	160	
8657	050470	047527		.WORD	DH44A	
8658	050472	006		.BYTE	6	
8659	050473	000		.BYTE	0	
8660	050474	047603		.WORD	DH44B	
8661	050476	003		.BYTE	3	
8662	050477	000		.BYTE	0	

8663				
8664	050500	000003		
8665	050502	007		
8666	050503	160		
8667	050504	047527		
8668	050506	00E		
8669	050507	000		
8670	050510	047631		
8671	050512	006		
8672	050513	000		
8673				
8674				
8675		050514		
8676				
8677	050514	005015	046412	044501
8678	050522	042116	041505	030455
8679	050530	026461	055104	046522
8680	050536	026506	006501	012
8681	050543	122	030115	020063
8682	050550	054105	042524	042116
8683	050556	042105	042040	044522
8684	050564	042526	052040	051505
8685	050572	006524	012	
8686	050575	112	046125	026531
8687	050602	034461	033467	005015
8688	050610	000012		
8689	050612	005015	047524	052040
8690	050620	051505	020124	051104
8691	050626	053111	020105	020060
8692	050634	042522	046120	041501
8693	050642	020105	044124	020105
8694	050650	054047	042130	023520
8695	050656	050040	041501	020113
8696	050664	047117	042040	044522
8697	050672	042526	030040	005015
8698	050700	044527	044124	040440
8699	050706	047516	044124	051105
8700	050714	050040	041501	026113
8701	050722	041440	042514	051101
8702	050730	046440	046505	051117
8703	050736	020131	047514	040503
8704	050744	044524	047117	032040
8705	050752	026060	040440	042116
8706	050760	051040	051505	040524
8707	050766	052122	005015	
8708	050772	044124	020105	051120
8709	051000	043517	040522	006515
8710	051006	000012		
8711	051010	005015	054523	052123
8712	051016	046505	044040	051501
8713	051024	030440	045466	046440
8714	051032	046505	051117	026131
8715	051040	023440	054130	050104
8716	051046	020047	047514	042101
8717	051054	051105	053440	046111
8718	051062	020114	042502	047440

DF45: .WORD 3  
 .BYTE 7  
 .BYTE 160  
 .WORD DH44A  
 .BYTE 6  
 .BYTE 0  
 .WORD DH45A  
 .BYTE 6  
 .BYTE 0

.EVEN  
BUFFER=.

TITLE: .ASCII <CR><LF><LF>/MAINDEC-11-DZRMF-A/<CR><LF>

.ASCII RMO3 EXTENDED DRIVE TEST<CR><LF>

.ASCIIZ /JULY-1977/<CR><LF><LF>

LOADRV: .ASCII <CR><LF>/TO TEST DRIVE 0 REPLACE THE 'XXDP' PACK ON DRIVE 0/<CR><LF>

.ASCII /WITH ANOTHER PACK, CLEAR MEMORY LOCATION 40, AND RESTART/<CR><LF>

.ASCIIZ /THE PROGRAM/<CR><LF>

NLOAD: .ASCIIZ <CR><LF>/SYSTEM HAS 16K MEMORY, 'XXDP' LOADER WILL BE OVERWRITTEN/<CR><L

```

8719 051070 042526 053522 044522
8720 051076 052124 047105 005015
8721 051104 000012
8722
8723
8724
8725
8726
8727
8728
8729
8730
8731
8732
8733 051106 010046
8734 051110 010146
8735 051112 013746 000004
8736 051116 013746 000006
8737 051122 010600
8738
8739 051124 104400
8740 051126 012637 000006
8741 051132 012737 051152 000004
8742 051140 012701 020000
8743 051144 005711
8744 051146 005721
8745 051150 000775
8746 051152 162701 000002
8747 051156 010006
8748 051160 012637 000006
8749 051164 012637 000004
8750 051170 010137 051202
8751 051174 012601
8752 051176 012600
8753 051200 000207
8754 051202 000000
8755
8756
8757
8758
8759
8760
8761
8762
8763
8764
8765
8766
8767
8768 051204 005737 001226
8769 051210 001427
8770 051212 005037 001226
8771 051216 012700 001366
8772 051222 012703 051362
8773 051226 011004
8774 051230 004037 032516

