

RM03,02

EXTENDED DRIVE TEST
CZRMFB0

AH-B007B-MC

JAN 1978

COPYRIGHT © 1977

digital

FICHE 1 OF 2

MADE IN USA

This microfiche card contains 24 frames of data arranged in a 4x6 grid. Each frame displays a different set of data, likely test results or system logs, in a structured, tabular format. The data is too small to read clearly but appears to consist of multiple columns of text and numbers.

RM03,02

EXTENDED DRIVE TEST
CZRMFB0

AH-B007B-MC

JAN 1978

COPYRIGHT © 1977

digital

FICHE 2 OF 2

MADE IN USA

(Faint, illegible data tables visible on the left edge of the page)

801

EOF1CZRMFB0411

00010000

780105

PDP10 BIENTIF IPBDAICZM088EG

00010000

780105
SEQ 0001

PRODUCT CODE: AC-B0068-MC
PRODUCT NAME: CZRMFB0 RMO3/RMO2 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 RH7D 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. RH7D/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. REQUIREMENTS2.1 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,RMD2 DISKLESS CONTROLLER TEST	CZRMJB
RMD3,RMD2 FUNCTIONAL TEST I	CZRMCB
RMD3,RMD2 FUNCTIONAL TEST II	CZRMDB
RMD3,RMD2 FUNCTIONAL TEST III	CZRMEB

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 RH70 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 VAULE 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

RMCS1=176700 / 177200.

EXAMPLE #2

RMCS1=176700 / 176300<CR>
RHVEC=254 / 260<CR>
RHPRIO=5 / 6.

EXAMPLE #3

RMCS1=176700<CR>
RHVEC=254 / 260.

EXAMPLE #4

RH70/RM03 FAILED TO RESPOND TO ADDRESSING
RMCS1 ERR PC
176300 XXXXXX
RMCS1=176300 / 176700.

EXAMPLE #5

RMCS1=176700 / 1776\67\6300<CR>
RHVEC=254<CR>
RHPRIO=5<CR>
RMCS1=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

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

SPECIAL CASES OF CONTROL CHARACTERS

IF <..> IS TYPED WHILE A TEST IS OPEN FOR MODIFICATION
(</>) AND OTHER TESTS IN THE "TEST COMMAND" STRING
ARE TO BE 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                ;50 ITERATIONS, MOVE TO NEXT PARAMETER
  R=1 / 50<CR>         ;DO NOT CHANGE 'FC' CYLINDER ADDRESS BUT CONTINUE
  FC=0 / <CR>         ;TEST 10 IS STILL PENDING AND WILL BE
  LC=822 / 50.<CR>    ;RETAIN ITS PRESENT PARAMETERS.

```

EXAMPLE #4


```

-----
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             ;CHANGE TO 100 ITERATIONS, TO TO THE NEXT TEST
  R=10 / 100.<CR>   ;CHANGE 'R' TO 1000 ITERATIONS, MOVE TO NEXT TEST
  TEST 1             ;CHANGE 'R' TO 10 ITERATIONS, GO TO NEXT PARAMETER
  R=100 / 1000.<CR> ;CHANGE 'FC' TO 50, GO TO NEXT PARAMETER
  TEST 2             ;CHANGE 'LC' TO 51, GO TO THE NEXT TEST
  R=1 / 10<CR>
  FC=0 / 50<CR>
  LC=822 / 51.<CR>
  TEST 3             ;MOVE TO NEXT TEST
  R=1.<CR>
  TEST 4             ;USE TEST 4'S PARAMETERS AND START PROGRAM EXECUTION
  R=1.<CR>

EXAMPLE #7
-----
DRIVE=1<CR>
TEST=D/<CR>           ;SELECT AND OPEN THE DATA TEST
  TEST 20           ;DO 1000 ITERATION OF TEST PATTERN
  R=1 / 1000<CR>   ;#8 ON CYLINDER 10, TRACK 2, SECTOR 4
  FC=0 / 10<CR>
  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>
TEST=D/<CR>           ;USE THE SAME PARAMETERS AS IN EXAMPLE
  TEST 20           ;#7, BUT ALSO SPECIFY A DATA PATTERN (PAT #0).
  R=1000 / <CR>
  FC=10 / <CR>

```



```

LC=10 / <CR>
YC=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

E. 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 TYPEOUT WILL CONTAIN THE FOLLOWING:

SEG 0013

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

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

6.1 ERROR TYPES

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

6.1.1 DRIVER ERROR

THESE ERRORS WILL BE DETECTED BY THE RH70/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 PROPER 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

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 (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 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	-	100
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

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
133333	004000	173777	155555	177757	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
---	---	-------

FC - 0
LC - 821

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 DDU. THE PROGRAM STALLS APPROXIMATELY 5 SECONDS BETWEEN SEEKS SO THAT THE ACCESS TIME INDICATORS ON THE DDU 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: RMD3 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:

<u>DRVSTA</u>	<u>DRIVE STATE</u>
>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:

<u>DRVTYP</u>	<u>CONDITION</u>
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 REGISTER TABLE POINTER
	.BYTE	0	;(10) SECTOR ADDRESS OR FIRST REG. INDEX

```

.BYTE 0      ;(11) TRACK ADDRESS OR
              ;LAST REG. INDEX
.WORD 0      ;(12) CYLINDER ADDRESS
.WORD 0      ;(14) ERROR TABLE POINTER
              ;POINTS TO THE FIRST OF TWENTY
              ;LOCATIONS OF WHERE THE DRIVER
              ;IS TO STORE THE RHD/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
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$: JSR    RD,RM03      ;CALL THE DRIVER
    WRTDPB ;DPB ADDRESS
BR   1$                ;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

SEQ 0028

10.5 RH70/RM03 REGISTERS

<u>MNEMONIC</u>	<u>INDEX</u>
RMCS1	0
RMWC	1
RMBA	2
RMDA	3
RMCS2	4
RMDS	5
RMER1	6
RMA	7
RMLA	8
RMDB	9
RMMR1	10
RMDT	11
RMSN	12
RMOF	13
RMDC	14
RMHP	15
RMMR2	16
RMER2	17
RMEC1	18
RMEC2	19

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

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.

BIT NO. -----	MEANING IF ON A "1" -----
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= RMCSI'S ADDRESS
2	UNEXPECTED ATTENTION OCCURRED	R1= DRIVE NUMBER R3= ATA BIT *R4= RMCSI'S ADDRESS R5= (RMAS) RMERRS =RMDS RMERRS+2=RMER1 RMERRS+4=RMER2 RMERRS+6=RMMR2
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= RMCSI'S ADDRESS
R5= (RMAS)
RMERRS =RMDS
RMERRS+2=RMER1
RMERRS+4=RMER2
RMERRS+6=RMMR2

SEQ 0031

* THIS IS THE ACTUAL UNIBUS ADDRESS (176700)

11. PROGRAM LISTING

11-11-77 10:00 AM

.TITLE MD-11-DZRMF-B RM03/RM02 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.
:*


```

68 ;HOOKS REQUIRED BY ACT11
69 $SVPC=. ;SAVE PC
70 . =46
71 000046 021300 $SENDAD ;;1)SET LOC.46 TO ADDRESS OF $SENDAD IN .SEOP
72 000052 000052 . =52
73 000052 000000 .WORD 0 ;;2)SET LOC.52 TO ZERO
74 000200 000200 .=$SVPC ;; PESTORE PC
75
76 .SBTTL STARTING ADDRESSES
77 . =200
78 ;*200 = NORMAL START
79 000200 000137 004740 JMP @#START1
80 ;*204 = SELECT OPERATING PARAMETERS
81 000204 000137 004762 JMP @#START2
82 ;*210 = SELECT RH70/RM03 ADDRESSES
83 000210 000137 004730 JMP @#START3
84 ;*214 = COMBINATION OF 204 AND 210
85 000214 000137 004752 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 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 R0= %0 ;;GENERAL REGISTER
108 000001 R1= %1 ;;GENERAL REGISTER
109 000002 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 PRO= 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
125 000300
126 000340

PR5= 240
PR6= 300
PR7= 340

:::PRIORITY LEVEL 5
:::PRIORITY LEVEL 6
:::PRIORITY LEVEL 7

127 100000
128 040000
129 020000
130 010000
131 004000
132 002000
133 001000
134 000400
135 000200
136 000100
137 000040
138 000020
139 000010
140 000004
141 000002
142 000001

:"SWITCH REGISTER" SWITCH DEFINITIONS

SW15= 100000
SW14= 40000
SW13= 20000
SW12= 10000
SW11= 4000
SW10= 2000
SW09= 1000
SW08= 400
SW07= 200
SW06= 100
SW05= 40
SW04= 20
SW03= 10
SW02= 4
SW01= 2
SW00= 1
.EQUIV SW09,SW9
.EQUIV SW08,SW8
.EQUIV SW07,SW7
.EQUIV SW06,SW6
.EQUIV SW05,SW5
.EQUIV SW04,SW4
.EQUIV SW03,SW3
.EQUIV SW02,SW2
.EQUIV SW01,SW1
.EQUIV SW00,SW0

143 100000
144 040000
145 020000
146 010000
147 004000
148 002000
149 001000
150 000400
151 000200
152 000100
153 000040
154 000020
155 000010
156 000004
157 000002
158 000001

:"DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15= 100000
BIT14= 40000
BIT13= 20000
BIT12= 10000
BIT11= 4000
BIT10= 2000
BIT09= 1000
BIT08= 400
BIT07= 200
BIT06= 100
BIT05= 40
BIT04= 20
BIT03= 10
BIT02= 4
BIT01= 2
BIT00= 1
.EQUIV BIT09,BIT9
.EQUIV BIT08,BIT8
.EQUIV BIT07,BIT7
.EQUIV BIT06,BIT6
.EQUIV BIT05,BIT5
.EQUIV BIT04,BIT4
.EQUIV BIT03,BIT3

179


```

180 .EQUIV BIT02,BIT2
181 .EQUIV BIT01,BIT1
182 .EQUIV BIT00,BIT0
183
184
185 ;*BASIC "CPU" TRAP VECTOR ADDRESSES
186 ERRVEC= 4 ;: TIME OUT AND OTHER ERRORS
187 RESVEC= 10 ;: RESERVED AND ILLEGAL INSTRUCTIONS
188 TBITVEC=14 ;: "T" BIT
189 TRTVEC= 14 ;: TRACE TRAP
190 BPTVEC= 14 ;: BREAKPOINT TRAP (BPT)
191 IOTVEC= 20 ;: INPUT/OUTPUT TRAP (IOT) **SCOPE**
192 PWRVEC= 24 ;: POWER FAIL
193 EMTVEC= 30 ;: EMULATOR TRAP (EMT) **ERROR**
194 TRAPVEC=34 ;: "TRAP" TRAP
195 TKVEC= 60 ;: TTY KEYBOARD VECTOR
196 TPVEC= 64 ;: TTY PRINTER VECTOR
197 PIRQVEC=240 ;: PROGRAM INTERRUPT REQUEST VECTOR
198 ;:*****
199
200 .SBTTL RH70 REGISTERS
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 US4= 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
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291

040000
100000

000001
000002
000004
000010
000020
000040
004000

000001
000100
000200
000400
001000
002000
004000
010000
020000
040000
100000

000001
000002
000004
000010
000020
000040
000100
000200
000400
001000
002000
004000
010000
020000
040000
100000

WCE= 40000 ;WRITE CHECK ERROR (BIT #14)
DLT= 100000 ;DATA LATE (BIT #15)

;DATA BUFFER REGISTER (RMD8)
;(EACH BIT IS CALLED BY BIT NUMBER)

.SBTTL RM03 REGISTERS

;CONTROL AND STATUS 1 REGISTER. (#00)

GO= 1 ;GO BIT (BIT #0)
F1= 2 ;FUNCTION CODE BIT #1
F2= 4 ;FUNCTION CODE BIT #2
F3= 10 ;FUNCTION CODE BIT #3
F4= 20 ;FUNCTION CODE BIT #4
F5= 40 ;FUNCTION CODE BIT #5
DVA= 4000 ;DEVICE AVAILABLE (BIT #11)

;DRIVE STATUS REGISTER (RMD5) (#01)

OM= 1 ;
VV= 100 ;VOLUME VALID (BIT #6)
DRY= 200 ;DRIVE READY (BIT #7)
DPR= 400 ;DRIVE PRESENT (BIT #8)
PGM= 1000 ;PROGRAMABLE (BIT #9)
LST= 2000 ;LAST SECTOR TRANSFERRED (BIT #10)
WRL= 4000 ;WRITE LOCK (BIT #11)
MOL= 10000 ;MEDIUM ON-LINE (BIT #12)
PIP= 20000 ;POSITIONING OPERATION IN PROGRESS (BIT #13)
ERR= 40000 ;COMPOSITE ERROR (BIT #14)
ATA= 100000 ;ATTENTION ACTIVE (BIT #15)

;ERROR REGISTER #01 (RMER1) (#02)

ILF= 1 ;ILLEGAL FUNCTION (BIT #0)
ILR= 2 ;ILLEGAL REGISTER (BIT #1)
RMR= 4 ;REGISTER MODIFICATION REFUSED (BIT #2)
PAR= 10 ;PARITY ERROR (BIT #3)
FER= 20 ;FORMAT ERROR (BIT #4)
WCF= 40 ;WRITE CLOCK FAIL (BIT #5)
ECH= 100 ;ECC HARD ERROR (BIT #6)
HCE= 200 ;HEADER COMPARE ERROR (BIT #7)
HCRC= 400 ;HEADER CRC ERROR (BIT #8)
AOE= 1000 ;ADDRESS OVERFLOW ERROR (BIT #9)
IAE= 2000 ;INVALID ADDRESS ERROR (BIT #10)
WLE= 4000 ;WRITE LOCK ERROR (BIT #11)
DTE= 10000 ;DRIVE TIMING ERROR (BIT #12)
OPI= 20000 ;OPERATION INCOMPLETE (BIT #13)
UNS= 40000 ;DRIVE UNSAFE (BIT #14)
DCK= 100000 ;DATA CHECK ERROR (BIT 15)

292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400

```

000001      DMD= 1
000002      LSIT= 2
              ;LS=4
000010      WD= 10
000020      EECC= 20
000040      WC= 40
000100      CONT= 100
000200      PHA= 200
000400      PDA= 400
001000      ECRC= 1000
002000      PLFS= 2000
004000      ESRC= 4000
010000      REX= 10000
020000      EBL= 20000
              ;R/G= 40000
100000      OCC= 100000

```

;MAINTAINABILITY REGISTER #01 (RMMR1)(#03) - READ ONLY BITS
;DIAGNOSTIC MODE

```

000001      DMD= 1
000002      MSC= 2
000004      MI= 4
000010      MWP= 10
000020      DTG= 20
000040      MS= 40
000100      MDF= 100
000200      MSER= 200
000400      MOC= 400
001000      MUR= 1000
002000      MRD= 2000
004000      MCLK= 4000
010000      MSEN= 10000
020000      DTG= 20000
040000      OBEN= 40000
100000      OBCK= 100000

```

;MAINTAINABILITY REGISTER #01 (RMMR1) (#03) - WRITE ONLY BITS
;DIAGNOSTIC MODE BIT

```

000001      AT0= 1
000002      AT1= 2
000004      AT2= 4
000010      AT3= 10
000020      AT4= 20
000040      AT5= 40
000100      AT6= 100
000200      AT7= 200

```

;ATTENTION SUMMARY PSEUDO-REGISTER (RMAS) (#04)
;DEVICE 0 (BIT #0)
;DEVICE 1 (BIT #1)
;DEVICE 2 (BIT #2)
;DEVICE 3 (BIT #3)
;DEVICE 4 (BIT #4)
;DEVICE 5 (BIT #5)
;DEVICE 6 (BIT #6)
;DEVICE 7 (BIT #7)

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

```

000001      DT00= 1
000002      DT01= 2

```

;DRIVE TYPE REGISTER (RMDT) (#06)
;DRIVE TYPE NUMBER BIT 1
;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					
361					
362					
363					
364	000100	SC0=	100	;	SECTOR COUNT FIELD 0 (BIT #6)
365	000200	SC1=	200	;	SECTOR COUNT FIELD 1 (BIT #7)
366	000400	SC2=	400	;	SECTOR COUNT FIELD 2 (BIT #8)
367		;SC3=	1000	;	SECTOR COUNT FIELD 3 (BIT #9)
368		;SC4=	2000	;	SECTOR COUNT FIELD 4 (BIT #10)
369					
370					
371					
372					
373					
374					
375					
376					
377					
378					
379					
380					
381					
382					
383					
384					
385					
386					
387					
388					
389					
390					
391					
392					
393					
394					
395					
396	000001	OFFDIR=	1	;	RM03 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					
402					
403					

;LOOK-AHEAD REGISTER (RMLA) (#07)

;RM03 MAINTAINABILITY REGISTER #2 (RMMR2) (#10)

;RM03 ERROR REGISTER #02 (RMMR2) (#10)

;OFFSET REGISTER (RMOF) (#11)

;DESIRED CYLINDER ADDRESS (RMDC) (#12)
 ;(EACH BIT IS CALLED BY BIT NUMBER)

104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200

020000
040000

000101
000103
000105
000107
000111
000113
000115
000117
000121
000123
000131
000151
000153
000161
000163
000171
000173
000141
000143
000145

177400
010000

;CURRENT CYLINDER ADDRESS (RMHR) (#13)
;(EACH BIT IS CALLED BY BIT NUMBER)

;SERIAL NUMBER REGISTER (RMSN) (#14)
;(EACH IS CALLED BY BIT NUMBER)

;RMO3 ERROR REGISTER #02 (RMER2) (#15)

OPE= 20000 ;OPERATOR PLUG ERROR (BIT #13)
SKI= 40000 ;SEEK INCOMPLETE (BIT #15)

;ECC POSITION REGISTER (RMEC1) (#16)
;(EACH BIT IS CALLED BY BIT NUMBER)

;ECC PATTERN REGISTER (RMEC2) (#17)
;(EACH BIT IS CALLED BY BIT NUMBER)

;*****

;OP CODE DEFINITIONS
NOOP=101
UNLOAD=103
SEEK=105
RECAL=107
DRVCLR=111
RELEASE=113
OFFSET=115
RTC=117
READIN=121
PACK=123
SEARCH=131
WRCKD=151
WRCKHD=153
WRITE=161
WRTHD=163
READ=171
READHD=173
GETREG=141
SETFORM=143
SELDRV=145

;OTHER EQUATES
SCTRWC = -256. ;WORD COUNT FOR SECTOR
FMT22=10000 ;FORMAT 22 BIT

453
454
455
456
457
458
459
460 000024
461
462 000044
463
464
465
466
467
468 000220
469 000220 000000
470 000222 001220
471 000224 001604
472 000226 001604
473 000230 001604
474 000232 000030

.SBTTL APT PARAMETER BLOCK
 ;*****
 ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
 ;*****
 .SX= . ;SAVE CURRENT LOCATION
 =24 ;SET POWER FAIL TO POINT TO START OF PROGRAM
 200 ;FOR APT START UP
 =44 ;POINT TO APT INDIRECT ADDRESS PNTR.
 \$APTHDR ;POINT TO APT HEADER BLOCK
 =.SX ;RESET LOCATION COUNTER
 ;*****
 ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
 ;INTERFACE SPEC.

\$APTHD: ;
 \$HI8TS: .WORD 0 ;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
 \$MBADR: .WORD \$MAIL ;ADDRESS OF APT MAILBOX (BITS 0-15)
 \$TSTM: .WORD 900. ;RUN TIM OF LONGEST TEST
 \$PASTM: .WORD 900. ;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
 \$UNITM: .WORD 900. ;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
 ;SETEND-\$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

475
476
477
478
479
480
481
482 001100 001100
483 001100 000000
484 001102 000
485 001103 000
486 001104 000000
487 001106 000000
488 001110 000000
489 001112 000000
490 001114 000
491 001115 001
492 001116 000000
493 001120 000000
494 001122 000000
495 001124 000000
496 001126 000000
497 001130 000000
498 001132 000000
499 001134 000
500 001135 000
501 001136 000000
502 001140 177570
503 001142 177570
504 001144 177560
505 001146 177562
506 001150 177564
507 001152 177566
508 001154 000
509 001155 002
510 001156 012
511 001157 000
512 001160 000000
513
514 001162 000000
515 001164 000000
516 001166 000000
517 001170 000000
518 001172 000000
519 001174 000000
520 001176 000000
521 001200 000000
522 001202 000000
523 001204 000000
524 001206 000000
525 001210 177607 000377
526 001214 077
527 001215 015
528 001216 000012
529
530

.SBTTL COMMON TAGS

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

SCMTAG: =1100

STSTNM: .WORD 0
SERFLG: .BYTE 00
SICNT: .WORD 000
SLPADR: .WORD 0000
SLPERR: .WORD 0000
SERTTL: .WORD 0000
SITEMB: .BYTE 00
SERMAX: .BYTE 1
SERRPC: .WORD 000
SGDADR: .WORD 000
SBDADR: .WORD 000
SGDAT: .WORD 0000
SBDDAT: .WORD 0000
SAUTOB: .BYTE 00
SINTAG: .BYTE 00
SWR: .WORD DSWR
DISPLAY: .WORD DDISP
STKS: 177560
STKB: 177562
STPS: 177564
STPB: 177566
SNUL: .BYTE 0
SFILLS: .BYTE 2
SFILLC: .BYTE 12
STPFLG: .BYTE 0
SREGAD: .WORD 0
SREG0: .WORD 0
SREG1: .WORD 00
SREG2: .WORD 00
SREG3: .WORD 00
SREG4: .WORD 00
SREG5: .WORD 00
STMP0: .WORD 00
STMP1: .WORD 0
STMP2: .WORD 0
STIMES: 0
SESCAPE: 0
SBELL: .ASCIZ <207><377><377>
SQUES: .ASCII /?/
SCRLF: .ASCII <15>
SLF: .ASCIZ <12>

;; START OF COMMON TAGS

CONTAINS THE TEST NUMBER
CONTAINS ERROR FLAG
CONTAINS SUBTEST ITERATION COUNT
CONTAINS SCOPE LOOP ADDRESS
CONTAINS SCOPE RETURN FOR ERRORS
CONTAINS TOTAL ERRORS DETECTED
CONTAINS ITEM CONTROL BYTE
CONTAINS MAX. ERRORS PER TEST
CONTAINS PC OF LAST ERROR INSTRUCTION
CONTAINS ADDRESS OF 'GOOD' DATA
CONTAINS ADDRESS OF 'BAD' DATA
CONTAINS 'GOOD' DATA
CONTAINS 'BAD' DATA
RESERVED--NOT TO BE USED
AUTOMATIC MODE INDICATOR
INTERRUPT MODE INDICATOR
ADDRESS OF SWITCH REGISTER
ADDRESS OF DISPLAY REGISTER
TTY KBD STATUS
TTY KBD BUFFER
TTY PRINTER STATUS REG. ADDRESS
TTY PRINTER BUFFER REG. ADDRESS
CONTAINS NULL CHARACTER FOR FILLS
CONTAINS # OF FILLER CHARACTERS REQUIRED
INSERT FILL CHARS. AFTER A "LINE FEED"
"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
CONTAINS THE ADDRESS FROM WHICH (SREG0) WAS OBTAINED
CONTAINS ((SREGAD)+0)
CONTAINS ((SREGAD)+2)
CONTAINS ((SREGAD)+4)
CONTAINS ((SREGAD)+6)
CONTAINS ((SREGAD)+10)
CONTAINS ((SREGAD)+12)
USER DEFINED
USER DEFINED
USER DEFINED
MAX NUMBER OF ITERATIONS
ESCAPE ON ERROR ADDRESS
CODE FOR BELL
QUESTION MARK
CARRIAGE RETURN
LINE FEED

.SBTTL APT MAILBOX-ETABLE

```

531
532
533
534 001220
535 001220 000000
536 001222 000000
537 001224 000000
538 001226 000000
539 001230 000000
540 001232 000000
541 001234 000000
542 001236 000000
543 001240
544 001240 000
545 001241 000
546 001242 000000
547 001244 000000
548 001246 000000
549
550
551
552
553
554
555 001250 000
556 001251 000
557
558
559
560
561 001252 000000
562
563 001254 000
564 001255 000
565 001256 000000
566 001260 000
567 001261 000
568 001262 000000
569 001264 000
570 001265 000
571 001266 000000
572 001270 000000
573 001272 000000
574 001274 000000
575 001276 000000
576 001300
577
578 000015
579 000012
580 001300 000000
581 001302 000000
582 001304 000000
583 001306 000000
584 001310 000000
585 001312 000000
586 001314 176777 000003

```

```

*****
.EVEN
$MAIL:
$MSGTY: .WORD   AMSGTY
$FATAL: .WORD   AFATAL
$TESTN: .WORD   ATESTN
$PASS:  .WORD   APASS
$DEVCT: .WORD   ADEVCT
$UNIT:  .WORD   AUNIT
$MSGAD: .WORD   AMSGAD
$MSGLG: .WORD   AMSGLG
$ETABLE:
$ENV:   .BYTE   AENV
$ENVM:  .BYTE   AENVM
$SWREG: .WORD   ASWREG
$USWR:  .WORD   AUSWR
$CPUOP: .WORD   ACPUOP
*****
:: APT MAILBOX
:: MESSAGE TYPE CODE
:: FATAL ERROR NUMBER
:: TEST NUMBER
:: PASS COUNT
:: DEVICE COUNT
:: I/O UNIT NUMBER
:: MESSAGE ADDRESS
:: MESSAGE LENGTH
:: APT ENVIRONMENT TABLE
:: ENVIRONMENT BYTE
:: ENVIRONMENT MODE BITS
:: APT SWITCH REGISTER
:: USER SWITCHES
:: CPU TYPE, OPTIONS
BITS 15-11=CPU TYPE
      11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
      11/70=06,PDQ=07,Q=10
BIT 10=REAL TIME CLOCK
BIT 9=FLOATING POINT PROCESSOR
BIT 8=MEMORY MANAGEMENT
:: HIGH ADDRESS, M.S. BYTE
:: MEM. TYPE, BLK#1
MEM. TYPE BYTE -- (HIGH BYTE)
      900 NSEC CORE=001
      300 NSEC BIPOLAR=002
      500 NSEC MOS=003
:: HIGH ADDRESS, BLK#1
MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
:: HIGH ADDRESS, M.S. BYTE
:: MEM. TYPE, BLK#2
MEM. LAST ADDRESS, BLK#2
:: HIGH ADDRESS, M.S. BYTE
MEM. TYPE, BLK#3
MEM. LAST ADDRESS, BLK#3
:: HIGH ADDRESS, M.S. BYTE
MEM. TYPE, BLK#4
MEM. LAST ADDRESS, BLK#4
:: INTERRUPT VECTOR#1, BUS PRIORITY#1
:: INTERRUPT VECTOR#2, BUS PRIORITY#2
:: BASE ADDRESS OF EQUIPMENT UNDER TEST
:: DEVICE MAP
CR = 15
LF = 12
C.SWR: .WORD 0 ; CONTROL SWITCHES
SAVCSW: .WORD 0 ; PREVIOUS CONTENTS OF 'C.SWR'
CNTRLC: .WORD 0 ; CONTROL "C" FLAG
BUSADR: .WORD 0 ; GET ADDRESSES FROM THE TTY FLAG (0=NO, -1=YES)
LPTAVL: .WORD 0 ; LPT AVAILABLE STATUS (0=NO, 1=YES)
DRVSEL: .WORD 0 ; DRIVES SELECTED FOR TESTING
TSTNMS: .WORD 176777,3 ; DEFAULT IS RUN TESTS 0-10 & 12-21

```


587	001320	000000	000000	OPNFLG: .WORD	0,0	:MODIFY TEST PARAMETER FLAGS
588	001324	000000		CLKSTA: .WORD	0	:CLOCK STATUS (0=NO CLOCK,+1=KW11-P, AND -1=KW11-L)
589						:16 MILLISECONDS PER CLOCK TICK
590	001326	000020		TICKMS: .WORD	16.	:16666 MIRCSECONDS PER CLOCK TICK
591	001330	040432		TICKUS: .WORD	16666.	
592	001332	000000		BYPASS: .WORD	0	
593	001334	000000		CHKDRV: .WORD	0	:DRIVE UNDER TEST
594	001336	000000		DRVMSK: .WORD	0	:DRIVE MASK BIT
595	001340	000000		SVSTAT: .WORD	0	:STATUS/ERROR INDICATOR IS SAVED HERE ON AN ERROR
596						:CYLINDER READ
597	001342	000000		CYL.RD: .WORD	0	
598	001344	000000		TRK.RD: .WORD	0	:TRACK READ
599	001346	000000		SEC.RD: .WORD	0	:SECTOR READ
600	001350	000000		CYL.DS: .WORD	0	:CYLINDER DESIRED
601	001352	000000		SEC.DS: .WORD	0	:SECTOR DESIRED
602	001354	000000		TRK.DS: .WORD	0	:TRACK DESIRED
603	001356	000000		TIM.UP: .WORD	0	:MINIMUM TIME
604	001360	000000		.WORD	0	:NUMBER OF COUNTS BELOW MIN. LIMIT
605	001362	000000		.WORD	0	:MAXIMUM TIME
606	001364	000000		.WORD	0	:NUMBER OF COUNTS ABOVE MAX. LIMIT
607	001366	000000	000000	.WORD	0,0	:TOTAL TIME OF ALL SEEKS
608	001372	000000		.WORD	0	:NUMBER OF SEEKS PERFORMED
609	001374	000000		TIM.DN: .WORD	0	:MINIMUM TIME
610	001376	000000		.WORD	0	:NUMBER OF COUNTS BELOW MIN. LIMIT
611	001400	000000		.WORD	0	:MAXIMUM TIME
612	001402	000000		.WORD	0	:NUMBER OF COUNTS ABOVE MAX. LIMIT
613	001404	000000	000000	.WORD	0,0	:TOTAL TIME OF ALL SEEKS
614	001410	000000		.WORD	0	:NUMBER OF SEEKS PERFORMED
615	001412	000000		TIM.PT: .WORD	0	:POINTS TO TABLE OF TIMES
616	001414	000000		WCEFLG: .WORD	0	:FATAL WRITE CHECK ERROR FLAG (TEST 20)
617	001416	000000		STALLO: .WORD	0	:VARIABLE STALL (TEST 4)
618	001420	000000	000000	SVADR: .WORD	0,0	:SAVE DISK ADDRESS (TEST 22)
619	001424	000000		SEKTRM: .WORD	0	:SEEK TIMER (TEST 10)
620	001426	000000		SEKCNT: .WORD	0	:SEEK COUNTER
621	001430	000000		DELTA: .WORD	0	:TESTING RANGE FOR SERVO SETTLE DOWN TEST
622	001432	160000		TRCKWC: .WORD	-(256.*32.)	:WORD COUNT FOR A FULL TRACK IN 16 BIT MODE
623	001434	000012		STALL1: .WORD	10.	:10 MILLISECONDS STALL (TEST 0-11)
624	001436	000012		STALL2: .WORD	10.	:10 MILLISECONDS STALL (TEST 16-21)
625	001440	011610		STALL3: .WORD	5000.	:5 SEC STALL (TEST 22)
626	001442	000031		MXSTAL: .WORD	25.	:MAX. INCREMENTING STALL ALLOWED IN TEST 4
627	001444	144		ERR.CT: .BYTE	100.	:NUMBER OF ERRORS ALLOWED IN TESTS 16 - 21 BEFORE GOING TO THE NEXT TEST
628						:RESERVED
629	001445	000		BASFLG: .BYTE	0	:FLAG FOR DETECTING BAD SECTOR
630	001446	000000		.WORD	0	
631						
632				:ADDRESSES AND VECTORS		
633	001450	176700		RH.ADR: .WORD	176700	:RH70 UNIBUS ADDRESS
634	001452	000254	000240	RHVEC: .WORD	254,5*32.	:RH70 VECTOR ADDRESS AND PRIORITY
635	001456	000104	000106	PKV: .WORD	104,106	:KW11-P VECTOR ADDRESS
636	001462	172540		PKCS: .WORD	172540	:KW11-P CONTROL AND STATUS REG.
637	001464	172542		PKB: .WORD	172542	:KW11-P COUNT SET BUFFER
638	001466	172544		PKC: .WORD	172544	:KW11-P COUNTER
639	001470	000100	000102	LKV: .WORD	100,102	:KW11-L VECTOR ADDRESS
640	001474	177546		LKS: .WORD	177546	:KW11-L STATUS REGISTER
641	001476	177564		TPS: .WORD	177564	:TTY PRINTER STATUS
642	001500	177566		TPB: .WORD	177566	:TTY PRINTER BUFFER

643 001502 177514
 644 001504 177516
 645
 646
 647 001506 000001
 648 001510 000002
 649 001512 000004
 650 001514 000010
 651 001516 000020
 652 001520 000040
 653 001522 000100
 654 001524 000200
 655 001526 000400
 656 001530 001000
 657 001532 002000
 658 001534 004000
 659 001536 010000
 660 001540 020000
 661 001542 040000
 662 001544 100000
 663 001546 000001
 664 001550 000002
 665 001552 000004
 666 001554 000010
 667 001556 000020
 668 001560 000040
 669 001562 000100
 670 001564 000200

LPS: .WORD 177514 ;LINE PRINTER STATUS
 LPB: .WORD 177516 ;LINE PRINTER BUFFER

: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

671
 672
 673
 674 001566 000000
 675
 676 001570 000000
 677 001572 000000
 678 001574 000000
 679 001576 000000
 680 001600 000000
 681 001602 000000
 682 001604 000000
 683 001606 000000
 684 001610 000000
 685 001612 000000
 686 001614 000000
 687 001616 000000
 688
 689

: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

690 001620 002412
 691 001622 002426
 692 001624 002454
 693 001626 002476
 694 001630 002520
 695 001632 002542
 696 001634 002564
 697 001636 002606
 698 001640 002626

:TABLE OF PARAMETER POINTERS
 PRMPT: .WORD PRM0
 .WORD PRM1
 .WORD PRM2
 .WORD PRM3
 .WORD PRM4
 .WORD PRM5
 .WORD PRM6
 .WORD PRM7
 .WORD PRM10

699	001642	002646	.WORD	PRM11	
700	001644	002670	.WORD	PRM12	
701	001646	002704	.WORD	PRM13	
702	001650	002720	.WORD	PRM14	
703	001652	002734	.WORD	PRM15	
704	001654	002750	.WORD	PRM16	
705	001656	002762	.WORD	PRM17	
706	001660	002774	.WORD	PRM20	
707	001662	003026	.WORD	PRM21	
708	001664	003042	.WORD	PRM22	
709	001666	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
PRMSG:  .WORD MSG.R
         .WORD MSG.FC
         .WORD MSG.LC
         .WORD MSGFCP
         .WORD MSGLCP
         .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

```

743	001746	001125	000310	000632	DFLT: .WORD 1125,200.,410.,822.,0,0	;RECAL/SEEK (T0)
744	001754	001466	000000	000000		
745	001762	003377	000144	000000	.WORD 3377,100.,0,128.,0,256.,0,0,0,0,0	;SEEK/SEEK (T1)
746	001770	000200	000000	000400		
747	001776	000000	000000	000000		
748	002004	000000	000000			
749	002010	001177	000001	000000	.WORD 1177,1,0,410.,0,822.,1,0,0	;INCREMENT SEEK (T2)
750	002016	000632	000000	001466		
751	002024	000001	000000	000000		
752	002032	001177	000010	000000	.WORD 1177,10,0,256.,0,512.,1,0,0	;STEPPING SEEK (T3)
753	002040	000400	000000	001000		
754	002046	000001	000000	000000		

755	002054	001177	000001	000000	.WORD	1177,1,0,410.,0,822.,1,0,0	;OSCILLATING SEEK (T4)
756	002062	000632	000000	001466			
757	002070	000001	000000	000000			
758	002076	001177	000001	000000	.WORD	1177,1,0,410.,0,822.,1,0,0	;CONVERGING/DIVERGING SEEK (T5)
759	002104	000632	000000	001466			
760	002112	000001	000000	000000			
761	002120	001177	000001	000000	.WORD	1177,1,0,410.,0,822.,1,0,0	;SERVO ADDRESSING LOGIC NOISE (T6)
762	002126	000632	000000	001466			
763	002134	000001	000000	000000			
764	002142	000337	011610	000000	.WORD	337,5000.,0,410.,0,822.,0,4	;RANDOM SEEK TEST (T7)
765	002150	000632	000000	001466			
766	002156	000000	000004				
767	002162	000177	000001	000000	.WORD	177,1,0,410.,0,822.,100.,0	;SERVO SETTLE DOWN TEST (T10)
768	002170	000632	000000	001466			
769	002176	000144	000000				
770	002202	001177	000001	000000	.WORD	1177,1,0,410.,0,822.,1,0,0	;ALL SEEKS TEST (T11)
771	002210	000632	000000	001466			
772	002216	000001	000000	000000			
773	002224	001113	000001	000000	.WORD	1113,1,0,0,0,0	;ROTATIONAL SPEED TIMING TEST (T12)
774	002232	000000	000000	000000			
775	002240	000037	000001	000000	.WORD	37,1,0,410.,0,822.	;ONE CYLINDER SEEK TIMING TEST (T13)
776	002246	000632	000000	001466			
777	002254	000037	000001	000000	.WORD	37,1,0,136.,0,255.	;ACCESS TIME MEASUREMENT TEST (T14)
778	002262	000210	000000	000377			
779	002270	000037	000001	000000	.WORD	37,1,0,410.,0,822.	;MAXIMUM SEEK TIMING TEST (T15)
780	002276	000632	000000	001466			
781	002304	000113	000001	000000	.WORD	113,1,0,0,0	;SECTOR ADDRESSING TEST (T16)
782	002312	000000	000000				
783	002316	001013	000001	000000	.WORD	1013,1,0,0,0	;TRACK ADDRESSING TEST (T17)
784	002324	000000	000000				
785	002330	007777	000001	000000	.WORD	7777,1,0,410.,0,821.,64.,0,4,1,1,0,177777	;DATA TEST (T20)
786	002336	000632	000000	001465			
787	002344	000100	000000	000004			
788	002352	000001	000001	000000			
789	002360	177777					
790	002362	000037	001750	000000	.WORD	37,1000.,0,410.,0,821.	;EXERCISER (T21)
791	002370	000632	000000	001465			
792	002376	000037	011610	000000	.WORD	37,5000.,0,136.,0,255.	;RMO3 ACCESS TIME ADJUSTMENT TEST (T22)
793	002404	000210	000000	000377			
794							
795							
796							
797							
798							
799							

;PARAMETER TABLES

;RECAL/SEEK (T0)

PRMO:	.WORD	1125
	.WORD	200.
	.WORD	410.
	.WORD	822.
	.WORD	0
	.WORD	0

;SEEK/SEEK (T1)

PRM1:	.WORD	3377
	.WORD	100.
	.WORD	0

800	002412	001125		
801	002414	000310		
802	002416	000632		
803	002420	001466		
804	002422	000000		
805	002424	000000		
806				
807				
808	002426	003377		
809	002430	000144		
810	002432	000000		

811	002434	000200	.WORD	128.
812	002436	000000	.WORD	0
813	002440	000400	.WORD	256.
814	002442	000000	.WORD	0
815	002444	000000	.WORD	0
816	002446	000000	.WORD	0
817	002450	000000	.WORD	0
818	002452	000000	.WORD	0

: INCREMENT SEEK (T2)

820			PRM2:	.WORD	1177
821	002454	001177		.WORD	1
822	002456	000001		.WORD	0
823	002460	000000		.WORD	410.
824	002462	000632		.WORD	0
825	002464	000000		.WORD	822.
826	002466	001466		.WORD	1
827	002470	000001		.WORD	0
828	002472	000000		.WORD	0
829	002474	000000		.WORD	0

: STEPPING SEEK (T3)

830			PRM3:	.WORD	1177
831				.WORD	1
832	002476	001177		.WORD	0
833	002500	000001		.WORD	256.
834	002502	000000		.WORD	0
835	002504	000400		.WORD	512.
836	002506	000000		.WORD	1
837	002510	001000		.WORD	0
838	002512	000001		.WORD	0
839	002514	000000		.WORD	0
840	002516	000000		.WORD	0

: OSCILLATING SEEK (T4)

841			PRM4:	.WORD	1177
842				.WORD	1
843	002520	001177		.WORD	0
844	002522	000001		.WORD	410.
845	002524	000000		.WORD	0
846	002526	000632		.WORD	822.
847	002530	000000		.WORD	1
848	002532	001466		.WORD	0
849	002534	000001		.WORD	0
850	002536	000000		.WORD	0
851	002540	000000		.WORD	0

: CONVERGING/DIVERGING SEEK (T5)

852			PRM5:	.WORD	1177
853				.WORD	1
854	002542	001177		.WORD	0
855	002544	000001		.WORD	410.
856	002546	000000		.WORD	0
857	002550	000632		.WORD	822.
858	002552	000000		.WORD	1
859	002554	001466		.WORD	0
860	002556	000001		.WORD	0
861	002560	000000		.WORD	0
862	002562	000000		.WORD	0

: SERVO ADDRESSING LOGIC NOISE GENERATOR (T6)

863			PRM6:	.WORD	1177
864				.WORD	1
865	002564	001177			
866	002566	000001			

867 002570 000000
868 002572 000632
869 002574 000000
870 002576 001466
871 002600 000001
872 002602 000000
873 002604 000000
874
875
876 002606 000337
877 002610 011610
878 002612 000000
879 002614 000632
880 002616 000000
881 002620 001466
882 002622 000000
883 002624 000004
884
885
886 002626 000177
887 002630 000001
888 002632 000000
889 002634 000632
890 002636 000000
891 002640 001466
892 002642 000144
893 002644 000000
894
895
896 002646 001177
897 002650 000001
898 002652 000000
899 002654 000632
900 002656 000000
901 002660 001466
902 002662 000001
903 002664 000000
904 002666 000000
905
906
907 002670 001113
908 002672 000001
909 002674 000000
910 002676 000000
911 002700 000000
912 002702 000000
913
914
915 002704 000037
916 002706 000001
917 002710 000000
918 002712 000632
919 002714 000000
920 002716 001466
921
922

.WORD 0
.WORD 410.
.WORD 0
.WORD 822.
.WORD 1
.WORD 0
.WORD 0
:RANDOM SEEK TEST (T7)
PRM7: .WORD 337
.WORD 5000.
.WORD 0
.WORD 410.
.WORD 0
.WORD 822.
.WORD 0
.WORD 4
:SERVO SETTLE DOWN TEST (T10)
PRM10: .WORD 177
.WORD 1
.WORD 0
.WORD 410.
.WORD 0
.WORD 822.
.WORD 100.
.WORD 0
:ALL SEEKS TEST (T11)
PRM11: .WORD 1177
.WORD 1
.WORD 0
.WORD 410.
.WORD 0
.WORD 822.
.WORD 1
.WORD 0
.WORD 0
:ROTATIONAL SPEED TIMING TEST (T12)
PRM12: .WORD 1113
.WORD 1
.WORD 0
.WORD 0
.WORD 0
.WORD 0
:ONE CYLINDER SEEK TIMING TEST (T13)
PRM13: .WORD 37
.WORD 1
.WORD 0
.WORD 410.
.WORD 0
.WORD 822.
:ACCESS TIME MEASUREMENT TEST (T14)

923 002720 000037
 924 002722 000001
 925 002724 000000
 926 002726 000210
 927 002730 000000
 928 002732 000377
 929
 930
 931 002734 000037
 932 002736 000001
 933 002740 000000
 934 002742 000632
 935 002744 000000
 936 002746 001466
 937
 938
 939 002750 000113
 940 002752 000001
 941 002754 000000
 942 002756 000000
 943 002760 000000
 944
 945
 946 002762 001013
 947 002764 000001
 948 002766 000000
 949 002770 000000
 950 002772 000000
 951
 952
 953 002774 007777
 954 002776 000001
 955 003000 000000
 956 003002 000632
 957 003004 000000
 958 003006 001465
 959 003010 000100
 960 003012 000000
 961 003014 000004
 962 003016 000001
 963 003020 000001
 964 003022 000000
 965 003024 177777
 966
 967
 968 003026 000037
 969 003030 001750
 970 003032 000000
 971 003034 000632
 972 003036 000000
 973 003040 001465
 974
 975
 976 003042 000037
 977 003044 011610
 978 003046 000000

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

: MAXIMUM SEEK TIMING TEST (T15)
 PRM15: .WORD 37
 .WORD 1
 .WORD 0
 .WORD 410.
 .WORD 0
 .WORD 822.