```

```

.EVEN
.SBTTL ROUTINE TO SIZE MEMORY
;*****
;CALL:
;* JSR PC,$SIZE
;* RETURN
;*$LSTAD WILL CONTAIN THE LAST AVAILABLE MEMORY LOCATION
$SIZE: MOV RO,-(SP) ;;SAVE RO ON THE STACK
MOV R1,-(SP) ;;SAVE R1 ON THE STACK
MOV @#ERRVEC,-(SP) ;;SAVE PRESENT ERROR VECTOR PS & PC
MOV @#ERRVEC+2,-(SP)
MOV SP,RO ;;SAVE THE STACK POINTER
;;SET THE ERRVEC PS TO THE PRESENT PS
TRAP ;;PUSH OLD PSW AND PC ON STACK
MOV (SP)+,@#ERRVEC+2 ;;SAVE THE PSW IN @#ERRVEC+2
MOV #2$,@#ERRVEC ;;SET FOR TIMEOUT
MOV #20000,R1 ;;FIRST ADDRESS
1$: TST (R1) ;;TEST THIS ADDRESS
TST (R1)+ ;;STEP TO NEXT ADDRESS
BR 1$ ;;TRY ANOTHER
2$: SUB #2,R1 ;;DROP BACK
MOV RO,SP ;;RESTORE THE STACK
MOV (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
MOV (SP)+,@#ERRVEC
MOV R1,$LSTAD ;;LAST ADDRESS
MOV (SP)+,R1 ;;RESTORE R1
MOV (SP)+,RO ;;RESTORE RO
RTS PC
$*LSTAD: .WORD 0 ;;CONTAINS THE LAST ADDRESS
;*****
.SBTTL GETADR - GET BUS ADDRESS AND VECTOR ADDRESS
;THIS ROUTINE IS USED TO ENSURE THE BUS ADDRESS
;OF THE RM70/RMO3 IS SETUP TO READ THE PROPER VALUE.
;IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
;REQUIRED.
;NOTE: THIS ROUTINE DESTROYS RO-R4
;CALL
; JSR PC,@#GETADR
; RETURN
GETADR: TST @#BUSADR ;;INPUT FROM TTY REQUESTED?
BEQ 7$ ;;NO--BRANCH
CLR @#BUSADR ;;YES--CLEAR THE REQUEST FLAG
1$: MOV #RH.ADR,RO ;;FIRST ADDRESS
MOV #RMCS1,R3 ;;RMCS1=
MOV (RO),R4 ;;PRESENT RMCS1 ADDRESS
JSR RO,@#GETNUM ;;GET NEW RMCS1

```

8775	051234	000402		BR	2\$	:	COMMA
8776	051236	000767		BR	1\$	:	PERIOD
8777	051240	000412		BR	5\$	:	DOUBLE PERIOD
8778	051242	010420		2\$:	MOV	R4,(R0)+	:SAVE NEW RMCS1
8779	051244	012703	051373		MOV	#MRHVEC,R3	: "RHVEC="
8780	051250	011004			MOV	(R0),R4	:PRESENT RH70 VECTOR ADDRESS
8781	051252	004037	032516		JSR	R0,@GETNUM	:GET NEW RHVEC
8782	051256	000402			BR	3\$	:
8783	051260	000756			BR	1\$	:
8784	051262	000401			BR	5\$	:
8785	051264	010420		3\$:	MOV	R4,(R0)+	:SAVE NEW RHVEC
8786	051266	010410		5\$:	MOV	R4,(R0)	:SAVE INPUT
8787	051270	013701	000004	7\$:	MOV	@ERRVEC,R1	:SAVE THE ERROR VECTOR
8788	051274	012737	051330	000004	MOV	#R5,@ERRVEC	:SETUP FOR TRAP
8789	051302	005777	130060		TST	@RH.ADR	:CHECK FOR RH70/RMO3
8790	051306	010137	000004		MOV	R1,@ERRVEC	:RESTORE ERROR VECTOR
8791	051312	012700	001366		MOV	#RH.ADR,R0	:FIRST ADDRESS OF NEW PARAMETERS
8792	051316	012701	035514		MOV	#RMAADR,R1	:FIRST ADDRESS OF WHERE TO PUT THEM
8793	051322	012021			MOV	(R0)+,(R1)+	:BUS ADDRESS
8794	051324	012021			MOV	(R0)+,(R1)+	:VECTOR ADDRESS
8795	051326	000207			RTS	PC	:RETURN
8796	051330	010137	000004	8\$:	MOV	R1,@ERRVEC	:RESTORE ERROR VECTOR
8797	051334	022626			CMP	(SP)+,(SP)+	:CLEAN OFF THE STACK
8798	051336	104010			ERROR	10	:REPORT THE ERROR
8799	051340	005737	000042		TST	@#42	:IS THERE A MONITOR?
8800	051344	001724			BEQ	1\$	:NO--GO ASK FOR ADDRESS
8801	051346	005037	001232		CLR	@DRVSEL	:YES -NO DRIVES SELECTED
8802	051352	005037	020562		CLR	@SEOPCT	:NO PASSES
8803	051356	000137	020406		JMP	@SEOP	:GO TO END OF PROGRAM
8804							
8805	051362	005015	046522	051503	MRMCS1:	.ASCIZ	<CR><LF>/RMCS1=/ 000
8806	051370	036461	000				
8807	051373	015	051012	053110	MRHVEC:	.ASCIZ	<CR><LF>/RHVEC=/ 000075
8808	051400	041505	000075				
8809		000001			.END		