: SECTOR ADDRESSING TEST (T16)
 PRM16: .WORD 113
 .WORD 1
 .WORD 0
 .WORD 0
 .WORD 0

: TRACK ADDRESSING TEST (T17)
 PRM17: .WORD 1013
 .WORD 1
 .WORD 0
 .WORD 0
 .WORD 0

: DATA TEST (T20)
 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)
 PRM21: .WORD 37
 .WORD 1000.
 .WORD 0
 .WORD 410.
 .WORD 0
 .WORD 821.

: RM03 ACCESS TIME ADJUSTMENT TEST (T22)
 PRM22: .WORD 37
 .WORD 5000.
 .WORD 0

979	003050	000210	.WORD	136.	
980	003052	000000	.WORD	0	
981	003054	000377	.WORD	255.	
982					
983					
984					
985					
986					
987	003056	045520	T7A: .WORD	MSG7	
988	003060	000000	.WORD	0	
989	003062	003142	.WORD	1634.	;(16.67-2%)*2
990	003064	003244	.WORD	1700.	;(16.67+2%)*2
991					
992	003066	045520	T7B: .WORD	MSG7	
993	003070	000000	.WORD	0	
994	003072	003131	.WORD	1625.	;(16.67-2.5%)*2
995	003074	003255	.WORD	1709.	;(16.67+2.5%)*2
996					
997					
998	003076	045520	T7A1: .WORD	MSG7	; ROTATION TEST TABLES FOR RM02 DRIVE
999	003100	000000	.WORD	0	;
1000	003102	004622	.WORD	2450.	; 25.00-2% MS X 100.
1001	003104	004766	.WORD	2550.	; 25.00+2% MS X 100.
1002					
1003	003106	045520	T7B1: .WORD	MSG7	;
1004	003110	000000	.WORD	0	
1005	003112	004605	.WORD	2437.	; 25.00-2.5%
1006	003114	005003	.WORD	2563.	; 25.00+2.5%
1007					
1008	003116	045552	T10: .WORD	MSG10A	
1009	003120	045621	.WORD	MSG10B	
1010	003122	000000	.WORD	0	; NO LOWER LIMIT
1011	003124	001274	.WORD	700.	;(7)*2
1012					
1013	003126	045636	T11: .WORD	MSG11A	
1014	003130	045705	.WORD	MSG11B	
1015	003132	000000	.WORD	0	; NO LOWER LIMIT
1016	003134	006200	.WORD	3200.	;(32)*2
1017					
1018	003136	045722	T12: .WORD	MSG12A	
1019	003140	045764	.WORD	MSG12B	
1020	003142	000000	.WORD	0	; NO LOWER LIMIT
1021	003144	012574	.WORD	5500.	;(55)*2
1022					
1023	003146	003206	PAT.PT: .WORD	PAT0	; TABLE OF POINTERS WHICH POINT TO THE
1024	003150	003246	.WORD	PAT1	; PATTERNS USED BY THE DATA TEST
1025	003152	003306	.WORD	PAT2	
1026	003154	003346	.WORD	PAT3	
1027	003156	003406	.WORD	PAT4	
1028	003160	003446	.WORD	PAT5	
1029	003162	003506	.WORD	PAT6	
1030	003164	003546	.WORD	PAT7	
1031	003166	003606	.WORD	PAT8	
1032	003170	003646	.WORD	PAT9	
1033	003172	003706	.WORD	PAT10	
1034	003174	003746	.WORD	PAT11	

1035	003176	004006	.WORD	PAT12
1036	003200	004046	.WORD	PAT13
1037	003202	004106	.WORD	PAT14
1038	003204	004146	.WORD	PAT15
1039				
1040				
1041				
1042	003206	066666	.WORD	PAT0
1043	003210	066666	.WORD	PAT0
1044	003212	066666	.WORD	PAT0
1045	003214	066666	.WORD	PAT0
1046	003216	066666	.WORD	PAT0
1047	003220	066666	.WORD	PAT0
1048	003222	066666	.WORD	PAT0
1049	003224	066666	.WORD	PAT0
1050	003226	066666	.WORD	PAT0
1051	003230	066666	.WORD	PAT0
1052	003232	066666	.WORD	PAT0
1053	003234	066666	.WORD	PAT0
1054	003236	066666	.WORD	PAT0
1055	003240	066666	.WORD	PAT0
1056	003242	066666	.WORD	PAT0
1057	003244	066666	.WORD	PAT0
1058				
1059	003246	000001	.WORD	PAT1
1060	003250	000003	.WORD	PAT1
1061	003252	000007	.WORD	PAT1
1062	003254	000017	.WORD	PAT1
1063	003256	000037	.WORD	PAT1
1064	003260	000077	.WORD	PAT1
1065	003262	000177	.WORD	PAT1
1066	003264	000377	.WORD	PAT1
1067	003266	000777	.WORD	PAT1
1068	003270	001777	.WORD	PAT1
1069	003272	003777	.WORD	PAT1
1070	003274	007777	.WORD	PAT1
1071	003276	017777	.WORD	PAT1
1072	003300	037777	.WORD	PAT1
1073	003302	077777	.WORD	PAT1
1074	003304	177777	.WORD	PAT1
1075				
1076	003306	177776	.WORD	PAT2
1077	003310	177774	.WORD	PAT2
1078	003312	177770	.WORD	PAT2
1079	003314	177760	.WORD	PAT2
1080	003316	177740	.WORD	PAT2
1081	003320	177700	.WORD	PAT2
1082	003322	177600	.WORD	PAT2
1083	003324	177400	.WORD	PAT2
1084	003326	177000	.WORD	PAT2
1085	003330	176000	.WORD	PAT2
1086	003332	174000	.WORD	PAT2
1087	003334	170000	.WORD	PAT2
1088	003336	160000	.WORD	PAT2
1089	003340	140000	.WORD	PAT2
1090	003342	100000	.WORD	PAT2

1091	003344	000000	.WORD	000000	
1092					
1093	003346	000000	PAT3: .WORD	000000	;PATTERN 3
1094	003350	000000	.WORD	000000	
1095	003352	000000	.WORD	000000	
1096	003354	177777	.WORD	177777	
1097	003356	177777	.WORD	177777	
1098	003360	177777	.WORD	177777	
1099	003362	000000	.WORD	000000	
1100	003364	000000	.WORD	000000	
1101	003366	177777	.WORD	177777	
1102	003370	177777	.WORD	177777	
1103	003372	000000	.WORD	000000	
1104	003374	177777	.WORD	177777	
1105	003376	000000	.WORD	000000	
1106	003400	177777	.WORD	177777	
1107	003402	000000	.WORD	000000	
1108	003404	177777	.WORD	177777	
1109					
1110	003406	000000	PAT4: .WORD	000000	;PATTERN 4
1111	003410	010421	.WORD	010421	
1112	003412	021042	.WORD	021042	
1113	003414	031463	.WORD	031463	
1114	003416	042104	.WORD	042104	
1115	003420	052525	.WORD	052525	
1116	003422	063146	.WORD	063146	
1117	003424	073567	.WORD	073567	
1118	003426	104210	.WORD	104210	
1119	003430	114631	.WORD	114631	
1120	003432	125252	.WORD	125252	
1121	003434	135673	.WORD	135673	
1122	003436	146314	.WORD	146314	
1123	003440	156735	.WORD	156735	
1124	003442	167356	.WORD	167356	
1125	003444	177777	.WORD	177777	
1126					
1127	003446	052525	PAT5: .WORD	052525	;PATTERN 5
1128	003450	052525	.WORD	052525	
1129	003452	052525	.WORD	052525	
1130	003454	125252	.WORD	125252	
1131	003456	125252	.WORD	125252	
1132	003460	125252	.WORD	125252	
1133	003462	052525	.WORD	052525	
1134	003464	052525	.WORD	052525	
1135	003466	125252	.WORD	125252	
1136	003470	125252	.WORD	125252	
1137	003472	052525	.WORD	052525	
1138	003474	125252	.WORD	125252	
1139	003476	052525	.WORD	052525	
1140	003500	125252	.WORD	125252	
1141	003502	052525	.WORD	052525	
1142	003504	125252	.WORD	125252	
1143					
1144	003506	007417	PAT6: .WORD	007417	;PATTERN 6
1145	003510	007417	.WORD	007417	
1146	003512	007417	.WORD	007417	

1147	003514	170360	.WORD	170360
1148	003516	170360	.WORD	170360
1149	003520	170360	.WORD	170360
1150	003522	007417	.WORD	007417
1151	003524	007417	.WORD	007417
1152	003526	170360	.WORD	170360
1153	003530	170360	.WORD	170360
1154	003532	007417	.WORD	007417
1155	003534	170360	.WORD	170360
1156	003536	007417	.WORD	007417
1157	003540	170360	.WORD	170360
1158	003542	007417	.WORD	007417
1159	003544	170360	.WORD	170360
1160				
1161	003546	026455	PAT7: .WORD	026455 ;PATTERN 7
1162	003550	026455	.WORD	026455
1163	003552	026455	.WORD	026455
1164	003554	151322	.WORD	151322
1165	003556	151322	.WORD	151322
1166	003560	151322	.WORD	151322
1167	003562	026455	.WORD	026455
1168	003564	026455	.WORD	026455
1169	003566	151322	.WORD	151322
1170	003570	151322	.WORD	151322
1171	003572	026455	.WORD	026455
1172	003574	151322	.WORD	151322
1173	003576	026455	.WORD	026455
1174	003600	151322	.WORD	151322
1175	003602	026455	.WORD	026455
1176	003604	151322	.WORD	151322
1177				
1178	003606	165555	PAT8: .WORD	165555 ;PATTERN 8
1179	003610	133333	.WORD	133333
1180	003612	165555	.WORD	165555
1181	003614	133333	.WORD	133333
1182	003616	165555	.WORD	165555
1183	003620	133333	.WORD	133333
1184	003622	165555	.WORD	165555
1185	003624	133333	.WORD	133333
1186	003626	165555	.WORD	165555
1187	003630	133333	.WORD	133333
1188	003632	165555	.WORD	165555
1189	003634	133333	.WORD	133333
1190	003636	165555	.WORD	165555
1191	003640	133333	.WORD	133333
1192	003642	165555	.WORD	165555
1193	003644	133333	.WORD	133333
1194				
1195	003646	000001	PAT9: .WORD	000001 ;PATTERN 9
1196	003650	000002	.WORD	000002
1197	003652	000004	.WORD	000004
1198	003654	000010	.WORD	000010
1199	003656	000020	.WORD	000020
1200	003660	000040	.WORD	000040
1201	003662	000100	.WORD	000100
1202	003664	000200	.WORD	000200

1203	003666	000400	.WORD	000400	
1204	003670	001000	.WORD	001000	
1205	003672	002000	.WORD	002000	
1206	003674	004000	.WORD	004000	
1207	003676	010000	.WORD	010000	
1208	003700	020000	.WORD	020000	
1209	003702	040000	.WORD	040000	
1210	003704	100000	.WORD	100000	
1211					
1212	003706	177776	PAT10: .WORD	177776	;PATTERN 10
1213	003710	177775	.WORD	177775	
1214	003712	177773	.WORD	177773	
1215	003714	177767	.WORD	177767	
1216	003716	177757	.WORD	177757	
1217	003720	177737	.WORD	177737	
1218	003722	177677	.WORD	177677	
1219	003724	177577	.WORD	177577	
1220	003726	177377	.WORD	177377	
1221	003730	176777	.WORD	176777	
1222	003732	175777	.WORD	175777	
1223	003734	173777	.WORD	173777	
1224	003736	167777	.WORD	167777	
1225	003740	157777	.WORD	157777	
1226	003742	137777	.WORD	137777	
1227	003744	077777	.WORD	077777	
1228					
1229	003746	172666	PAT11: .WORD	172666	;PATTERN 11
1230	003750	155555	.WORD	155555	
1231	003752	172666	.WORD	172666	
1232	003754	155555	.WORD	155555	
1233	003756	172666	.WORD	172666	
1234	003760	155555	.WORD	155555	
1235	003762	172666	.WORD	172666	
1236	003764	155555	.WORD	155555	
1237	003766	172666	.WORD	172666	
1238	003770	155555	.WORD	155555	
1239	003772	172666	.WORD	172666	
1240	003774	155555	.WORD	155555	
1241	003776	172666	.WORD	172666	
1242	004000	155555	.WORD	155555	
1243	004002	172666	.WORD	172666	
1244	004004	155555	.WORD	155555	
1245					
1246	004006	077777	PAT12: .WORD	077777	;PATTERN 12
1247	004010	137777	.WORD	137777	
1248	004012	157777	.WORD	157777	
1249	004014	167777	.WORD	167777	
1250	004016	173777	.WORD	173777	
1251	004020	175777	.WORD	175777	
1252	004022	176777	.WORD	176777	
1253	004024	177377	.WORD	177377	
1254	004026	177577	.WORD	177577	
1255	004030	177677	.WORD	177677	
1256	004032	177737	.WORD	177737	
1257	004034	177757	.WORD	177757	
1258	004036	177767	.WORD	177767	

1259	004040	177773	.WORD	177773	
1260	004042	177775	.WORD	177775	
1261	004044	177776	.WORD	177776	
1262					
1263	004046	153333	PAT13: .WORD	153333	;PATTERN 13
1264	004050	066667	.WORD	066667	
1265	004052	153333	.WORD	153333	
1266	004054	066667	.WORD	066667	
1267	004056	153333	.WORD	153333	
1268	004060	066667	.WORD	066667	
1269	004062	153333	.WORD	153333	
1270	004064	066667	.WORD	066667	
1271	004066	153333	.WORD	153333	
1272	004070	066667	.WORD	066667	
1273	004072	153333	.WORD	153333	
1274	004074	066667	.WORD	066667	
1275	004076	153333	.WORD	153333	
1276	004100	066667	.WORD	066667	
1277	004102	153333	.WORD	153333	
1278	004104	066667	.WORD	066667	
1279					
1280	004106	000000	PAT14: .WORD	000000	;PATTERN 14
1281	004110	177777	.WORD	177777	
1282	004112	177777	.WORD	177777	
1283	004114	177777	.WORD	177777	
1284	004116	177777	.WORD	177777	
1285	004120	177777	.WORD	177777	
1286	004122	177777	.WORD	177777	
1287	004124	177777	.WORD	177777	
1288	004126	177777	.WORD	177777	
1289	004130	177777	.WORD	177777	
1290	004132	177777	.WORD	177777	
1291	004134	177777	.WORD	177777	
1292	004136	177777	.WORD	177777	
1293	004140	177777	.WORD	177777	
1294	004142	177777	.WORD	177777	
1295	004144	177777	.WORD	177777	
1296					
1297	004146	177777	PAT15: .WORD	177777	;PATTERN 15
1298	004150	000000	.WORD	000000	
1299	004152	000000	.WORD	000000	
1300	004154	000000	.WORD	000000	
1301	004156	000000	.WORD	000000	
1302	004160	000000	.WORD	000000	
1303	004162	000000	.WORD	000000	
1304	004164	000000	.WORD	000000	
1305	004166	000000	.WORD	000000	
1306	004170	000000	.WORD	000000	
1307	004172	000000	.WORD	000000	
1308	004174	000000	.WORD	000000	
1309	004176	000000	.WORD	000000	
1310	004200	000000	.WORD	000000	
1311	004202	000000	.WORD	000000	
1312	004204	000000	.WORD	000000	
1313					
1314					

;DPB (DATA PARAMETER BLOCK)

1371	004256	000	.BYTE	0	;(10) SECTOR ADDRESS OR
1372					:FIRST REG. INDEX
1373	004257	000	.BYTE	0	;(11) TRACK ADDRESS OR
1374					:LAST REG. INDEX
1375	004260	000000	.WORD	0	;(12) CYLINDER ADDRESS
1376	004262	004306	.WORD	RM.REG	;(14) ERROR TABLE POINTER
1377					:POINTS TO THE FIRST OF TWENTY
1378					:LOCATIONS OF WHERE THE DRIVER
1379					:IS TO STORE THE RH70/RM03
1380					:REGISTERS ON AN ERROR. IF LEFT
1381					:ZERO REGISTERS ARE NOT SAVED.
1382	004264	000000	.WORD	0	;(16) STATUS/ERROR INDICATOR
1383					:BIT15=1=>ERROR OCCURRED
1384					:BIT07=1=>DONE
1385					:BIT14-BIT09 AND BIT06-BIT03
1386					:INDICATE TYPE OF ERROR
1387					
1388	004266	000	DTADPB: .BYTE	0	;(0) DRIVE NUMBER
1389	004267	000	.BYTE	0	;(1) OFFSET VALUE OR FMT22, ECT, AND HCI
1390	004270	000	.BYTE	0	;(2) COMMAND
1391	004271	000	.BYTE	0	;(3) PSEL AND A17 AND A16
1392	004272	000000	.WORD	0	;(4) WORD COUNT (MUST BE NEG.)
1393	004274	051776	.WORD	BUFFER	;(6) BUFFER ADDRESS OR
1394					:REGISTER TABLE POINTER
1395	004276	000	.BYTE	0	;(10) SECTOR ADDRESS OR
1396					:FIRST REG. INDEX
1397	004277	000	.BYTE	0	;(11) TRACK ADDRESS OR
1398					:LAST REG. INDEX
1399	004300	000000	.WORD	0	;(12) CYLINDER ADDRESS
1400	004302	004306	.WORD	RM.REG	;(14) ERROR TABLE POINTER
1401					:POINTS TO THE FIRST OF TWENTY
1402					:LOCATIONS OF WHERE THE DRIVER
1403					:IS TO STORE THE RH70/RM03
1404					:REGISTERS ON AN ERROR. IF LEFT
1405					:ZERO REGISTERS ARE NOT SAVED.
1406	004304	000000	.WORD	0	;(16) STATUS/ERROR INDICATOR
1407					:BIT15=1=>ERROR OCCURRED
1408					:BIT07=1=>DONE
1409					:BIT14-BIT09 AND BIT06-BIT03
1410					:INDICATE TYPE OF ERROR
1411					
1412					
1413					
1414					
1415					:SAVE RH70/RM03 REGISTERS HERE ON ERROR
1416	004306	000000	RM.REG: .WORD	0	:RMCS1 (776700) CONTROL & STATUS #1
1417	004310	000000	.WORD	0	:RMWC (776702) WORD COUNT
1418	004312	000000	.WORD	0	:RMB (776704) BUS ADDRESS
1419	004314	000000	.WORD	0	:RMDA (776706) DESIRED SECTOR/TRACK
1420	004316	000000	.WORD	0	:RMCS2 (776710) CONTROL & STATUS #2
1421	004320	000000	.WORD	0	:RMD5 (776712) DISK STATUS
1422	004322	000000	.WORD	0	:RMER1 (776714) ERROR REG. #1
1423	004324	000000	.WORD	0	:RMAS (776716) ATTENTION SUMMARY
1424	004326	000000	.WORD	0	:RMLA (776720) LOOK AHEAD
1425	004330	000000	.WORD	0	:RMDB (776722) DATA BUFFER
1426	004332	000000	.WORD	0	:RMMR1 (776724) MAINTAINABILITY

1427	004334	000000	.WORD	0	:RMDT (776726) DRIVE TYPE
1428	004336	000000	.WORD	0	:RMSN (776730) SERIAL NUMBER
1429	004340	000000	.WORD	0	:RMOF (776732) OFFSET
1430	004342	000000	.WORD	0	:RMDC (776734) DESIRED CYLINDER
1431	004344	000000	.WORD	0	:RMHR (776736) CURRENT CYLINDER
1432	004346	000000	.WORD	0	:RMMR2 (776740) ERROR REG #2
1433	004350	000000	.WORD	0	:RMR2 (776742) ERROR REG #3
1434	004352	000000	.WORD	0	:RMEC1 (776744) ECC POSITION
1435	004354	000000	.WORD	0	:RMEC2 (776746) ECC PATTERN

:STATUS/ERROR MESSAGE POINTER TABLE

1437			STATBL: .WORD	MSGB14	:OFFLINE OR UNSAFE DRIVE REQUESTED
1438	004356	047013	.WORD	MSGB13	:UNLOAD DRIVE REQUESTED
1439	004360	047055	.WORD	MSGB12	:PERSISTENT UNSAFE
1440	004362	047106	.WORD	MSGB11	:PARITY ERROR OCCURRED
1441	004364	047130	.WORD	MSGB10	:FATAL PARITY ERROR
1442	004366	047156	.WORD	MSGB09	:SOFTWARE TIMEOUT ON THIS DRIVE
1443	004370	047201	.WORD	MSGB08	:SOFTWARE TIMEOUT ON ANOTHER DRIVE
1444	004372	047240	.WORD	MSGB06	:ERROR OCCURRED DURING I/O OPERATION
1445	004374	047302	.WORD	MSGB05	:ERROR OCCURRED DURING NON-I/O OPERATION
1446	004376	047346	.WORD	MSGB04	:UNSAFE OCCURRED
1447	004400	047416	.WORD	MSGB03	:AUTOMATIC RECALIBRATE SEQUENCE OCCURRED
1448	004402	047436	.WORD	MSGB02	:DRIVE HAS NOT RESPONDED TO PORT REQUEST
1449	004404	047506	.WORD	MSGB01	:DRIVE HAS BECOME NONEXISTENT
1450	004406	047556	.WORD		

.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
:* RH70 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
:* \$TMPD \$ERRPC RD.ADR RD.WRD

EM3
DH3
DT3
DF3

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

1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465 004410
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479 004410 046144
1480 004412 047614
1481 004414 051170
1482 004416 051622
1483
1484
1485
1486
1487
1488
1489 004420 046207
1490 004422 047631
1491 004424 051174
1492 004426 051626
1493
1494
1495
1496
1497
1498
1499 004430 046245
1500 004432 047716
1501 004434 051212
1502 004436 051632
1503
1504
1505
1506

```

1507      ;*   STMPO  $ERRPC  WRT.ADR WRT.WD  RD.WRD
1508
1509      004440  046302      EM4
1510      004442  047753      DH4
1511      004444  051222      DT4
1512      004446  051636      DF4
1513
1514      ;*   ERROR ITEM 5
1515      ;**   ADDRESS PLUG CHANGE BIT SET
1516      ;**   ERR PC  DRIVE  RMAS  RMDS  RMER1  RMMR2  RMER2
1517      ;**   $ERRPC  $REG1  $REG3  RMERRS  RMERRS+2  RMERRS+4  RMERRS+6
1518
1519      004450  046336      EM5
1520      004452  047631      DH2
1521      004454  051174      DT2
1522      004456  051626      DF2
1523
1524      ;*   ERROR ITEM 6 -- NOT USED
1525
1526      004460  000000      0
1527      004462  000000      0
1528      004464  000000      0
1529      004466  000000      0
1530
1531      ;*   ERROR ITEM 7 -- NOT USED
1532
1533      004470  000000      0
1534      004472  000000      0
1535      004474  000000      0
1536      004476  000000      0
1537
1538      ;*   ERROR ITEM 10
1539      ;**   RH70/RM03 FAILED TO RESPOND TO ADDRESSING
1540      ;**   RMCS1  ERR PC
1541      ;**   RH.ADR  $ERRPC
1542
1543      004500  046372      EM10
1544      004502  050022      DH10
1545      004504  051254      DT10
1546      004506  051642      DF10
1547
1548      ;*   ERROR ITEM 11
1549      ;**   DRIVE SELECTED IS NOT ONLINE
1550      ;**   DRIVE  ERR PC
1551      ;**   $REG2  $ERRPC
1552
1553      004510  046444      EM11
1554      004512  050041      DH11
1555      004514  051260      DT11
1556      004516  051646      DF11
1557
1558      ;*   ERROR ITEM 12
1559      ;**   IMPROPER HEADER DATA
1560      ;**   TEST  ERR PC  TST PC  DRIVE  CYLNDR  TRACK  SECTOR
1561      ;**   $STMPO  $ERRPC  $REG0  CHKDRV  CYL.DS  TRK.DS  SEC.DS
1562      ;**   GDCYL  GDTRK  GDSCTR  BDCYL  BDTRK  BDSCTR

```



```

1563      : *      CYL.DS TRK.DS SEC.DS CYL.RD TRK.RD SEC.RD
1564      : *      CYLNDR, TRACK, AND SECTOR ARE DECIMAL
1565
1566      004520 046501      EM12
1567      004522 050060      DH12
1568      004524 051264      DT12
1569      004526 051652      DF12
1570
1571      : *      ERROR ITEM 13
1572      : *      DATA COMPARE FAILURE
1573      : *      TEST      ERR PC TST PC DRIVE CYLNDR TRACK SECTOR
1574      : *      STMPO      $ERRPC $REGO  CHKDRV CYL.DS TRK.DS SEC.DS
1575      : *      GDDAT      BDDAT  WRDCNT GDADR  BDADR
1576      : *      $GDDAT  $BDDAT  $REG4  $GDADR  $BDADR
1577      : *      CYLNDR, TRACK, SECTOR, AND WRDCNT ARE DECIMAL
1578
1579      004530 046526      EM13
1580      004532 050060      DH12
1581      004534 051316      DT13
1582      004536 051662      DF13
1583
1584      : *      ERROR ITEM 14 -- FOLLOWS #13
1585      : *      $GDDAT  $BDDAT  $REG4  $GDADR  $BDADR
1586
1587      004540 000000      0
1588      004542 000000      0
1589      004544 051334      DT13A
1590      004546 051672      DF14
1591
1592      : *      ERROR ITEM 15
1593      : *      DATA COMPARE FAILURE
1594      : *      TEST      ERR PC TST PC DRIVE CYLNDR TRACK SECTOR
1595      : *      STMPO      $ERRPC $REGO  CHKDRV CYL.DS TRK.DS SEC.DS
1596      : *      GDDAT      BDDAT  WRDCNT GDADR  BDADR
1597      : *      $GDDAT  $BDDAT  $REG4  $GDADR  $BDADR
1598      : *      CYLNDR, TRACK, SECTOR, AND WRDCNT ARE DECIMAL
1599
1600      004550 046526      EM13
1601      004552 050060      DH12
1602      004554 051316      DT13
1603      004556 051662      DF13
1604
1605      : *      ERROR ITEM 16 -- FOLLOWS #15
1606      : *      $GDDAT  $BDDAT  $REG4  $GDADR  $BDADR
1607
1608      004560 000000      0
1609      004562 000000      0
1610      004564 051334      DT13A
1611      004566 051672      DF14
1612
1613      : *      ERROR ITEM 17
1614      : *      DISK ERROR IN TIMING TEST
1615      : *      TEST      ERR PC DRIVE RMCS1  RMS  RMER1  RMMR2  RMER2
1616      : *      STMPO      $ERRPC  CHKDRV RM.REG  RM.REG+12 RM.REG+14 RM.REG+40 RM.REG+42
1617
1618      004570 046553      EM17

```

1619	004572	050274	DH17
1620	004574	051346	DT17
1621	004576	051676	DF17
1622			
1623			::* ERROR ITEM 20
1624			::* CLOCK (KW11-P) OVERFLOW IN TIMING TEST
1625			::* TEST ERR PC DRIVE RMCS1 RMDS RMER1 RMMR2 RMER2
1626			::* STMPO \$ERRPC CHKDRV RM.REG RM.REG+12 RM.REG+14 RM.REG+40 RM.REG+42
1627			
1628	004600	046605	EM20
1629	004602	050274	DH17
1630	004604	051346	DT17
1631	004606	051676	DF17
1632			
1633			::* ERROR ITEM 21
1634			::* DATA COMPARE FAILURE
1635			::* TEST ERR PC TST PC DRIVE CYLNR TRACK
1636			::* STMPO \$ERRPC \$REGD CHKDRV CYL.DS TRK.DS
1637			::* GDDAT BDDAT WRDCNT SECTOR
1638			::* \$REG1 \$BDDAT \$REG4 \$REG1
1639			::* CYLINDR, TRACK, WRDCNT, AND SECTOR ARE DECIMAL
1640			
1641	004610	046526	EM13
1642	004612	050371	DH21
1643	004614	051366	DT21
1644	004616	051702	DF21
1645			
1646			::* ERROR ITEM 22--FOLLOWS #21
1647			::* \$REG1 \$BDDAT \$REG4 \$REG1
1648			
1649	004620	000000	0
1650	004622	000000	0
1651	004624	051402	DT21A
1652	004626	051712	DF22
1653			
1654			::* ERROR ITEM 23
1655			::* DISK ERROR DURING SEEK
1656			::* TEST ERR PC DRIVE CYLNR RMCS1 RMCS2 RMDS
1657			::* STMPO \$ERRPC CHKDRV CYL.DS RM.REG RM.REG+10 RM.REG+12
1658			::* RMER1 RMMR2 RMER2 RMDC RMHR
1659			::* RM.REG+14 RM.REG+40 RM.REG+42 RM.REG+34 RM.REG+36
1660			
1661	004630	046654	EM23
1662	004632	050506	DH23
1663	004634	051412	DT23
1664	004636	051716	DF23
1665			
1666			::* ERROR ITEM 24
1667			::* SEEK NOT COMPLETE WITHIN 120 MS
1668			::* TEST ERR PC DRIVE CYLNR RMCS1 RMCS2 RMDS
1669			::* STMPO \$ERRPC CHKDRV CYL.DS RM.REG RM.REG+10 RM.REG+12
1670			::* RMER1 RMMR2 RMER2 RMDC RMHR
1671			::* RM.REG+14 RM.REG+40 RM.REG+42 RM.REG+34 RM.REG+36
1672			
1673	004640	046703	EM24
1674	004642	050506	DH23

1675 004644 051412
1676 004646 051716

DT23
DF23

1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730

004650

004650 046743
004652 050640
004654 051442
004656 051726

004660 046743
004662 050676
004664 051452
004666 051732

004670 046743
004672 050676
004674 051470

* 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/RM03 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/RM03 ERROR (MESSAGE)
* TEST ERR PC TST PC DRIVE
* STMPO \$ERRPC \$REGO CHKDRV

EM41
DH41
DT41
DF41

* ERROR ITEM 42
* RH70/RM03 ERROR (MESSAGE)
* TEST ERR PC TST PC DRIVE RMCS1 RMCS2 RMDS
* STMPO \$ERRPC \$REGO CHKDRV RM.REG RM.REG+10 RM.REG+12

EM41
DH42
DT42
DF42

* ERROR ITEM 43
* RH70/RM03 ERROR (MESSAGE)
* TEST ERR PC TST PC DRIVE RMCS1 RMCS2 RMDS
* STMPO \$ERRPC \$REGO CHKDRV RM.REG RM.REG+10 RM.REG+12
* RMER1 RMR2 RMER2
* RM.REG+14 RM.REG+40 RM.REG+42

EM41
DH42
DT43

1731 004676 051736

DF43

```

:* 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 RMDS 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

```

1743 004700 046743

EM41

1744 004702 050060

DH12

1745 004704 051514

DT44

1746 004706 051746

DF44

1748

```

:* 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 RMDS 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

```

1758 004710 046743

EM41

1759 004712 050060

DH12

1760 004714 051554

DT45

1761 004716 051762

DF45

1763

```

:* 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 RMDS 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

```

1773 004720 046763

EM46

1774 004722 050060

DH12

1775 004724 051554

DT45

1776 004726 051762

DF45

1777


```

1778
1779
1780 .SBTTL START OF PROGRAM
1781
1782 004730 012737 177777 001306 START3: MOV #-1, @BUSADR ;GET BUSADR FLAG
1783 004736 000402 BR START1A
1784 004740 005037 001306 START1: CLR @BUSADR ;CLR BUSADR FLAG
1785 004744 005037 001304 START1A: CLR @CNTRLC ;NO CONTROL "C"
1786 004750 000411 BR START
1787 004752 012737 177777 001306 START4: MOV #-1, @BUSADR ;SET BUSADR FLAG
1788 004750 000402 BR START2A
1789 004762 005037 001306 START2: CLR @BUSADR ;CLR BUSADR FLAG
1790 004766 012737 177777 001304 START2A: MOV #-1, @CNTRLC ;SET CONTROL "C" FLAG
1791 004774 000005 START: RESET
1792
1793 .SBTTL INITIALIZE THE COMMON TAGS
1794 ;;CLEAR THE COMMON TAGS (SCMTAG) AREA
1795 MOV #SCMTAG, R6 ;;FIRST LOCATION TO BE CLEARED
1796 CLR (R6)+ ;;CLEAR MEMORY LOCATION
1797 CMP #SWR, R6 ;;DONE?
1798 BNE -6 ;;LOOP BACK IF NO
1799 MOV #STACK, SP ;;SETUP THE STACK POINTER
1800 ;;INITIALIZE A FEW VECTORS
1801 MOV #SSCOPE, @IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
1802 MOV #340, @IOTVEC+2 ;;LEVEL 7
1803 MOV #SEERROR, @EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
1804 MOV #340, @EMTVEC+2 ;;LEVEL 7
1805 MOV #STRAP, @TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1806 MOV #340, @TRAPVEC+2 ;;LEVEL 7
1807 MOV SENDCT, SEOPCT ;;SETUP END-OF-PROGRAM COUNTER
1808 MOV #176543, $MINUM ;;PRIME THE RANDOM NUMBER GENERATOR
1809 MOV #123456, $LONUM ;;BOTH HIGH AND LOW WORDS
1810 CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
1811 CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1812 MOV #1, $ERMAX ;;ALLOW ONE ERROR PER TEST
1813 MOV #., $LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1814 MOV #., $LPERR ;;SETUP THE ERROR LOOP ADDRESS
1815 ;;SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
1816 ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
1817 MOV @ERRVEC, -(SP) ;;SAVE ERROR VECTOR
1818 MOV #64$, @ERRVEC ;;SET UP ERROR VECTOR
1819 MOV #DSWR, SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
1820 MOV #DDISP, DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
1821 CMP #-1, @SWR ;;TRY TO REFERENCE HARDWARE SWR
1822 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1823 BR 65$ ;;AND THE HARDWARE SWR IS NOT = -1
1824 MOV #65$, (SP) ;;BRANCH IF NO TIMEOUT
1825 RTI ;;SET UP FOR TRAP RETURN
1826 MOV #SWREG, SWR ;;POINT TO SOFTWARE SWR
1827 MOV #DISPREG, DISPLAY
1828 MOV (SP)+, @ERRVEC ;;RESTORE ERROR VECTOR
1829
1830 CLR $PASS ;;CLEAR PASS COUNT
1831 BITB #APTSIZE, $ENVM ;;TEST USER SIZE UNDER APT
1832 BEQ 67$ ;;YES, USE NON-APT SWITCH
1833 MOV #SSWREG, SWR ;;NO, USE APT SWITCH REGISTER
    
```


1834	005246			67\$:						
1835	005246	012700	001160		MOV	#\$REGAD,RO			:FIRST ADDRESS	
1836	005252	005020		1\$:	CLR	(RO)+			:CLEAR VARIABLE STORAGE	
1837	005254	022700	001210		CMP	#\$BELL,RO			:DONE?	
1838	005260	001374			BNE	1\$:NO--BRANCH	
1839	005262	013737	001476	001150	MOV	2#TIPS,2#STIPS			:SETUP THE STATUS AND BUFFER REG'S	
1840	005270	013737	001500	001152	MOV	2#TPB,2#STPB			:FOR THE TYPE ROUTINE	
1841	005276	005227	177777		INC	#-1			:FIRST START ?	
1842	005302	001027			BNE	3\$:BR IF NOT	
1843	005304	104401	051776		TYPE	TITLE			:TYPE THE PROGRAM'S TITLE	
1844	005310	005737	000042		TST	42			:AUTO ACCEPT OR CHAIN MODE ?	
1845	005314	001007			BNE	2\$:BR IF EITHER	
1846	005316	122737	000077	000041	CMPB	877,41			:LOADED FROM AN RMO3 ?	
1847	005324	001003			BNE	2\$:BR IF NOT	
1848	005326	104401	052052		TYPE	,LOADRV			:INSTRUCT THE OPERATOR TO REMOVE THE PACK	
1849									:ON DRIVE 0 IF DRIVE 0 IS TO BE TESTED	
1850	005332	000000			HALT				:HALT IF LOAD FROM RMO3.RMO2	
1851	005334	105737	000041	2\$:	TSTB	2#41			:LOADED FROM PAPER TAPE ?	
1852	005340	001410			BEQ	3\$:BR IF NOT	
1853	005342	004737	052346		JSR	PC,\$SIZE			:SIZE THE MEMORY	
1854	005346	023727	052442	100000	CMP	SLSTAD,#100000			:16K OR MORE ON THE SYSTEM ?	
1855	005354	103002			BHIS	3\$:BR IF YES	
1856	005356	104401	052250		TYPE	,NOLOAD			:INFORM THE OPERATOR THAT THE 'XXDP' LOADER	
1857									:WILL BE OVERWRITTEN	
1858	005362	004737	023366	3\$:	JSR	PC,\$TKINT			:TURN ON THE TTY KEYBOARD INTERRUPT	
1859				.SBTTL	GET VALUE FOR SOFTWARE	SWITCH REGISTER				
1860	005366	005737	000042		TST	2#42			:ARE WE RUNNING UNDER XXDP/ACT?	
1861	005372	001012			BNE	68\$:BRANCH IF YES	
1862	005374	123727	001240	000001	CMPB	\$ENV,#1			:ARE WE RUNNING UNDER APT?	
1863	005402	001406			BEQ	68\$:BRANCH IF YES	
1864	005404	023727	001140	000176	CMP	SWR,#SWREG			:SOFTWARE SWITCH REG SELECTED?	
1865	005412	001005			BNE	69\$:BRANCH IF NO	
1866	005414	104406			GTSWR				:GET SOFT-SWR SETTINGS	
1867	005416	000403			BR	69\$				
1868	005420	112737	000001	001134	68\$:	MOV B	#1,\$AUTOB		::SET AUTO-MODE INDICATOR	
1869	005426			69\$:						
1870	005426	005227	177777		INC	#-1			:SEE IF FIRST START	
1871	005432	001002			BNE	SRTINT			:BR IF NOT	
1872	005434	004737	052444		JSR	PC,GETADR			:GET OR CHECK THE RH70 ADDRESS	
1873	005440	104401	001215	SRTINT:	TYPE	,\$CRLF			:CR-LF	
1874	005444	004737	026054		JSR	PC,2#LP.AVL			:CHECK FOR A LINE PRINTER	
1875	005450	005037	177776		CLR	2#PS			:ENSURE THE PRIORITY = 0	
1876	005454	012737	000001	001104	MOV	#1,\$ICNT			:SET ITERATION COUNT TO 1	
1877	005462	004737	033646		JSR	PC,2#GETSWR			:GO CHECK FOR CONTROL SWITCHES	
1878	005466	004737	026116		JSR	PC,2#ST.CLK			:INITIALIZE THE CLOCK	
1879	005472	004737	036754	SETVEC:	JSR	PC,RMINIT			:CHECK THE DRIVE STATUS	
1880	005476	012737	177777	036676	MOV	#-1,\$SAVEFG			:SET THE SAVE REGISTERS FLAG	
1881	005504	062727	177777	000000	ADD	#-1,#0			:FIRST START ?	
1882	005512	103003			BCC	11\$:BR IF YES	
1883	005514	005737	001304		TST	CNTRLC			:CONTROL 'C' SWITCH SET ?	
1884	005520	001101			BNE	SRTDRV			:CONTINUE IF YES	
1885	005522	012737	000340	177776	11\$:	MOV	#PR7,PS		:SET PRIORITY TO 7	
1886	005530	005004			CLR	R4			:DRIVE TABLE POINTER	
1887	005532	104401	001215		TYPE	,\$CRLF			:CR-LF	
1888	005536	104401	045204		TYPE	,\$SYSTAT			:TYPE STATUS HEADING	
1889	005542			1\$:						

Line	Address	Offset	Value	Label	Register	Comment
1890	005542	010446		MOV	R4, -(SP)	:: SAVE R4 FOR TYPEOUT
1891						:: TYPE DRIVE NUMBER
1892	005544	104403		TYPOS		:: GO TYPE--OCTAL ASCII
1893	005546	002		.BYTE	2	:: TYPE 2 DIGIT(S)
1894	005547	000		.BYTE	0	:: SUPPRESS LEADING ZEROS
1895	005550	104401	046141	TYPE	, MSG.SP	:: SPACES
1896	005554	104401	046141	TYPE	, MSG.SP	:: SPACES
1897	005560	105764	036610	TSTB	DRVSTA(R4)	:: CHECK DRIVE'S STATUS
1898	005564	100416		BMI	4\$:: BR IF UNSAFE
1899	005566	001020		BNE	5\$:: BR IF ONLINE
1900	005570	105764	036620	TSTB	DRVTYP(R4)	:: SEE IF OFFLINE OR NONEXISTENT
1901	005574	001404		BEQ	2\$:: BR IF NONEXISTENT
1902	005576	100006		BPL	3\$:: BR IF OFFLINE
1903	005600	104401	045300	TYPE	, NOTRM	:: DRIVE NOT AN RMO3
1904	005604	000437		BR	4\$:: CHECK NEXT DRIVE
1905	005606	104401	045253	TYPE	, NOTPRS	:: DRIVE NOT PRESENT
1906	005612	000434		BR	4\$:: CHECK NEXT DRIVE
1907	005614	104401	045232	TYPE	, UNTOFF	:: DRIVE OFFLINE
1908	005620	000405		BR	6\$:: PRINT DRIVE TYPE
1909	005622	104401	045270	TYPE	, NOTSAF	:: DRIVE UNSAFE
1910	005626	000402		BR	6\$:: PRINT DRIVE TYPE
1911	005630	104401	045243	TYPE	, UNTON	:: DRIVE ONLINE
1912	005634	104401	046141	TYPE	, MSG.SP	:: SPACES
1913	005640	012737	045324	MOV	#1, RMO3, 8\$:: ADDRESS OF RMO3 MESSAGE
1914	005646	122764	000004	CMPB	#4, DRVTYP(R4)	:: RMO3 ?
1915	005654	001411		BEQ	7\$:: TYPE IT IF YES
1916	005656	012737	045317	MOV	#1, RMO2, 8\$:: MESSAGE OF RMO2
1917	005664	122764	000005	CMPB	#5, DRVTYP(R4)	:: RMO2 ?
1918	005672	001402		BEQ	7\$:: BRANCH IF SO
1919	005674	104401	045300	TYPE	, NOTRM	:: TYPE DRIVE NOT AN RMO3 IF NOT
1920	005700	104401		TYPE		:: TYPE THE DRIVE TYPE MESSAGE
1921	005702	000000		.WORD	0	:: MESSAGE ADDRESS HERE
1922	005704	104401	001215	TYPE	, \$CRLF	:: CR-LF
1923	005710	005204		INC	R4	:: INCREMENT DRIVE NUMBER/TABLE POINTER
1924	005712	020427	000010	CMP	R4, #8.	:: FINISHED ?
1925	005716	001311		BNE	1\$:: BR IF NOT
1926	005720	005037	177776	CLR	PS	:: SET PRIORITY BACK TO '0'
1927	005724	005737	001304	TST	#CNTRLC	:: CONTROL "C" START/RESTART?
1928	005730	001417		BEQ	1\$:: NO--BRANCH
1929	005732	013746	001302	MOV	SAVCSW, -(SP)	:: GET THE PREVIOUS 'C.SWR' CONTENTS
1930	005736	063716	001300	ADD	C.SWR, (SP)	:: SET UP TO SEE IF 'BIT00' IS DIFFERENT
1931	005742	032726	000001	BIT	#BIT00, (SP)+	:: IS 'BIT00' DIFFERENT ?
1932	005746	001405		BEQ	9\$:: BR IF NOT
1933	005750	013737	001300	MOV	C.SWR, SAVCSW	:: STORE PRESENT 'C.SWR' VALUE
1934	005756	004737	026374	JSR	PC, LODFLT	:: RESET PARAMETERS TO THEIR DEFAULT VALUES
1935	005762	004737	034076	JSR	PC, #GT.PRM	:: GET PARAMETERS
1936	005766	000420		BR	4\$	
1937	005770	004737	026374	JSR	PC, LODFLT	:: SETUP DEFAULT PARAMETERS
1938	005774	005037	001312	CLR	DRVSEL	:: NO DRIVES SELECTED
1939	006000	005000		CLR	RO	:: DETERMINE THE DRIVES THAT
1940	006002	012701	000001	MOV	#1, R1	:: ARE AVAILABLE FOR TESTING
1941	006006	105760	036610	TSTB	DRVSTA(RO)	
1942	006012	003403		BLE	3\$	
1943	006014	156037	036724	BISB	ATABIT(RO), #DRVSEL	
1944	006022	005200		INC	RO	
1945	006024	106301		ASLB	R1	

E06

MD-11-DZRMF-B RMO3/RMO2 EXTENDED DRIVE TEST MACY11 30(1046) 22-NOV-77 08:04 PAGE 39
 CZRMFB.P11 21-NOV-77 14:10 GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0069

1946	006026	001367				BNE	2\$		
1947	006030	005037	036700		4\$:	CLR	2#SEEKFG	; CLEAR SEEK FLAG	
1948	006034	032737	000400	001300		BIT	#SW08,2#C.SWR	; DO SEEK BEFORE DATA TRANSFER?	
1949	006042	001002				BNE	5\$; YES--BRANCH	
1950	006044	005137	036700			COM	2#SEEKFG	; NO	
1951	006050	122737	000011	000041	5\$:	CMPB	#11,41 ;LOADED	FROM AN RMO3 ?	
1952	006056	001003				BNE	10\$; BR IF NOT	
1953	006060	042737	000001	001312		BIC	#BIT00,DRVSEL	; CLEAR THE DRIVE 0 SELECTION BIT	
1954	006066	104401	045331		10\$:	TYPE	DRIVES	; 'DRIVES(S) TO BE TESTED'	
1955	006072	005037	021234			CLR	2#SENDCT	; DETERMINE PASSES TO MAKE AND	
1956	006076	005000				RO		THE DRIVES TO BE TESTED	
1957	006100	013701	001312			MOV	2#DRVSEL,R1	; ANY DRIVES SELECTED?	
1958	006104	001004				BNE	6\$; YES--BRANCH	
1959	006106	104401	045362			TYPE	,NONE	; 'NONE'	
1960									
1961	006112	000137	021052			JMP	2#SEOP	; GO TO END OF PROGRAM	
1962	006116	006201			6\$:	ASR	R1	; REPORT THE DRIVES TO BE TESTED	
1963	006120	103011				BCC	7\$		
1964	006122	005237	021234			INC	2#SENDCT	; GIVE THIS DRIVE A PASS	
1965	006126	010046				MOV	RO,-(SP)	; SAVE RO FOR TYPEOUT	
1966	006130	104403				TYPOS		; GO TYPE--OCTAL ASCII	
1967	006132	001				.BYTE	1	; TYPE 1 DIGIT(S)	
1968	006133	000				.BYTE	0	; SUPPRESS LEADING ZEROS	
1969	006134	005701				TST	R1	; MORE DRIVES?	
1970	006136	001404				BEQ	8\$; NO--BRANCH	
1971	006140	104401	045367			TYPE	COMMA		
1972	006144	005200			7\$:	INC	RO	; FORM DRIVE NUMBER	
1973	006146	000763				BR	6\$		
1974	006150	013737	021234	021226	8\$:	MOV	2#SENDCT,2#SEOPCT		
1975	006156	005737	001324			TST	2#CLKSTA	; KW11-P AVAILABLE	
1976	006162	003006				BGT	RSTRT1	; YES--BRANCH	
1977	006164	032737	036000	001314		BIT	#36000,2#TSTNMS	; NO--ANY TIMING TESTS TO BE PERFORMED?	
1978	006172	001402				BEQ	RSTRT1	; NO--BRANCH	
1979	006174	104401	045371			TYPE	NOCLK	; 'NO KW11-P CLOCK, TIMING TESTS WILL NOT BE PERFORMED'	
1980	006200	005737	001312		RSTRT1:	TST	DRVSEL	; ANY DRIVES SELECTED ?	
1981	006204	001002				BNE	1\$; BR IF YES	
1982	006206	000137	004762			JMP	START2	; GET DRIVE SELECTION ENTRY	
1983								& RESTART AT BEGINNING ----->	
1984									
1985	006212	005037	001334		1\$:	CLR	CHKDRV	; INIT. THE CHECK DRIVE KEY	
1986	006216	012737	000001	001336		MOV	#1,DRVMSK	; START TO CHECK DESIRED DRIVES	
1987	006224	033737	001336	001312	RSTRT2:	BIT	DRVMSK,DRVSEL	; IS THIS DRIVE SELECTED?	
1988	006232	001010				BNE	DRVOK	; YES--GO CHECK IF DRIVE IS READY FOR TESTING	
1989	006234	012706	001100		RESTART:	MOV	#STACK,SP	; SETUP THE STACK POINTER	
1990	006240	005237	001334			INC	CHKDRV	; MOVE TO NEXT DRIVE NUMBER	
1991	006244	106337	001336			ASLB	DRVMSK	; POSITION THE MASK	
1992	006250	103753				BCS	RSTRT1	; BRANCH IF THE DRIVE NUMBER NEEDS INITIALIZED	
1993	006252	000764				BR	RSTRT2		
1994									
1995	006254	013702	001334		DRVOK:	MOV	CHKDRV,R2	; PICKUP THE DRIVE NUMBER	
1996	006260	105762	036610			TSTB	DRVSTA(R2)	; IS DESIRED DRIVE ON-LINE?	
1997	006264	003005				BGT	1\$; YES, BRANCH	
1998	006266	104011				ERROR	11	; DRIVE SELECTED IS NOT ONLINE	
1999	006270	043737	001336	001312		BIC	DRVMSK,DRVSEL	; CLEAR DRIVE'S SELECTION BIT	
2000	006276	000756				BR	RESTART	; RETURN	
2001	006300	010237	004206		1\$:	MOV	R2,2#DPB.A	; SET THE DRIVE NUMBER INTO THE DPB'S	

2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152

```
;* 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.
```

```
TSTO:
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 (SERFLG)
JSR PC,LODPRM ;LOAD THE PARAMETERS FOR THE TEST
MOV #TESTO,2#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
MOV $TSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
MOV #RPT,$TIMES ;GET THE ITERATION COUNT
MOVB #25,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
MOV #0,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
```

```
: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 20$ ;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 $RCLF
MOV RO,-(SP) ;;SAVE RO FOR TYPEOUT
;;TEST NUMBER
;;GO TYPE--OCTAL ASCII
;;TYPE 2 DIGIT(S)
;;SUPPRESS LEADING ZEROS
20$:
TYPE ,RCLF
```

```
MOV #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 #EXITO,2#BYPASS ;GO TO EXITO ON ERROR
MOV #TESTO,$LPADR ;SETUP LOOP ADDRESS
TESTO: MOV #STACK,SP ;SET UP STACK POINTER
JSR RO,2#CALL.A ;GO EXECUTE THE COMMAND
EXITO: JSR RO,2#CALL.B ;GO EXECUTE THE COMMAND
;LOOP
SCOPE
```

*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
```

```

2153
2154 006714
2155 006714 000240
2156 006716 033737 001510 001314
2157 006724 001002
2158 006726 000137 007142
2159 006732 012737 000001 001102 64$:
2160
2161 006740 004737 026632
2162 006744 012737 007124 001110
2163 006752 013777 001102 172162
2164 006760 013737 001570 001204
2165 006766 112737 000031 001115
2166 006774 012737 000001 001224
2167
2168
2169 007002 032777 010000 172130
2170 007010 001413
2171 007012 013700 001102
2172 007016 042700 177000
2173 007022 104401 001215
2174 007026 010046
2175
2176 007030 104403
2177 007032 002
2178 007033 000
2179 007034 104401 001215 20$:
2180 007040
2181
2182 007040 005037 001446
2183 007044 113737 001606 004236
2184 007052 113737 001610 004256
2185 007060 113737 001600 004237
2186 007066 113737 001602 004257
2187 007074 013737 001572 004240
2188 007102 013737 001574 004260
2189 007110 012737 007140 001332
2190 007116 012737 007124 001106
2191 007124 012706 001100
2192 007130 004037 027410
2193 007134 004037 027206
2194 007140 000004
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208

:*****
↑ST1:
NOP
BIT 2#BITS+<1*2>,TSTNMS ;DO THIS TEST?
BNE 64$ ;YES--BRANCH
JMP TST2 ;NO--GO TO THE NEXT TEST
MOV #1,2#STSTNM ;SET UP TEST NUMBER AND
;CLEAR THE ERROR FLAG (SERFLG)
JSR PC,LODPRM ;LOAD THE PARAMETERS FOR THE TEST
MOV #TEST1,2#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
MOV STSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
MOV 2#RPT,2#TIMES ;GET THE ITERATION COUNT
MOVB #25,2#ERMAX ;MAX ERRORS ALLOWED FOR TEST
MOV #1,2#STSTN ;SET TEST NUMBER IN APT MAIL BOX

;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 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

CLR BASFLG ;CLEAR BAD SECTOR ENCOUNTER FOR THE DRIVE
MOVB 2#FS,2#DPB.B+10 ;FS
MOVB 2#LS,2#DPB.C+10 ;LS
MOVB 2#FT,2#DPB.B+11 ;FT
MOVB 2#LT,2#DPB.C+11 ;LT
MOV 2#FC,2#DPB.B+12 ;FC
MOV 2#LC,2#DPB.C+12 ;LC
MOV #EXIT1,2#BYPASS ;GO TO EXIT1 ON ERROR
MOV #TEST1,2#LPADR ;SETUP LOOP ADDRESS
TEST1: MOV #STACK,SP ;SET THE STACK POINTER
JSR R0,2#CALL.C ;GO EXECUTE THE COMMAND
JSR R0,2#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.
;*****

```



```

2209 007142          TST2:
2210 007142 000240    NOP
2211 007144 033737 001512 001314    BIT      @#BITS+(2*2),TSTNMS ;DO THIS TEST?
2212 007152 001002    BNE      64$ ;YES--BRANCH
2213 007154 000137 007424    JMP      TST3 ;NO--GO TO THE NEXT TEST
2214 007160 012737 000002 001102 64$:    MOV      #2,@#STSTNM ;SET UP TEST NUMBER AND
2215                                     ;CLEAR THE ERROR FLAG (SERFLG)
2216 007166 004737 026632    JSR      PC.LODPRM ;LOAD THE PARMETERS FOR THE TEST
2217 007172 012737 007316 001110    MOV      #TEST2,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
2218 007200 013777 001102 171734    MOV      $STSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2219 007206 013737 001570 001204    MOV      @#RPT,$TIMES ;GET THE ITERATION COUNT
2220 007214 112737 000031 001115    MOVVB   #25,$SERMAX ;MAX ERRORS ALLOWED FOR TEST
2221 007222 012737 000002 001224    MOV      #2,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
2222
2223                                     ;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
2224 007230 032777 010000 171702    BIT      #SW12,@SWR ;CHECK FOR SWITCH 12 SET
2225 007236 001413    BEQ      20$ ;IF = 0, THEN DON'T TYPE TEST NUMBER
2226 007240 013700 001102    MOV      $STSTNM,RO ;ELSE, TYPE THE NUMBER
2227 007244 042700 177000    BIC      #177000,RO ;MASK OUT THE HIGH ORDER BYTE
2228 007250 104401 001215    TYPE    $CRLF
2229 007254 010046    MOV      RO,-(SP) ;;SAVE RO FOR TYPEOUT
2230                                     ;;TEST NUMBER
2231 007256 104403    TYPOS   ;;GO TYPE--OCTAL ASCII
2232 007260 002 ;;TYPE 2 DIGIT(S)
2233 007261 000 ;;SUPPRESS LEADING ZEROS
2234 007262 104401 001215    TYPE    ,CRLF
2235 007266          20$:
2236
2237 007266 012737 007274 001106    MOV      #1$,$LPADR ;SETUP LOOP ADDRESS
2238 007274 113737 001606 004236 1$:    MOVVB   @#FS,@#DPB.B+10 ;FS
2239 007302 113737 001600 004237    MOVVB   @#FT,@#DPB.B+11 ;FT
2240 007310 012737 007422 001332    MOV      @#EXIT2,@#BYPASS ;GO TO EXIT2 ON ERROR
2241 007316 013737 001572 004240    TEST2:  MOV      @#FC,@#DPB.B+12 ;FC
2242 007324 012737 007324 001110    MOV      #,$SLPERR ;SETUP THE ERROR LOOP ADDRESS
2243 007332 012706 001100    MOV      #STACK,SP ;LOAD THE STACK POINTER
2244 007336          INCSK:
2245 007336 004037 027206    JSR      RO,@CALL.B ;GO EXECUTE THE COMMAND
2246 007342 063737 001576 004240    ADD      @#IC,@#DPB.B+12 ;MOVE TO NEXT CYLINDER
2247 007350 023737 001574 004240    CMP      @#LC,@#DPB.B+12 ;OUT OF CYLINDERS?
2248 007356 002367    BGE      INCSK ;NO--BRANCH
2249 007360 013737 001574 004240    MOV      @#LC,@#DPB.B+12
2250 007366 012737 007366 001110    MOV      #,$SLPERR ;SETUP THE ERROR LOOP ADDRESS
2251 007374 012706 001100    MOV      #STACK,SP ;LOAD THE STACK POINTER
2252 007400          DECSK:
2253 007400 004037 027206    JSR      RO,@CALL.B ;GO EXECUTE THE COMMAND
2254 007404 163737 001576 004240    SUB      @#IC,@#DPB.B+12
2255 007412 023737 001572 004240    CMP      @#FC,@#DPB.B+12
2256 007420 003767    BLE      DECSK
2257 007422 000004    EXIT2:  SCOPE ;LOOP
2258
2259 ;*****
2260 ;*TEST 3 STEPPING SEEK TEST
2261
2262 ;* THIS TEST WILL COMMAND SEEK CYCLES TO CYLINDER 0,1,2,4
2263 ;* 8,16,32,64,128, AND 256. AT THE COMPLETION OF EACH SEEK
2264 ;* COMMAND THE PROPER INDICATORS ARE EXAMINED TO ENSURE PROPER

```

```

2265 ;* OPERATION.
2266
2267 ;*****
2268 †ST3:
2269 007424 000240 NOP
2270 007426 033737 001514 001314 BIT 2#BITS+(3*2),TSTNMS ;DO THIS TEST?
2271 007434 001002 BNE 64$ ;YES--BRANCH
2272 007436 000137 007664 JMP TST4 ;NO--GO TO THE NEXT TEST
2273 007442 012737 000003 001102 64$: MOV #3,2#STSTNM ;SET UP TEST NUMBER AND
2274 ;CLEAR THE ERROR FLAG (SERFLG)
2275 007450 004737 026632 JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
2276 007454 012737 007600 001110 MOV #TEST3,2#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
2277 007462 013777 001102 171452 MOV $TSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2278 007470 013737 001570 001204 MOV 2#RPT,$TIMES ;GET THE ITERATION COUNT
2279 007476 112737 000031 001115 MOVB #25,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
2280 007504 012737 000003 001224 MOV #3,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
2281
2282 ;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
2283 007512 032777 010000 171420 BIT #SW12,2#SWR ;CHECK FOR SWITCH 12 SET
2284 007520 001413 BEQ 20$ ;IF = 0, THEN DON'T TYPE TEST NUMBER
2285 007522 013700 001102 MOV $TSTNM,RO ;ELSE, TYPE THE NUMBER
2286 007526 042700 177000 BIC #177000,RO ;MASK OUT THE HIGH ORDER BYTE
2287 007532 104401 001215 TYPE $SCRLF
2288 007536 010046 MOV RO,-(SP) ;SAVE RO FOR TYPEOUT
2289 ;TEST NUMBER
2290 007540 104403 TYPOS ;GO TYPE--OCTAL ASCII
2291 007542 002 .BYTE 2 ;TYPE 2 DIGIT(S)
2292 007543 000 .BYTE 0 ;SUPPRESS LEADING ZEROS
2293 007544 104401 001215 TYPE $SCRLF
2294 007550 20$:
2295
2296 007550 012737 007556 001106 MOV #1$,$LPADR ;SETUP TEST LOOP ADDRESS
2297 007556 113737 001606 004236 1$: MOVB 2#FS,2#DPB.B+10 ;FS
2298 007564 113737 001600 004237 MOVB 2#FT,2#DPB.B+11 ;FT
2299 007572 012737 007662 001332 MOV #EXIT3,2#BYPASS ;GO TO BYPASS ON ERROR
2300 007600 013737 001572 004240 TEST3: MOV 2#FC,2#DPB.B+12 ;FC
2301 007606 012737 007606 001110 MOV #,$SLPERR ;SETUP THE ERROR LOOP ADDRESS
2302 007614 012706 001100 MOV #,$STACK,SP ;LOAD THE STACK POINTER
2303 007620 004037 027206 JSR RO,2#CALL.B ;GO EXECUTE THE COMMAND
2304 007624 013701 001576 MOV IC,R1 ;CYLINDER 1
2305 007630 012737 007630 001110 MOV #,$SLPERR ;SETUP THE ERROR LOOP ADDRESS
2306 007636 012706 001100 MOV #,$STACK,SP ;LOAD THE STACK POINTER
2307 007642 010137 004240 1$: MOV R1,2#DPB.B+12 ;DESIRED CYLINDER
2308 007646 004037 027206 JSR RO,2#CALL.B ;GO EXECUTE THE COMMAND
2309 007652 006301 ASL R1 ;MOVE TO NEXT CYLINDER
2310 007654 020137 001574 CMP R1,2#LC ;DONE?
2311 007660 003770 BLE 1$ ;NO--LOOP
2312 EXIT3: SCOPE
2313
2314 ;*****
2315 ;*TEST 4 OSCILLATING SEEK TEST
2316
2317 ;* THIS TEST WILL COMMAND SEEK CYCLES FROM "FC" TO "NC" AND BACK
2318 ;* TO "FC". "NC" STARTS AT "FC" AND INCREMENTS BY "IC" UP TO CYLINDER
2319 ;* "LC", THEN IS DECREMENTED BY "IC" BACK TO CYLINDER "FC". AT THE
2320 ;* COMPLETION OF EVERY SEEK COMMAND THE PROPER INDICATORS ARE

```



```

2321 ;* EXAMINED TO ENSURE PROPER OPERATION.
2322
2323 ;*****
2324 †ST4:
2325 007664 000240 NOP
2326 007666 033737 001516 001314 BIT 2#BITS+(4*2),TSTNMS ;DO THIS TEST?
2327 007674 001002 BNE 64$ ;YES--BRANCH
2328 007676 000137 010316 JMP TST5 ;NO--GO TO THE NEXT TEST
2329 007702 012737 000004 001102 64$: MOV #4,2#STSTNM ;SET UP TEST NUMBER AND
2330 ;CLEAR THE ERROR FLAG (SERFLG)
2331 007710 004737 026632 JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
2332 007714 012737 010054 001110 MOV #TEST4,2#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
2333 007722 013777 001102 171212 MOV STSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2334 007730 013737 001570 001204 MOV 2#RPT,$TIMES ;GET THE ITERATION COUNT
2335 007736 112737 000031 001115 MOV#B #25,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
2336 007744 012737 000004 001224 MOV #4,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
2337
2338 ;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
2339 007752 032777 010000 171160 BIT #SW12,2#SWR ;CHECK FOR SWITCH 12 SET
2340 007760 001413 BEQ 20$ ;IF = 0, THEN DON'T TYPE TEST NUMBER
2341 007762 013700 001102 MOV STSTNM,RO ;ELSE, TYPE THE NUMBER
2342 007766 042700 177000 BIC #177000,RO ;MASK OUT THE HIGH ORDER BYTE
2343 007772 104401 001215 TYPE $CRLF
2344 007776 010046 MOV RO,-(SP) ;SAVE RO FOR TYPEOUT
2345 ;TEST NUMBER
2346 010000 104403 TYPOS ;GO TYPE--OCTAL ASCII
2347 010002 002 .BYTE 2 ;TYPE 2 DIGIT(S)
2348 010003 000 .BYTE 0 ;SUPPRESS LEADING ZEROS
2349 010004 104401 001215 TYPE ,CRLF
2350 010010 20$:
2351
2352 010010 012737 010016 001106 MOV #15,$LPADR ;SETUP LOOP ADDRESS
2353 010016 113737 001606 004236 1$: MOV#B 2#FS,2#DPB.B+10 ;FS
2354 010024 113737 001600 004237 MOV#B 2#FT,2#DPB.B+11 ;FT
2355 010032 012737 010314 001332 MOV #EXIT4,2#BYPASS ;GO TO EXIT4 ON ERROR
2356 010040 005002 CLR R2 ;CLEAR STALL SWITCH (NO STALL)
2357 010042 032737 010000 001300 BIT #SW12,2#C.SWR ;STALL REQUIRED?
2358 010050 001401 BEQ TEST4 ;NO--BRANCH
2359 010052 005102 COM R2 ;YES--SET SWITCH
2360 010054 013701 001572 TEST4: MOV 2#FC,R1 ;SET NC TO FC
2361 010060 005037 001416 CLR 2#STALLO ;START AT ZERO IF STALLS REQUIRED
2362 010064 012737 010064 001110 MOV #,$SLPERR ;SETUP THE ERROR LOOP ADDRESS
2363 010072 012706 001100 MOV #,$STACK,SP ;LOAD THE STACK POINTER
2364 010076 010137 004240 1$: MOV R1,2#DPB.B+12 ;NC
2365 010102 004037 027206 JSR RO,2#CALL.B ;GO EXECUTE THE COMMAND
2366 010106 005702 TST R2 ;STALL?
2367 010110 001403 BEQ 2$ ;NO--BRANCH
2368 010112 004037 030606 JSR RO,2#STALL ;YES--GO TO STALL ROUTINE
2369 010116 001416 .WORD STALLO ;TIME POINTER
2370 010120 013737 001572 004240 2$: MOV FC,2#DPB.B+12 ;FC
2371 010126 004037 027206 JSR RO,2#CALL.B ;GO EXECUTE THE COMMAND
2372 010132 005702 TST R2 ;STALL?
2373 010134 001413 BEQ 3$ ;NO--BRANCH
2374 010136 004037 030606 JSR RO,2#STALL ;YES--GO TO STALL ROUTINE
2375 010142 001416 .WORD STALLO ;TIME POINTER
2376 010144 005237 001416 INC 2#STALLO ;UPDATE THE TIME

```


M06

MD-11-DZRMF-B RMO3/RMO2 EXTENDED DRIVE TEST
CZRMFB.P11 21-NOV-77 14:10 T4

MACY11 30(1046) 22-NOV-77 08:04 PAGE 47
OSCILLATING SEEK TEST

SEQ 0077

2377	010150	023737	001442	001416		CMP	%MXSTAL,%STALLO	:TIME TO BIG?
2378	010156	003347				BGT	1\$:NO--BRANCH
2379	010160	005037	001416			CLR	%STALLO	:YES--START OVER AT ZERO
2380	010164	063701	001576		3\$:	ADD	%IC,R1	:MOVE TO NEXT CYLINDER
2381	010170	020137	001574			CMP	R1,%LC	:LAST CYLINDER COMPLETED?
2382	010174	003740				BLE	1\$:NO--BRANCH
2383	010176	013701	001574			MOV	%LC,R1	:SET NC TO LC
2384	010202	012737	010202	001110		MOV	%SLPERR	:SETUP THE ERROR LOOP ADDRESS
2385	010210	012706	001100			MOV	%STACK,SP	:LOAD THE STACK POINTER
2386	010214	010137	004240		4\$:	MOV	R1,%DPB.B+12	:NC
2387	010220	004037	027206			JSR	RO,%CALL.B	:GO EXECUTE THE COMMAND
2388	010224	005702				TST	R2	:STALL?
2389	010226	001403				BEQ	5\$:NO--BRANCH
2390	010230	004037	030606			JSR	RO,%STALL	:YES--GO TO STALL ROUTINE
2391	010234	001416				.WORD	STALLO	:TIME POINTER
2392	010236	013737	001574	004240	5\$:	MOV	%LC,%DPB.B+12	:LC
2393	010244	004037	027206			JSR	RO,%CALL.B	:GO EXECUTE THE COMMAND
2394	010250	005702				TST	R2	:STALL?
2395	010252	001413				BEQ	6\$:NO--BRANCH
2396	010254	004037	030606			JSR	RO,%STALL	:YES--GO TO STALL ROUTINE
2397	010260	001416				.WORD	STALLO	:TIME POINTER
2398	010262	005237	001416			INC	%STALLO	:UPDATE STALL TIME
2399	010266	023737	001442	001416		CMP	%MXSTAL,%STALLO	:TIME TOO BIG?
2400	010274	003347				BGT	4\$:NO--BRANCH
2401	010276	005037	001416			CLR	%STALLO	:YES--SET STALL TIME BACK TO ZERO
2402	010302	163701	001576		6\$:	SUB	%IC,R1	:NEXT CYLINDER
2403	010306	020137	001572			CMP	R1,%FC	:DONE?
2404	010312	002340				BGE	4\$:NO--BRANCH
2405	010314	000004			EXIT4:	SCOPE		:LOOP

*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.

*TESTS:

2420	010316					NOP		
2421	010316	000240				BIT	%BITS+(5*2),TSTNMS	:DO THIS TEST?
2422	010320	033737	001520	001314		BNE	64\$:YES--BRANCH
2423	010326	001002				JMP	TST6	:NO--GO TO THE NEXT TEST
2424	010330	000137	010562			MOV	%5,%STSTNM	:SET UP TEST NUMBER AND
2425	010334	012737	000005	001102	64\$:			:CLEAR THE ERROR FLAG (SERFLG)
2426						JSR	PC,LODPRM	:LOAD THE PARAMETERS FOR THE TEST
2427	010342	004737	026632			MOV	%TESTS,%SLPERR	:SETUP THE LOOP ON ERROR ADDRESS
2428	010346	012737	010472	001110		MOV	%STSTNM,%DISPLAY	:LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2429	010354	013777	001102	170560		MOV	%RPT,\$TIMES	:GET THE ITERATION COUNT
2430	010362	013737	001570	001204		MOVB	%25,\$ERMAX	:MAX ERRORS ALLOWED FOR TEST
2431	010370	112737	000031	001115		MOV	%5,\$TESTN	:SET TEST NUMBER IN APT MAIL BOX
2432	010376	012737	000005	001224				

N06

```

2433
2434
2435 010404 032777 010000 170526
2436 010412 001413
2437 010414 013700 001102
2438 010420 042700 177000
2439 010424 104401 001215
2440 010430 010046
2441
2442 010432 104403
2443 010434 002
2444 010435 000
2445 010436 104401 001215
2446 010442
2447
2448 010442 012737 010450 001106
2449 010450 113737 001606 004236
2450 010456 113737 001600 004237
2451 010464 012737 010560 001332
2452 010472 013701 001572
2453 010476 013702 001574
2454 010502 012737 010502 001110
2455 010510 012706 001100
2456 010514 010137 004240
2457 010520 004037 027206
2458 010524 010237 004240
2459 010530 004037 027206
2460 010534 063701 001576
2461 010540 163702 001576
2462 010544 020137 001574
2463 010550 003003
2464 010552 020237 001572
2465 010556 002356
2466 010560 000004
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479 010562
2480 010562 000240
2481 010564 033737 001522 001314
2482 010572 001002
2483 010574 000137 011072
2484 010600 012737 000006 001102
2485
2486 010606 004737 026632
2487 010612 012737 010736 001110
2488 010620 013777 001102 170314

; 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 $TSTNM, R0 ; ELSE, TYPE THE NUMBER
BIC #177000, R0 ; MASK OUT THE HIGH ORDER BYTE
TYPE $CRLF
MOV R0, -(SP) ; SAVE R0 FOR TYPEOUT
; TEST NUMBER
; GO TYPE--OCTAL ASCII
; TYPE 2 DIGIT(S)
; SUPPRESS LEADING ZEROS
20$:
MOV #1$, $SLPADR ; SETUP LOOP ADDRESS
1$:
MOV $FS, @DPB.B+10 ; FS
MOV $FT, @DPB.B+11 ; FT
MOV $EXITS, @BYPASS ; GO TO EXITS ON ERROR
TESTS:
MOV $FC, R1 ; START NC1 AT FC
MOV $LC, R2 ; START NC2 AT LC
MOV #, $LPERR ; SETUP THE ERROR LOOP ADDRESS
1$:
MOV $STACK, SP ; LOAD THE STACK POINTER
MOV R1, @DPB.B+12 ; NC1
JSR R0, @CALL.B ; GO EXECUTE THE COMMAND
MOV R2, @DPB.B+12 ; NC2
JSR R0, @CALL.B ; GO EXECUTE THE COMMAND
ADD @IC, R1 ; NEXT NC1
SUB @IC, R2 ; NEXT NC2
CMP R1, @LC ; DONE?
BGT EXITS ; YES--BRANCH
CMP R2, @FC ; ? ; NO--BRANCH
BGE 1$ ; LOOP
EXITS:
SCOPE

; *****
; *TEST 6 SERVO ADDRESSING LOGIC NOISE GENERATOR
; *****
; *
; * 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.
; *****
†TST6:
NOP
BIT @BITS+(<6*2>), TSTNMS ; DO THIS TEST?
BNE 64$ ; YES--BRANCH
JMP TST7 ; NO--GO TO THE NEXT TEST
64$:
MOV #6, @STSTNM ; SET UP TEST NUMBER AND
; CLEAR THE ERROR FLAG ($ERFLG)
JSR PC, LODPRM ; LOAD THE PARAMETERS FOR THE TEST
MOV #TST6, @SLPERR ; SETUP THE LOOP ON ERROR ADDRESS
MOV $TSTNM, @DISPLAY ; LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER

```



```

2489 010626 013737 001570 001204
2490 010634 112737 000031 001115
2491 010642 012737 000006 001224
2492
2493
2494 010650 032777 010000 170262
2495 010656 001413
2496 010660 013700 001102
2497 010664 042700 177000
2498 010670 104401 001215
2499 010674 010046
2500
2501 010676 104403
2502 010700 002
2503 010701 000
2504 010702 104401 001215
2505 010706
2506
2507 010706 012737 010714 001106
2508 010714 113737 001606 004236
2509 010722 113737 001600 004237
2510 010730 012737 011070 001332
2511 010736 013701 001572
2512 010742 013702 001574
2513 010746 162702 000005
2514 010752 012737 010752 001110
2515 010760 012706 001100
2516 010764 020102
2517 010766 003040
2518 010770 010137 004240
2519 010774 004037 027206
2520 011000 062737 000004 004240
2521 011006 004037 027206
2522 011012 162737 000003 004240
2523 011020 004037 027206
2524 011024 062737 000002 004240
2525 011032 004037 027206
2526 011036 162737 000001 004240
2527 011044 004037 027206
2528 011050 062737 000003 004240
2529 011056 004037 027206
2530 011062 063701 001576
2531 011066 000736
2532 011070 000004

```

```

MOV @#APT,STIMES ;GET THE ITERATION COUNT
MOV#B @#25,$SERMAX ;MAX ERRORS ALLOWED FOR TEST
MOV @#6,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

;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 $STNM,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
TYPE ,SCLF

20$:
MOV @#1$,SLPADR ;SETUP LOOP ADDRESS
1$:
MOV#B @#FS,@#DPB.B+10 ;FS
MOV#B @#FT,@#DPB.B+11 ;FT
MOV @#EXIT6,@#BYPASS ;GO TO EXIT6 ON ERROR
TEST6:
MOV @#FC,R1 ;PICKUP "FC"
MOV @#LC,R2 ;FORM LAST CYLINDER THAT
SUB @#5,R2 ;IS AVAILABLE FOR TESTING
MOV @#,$SLPERR ;SETUP THE ERROR LOOP ADDRESS
MOV @#STACK,SP ;LOAD THE STACK POINTER
1$:
CMP R1,R2 ;LAST CYLINDER
BGT EXIT6 ;YES--BRANCH
MOV R1,@#DPB.B+12 ;NC
JSR R0,@#CALL.B ;GO EXECUTE THE COMMAND
ADD @#4,@#DPB.B+12 ;NC+4
JSR R0,@#CALL.B ;GO EXECUTE THE COMMAND
SUB @#3,@#DPB.B+12 ;NC+1
JSR R0,@#CALL.B ;GO EXECUTE THE COMMAND
ADD @#2,@#DPB.B+12 ;NC+3
JSR R0,@#CALL.B ;GO EXECUTE THE COMMAND
SUB @#1,@#DPB.B+12 ;NC+2
JSR R0,@#CALL.B ;GO EXECUTE THE COMMAND
ADD @#3,@#DPB.B+12 ;NC+5
JSR R0,@#CALL.B ;GO EXECUTE THE COMMAND
ADD @#IC,R1
BR 1$
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'.
;*****

```