ACTDRV	035432	6810#	7078*	7129*	7447*	7456*	7694								
ACTSTR	035433	6816#	7696*	7710*											
AOE	= 001000	284#													
ATA	= 100000	271#													
ATABIT	035502	1849	6158	6164	6882#	7045	7141	7247	7617	7636	7644	7645	7665	7751	
ATO	= 000001	332#													
AT1	= 000002	333#													
AT2	= 000004	334#													
AT3	= 000010	335#													
AT4	= 000020	336#													
AT5	= 000040	337#													
AT6	= 000100	338#													
AT7	= 000200	339#													
A16	= 000400	207#													
A17	= 001000	208#													
BAI	= 000010	225#													
BITS	001424	576#	2012	2061	2114	2172	2227	2322	2380	2445	2549	2721	2802	2921	
		3026	3139	3268	3354	3498	3572	3658	3771	6225	6256	6273	6326	6500	
BIT0	= 000001	182#													
BIT00	= 000001	172#	182	576	592	1837	1859	1914	4972						
BIT01	= 000002	171#	181	577	593	5298	5326	7121	7369						
BIT02	= 000004	170#	180	578	594	1823	5035	5298	5329						
BIT03	= 000010	169#	179	579	595	5335	7576	7869							
BIT04	= 000020	168#	178	580	596	5332	7611								
BIT05	= 000040	167#	177	581	597	2607	2623	2699	2859	2873	2888	2893	2973	2997	
		3002	3076	3094	3111	3116	3189	3207	3224	3229	5335	5445	6972	7389	
		7591	7746												
BIT06	= 000100	166#	176	582	598	5338	7054	7112	7186	7487	7930				
BIT07	= 000200	165#	175	583	599	7054	7321	7468	7576	7591	7611	7640	7902		
BIT08	= 000400	164#	174	584	5326	7054									
BIT09	= 001000	163#	173	585	3898	4520	5341	7744							
BIT1	= 000002	181#													
BIT10	= 002000	162#	586	3882	4984	5298	5329	6519	7371						
BIT11	= 004000	161#	587	4527	5329	7019	7148	7340	7680						
BIT12	= 010000	160#	588	5298	5332	7014	7036	7054	7123	7156	7336	7355	7367	7380	
		7601	7603	7807	7864	7931									
BIT13	= 020000	159#	589	3889	5298	5326	7106	7549	7810						
BIT14	= 040000	158#	590	2618	2854	2868	2968	3071	3089	3184	3202	4502	5226	5298	
		5332	7118	7153	7536	7676	7772	7819	7823	7928					
BIT15	= 100000	157#	591	5490	5491	7106	7118	7121	7123	7153	7156	7340	7369	7371	
		7487	7576	7591	7601	7611	7676	7744	7772	7779					
BIT2	= 000004	180#	7779												
BIT3	= 000010	179#													
BIT4	= 000020	178#													
BIT5	= 000040	177#													
BIT6	= 000100	176#													
BIT7	= 000200	175#													
BIT8	= 000400	174#													
BIT9	= 001000	173#													
BPTVEC	= 000014	189#													
BUFFER	= 050514	1239	1263	1287	1311	2584	3301	3316	3329	3387	3397	3411	3556	3689	
		3716	3726	3735	3743	3752	5409*	5410	5412	5414	5415	5416	5489	5526	
		5669	5688	5689	5713	5802	5834	5877	5919	5936	5937	5954	8675#		
BUSADR	001226	513#	1700*	1702*	1705*	1707*	8768	8770*							
BYPASS	001252	522#	1912*	2041*	2092*	2142*	2200*	2255*	2350*	2408*	2474*	2591*	2753*	3293*	
		3379*	3557*	3684*	5280	5760	5906	5971							