```

2545 011072          TST7:
2546 011072 000240      NOP
2547 011074 033737 001524 001314  BIT      3#BITS+(<7*2>),TSTNMS ;DO THIS TEST?
2548 011102 001002      BNE      64$ ;YES--BRANCH
2549 011104 000137 011516  JMP      TST10 ;NO--GO TO THE NEXT TEST
2550 011110 012737 000007 001102 64$: MOV      #7,#STSTNM ;SET UP TEST NUMBER AND
2551                                     ;CLEAR THE ERROR FLAG (SERFLG)
2552 011116 004737 026632  JSR      PC,LODPRM ;LOAD THE PARAMETERS FOR THE TEST
2553 011122 012737 011260 001110  MOV      #TST7,#$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
2554 011130 013777 001102 170004  MOV      $TSTNM,$DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2555 011136 013737 001570 001204  MOV      #RPT,$TIMES ;GET THE ITERATION COUNT
2556 011144 112737 000031 001115  MOV      #25,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
2557 011152 012737 000007 001224  MOV      #7,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
2558
2559                                     ;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
2560 011160 032777 010000 167752  BIT      #SW12,$SWR ;CHECK FOR SWITCH 12 SET
2561 011166 001413      BEQ      20$ ;IF = 0, THEN DON'T TYPE TEST NUMBER
2562 011170 013700 001102  MOV      $TSTNM,R0 ;ELSE, TYPE THE NUMBER
2563 011174 042700 177000  BIC      #17000,R0 ;MASK OUT THE HIGH ORDER BYTE
2564 011200 104401 001215  TYPE     $,R0
2565 011204 010046      MOV      R0,-(SP) ;SAVE R0 FOR TYPEOUT
2566                                     ;TEST NUMBER
2567 011206 104403      TYPOS    ;GO TYPE--OCTAL ASCII
2568 011210 002 ;TYPE 2 DIGIT(S)
2569 011211 000 ;SUPPRESS LEADING ZEROS
2570 011212 104401 001215  TYPE     $,R0
2571 011216          20$:
2572
2573 011216 113737 001600 004237  MOV      FT,DPB.B+11 ;LOAD STARTING TRACK ADDRESS
2574 011224 112737 000105 004210  MOV      #SEEK,#DPB.A+2 ;SEEK=COMMAND
2575 011232 112737 000173 004230  MOV      #READHD,DPB.B+2 ;READ HEADER & DATA COMMAND
2576 011240 013704 036736  MOV      RMADR,R4 ;UNIBUS ADDRESS OF THE RH70
2577 011244 012737 011514 001332  MOV      #EXIT7,BYPASS ;ERROR TERMINATION ADDRESS
2578 011252 012737 011260 001106  MOV      #TST7,$LPADR ;SETUP THE LOOP ON TEST ADDRESS
2579 011260 012706 001100  TEST7: MOV      #STACK,SP ;SETUP THE STACK POINTER
2580 011264 013737 001572 004240  MOV      FC,DPB.B+12 ;INITIAL CYLINDER ADDRESS
2581 011272 023737 001572 001574  CMP      FC,LC ;CYLINDER LIMITS THE SAME ?
2582 011300 001422      BEQ      1$ ;OR IF THEY ARE
2583 011302 004737 025532  JSR      PC,$RAND ;CYCLE THE RANDOM NUMBER GENERATOR
2584 011306 013746 025630  MOV      $HNUM,-(SP) ;USE THE HIGH RANDOM NUMBER
2585 011312 005046      CLR      -(SP) ;UPPER DIVIDEND
2586 011314 013746 001574  MOV      LC,-(SP) ;FORM THE DIVISOR
2587 011320 005216      INC      (SP) ;INCREMENT
2588 011322 163716 001572  SUB      FC,(SP) ;SUBTRACT THE LOWER LIMIT
2589 011326 004737 025634  JSR      PC,$DIV ;DIVIDE
2590 011332 062637 004240  ADD      (SP)+,DPB.B+12 ;ADD THE REMAINDER TO THE INITIAL CYLINDER
2591 011336 005726      TST     (SP)+ ;DISCARD THE QUOTIENT
2592 011340 013737 004240 004220  MOV      DPB.B+12,DPB.A+12 ;COPY NEW CYLINDER ADDRESS
2593 011346          1$:
2594 011346 012737 011346 001110  MOV      #,$LPERR ;SETUP THE ERROR LOOP ADDRESS
2595 011354 012706 001100  MOV      #STACK,SP ;LOAD THE STACK POINTER
2596 011360 004037 027054  JSR      R0,#CALL.A ;GO EXECUTE THE COMMAND
2597 011364 012737 011364 001110  MOV      #,$LPERR ;SETUP THE ERROR LOOP ADDRESS
2598 011372 012706 001100  MOV      #STACK,SP ;LOAD THE STACK POINTER
2599 011376 113764 004206 000010  MOV      DPB.A,RMCS2(R4) ;SELECT THE DRIVE
2600 011404 016446 000020  MOV      RMLA(R4),-(SP) ;GET THE LOOK AHEAD REGISTER

```

2601	011410	006316			ASL	(SP)	:ALIGN THE SECTOR ADDRESS
2602	011412	006316			ASL	(SP)	:ALIGN THE SECTOR ADDRESS
2603	011414	000316			SWAB	(SP)	:PUT ADDRESS IN LOWER BYTE
2604	011416	105766	000001		TSTB	1(SP)	:IN THE 1ST 20% OF SECTOR ?
2605	011422	001401			BEG	2\$:BR IF YES
2606	011424	105216			INCB	(SP)	:INCREMENT THE SECTOR ADDRESS
2607	011426	105216		2\$:	INCB	(SP)	:INCREMENT THE SECTOR ADDRESS
2608	011430	112637	004276		MOVB	(SP)+,DTADPB+10	:LOAD THE DPB
2609	011434	013746	001712		MOV	PRMLM+22,-(SP)	:PUT LAST SECTOR ADDRESS ON THE STACK
2610	011440	005216			INC	(SP)	:INCREMENT IT
2611	011442	122637	004276		CMPB	(SP)+,DTADPB+10	:NEW SECTOR ADDRESS TOO LARGE ?
2612	011446	103007			BHIS	4\$:BR IF NOT
2613	011450	103403			BLO	3\$:BR IF ADDRESS IS 2 GREATER
2614	011452	105037	004276		CLRB	DTADPB+10	:RESET TO SECTOR ADDRESS 0
2615	011456	000403			BR	4\$:CONTINUE
2616	011460	112737	000001	004276	MOVB	#1,DTADPB+10	:RESET ADDRESS TO SECTOR 1
2617	011466			4\$:			
2618	011466	004037	027206		JSR	RD, @CALL.B	:GO EXECUTE THE COMMAND
2619	011472	105237	004237		INCB	DPB.B+11	:INCREMENT THE TRACK ADDRESS
2620	011476	123737	004237	001602	CMPB	DPB.B+11,LT	:MAXIMUM ?
2621	011504	101403			BLOS	EXIT7	:BR IF NOT
2622	011506	113737	001600	004237	MOVB	FT,DPB.B+11	:RELOAD STARTING TRACK ADDRESS
2623	011514	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 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 FREQUENTLY ENOUGH WITHIN THE REQUIRED RANGE TO PERMIT THIS TEST TO BE EFFECTIVE.

:ST10:

2650	011516				NOP		
2651	011516	000240			BIT	@BITS+<10*2>,TSTNMS	:DO THIS TEST?
2652	011520	033737	001526	001314	BNE	64\$:YES--BRANCH
2653	011526	001002			JMP	TST11	:NO--GO TO THE NEXT TEST
2654	011530	000137	012666		MOV	#10,@\$TSTNM	:SET UP TEST NUMBER AND
2655	011534	012737	000010	001102	64\$:		:CLEAR THE ERROR FLAG (\$ERFLG)
2656							

2657	011542	004737	026632		JSR	PC,LODPRM	;LOAD THE PARMETERS FOR THE TEST	
2658	011546	012737	011770	001110	MOV	#TEST10,SLPERR	;SETUP THE LOOP ON ERROR ADDRESS	
2659	011554	013777	001102	167360	MOV	STSTNM,DISPLAY	;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER	
2660	011562	013737	001570	001204	MOV	RPT,STIMES	;GET THE ITERATION COUNT	
2661	011570	112737	000031	001115	MOVB	#25,SEMAX	;MAX ERRORS ALLOWED FOR TEST	
2662	011576	012737	000010	001224	MOV	#10,STSTN	;SET TEST NUMBER IN APT MAIL BOX	
2663								
2664								
2665	011604	032777	010000	167326				
2666	011612	001413			BIT	#SW12,SWR	;CHECK FOR SWITCH 12 SET	
2667	011614	013700	001102		BEQ	20\$;IF = 0, THEN DON'T TYPE TEST NUMBER	
2668	011620	042700	177000		MOV	STSTNM,R0	;ELSE, TYPE THE NUMBER	
2669	011624	104401	001215		BIC	#177000,R0	;MASK OUT THE HIGH ORDER BYTE	
2670	011630	010046			TYPE	\$SCRLF		
2671					MOV	R0,-(SP)	;SAVE R0 FOR TYPEOUT	
2672	011632	104403						
2673	011634	002			TYPOS		;TEST NUMBER	
2674	011635	000			.BYTE	2	;GO TYPE--OCTAL ASCII	
2675	011636	104401	001215		.BYTE	0	;TYPE 2 DIGIT(S)	
2676	011642				TYPE	,SCRLF	;SUPPRESS LEADING ZEROS	
2677								
2678	011642	012737	011650	001106				
2679	011650				20\$:			
2680	011650	112737	000105	004210	1\$:	MOV	#1\$,SLPADR	;SETUP THE LOOP ADDRESS
2681	011656	112737	000161	004270				
2682	011664	113737	001600	004277	MOVB	#SEEK,DPB.A+2	;SEEK=COMMAND	
2683	011672	013737	001572	004220	MOVB	#WRITE,DTADPB+2	;COMMAND	
2684	011700	013737	001572	004300	MOV	FT,DTADPB+11	;TRACK ADDRESS FOR THE WRITE	
2685	011706	013737	001572	001614	MOV	FC,DPB.A+12	;CYLINDER ADDRESS FOR THE SEEK	
2686	011714	013737	001576	001430	MOV	FC,DTADPB+12	;CYLINDER ADDRESS FOR THE WRITE	
2687	011722	012737	176000	004272	MOV	FC,NC1	;STARTING CYLINDER	
2688	011730	012737	051776	004274	MOV	IC,DELTA	;CYLINDER INCREMENT VALUE	
2689	011736	005000			MOV	#-<256,*,>,DTADPB+4	;WORD COUNT	
2690	011740	004737	032602		MOV	#BUFFER,DTADPB+6	;BUFFER ADDRESS	
2691	011744	005001			RO		;PATTERN POINTER (WC PATTERN)	
2692	011746	113701	004206		CLR	PC,SETBUF	;LOAD THE WRITE BUFFER	
2693	011752	013704	036736		CLR	R1	;CLEAR REGISTER	
2694	011756	004737	044616		MOVB	DPB.A,R1	;LOAD DRIVE ADDRESS	
2695	011762	012737	012664	001332	MOV	RMADR,R4	;UNIBUS ADDRESS OF THE RH70	
2696	011770				JSR	PC,CLIQUE	;CLEAR THE OPERATION QUEUES	
2697	011770	012737	011770	001110	MOV	#EXIT10,BYPASS	;ERROR EXIT FROM TEST	
2698	011776	012706	001100		TEST10:			
2699	012002	012737	000340	177776	MOV	#,SLPERR	;SETUP THE ERROR LOOP ADDRESS	
2700	012010	005737	001324		MOV	#STACK,SP	;LOAD THE STACK POINTER	
2701	012014	001415			MOV	#PR7,PS	;SET PRIORITY TO 7	
2702	012016	100405			TST	CLKSTA	;SEE WHICH CLOCK ON SYSTEM	
2703	012020	017746	167432		BEQ	3\$;BR IF NO CLOCK	
2704	012024	013746	001456		BMI	1\$;BR IF KW11-L CLOCK	
2705	012030	000404			MOV	PKV,-(SP)	;SAVE THE VECTOR	
2706	012032	017746	167432		MOV	PKV,-(SP)	;SAVE THE VECTOR ADDRESS	
2707	012036	013746	001470		BR	2\$;CONTINUE	
2708	012042	012776	012620	000000	1\$:	MOV	LKV,-(SP)	;SAVE THE 'L' CLOCK VECTOR
2709	012050	012777	031274	024662	2\$:	MOV	#TST10B,(SP)	;CHANGE THE VECTOR
2710	012056	012737	000010	001424	3\$:	MOV	#DORTI,ARMVEC	;CHANGE THE RMO3 VECTOR
2711	012064	012764	000040	000010	MOV	#8,SEKTR	;LOAD THE SEEK TIMER	
2712	012072	110164	000010		MOV	#BIT05,RMCS2(R4)	;INIT THE MASSBUS	
					MOVB	R1,RMCS2(R4)	;RESELECT THE DRIVE	

2769	012426	004037	027632			JSR	RO,DRVCL1	:WAIT FOR OPERATION TO COMPLETE
2770	012432	023727	001426	001750	5\$:	CMP	SEKCNT,#1000.	:FINISHED SEEKS ?
2771	012440	001026				BNE	6\$:BR IF NOT
2772	012442	005037	001426			CLR	SEKCNT	:CLEAR THE SEEK COUNT
2773	012446	063737	001576	001614		ADD	IC,NC1	:ADD THE INCREMENT
2774	012454	023737	001614	001574		CMP	NC1,LC	:EXCEEDED THE CYLINDER LIMIT ?
2775	012462	103100				BHIS	EXIT10	:BR IF IT HAS
2776	012464	013737	001574	001430		MOV	LC,DELTA	:GET THE NEXT 'ZONE' ADDRESS
2777	012472	163737	001614	001430		SUB	NC1,DELTA	:CHECK THE DIFFERENCE
2778	012500	023737	001576	001430		CMP	IC,DELTA	:DIFFERENCE GREATER THAN THE INCREMENT ?
2779	012506	101003				BHI	6\$:BR IF IT IS
2780	012510	013737	001576	001430		MOV	IC,DELTA	:USE THE INCREMENT PARAMETER
2781	012516	005237	001426		6\$:	INC	SEKCNT	:COUNT THE NEXT SEEK
2782	012522	023737	001572	001574		CMP	FC,LC	:BEGINNING AND ENDING CYLINDERS THE SAME ?
2783	012530	001002				BNE	7\$:BR IF NOT
2784	012532	000137	011770			JMP	TEST10	:BR IF THEY ARE
2785	012536	013737	001614	004220	7\$:	MOV	NC1,DPB.A+12	:RESET THE CYLINDER ADDRESS
2786	012544	004737	025532			JSR	PC,\$RAND	:CYCLE THE RANDOM NUMBER GENERATOR
2787	012550	013746	025630			MOV	\$HNUM,-(SP)	:USE THE HIGH RANDOM NUMBER
2788	012554	005046				CLR	-(SP)	:CLEAR THE UPPER DIVIDEND
2789	012556	013746	001430			MOV	DELTA,-(SP)	:FORM THE DIVISOR
2790	012562	005216				INC	(SP)	:INCREMENT
2791	012564	004737	025634			JSR	PC,\$DIV	:DIVIDE
2792	012570	062637	004220			ADD	(SP)+,DPB.A+12	:ADD THE REMAINDER TO THE INITIAL CYLINDER
2793	012574	005726				TST	(SP)+	:DISCARD THE QUOTIENT
2794	012576	023737	004220	004300		CMP	DPB.A+12,DTADPB+12	:SAME CYLINDER SELECTED AS LAST TIME ?
2795	012604	001754				BEQ	7\$:BR IF IT WAS
2796	012606	013737	004220	004300		MOV	DPB.A+12,DTADPB+12	:COPY NEW CYLINDER ADDRESS
2797	012614	000137	011770			JMP	TEST10	:CONTINUE
2798	012620	005337	001424		TST10B:	DEC	SEKTR	:DECREMENT THE SEEK TIMER
2799	012624	001016				BNE	1\$:CONTINUE IF NOT DONE
2800	012626	012702	004206			MOV	#DPB.A,R2	:DPB ADDRESS
2801	012632	004737	044134			JSR	PC,\$VRH70	:SAVE THE REGISTERS
2802	012636	104024				ERROR	24	:TIMEOUT DURING SEEK
2803	012640	012764	000040	000010		MOV	#BIT05,RMC52(R4)	:INIT THE MASSBUS
2804	012646	110164	000010			MOVB	R1,RMC52(R4)	:RESELECT THE DRIVE
2805	012652	016676	000002	000000		MOV	2(SP),2(SP)	:RESTORE THE CLOCK VECTOR ADDRESS
2806	012660	000401				BR	EXIT10	:ABORT THE TEST
2807	012662	000002			1\$:	RTI		:RETURN
2808	012664	000004			EXIT10:	SCOPE		:LOOP ?

```

2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824

```

```

*****
*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.
*****

```

```

†ST11:      NOP

```

```

2825 012670 033737 001530 001314 BIT 2#BITS+(11*2),TSTNMS ;DO THIS TEST?
2826 012676 001002 BNE 64$ ;YES--BRANCH
2827 012700 000137 013152 JMP TST12 ;NO--GO TO THE NEXT TEST
2828 012704 012737 000011 001102 64$: MOV #11,2#STSTNM ;SET UP TEST NUMBER AND
2829 ;CLEAR THE ERROR FLAG (SERFLG)
2830 012712 004737 026632 JSR PC,LOPRM ;LOAD THE PARAMETERS FOR THE TEST
2831 012716 012737 013072 001110 MOV #TST11,2#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
2832 012724 013777 001102 166210 MOV $TSTNM,DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2833 012732 013737 001570 001204 MOV 2#RPT,$TIMES ;GET THE ITERATION COUNT
2834 012740 112737 000031 001115 MOV#S $SERMAX ;MAX ERRORS ALLOWED FOR TEST
2835 012746 012737 000011 001224 MOV #11,$TSTN ;SET TEST NUMBER IN APT MAIL BOX
2836
2837 ;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
2838 012754 032777 010000 166156 BIT #SW12,2SWR ;CHECK FOR SWITCH 12 SET
2839 012762 001413 BEQ 20$ ;IF = 0, THEN DON'T TYPE TEST NUMBER
2840 012764 013700 001102 MOV $TSTNM,RO ;ELSE, TYPE THE NUMBER
2841 012770 042700 177000 BIC #177000,RO ;MASK OUT THE HIGH ORDER BYTE
2842 012774 104401 001215 TYPE $CRLF
2843 013000 010046 MOV RO,-(SP) ;;SAVE RO FOR TYPEOUT
2844 ;;TEST NUMBER
2845 013002 104403 TYPOS ;;GO TYPE--OCTAL ASCII
2846 013004 002 .BYTE 2 ;;TYPE 2 DIGIT(S)
2847 013005 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
2848 013006 104401 001215 TYPE , $CRLF
2849 013012 20$:
2850
2851 013012 012737 013020 001106 MOV #1$, $LPADR ;SETUP THE LOOP ADDRESS
2852 013020 113737 001606 004236 1$: MOV#B FS,DPB.B+10 ;SECTOR ADDRESS
2853 013026 113737 001606 004256 MOV#B FS,DPB.C+10 ;SECTOR ADDRESS
2854 013034 113737 001600 004237 MOV#B FT,DPB.B+11 ;TRACK ADDRESS
2855 013042 113737 001600 004257 MOV#B FT,DPB.C+11 ;TRACK ADDRESS
2856 013050 013737 001572 004240 MOV FC,DPB.B+12 ;STARTING CYLINDER ADDRESS
2857 013056 013737 001572 004260 MOV FC,DPB.C+12 ;STARTING CYLINDER ADDRESS
2858 013064 012737 013150 001332 MOV #EXIT11,BYPASS ;TEST ABORT EXIT
2859 013072 012706 001100 TEST11: MOV #STACK,$P ;SETUP THE STACK POINTER
2860 013076 1$:
2861 013076 004037 027410 JSR RO,2#CALL.C ;GO EXECUTE THE COMMAND
2862 013102 004037 027206 JSR RO,2#CALL.B ;GO EXECUTE THE COMMAND
2863 013106 063737 001576 004260 ADD IC,DPB.C+12 ;INCREMENT THE ENDING CYLINDER ADDRESS
2864 013114 023737 001574 004260 CMP LC,DPB.C+12 ;CHECK IF EXCEEDING MAXIMUM
2865 013122 002365 BGE 1$ ;BR IF NOT
2866 013124 013737 001572 004260 MOV FC,DPB.C+12 ;RESET ENDING CYLINDER ADDRESS
2867 013132 063737 001576 004240 ADD IC,DPB.B+12 ;INCREMENT THE STARTING ADDRESS
2868 013140 023737 001574 004240 CMP LC,DPB.B+12 ;EXCEEDING MAXIMUM ?
2869 013146 002353 BGE 1$ ;BR IF NOT
2870 013150 000004 EXIT11: SCOPE ;LOOP ?
2871

```


2928	013263	000				BYTE	0	;;SUPPRESS LEADING ZEROS
2929	013264	104401	001215			TYPE	,SCRLF	
2930	013270			20\$:				
2931								
2932	013270	005737	001324			TST	Q#CLKSTA	:KW11-P CLOCK?
2933	013274	003002				BGT	1\$:YES--START TEST
2934	013276	000137	014042			JMP	TST13	:NO--GO TO NEXT TEST
2935	013302	012737	013302	001106	1\$:	MOV	#1\$, \$LPADR	:SETUP LOOP ADDRESS
2936	013310	004037	031112			JSR	RO, Q#SRCHOO	:DO A MASSBUS INIT & RECAL
2937	013314	000402				BR	2\$:RETURN HERE IF NO ERROR
2938	013316	000137	014040			JMP	EXIT12	:RETURN HERE IF ERROR
2939	013322	013764	001572	000034	2\$:	MOV	Q#FC, RMDA(R4)	:FC
2940	013330	013746	001606			MOV	Q#FS, -(SP)	:FS
2941	013334	113766	001600	000001		MOVB	Q#FT, 1(SP)	:FT
2942	013342	012664	000006			MOV	(SP)+, RMDA(R4)	:LOAD FT/FS
2943	013346	012737	014040	001206		MOV	#EXIT12, \$ESCAPE	:ESCAPE TO EXIT12 ON ERROR
2944	013354	005005				CLR	R5	:COUNT UP
2945						MOV	#T7A, R3	:60HZ PARAMETERS
2946						BIT	#SW06, Q#C.SWR	:60 HZ?
2947						BEQ	TEST12	:YES--BRANCH
2948						MOV	#T7B, R3	:NO--50 HZ PARAMETERS
2949								:SETUP PARAMETER TABLE FOR RMO2/RMO3
2950	013356	010046				MOV	RO, -(SP)	:SAVE RO
2951	013360	113700	004266			MOVB	DTADPB, RO	:DRIVE ADDRESS
2952	013364	032737	000100	001300		BIT	#SW06, Q#C.SWR	:60 HZ ?
2953	013372	001417				BEQ	3\$:BRANCH IF SO
2954	013374	012703	003066			MOV	#T7B, R3	:50 HZ TABLE FOR RMO3
2955	013400	012737	003066	014036		MOV	#T7B, TP50	:50 HZ TABLE FOR RMO3
2956	013406	122760	000004	036620		CMPB	#4, DRV TYP (RO)	:A RMO3 DRIVE ?
2957	013414	001424				BEQ	4\$:BRANCH IF SO
2958	013416	012703	003106			MOV	#T7B1, R3	:OTHERWISE LOAD THE TABLE FOR RMO2
2959	013422	012737	003106	014036		MOV	#T7B1, TP50	
2960	013430	000416				BR	4\$:EXIT
2961	013432	012737	003056	014026	3\$:	MOV	#T7A, TP60	:60 HZ TABLE FOR RMO3
2962	013440	012703	003056			MOV	#T7A, R3	:60 HZ TABLE FOR RMO3
2963	013444	122760	000004	036620		CMPB	#4, DRV TYP (RO)	:A RMO3 DRIVE ?
2964	013452	001405				BEQ	4\$:BRANCH IF SO
2965	013454	012737	003076	014026		MOV	#T7A1, TP60	:OTHERWISE LOAD THE TABLE FOR RMO2
2966	013462	012703	003076			MOV	#T7A1, R3	
2967	013466	012600			4\$:	MOV	(SP)+, RO	:RESTORE RO
2968	013470	000240				NOP		:EXIT
2969	013472	012706	001100			TEST12:	#STACK, SP	:SETUP STACK
2970	013476	012701	000012			MOV	#10, R1	:TIME 10 SEARCHES
2971	013502	004737	031276			JSR	PC, Q#STR TMR	:INITIALIZE THE TIMERS
2972	013506	012777	013714	165742		MOV	#7\$, QPKV	:SETUP VECTOR IN CASE OF OVERFLOW
2973	013514	012777	031274	023216		MOV	#DORTI, QRMVEC	:SETUP RMO3 VECTOR
2974	013522	005077	165736		1\$:	CLR	QPKB	:START COUNTING AT ZERO
2975	013526	012777	000131	165726		MOV	#131, QPKCS	:INT.EN., COUNT UP AT 100KHZ
2976	013534	012714	000131			MOV	#SEARCH, (R4)	:START A SEARCH
2977	013540	000001				WAIT		:WAIT ON INTERRUPT
2978	013542	042777	000101	165712		BIT	#101, QPKCS	:STOP THE CLOCK
2979	013550	032764	040000	000012		BIT	#BIT14, RMDA(R4)	:ERROR?
2980	013556	001415				BEQ	2\$:NO--BRANCH
2981	013560	104412				SAVREG		:SAVE RO-R5
2982	013562	012702	004266			MOV	#DTADPB, R2	:DPB POINTER
2983	013566	004737	044134			JSR	PC, Q#SVRH70	:SAVE ALL THE RH70/RMO3 REGISTERS

K07

MD-11-DZRMF-B RMO3/RMO2 EXTENDED DRIVE TEST
CZRMFB.P11 21-NOV-77 14:10 T12

MACY11 30(1046) 22-NOV-77 08:04 PAGE 58
ROTATIONAL SPEED TIMING TEST

SEQ 0088

2984	013572	012764	000040	000010	MOV	#BIT05, RMCS2(R4)	: MASSBUS CLEAR	
2985	013600	013764	004266	000010	MOV	DTADPB, RMCS2(R4)	: SELECT DRIVE	
2986	013606	104413			RESREG		: RESTORE RO-R5	
2987	013610	104017			ERROR	17		
2988	013612	005077	165646		CLR	PKB	: START THE COUNT AT ZERO	
2989	013616	012714	000131		MOV	#SEARCH, (R4)	: START A SEARCH	
2990	013622	012777	000131	165632	MOV	#131, PKCS	: START THE CLOCK	
2991	013630	000001			WAIT		: WAIT ON INTERRUPT	
2992	013632	042777	000101	165622	BIC	#101, PKCS	: STOP THE CLOCK	
2993	013640	032764	040000	000012	BIT	#BIT14, RMD5(R4)	: IS "ERR=1"?	
2994	013646	001415			BEQ	3\$: NO--BRANCH	
2995	013650	104412			SAVREG		: SAVE RO-R5	
2996	013652	012702	004266		MOV	DTADPB, R2	: DPB POINTER	
2997	013656	004737	044134		JSR	PC, SVRH70	: SAVE ALL THE RH70/RMO3 REGISTERS	
2998	013662	012764	000040	000010	MOV	#BIT05, RMCS2(R4)	: MASSBUS CLEAR	
2999	013670	013764	004266	000010	MOV	DTADPB, RMCS2(R4)	: SELECT DRIVE	
3000	013676	104413			RESREG		: RESTORE RO-R5	
3001	013700	104017			ERROR	17	: DISK ERROR OCCURRED	
3002	013702	004737	031342		JSR	PC, COUNT	: UPDATE THE COUNT	
3003	013706	005301			DEC	R1	: DONE?	
3004	013710	003304			BGT	1\$: NO--BRANCH	
3005	013712	000424			BR	8\$: YES--GO TO THE EXIT	
3006	013714	042777	000101	165540	BIC	#101, PKCS	: STOP THE CLOCK	
3007	013722	005037	177776		CLR	PS	: DROP THE PRIORITY	
3008	013726	012600			MOV	(SP)+, RO	: PC OF WAIT+2	
3009	013730	005726			TST	(SP)+	: POP THE PS FROM THE STACK	
3010	013732	104412			SAVREG		: SAVE RO-R5	
3011	013734	012702	004266		MOV	DTADPB, R2	: DPB POINTER	
3012	013740	004737	044134		JSR	PC, SVRH70	: SAVE ALL THE RH70/RMO3 REGISTERS	
3013	013744	012764	000040	000010	MOV	#BIT05, RMCS2(R4)	: MASSBUS CLEAR	
3014	013752	013764	004266	000010	MOV	DTADPB, RMCS2(R4)	: SELECT DRIVE	
3015	013760	104413			RESREG		: RESTORE RO-R5	
3016	013762	104020			ERROR	20	: CLOCK OVERFLOWED	
3017	013764							
3018	013764	012764	000040	000010	MOV	#BIT05, RMCS2(R4)	: MASSBUS INIT.	
3019	013772	013764	004266	000010	MOV	DTADPB, RMCS2(R4)	: SELECT DRIVE	
3020	014000	004737	026116		JSR	PC, ST.CLK	: INITIALIZE THE CLOCK	
3021	014004	012777	041566	022726	MOV	ISR, RMVEC	: RESTORE RH70/RMO3 INT. VECTOR	
3022	014012	032737	000100	001300	BIT	#SW06, C.SWR	: 60 HZ?	
3023					BNE	9\$: NO -- BRANCH	
3024					REPORT	T7A	: REPORT THE TIMES	
3025					BR	EXIT12	: GO TO EXIT	
3026								
3027	014020	001004			REPORT	T7B		
3028	014022	004037	031474		BNE	EXIT.A	: NO-BRANCH	
3029	014026	003056			JSR	RO, TYPTIM	: TYPE THE TIMING	
3030	014030	000403			T7A		: TABLE ADDRESS	
3031	014032	004037	031474		BR	EXIT12	: EXIT	
3032	014036	003066			EXIT.A:	JSR	RO, TYPTIM	: TYPE THE TIMING
3033	014040	000004			TP50:	T7B	: TABLE ADDRESS	
3034					EXIT12:	SCOPE	: LOOP ?	
3035								
3036								
3037								
3038								
3039								

: *TEST 13 ONE CYLINDER SEEK TIMING TEST
: * THIS TEST WILL COMMAND FORWARD SEEK CYCLES TO ADVANCE THE

```

3040 ;* CYLINDER BY ONE UNTIL THE INCREMENT IS GREATER THAN THE
3041 ;* CYLINDER 'LC', THEN REVERSE SEEK TO CYLINDER 'FC'. THE
3042 ;* TIME TO PERFORM EACH SEEK IS CHECKED TO ENSURE IT DOES NOT
3043 ;* EXCEED THE MAXIMUM TIME PERMITTED FOR A ONE CYLINDER SEEK.
3044 ;* THE TIME MUST BE LESS THAN 10MS.
3045
3046 ;*****
3047 †TST13:
3048 014042 000240 NOP
3049 014044 033737 001534 001314 BIT 2#BITS+(13*2),TSTNMS ;DO THIS TEST?
3050 014052 001002 BNE 64$ ;YES--BRANCH
3051 014054 000137 014552 JMP TST14 ;NO--GO TO THE NEXT TEST
3052 014060 012737 000013 001102 64$: MOV #13,2#STSTNM ;SET UP TEST NUMBER AND
3053 ;CLEAR THE ERROR FLAG (SERFLG)
3054 014066 004737 026632 JSR PC,LODRM ;LOAD THE PARAMETERS FOR THE TEST
3055 014072 012737 014042 001110 MOV #TST13,2#SLPERR ;SETUP THE ERROR LOOP ADDRESS
3056 014100 013777 001102 165034 MOV $TSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
3057 014106 013737 001570 001204 MOV 2#RPT,$TIMES ;GET THE ITERATION COUNT
3058 014114 12737 000031 001115 MOVB #25,$SERMAX ;MAX ERRORS ALLOWED FOR TEST
3059 014122 012737 000013 001224 MOV #13,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
3060
3061 ;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
3062 014130 032777 010000 165002 BIT #SW12,2#SWR ;CHECK FOR SWITCH 12 SET
3063 014136 001413 BEQ 20$ ;IF = 0, THEN DON'T TYPE TEST NUMBER
3064 014140 013700 001102 MOV $TSTNM,R0 ;ELSE, TYPE THE NUMBER
3065 014144 042700 177000 BIC #177000,R0 ;MASK OUT THE HIGH ORDER BYTE
3066 014150 104401 001215 TYPE $SCLF
3067 014154 010046 MOV R0,-(SP) ;;SAVE R0 FOR TYPEOUT
3068 ;;TEST NUMBER
3069 014156 104403 TYPOS ;;GO TYPE--OCTAL ASCII
3070 014160 002 .BYTE 2 ;;TYPE 2 DIGIT(S)
3071 014161 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
3072 014162 104401 001215 TYPE ;,SCLF
3073 014166 20$:
3074
3075 014166 005737 001324 TST 2#CLKSTA ;KW11-P CLOCK?
3076 014172 003002 BGT 1$ ;YES--START TEST
3077 014174 000137 014552 JMP TST14 ;NO--GO TO NEXT TEST
3078 014200 012737 014200 001106 1$: MOV #1$,2#LPADR ;SETUP THE LOOP ADDRESS
3079 014206 004037 031112 JSR R0,2#SRCH00 ;DO A MASSBUS INIT. AND RECAL
3080 014212 000402 BR 2$ ;NO ERROR RETURN
3081 014214 000137 014550 JMP EXIT13 ;ERROR RETURN--SCOPE LOOP CALL
3082 014220 012703 003116 2$: MOV #T10,R3 ;PARAMETER POINTER
3083 014224 012737 014550 001206 MOV #EXIT13,$ESCAPE ;ESCAPE TO EXIT13 ON ERROR
3084 014232 012706 001100 TEST13: MOV #STACK,$P ;SETUP STACK
3085 014236 013737 001572 004300 MOV FC,2#DTADPB+12 ;START WITH BEGINNING CYLINDER
3086 014244 005237 004300 INC DTADPB+12 ;INCREMENT THE BEGINNING CYLINDER
3087 014250 005005 CLR R5 ;SET THE UP/DOWN SWITCH TO UP
3088 014252 004737 031276 JSR PC,2#STRTMR ;INITIALIZE THE TIMERS
3089 014256 012777 014444 165172 MOV #7$,2#PKV ;SETUP INCASE OF OVERFLOW
3090 014264 012777 031274 022446 MOV #DORTI,2#RMEVC ;SET RMO3 VECTOR
3091 014272 005077 165166 1$: CLR 2#PKB ;START THE COUNTER AT ZERO
3092 014276 013764 004300 000034 MOV 2#DTADPB+12,RMDC(R4) ;LOAD DESIRED CYLINDER
3093 014304 012714 000105 MOV #SEEK,(R4) ;START A SEEK
3094 014310 012777 000131 165144 MOV #131,2#PKCS ;START THE CLOCK
3095 014316 000001 WAIT ;WAIT ON INTERRUPT

```


M07

MD-11-DZRMF-B RMO3/RMO2 EXTENDED DRIVE TEST
CZRMFB.P11 21-NOV-77 14:10 T13

MACY11 30(1046) 22-NOV-77 08:04 PAGE 60
ONE CYLINDER SEEK TIMING TEST

SEG 0090

3096	014320	042777	000101	165134		BIC	#101,APKCS	:STOP THE CLOCK
3097	014326	032764	040000	000012		BIT	#BIT14,RMDS(R4)	:ANY DISK ERRORS?
3098	014334	001415				BEG	25	:NO--BRANCH
3099	014336	104412				SAVREG		:SAVE R0-R5
3100	014340	012702	004266			MOV	#DTADPB,R2	:DPB POINTER
3101	014344	004737	044134			JSR	PC,#SVRH70	:SAVE ALL THE RH70/RMO3 REGISTERS
3102	014350	012764	000040	000010		MOV	#BIT05,RMCS2(R4)	:MASSBUS CLEAR
3103	014356	013764	004266	000010		MOV	#DTADPB,RMCS2(R4)	:SELECT DRIVE
3104	014364	104413				RESREG		:RESTORE R0-R5
3105	014366	104017				ERROR	17	:REPORT THE ERROR
3106	014370	004737	031342		25:	JSR	PC,#COUNT	:COUNT THIS SEEKS TIME
3107	014374	004737	030570			JSR	PC,#TWOMS	:STALL FOR 2 MILLISECONDS
3108	014400	005705				TST	R5	:UP OR DOWN?
3109	014402	001011				BNE	45	:DOWN--BRANCH
3110	014404	005237	004300		35:	INC	#DTADPB+12	:MOVE TO NEXT CYLINDER
3111	014410	023737	004300	001574		CMP	#DTADPB+12,LC	:OUT OF CYLINDERS?
3112	014416	002725				BLT	15	:NO--GO DO THE NEXT SEEK
3113	014420	012705	177777			MOV	#-1,R5	:SET UP/DOWN SWITCH TO DOWN
3114	014424	000722				BR	15	:GO DO THE NEXT SEEK
3115	014426	005337	004300		45:	DEC	#DTADPB+12	:MOVE TO NEXT CYLINDER
3116	014432	023727	004300	000000		CMP	#DTADPB+12,#0	:OUT OF CYLINDERS?
3117	014440	003314				BGT	15	:NO--GO DO THE NEXT SEEK
3118	014442	000424				BR	85	:GO TO THE EXIT
3119	014444	042777	000101	165010	75:	BIC	#101,APKCS	:STOP THE CLOCK
3120	014452	005037	177776			CLR	#PS	:DROP THE PRIORITY
3121	014456	012600				MOV	(SP)+,R0	:PC OF WAIT+2
3122	014460	005726				TST	(SP)+	:POP THE PS FROM THE STACK
3123	014462	104412				SAVREG		:SAVE R0-R5
3124	014464	012702	004266			MOV	#DTADPB,R2	:DPB POINTER
3125	014470	004737	044134			JSR	PC,#SVRH70	:SAVE ALL THE RH70/RMO3 REGISTERS
3126	014474	012764	000040	000010		MOV	#BIT05,RMCS2(R4)	:MASSBUS CLEAR
3127	014502	013764	004266	000010		MOV	#DTADPB,RMCS2(R4)	:SELECT DRIVE
3128	014510	104413				RESREG		:RESTORE R0-R5
3129	014512	104020				ERROR	20	:REPORT CLOCK OVERFLOW
3130	014514				85:			
3131	014514	012764	000040	000010		MOV	#BIT05,RMCS2(R4)	:MASSBUS INIT.
3132	014522	013764	004266	000010		MOV	#DTADPB,RMCS2(R4)	:SELECT DRIVE
3133	014530	004737	026116			JSR	PC,#ST.CLK	:INITIALIZE THE CLOCK
3134	014534	012777	041566	022176		MOV	#ISR,ARMVEC	:RESTORE RH70/RMO3 INT. VECTOR
3135	014542	004037	031474			JSR	R0,#TYPTIM	:GO TYPE THE TIMES
3136	014546	003116				TIO		:POINTER
3137	014550	000004				EXIT13:	SCOPE	:LOOP ?

: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.

3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151


```

3152
3153 014552
3154 014552 000240
3155 014554 033737 001536 001314
3156 014562 001002
3157 014564 000137 015334
3158 014570 012737 000014 001102 64$:
3159
3160 014576 004737 026632
3161 014602 012737 014552 001110
3162 014610 013777 001102 164324
3163 014616 013737 001570 001204
3164 014624 112737 000031 001115
3165 014632 012737 000014 001224
3166
3167
3168 014640 032777 010000 164272
3169 014646 001413
3170 014650 013700 001102
3171 014654 042700 177000
3172 014660 104401 001215
3173 014664 010046
3174
3175 014666 104403
3176 014670 002
3177 014671 000
3178 014672 104401 001215
3179 014676
3180
3181 014676 005737 001324
3182 014702 003002
3183 014704 000137 015334
3184 014710 012737 014710 001106 1$:
3185 014716 004037 031112
3186 014722 000402
3187 014724 000137 015332
3188 014730 012703 003126
3189 014734 012737 015332 001206 2$:
3190 014742 012706 001100 TEST14:
3191 014746 012701 000200
3192 014752 004737 031276
3193 014756 012777 015226 164472
3194 014764 012777 031274 021746
3195 014772 005077 164466 1$:
3196 014776 013764 001574 000034
3197 015004 012764 000105 000000
3198 015012 012777 000131 164442
3199 015020 000001
3200 015022 042777 000101 164432
3201 015030 032764 040000 000012
3202 015036 001415
3203 015040 104412
3204 015042 012702 004266
3205 015046 004737 044134
3206 015052 012764 000040 000010
3207 015060 013764 004266 000010

```

```

*****
†ST14:
NOP
BIT @#BITS+(14*2),TSTNMS ;DO THIS TEST?
BNE 64$ ;YES--BRANCH
JMP TST15 ;NO--GO TO THE NEXT TEST
MOV #14,@#STSTNM ;SET UP TEST NUMBER AND
;CLEAR THE ERROR FLAG (SERFLG)
JSR PC,LODPRM ;LOAD THE PARAMETERS FOR THE TEST
MOV #TST14,@#SLPERR ;SETUP THE ERROR LOOP ADDRESS
MOV $STSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
MOV @RPT,$TIMES ;GET THE ITERATION COUNT
MOVB #25,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
MOV #14,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

;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

TST @#CLKSTA ;KW11-P CLOCK?
BGT 1$ ;YES--START TEST
JMP TST15 ;NO--GO TO NEXT TEST
MOV #1$,@#SLPADR ;SET THE LOOP ADDRESS
JSR R0,@#SRCH00 ;DO A MASSBUS INIT & RECAL
BR 2$ ;RETURN HERE IF NO ERROR
JMP EXIT14 ;RETURN HERE ON ERROR
MOV #T11,R3 ;PARAMETER POINTER
MOV #EXIT14,$ESCAPE ;;ESCAPE TO EXIT14 ON ERROR
MOV #STACK,$SP ;SETUP STACK
MOV #128,R1 ;REPEAT "0-136-0" 128 TIMES
JSR PC,@#STATMR ;INIT. THE COUNTERS
MOV #7$,@#PKV ;SET UP VECTOR IN CASE OF OVERFLOW
MOV #DORTI,@#RMVEC ;SETUP RMO3 VECTOR
CLR @#PKB ;START COUNT AT ZERO
MOV LC,RMDC(R4) ;'MIDDLE' CYLINDER
MOV #SEEK,RMCS1(R4) ;START A SEEK
MOV #131,@#PKCS ;START THE CLOCK
WAIT ;WAIT ON INTERRUPT
BIC #101,@#PKCS ;STOP CLOCK
BIT #BIT14,RMDS(R4) ;ERR=1?
BEQ 2$ ;NO--BRANCH
SAVREG ;SAVE R0-R5
MOV #DTADPB,R2 ;DPB POINTER
JSR PC,@#SVRH70 ;SAVE ALL THE RH70/RMO3 REGISTERS
MOV #BIT05,RMCS2(R4) ;MASSBUS CLEAR
MOV @#DTADPB,RMCS2(R4) ;SELECT DRIVE

```



```

3208 015066 104413
3209 015070 104017
3210 015072 005005
3211 015074 004737 031342
3212 015100 004737 030670
3213 015104 005077 164354
3214 015110 013764 001572 000034
3215 015116 012764 000105 000000
3216 015124 012777 000131 164330
3217 015132 000001
3218 015134 042777 000101 164320
3219 015142 032764 040000 000012
3220 015150 001415
3221 015152 104412
3222 015154 012702 004266
3223 015160 004737 044134
3224 015164 012764 000040 000010
3225 015172 013764 004266 000010
3226 015200 104413
3227 015202 104017
3228 015204 012705 177777
3229 015210 004737 031342
3230 015214 004737 030670
3231 015220 005301
3232 015222 003263
3233 015224 000424
3234 015226 042777 000101 164226
3235 015234 005037 177776
3236 015240 012600
3237 015242 005726
3238 015244 104412
3239 015246 012702 004266
3240 015252 004737 044134
3241 015256 012764 000040 000010
3242 015264 013764 004266 000010
3243 015272 104413
3244 015274 104020
3245 015276
3246 015276 012764 000040 000010
3247 015304 013764 004266 000010
3248 015312 004737 026116
3249 015316 012777 041566 021414
3250 015324 004037 031474
3251 015330 003126
3252 015332 000004

```

```

RESREG ;RESTORE R0-R5
ERROR 17
2S: CLR R5 ;SET UP/DOWN SWITCH TO UP
JSR PC,@#COUNT ;UPDATE THE COUNT
JSR PC,@#TWOMS ;STALL FOR 2 MILLISECOND
CLR @PKB ;START THE COUNT AT ZERO
MOV FC,RMDC(R4) ;BEGINNING CYLINDER
MOV #SEEK,RMCS1(R4) ;START A SEEK
MOV #131,@PKCS ;START THE CLOCK
WAIT ;WAIT ON INTERRUPT
BIC #101,@PKCS ;STOP THE CLOCK
BIT #BIT14,RMDS(R4) ;ERR=1?
BEQ 3S ;NO--BRANCH
SAVREG ;SAVE R0-R5
MOV #DTADPB,R2 ;DPB POINTER
JSR PC,@#SVRH70 ;SAVE ALL THE RH70/RM03 REGISTERS
MOV #BIT05,RMCS2(R4) ;MASSBUS CLEAR
MOV @#DTADPB,RMCS2(R4) ;SELECT DRIVE
RESREG ;RESTORE R0-R5
ERROR 17
3S: MOV #-1,R5 ;SET UP/DOWN SWITCH TO DOWN
JSR PC,@#COUNT ;UPDATE THE COUNT
JSR PC,@#TWOMS ;STALL FOR 2 MILLISECOND
DEC R1 ;DONE?
BGT 1S ;NO--BRANCH
BR 8S ;YES--EXIT
BIC #101,@PKCS ;STOP THE CLOCK
CLR @#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,@#SVRH70 ;SAVE ALL THE RH70/RM03 REGISTERS
MOV #BIT05,RMCS2(R4) ;MASSBUS CLEAR
MOV @#DTADPB,RMCS2(R4) ;SELECT DRIVE
RESREG ;RESTORE R0-R5
ERROR 20 ;CLOCK OVERFLOWED
8S: MOV #BIT05,RMCS2(R4) ;MASSBUS INIT.
MOV @#DTADPB,RMCS2(R4) ;SELECT DRIVE
JSR PC,@#ST.CLK ;INITIALIZE THE CLOCK
MOV #ISR,@RMVEC ;RESTORE RH70/RM03 INT. VECTOR
JSR R0,@#TYPTIM ;GO TYPE THE TIMES
T11 ;POINTER
EXIT14: SCOPE ;LOOP ?

```

```

3253 *****
3254 ;*TEST 15 MAXIMUM SEEK TIMING TEST
3255 *****
3256 ;*
3257 ;* THIS TEST WILL COMMAND A FORWARD SEEK FROM CYLINDER 0 TO
3258 ;* CYLINDER 'LC' THEN A REVERSE SEEK FROM CYLINDER 'LC' TO
3259 ;* CYLINDER 0. BOTH SEEKS ARE TIMED AND CHECKED TO ENSURE
3260 ;* THEY ARE WITHIN THE TOLERANCE ALLOWED FOR THE MAXIMUM SEEK
3261 ;* TIME. THIS SEQUENCE IS REPEATED 128 TIMES (FOR
3262 ;* A TOTAL OF 256 SEEKS). THE MAXIMUM SEEK TIME MUST BE LESS THAN
3263 ;* 54 MS. 'LC' DEFAULTS TO 410 (10) FOR RM03/5'S AND TO 814 (10)

```



```

3264 ;* FOR RMO3'S.
3265
3266 :*****
3267 TST15:
3268 015334 000240 NOP
3269 015336 033737 001540 001314 BIT 2#BITS+<15*2>,TSTNMS ;DO THIS TEST?
3270 015344 001002 BNE 64$ ;YES--BRANCH
3271 015346 000137 016116 JMP TST16 ;NO--GO TO THE NEXT TEST
3272 015352 012737 000015 001102 64$: MOV #15,2#STSTNM ;SET UP TEST NUMBER AND
3273 ;CLEAR THE ERROR FLAG (SERFLG)
3274 015360 004737 026632 JSR PC,LODPRM ;LOAD THE PARAMETERS FOR THE TEST
3275 015364 012737 015334 001110 MOV #TST15,2#SLPERR ;SETUP THE ERROR LOOP ADDRESS
3276 015372 013777 001102 163542 MOV #STSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
3277 015400 013737 001570 001204 MOV #2#APT,2#TIMES ;GET THE ITERATION COUNT
3278 015406 112737 000031 001115 MOV#B #25,2#ERMAX ;MAX ERRORS ALLOWED FOR TEST
3279 015414 012737 000015 001224 MOV #15,2#TESTN ;SET TEST NUMBER IN APT MAIL BOX
3280
3281 :THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
3282 015422 032777 010000 163510 BIT #SW12,2#SWR ;CHECK FOR SWITCH 12 SET
3283 015430 001413 BEQ 20$ ;IF = 0, THEN DON'T TYPE TEST NUMBER
3284 015432 013700 001102 MOV #STSTNM,R0 ;ELSE, TYPE THE NUMBER
3285 015436 042700 177000 BIC #177000,R0 ;MASK' OUT THE HIGH ORDER BYTE
3286 015442 104401 001215 TYPE $,SCLF
3287 015446 010046 MOV R0,-(SP) ;SAVE R0 FOR TYPEOUT
3288 ;TEST NUMBER
3289 015450 104403 TYPOS ;GO TYPE--OCTAL ASCII
3290 015452 002 .BYTE 2 ;TYPE 2 DIGIT(S)
3291 015453 000 .BYTE 0 ;SUPPRESS LEADING ZEROS
3292 015454 104401 001215 TYPE $,SCLF
3293
3294 20$:
3295 015460 005737 001324 TST 2#CLKSTA ;KW11-P CLOCK
3296 015464 033002 BGT 1$ ;YES--START TEST
3297 015466 000137 016116 JMP TST16 ;NO--GO TO NEXT TEST
3298 015472 012737 015472 001106 1$: MOV #15,2#SLPADR ;SETUP THE LOOP ADDRESS
3299 015500 004037 031112 JSR R0,2#SRCH00 ;DO A MASSBUS INIT & RECAL
3300 015504 000402 BR 2$ ;RETURN HERE IF NO ERROR
3301 015506 000137 016114 JMP EXIT15 ;RETURN HERE ON ERROR
3302 015512 012703 003136 2$: MOV #T12,R3 ;PARAMETER POINTER
3303 015516 012737 016114 001206 MOV #EXIT15,2#ESCAPE ;ESCAPE TO EXIT15 ON ERROR
3304 015524 012706 001100 TEST15: MOV #STACK,SP ;SETUP STACK
3305 015530 012701 000200 MOV #128,R1 ;REPEAT "0-'LC'-0" 128 TIMES
3306 015534 004737 031276 JSR PC,2#STATMR ;INIT. THE TIMERS
3307 015540 012777 016010 163710 MOV #7$,2#PKV ;SETUP VECTOR IN CASE OF OVERFLOW
3308 015546 012777 031274 021164 MOV #DORTI,2#RMVEC ;SETUP RMO3 VECTOR
3309 015554 005077 163704 1$: CLR #PKB ;START COUNTING FROM ZERO
3310 015560 013764 001574 000034 MOV LC,RMDC(R4) ;MAXIMUM CYLINDER
3311 015566 012764 000105 000000 MOV #SEEK,RMCS1(R4) ;START A SEEK
3312 015574 012777 000131 163660 MOV #131,2#PKCS ;START THE CLOCK
3313 015602 000001 WAIT ;WAIT ON INTERRUPT
3314 015604 042777 000101 163650 BIC #101,2#PKCS ;STOP THE CLOCK
3315 015612 032764 040000 000012 BIT #BIT14,RMDS(R4) ;ERR=1?
3316 015620 001415 BEQ 2$ ;NO--BRANCH
3317 015622 104412 SAVREG ;SAVE R0-R5
3318 015624 012702 004266 MOV #DTADPB,R2 ;DPB POINTER
3319 015630 004737 044134 JSR PC,2#SVRH70 ;SAVE ALL THE RH70/RMO3 REGISTERS

```


3320	015634	012764	000040	000010	MOV	#BIT05, RMCS2(R4)	: MASSBUS CLEAR
3321	015642	013764	004266	000010	MOV	#DTADPB, RMCS2(R4)	: SELECT DRIVE
3322	015650	104413			RESREG		: RESTORE RO-R5
3323	015652	104017			ERROR	17	
3324	015654	005005			2S: CLR	R5	: SET THE UP/DOWN SWITCH TO UP
3325	015656	004737	031342		JSR	PC, #COUNT	: UP THE COUNT
3326	015662	004737	030670		JSR	PC, #TWOMS	: STALL FOR 2 MILLISECONDS
3327	015666	005077	163572		CLR	#PKB	: START COUNT AT ZERO
3328	015672	013764	001572	000034	MOV	FC, RMDC(R4)	: BEGINNING CYLINDER
3329	015700	012764	000105	000000	MOV	#SEEK, RMCS1(R4)	: START A SEEK
3330	015706	012777	000131	163546	MOV	#131, #PKCS	: START THE CLOCK
3331	015714	000001			WAIT		: WAIT ON INTERRUPT
3332	015716	042777	000101	163536	BIC	#101, #PKCS	: STOP THE CLOCK
3333	015724	032764	040000	000012	BIT	#BIT14, RMD5(R4)	: "ERR"=1?
3334	015732	001415			BEQ	3S	: NO--BRANCH
3335	015734	104412			SAVREG		: SAVE RO-R5
3336	015736	012702	004266		MOV	#DTADPB, R2	: DPB POINTER
3337	015742	004737	044134		JSR	PC, #SVRH70	: SAVE ALL THE RH70/RMO3 REGISTERS
3338	015746	012764	000040	000010	MOV	#BIT05, RMCS2(R4)	: MASSBUS CLEAR
3339	015754	013764	004266	000010	MOV	#DTADPB, RMCS2(R4)	: SELECT DRIVE
3340	015762	104413			RESREG		: RESTORE RO-R5
3341	015764	104017			ERROR	17	: REPORT THE ERROR
3342	015766	012705	177777		3S: MOV	#-1, R5	: SET THE UP/DOWN SWITCH TO DOWN
3343	015772	004737	031342		JSR	PC, #COUNT	: UPDATE THE COUNT
3344	015776	004737	030670		JSR	PC, #TWOMS	: STALL FOR 2 MILLISECONDS
3345	016002	005301			DEC	R1	: DONE?
3346	016004	003263			BGT	1S	: NO--BRANCH
3347	016006	000424			BR	8S	: YES--EXIT
3348	016010	042777	000101	163444	7S: BIC	#101, #PKCS	: STOP THE CLOCK
3349	016016	005037	177776		CLR	#PS	: DROP THE PRIORITY
3350	016022	012600			MOV	(SP)+, RO	: PC OF WAIT+2
3351	016024	005726			TST	(SP)+	: POP THE PS FROM THE STACK
3352	016026	104412			SAVREG		: SAVE RO-R5
3353	016030	012702	004266		MOV	#DTADPB, R2	: DPB POINTER
3354	016034	004737	044134		JSR	PC, #SVRH70	: SAVE ALL THE RH70/RMO3 REGISTERS
3355	016040	012764	000040	000010	MOV	#BIT05, RMCS2(R4)	: MASSBUS CLEAR
3356	016046	013764	004266	000010	MOV	#DTADPB, RMCS2(R4)	: SELECT DRIVE
3357	016054	104413			RESREG		: RESTORE RO-R5
3358	016056	104020			ERROR	20	: CLOCK OVERFLOWED
3359	016060				8S:		
3360	016060	012764	000040	000010	MOV	#BIT05, RMCS2(R4)	: MASSBUS INIT.
3361	016066	013764	004266	000010	MOV	#DTADPB, RMCS2(R4)	: SELECT DRIVE
3362	016074	004737	026116		JSR	PC, #ST.CLK	: INITIALIZE THE CLOCK
3363	016100	012777	041566	020632	MOV	#ISR, #RMVEC	: RESTORE RH70/RMO3 INT. VECTOR
3364	016106	004037	031474		JSR	RO, #TYPTIM	: GO TYPE THE TIMES
3365	016112	003136			T12		: POINTER
3366	016114	000004			EXIT15: SCOPE		: LOOP ?

F08

```

3423 016242 205:
3424
3425 016242 012737 016556 001332
3426 016250 012737 016256 001106
3427 016256 012706 001100 TEST16:
3428 016262 004737 032126
3429 016266 013737 001572 004300
3430 016274 113737 001600 004277
3431 016302 105037 004276
3432 016306 013737 001432 004272
3433 016314 012737 051776 004274
3434 016322 012737 016322 001110
3435 016330 012706 001100
3436 016334 012737 000161 004270
3437 016342 004037 027612
3438 016346 012737 000151 004270
3439 016354 012737 016354 001110
3440 016362 012706 001100
3441 016366 004037 027612
3442 016372 012737 016372 001110
3443 016400 012706 001100
3444 016404 004037 032164
3445 016410 012737 000171 004270
3446 016416 004037 027612
3447 016422 004037 032232
3448 016426 012700 051776
3449 016432 005001
3450 016434 012737 016434 001110
3451 016442 012706 001100
3452 016446 012737 000161 004270 15:
3453 016454 012737 177400 004272
3454 016462 010037 004274
3455 016466 110137 004276
3456 016472 004037 027612
3457 016476 012737 016476 001110
3458 016504 012706 001100
3459 016510 012737 000151 004270
3460 016516 013737 001432 004272
3461 016524 012737 051776 004274
3462 016532 105037 004276
3463 016536 004037 027612
3464 016542 062700 001000
3465 016546 005201
3466 016550 023701 001712
3467 016554 103334
3468 016556 000004

```

```

MOV #EXIT16, @#BYPASS
MOV #TEST16, $LPADR ; SETUP THE LOOP ADDRESS
MOV #STACK, SP ; SET THE STACK POINTER
JSR PC, @#FILBUF ; FILL THE BUFFER WITH SECTOR ADDRESSES
MOV @#FC, @#DTADPB+12 ; CYLINDER
MOV @#FT, @#DTADPB+11 ; TRACK
CLRB @#DTADPB+10 ; SECTOR
MOV TRCKWC, @#DTADPB+4 ; WORD COUNT
MOV #BUFFER, @#DTADPB+6 ; BUFFER ADDRESS
MOV #, $LPERR ; SETUP THE ERROR LOOP ADDRESS
MOV #STACK, SP ; LOAD THE STACK POINTER
MOV #WRITE, @#DTADPB+2 ; COMMAND=WRITE DATA
JSR RO, @#DRVCAL ; START A DATA TRANSFER
MOV #WRCKD, @#DTADPB+2 ; COMMAND=WRITE CHECK DATA
MOV #, $LPERR ; SETUP THE ERROR LOOP ADDRESS
MOV #STACK, SP ; LOAD THE STACK POINTER
JSR RO, @#DRVCAL ; START A DATA TRANSFER
MOV #, $LPERR ; SETUP THE ERROR LOOP ADDRESS
MOV #STACK, SP ; LOAD THE STACK POINTER
JSR RO, @#CLRBUF ; CLEAR BUFFER
MOV #READ, @#DTADPB+2 ; COMMAND = READ
JSR RO, @#DRVCAL ; START A DATA TRANSFER
JSR RO, @#CKSCTR ; CHECK THE SECTOR DATA READ
MOV #BUFFER, RO ; BUFFER ADDRESS
CLR R1 ; FIRST SECTOR
MOV #, $LPERR ; SETUP THE ERROR LOOP ADDRESS
MOV #STACK, SP ; LOAD THE STACK POINTER
MOV #WRITE, @#DTADPB+2 ; COMMAND=WRITE DATA
MOV #SCTRWC, @#DTADPB+4 ; WORD COUNT
MOV RO, @#DTADPB+6 ; BUFFER ADDRESS
MOV R1, @#DTADPB+10 ; SECTOR
JSR RO, @#DRVCAL ; START A DATA TRANSFER
MOV #, $LPERR ; SETUP THE ERROR LOOP ADDRESS
MOV #STACK, SP ; LOAD THE STACK POINTER
MOV #WRCKD, @#DTADPB+2 ; COMMAND=WRITE CHECK DATA
MOV TRCKWC, @#DTADPB+4 ; WORD COUNT
MOV #BUFFER, @#DTADPB+6 ; BUFFER ADDRESS
CLRB @#DTADPB+10 ; SECTOR
JSR RO, @#DRVCAL ; START A DATA TRANSFER
ADD #512., RO ; MOVE TO NEXT SECTOR
INC R1
CMP PRMLMT+22, R1 ; DONE?
BHS 15 ; NO--BRANCH
EXIT16: SCOPE ; LOOP ?

```

```

*****
; *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

```

```

3469
3470
3471
3472
3473
3474
3475
3476
3477
3478

```

```

3479      ;*      REWRITTEN AND TRACK 2 THROUGH TRACK 18 IS WRITE CHECKED.
3480      ;*      THIS PROCEDURE IS CONTINUED UP THROUGH REWRITING TRACK 17
3481      ;*      AND WRITE CHECKING TRACK 18.
3482
3483      ;*****
3484      TST17:
3485      016560      000240      NOP
3486      016562      033737      001544      001314      BIT      @#BITS+<17*2>,TSTNMS ;DO THIS TEST?
3487      016570      001002      BNE      64$      ;YES--BRANCH
3488      016572      000137      017244      JMP      TST20      ;NO--GO TO THE NEXT TEST
3489      016576      012737      000017      001102      64$:      MOV      @17,@#STSTNM      ;SET UP TEST NUMBER AND
3490      ;CLEAR THE ERROR FLAG (SERFLG)
3491      016604      004737      026632      JSR      PC,LODPRM      ;LOAD THE PARAMETERS FOR THE TEST
3492      016610      012737      016720      001110      MOV      @TEST17,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
3493      016616      013777      001102      162316      MOV      $STSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
3494      016624      013737      001570      001204      MOV      @RPT,$TIMES      ;GET THE ITERATION COUNT
3495      016632      113737      001444      001115      MOVVB   ERR,CT,SERMAX      ;MAX ERRORS ALLOWED FOR TEST
3496      016640      012737      000017      001224      MOV      @17,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
3497
3498      ;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
3499      016646      032777      010000      162264      BIT      #SW12,@SWR      ;CHECK FOR SWITCH 12 SET
3500      016654      001413      BEQ      20$      ;IF = 0, THEN DON'T TYPE TEST NUMBER
3501      016656      013700      001102      MOV      $STSTNM,RO      ;ELSE, TYPE THE NUMBER
3502      016662      042700      177000      BIC      #177000,RO      ;MASK'OUT THE HIGH ORDER BYTE
3503      016666      104401      001215      TYPE    $CRLF
3504      016672      010046      MOV      RO,-(SP)      ;;SAVE RO FOR TYPEOUT
3505      ;TEST NUMBER
3506      016674      104403      TYPOS
3507      016676      002      .BYTE   2      ;GO TYPE--OCTAL ASCII
3508      016677      000      .BYTE   0      ;TYPE 2 DIGIT(S)
3509      016700      104401      001215      TYPE    ,CRLF      ;SUPPRESS LEADING ZEROS
3510      016704
3511      20$:
3512      016704      012737      017242      001332      MOV      #EXIT17,@#BYPASS
3513      016712      012737      016720      001106      MOV      @TEST17,$LPADR ;SETUP THE LOOP ADDRESS
3514      016720      012706      001100      TEST17: MOV      @STACK,SP      ;SET THE STACK POINTER
3515      016724      004737      032126      JSR      PC,@#FILBUF      ;FILL THE BUFFER WITH TRACK ADDRESS
3516      016730      012737      000161      004270      MOV      @WRITE,@#DTADPB+2 ;COMMAND=WRITE DATA
3517      016736      013737      001572      004300      MOV      @#FC,@#DTADPB+12 ;CYLINDER
3518      016744      113737      001606      004276      MOVVB   @#FS,@#DTADPB+10 ;SECTOR
3519      016752      012737      177400      004272      MOV      @SCTAWC,@#DTADPB+4 ;WORD COUNT
3520      016760      012737      051776      004274      MOV      @BUFFER,@#DTADPB+6 ;BUFFER ADDRESS
3521      016766      005000      CLR      RO      ;TRACK=0
3522      016770      012737      016770      001110      MOV      #,$SLPERR      ;SETUP THE ERROR LOOP ADDRESS
3523      016776      012706      001100      MOV      @STACK,SP      ;LOAD THE STACK POINTER
3524      017002      110037      004277      1$:      MOVVB   RO,@#DTADPB+11 ;TRACK ADDRESS
3525      017006      004037      027612      JSR      RO,@#DRVCAL      ;START A DATA TRANSFER
3526      017012      062737      001000      004274      ADD      @256.*2,@#DTADPB+6 ;UPDATE BUFFER ADDRESS
3527      017020      005200      INC      RO      ;UPDATE TRACK NUMBER
3528      017022      022700      000005      CMP      #5,RO      ;OUT OF TRACKS?
3529      017026      003365      BGT      1$      ;NO--BRANCH
3530      017030      012737      051776      004274      MOV      @BUFFER,@#DTADPB+6 ;BUFFER ADDRESS
3531      017036      005000      CLR      RO
3532      017040      012737      017040      001110      MOV      #,$SLPERR      ;SETUP THE ERROR LOOP ADDRESS
3533      017046      012706      001100      MOV      @STACK,SP      ;LOAD THE STACK POINTER
3534      017052      012737      000151      004270      MOV      @WRCKD,@#DTADPB+2 ;COMMAND=WRITE CHECK

```


3535	017060	110037	004277	2\$:
3536	017064	004037	027612	
3537	017070	062737	001000	004274
3538	017076	005200		
3539	017100	022700	000005	
3540	017104	003365		
3541	017106	005000		
3542	017110	110037	004277	3\$:
3543	017114	010001		
3544	017116	012737	051776	004274
3545	017124	005301		4\$:
3546	017126	002411		
3547	017130	062737	001000	004274
3548	017136	000772		
3549	017140	012737	017140	001110
3550	017146	012706	001100	
3551	017152	012737	000161	004270 5\$:
3552	017160	004037	027612	
3553	017164	062737	001000	004274 6\$:
3554	017172	105237	004277	
3555	017176	012737	017176	001110
3556	017204	012706	001100	
3557	017210	012737	000151	004270
3558	017216	004037	027612	
3559	017222	122737	000004	004277
3560	017230	003355		
3561	017232	005200		
3562	017234	022700	000004	
3563	017240	003323		
3564	017242	000004		

```
MOV B RO, @DTADPB+11 ; TRACK ADDRESS
JSR RO, @DRVCAL ; START A DATA TRANSFER
ADD #256.*2., @DTADPB+6 ; UPDATE BUFFER ADDRESS
INC RO ; UPDATE TRACK NUMBER
CMP #5, RO ; OUT OF TRACKS?
BGT 2$ ; NO--BRANCH
CLR RO ; FIRST TRACK ADDRESS
MOV B RO, @DTADPB+11 ; TRACK
MOV RO, R1 ; FORM BUFFER ADDRESS
MOV #BUFFER, @DTADPB+6 ; BUFFER ADDRESS
4$: DEC R1
BLT 5$
ADD #256.*2., @DTADPB+6
BR 4$
MOV #. SLPERR ; SETUP THE ERROR LOOP ADDRESS
MOV #STACK, SP ; LOAD THE STACK POINTER
5$: MOV #WRITE, @DTADPB+2 ; COMMAND=WRITE DATA
JSR RO, @DRVCAL ; START A DATA TRANSFER
6$: ADD #256.*2., @DTADPB+6 ; UPDATE BUFFER ADDRESS
INCB @DTADPB+11 ; MOVE TO NEXT TRACK
MOV #. SLPERR ; SETUP THE ERROR LOOP ADDRESS
MOV #STACK, SP ; LOAD THE STACK POINTER
MOV #WRCKD, @DTADPB+2 ; COMMAND=WRITE CHECK DATA
JSR RO, @DRVCAL ; START A DATA TRANSFER
CMPB #4, @DTADPB+11 ; OUT OF TRACKS?
BGT 6$ ; NO--BRANCH
RO ; NEXT TRACK TO WRITE
CMP #4, RO ; OUT OF TRACKS?
BGT 3$ ; NO--BRANCH
EXIT17: SCOPE
```

.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	065667	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

3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620

J08

```

3621 : * 133333 004000 173777 155555 177757 066667 177777 000000
3622 : * 165555 010000 167777 172666 177767 153333 177777 000000
3623 : * 133333 020000 157777 155555 177773 066667 177777 000000
3624 : * 165555 040000 137777 172666 177775 153333 177777 000000
3625 : * 133333 100000 077777 155555 177776 066667 177777 000000
3626 : *
3627 : *
3628 : *
3629 : *

```

```

3629 017244 : *****
3630 017244 000240 †ST20: NOP
3631 017246 033737 001506 001316 BIT BITS+<20*2-40>,TSTNMS+2 ;DO THIS TEST ?
3632 017254 001002 BNE 64$ ;YES--BRANCH
3633 017256 000137 020044 JMP TST21 ;NO--GO TO THE NEXT TEST
3634 017262 012737 000020 001102 64$: MOV #20,#STSTNM ;SET UP TEST NUMBER AND
3635 ;CLEAR THE ERROR FLAG (SERFLG)
3636 017270 004737 026632 JSR PC,LODPRM ;LOAD THE PARAMETERS FOR THE TEST
3637 017274 012737 017512 001110 MOV #TST20,#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
3638 017302 013777 001102 161632 MOV $STSTNM,#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
3639 017310 013737 001570 001204 MOV #RPT,$TIMES ;GET THE ITERATION COUNT
3640 017316 113737 001444 001115 MOV#B ERR.CT,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
3641 017324 012737 000020 001224 MOV #20,$TSTN ;SET TEST NUMBER IN APT MAIL BOX
3642
3643 ;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
3644 017332 032777 010000 161600 BIT #SW12,#SWR ;CHECK FOR SWITCH 12 SET
3645 017340 001413 BEQ 20$ ;IF = 0, THEN DON'T TYPE TEST NUMBER
3646 017342 013700 001102 MOV $STSTNM,R0 ;ELSE, TYPE THE NUMBER
3647 017346 042700 177000 BIC #177000,R0 ;MASK'OUT THE HIGH ORDER BYTE
3648 017352 104401 001215 TYPE $R0L ;SAVE R0 FOR TYPEOUT
3649 017356 010046 MOV R0,-(SP) ;TEST NUMBER
3650 ;GO TYPE--OCTAL ASCII
3651 017360 104403 TYPOS ;TYPE 2 DIGIT(S)
3652 017362 002 .BYTE 2 ;SUPPRESS LEADING ZEROS
3653 017363 000 .BYTE 0
3654 017364 104401 001215 TYPE , $R0L
3655 017370 20$:
3656
3657 ;SET UP THE TRACK WORD COUNT FOR DATA TRANSFER
3658
3659
3660 017370 012737 017370 001106 MOV #,$SLPADR ;SETUP THE LOOP ADDRESS
3661 017376 005000 CLR R0 ;CLEAR SWITCH
3662 017400 005004 CLR R4 ;FORM WORD COUNT IN R4
3663 017402 013701 001610 MOV #LS,R1 ;LOAD LAST SECTOR
3664 017406 163701 001606 SUB #FS,R1 ;COMPARE WITH FIRST SECTOR
3665 017412 002004 BGE 1$ ;SKIP NEXT IF FS < OR = LS
3666
3667 017414 063701 001712 ADD PRMLMT+22,R1 ;ADD MAXIMUM SECTOR ADDRESS TO
3668 017420 005201 INC R1 ;MAKE THE DIFFERENCE POSITIVE
3669 017422 005100 COM R0 ;SET SWITCH FOR FS > LS
3670
3671 017424 062704 000400 1$: ADD #256.,R4 ;WORDS/SECTOR
3672 017430 005301 DEC R1 ;LS - FS SECTORS MINUS ONE
3673 017432 002374 BGE 1$ ;INCR. WORD COUNT IF NO. SECTORS STILL +
3674 017434 005404 NEG R4 ;FORM NEGATIVE WORD COUNT
3675 017436 010405 MOV R4,R5 ;COPY NORMAL WORD COUNT INTO SMALL WC
3676 017440 005700 TST R0 ;FS > LS SWITCH SET?

```

```

3677 017442 001412          BEQ      3$          ;NO, SKIP NEXT
3678
3679 017444 005005          CLR      R5          ;FORM WORD COUNT FOR FS > LS
3680 017446 013701 001712    MOV      PRMLMT+22,R1 ;MAX SECTOR ADDRESS
3681 017452 163701 001606    SUB      @#FS,R1     ;SUBTRACT THE FIRST SECTOR
3682 017456 062705 000400    2$:     ADD      #256.,R5 ;WORDS/SECTOR
3683 017462 005301          DEC      R1          ;NO SECTORS MINUS ONE
3684 017464 002374          BGE     2$          ;INCR. WORD COUNT IF NO. SECTORS STILL +
3685 017466 005405          NEG      R5          ;FORM NEGATIVE WORD COUNT FOR THIS CASE
3686
3687                          ;SET UP FOR DATA TRANSFERS AND PATTERN VARIATIONS
3688
3689 017470 113737 001606 004276 3$:     MOV      @#FS,@#DTADPB+10 ;SECTOR
3690 017476 012737 051776 004274    MOV      #BUFFER,@#DTADPB+6 ;DATA BUFFER
3691 017504 012737 020042 001332    MOV      #EXIT20,@#BYPASS
3692 017512 012706 001100    TEST20: MOV      #STACK,SP   ;LOAD THE STACK POINTER
3693 017516 005037 001414    CLR      @#WCEFLG    ;CLEAR THE WRITE CHECK ERROR FLAG
3694 017522 013701 001572    MOV      @#FC,R1     ;PICKUP FIRST CYLINDER
3695 017526 000407          BR       2$          ;SKIP PATTERN SET-UP FIRST TIME THRU
3696
3697 017530 005720          1$:     TST      (R0)+       ;MOVE TO NEXT DATA PATTERN
3698 017532 022700 000040    CMP      #16.*2.,R0 ;OUT OF PATTERNS?
3699 017536 003004          BGT     3$          ;NO, STAY ON THIS CYLINDER UNTIL DONE
3700 017540 004037 032052    JSR      R0,@#INCCYL ;MOVE TO NEXT CYLINDER
3701 017544 000536          BR       EXIT20     ;OUT OF CYLINDERS ----->
3702
3703                          ;DO NEXT CYLINDER
3704
3705 017546 005000          2$:     CLR      R0          ;START WITH PATTERN 0
3706 017550 036037 001506 001612 3$:     BIT      BITS(R0),@#PAT ;THIS PATTERN SELECTED?
3707 017556 001764          BEQ     1$          ;NO, GO BACK AND GET ONE THAT WAS
3708 017560 013702 001600    MOV      @#FT,R2     ;FIRST TRACK
3709 017564 010137 004300    MOV      R1,@#DTADPB+12 ;LOAD THE CYLINDER
3710 017570 110237 004277    4$:     MOV      R2,@#DTADPB+11 ;LOAD THE TRACK
3711
3712 017574 020127 001466    CMP      R1,#822.    ;CHECK FOR LAST DISK CYLINDER
3713 017600 001003          BNE     10$         ;SKIP LAST TRACK CHECK IF NOT
3714 017602 020227 000004    CMP      R2,#4      ;LAST DISK TRACK ?
3715 017606 001515          BEQ     EXIT20     ;DON'T TEST LAST TRACK AS IT HAS BAD
3716 017610          10$:    BEQ     EXIT20     ;BLOCK INFORMATION STORED ON IT ----->
3717
3718 017610 010437 004272    MOV      R4,@#DTADPB+4 ;LOAD THE WORD COUNT
3719 017614 023701 001574    CMP      LC,R1       ;LAST DISK CYLINDER TO TEST ?
3720 017620 003005          BGT     5$          ;NO, SKIP TRACK CHECK
3721 017622 022702 000004    CMP      #4,R2       ;LAST DISK TRACK TO TEST ?
3722 017626 003002          BGT     5$          ;NO, SKIP NEXT
3723 017630 010537 004272    MOV      R5,@#DTADPB+4 ;SHORT WORD COUNT
3724 017634 017703 161300    5$:     MOV      @SWR,R3     ;INHIBIT WRITE AND
3725 017640 005103          COM     R3          ;WRITE CHECK?
3726 017642 032703 000030    BIT      #SW04!SW03,R3
3727 017646 001436          BEQ     7$          ;YES--BRANCH
3728 017650 004737 032602    JSR     PC,@#SETBUF  ;MOVE DATA PATTERN INTO THE BUFFER
3729 017654 032777 000020 161256 BIT      #SW04,@SWR  ;INHIBIT WRITE?
3730 017662 001012          BNE     6$          ;YES, DO NEXT PORTION OF TESTING
3731 017664 012737 017664 001110 MOV      #.$LPERR    ;SETUP THE ERROR LOOP ADDRESS
3732 017672 012706 001100    MOV      #STACK,SP  ;LOAD THE STACK POINTER

```


3733	017676	012737	000161	004270		MOV	#WRITE, @#DTADPB+2 ;COMMAND=WRITE DATA	
3734								
3735								
3736								
3737	017704	004037	027612			JSR	RO, @#DRVCAL ;START A DATA TRANSFER	
3738	017710	032777	000010	161222	6\$:	BIT	#SW03, @SWR ;INHIBIT WRITE CHECK?	
3739	017716	001012				BNE	7\$;YES--BRANCH	
3740	017720	012737	017720	001110		MOV	#, \$LPERR ;SETUP THE ERROR LOOP ADDRESS	
3741	017726	012706	001100			MOV	#STACK, SP ;LOAD THE STACK POINTER	
3742	017732	012737	000151	004270		MOV	#WRCKD, @#DTADPB+2 ;COMMAND=WRITE CHECK DATA	
3743	017740	004037	027612			JSR	RO, @#DRVCAL ;START A DATA TRANSFER	
3744	017744	005737	001414		7\$:	TST	@#WCEFLG ;WRITE CHECK ERROR FLAG SET?	
3745	017750	001404				BEQ	8\$;NO--BRANCH	
3746	017752	032777	000001	161160		BIT	#SW00, @SWR ;PERFORM READ AFTER FATAL "WCE"?	
3747	017760	001424				BEQ	9\$;NO--BRANCH	
3748	017762	032777	000004	161150	8\$:	BIT	#SW02, @SWR ;INHIBIT READ DATA AND SOFTWARE COMPARE?	
3749	017770	001020				BNE	9\$;YES--BRANCH	
3750	017772	012737	017772	001110		MOV	#, \$LPERR ;SETUP THE ERROR LOOP ADDRESS	
3751	020000	012706	001100			MOV	#STACK, SP ;LOAD THE STACK POINTER	
3752	020004	012737	000171	004270		MOV	#READ, @#DTADPB+2 ;COMMAND=READ	
3753	020012	004037	027612			JSR	RO, @#DRVCAL ;START A DATA TRANSFER	
3754	020016	032777	000002	161114		BIT	#SW01, @SWR ;COMPARE THE DATA?	
3755	020024	001002				BNE	9\$;NO--BRANCH	
3756	020026	004737	032672			JSR	PC, @#DATCMP ;YES--DO IT	
3757	020032	004037	032022		9\$:	JSR	RO, @#INCTRK ;MOVE TO NEXT TRACK	
3758	020036	000634				BR	1\$;OUT OF TRACKS GO TO NEXT PATTERN	
3759	020040	000653				BR	4\$;LOOP	
3760	020042	000004				EXIT20:	SCOPE ;SCOPE LOOP	

3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816

.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

:TST21:

```

020044 000240      NOP
020044 033737 001510 001316 BIT      BITS+(21*2-40),TSTNMS+2 ;DO THIS TEST ?
020054 001002      BNE      64$ ;YES--BRANCH
020056 000137 020702      JMP      TST22 ;NO--GO TO THE NEXT TEST
020062 012737 000021 001102 64$: MOV     #21,#$TSTNM ;SET UP TEST NUMBER AND
;CLEAR THE ERROR FLAG (SERFLG)
020070 004737 026632      JSR     PC,LODPRM ;LOAD THE PARAMETERS FOR THE TEST
020074 012737 020360 001110 MOV     #TST21,#$SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
020102 013777 001102 161032 MOV     $TSTNM,$DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
020110 013737 001570 001204 MOV     #RPT,$TIMES ;GET THE ITERATION COUNT
020116 113737 001444 001115 MOV     ERR.CT,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
020124 012737 000021 001224 MOV     #21,$TSTN ;SET TEST NUMBER IN APT MAIL BOX

;THIS IS A MACRO TO TYPE OUT THE TEST BEING EXECUTED IF SW12 IS SET
020132 032777 010000 161000 BIT     #SW12,$SWR ;CHECK FOR SWITCH 12 SET
020140 001413      BEQ     20$ ;IF = 0, THEN DON'T TYPE TEST NUMBER.
020142 013700 001102      MOV     $TSTNM,R0 ;ELSE, TYPE THE NUMBER
020146 042700 177000      BIC     #177000,R0 ;MASK OUT THE HIGH ORDER BYTE
020152 104401 001215      TYPE   $CRLF
020156 010046      MOV     R0,-(SP) ;SAVE R0 FOR TYPEOUT
;TEST NUMBER
020160 104403      TYPOS ;GO TYPE--OCTAL ASCII
020162 002 ;TYPE 2 DIGIT(S)
020163 000 ;SUPPRESS LEADING ZEROS
020164 104401 001215      TYPE   ,CRLF
020170
20$:

```


3817										
3818	020170	012737	020170	001106		MOV	#, SLPADR	; SETUP THE LOOP ADDRESS		
3819	020176	012737	020700	001332		MOV	#EXIT21, #BYPASS			
3820	020204	012737	176543	025630		MOV	#176543, #SHINUM	; PRIME THE RANDOM NUMBER GENERATOR		
3821	020212	012737	123456	025632		MOV	#123456, #SLONUM			
3822	020220	013737	001572	004300		MOV	#FC, #DTADPB+12	; CYLINDER		
3823	020226	013737	001432	004272		MOV	TRCKWC, #DTADPB+4	; WORD COUNT FOR 32/30 SECTORS (FULL TRACK)		
3824	020234	012737	051776	004274		MOV	#BUFFER, #DTADPB+6	; BUFFER ADDRESS		
3825	020242	012737	000161	004270		MOV	#WRITE, #DTADPB+2	; COMMAND		
3826	020250	032737	100000	001300		BIT	#SW15, #C.SWR	; WRITE THE DISK PACK BEFORE TESTING?		
3827	020256	001027				BNE	3\$; NO--BEGIN TESTING		
3828	020260	004037	033210			JSR	RO, #FILRAN	; FILL DATA BUFFER WITH RANDOM DATA		
3829	020264	005037	004276		1\$:	CLR	#DTADPB+10	; SECTOR AND TRACK		
3830	020270	012737	020270	001110		MOV	#, SLPERR	; SETUP THE ERROR LOOP ADDRESS		
3831	020276	012706	001100			MOV	#STACK, SP	; LOAD THE STACK POINTER		
3832	020302				2\$:					
3833	020302	004037	027612			JSR	RO, #DRVCL	; START A DATA TRANSFER		
3834	020306	105237	004277			INCB	#DTADPB+11	; NEXT TRACK		
3835	020312	122737	000005	004277		CMPB	#5, #DTADPB+11	; TIME FOR NEXT CYLINDER ?		
3836	020320	003370				BGT	2\$; NO--DO NEXT TRACK ON THIS CYL.		
3837	020322	005237	004300			INC	#DTADPB+12	; INCR CYLINDER ADDRESS		
3838	020326	023737	001574	004300		CMP	#LC, #DTADPB+12	; OUT OF CYLINDERS?		
3839	020334	002353				BGE	1\$; NO--CONTINUE SEQUENTIAL RANDOM WRITE		
3840										
3841										
3842	020336	012737	177400	004272	3\$:	MOV	#SCTRWC, #DTADPB+4	; WORD COUNT		
3843	020344	012737	020360	001106		MOV	#TEST21, #SLPADR			
3844	020352	012737	020360	001110		MOV	#TEST21, #SLPERR			
3845	020360	012706	001100		TEST21:	MOV	#STACK, SP	; SET STACK POINTER		
3846										
3847	020364	004037	033464			JSR	RO, #RANADR	; GENERATE A RANDOM ADDRESS		
3848	020370	013737	004276	001420		MOV	#DTADPB+10, #SVADR	; SAVE THE TRACK/SECTOR		
3849	020376	013737	004300	001422		MOV	#DTADPB+12, #SVADR+2	; SAVE THE CYLINDER		
3850	020404	012737	000161	004270		MOV	#WRITE, #DTADPB+2	; COMMAND=WRITE DATA		
3851	020412	012701	051776			MOV	#BUFFER, R1	; WRITE BUFFER ADDRESS FOR "RANPAT"		
3852	020416	010137	004274			MOV	R1, #DTADPB+6	; INTO THE DATA PARAMETER BLOCK		
3853	020422	004037	033430			JSR	RO, #RANPAT	; GENERATE RANDOM 256 WORD PATTERN		
3854								; AND PUT INTO THE WRITE BUFFER		
3855	020426	012737	020426	001110		MOV	#, SLPERR	; SETUP THE ERROR LOOP ADDRESS		
3856	020434	012706	001100			MOV	#STACK, SP	; LOAD THE STACK POINTER		
3857	020440	004037	027612			JSR	RO, #DRVCL	; START A DATA TRANSFER		
3858										
3859	020444	004037	033464			JSR	RO, #RANADR	; GENERATE A NEW RANDOM ADDRESS		
3860	020450	012737	000171	004270		MOV	#READ, #DTADPB+2	; COMMAND=READ DATA		
3861	020456	012737	052776	004274		MOV	#BUFFER+512, #DTADPB+6	; READ BUFFER ADDRESS		
3862	020464	012737	020464	001110		MOV	#, SLPERR	; SETUP THE ERROR LOOP ADDRESS		
3863	020472	012706	001100			MOV	#STACK, SP	; LOAD THE STACK POINTER		
3864	020476	004037	027612			JSR	RO, #DRVCL	; START A DATA TRANSFER		
3865	020502	005737	001446			TST	BASFLG	; IF BAD SECTOR ENCOUNTERED, SKIP NEXT CALL		
3866	020506	100402				BMI	+6	; DON'T COMPARE DATA		
3867	020510	004037	033232			JSR	RO, #RANCK	; SOFTWARE CHECK THE DATA		
3868										
3869	020514	013737	001420	004276		MOV	#SVADR, #DTADPB+10	; GET ADDRESS OF WHERE THE LAST		
3870	020522	013737	001422	004300		MOV	#SVADR+2, #DTADPB+12	; WRITE WAS PERFORMED		
3871	020530	012737	000151	004270		MOV	#WRCKD, #DTADPB+2	; COMMAND=WRITE CHECK DATA		
3872	020536	012737	051776	004274		MOV	#BUFFER, #DTADPB+6	; DATA BUFFER ADDRESS FOR HARDWARE		


```

3873
3874 020544 012737 020544 001110      MOV      #. $LPERR      ;CHECK OF THE DATA
3875 020552 012706 001100      MOV      $STACK, SP    ;SETUP THE ERROR LOOP ADDRESS
3876 020556 004037 027612      JSR      RO, $DRVCAL    ;LOAD THE STACK POINTER
3877                                     ;START A DATA TRANSFER
3878 020562 004037 033464      JSR      RO, $RANADR    ;GENERATE A NEW RANDOM ADDRESS
3879 020566 012737 000171 004270      MOV      $READ, $DTADPB+2 ;COMMAND=READ
3880 020574 012737 052776 004274      MOV      $BUFFER+512., $DTADPB+6 ;DATA BUFFER ADDRESS
3881 020602 012737 020602 001110      MOV      #. $LPERR      ;SETUP THE ERROR LOOP ADDRESS
3882 020610 012706 001100      MOV      $STACK, SP    ;LOAD THE STACK POINTER
3883 020614 004037 027612      JSR      RO, $DRVCAL    ;START A DATA TRANSFER
3884 020620 005737 001446      TST      BASFLG        ;ENCOUNTER BAD SECTOR ?
3885 020624 100402      BMI      +6            ;DON'T COMPARE DATA IF SO
3886 020626 004037 033232      JSR      RO, $RANCK    ;SOFTWARE CHECK THE DATA
3887
3888 020632 013737 001420 004276      MOV      $SVADR, $DTADPB+10 ;GET DISK ADDRESS OF THE
3889 020640 013737 001422 004300      MOV      $SVADR+2, $DTADPB+12 ;LAST WRITE
3890 020646 012737 000151 004270      MOV      $WRCKD, $DTADPB+2 ;COMMAND=WRITE CHECK DATA
3891 020654 012737 051776 004274      MOV      $BUFFER, $DTADPB+6 ;DATA BUFFER ADDRESS
3892 020662 012737 020662 001110      MOV      #. $LPERR      ;SETUP THE ERROR LOOP ADDRESS
3893 020670 012706 001100      MOV      $STACK, SP    ;LOAD THE STACK POINTER
3894 020674 004037 027612      JSR      RO, $DRVCAL    ;START A DATA TRANSFER
3895 020700 000004      EXIT21: SCOPE        ;LOOP ?
3896
3897 .SBTTL *** RMO3 ACCESS TIME ADJUSTMENT TEST ***
3898
3899 ;:*****
3900 ;*TEST 22 RMO3 ACCESS TIME ADJUSTMENT TEST
3901
3902 ;* THIS TEST PERFORMS SEEKS BETWEEN CYLINDERS 0 & 136 TO ALLOW THE
3903 ;* OPERATOR TO ADJUST THE ACCESS TIME ON AN RMO3 USING THE
3904 ;* DDU. THE PROGRAM STALLS APPROXIMATELY 5 SECONDS BETWEEN SEEKS
3905 ;* SO THAT THE ACCESS TIME INDICATORS ON THE DDU MAY BE OBSERVED.
3906
3907 ;:*****
3908 †ST22:
3909 020702 000240      NOP
3910 020704 033737 001512 001316      BIT      BITS+(22*2-40), TSTNMS+2 ;DO THIS TEST ?
3911 020712 001002      BNE     64$          ;YES--BRANCH
3912 020714 000137 021052      JMP     SEOP        ;NO--GO TO THE END OF THE PROGRAM
3913 020720 012737 000022 001102 64$: MOV      #22, $STSTNM ;SET UP TEST NUMBER AND
3914                                     ;CLEAR THE ERROR FLAG ($ERFLG)
3915 020726 004737 026632      JSR     PC, LODPRM  ;LOAD THE PARAMETERS FOR THE TEST
3916 020732 012737 020770 001110      MOV     $TEST22, $LPERR ;SETUP THE LOOP ON ERROR ADDRESS
3917 020740 013777 001102 160174      MOV     $STSTNM, $DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
3918 020746 013737 001570 001204      MOV     $RPT, $TIMES  ;GET THE ITERATION COUNT
3919 020754 112737 000144 001115      MOV     #100, $ERMAX  ;MAX ERRORS ALLOWED FOR TEST
3920 020762 012737 020770 001106      MOV     $TEST22, $LPADR ;SETUP THE LOOP ADDRESS
3921 020770 012706 001100      TEST22: MOV     $STACK, SP ;SETUP THE STACK POINTER
3922 020774 013737 001574 004220      MOV     LC, DPB.A+12  ;ENDING CYLINDER
3923 021002 112737 000105 004210      MOV     $SEEK, $DPB.A+2 ;SEEK=COMMAND
3924 021010 004037 027054      JSR     RO, $CALL.A  ;GO EXECUTE THE COMMAND
3925 021014 004037 030606      JSR     RO, STALL    ;STALL
3926 021020 001440      .WORD  STALL3       ;ADDRESS OF STALL VALUE
3927 021022 013737 001572 004220      MOV     FC, DPB.A+12 ;STARTING CYLINDER
3928 021030 112737 000105 004210      MOV     $SEEK, $DPB.A+2 ;SEEK=COMMAND

```



```

3929 021036 004037 027054
3930 021042 004037 030606
3931 021046 001440
3932 021050 000004
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944 021052
3945 021052 104401 021060
3946 021056 000410
3947
3948 021100
3949 021100 005737 001312
3950 021104 001434
3951 021106 104401 021114
3952 021112 000405
3953
3954 021126
3955 021126 013746 001334
3956 021132 104403
3957 021134 002
3958 021135 000
3959 021136 104401 021144
3960 021142 000412
3961
3962 021170
3963 021170 013746 001112
3964 021174 104402
3965 021176 005037 001112
3966 021202 005037 001102
3967 021206 005037 001204
3968 021212 005237 001226
3969 021216 042737 100000 001226
3970 021224 005327
3971 021226 000010
3972 021230 003027
3973 021232 012737
3974 021234 000010
3975 021236 021226
3976 021240 104401 021246
3977 021244 000407
3978
3979 021264
3980 021264 104401 021314
3981 021270 013700 000042
3982 021274 001405
3983 021276 000005
3984 021300 004710

```

```

JSR RO, @CALL.A ;GO EXECUTE THE COMMAND
JSR RO, STALL ;STALL
;WORD STALL3 ;ADDRESS OF STALL VALUE
EXIT22: SCOPE ;LOOP ?