M16

MD-11-DZRMF-A RMO3 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 176  
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0207

RO.RM2	042460	7797*	7830#											
RO.RM3	042464	7808	7822	7832#										
RO.RM4	042470	7813	7834#											
RO.WRD	042340	7803#	7804	8518	8520									
READ =	000171	442#	3313	3618	3725	3742								
READHD =	000173	443#	2472	5064	5065	5135	5178							
READIN =	000121	435#												
RECAL =	000107	430#	1919	2037	5423	5448								
RELEAS =	000113	432#												
RESREG =	104413	2861	2875	2890	2975	2999	3078	3096	3113	3191	3209	3226	4011	4633#
		4693	5492	5677	5700	5762	5823	5874	6165	6317	6475	6479	6498	6521
		6991	7124	7127	7190	7395	7455	7709	7915	8014				
RESTAR	006052	1895#	1906	3851										
RESVEC =	000010	186#												
REX =	010000	306#												
RHVEC	001370	563#												
RH.ADR	001366	562#	8525	8771	8789	8791								
RMAOR	035514	2473	2589	5444	6899#	6971	7083	7207	7227	7249	7408	7449	7728	7799
		7805	7809	7825	7858	7862	7895	7925	8792					
RMA5 =	000016	6924#	7045*	7498	7617*	7644*	7665*	7750						
RMA8 =	000004	2654*	6919#											
RMCS1 =	000000	2611*	2652*	2663*	3067*	3085*	3180*	3198*	6917#	7013*	7019	7034	7147*	7148
		7223	7245	7287	7307	7318	7333	7344	7471	7501	7553	7585	7622	7680
		7853												
RMCS2 =	000010	2496*	2607*	2608*	2623*	2624*	2633*	2699*	2700*	2859*	2860*	2873*	2874*	2888*
		2889*	2893*	2894*	2973*	2974*	2997*	2998*	3002*	3003*	3076*	3077*	3094*	3095*
		3111*	3112*	3116*	3117*	3189*	3190*	3207*	3208*	3224*	3225*	3229*	3230*	5445*
		6921#	6972*	7012*	7014	7096*	7146*	7208*	7228*	7250*	7336	7355	7367	7380
		7389*	7409*	7469*	7567*	7651*	7664*	7729*	7746*	7806	7863	7896*	7902	7926*
		7931												
RMDA =	000006	2655*	2837*	5470*	5654	6920#	7215	7241	7256					
RMDB =	000022	6926#	7900											
RMDC =	000034	2609*	2834*	2963*	3066*	3084*	3179*	3197*	5471*	6931#	7219	7231	7263	
RMOS =	000012	2613	2618	2854	2868	2968	3071	3089	3184	3202	6922#	7041	7411	7569
		7594	7652	7731										
		6928#	7022											
RMOT =	000026	6935#												
RMEC1 =	000044	6936#	7910											
RMEC2 =	000046	6936#												
RMERRS	035346	6721#	6955	7536	7539	7549	7652*	7653*	7654*	7655*	8515	8522		
RMER1 =	000014	5127	5129	5170	5172	6923#	7047	7579	7653	7867				
RMER2 =	000042	6934#	7654											
RMHR =	000036	6932#												
RMINIT	035532	1788	6951#											
RMLA =	000020	2497	2634	6925#	7425	7427								
RMNR1 =	000024	6927#												
RMNR2 =	000040	6933#	7655											
RMOF =	000032	6930#	7038	7269	7273	7293	7297							
RMR =	000004	277#												
RMSN =	000030	6929#												
RMVEC	035516	2605*	2625*	2848*	2896*	2961*	3005*	3064*	3119*	3177*	3232*	5443*	6900#	6968
		6970	7077											
RMWC =	000002	2653*	6918#											
RM.REG	004204	1246	1270	1294	1318	1334#	1929	5127	5129	5170	5172	5226	5654	8537
		8544	8547	8551	8554	8557	8561	8563	8567	8569				
RM03	036236	5079	5110	5153	5200	5232	5251	5257	5450	7076#				
RPT	001506	605#	2020	2069	2122	2180	2235	2330	2388	2453	2557	2729	2810	2929

