.SBTTL END OF PASS ROUTINE

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

SEOP:
TYPE 65$ ;:TYPE ASCIZ STRING
BR 64$ ;:GET OVER THE ASCIZ
;:65$: .ASCIZ <CR><LF><LF>/END OF PASS/
64$:
TST @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 @CHKDRV, -(SP) ;:SAVE @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 @SERTTL, -(SP) ;:SAVE @SERTTL FOR TYPEOUT
TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1$: CLR @SERTTL ;:ZERO ERROR TOTAL
CLR $STNM ;:ZERO THE TEST NUMBER
CLR $TIMES ;:ZERO THE NUMBER OF ITERATIONS
INC $PASS ;:INCREMENT THE PASS NUMBER
BIC #10000, $PASS ;:DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;:LOOP?
SEOPCT: .WORD 8.
BGT $DOAGN ;:YES
MOV (PC)+, @PC+ ;:RESTORE COUNTER
SENDCT: .WORD 8.
SEOPCT
TYPE 65$ ;:TYPE ASCIZ STRING
BR 64$ ;:GET OVER THE ASCIZ
;:65$: .ASCIZ <CR><LF>/END OF TEST/
64$:
TYPE $ENULL ;:TYPE NULL CHARACTER
$GET42: MOV @42, RO ;:GET MONITOR ADDRESS
BEQ $DOAGN ;:BRANCH IF NO MONITOR
RESET ;:CLEAR THE WORLD
SENDAD: JSR PC, (RO) ;:GO TO MONITOR

```



```

3993
3994
3995
3996
3997
3998
3999 021320 112737 000001 021564 $ATY1: MOV  #1,$FFLG ;; TO REPORT FATAL ERROR
4000 021326 112737 000001 021562 $ATY3: MOV  #1,$MFLG ;; TO TYPE A MESSAGE
4001 021334 000403          $ATYC: BR          ;;
4002 021336 112737 000001 021564 $ATY4: MOV  #1,$FFLG ;; TO ONLY REPORT FATAL ERROR
4003 021344          $ATYC:
4004 021344 010046          MOV  RO,-(SP) ;; PUSH RO ON STACK
4005 021346 010146          MOV  R1,-(SP) ;; PUSH R1 ON STACK
4006 021350 105737 021562          TSTB $MFLG ;; SHOULD TYPE A MESSAGE?
4007 021354 001450          BEQ  5$ ;; IF NOT: BR
4008 021356 122737 000001 001240          CMPB #APTENV,$ENV ;; OPERATING UNDER APT?
4009 021364 001031          BNE  3$ ;; IF NOT: BR
4010 021366 132737 000100 001241          BITB #APTPOOL,$ENVM ;; SHOULD SPOOL MESSAGES?
4011 021374 001425          BEQ  3$ ;; IF NOT: BR
4012 021376 017600 000004          MOV  #4(SP),RO ;; GET MESSAGE ADDR.
4013 021402 062766 000002 000004          ADD  #2,4(SP) ;; BUMP RETURN ADDR.
4014 021410 005737 001220          1$: TST  $MSGTYPE ;; SEE IF DONE W/ LAST XMISSION?
4015 021414 001375          BNE  1$ ;; IF NOT: WAIT
4016 021416 010037 001234          MOV  RO,$MSGAD ;; PUT ADDR IN MAILBOX
4017 021422 105720          2$: TSTB (RO)+ ;; FIND END OF MESSAGE
4018 021424 001376          BNE  2$
4019 021426 163700 001234          SUB  $MSGAD,RO ;; SUB START OF MESSAGE
4020 021432 006200          ASR  RO ;; GET MESSAGE LNTH IN WORDS
4021 021434 010037 001236          MOV  RO,$MSG LGT ;; PUT LENGTH IN MAILBOX
4022 021440 012737 000004 001220          MOV  #4,$MSGTYPE ;; TELL APT TO TAKE MSG.
4023 021446 000413          BR  5$
4024 021450 017637 000004 021474 3$: MOV  #4(SP),4$ ;; PUT MSG ADDR IN JSR LINKAGE
4025 021456 062766 000002 000004          ADD  #2,4(SP) ;; BUMP RETURN ADDRESS
4026 021464 013746 177776          MOV  177776,-(SP) ;; PUSH 177776 ON STACK
4027 021470 004737 022422          JSR  PC,$TYPE ;; CALL TYPE MACRO
4028 021474 000000          4$: .WORD 0
4029 021476          5$:
4030 021476 105737 021564          10$: TSTB $FFLG ;; SHOULD REPORT FATAL ERROR?
4031 021502 001416          BEQ  12$ ;; IF NOT: BR
4032 021504 005737 001240          TST  $ENV ;; RUNNING UNDER APT?
4033 021510 001413          BEQ  12$ ;; IF NOT: BR
4034 021512 005737 001220          11$: TST  $MSGTYPE ;; FINISHED LAST MESSAGE?
4035 021516 001375          BNE  11$ ;; IF NOT: WAIT
4036 021520 017637 000004 001222          MOV  #4(SP),$FATAL ;; GET ERROR #
4037 021526 062766 000002 000004          ADD  #2,4(SP) ;; BUMP RETURN ADDR.
4038 021534 005237 001220          INC  $MSGTYPE ;; TELL APT TO TAKE ERROR
4039 021540 105037 021564          12$: CLRB $FFLG ;; CLEAR FATAL FLAG
4040 021544 105037 021563          CLRB $LFLG ;; CLEAR LOG FLAG
4041 021550 105037 021562          CLRB $MFLG ;; CLEAR MESSAGE FLAG
4042 021554 012601          MOV  (SP)+,R1 ;; POP STACK INTO R1
4043 021556 012600          MOV  (SP)+,RO ;; POP STACK INTO RO
4044 021560 000207          RTS  PC ;; RETURN
4045 021562 000          $MFLG: .BYTE 0 ;; MESSG. FLAG
4046 021563 000          $LFLG: .BYTE 0 ;; LOG FLAG
4047 021564 000          $FFLG: .BYTE 0 ;; FATAL FLAG
4048          021566          .EVEN

```

```

4049      000200      APTSIZE=200
4050      000001      APTENV=001
4051      000100      APTSPOOL=100
4052      000040      APTCSUP=040
4053      .SBTTL      ERROR HANDLER ROUTINE
4054
4055      ;*****
4056      ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
4057      ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
4058      ;*AND GO TO TYPERR ON ERROR
4059      ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4060      ;*SW15=1      HALT ON ERROR
4061      ;*SW13=1      INHIBIT ERROR TYPEOUTS
4062      ;*SW10=1      BELL ON ERROR
4063      ;*SW09=1      LOOP ON ERROR
4064      ;*CALL
4065      ;*      ERROR      N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
4066
4067      021566      104407      $ERROR:      CKSWR      ;:TEST FOR CHANGE IN SOFT-SWR
4068      021566      032777      000400      157342      BIT      #SW08, @SWR      ;:SEND ERROR MESSAGE TO TTY?
4069      021570      001411      BEQ      7$      ;:YES--BRANCH
4070      021576      005737      001310      TST      @LPTAVL      ;:IS THERE A LINE PRINTER AVAILABLE?
4071      021600      001406      BEQ      7$      ;:NO--BRANCH
4072      021604      013737      001502      001150      MOV      @LPS, @STPS      ;:YES--SETUP STATUS
4073      021614      013737      001504      001152      MOV      @LPB, @$.TPB      ;:AND BUFFER REG.'S FOR LINE PRINTER
4074      021622      105237      001103      7$:      INCB      $ERFLG      ;:SET THE ERROR FLAG
4075      021626      001775      BEQ      7$      ;:DON'T LET THE FLAG GO TO ZERO
4076      021630      013777      001102      157304      MOV      $STNM, @DISPLAY      ;:DISPLAY TEST NUMBER AND ERROR FLAG
4077      021636      032777      002000      157274      BIT      @BIT10, @SWR      ;:BELL ON ERROR?
4078      021644      001402      BEQ      1$      ;:NO - SKIP
4079      021646      104401      001210      TYPE      $BELL      ;:RING BELL
4080      021652      005237      001112      1$:      INC      $ERTTL      ;:COUNT THE NUMBER OF ERRORS
4081      021656      011637      001116      MOV      (SP), $ERRPC      ;:GET ADDRESS OF ERROR INSTRUCTION
4082      021662      162737      000002      001116      SUB      @2, $ERRPC
4083      021670      117737      157222      001114      MOV      @2, $ERRPC
4084      021676      032777      020000      157234      MOV      @$.ERRPC, $ITEMB      ;:STRIP AND SAVE THE ERROR ITEM CODE
4085      021704      001004      BIT      @BIT13, @SWR      ;:SKIP TYPEOUT IF SET
4086      021706      004737      022034      BNE      20$      ;:SKIP TYPEOUTS
4087      021712      104401      001215      JSR      PC, TYPERR      ;:GO TO USER ERROR ROUTINE
4088      021716      122737      000001      001240      20$:      TYPE      , $CRLF
4089      021724      001007      CMPB      #APTENV, $ENV      ;:RUNNING IN APT MODE
4090      021726      113737      001114      021740      BNE      2$      ;:NO SKIP APT ERROR REPORT
4091      021734      004737      021336      MOV      $ITEMB, 21$      ;:SET ITEM NUMBER AS ERROR NUMBER
4092      021740      000      JSR      PC, $ATY4      ;:REPORT FATAL ERROR TO APT
4093      021741      000      21$:      .BYTE      0
4094      021742      000777      BR      22$      ;:APT ERROR LOOP
4095      021744      005777      157170      22$:      TST      @SWR      ;:HALT ON ERROR
4096      021750      100002      BPL      3$      ;:SKIP IF CONTINUE
4097      021752      000000      HALT      ;:HALT ON ERROR!
4098      021754      104407      CKSWR      ;:TEST FOR CHANGE IN SOFT-SWR
4099      021756      032777      001000      157154      3$:      BIT      #BIT09, @SWR      ;:LOOP ON ERROR SWITCH SET?
4100      021764      001402      BEQ      4$      ;:BR IF NO
4101      021766      013716      001110      MOV      $LPERR, (SP)      ;:FUDGE RETURN FOR LOOPING
4102      021772      005737      001206      4$:      TST      $ESCAPE      ;:CHECK FOR AN ESCAPE ADDRESS

```



```

4105 021776 001402          BEQ      5$          ;;BR IF NONE
4106 022000 013716 001206    MOV      $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
4107 022004          5$:          CMP      #SENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
4108 022004 022737 021300 000042 BNE      6$          ;;BRANCH IF NO
4109 022012 001001          HALT          ;;YES
4110 022014 000000          6$:          MOV      @#TPS,@#STPS ;;SET STATUS AND BUFFER REG.'S
4111 022016          MOV      @#TPB,@#STPB ;;FOR TTY
4112 022016 013737 001476 001150 RTI          ;;RETURN FROM ERROR CALL
4113 022024 013737 001500 001152
4114 022032 000002
4115
4116 ;;*****
4117 ;SBTTL  TYPERR - TYPE ERROR ROUTINE
4118 ;THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE
4119 ;WHICH ERROR IS TO BE REPORTED, IT THEN OBTAINS, FROM THE "ERROR
4120 ;TABLE" ($ERRTB), AND REPORTS THE APPROPRIATE INFORMATION
4121 ;CONCERNING THE ERROR.
4122 ;CALL
4123 ;
4124 ;      JSR      PC,@#TYPERR
4125 ;
4126 022034 113737 001102 001176 TYPERR: MOVB    @#STSTNM,@#STMPO ;SAVE THE TEST NUMBER
4127 022042 104412          SAVREG          ;SAVE R0 - R5
4128 022044 162700 000004          SUB      #4,R0          ;FORM TEST PC
4129 022050 010037 001162          MOV      R0,@#$REG0    ;COPY R0-R5 IN $REG0-$REG5
4130 022054 010137 001164          MOV      R1,@#$REG1
4131 022060 010237 001166          MOV      R2,@#$REG2
4132 022064 010337 001170          MOV      R3,@#$REG3
4133 022070 010437 001172          MOV      R4,@#$REG4
4134 022074 010537 001174          MOV      R5,@#$REG5
4135 022100 113700 001114          MOVB    @#$ITEMB,R0    ;PICKUP ERROR ITEM NUMBER
4136 022104 010001          MOV      R0,R1          ;AND COPY IT INTO R1
4137 022106 005300          DEC      R0              ;FORM INDEX FOR ERROR TABLE
4138 022110 106300          ASLB    R0
4139 022112 106300          ASLB    R0
4140 022114 106300          ASLB    R0
4141 022116 103002          BCC     1$              ;IS ERROR > 37?
4142 022120 062700 000240          ADD     #ITEM41-$ERRTB,R0 ;YES--FORM OFFSET
4143 022124 062700 004410          1$:    ADD     #$ERRTB,R0    ;FORM ADDRESS
4144 022130 012037 022144          MOV     (R0)+,2$        ;GET ERROR MESSAGE (EM) POINTER
4145 022134 001447          BEQ     7$              ;BRANCH IF THERE ISN'T ONE
4146 022136 104401 001215          TYPE   ,SCRLF          ;"CARRIAGE RETURN - LINE FEED"
4147 022142 104401          TYPE
4148 022144 000000          2$:    .WORD 0              ;"EM" POINTER GOES HERE
4149 022146 162701 000041          SUB     #41,R1          ;SPECIAL ERROR ITEM NUMBER?
4150 022152 100440          BMI     7$              ;NO--BRANCH
4151 022154 013701 001340          MOV     @#SVSTAT,R1     ;GET STATUS/ERROR INDICATOR
4152 022160 106301          ASLB    R1              ;STRIP "DONE" BIT (BIT07)
4153 022162 006301          ASL     R1              ;STRIP "ERROR" BIT (BIT15)
4154 022164 012702 004356          MOV     #STATBL,R2      ;1ST ADDRESS ON STATUS MESSAGE POINTERS
4155 022170 005003          CLR     R3              ;CARRIAGE RETURN-LINE FEED SWITCH
4156 022172 104401 022200          TYPE   ,65$           ;;TYPE ASCIZ STRING
4157 022176 000402          BR      64$           ;;GET OVER THE ASCIZ
4158
4159 022204          65$: .ASCIZ / (/
4160 022204 012237 022226          64$: 3$: MOV     (R2)+,5$      ;MESSAGE POINTER

```

4161	022210	006301		ASL	R1		;TYPE THIS MESSAGE?
4162	022212	103013		BCC	6\$;NO--BRANCH
4163	022214	005103		COM	R3		;YES--TYPE A "CR" & "LF"?
4164	022216	001002		BNE	4\$;NO--BRANCH
4165	022220	104401	001215	TYPE	, \$CRLF		;YES
4166	022224	104401		4\$:	TYPE		
4167	022226	000000		5\$:	.WORD	0	;MESSAGE POINTER GOES HERE
4168	022230	005701		TST	R1		;MORE TO TYPE?
4169	022232	001403		BEQ	6\$;NO--BRANCH
4170	022234	104401	046141	TYPE	,MSG.SP		;YES--SPACES
4171	022240	000761		BR	3\$;LOOP
4172	022242	001360		6\$:	BNE	3\$;BRANCH IF NOT FINISHED
4173	022244	104401	022252	TYPE	,67\$;TYPE ASCIZ STRING
4174	022250	000401		BR	66\$;GET OVER THE ASCIZ
4175				67\$:	.ASCIZ	/)/	
4176	022254			66\$:			
4177	022254	012037	022270	7\$:	MOV	(R0)+,8\$;PICK UP DATA HEADER (DH) POINTER
4178	022260	001404		BEQ	9\$;BRANCH IF NONE
4179	022262	104401	001215	TYPE	, \$CRLF		;CARRIAGE RETURN-LINE FEED
4180	022266	104401		TYPE			
4181	022270	000000		8\$:	.WORD	0	; "DH" POINTER GOES HERE
4182	022272	012001		9\$:	MOV	(R0)+,R1	;PICKUP DATA TABLE (DT) POINTER
4183	022274	001450		BEQ	20\$;BRANCH IF NONE
4184	022276	005005		CLR	R5		;SET INDENT SWITCH
4185	022300	012000		MOV	(R0)+,R0		;DATA FORMAT (DF) POINTER
4186	022302	012002		MOV	(R0)+,R2		;NUMBER OF DH'S TO TYPE
4187	022304	001441		BEQ	17\$;BRANCH IF DH NUMBER IS 0
4188	022306	005105		COM	R5		;NO INDENT
4189	022310	104401	001215	TYPE	, \$CRLF		;CARRIAGE RETURN-LINE FEED
4190	022314	112003		10\$:	MOVB	(R0)+,R3	;NUMBER OF DATA WORDS TO TYPE
4191	022316	112004		MOVB	(R0)+,R4		;AND HOW TO TYPE THEM
4192	022320	006004		11\$:	ROR	R4	;OCTAL OR DECIMAL?
4193	022322	103403		BCS	12\$;DECIMAL--BRANCH
4194	022324	013146		MOV	2(R1)+,-(SP)		;SAVE 2(R1)+ FOR TYPEOUT
4195	022326	104402		TYPOC			;GO TYPE--OCTAL ASCII(ALL DIGITS)
4196	022330	000402		BR	13\$		
4197	022332			12\$:			
4198	022332	013146		MOV	2(R1)+,-(SP)		;SAVE 2(R1)+ FOR TYPEOUT
4199	022334	104405		TYPDS			;GO TYPE--DECIMAL ASCII WITH SIGN
4200	022336	005303		13\$:	DEC	R3	;MORE NUMBERS TO TYPE?
4201	022340	001403		BEQ	14\$;NO--BRANCH
4202	022342	104401	046141	TYPE	,MSG.SP		;YES--TYPE SEPERATORS
4203	022346	000764		BR	11\$;LOOP
4204	022350	005302		14\$:	DEC	R2	;MORE DH'S?
4205	022352	003421		BLE	20\$;NO--BRANCH
4206	022354	104401	001215	TYPE	, \$CRLF		;YES--START A NEW LINE
4207	022360	005105		COM	R5		;INDENT?
4208	022362	001002		BNE	15\$;NO--BRANCH
4209	022364	104401	046141	TYPE	,MSG.SP		;YES--TYPE SPACES
4210	022370	012037	022376	15\$:	MOV	(R0)+,16\$;GET NEXT DH
4211	022374	104401		TYPE			;AND TYPE IT
4212	022376	000000		16\$:	.WORD	0	;DH POINTER GOES HERE
4213	022400	104401	001215	TYPE	, \$CRLF		;CARRIAGE RETURN-LINE FEED
4214	022404	005705		TST	R5		;INDENT?
4215	022406	001342		BNE	10\$;NO--BRANCH
4216	022410	104401	046141	17\$:	TYPE	,MSG.SP	;YES--TYPE SPACES

4217 022414 000737
4218 022416 104413
4219 022420 000207

20\$: BR 10\$;LOOP
RESREG ;RESTORE R0 - R5
RTS PC ;RETURN

.SBTTL TYPE ROUTINE

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

*CALL:
*1) USING A TRAP INSTRUCTION
* TYPE ,MESADR ; ;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
*OR
* TYPE
* MESADR
*

4220 022422 105737 001157
4221 022426 100002
4222 022430 000000
4223 022432 000430
4224 022434 010046
4225 022436 017600 000002
4226 022442 122737 000001 001240
4227 022450 001011
4228 022452 132737 000100 001241
4229 022460 001405
4230 022462 010037 022472
4231 022466 004737 021326
4232 022472 000000
4233 022474 132737 000040 001241
4234 022502 001003
4235 022504 112046
4236 022506 001005
4237 022510 005726
4238 022512 012600
4239 022514 062716 000002
4240 022520 000002
4241 022522 122716 000011
4242 022526 001430
4243 022530 122716 000200
4244 022534 001006
4245 022536 005726
4246 022540 104401
4247 022542 001215
4248 022544 105037 022700
4249 022550 000755
4250 022552 004737 022634
4251 022556 123726 001156
4252 022562 001350
4253 022564 013746 001154

\$TYPE: TSTB \$TPFLG ; IS THERE A TERMINAL?
BPL 1\$; BR IF YES
HALT ; HALT HERE IF NO TERMINAL
BR 3\$; LEAVE
1\$: MOV RO, -(SP) ; SAVE RO
MOV @2(SP), RO ; GET ADDRESS OF ASCIZ STRING
CMPB #APTENV, \$ENV ; RUNNING IN APT MODE
BNE 62\$; NO, GO CHECK FOR APT CONSOLE
BITB #APTSPOOL, \$ENV ; SPOOL MESSAGE TO APT
BEQ 62\$; NO, GO CHECK FOR CONSOLE
MOV RO, 61\$; SETUP MESSAGE ADDRESS FOR APT
JSR PC, \$ATY3 ; SPOOL MESSAGE TO APT
0 ; MESSAGE ADDRESS
61\$: BITB #APTCSUP, \$ENV ; APT CONSOLE SUPPRESSED
62\$: BNE 60\$; YES, SKIP TYPE OUT
2\$: MOVB (RO)+, -(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)+, RO ; RESTORE RO
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.

```

4273 022570 105366 000001 7$: DEC B 1(SP) ;: DOES A NULL NEED TO BE TYPED?
4274 022574 002770 BLT 6$ ;: BR IF NO--GO POP THE NULL OFF OF STACK
4275 022576 004737 022634 JSR PC,$TYPE C ;: GO TYPE A NULL
4276 022602 105337 022700 DEC B $CHARCNT ;: DO NOT COUNT AS A COUNT
4277 022606 000770 BR 7$ ;: LOOP

```

;HORIZONTAL TAB PROCESSOR

```

4281 022610 112716 000040 8$: MOV B #' (SP) ;: REPLACE TAB WITH SPACE
4282 022614 004737 022634 9$: JSR PC,$TYPE C ;: TYPE A SPACE
4283 022620 132737 000007 022700 BIT B #',$CHARCNT ;: BRANCH IF NOT AT
4284 022626 001372 BNE 9$ ;: TAB STOP
4285 022630 005726 TST (SP)+ ;: POP SPACE OFF STACK
4286 022632 000724 BR 2$ ;: GET NEXT CHARACTER
4287 022634 105777 156310 $TYPE C: TST B $STPS ;: WAIT UNTIL PRINTER IS READY
4288 022640 100375 BPL $TYPE C
4289 022642 116677 000002 156302 MOV B 2(SP),$STPB ;: LOAD CHAR TO BE TYPED INTO DATA REG.
4290 022650 122766 000015 000002 CMP B #CR,2(SP) ;: IS CHARACTER A CARRIAGE RETURN?
4291 022656 001003 BNE 1$ ;: BRANCH IF NO
4292 022660 105037 022700 CLRB $CHARCNT ;: YES--CLEAR CHARACTER COUNT
4293 022664 000406 BR $TYPE X ;: EXIT
4294 022666 122766 000012 000002 1$: CMP B #LF,2(SP) ;: IS CHARACTER A LINE FEED?
4295 022674 001402 BEQ $TYPE X ;: BRANCH IF YES
4296 022676 105227 INCB (PC)+ ;: COUNT THE CHARACTER
4297 022700 000000 $CHARCNT: WORD 0 ;: CHARACTER COUNT STORAGE
4298 022702 000207 $TYPE X: RTS PC

```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

4303 ;: *****
4304 ;: *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
4305 ;: *OCTAL (ASCII) NUMBER AND TYPE IT.
4306 ;: *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
4307 ;: *CALL:
4308 ;: * MOV NUM,-(SP) ;: NUMBER TO BE TYPED
4309 ;: * TYPOS ;: CALL FOR TYPEOUT
4310 ;: * .BYTE N ;: N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
4311 ;: * .BYTE M ;: M=1 OR 0
4312 ;: * ;: ;: 1=TYPE LEADING ZEROS
4313 ;: * ;: ;: 0=SUPPRESS LEADING ZEROS
4314 ;: *
4315 ;: *$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
4316 ;: *$TYPOS OR $TYPOC
4317 ;: *CALL:
4318 ;: * MOV NUM,-(SP) ;: NUMBER TO BE TYPED
4319 ;: * TYPON ;: CALL FOR TYPEOUT
4320 ;: *
4321 ;: *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
4322 ;: *CALL:
4323 ;: * MOV NUM,-(SP) ;: NUMBER TO BE TYPED
4324 ;: * TYPOC ;: CALL FOR TYPEOUT
4325 ;: *
4326 022704 017646 000000 $TYPOS: MOV 2(SP),-(SP) ;: PICKUP THE MODE
4327 022710 116637 000001 023127 MOV B 1(SP),$OFILL ;: LOAD ZERO FILL SWITCH
4328 022716 112637 023131 MOV B (SP)+,$OMODE+1 ;: NUMBER OF DIGITS TO TYPE

```



```

4329 022722 062716 000002          ADD      #2,(SP)          ;;ADJUST RETURN ADDRESS
4330 022726 000406          BR       $TYPON
4331 022730 112737 000001 023127 $TYPON: MOVB    #1,$OFILL        ;;SET THE ZERO FILL SWITCH
4332 022736 112737 000006 023131 MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
4333 022744 112737 000005 023126 $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
4334 022752 010346          MOV     R3,-(SP)        ;;SAVE R3
4335 022754 010446          MOV     R4,-(SP)        ;;SAVE R4
4336 022756 010546          MOV     R5,-(SP)        ;;SAVE R5
4337 022760 113704 023131          MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
4338 022764 005404          NEG     R4
4339 022766 062704 000006          ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
4340 022772 110437 023130          MOVB    R4,$OMODE      ;;SAVE IT FOR USE
4341 022776 113704 023127          MOVB    $OFILL,R4     ;;GET THE ZERO FILL SWITCH
4342 023002 016605 000012          MOV     12(SP),R5     ;;PICKUP THE INPUT NUMBER
4343 023006 005003          CLR     R3             ;;CLEAR THE OUTPUT WORD
4344 023010 006105          1$:    ROL     R5           ;;ROTATE MSB INTO "C"
4345 023012 000404          BR      3$
4346 023014 006105          2$:    ROL     R5           ;;FORM THIS DIGIT
4347 023016 006105          ROL     R5
4348 023020 006105          ROL     R5
4349 023022 010503          MOV     R5,R3
4350 023024 006103          3$:    ROL     R3           ;;GET LSB OF THIS DIGIT
4351 023026 105337 023130          DECB   $OMODE         ;;TYPE THIS DIGIT?
4352 023032 100016          BPL    7$             ;;BR IF NO
4353 023034 042703 177770          BIC    #177770,R3     ;;GET RID OF JUNK
4354 023040 001002          BNE    4$             ;;TEST FOR 0
4355 023042 005704          TST   R4             ;;SUPPRESS THIS 0?
4356 023044 001403          BEQ   5$             ;;BR IF YES
4357 023046 005204          4$:    INC    R4           ;;DON'T SUPPRESS ANYMORE 0'S
4358 023050 052703 000060          BIS   #'0,R3         ;;MAKE THIS DIGIT ASCII
4359 023054 052703 000040          5$:    BIS   #' ,R3      ;;MAKE ASCII IF NOT ALREADY
4360 023060 110337 023124          MOVB   R3,8$         ;;SAVE FOR TYPING
4361 023064 104401 023124          TYPE  8$           ;;GO TYPE THIS DIGIT
4362 023070 105337 023126          7$:    DECB   $OCNT     ;;COUNT BY 1
4363 023074 003347          BGT   2$             ;;BR IF MORE TO DO
4364 023076 002402          BLT   6$             ;;BR IF DONE
4365 023100 005204          INC   R4             ;;INSURE LAST DIGIT ISN'T A BLANK
4366 023102 000744          BR    2$             ;;GO DO THE LAST DIGIT
4367 023104 012605          6$:    MOV   (SP)+,R5   ;;RESTORE R5
4368 023106 012604          MOV   (SP)+,R4       ;;RESTORE R4
4369 023110 012603          MOV   (SP)+,R3       ;;RESTORE R3
4370 023112 016666 000002 000004          MOV   2(SP),4(SP)    ;;SET THE STACK FOR RETURNING
4371 023120 012616          MOV   (SP)+,(SP)
4372 023122 000002          RTI
4373 023124          8$:    .BYTE 0           ;;RETURN
4374 023125          .BYTE 0           ;;STORAGE FOR ASCII DIGIT
4375 023126          .BYTE 0           ;;TERMINATOR FOR TYPE ROUTINE
4376 023127          .BYTE 0           ;;OCTAL DIGIT COUNTER
4377 023130 000000          .WORD 0           ;;ZERO FILL SWITCH
4378          .WORD 0           ;;NUMBER OF DIGITS TO TYPE
4379          .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
4380
4381 *****
4382 ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
4383 ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
4384 ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED

```



```

4385 ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
4386 ;*REPLACED WITH SPACES.
4387 ;*CALL:
4388 ;*      MOV      NUM,-(SP)      ;:PUT THE BINARY NUMBER ON THE STACK
4389 ;*      TYPDS                    ;:GO TO THE ROUTINE
4390
4391 $TYPDS:
4392      MOV      R0,-(SP)      ;:PUSH R0 ON STACK
4393      MOV      R1,-(SP)      ;:PUSH R1 ON STACK
4394      MOV      R2,-(SP)      ;:PUSH R2 ON STACK
4395      MOV      R3,-(SP)      ;:PUSH R3 ON STACK
4396      MOV      R5,-(SP)      ;:PUSH R5 ON STACK
4397      MOV      #20200,-(SP)  ;:SET BLANK SWITCH AND SIGN
4398      MOV      20(SP),R5    ;:GET THE INPUT NUMBER
4399      BPL      1$          ;:BR IF INPUT IS POS.
4400      NEG      R5          ;:MAKE THE BINARY NUMBER POS.
4401      MOVSB   #'-,1(SP)    ;:MAKE THE ASCII NUMBER NEG.
4402      CLR      R0          ;:ZERO THE CONSTANTS INDEX
4403      MOV      #SDBLK,R3    ;:SETUP THE OUTPUT POINTER
4404      MOVSB   #'',(R3)+    ;:SET THE FIRST CHARACTER TO A BLANK
4405      CLR      R2          ;:CLEAR THE BCD NUMBER
4406      MOV      $DTBL(R0),R1 ;:GET THE CONSTANT
4407      SUB      R1,R5        ;:FORM THIS BCD DIGIT
4408      BLT     4$          ;:BR IF DONE
4409      INC     R2          ;:INCREASE THE BCD DIGIT BY 1
4410      BR     3$
4411      ADD     R1,R5        ;:ADD BACK THE CONSTANT
4412      TST    R2          ;:CHECK IF BCD DIGIT=0
4413      BNE    5$          ;:FALL THROUGH IF 0
4414      TSTB   (SP)        ;:STILL DOING LEADING 0'S?
4415      BMI    7$          ;:BR IF YES
4416      ASLB   (SP)        ;:MSD?
4417      BCC   6$          ;:BR IF NO
4418      MOVB   1(SP),-1(R3) ;:YES--SET THE SIGN
4419      BIS    #'0,R2       ;:MAKE THE BCD DIGIT ASCII
4420      BIS    #' ,R2       ;:MAKE IT A SPACE IF NOT ALREADY A DIGIT
4421      MOVB   R2,(R3)+    ;:PUT THIS CHARACTER IN THE OUTPUT BUFFER
4422      TST    (R0)+      ;:JUST INCREMENTING
4423      CMP    R0,#10     ;:CHECK THE TABLE INDEX
4424      BLT    2$          ;:GO DO THE NEXT DIGIT
4425      BGT    8$          ;:GO TO EXIT
4426      MOV    R5,R2      ;:GET THE LSD
4427      BR     6$          ;:GO CHANGE TO ASCII
4428      TSTB   (SP)+      ;:WAS THE LSD THE FIRST NON-ZERO?
4429      BPL    9$          ;:BR IF NO
4430      MOVB   -1(SP),-2(R3) ;:YES--SET THE SIGN FOR TYPING
4431      CLRB   (R3)       ;:SET THE TERMINATOR
4432      MOV    (SP)+,R5    ;:POP STACK INTO R5
4433      MOV    (SP)+,R3    ;:POP STACK INTO R3
4434      MOV    (SP)+,R2    ;:POP STACK INTO R2
4435      MOV    (SP)+,R1    ;:POP STACK INTO R1
4436      MOV    (SP)+,R0    ;:POP STACK INTO R0
4437      TYPE   $SDBLK     ;:NOW TYPE THE NUMBER
4438      MOV    2(SP),4(SP) ;:ADJUST THE STACK
4439      MOV    (SP)+,(SP)
4440      RTI

```



```

4441 023336 023420
4442 023340 001750
4443 023342 000144
4444 023344 000012
4445 023346 000004
4446
4447
4448
4449
4450
4451 023356 000000
4452 023360 000000
4453 023362 000000
4454 023364 000002
4455 023366
4456
4457
4458
4459
4460
4461
4462
4463
4464
4465 023366 005037 023356
4466 023372 012737 023364 023360
4467 023400 013737 023360 023362
4468 023406 012737 023436 000060
4469 023414 012737 000200 000062
4470 023422 005777 155520
4471 023426 012777 000100 155510
4472 023434 000207
4473
4474
4475
4476
4477
4478
4479
4480
4481 023436 117746 155504
4482 023442 042716 177600
4483 023446 021627 000003
4484 023452 001007
4485 023454 104401 024566
4486 023460 004737 023366
4487 023464 005726
4488 023466 000137 004762
4489 023472 021627 000007
4490 023476 001004
4491 023500 022737 000176 001140
4492 023506 001500
4493
4494 023510
4495 023510 022737 000002 023356
4496 023516 001004

```

```

SDTBL: 10000.
        1000.
        100.
        10.
SDBLK: .BLKW 4
.SBTTL TTY INPUT ROUTINE
;*****
.ENABLE LSB
$TKCNT: .WORD 0 ;: NUMBER OF ITEMS IN QUEUE
$TKQIN: .WORD 0 ;: INPUT POINTER
$TKQOUT: .WORD 0 ;: OUTPUT POINTER
$TKQSRT: .BLKW 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:
; * JSR PC,$TKINT
; * RETURN
$TKINT: CLR $TKCNT ;: CLEAR COUNT OF ITEMS IN QUEUE
        MOV $TKQSRT,$TKQIN ;: MOVE THE STARTING ADDRESS OF THE
        MOV $TKQIN,$TKQOUT ;: QUEUE INTO THE INPUT & OUTPUT POINTERS.
        MOV $TKSRV,$TKVEC ;: INITIALIZE THE KEYBOARD VECTOR
        MOV $200,$TKVEC+2 ;: "BR" LEVEL 4
        TST $TKB ;: CLEAR DONE FLAG
        MOV $100,$TKS ;: ENABLE TTY KEYBOARD INTERRUPT
        RTS PC ;: RETURN TO CALLER

; *TK SERVICE ROUTINE
; *THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
; *BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
; *IT IN THE QUEUE.
; *IF THE CHARACTER IS A "CONTROL-C" (1C) $TKINT IS CALLED AND
; *UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (START2)
$TKSRV: MOVB $TKB,-(SP) ;: PICKUP THE CHARACTER
        BIC #1C177,(SP) ;: STRIP THE JUNK
        CMP (SP),#3 ;: IS IT A CONTROL C?
        BNE 1$ ;: BRANCH IF NO
        TYPE ,SCNTLC ;: TYPE A CONTROL-C (1C)
        JSR PC,$TKINT ;: INIT THE KEYBOARD
        TST (SP)+ ;: CLEAN UP STACK
        JMP START2 ;: CONTROL C RESTART
1$: CMP (SP),#7 ;: IS IT A CONTROL G?
    BNE 2$ ;: BRANCH IF NO
    CMP #SWREG,SWR ;: IS SOFT-SWR SELECTED?
    BEQ 6$ ;: GO TO SWR CHANGE
2$: CMP #2,$TKCNT ;: IS THE QUEUE FULL?
    BNE 3$ ;: BRANCH IF NO

```



```

4497 023520 104401 001210          TYPE      $BELL          ;; RING THE TTY BELL
4498 023524 005726          TST        (SP)+          ;; CLEAN CHARACTER OFF OF STACK
4499 023526 000451          BR         5$             ;; EXIT
4500 023530 021627 000023          3$: CMP      (SP), #23      ;; IS IT A CONTROL-S?
4501 023534 001021          BNE       32$            ;; BRANCH IF NO
4502 023536 005077 155402          CLR       @STKS          ;; DISABLE TTY KEYBOARD INTERRUPTS
4503 023542 005726          TST        (SP)+          ;; CLEAN CHAR OFF STACK
4504 023544 105777 155374          31$: TSTB     @STKS          ;; WAIT FOR A CHAR
4505 023550 100375          BPL       31$            ;; LOOP UNTIL ITS THERE
4506 023552 117746 155370          MOVB     @STKB, -(SP)     ;; GET THE CHARACTER
4507 023556 042716 177600          BIC      #1C17?, (SP)    ;; MAKE IT 7-BIT ASCII
4508 023562 022627 000021          CMP      (SP)+, #21      ;; IS IT A CONTROL-Q?
4509 023566 001366          BNE       31$            ;; BRANCH IF NO
4510 023570 012777 000100 155346          MOV      #100, @STKS     ;; REENABLE TTY KEYBOARD INTERRUPTS
4511 023576 000002          RTI                          ;; RETURN
4512 023600 005237 023356          32$: INC      STKCNT       ;; COUNT THIS CHARACTER
4513 023604 021627 000140          CMP      (SP), #140      ;; IS IT UPPER CASE?
4514 023610 002405          BLT      4$             ;; BRANCH IF YES
4515 023612 021627 000175          CMP      (SP), #175      ;; IS IT A SPECIAL CHAR?
4516 023616 003002          BGT      4$             ;; BRANCH IF YES
4517 023620 042716 000040          BIC      #40 (SP)        ;; MAKE IT UPPER CASE
4518 023624 112677 177530          4$: MOVB     (SP)+, @STKQIN ;; AND PUT IT IN QUEUE
4519 023630 005237 023360          INC      STKQIN          ;; UPDATE THE POINTER
4520 023634 023727 023360 023366          CMP      STKQIN, #STKGEND ;; GO OFF THE END?
4521 023642 001003          BNE       5$             ;; BRANCH IF NO
4522 023644 012737 023364 023360          MOV      @STKQSR, STKQIN ;; RESET THE POINTER
4523 023652 000002          5$: RTI                          ;; RETURN

```

```

4524
4525 *****
4526 *SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
4527 *ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
4528 *SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP.
4529 *CALL WHEN OPERATING IN TTY INTERRUPT MODE.

```

```

4530 023654 022737 000176 001140 $CKSWR: CMP      #SWREG, SWR      ;; IS THE SOFT-SWR SELECTED
4531 023662 001124          BNE       15$            ;; EXIT IF NOT
4532 023664 105777 155254          TSTB     @STKS          ;; IS A CHAR WAITING?
4533 023670 100121          BPL       15$            ;; IF NOT, EXIT
4534 023672 117746 155250          MOVB     @STKB, -(SP)    ;; YES
4535 023676 042716 177600          BIC      #1C17?, (SP)    ;; MAKE IT 7-BIT ASCII
4536 023702 021627 000007          CMP      (SP), #7        ;; IS IT A CONTROL-G?
4537 023706 001300          BNE       2$             ;; IF NOT, PUT IT IN THE TTY QUEUE
4538
4539
4540
4541 *****
4542 *CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
4543 *ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
4544 *CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.