I01

MD-11-DZRMF-A RMO3 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 184  
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0215

STYPOS 021670  
SXTSTR 023626  
SSGET4= 000000  
SOFILL 022113  
S4OCAT= \*\*\*\*\* U  
= 051404

4110#	4623													
4505#														
3844#														
4111#	4115*	4125	4160#											
3891	4502													
57#	61#	69	70#	72#	74#	77#	459#	507	1715	1729	1730	2144		
2152	2202	2206	2262	2284	2353	2412	2491	2494	2593	2631	3302	3307		
3310	3318	3325	3389	3399	3416	3422	3526	3597	3606	3616	3683	3695		
3720	3727	3737	3744	3753	3809#	3852	3853#	3908	3952#	4084	4229#	4234		
4238#	4239	4240	4477#	4478	4485#	4542	4543	4715#	6331#	7983#	7984#	7985#		
7986#	7987#	7988#	7989#	7990#	7991	8675								

CKCHR	1696#	6092	6142	6369	6406	6446	6609	6622	6633	6684					
CKDIG	1696#	6150	6388												
CKNUM	1696#	6100	6421	6454											
COMMEN	197#	1950	1985	2770	3239										
COMMND	1696#	1919	2037	2471	2575	3784	3789								
DO	1696#	1918	1920	2044	2045	2095	2096	2147	2155	2204	2209	2265	2271	2287	2293
	2356	2358	2417	2419	2421	2423	2425	2427	2493	2514	2755	2757	3785	3790	5424
DODTA	1696#	3305	3309	3314	3324	3331	3392	3403	3419	3425	3603	3609	3619	3697	3722
	3729	3739	3746	3755											
DRV. IN	1696#	2892	3001	3115	3228										
ENDCOM	197#	1978	1996	2783	3248										
ENDPAS	1696#	3837													
ERRCAL	6704#	7505	7556	7559	7814	7875									
ERREND	1696#	3905													
ERROR	91#	1904	2622	2698	2862	2876	2891	2976	3000	3079	3097	3114	3192	3210	3227
	5092	5093	5094	5095	5096	5123	5124	5125	5126	5131	5166	5167	5168	5169	5174
	5215	5216	5217	5218	5219	5245	5246	5247	5248	5249	5271	5272	5273	5274	5275
	5422	5463	5464	5465	5466	5467	5779	5784	5895	5901	5959	5960	7506	7557	7561
	7816	7876	8798												
ERRTYP	1696#	3873													
ER.NDX	1696#	5086	5117	5160	5209	5239	5265	5457							
ESCAPE	197#	2838	2954	3059	3172	5085	5116	5159	5207	5238	5264	5420	5456	5778	5782
	5894	5899	5947												
GETPRI	197#	4803	8739												
GETSWR	1#	197#	1770												
LOOP	1696#	2144	2152	2202	2206	2262	2284	2353	2412	2490	2494	2592	2630	3302	3307
	3310	3318	3325	3389	3399	3416	3422	3597	3606	3616	3695	3720	3727	3737	3744
	3753														
MORETA	453#	508													
MORE.S	1696#	2010	2059	2112	2170	2225	2320	2378	2443	2547	2719	2800	2919	3024	3137
	3266	3352	3496	3656	3769										
MSG	2001#	2003	2048#	2050	2099#	2101	2161#	2163	2215#	2217	2307#	2309	2367#	2369	2432#
	2434	2522#	2525	2706#	2708	2787#	2790	2908#	2910	3011#	3013	3124#	3126	3253#	3255
	3338#	3340	3434#	3436	3629#	3631	3760#	3762							
MULT	197#														
NEWTST	197#	2001	2048	2099	2161	2215	2307	2367	2432	2523	2706	2788	2908	3011	3124
	3253	3338	3434	3629	3760										
POP	197#	4216	4582	4771	4854	4954	5048								
PUSH	197#	4175	4562	4752	4806	4954	5012								
REPORT	197#	1696#	2899	2902	3006	3120	3233								
RMO3.D	1#	6704													
SAV. RH	1696#	2856	2870	2885	2970	2994	3073	3091	3108	3186	3204	3221			
SCOPE	92#	2046	2097	2159	2213	2305	2365	2430	2520	2704	2765	2905	3008	3122	3235
	3336	3431	3626	3756	3793										
SETPRI	197#	4406													
SETTRA	4613#	4622	4623	4624	4625	4627	4629	4630	4631	4632	4633				
SETUP	197#	1710													
SET. TN	1696#	2011	2060	2113	2171	2226	2321	2379	2444	2548	2720	2801	2920	3025	3138
	3267	3353	3497	3657	3770										
SKIP	197#														
SLASH	197#														
SPACE	197#														
STARS	67	197#	201	243	247	424	455	507	1598	2001	2009	2048	2058	2099	2111
	2161	2169	2215	2224	2307	2319	2367	2377	2432	2442	2523	2546	2706	2718	2788
	2799	2908	2918	3011	3023	3124	3136	3253	3265	3338	3351	3434	3495	3629	3655
	3760	3768	3799	3859	3909	4016	4087	4165	4233	4309	4324	4395	4419	4489	4546