```

```

4544 023710 123727 001134 000001 6$: CMPB     $AUTOB, #1      ;; ARE WE RUNNING IN AUTO-MODE?
4545 023716 001674          BEQ      2$             ;; BRANCH IF YES
4546 023720 005726          TST        (SP)+          ;; CLEAR CONTROL-G OFF STACK
4547 023722 004737 023366          JSR      PC, $TKINT      ;; FLUSH THE TTY INPUT QUEUE
4548 023726 005077 155212          CLR       @STKS          ;; DISABLE TTY KEYBOARD INTERRUPTS
4549 023732 112737 000001 001135          MOVB     #1, $INTAG      ;; SET INTERRUPT MODE INDICATOR
4550
4551 023740 104401 024600          TYPE     , $CNTLG        ;; ECHO THE CONTROL-G (↑G)
4552 023744 104401 024605          $GTSWR: TYPE     , $MSWR   ;; TYPE CURRENT CONTENTS

```


4553	023750	013746	000176		MOV	SWREG, -(SP)	:: SAVE SWREG FOR TYPEOUT
4554	023754	104402			TYPOC		:: GO TYPE--OCTAL ASCII(ALL DIGITS)
4555	023756	104401	024616		TYPE	, SMNEW	:: PROMPT FOR NEW SWR
4556	023762	005046		19\$:	CLR	-(SP)	:: CLEAR COUNTER
4557	023764	005046			CLR	-(SP)	:: THE NEW SWR
4558	023766	105777	155152	7\$:	TSTB	2STKS	:: CHAR THERE?
4559	023772	100375			BPL	7\$:: IF NOT TRY AGAIN
4560							
4561	023774	117746	155146		MOVB	2STKB, -(SP)	:: PICK UP CHAR
4562	024000	042716	177600		BIC	#1C177, (SP)	:: MAKE IT 7-BIT ASCII
4563							
4564	024004	021627	000003		CMP	(SP), #3	:: IS IT A CONTROL-C?
4565	024010	001015			BNE	9\$:: BRANCH IF NOT
4566	024012	104401	024566		TYPE	, SCNTLC	:: YES, ECHO CONTROL-C (↑C)
4567	024016	062706	000006		ADD	#6, SP	:: CLEAN UP STACK
4568	024022	123727	001135	000001	CMPB	\$INTAG, #1	:: REENABLE TTY KEYBOARD INTERRUPTS?
4569	024030	001003			BNE	8\$:: BRANCH IF NO
4570	024032	012777	000100	155104	MOV	#100, 2STKS	:: ALLOW TTY KEYBOARD INTERRUPTS
4571	024040	000137	004762	8\$:	JMP	START2	:: CONTROL-C RESTART
4572							
4573							
4574	024044	021627	000025	9\$:	CMP	(SP), #25	:: IS IT A CONTROL-U?
4575	024050	001005			BNE	10\$:: BRANCH IF NOT
4576	024052	104401	024573		TYPE	, SCNTLU	:: YES, ECHO CONTROL-U (↑U)
4577	024056	062706	000006	20\$:	ADD	#6, SP	:: IGNORE PREVIOUS INPUT
4578	024062	000737			BR	19\$:: LET'S TRY IT AGAIN
4579							
4580							
4581	024064	021627	000015	10\$:	CMP	(SP), #15	:: IS IT A <CR>?
4582	024070	001022			BNE	16\$:: BRANCH IF NO
4583	024072	005766	000004		TST	4(SP)	:: YES, IS IT THE FIRST CHAR?
4584	024076	001403			BEQ	11\$:: BRANCH IF YES
4585	024100	016677	000002	155032	MOV	2(SP), 2SWR	:: SAVE NEW SWR
4586	024106	062706	000006	11\$:	ADD	#6, SP	:: CLEAN UP STACK
4587	024112	104401	001215	14\$:	TYPE	, SCRLF	:: ECHO <CR> AND <LF>
4588	024116	123727	001135	000001	CMPB	\$INTAG, #1	:: RE-ENABLE TTY KBD INTERRUPTS?
4589	024124	001003			BNE	15\$:: BRANCH IF NOT
4590	024126	012777	000100	155010	MOV	#100, 2STKS	:: RE-ENABLE TTY KBD INTERRUPTS
4591	024134	000002		15\$:	RTI		:: RETURN
4592	024136	004737	022634	16\$:	JSR	PC, STYPEC	:: ECHO CHAR
4593	024142	021627	000060		CMP	(SP), #60	:: CHAR < 0?
4594	024146	002420			BLT	18\$:: BRANCH IF YES
4595	024150	021627	000067		CMP	(SP), #67	:: CHAR > 7?
4596	024154	003015			BGT	18\$:: BRANCH IF YES
4597	024156	042726	000060		BIC	#60, (SP)+	:: STRIP-OFF ASCII
4598	024162	005766	000002		TST	2(SP)	:: IS THIS THE FIRST CHAR
4599	024166	001403			BEQ	17\$:: BRANCH IF YES
4600	024170	006316			ASL	(SP)	:: NO, SHIFT PRESENT
4601	024172	006316			ASL	(SP)	:: CHAR OVER TO MAKE
4602	024174	006316			ASL	(SP)	:: ROOM FOR NEW ONE.
4603	024176	005266	000002	17\$:	INC	2(SP)	:: KEEP COUNT OF CHAR
4604	024202	056616	177776		BIS	-2(SP), (SP)	:: SET IN NEW CHAR
4605	024206	000667			BR	7\$:: GET THE NEXT ONE
4606	024210	104401	001214	18\$:	TYPE	, \$QUES	:: TYPE ?<CR><LF>
4607	024214	000720			BR	20\$:: SIMULATE CONTROL-U
4608					.DSABL	LSB	

4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662
4663
4664

024216 011646
024220 016666 000004 000002
024226 005066 000004
024232 005046
024234 012746 024242
024240 000002
024242
024242 005737 023356
024246 001775
024250 005337 023356
024254 117766 177102 000004
024262 005237 023362
024266 023727 023362 023366
024274 001003
024276 012737 023364 023362
024304 000002
024306 010346
024310 005046
024312 012703 024542
024316 022703 024566
024322 101456
024324 104410
024326 112613
024330 122713 000177
024334 001022
024336 005716
024340 001007
024342 112737 000134 024540
024350 104401 024540
024354 012716 177777
024360 005303
024362 020327 024542
024366 103434
024370 111337 024540
024374 104401 024540
024400 000746
024402 005716
024404 001406
024406 112737 000134 024540

```
*****
: THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
: CALL:
: * RDCHR ; GET A CHARACTER FROM THE QUEUE
: * RETURN HERE ; CHARACTER IS ON THE STACK
: * ; WITH PARITY BIT STRIPPED OFF
:
$RDCHR: MOV (SP), -(SP) ; PUSH DOWN THE PC AND
MOV 4(SP), 2(SP) ; THE PS
CLR 4(SP) ; GET READY FOR A CHARACTER
CLR -(SP) ; PUT NEW PS ON STACK
MOV #64$, -(SP) ; PUT NEW PC ON STACK
RTI ; POP NEW PC AND PS

64$:
1$: TST $STKCNT ; WAIT ON A CHARACTER
BEQ 1$
DEC $STKCNT ; DECREMENT THE COUNTER
MOVB $STKQOUT, 4(SP) ; GET ONE CHARACTER
INC $STKQOUT ; UPDATE THE POINTER
CMP $STKQOUT, $STKGEND ; DID IT GO OFF OF THE END?
BNE 2$ ; BRANCH IF NO
MOV $STKQRT, $STKQOUT ; RESET THE POINTER
RTI ; RETURN

2$:
*****
: THIS ROUTINE WILL INPUT A STRING FROM THE TTY
: CALL:
: * RDLIN ; INPUT A STRING FROM THE TTY
: * RETURN HERE ; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
: * ; TERMINATOR WILL BE A BYTE OF ALL 0'S
:
$RDLIN: MOV R3, -(SP) ; SAVE R3
CLR -(SP) ; CLEAR THE RUBOUT KEY
1$: MOV $STTYIN, R3 ; GET ADDRESS
2$: CMP $STTYIN+20., R3 ; BUFFER FULL?
BLOS 4$ ; BR IF YES
RDCHR ; GO READ ONE CHARACTER FROM THE TTY
MOVB (SP)+, (R3) ; GET CHARACTER
10$: CMPB #177, (R3) ; IS IT A RUBOUT
BNE 5$ ; BR IF NO
TST (SP) ; IS THIS THE FIRST RUBOUT?
BNE 6$ ; BR IF NO
MOVB #' \, 9$ ; TYPE A BACK SLASH
TYPE 9$
6$: MOV #-1, (SP) ; SET THE RUBOUT KEY
DEC R3 ; BACKUP BY ONE
CMP R3, $STTYIN ; STACK EMPTY?
BLO 4$ ; BR IF YES
MOVB (R3), 9$ ; SETUP TO TYPEOUT THE DELETED CHAR.
TYPE 9$ ; GO TYPE
BR 2$ ; GO READ ANOTHER CHAR.
5$: TST (SP) ; RUBOUT KEY SET?
BEQ 7$ ; BR IF NO
MOVB #' \, 9$ ; TYPE A BACK SLASH
```



```

4665 024414 104401 024540          TYPE          9$
4666 024420 005016          CLR            (SP)
4667 024422 122713 000025      7$: CMPB       #25,(R3)
4668 024426 001003          BNE           8$
4669 024430 104401 024573      TYPE          $CNTLU
4670 024434 000726          BR            1$
4671 024436 122713 000022      8$: CMPB       #22,(R3)
4672 024442 001011          BNE           3$
4673 024444 105013          CLRB         (R3)
4674 024446 104401 001215      TYPE          $CRLF
4675 024452 104401 024542      TYPE          $TTYIN
4676 024456 000717          BR            2$
4677 024460 104401 001214      4$: TYPE          $QUES
4678 024464 000712          BR            1$
4679 024466 111337 024540      3$: MOVB       (R3),9$
4680 024472 104401 024540      TYPE          9$
4681 024476 122723 000015      CMPB         #15,(R3)+
4682 024502 001305          BNE           2$
4683 024504 105063 177777      CLRB         -1(R3)
4684 024510 104401 001216      TYPE          $LF
4685 024514 005726          TST         (SP)+
4686 024516 012603          MOV         (SP)+,R3
4687 024520 011646          MOV         (SP)-,(SP)
4688 024522 016666 000004 000002      MOV         4(SP),2(SP)
4689 024530 012766 024542 000004      MOV         $TTYIN,4(SP)
4690 024536 000002          RTI
4691 024540 000          9$: .BYTE      0
4692 024541 000          .BYTE      0
4693 024542 000024          $TTYIN: .BLKB  20.
4694 024566 041536 005015 000  $CNTLC: .ASCIZ /?C/<15><12>
4695 024573 136 006525 000012  $CNTLU: .ASCIZ /?U/<15><12>
4696 024600 043536 005015 000  $CNTLG: .ASCIZ /?G/<15><12>
4697 024605 015 051412 051127  $MSWR: .ASCIZ <15><12>/SWR = /
4698 024612 036440 000040
4699 024616 020040 042516 020127  $MNEW: .ASCIZ / NEW = /
4700 024624 020075 000
4701 024630          .EVEN
4702
4703          .SBTTL SCOPE HANDLER ROUTINE
4704
4705          ;*****
4706          ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
4707          ;*AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
4708          ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
4709          ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4710          ;*SW14=1 LOOP ON TEST
4711          ;*SW11=1 INHIBIT ITERATIONS
4712          ;*SW09=1 LOOP ON ERROR
4713          ;*CALL
4714          ;* SCOPE ;:SCOPE=IOT
4715
4716          $SCOPE:
4717 024630 104407          CKSWR
4718 024632 032777 040000 154300 1$: BIT        #BIT14,$SWR ;:TEST FOR CHANGE IN SOFT-SWR
4719 024640 001101          BNE         $OVER ;:LOOP ON PRESENT TEST?
4720          ;:YES IF SW14=1
          ;*****START OF CODE FOR THE XOR TESTER*****

```

```

4721 024642 000416 $XTSTR: BR 6$ ;: IF RUNNING ON THE "XOR" TESTER CHANGE
4722 ;: THIS INSTRUCTION TO A "NOP" (NOP=240)
4723 024644 013746 000004 MOV @#ERRVEC, -(SP) ;: SAVE THE CONTENTS OF THE ERROR VECTOR
4724 024650 012737 024670 000004 MOV #5$, @#ERRVEC ;: SET FOR TIMEOUT
4725 024656 005737 177060 TST @#177060 ;: TIME OUT ON XOR?
4726 024662 012637 000004 MOV (SP)+, @#ERRVEC ;: RESTORE THE ERROR VECTOR
4727 024666 000450 BR $SVLAD ;: GO TO THE NEXT TEST
4728 024670 022626 5$: CMP (SP)+, (SP)+ ;: CLEAR THE STACK AFTER A TIME OUT
4729 024672 012637 000004 MOV (SP)+, @#ERRVEC ;: RESTORE THE ERROR VECTOR
4730 024676 000413 BR 7$ ;: LOOP ON THE PRESENT TEST
4731 024700 6$: ; #####END OF CODE FOR THE XOR TESTER#####
4732 024700 105737 001103 2$: TSTB $ERFLG ;: HAS AN ERROR OCCURRED?
4733 024704 001421 BEQ 3$ ;: BR IF NO
4734 024706 123737 001115 001103 CMPB $ERMAX, $ERFLG ;: MAX. ERRORS FOR THIS TEST OCCURRED?
4735 024714 101015 BHI 3$ ;: BR IF NO
4736 024716 032777 001000 154214 BIT #BIT09, @SWR ;: LOOP ON ERROR?
4737 024724 001404 BEQ 4$ ;: BR IF NO
4738 024726 013737 001110 001106 7$: MOV $LPERR, $LPADR ;: SET LOOP ADDRESS TO LAST SCOPE
4739 024734 000443 BR $OVER
4740 024736 105037 001103 4$: CLRB $ERFLG ;: ZERO THE ERROR FLAG
4741 024742 005037 001204 CLR $TIMES ;: CLEAR THE NUMBER OF ITERATIONS TO MAKE
4742 024746 000412 BR 1$ ;: ESCAPE TO THE NEXT TEST
4743 024750 032777 004000 154162 3$: BIT #BIT11, @SWR ;: INHIBIT ITERATIONS?
4744 024756 001006 BNE 1$ ;: BR IF YES
4745 024760 005237 001104 INC $ICNT ;: INCREMENT ITERATION COUNT
4746 024764 023737 001204 001104 CMP $TIMES, $ICNT ;: CHECK THE NUMBER OF ITERATIONS MADE
4747 024772 002024 BGE $OVER ;: BR IF MORE ITERATION REQUIRED
4748 024774 012737 000001 001104 1$: MOV #1, $ICNT ;: REINITIALIZE THE ITERATION COUNTER
4749 025002 013737 025060 001204 MOV $MXCNT, $TIMES ;: SET NUMBER OF ITERATIONS TO DO
4750 025010 105237 001102 $SVLAD: INCB $STNM ;: COUNT TEST NUMBERS
4751 025014 113737 001102 001224 MOVB $STNM, $STNM ;: SET TEST NUMBER IN APT MAILBOX
4752 025022 011637 001106 MOV (SP), $LPADR ;: SAVE SCOPE LOOP ADDRESS
4753 025026 011637 001110 MOV (SP), $LPERR ;: SAVE ERROR LOOP ADDRESS
4754 025032 005037 001206 CLR $ESCAPE ;: CLEAR THE ESCAPE FROM ERROR ADDRESS
4755 025036 112737 000001 001115 MOVB #1, $ERMAX ;: ONLY ALLOW ONE(1) ERROR ON NEXT TEST
4756 025044 013777 001102 154070 $OVER: MOV $STNM, @DISPLAY ;: DISPLAY TEST NUMBER
4757 025052 013716 001106 MOV $LPADR, (SP) ;: FUDGE RETURN ADDRESS
4758 025056 000002 RTI ;: FIXES PS
4759 025060 000001 $MXCNT: 1 ;: MAX. NUMBER OF ITERATIONS

```

```

4760
4761 .SBTTL SAVE AND RESTORE R0-R5 ROUTINES
4762
4763 ;: *****
4764 ;: *SAVE R0-R5
4765 ;: *CALL:
4766 ;: * SAVREG
4767 ;: *UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
4768 ;: *
4769 ;: *TOP---(+16)
4770 ;: * +2---(+18)
4771 ;: * +4---R5
4772 ;: * +6---R4
4773 ;: * +8---R3
4774 ;: * +10---R2
4775 ;: * +12---R1
4776 ;: * +14---R0

```


4777
4778 025062
4779 025062 010046
4780 025064 010146
4781 025066 010246
4782 025070 010346
4783 025072 010446
4784 025074 010546
4785 025076 016646 000022
4786 025102 016646 000022
4787 025106 016646 000022
4788 025112 016646 000022
4789 025116 000002

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

4790
4791
4792
4793
4794 025120
4795 025120 012666 000022
4796 025124 012666 000022
4797 025130 012666 000022
4798 025134 012666 000022
4799 025140 012605
4800 025142 012604
4801 025144 012603
4802 025146 012602
4803 025150 012601
4804 025152 012600
4805 025154 000002

```
;;*RESTORE RO-R5
;;*CALL:
;;*RESREG
$RESREG:
MOV (SP)+,22(SP) ;;RESTORE PC OF CALL
MOV (SP)+,22(SP) ;;RESTORE PS OF CALL
MOV (SP)+,22(SP) ;;RESTORE PC OF MAIN FLOW
MOV (SP)+,22(SP) ;;RESTORE PS OF MAIN FLOW
MOV (SP)+,R5 ;;POP STACK INTO R5
MOV (SP)+,R4 ;;POP STACK INTO R4
MOV (SP)+,R3 ;;POP STACK INTO R3
MOV (SP)+,R2 ;;POP STACK INTO R2
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,RO ;;POP STACK INTO RO
RTI
```

4806
4807
4808
4809
4810
4811
4812
4813
4814

```
.SBTTL TRAP DECODER
;*****
;THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
;AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
;OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
;GO TO THAT ROUTINE.
```

4815 025156 010046
4816 025160 016600 000002
4817 025164 005740
4818 025166 111000
4819 025170 006300
4820 025172 016000 025212
4821 025176 000200

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

4822
4823
4824
4825
4826 025200 011646
4827 025202 016666 000004 000002
4828 025210 000002

```
;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
$TRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN
MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN
RTI ;;RESTORE THE PSW
```

4829
4830
4831
4832

```
.SBTTL TRAP TABLE
;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
```

```

4833 ;*BY THE "TRAP" INSTRUCTION.
4834 ;
4835 ; ROUTINE
4836 ;-----
4837 025212 025200 $TRPAD: .WORD $TRAP2
4838 025214 022422 $TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
4839 025216 022730 $TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
4840 025220 022704 $TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
4841 025222 022744 $TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
4842 025224 023132 $TYPDS ;;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
4843
4844 025226 023744 $GTSWR ;;CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING
4845
4846 025230 023654 $CKSWR ;;CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
4847 025232 024216 $RDCHR ;;CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
4848 025234 024306 $RDLIN ;;CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE
4849 025236 025062 $$SAVREG ;;CALL=SAVREG TRAP+12(104412) SAVE R0-R5 ROUTINE
4850 025240 025120 $RESREG ;;CALL=RESREG TRAP+13(104413) RESTORE R0-R5 ROUTINE
4851
4852 .SBTTL SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE
4853
4854 ;*****
4855 ;*THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
4856 ;*UNSIGNED DECIMAL ASCIZ NUMBER.
4857 ;*CALL
4858 ;* MOV NUMBER, -(SP) ;;PUT BINARY NUMBER ON THE STACK
4859 ;* JSR PC, @#$SB2D ;;CALL
4860 ;* RETURN ;;ADDRESS OF THE 1ST ASCIZ CHAR. IS ON THE STACK
4861
4862
4863 025242 016637 000002 025272 $SB2D: MOV 2(SP), 1$ ;;SAVE BINARY NUMBER
4864 025250 012746 025272 MOV #1$, -(SP) ;;SET POINTER
4865 025254 004737 025276 JSR PC, @#$DB2D ;;CALL DOUBLE LENGTH CONVERT
4866 025260 062716 000005 ADD #5, (SP) ;;ONLY ALLOW FIVE CHARACTERS
4867 025264 012666 000002 MOV (SP)+, 2(SP) ;;PICKUP POINTER
4868 025270 000207 RTS PC ;;RETURN
4869 025272 000000 000000 1$: .WORD 0,0
4870
4871 .SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
4872
4873 ;*****
4874 ;*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
4875 ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
4876 ;*POSITIVE.
4877 ;*CALL
4878 ;* MOV #PNTR, -(SP) ;;POINTER TO LOW WORD OF BINARY NUMBER
4879 ;* JSR PC, @#$DB2D ;;CALL
4880 ;* RETURN ;;THE FIRST ADDRESS OF ASCIZ
4881 ;* ;;IS ON THE STACK
4882
4883
4884 025276 104412 $DB2D: SAVREG ;;SAVE REGISTERS
4885 025300 016602 000002 MOV 2(SP), R2 ;;PICKUP THE DATA POINTER
4886 025304 012700 025456 MOV #$DECVL, R0 ;;GET ADDRESS OF "$DECVL" STRING
4887 025310 010066 000002 MOV R0, 2(SP) ;;PUT ADDRESS OF ASCIZ STRING ON STACK
4888 025314 012201 MOV (R2)+, R1 ;;PICKUP THE BINARY NUMBER

```


H10

MD-11-DZRMF-B RM03/RM02 EXTENDED DRIVE TEST
CZRMFB.P11 21-NOV-77 14:10

MACY11 30(1046) 22-NOV-77 08:04 PAGE 94
DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

SEG 0124

```
4889 025316 012202          MOV      (R2)+,R2
4890 025320 012737 000012 025374  MOV      #10,4$          ;; SET UP TO DO 10 CONVERSIONS
4891 025326 012704 025406  MOV      #STNPWR,R4      ;; ADDRESS OF TEN POWER
4892 025332 012705 025410  MOV      #STNPWR+2,R5
4893 025336 005003          1$: CLR      R3          ;; CLEAR PARTIAL
4894 025340 161401          2$: SUB      (R4),R1      ;; SUBTRACT TEN POWER
4895 025342 005602          SBC      R2
4896 025344 161502          SUB      (R5),R2
4897 025346 002402          BLT      3$          ;; BR IF TEN POWER TO LARGE
4898 025350 005203          INC      R3          ;; ADD 1 TO PARTIAL
4899 025352 000772          BR       2$          ;; LOOP
4900 025354 062401          3$: ADD      (R4)+,R1      ;; RESTORE SUBTRACTED VALUE
4901 025356 005502          ADC      R2
4902 025360 062402          ADD      (R4)+,R2
4903 025362 022525          CMP      (R5)+,(R5)+  ;; MOVE TO NEXT TEN POWER
4904 025364 052703 000060  BIS      #0,R3          ;; CHANGE PARTIAL TO ASCII
4905 025370 110320          MOVB     R3,(R0)+      ;; SAVE IT
4906 025372 005327          DEC      (PC)+        ;; DONE?
4907 025374 000000          4$: .WORD    0
4908 025376 001357          BNE     1$          ;; BR IF NO
4909 025400 105020          CLRB    (R0)+        ;; TERMINATOR
4910 025402 104413          RESREG  ;; RESTORE REGISTERS
4911 025404 000207          RTS     PC          ;; RETURN
4912 025406 145000          $STNPWR: 145000      ;; 1.0E09
4913 025410 035632          35632
4914 025412 160400          160400      ;; 1.0E08
4915 025414 002765          2765
4916 025416 113200          113200      ;; 1.0E07
4917 025420 000230          230
4918 025422 041100          041100      ;; 1.0E06
4919 025424 000017          17
4920 025426 103240          103240      ;; 1.0E05
4921 025430 000001          1
4922 025432 023420          23420      ;; 1.0E04
4923 025434 000000          0
4924 025436 001750          1750      ;; 1.0E03
4925 025440 000000          0
4926 025442 000144          144      ;; 1.0E02
4927 025444 000000          0
4928 025446 000012          12      ;; 1.0E01
4929 025450 000000          0
4930 025452 000001          1      ;; 1.0E00
4931 025454 000000          0
4932 025456 000014          $DECVL: .BLKB 12.      ;; RESERVE STORAGE FOR ASCII STRING
4933
4934          .SBTTL TYPE NUMERICAL ASCII STRING SUPPRESS LEADING ZEROS
4935
4936          ;; *****
4937          ;; *THIS ROUTINE IS USED TO TYPE AN ASCII NUMBER SUPPRESSING THE
4938          ;; *LEADING NUMBERS.
4939          ;; *CALL
4940          ;; *      MOV      #NUMADR,-(SP)      ;; FIRST ADDRESS OF ASCII STRING
4941          ;; *      JSR      PC,@#$SUPRS
4942
4943          $SUPRS: MOV      RO,-(SP)      ;; SAVE RO
4944 025472 010046
```

4945 025474 016600 000004
4946 025500 105710
4947 025502 001403
4948 025504 122720 000060
4949 025510 001773
4950 025512 005300
4951 025514 010037 025522
4952 025520 104401
4953 025522 000000
4954 025524 012600
4955 025526 012616
4956 025530 000207

MOV 4(SP),RO ;: PICKUP THE POINTER
1\$: TSTB (RO) ;: TERMINATE OR?
BEQ 2\$;: BR IF YES
CMPB #'0,(RO)+ ;: IS THIS AN ASCII "0" ?
BEQ 1\$;: BR IF YES
2\$: DEC RO ;: BACKUP BY "1"
MOV RO,3\$;: SAVE FOR TYPING
TYPE ;: GO TYPE
3\$: .WORD 0 ;: ASCIZ POINTER GOES HERE
MOV (SP)+,RO ;: RESTORE RO
MOV (SP)+,(SP) ;: RESTORE THE STACK
RTS PC ;: RETURN

.SBTTL RANDOM NUMBER GENERATOR ROUTINE

;: THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
;: WITH A RANGE OF 0 TO 2(+33)-1.

CALL: ;: CALL THE ROUTINE
* JSR PC,\$RAND ;: RETURN HERE THE RANDOM
* RETURN ;: NUMBER WILL BE IN
* ;: \$HINUM,\$LONUM
*

\$RAND: MOV RO,-(SP) ;: PUSH RO ON STACK
MOV R1,-(SP) ;: PUSH R1 ON STACK
MOV R2,-(SP) ;: PUSH R2 ON STACK
MOV \$LONUM,RO ;: SET RO WITH LOW
MOV \$HINUM,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 \$LONUM,RO ;: ADD NUMBER TO MAKE X 129
ADC R1 ;: PROPOGATE CARRY
ADD \$HINUM,R1 ;: ADD NUMBER TO MAKE X 129
ADD #1057,RO ;: ADD LOW CONSTANT
ADC R1 ;: PROPOGATE CARRY
ADD #47401,R1 ;: ADD HIGH CONSTANT
MOV RO,\$LONUM ;: SAVE RO
MOV R1,\$HINUM ;: 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

\$HINUM: .WORD 176543
\$LONUM: .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.

4957
4958
4959
4960
4961
4962
4963
4964
4965
4966
4967
4968
4969 025532
4970 025532 010046
4971 025534 010146
4972 025536 010246
4973 025540 013700 025632
4974 025544 013701 025630
4975 025550 012702 177771
4976 025554 006300
4977 025556 006101
4978 025560 005202
4979 025562 001374
4980 025564 063700 025632
4981 025570 005501
4982 025572 063701 025630
4983 025576 062700 001057
4984 025602 005501
4985 025604 062701 047401
4986 025610 010037 025632
4987 025614 010137 025630
4988 025620 012602
4989 025622 012601
4990 025624 012600
4991 025626 000207
4992 025630 176543
4993 025632 123456
4994
4995
4996
4997
4998
4999
5000


```

5001 ;*DIVISION WILL BE PERFORMED SO THAT THE REMAINDER IS OF THE
5002 ;*SAVE SIGN AS THE DIVIDEND.
5003 ;*CALL:
5004 ;*      MOV      LOW DIVIDEND,-(SP) ;;THE HIGH DIVIDEND MUST BE < 1/2
5005 ;*      MOV      HIGH DIVIDEND,-(SP); AS LARGE AS THE DIVISOR
5006 ;*      MOV      DIVISOR,-(SP)
5007 ;*      JSR      PC,$DIV
5008 ;*      RETURN   ;;QUOTIENT & REMAINDER ARE ON THE STACK
5009 ;*      "V"=0    IMPLIES NO ERROR
5010 ;*      "V"=1    IMPLIES ERROR OCCURRED
5011 ;*      "C"=0    DIVIDE OVERFLOW OCCURRED
5012 ;*      "C"=1    ATTEMPTED TO DIVIDE BY ZERO
5013
5014
5015 ;*      STACK   NO ERROR      OVERFLOW      DIVIDE BY ZERO
5016 ;*      -----
5017 ;*      TOP     REMAINDER    ALL ZEROS    ALL ONES
5018 ;*      +2     QUOTIENT     ALL ZEROS    ALL ONES
5019
5020 $DIV:
5021 025634 104400 TRAP ;;PUSH OLD PSW AND PC ON STACK
5022 025636 042716 000017 BIC #17,(SP) ;;STRIP AWAY CONDITION CODES
5023 025642 010046 MOV RO,-(SP) ;;PUSH RO ON STACK
5024 025644 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
5025 025646 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
5026 025650 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
5027 025652 005046 CLR -(SP) ;;SAVE A PLACE FOR SIGNS
5028 025654 012746 000021 MOV #17,-(SP) ;;SETUP THE ITERATION COUNTER
5029 025660 016601 000024 MOV 24(SP),R1 ;;PICKUP THE DIVIDEND
5030 025664 016600 000022 MOV 22(SP),RO
5031 025670 100005 BPL 1$ ;;CHECK THE SIGN
5032 025672 105366 000003 DECB 3(SP) ;;KEEP TRACK OF THE SIGN
5033 025676 005400 NEG RO ;;AND NEGATE THE ORIGINAL
5034 025700 005401 NEG R1 ;;NUMBER
5035 025702 005600 SBC RO
5036 025704 016602 000020 1$: MOV 20(SP),R2 ;;PICKUP THE DIVISOR
5037 025710 002407 BLT 2$ ;;CHECK THE SIGN
5038 025712 003011 BGT 3$ ;;DIVISOR OF 0 IS A NO-NO
5039 025714 052766 000003 000014 BIS #3,14(SP) ;;SET "V" & "C"
5040 025722 012700 177777 MOV #-1,RO ;;SET REMAINDER TO ALL ONES
5041 025726 000424 BR 7$ ;;EXIT
5042 025730 005266 000002 2$: INC 2(SP) ;;KEEP TRACK OF DIVISORS SIGN
5043 025734 000401 BR 4$
5044 025736 005402 3$: NEG R2 ;;NEGATE THE ORIGINAL NUMBER
5045 025740 000241 4$: CLC ;;CLEAR "C"
5046 025742 000405 BR 6$ ;;START FORMING QUOTIENT
5047 025744 006100 5$: ROL RO ;;POSITION MSB'S
5048 025746 010003 MOV RO,R3 ;;COPY
5049 025750 060203 ADD R2,R3 ;;COMPARE DIVIDEND & DIVISOR
5050 025752 103001 BCC 6$ ;;BR IF DIVIDEND > DIVISOR
5051 025754 010300 MOV R3,RO ;;REMAINDER AFTER THIS LOOP
5052 025756 006101 6$: ROL R1 ;;QUOTIENT BIT ENTERS HERE
5053 025760 005316 DEC (SP) ;;DONE?
5054 025762 001370 BNE 5$ ;;BR IF NO
5055 025764 005701 TST R1 ;;OVERFLOW?
5056 025766 100005 BPL 8$ ;;BR IF NO

```

```

5057 025770 052766 000002 000014 BIS #2,14(SP) ;:SET "V" IN RETURN STATUS WORD
5058 025776 005000 CLR R0 ;:SET REMAINDER TO ALL ZEROS
5059 026000 010001 7$: MOV R0,R1 ;:COPY REMAINDER INTO QUOTIENT
5060 026002 005726 8$: TST (SP)+ ;:CLEAR COUNTER FROM STACK
5061 026004 005716 TST (SP) ;:REMAINDER SIGN CORRECTION NEEDED?
5062 026006 002004 BGE 9$ ;:BR IF NO
5063 026010 005400 NEG R0 ;:NEGATE REMAINDER
5064 026012 105066 000001 CLR R1 ;:CLEAR SIGN
5065 026016 005316 DEC (SP) ;:BUT DON'T FORGET QUOTIENT
5066 026020 005726 9$: TST (SP)+ ;:QUOTIENT SIGN CORRECTION NEEDED?
5067 026022 001401 BEQ 10$ ;:BR IF NO
5068 026024 005401 NEG R1 ;:NEGATE QUOTIENT
5069 026026 010166 000020 10$: MOV R1,20(SP) ;:RETURN QUOTIENT AND
5070 026032 010066 000016 MOV R0,16(SP) ;:REMAINDER TO USER
5071 026036 012603 MOV (SP)+,R3 ;:POP STACK INTO R3
5072 026040 012602 MOV (SP)+,R2 ;:POP STACK INTO R2
5073 026042 012601 MOV (SP)+,R1 ;:POP STACK INTO R1
5074 026044 012600 MOV (SP)+,R0 ;:POP STACK INTO R0
5075 026046 012666 000002 MOV (SP)+,2(SP) ;:SETUP TO RETURN CONDITION CODES
5076 026052 000002 RTI ;:RETURN

```

.SBTTL *** PROGRAM SUBROUTINES ***

```

5077
5078 ;:SET "LPTAVL" TO THE PROPER STATE.
5079 ;:LPTAVL = 0 IF NO LINE PRINTER AVAILABLE
5080 ;:LPTAVL = 1 IF LINE PRINTER IS AVAILABLE
5081 ;:CALL
5082 ;:
5083 ;: JSR PC, @#LP.AVL
5084 ;:
5085 ;:
5086 ;:
5087 026054 005037 001310 LP.AVL: CLR @#LPTAVL ;:START WITH NO PRINTER AVAIABLE
5088 026060 012737 026104 000004 MOV #1$, @#ERRVEC ;:SETUP THE TIMEOUT VECTOR
5089 026066 005037 000006 CLR @#ERRVEC+2
5090 026072 005777 153404 TST @LPS ;:IS THERE A LINE PRINTER?
5091 026076 005237 001310 INC @#LPTAVL ;:YES--SET AVAILABLE SWITCH
5092 026102 000401 BR 2$
5093 026104 022626 1$: CMP (SP)+,(SP)+ ;:NO--POP STACK
5094 026106 012737 000006 2$: MOV @#ERRVEC+2,@#ERRVEC ;:RESTORE TIMEOUT VECTOR
5095 026114 000207 RTS PC ;:RETURN

```

;:THIS ROUTINE WILL DETERMINE IF THERE IS A CLOCK ON THE SYSTEM
AND IF THERE IS IT WILL SETUP THE VECTOR AND START THE CLOCK

```

5096 ;:
5097 ;:"CLKSTA" WILL INDICATE THE CLOCK TYPE
5098 ;:0= NO CLOCK
5099 ;:+1= KW11-P
5100 ;:-1= KW11-L
5101 ;:THIS ROUTINE WILL ALSO SETUP "TICKMS" (TIME
5102 ;:PER CLOCK TICK IN MILLISECONDS) AND "TICKUS"
5103 ;:(TIME PER CLOCK TICK IN MICROSECONDS) AS
5104 ;:PER SW00.
5105 ;:SW00=0 -- 60HZ
5106 ;:SW00=1 -- 50HZ
5107 ;:CALL
5108 ;: JSR PC, @#ST.CLK
5109 ;:
5110 ;:
5111 ;:
5112 ;:

```



```

5113 026116 010146          ST.CLK: MOV      R1, -(SP)          ;SAVE R1
5114 026120 012701 000006  MOV      #ERRVEC+2, R1      ;SAVE AND SETUP TIMEOUT VECTOR
5115 026124 011146          MOV      (R1), -(SP)
5116 026126 005011          CLR      (R1)              ;LEVEL 0
5117 026130 014146          MOV      -(R1), -(SP)
5118 026132 012711 026162  MOV      #1$, (R1)         ;GO TO 1$ ON TIMEOUT
5119 026136 005037 001324  CLR      CLKSTA           ;SET CLOCK STATUS TO NO CLOCK
5120 026142 005777 153314  TST     @PKCS             ;IS THERE A KW11-P?
5121 026146 012737 000001 001324  MOV      #1, CLKSTA       ;YES--SET STATUS TO KW11-P
5122 026154 004737 026264  JSR     PC, ST.PCLK       ;START THE KW11-P
5123 026160 000414          BR      3$                ;GO TO EXIT
5124 026162 022626          1$:  CMP     (SP)+, (SP)+     ;CLEAN UP THE STACK
5125 026164 012711 026210  MOV      #2$, (R1)         ;IF TIMEOUT GO TO 2$
5126 026170 005777 153300  TST     @LKS              ;IS THERE A KW11-L?
5127 026174 012737 177777 001324  MOV      #-1, CLKSTA      ;YES-- SET STATUS TO KW11-L
5128 026202 004737 026326  JSR     PC, ST.LCLK       ;START THE KW11-L
5129 026206 000401          BR      3$                ;EXIT
5130 026210 022626          2$:  CMP     (SP)+, (SP)+     ;CLEAN UP THE STACK
5131 026212 012621 03$:  MOV      (SP)+, (R1)+     ;RESTORE THE TIMEOUT VECTOR
5132 026214 012621          MOV      (SP)+, (R1)+
5133 026216 012601          MOV      (SP)+, R1        ;RESTORE R1
5134 026220 032737 000100 001300  BIT     #SW06, @#C.SWR     ;50HZ OR 60HZ?
5135 026226 001407          BEQ     4$                ;BRANCH IF 50
5136 026230 012737 000020 001326  MOV      #20, @TICKMS      ;SETUP TIME PER
5137 026236 012737 047040 001330  MOV      #20000., @TICKUS ;TICK FOR 50HZ
5138 026244 000406          BR      5$
5139 026246 012737 000016 001326  4$:  MOV      #16, @TICKMS    ;SETUP TIME PER
5140 026254 012737 040432 001330  MOV      #16666., @TICKUS ;TICK FOR 60HZ
5141 026262 000207          5$:  RTS      PC              ;RETURN
5142
5143 026264          ST.PCLK:
5144 026264 032737 000040 001300  BIT     #SW05, @#C.SWR     ;ALLOW SOFTWARE TIMEOUTS?
5145 026272 001014          BNE     1$                ;NO--BRANCH
5146 026274 012777 026362 153154  MOV      #SRVCLK, @PKV     ;SETUP THE KW11-P VECTOR
5147 026302 012777 000300 153150  MOV      #300, @PKV+2
5148 026310 012777 000001 153146  MOV      #1, @PKB          ;COUNT ONE TICK
5149 026316 012777 000115 153136  MOV      #115, @PKCS      ;"INT.EN." COUNT DOWN" "MODE 1 (REPEAT)".
5150
5151 026324 000207          1$:  RTS      PC              ;RETURN
5152
5153 026326          ST.LCLK:
5154 026326 032737 000040 001300  BIT     #SW05, @#C.SWR     ;ALLOW SOFTWARE TIMEOUTS?
5155 026334 001011          BNE     1$                ;NO--BRANCH
5156 026336 012777 026362 153124  MOV      #SRVCLK, @LKV     ;SETUP THE KW11-L VECTOR
5157 026344 012777 000300 153120  MOV      #300, @LKV+2
5158 026352 012777 000100 153114  MOV      #100, @LKS       ;START THE KW11-L
5159 026360 000207          1$:  RTS      PC              ;RETURN
5160
5161 026362 013746 001326          SRVCLK: MOV      @TICKMS, -(SP) ;TIME PER TICK IN MILLISECONDS
5162 026366 004737 043150          JSR     PC, @RPTMR        ;COUNT THE ELAPSED TIME
5163 026372 000002          RTI                      ;RETURN AFTER INTERRUPT
5164
5165          ; THIS ROUTINE SETS UP DEFAULT PARAMETER VALUES WHEN THE PROGRAM IS
5166          ; STARTED OR WHEN THE VALUE OF BIT00 IN 'C.SWR' IS CHANGED.
5167          ; CALL
5168          ; JSR      PC, LODFLT
  
```


M10

```

; RETURN
S169 ;
S170 ;
S171 026374 LODFLT:
S172 026374 010046 MOV R0,-(SP) ;: PUSH R0 ON STACK
S173 026376 010146 MOV R1,-(SP) ;: PUSH R1 ON STACK
S174 026400 010246 MOV R2,-(SP) ;: PUSH R2 ON STACK
S175 026402 010346 MOV R3,-(SP) ;: PUSH R3 ON STACK
S176 026404 012737 176777 001314 MOV #176777,TSTNMS ;: SELECT TESTS 0-10, 12-17
S177 026412 012737 000003 001316 MOV #3,TSTNMS+2 ;: SELECT TESTS 20 & 21
S178 026420 012700 001746 MOV #DFLT,R0 ;: DEFAULT PARAMETERS POINTER
S179 026424 012701 002412 MOV #PRMO,R1 ;: TABLE POINTER
S180 026430 010102 MOV R1,R2 ;: STOP ADDRESS
S181 026432 012021 1$: MOV (R0)+,(R1)+ ;: MOVE DEFAULT PARAMETERS INTO
S182 026434 020002 CMP R0,R2 ;: RUN TIME TABLES ** DONE?
S183 026436 103775 BLO 1$ ;: NO--BRANCH
S184 026440 012700 003606 MOV #PAT8,R0 ;: PATO DEFAULTS TO PATTERN 8
S185 026444 012701 003206 MOV #PATO,R1
S186 026450 012021 2$: MOV (R0)+,(R1)+
S187 026452 020027 003646 CMP R0,#PAT9
S188 026456 103774 BLO 2$
S189 026460 032737 000001 001300 BIT #BIT00,C.SWR ;: 16 BIT MODE ?
S190 026466 001012 BNE 3$ ;: BR IF 16 BIT MODE
S191 026470 012737 000037 001712 MOV #31,PRMLMT+22 ;: SET 'FS' LIMIT TO 31.
S192 026476 012737 000037 001714 MOV #31,PRMLMT+24 ;: SET 'LS' LIMIT TO 31.
S193 026504 012737 160000 001432 MOV #-(<256.*32.>),TRCKWC ;: WORD COUNT FOR A 16 BIT TRACK
S194 026512 000411 BR 4$ ;: CONTINUE
S195 026514 012737 000035 001712 3$: MOV #29,PRMLMT+22 ;: SET 'FS' LIMIT TO 29.
S196 026522 012737 000035 001714 3$: MOV #29,PRMLMT+24 ;: SET 'LS' LIMIT TO 29.
S197 026530 012737 161000 001432 3$: MOV #-(<256.*30.>),TRCKWC ;: WORD COUNT FOR AN 18 BIT TRACK
S198 026536 012701 001620 4$: MOV #PRMPT,R1 ;: ADDRESS OF PARAMETER POINTER TABLE
S199 026542 005711 5$: TST (R1) ;: END OF THE TABLE ?
S200 026544 001425 BEQ 3$ ;: BR IF END
S201 026546 032731 002000 BIT #BIT10,2(R1)+ ;: 'LS' SELECTED ?
S202 026552 001773 BEQ 5$ ;: BR IF NOT
S203 026554 016102 177776 MOV -2(R1),R2 ;: PARAMETER TABLE ADDRESS
S204 026560 011246 MOV (R2),-(SP) ;: PARAMETER ALLOCATION BITS
S205 026562 012703 000013 MOV #11,R3 ;: NUMBER OF PARAMETERS (MAXIMUM) BEFORE 'LS'
S206 026566 006216 6$: ASR (SP),R3 ;: COUNT THE PARAMETER
S207 026570 103002 BCC 7$ ;: BR IF NOT USED
S208 026572 062702 000002 ADD #2,R2 ;: INCREMENT THE PARAMETER TABLE ADDRESS
S209 026576 005303 7$: DEC R3 ;: COUNT THE PARAMETER
S210 026600 001372 BNE 6$ ;: BR IF NOT THERE YET
S211 026602 005726 TST (SP)+ ;: CORRECT THE STACK POINTER
S212 026604 021237 001712 CMP (R2),PRMLMT+22 ;: IS 'LS' TOO LARGE FOR THE MODE SELECTED ?
S213 026610 101754 BLOS 5$ ;: BR IF NOT
S214 026612 013712 001712 MOV PRMLMT+22,(R2) ;: RESET VALUE FOR MODE USED
S215 026616 000751 BR 5$ ;: CONTINUE
S216 026620 8$:
S217 026620 012603 MOV (SP)+,R3 ;: POP STACK INTO R3
S218 026622 012602 MOV (SP)+,R2 ;: POP STACK INTO R2
S219 026624 012601 MOV (SP)+,R1 ;: POP STACK INTO R1
S220 026626 012600 MOV (SP)+,R0 ;: POP STACK INTO R0
S221 026630 000207 RTS PC ;: RETURN