K01

MD-11-DZRMF-A RMO3 EXTENDED DRIVE TEST  
DZRMFA.P11 22-JUL-77 14:59

MACY11 30(1046) 22-JUL-77 16:19 PAGE 187  
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0217

	4592	4637	4656	4719	4743	4780	6703	6713	8727	8756					
SWRSU	197#	1731#													
TRMTRP	4613#														
TSTTYP	12#	2022	2071	2124	2182	2237	2332	2390	2455	2559	2731	2812	2931	3036	3149
	3278	3364	3508	3668											
TYPBIN	197#														
TYPB2D	1696#	5560	5567	5571	5576	5583	5587	5599	5604	6363	6438				
TYP:EC	197#	3990													
TYPEND	1696#	3806													
TYPNAM	197#														
TYPNUM	197#														
TYPOCS	197#	1798	1871	1922	2029	2078	2131	2189	2244	2339	2397	2462	2566	2738	2819
	2938	3043	3156	3285	3371	3515	3675	3816	6332						
TYPOCT	197#	3824	3987	4337	6087	6401	6442								
TYPTXT	197#	3806	3812	3820	3837	3949	3966	6056	6136	6168	6328	6433			
\$\$CMRE	453#	492	493	494	495	496	497								
\$\$CMTM	453#	498	499	500											
\$\$ESCA	197#														
\$\$NEWT	197#	2001	2048	2099	2161	2215	2307	2367	2432	2523	2706	2788	2908	3011	3124
	3253	3338	3434	3629	3760										
\$\$SET	4613#	4622	4623	4624	4625	4627	4629	4630	4631	4632	4633				
\$\$SKIP	197#														
.EQUAT	1#	87													
.HEADE	1#														
.SETUP	1#	1700													
.SWRHI	1#	36													
.SWRLO	46#	47	48	49	50	51	52	53							
.\$ACT1	1#	65													
.\$CATC	1#	55													
.\$CMTA	1#	453													
.\$OB2D	1#	4654													
.\$DIV	1#	4778													
.\$EOP	1#	3797													
.\$ERRO	1#	3857													
.\$RAND	1#	4741													
.\$RDOE	1#														
.\$RDOC	1#														
.\$READ	1#	4231													
.\$SAVE	1#	4544													
.\$SB2D	1#	4635													
.\$SCOP	1#	4487													
.\$SIZE	1#	8725													
.\$SUPR	1#	4717													
.\$STRAP	1#	4590													
.\$STYPD	1#	4163													
.\$STYPE	1#	4014													
.\$STYPO	1#	4085													

. ABS. 051404 000

ERRORS DETECTED: 0

RMO3:DZRMFA,DZRMFA/NL:MD/SOL/CRF=RMO3:RMDRV4.P11,DZRMFA.P11  
RUN-TIME: 30 26 2 SECONDS

L01

MD-11-DZRMF-A RMD3 EXTENDED DRIVE TEST MACY11 30(1046) 22-JUL-77 16:19 PAGE 188  
DZRMFA.P11 22-JUL-77 14:59 CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0218

RUN-TIME RATIO: 649/59=10.8  
CORE USED: 45K (89 PAGES)