```

;THIS ROUTINE FILLS THE PARAMETER TABLE THE CURRENT TEST.
 ;CALL


```

5225      ;      MOV      #TESTNUM,STSTNM ;LOAD THE TEST NUMBER
5226      ;      JSR      PC,LODPRM
5227      ;      RETURN
5228
5229      LODPRM:
5230      MOV      R1,-(SP) ;: PUSH R1 ON STACK
5231      MOV      R2,-(SP) ;: PUSH R2 ON STACK
5232      MOV      R3,-(SP) ;: PUSH R3 ON STACK
5233      MOV      R4,-(SP) ;: PUSH R4 ON STACK
5234      CLR      R4 ;: CLEAR R4
5235      MOV      STSTNM,R4 ;: GET THE TEST NUMBER
5236      ASL      R4 ;: SETUP TO ADDRESS WORDS
5237      MOV      PRMPT(R4),R1 ;: GET THE TEST'S PARAMETER TABLE ADDRESS
5238      MOV      #PRM,R2 ;: PARAMETER EXECUTION TABLE
5239      CLR      R3 ;: R3 IS USED AS A COUNTER
5240      MOV      CHKDRV,R4 ;: DRIVE'S ADDRESS
5241      MOV      (R1)+,(R2)+ ;: PARAMETER SPECIFIER
5242      1$: ASR      PRM ;: THIS PARAMETER USED IN THE TEST ?
5243      BCC      2$ ;: BR IF NOT
5244      MOV      (R1)+,(R2)+ ;: LOAD THE VALUE
5245      BR      3$ ;: CONTINUE
5246      2$: CLR      (R2)+ ;: CLEAR THE UNUSED PARAMETER LOCATION
5247      3$: INC      R3 ;: COUNT THE POSITION IN THE OUTPUT TABLE
5248      CMP      R3,#12. ;: FINISHED ?
5249      BEQ      6$ ;: BR IF YES
5250      CMP      R3,#3 ;: DOING THE CYLINDER ADDRESSES ?
5251      BNE      1$ ;: BR IF NOT
5252      BITB     #BIT02,DRV TYP(R4) ;: RMO3 ?
5253      BNE      5$ ;: IF IT IS OVERLAY FC & LC WITH FC' & LC'
5254      ADD      #2,R3 ;: COUNT THE BYPASSED PARAMETERS (FC' & LC')
5255      ASR      PRM ;: SHIFT THE COUNTER
5256      BCC      4$ ;: BR IF FC' IS NOT USED
5257      ADD      #2,R1 ;: MOVE THE INPUT POINTER
5258      4$: ASR      PRM ;: COUNT THE PARAMETER
5259      BCC      1$ ;: BR IF LC' NOT USED
5260      ADD      #2,R1 ;: MOVE THE INPUT PINTER
5261      BR      1$ ;: KEEP GOING
5262      BR      1$ ;: KEEP GOING
5263      5$: SUB      #4,R2 ;: BACKUP THE OUTPUT POINTER
5264      BR      1$ ;: KEEP GOING
5265      6$:
5266      MOV      (SP)+,R4 ;: POP STACK INTO R4
5267      MOV      (SP)+,R3 ;: POP STACK INTO R3
5268      MOV      (SP)+,R2 ;: POP STACK INTO R2
5269      MOV      (SP)+,R1 ;: POP STACK INTO R1
5270      RTS      PC ;: RETURN
5271
5272      ; THIS ROUTINE LOADS A READ HEADER AND DATA COMMAND OR A SEEK COMMAND.
5273      ; INTO DPB.B+2 AND DPB.C+2, DEPENDING ON THE STATE OF "CONTROL SWITCH"
5274      ; BIT07.
5275      ; CALL
5276      ;
5277      ; JSR      PC,@#LDCMD
5278      ; RETURN
5279      LDCMD: BIT      #SW07,@#C.SWR ;: DO EXPLICIT SEEKS?
5280      BNE      1$ ;: YES--BRANCH
  
```



```

5281 027020 012737 000173 004230      MOV      #READHD,2#DPB.B+2 ;NO--SET UP FOR READ HEADER AND
5282 027026 012737 000173 004250      MOV      #READHD,2#DPB.C+2 ;DATA COMMAND
5283 027034 000406                BR      2$
5284 027036 012737 000105 004230 1$:  MOV      #SEEK,2#DPB.B+2 ;SETUP FOR SEEK COMMAND
5285 027044 012737 000105 004250      MOV      #SEEK,2#DPB.C+2
5286 027052 000207                2$:  RTS      PC
5287
5288 ;THIS ROUTINE WILL CALL THE RMO3 DRIVER AND THEN WAIT ON THE FUNCTION
5289 ;TO COMPLETE. IF AN ERROR OCCURS IT IS REPORTED.
5290 ;CALL
5291 ;      FILL "DPB" WITH COMMAND INFORMATION
5292 ;      JSR      RO,2#CALL.A
5293 ;      RETURN
5294
5295 027054 005037 001206      CALL.A: CLR      2#SEESCAPE      ;NO ESCAPE ADDRESS
5296 027060 004037 037502      JSR      RO,2#RMO3      ;CALL RMO3 DRIVER
5297 027064 004206      DPB.A
5298 027066 000772      BR      CALL.A
5299 027070 005737 004224 1$:  TST      2#DPB.A+16      ;DONE?
5300 027074 001775      BEQ      1$              ;NO--LOOP
5301 027076 100042      BPL      4$              ;BRANCH IF NO ERROR
5302 027100 012737 027174 001206      MOV      #3$,SEESCAPE      ;ESCAPE TO 3$ ON ERROR
5303 027106 013737 004220 001350      MOV      2#DPB.A+12,2#CYL.DS ;CYLINDER
5304 027114 113737 004217 001354      MOV      2#DPB.A+11,2#TRK.DS ;TRACK
5305 027122 113737 004216 001352      MOV      2#DPB.A+10,2#SEC.DS ;SECTOR
5306 027130 012746 004224      MOV      #DPB.A+16, -(SP) ;STATUS/ERROR INDICATOR ADDRESS
5307 027134 004737 030450      JSR      PC,2#ERINDX      ;FORM DISPATCH INDEX
5308 027140 062607      ADD      (SP)+,PC      ;REPORT PROPER ERROR
5309 027142 104041      ERROR   41
5310 027144 104042      ERROR   42              ;PARITY ERROR
5311 027146 104043      ERROR   43              ;UNSAFE ERROR
5312 027150 104044      ERROR   44              ;NON-I/O ERROR
5313 027152 000240      NOP
5314 027154 005737 004322      TST      RM.REG+RMER1      ;TO SYNC THE CALLING SEQ OF ERINDX
5315 027160 001004      BNE      2$              ;ANY DRIVE ERROR
5316 027162 032737 100000 004350      BIT      #BIT15,RM.REG+RMER2 ;BRANCH IF 50
5317 027170 001005      BNE      4$              ;BAD SPOT ERROR
5318 027172 104045      ERROR   45              ;BRANCH IF 50
5319 027174 013746 004224 2$:  ERROR   45              ;I/O ERROR
5320 027200 034737 030410 3$:  MOV      DPB.A+16, -(SP) ;STATUS WORD
5321 027204 000200      JSR      PC,LOP.CK      ;SEE IF LOOP, ABORT, OR CONTINUE
5322      RTS      RO          ;RETURN
5323
5324 ;THIS ROUTINE IS THE SAME AS "CALL.A" EXCEPT FOR THE DPB USED AND IF
5325 ;THE COMMAND IS A READ HEADER AND DATA THE HEADER (CYLINDER, TRACK,
5326 ;AND SECTOR) READ IS CHECKED FOR VALIDITY.
5327 ;CALL
5328 ;      FILL DPB
5329 ;      JSR      RG,2#CALL.B
5330 ;      RETURN
5331
5331 027206 005037 001206      CALL.B: CLR      2#SEESCAPE      ;NO ESCAPE ADDRESS
5332 027212 004037 037502      JSR      RO,2#RMO3      ;CALL RMO3 DRIVER
5333 027216 004226      DPB.B
5334 027220 000772      BR      CALL.B
5335 027222 005737 004244 1$:  TST      DPB.B+16      ;DONE?
5336 027226 001775      BEQ      1$              ;NO--BRANCH

```



```

5337 027230 100047          BPL      4$          ;BRANCH IF NO ERROR
5338 027232 012737 027336 001206    MOV      #3$, $ESCAPE ; ESCAPE TO 3$ ON ERROR
5339 027240 013737 004240 001350    MOV      @DPB.B+12,@CYL.DS ;CYLINDER
5340 027246 113737 004237 001354    MOV      @DPB.B+11,@TRK.DS ;TRACK
5341 027254 113737 004236 001352    MOV      @DPB.B+10,@SEC.DS ;SECTOR
5342 027262 012746 004244          MOV      @DPB.B+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
5343 027266 004737 030450          JSR      PC,@ERINDX    ;FORM DISPATCH INDEX
5344 027272 062607          ADD      (SP)+,PC     ;REPORT PROPER ERROR
5345 027274 104041          ERROR   41          ;
5346 027276 104042          ERROR   42          ;PARITY ERROR
5347 027300 104043          ERROR   43          ;UNSAFE ERROR
5348 027302 104044          ERROR   44          ;NON-I/O ERROR
5349 027304 000240          NOP                      ;TO SYNC THE CALLING SEQ OF ERINDX: RT.
5350 027306 005737 004322          TST      RM.REG+RMR1   ;DRIVE ERROR ?
5351 027312 001404          BEQ      2$          ;BR IF NOT
5352 027314 032737 177677 004322          BIT      #1C100,RM.REG+RMR1 ;SEE IF ONLY 'HCE' SET
5353 027322 001412          BEQ      4$          ;BR IF IT IS
5354 027324          ;
5355 027324 032737 100000 004350 2$:    BIT      #BIT15,RM.REG+RMR2 ;BSE ERROR
5356 027332 001025          BNE                      ;BRANCH IF SO
5357 027334 104045          ERROR   45          ;I/O ERROR
5358 027336 013746 004244 3$:    MOV      DPB.B+16,-(SP) ;STATUS WORD
5359 027342 004737 030410          JSR      PC,L0P.CK    ;SEE IF LOOP, ABORT, OR CONTINUE
5360 027346 000410          BR                      ;CHECK FOR STALL
5361 027350 123727 004230 000173 4$:    CMP      @DPB.B+2,@READHD ;DOING IMPLIED SEEKS?
5362 027356 001004          BNE      5$          ;NO--BRANCH
5363 027360 004037 030750          JSR      RO,@VERIFY  ;YES--GO CHECK THE DATA
5364 027364 004236          DPB.B+10
5365 027366 000407          BR                      ;ERROR DURING VERIFY
5366 027370 032737 040000 001300 5$:    BIT      #SW14,@C.SWR  ;STALL?
5367 027376 001403          BEQ      6$          ;NO--BRANCH
5368 027400 004037 030606          JSR      RO,@STALL   ;YES--CALL STALL ROUTINE
5369 027404 001434          WORD   STALL1      ;STALL TIME POINTER
5370 027406 000200          RTS      RO         ;RETURN
5371          ;
5372          ;THIS ROUTINE IS THE SAME AS "CALL.B" EXCEPT FOR THE DPB USED.
5373          ;CALL
5374          ;
5375          ;FILL DPB
5376          ;JSR      RO,@CALL.C
5377          ;RETURN
5378 027410 005037 001206    CALL.C: CLR      @SEESCAPE ;NO ESCAPE ADDRESS
5379 027414 004037 037502          JSR      RO,@RMO3    ;CALL RMO3 DRIVER
5380 027420 004246          DPB.C
5381 027422 000772          BR                      ;
5382 027424 005737 004264 1$:    TST      @DPB.C+16   ;DONE?
5383 027430 001775          BEQ      1$          ;NO--LOOP
5384 027432 100047          BPL      4$          ;YES--BRANCH IF NO ERROR
5385 027434 012737 027540 001206    MOV      #3$, $ESCAPE ; ESCAPE TO 3$ ON ERROR
5386 027442 013737 004260 001350    MOV      @DPB.C+12,@CYL.DS ;CYLINDER
5387 027450 113737 004257 001354    MOV      @DPB.C+11,@TRK.DS ;TRACK
5388 027456 113737 004256 001352    MOV      @DPB.C+10,@SEC.DS ;SECTOR
5389 027464 012746 004264          MOV      @DPB.C+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
5390 027470 004737 030450          JSR      PC,@ERINDX  ;FORM DISPATCH INDEX
5391 027474 062607          ADD      (SP)+,PC     ;REPORT PROPER ERROR
5392 027476 104041          ERROR   41          ;

```

```

5393 027500 104042 ERROR 42 ; PARITY ERROR
5394 027502 104043 ERROR 43 ; UNSAFE ERROR
5395 027504 104044 ERROR 44 ; NON-I/O ERROR
5396 027506 000240 NOP ; TO SYNC THE CALLING SEQ OF ERINDX: RT.
5397 027510 005737 004322 TST RM.REG+RMR1 ; DRIVE ERROR ?
5398 027514 001404 BEQ 2$ ; BR IF NOT
5399 027516 032737 177677 004322 BIT #C100,RM.REG+RMR1 ; SEE IF ONLY 'HCE' SET
5400 027524 001412 BEQ 4$ ; BR IF IT IS
5401 027526 2$:
5402 027526 032737 100000 004350 BIT #BIT15,RM.REG+RMR2 ; BSE ERROR ONLY ?
5403 027534 001025 BNE 6$ ; BRANCH IF 5$
5404 027536 104045 ERROR 45 ; I/O ERROR
5405 027540 013746 004264 3$: MOV DPB.C+16,-(SP) ; STATUS WORD
5406 027544 004737 030410 JSR PC,LOP.CK ; SEE IF LOOP, ABORT, OR CONTINUE
5407 027550 000410 BR 5$
5408 027552 123727 004250 000173 4$: CMPB @DPB.C+2,@READHD ; DOING IMPLIED SEEK?
5409 027560 001004 BNE 5$ ; NO--EXIT
5410 027562 004037 030750 JSR RO,@VERIFY ; YES--CHECK THE DATA
5411 027566 004256 DPB.C+10
5412 027570 000407 BR 6$ ; ERROR DURING VERIFY
5413 027572 032737 040000 001300 5$: BIT #SW14,@C.SWR ; STALL?
5414 027600 001403 BEQ 6$ ; NO--BRANCH
5415 027602 004037 030606 JSR RO,@STALL ; YES--CALL STALL ROUTINE
5416 027606 001434 .WORD STALL1 ; STALL TIME POINTER
5417 027610 6$: RTS RO
5418
5419
5420 ; THIS ROUTINE IS THE SAME AS "CALL.A" EXCEPT FOR THE DPB USED AND
5421 ; ON AN ERROR LOCATION "ERR.CT" IS EXAMINED. IF ERR.CT IS EQUAL TO
5422 ; $ERFLG EXIT IS TO THE NEXT TEST.
5423 ; CALL
5424 ;
5425 ; FILL DPB
5426 ; JSR RO,@DRVCAL
5427 ; RETURN
5428
5429 DRVCAL: CLR @SEESCAPE ; NO ESCAPE ADDRESS
5430 027612 005037 001206 CLR @MCEFLG ; CLEAR WRITE CHECK ERROR FLAG
5431 027616 005037 001414 JSR RO,@RMO3 ; CALL RMO3 DRIVER
5432 027622 004037 037502 DTADPB
5433 027626 004266 BR DRVCAL
5434 027630 000770 DRVCAL
5435 027632 005737 004304 DRVCL1: TST @DTADPB+16 ; DONE
5436 027636 001775 BEQ DRVCL1 ; NO--LOOP
5437 027640 100402 BMI 1$ ; BR IF ERRORS
5438 027642 000137 030370 JMP 10$ ; NO ERRORS
5439 1$: MOV #2$,SEESCAPE ; ESCAPE TO 2$ ON ERROR
5440 027646 012737 027754 001206 MOV @DTADPB+12,@CYL.DS ; CYLINDER
5441 027654 013737 004300 001350 MOVB @DTADPB+11,@TRK.DS ; TRACK
5442 027662 113737 004277 001354 MOVB @DTADPB+10,@SEC.DS ; SECTOR
5443 027670 113737 004276 001352 MOV @DTADPB+16,-(SP) ; STATUS/ERROR INDICATOR ADDRESS
5444 027676 012746 004304 JSR PC,@ERINDX ; FORM DISPATCH INDEX
5445 027702 004737 030450 ADD (SP)+,PC ; REPORT PROPER ERROR
5446 027706 062607 ERROR 41
5447 027710 104041 ERROR 42 ; PARITY ERROR
5448 027712 104042 ERROR 43 ; UNSAFE ERROR
5449 027714 104043 ERROR 44 ; NON-I/O ERROR
5450 027716 104044

```



```

5505 030226 001775 BEQ 6$ ;NO--LOOP
5506 030230 100057 BPL 10$ ;YES--BRANCH IF NO ERROR
5507 030232 012737 000001 001414 7$: MOV #1,#WCEFLG ;SET THE WRITE CHECK ERROR FLAG
5508 030240 012737 030344 001206 MOV #8$,SESCAPE ;ESCAPE TO 8$ ON ERROR
5509 030246 013737 004300 001350 MOV #DTADPB+12,#CYL.DS ;CYLINDER
5510 030254 113737 004277 001354 MOV#B #DTADPB+11,#TRK.DS ;TRACK
5511 030262 113737 004276 001352 MOV#B #DTADPB+10,#SEC.DS ;SECTOR
5512 030270 012746 004304 MOV #DTADPB+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
5513 030274 004737 030450 JSR PC,#ERINDX ;FORM DISPATCH INDEX
5514 030300 062607 ADD (SP)+,PC ;REPORT PROPER ERROR
5515 030302 104041 ERROR 41 ;
5516 030304 104042 ERROR 42 ;PARITY ERROR
5517 030306 104043 ERROR 43 ;UNSAFE ERROR
5518 030310 104044 ERROR 44 ;NON-I/O ERROR
5519 030312 000240 NOP ;SO SYN THE CALLING SEQ OF ERINDX:
5520 030314 005737 004322 TST RM.REG+RMER1 ;ANY DRIVE ERROR
5521 030320 001010 BNE 88$ ;BRANCH IF SO
5522 030322 032737 100000 004350 BIT #BIT15,RM.REG+RMER2 ;BAD SOPT ERROR ?
5523 030330 001404 BEQ 88$ ;BRANCH IF NOT
5524 030332 012737 177777 001446 MOV #-1,BASFLG ;SET BAD SECTOR ENCOUNTER FLAG
5525 030340 000413 BR 10$ ;OTHERWISE EXIT
5526 030342 104046 88$: ERROR 46 ;REPORT THE FATAL WRITE CHECK ERROR
5527 030344 013746 004304 8$: MOV DTADPB+16,-(SP) ;STATUS WORD
5528 030350 004737 030410 JSR PC,LOP.CK ;SEE IF LOOP, ABORT, OR CONTINUE
5529 030354 123737 001444 001103 12$: CMPB #ERR.CT,#SERFLG ;GO TO NEXT TEST?
5530 030362 101002 BHI 10$ ;NO--BRANCH
5531 030364 013700 001332 9$: MOV #BYPASS,RO ;YES--GET EXIT ADDRESS
5532 030370 032737 040000 001300 10$: BIT #SW14,#C.SWR ;STALL?
5533 030376 001403 BEQ 11$ ;NO--BRANCH
5534 030400 004037 030606 JSR RO,#STALL ;YES--CALL STALL ROUTINE
5535 030404 001436 .WORD STALL2 ;STALL TIME POINTER
5536 030406 000200 11$: RTS RO
5537
5538 ;THIS SUBROUTINE CHECK FOR LOOP, ABORT, OR CONTINUE SWITCHES AFTER
5539 ;ERRORS 41, 42, 43, 44, 45, AND 46.
5540 ;CALL
5541 ;
5542 ; MOV DTA+16,-(SP) ;STATUS WORD FROM DPB IN USE
5543 ; JSR PC,LOP.CK
5544 ; RETURN
5545 030410 032777 001000 150522 LOP.CK: BIT #SW9,#SWR ;LOOP ON ERROR
5546 030416 001402 BEQ 1$ ;BR IF NOT
5547 030420 000177 150464 JMP #SLPERR ;START AT THE LOOP ADDRESS
5548 030424 005037 001206 1$: CLR #SESCAPE ;CLEAR ERROR ESCAPE FLAG
5549 030430 032766 072006 000002 BIT #BIT14!BIT13!BIT12!BIT10!BIT02!BIT01,2(SP) ;CHECK ERROR TYPE
5550 030436 001402 BEQ 2$ ;BR IF DRIVE NOT OFFLINE, UNLOADED, OR
5551 ;PERSISTENT UNSAFE OR FATAL MASSBUS PARITY
5552 030440 000137 021052 JMP #EOP ;TERMINATE DRIVE
5553 030444 012616 2$: MOV (SP)+,(SP) ;ADJUST RETURN ADDRESS
5554 030446 000207 RTS PC
5555
5556 ;THIS ROUTINE FORMS AN INDEX THAT WILL BE USED TO DISPATCH
5557 ;TO THE PROPER ERROR CALL. THE INDEX IS FORMED BY EXAMINING
5558 ;THE STATUS/ERROR INDICATOR OF THE APPLICABLE DPB.
5559 ;INDEX STATUS/ERROR
5560 ;-----
    
```



```

5561      0 BIT14!BIT13!BIT08!BIT01
5562      2 BIT11!BIT10!BIT02
5563      4 BIT12!BIT04
5564      6 BIT05!BIT03!<BIT09 & COMMAND=NON-I/O>
5565     10 BIT06!<BIT09 & COMMAND=I/O>
5566     CALL
5567      JSR      #DPB+16,-(SP) ;ADDRESS OF STATUS/ERROR INDICATOR
5568      JSR      PC,#ERINDX  ;FORM INDEX
5569      RETURN   ;INDEX IS ON THE STACK
5570
5571 030450 010046 ERINDX: MOV    RO,-(SP) ;SAVE RO
5572 030452 010146      MOV    R1,-(SP) ;SAVE R1
5573 030454 016600 000006      MOV    6(SP),RO ;GET STATUS/ERROR INDICATOR POINTER
5574 030460 011037 001340      MOV    (RO),#SVSTAT ;SAVE THE STATUS/ERROR INDICATOR
5575 030464 005001      CLR    R1 ;START INDEX AT ZERO
5576 030466 032710      BIT    (PC)+(RO) ;FORM INDEX OF 0?
5577 030470 020402      .WORD BIT13!BIT08!BIT01
5578 030472 001037      BNE    5$ ;YES--BRANCH
5579 030474 032710      BIT    (PC)+(RO) ;FORM PARITY ERROR OR PORT REQUEST INDEX (2)?
5580 030476 006004      .WORD BIT11!BIT10!BIT02
5581 030500 001033      BNE    4$ ;YES--BRANCH
5582 030502 032710      BIT    (PC)+(RO) ;FORM UNSAFE INDEX (4)?
5583 030504 050020      .WORD BIT14!BIT12!BIT04
5584 030506 001027      BNE    3$ ;YES--BRANCH
5585 030510 032710      BIT    (PC)+(RO) ;FORM NON-I/O ERROR INDEX (6)?
5586 030512 000050      .WORD BIT05!BIT03
5587 030514 001023      BNE    2$ ;YES--BRANCH
5588 030516 032710      BIT    (PC)+(RO) ;FORM I/O ERROR INDEX (10)?
5589 030520 000100      .WORD BIT06
5590 030522 001017      BNE    1$ ;YES--BRANCH
5591 030524 032710      BIT    (PC)+(RO) ;SOFTWARE TIMEOUT?
5592 030526 001000      .WORD BIT09
5593 030530 001420      BEQ    5$ ;NO--FORM INDEX OF 0
5594 030532 122760 000150 177762      CMPB  #150,-16(RO) ;YES--I/O?
5595 030540 003011      BGT    2$ ;NO--BRANCH
5596 030542 005737 004322      TST   RM.REG+RMER1 ;ANY DRIVE ERROR ?
5597 030546 001005      BNE    1$ ;BRANCH IF SO
5598 030550 032737 100000 004350      BIT   #BIT15,RM.REG+RMER2 ;BSE ERROR
5599 030556 001401      BEQ    1$ ;BRANCH IF NOT
5600 030560 005201      INC    R1 ;SKIP , NOT REPORT BSE ERROR
5601 030562 005201      1$: INC    R1 ;INDEX=10---ERROR=45 OR 46
5602 030564 005201      2$: INC    R1 ;INDEX=6---ERROR=44
5603 030566 005201      3$: INC    R1 ;INDEX=4---ERROR=43
5604 030570 005201      4$: INC    R1 ;INDEX=2---ERROR=42
5605 030572 006301      5$: ASL   R1 ;INDEX=0---ERROR=41
5606 030574 010166 000006      MOV   R1,6(SP) ;RETURN INDEX TO USER
5607 030600 012601      MOV   (SP)+,R1 ;RESTORE R1
5608 030602 012600      MOV   (SP)+,RO ;RESTORE RO
5609 030604 000207      RTS   PC ;RETURN FROM CALL

```

```

; THIS ROUTINE WILL PROVIDE A STALL IN MILLISECONDS FOR A SPECIFIC
; AMOUNT OF TIME IF BIT13 OF C.SWR = 0 OR A RANDOM AMOUNT OF TIME
; IF BIT 13 OF C.SWR = 1.
; STALL1 CONTAINS SPECIFIED TIME FOR TESTS 0 - 7, AND STALL2
; CONTAINS THE TIME FOR TESTS 16-21.
; CALL

```

5610
5611
5612
5613
5614
5615
5616

```

5617      ;      JSR      RO,@#STALL
5618      ;      TIME POINTER      ;WHERE TO FIND THE STALL TIME
5619
5620      030606 013046      STALL:  MOV      @ (RO)+, -(SP)      ; PICKUP STALL TIME
5621      030610 032737 020000 001300      BIT      @SW13,@#C.SWR      ; USE A RANDOM TIME?
5622      030616 001406      BEQ      1$      ; NO--BRANCH
5623      030620 004737 025532      JSR      PC,@#SRAND      ; YES--FORM RANDOM NUMBER
5624      030624 013716 025632      MOV      @#$LONUM,(SP)      ; AND USE IT FOR THE STALL TIME
5625      030630 042716 177700      BIC      @#C77,(SP)      ; BUT NEVER > 64 MILLISECONDS
5626      030634 005046      1$:  CLR      -(SP)      ; CLEAR TEMP. LOCATION
5627      030636 162766 000001 000002      2$:  SUB      @1,2(SP)      ; MORE STALL REQUIRED?
5628      030644 103407      BLO      4$      ; NO--BRANCH
5629      030646 012716 000144      MOV      @100.,(SP)      ; STALL FOR ABOUT 1 MILLISECOND
5630      030652 005700      3$:  TST      RO      ; NOP TO KILL TIME
5631      030654 005366 000000      DEC      0(SP)      ; COUNT
5632      030660 001374      BNE      3$      ; LOOP IF MORE COUNTS NEEDED
5633      030662 000765      BR       2$
5634      030664 022626      4$:  CMP      (SP)+,(SP)+      ; CLEAN OFF THE STACK
5635      030666 000200      RTS      RO      ; EXIT
5636
5637
5638      ;ROUTINE TO PROVIDE A 2 MS STALL AFTER A SEEK OPERATION IN THE SEEK TIMING
5639      ;TESTS. THIS STALL IS REQUIRED TO COMPENSATE FOR THE 'ACCESS READY' DELAY
5640      ;IN THE RMO3. THIS STALL TIME IS NOT INCLUDED IN THE CALCULATED SEEK TIMES.
5641      ;CALL
5642      ;      JSR      PC,@#TWOMS
5643      ;      RETURN
5644
5645      030670 013746 177776      TWOMS:  MOV      @#PS, -(SP)      ; SAVE THE PRESENT PROCESSOR STATUS
5646      030674 012737 000240 177776      MOV      @(<5*32.>,@#PS      ; SET THE PROCESSOR PRIORITY TO 5
5647      030702 017746 150550      MOV      @PKV, -(SP)      ; SAVE THE OLD CLOCK VECTOR ADDRESS
5648      030706 012777 030732 150542      MOV      @1$,@PKV      ; SETUP NEW VECTOR ADDRESS
5649      030714 012777 000310 150542      MOV      @200.,@PKB      ; LOAD THE CLOCK BUFFER
5650      030722 012777 000101 150532      MOV      @101,@PKCS      ; START THE CLOCK
5651      030730 000001      WAIT      ; WAIT FOR 2 MS
5652      030732 062706 000004      1$:  ADD      @4,SP      ; INCREMENT STACK FOR RETURN
5653      030736 012677 150514      MOV      (SP)+,@PKV      ; RESTORE OLD CLOCK VECTOR
5654      030742 012637 177776      MOV      (SP)+,@#PS      ; RESTORE THE OLD PROCESSOR STATUS
5655      030746 000207      RTS      PC      ; RETURN
5656
5657      ;ROUTINE TO SOFTWARE COMPARE HEADER ON IMPLIED SEEKS
5658      ;CALL
5659      ;      JSR      RO,@#VERIFY
5660      ;      ADR POINTER      ; ADDRESS OF DPB+10 (SECTOR NUMBER)
5661      ;      RETURN
5662
5663      030750 010146      VERIFY:  MOV      R1, -(SP)      ; SAVE R1
5664      030752 012001      MOV      (RO)+, R1      ; GET ADDRESS OF DPB+10
5665      030754 042737 150000 051776      BIC      @150000,@#BUFFER      ; STRIP FORMAT AND BAD SECTOR BITS FROM CYLINDER NUMBER
5666      030762 023761 051776 000002      CMP      @#BUFFER,2(R1)      ; CYLINDER NUMBER OK?
5667      030770 001003      BNE      1$      ; NO--BRANCH
5668      030772 023711 052000      CMP      @#BUFFER+2,(R1)      ; YES--HOW ABOUT TRACK/SECTOR?
5669      030776 001441      BEQ      3$      ; BRANCH IF GOOD
5670      031000 013737 051776 001342      1$:  MOV      @#BUFFER,@#CYL.RD      ; SAVE THE EXPECTED AND THE
5671      031006 113737 052001 001344      MOV      @#BUFFER+3,@#TRK.RD      ; RECIEVED CYLINDER, TRACK,
5672      031014 113737 052000 001346      MOV      @#BUFFER+2,@#SEC.RD      ; AND SECTOR

```



```

5673 031022 112137 001352      MOV      (R1)+, @#SEC.DS
5674 031026 112137 001354      MOV      (R1)+, @#TRK.DS
5675 031032 011137 001350      MOV      (R1), @#CYL.DS
5676 031036 012737 031050 001206  MOV      @2$, $ESCAPE
5677 031044 005740          TST      -(R0)
5678 031046 104012          ERROR   12
5679 031050 012737 000107 004210 2$:  MOV      @RECAL.DPB.A+2
5680 031056 004037 027054      JSR      R0, @#CALL.A
5681 031062 005037 001206      CLR      $ESCAPE
5682 031066 032777 001000 150044  BIT      @SW9, @SWR
5683 031074 001404          BEQ     4$
5684 031076 000177 150006      JMP     @SLPERR
5685 031102 062700 000002      3$:  ADD     @2, R0
5686 031106 012601      4$:  MOV     (SP)+, R1
5687 031110 000200      RTS     R0
5688
5689
5690
5691
5692
5693
5694
5695
5696
5697 031112 005001          SRCH00: CLR     R1
5698 031114 005037 177776      CLR     @#PS
5699 031120 012777 041566 005612  MOV     @ISR, @RMVEC
5700 031126 013704 036736      MOV     @#RMADR, R4
5701 031132 012764 000040 000010  MOV     @BIT05, @MCS2(R4)
5702 031140 005037 004276      CLR     @#DTADPB+10
5703 031144 005037 004300      CLR     @#DTADPB+12
5704 031150 012737 000107 004270  MOV     @RECAL, @#DTADPB+2
5705 031156 005037 001206      CLR     @#ESCAPE
5706 031162 004037 037502      JSR     R0, @#RMO3
5707 031166 004266      DTADPB
5708 031170 000440          BR     4$
5709 031172 005737 004304 1$:  TST     DTADPB+16
5710 031176 001775          BEQ     1$
5711 031200 100030          BPL     3$
5712 031202 012737 031256 001206  MOV     @2$, $ESCAPE
5713 031210 013737 004300 001350  MOV     @#DTADPB+12, @#CYL.DS
5714 031216 113737 004277 001354  MOV     @#DTADPB+11, @#TRK.DS
5715 031224 113737 004276 001352  MOV     @#DTADPB+10, @#SEC.DS
5716 031232 012746 004304          MOV     @DTADPB+16, -(SP)
5717 031236 004737 030450      JSR     PC, @#ERINDX
5718 031242 062607          ADD     (SP)+, PC
5719 031244 104041          ERROR  41
5720 031246 104042          ERROR  42
5721 031250 104043          ERROR  43
5722 031252 104044          ERROR  44
5723 031254 104045          ERROR  45
5724 031256 005720 2$:  TST     (R0)+
5725 031260 000404          BR     4$
5726 031262 005064 000006 3$:  CLR     RMDA(R4)
5727 031266 005064 000034  CLR     RMDC(R4)
5728 031272 000200 4$:  RTS     R0

```

: THIS ROUTINE WILL PERFORM A "MASSBUS" INIT. FOLLOWED BY
: A "RECALIBRATE" ON THE DRIVE UNDER TEST.
: NOTE: THIS ROUTINE DESTROYS R1 AND R4

: CALL
: JSR R0, SRCH00 ; DO A MASSBUS INIT. AND RECAL
: RETURN1 ; RETURN HERE IF NO ERROR
: RETURN2 ; RETURN HERE ON ERROR

SRCH00: CLR R1 ; INCASE OF ERROR (TYPTIM)
CLR @#PS
MOV @ISR, @RMVEC ; SETUP INTERRUPT VECTOR
MOV @#RMADR, R4 ; PICKUP ADDRESS OF RMCS1
MOV @BIT05, @MCS2(R4) ; MASSBUS INIT.
CLR @#DTADPB+10 ; TRACK=0; SECTOR=0
CLR @#DTADPB+12 ; CYLINDER = 0
MOV @RECAL, @#DTADPB+2 ; COMMAND = RECALIBRATE
CLR @#ESCAPE ; NO ESCAPE ADDRESS
JSR R0, @#RMO3 ; CALL THE DRIVER
DTADPB ; DPB POINTER
BR 4\$; QUEUE IS FULL
1\$: TST DTADPB+16 ; WAIT ON DONE
BEQ 1\$
BPL 3\$; TAKE NORMAL EXIT IF NO ERROR
MOV @2\$, \$ESCAPE ; ESCAPE TO 2\$ ON ERROR
MOV @#DTADPB+12, @#CYL.DS ; CYLINDER
MOV @#DTADPB+11, @#TRK.DS ; TRACK
MOV @#DTADPB+10, @#SEC.DS ; SECTOR
MOV @DTADPB+16, -(SP) ; STATUS/ERROR INDICATOR ADDRESS
JSR PC, @#ERINDX ; FORM DISPATCH INDEX
ADD (SP)+, PC ; REPORT PROPER ERROR
ERROR 41
ERROR 42 ; PARITY ERROR
ERROR 43 ; UNSAFE ERROR
ERROR 44 ; NON-I/O ERROR
ERROR 45 ; I/O ERROR
2\$: TST (R0)+ ; ADJUST FOR ERROR EXIT
BR 4\$; GO TO THE EXIT
3\$: CLR RMDA(R4) ; TRACK AND SECTOR = 0
CLR RMDC(R4) ; CYLINDER = 0
4\$: RTS R0 ; RETURN

```

5729
5730 ;THIS IS AN RTI WHICH IS USED BY THE TIMING TESTS & THE SERVO SETTLE DOWN TEST
5731
5732 031274 000002 DORTI: RTI ;RETURN FROM INTERRUPT
5733
5734 ;THIS ROUTINE WILL INITIALIZE THE TIMERS USED BY THE "TIMING ROUTI ES
5735 ;CALL
5736 ; JSR PC,@#STRTMR
5737 ; RETURN
5738
5739 031276 104412 STRTMR: SAVREG ;SAVE RO-R5
5740 031300 012700 001356 MOV #TIM.UP,R0 ;START AT TIM.UP (MINIMUM)
5741 031304 012701 001412 MOV #TIM.PT,R1 ;STOP AT TIM.PT
5742 031310 005020 1$: CLR (R0)+ ;CLEAR
5743 031312 020001 CMP RO,R1 ;DONE?
5744 031314 103775 BLO 1$ ;NO--BRANCH
5745 031316 012710 051776 MOV #BUFFER,(R0) ;SETUP POINTER
5746 031322 012737 077777 001356 MOV #↑CBIT15,@#TIM.UP ;SET MINIMUM TIME TO MAXIMUM
5747 031330 012737 077777 001374 MOV #↑CBIT15,@#TIM.DN ;POSITIVE NUMBER
5748 031336 104413 RESREG ;RESTORE RO-R5
5749 031340 000207 RTS PC ;RETURN
5750
5751 ;THIS ROUTINE WILL ADD THE ELAPSED TIME TO THE AVERAGE COUNTER AND
5752 ;MAINTAIN THE MINIMUM AND MAXIMUM TIMES.
5753 ;NOTE: THIS ROUTINE DESTROYS R2
5754 ;CALL
5755 ; MOV #TP,R3 ;PARAMETER POINTER
5756 ; MOV FLAG,R5 ;FLAG=0=COUNT UP
5757 ; ;FLAG=-1=COUNT DOWN
5758 ; JSR PC,@#COUNT
5759 ; RETURN
5760
5761 031342 012702 001356 COUNT: MOV #TIM.UP,R2 ;PICKUP THE "UP" POINTER
5762 031346 005705 TST R5 ;USE IT?
5763 031350 001402 BEQ 1$ ;YES--BRANCH
5764 031352 012702 001374 MOV #TIM.DN,R2 ;NO--PICKUP "DOWN" POINTER
5765 031356 027722 150104 1$: CMP @PKC,(R2)+ ;LESS THAN PREVIOUS LOW?
5766 031362 002003 BGE 2$ ;NO--BRANCH
5767 031364 017762 150076 177776 MOV @PKC,-2(R2) ;YES--SAVE IT
5768 031372 027763 150070 000004 2$: CMP @PKC,4(R3) ;LESS THAN THE LOW LIMIT?
5769 031400 002001 BGE 3$ ;NO--BRANCH
5770 031402 005212 INC (R2) ;YES--COUNT IT
5771 031404 005722 3$: TST (R2)+ ;ADVANCE THE POINTER
5772 031406 027722 150054 CMP @PKC,(R2)+ ;GREATER THAN PREVIOUS HIGH?
5773 031412 003403 BLE 4$ ;NO--BRANCH
5774 031414 017762 150046 177776 MOV @PKC,-2(R2) ;YES--SAVE IT
5775 031422 027763 150040 000006 4$: CMP @PKC,6(R3) ;GREATER THAN THE HIGH LIMIT?
5776 031430 003401 BLE 5$ ;NO--BRANCH
5777 031432 005212 INC (R2) ;YES--COUNT IT
5778 031434 005722 5$: TST (R2)+ ;ADVANCE THE POINTER
5779 031436 067722 150024 ADD @PKC,(R2)+ ;ADD THIS COUNT TO THE TOTAL
5780 031442 005522 ADC (R2)+
5781 031444 005212 INC (R2) ;COUNT THIS READING
5782 031446 022737 060266 001412 CMP #BUFFER+<4*814.>,@#TIM.PT ;SAVE THIS COUNT?
5783 031454 101406 BLOS 6$ ;NO--BRANCH
5784 031456 017777 150004 147726 MOV @PKC,@TIM.PT ;YES--WELL SAVE IT THEN

```



```

5785 031464 062737 000002 001412      ADD    #2, @TIM.PT      ;ADVANCE THE POINTER
5786 031472 000207      RTS      PC              ;RETURN
5787
5788      ; THIS ROUTINE IS USED TO TYPE THE MINIMUM,
5789      ; MAXIMUM, AND AVERAGE TIMES FOR THE TIMING TESTS
5790      ; IT WILL ALSO CHECK THE TIMES TO ENSURE
5791      ; THEY ARE WITHIN TOLERANCE AND IF NOT FLAG THE BAD TIMES.
5792      ; NOTE: THIS ROUTINE DESTROYS R2-R5
5793      ; CALL
5794      JSR    RO, @TYPTIM      ;GO REPORT THE TIMES
5795      TABLE      ;POINT TO THE PROPER TABLE
5796      RETURN
5797
5798      TABLE:MSGADR1      ; ADDRESS OF ASCIZ MESSAGE NUMBER 1
5799      MSGADR2      ; ADDRESS OF ASCIZ MESSAGE NUMBER 2
5800      MIN.ALLOWED      ; MINIMUM TIME ALLOWED
5801      MAX.ALLOWED      ; MAXIMUM TIME ALLOWED
5802
5803 031474 012002      TYPTIM: MOV    (R0)+, R2      ; PICKUP THE TABLE POINTER
5804 031476 032777 000100 147434      BIT    #SW06, @SWR      ; INHIBIT TIME REPORTS?
5805 031504 001145      BNE    7$              ; YES--BRANCH
5806 031506 012237 031526      MOV    (R2)+, 2$      ; ADDRESS OF MESSAGE NUMBER 1
5807 031512 012205      MOV    (R2)+, R5      ; ADDRESS OF MESSAGE NUMBER 2
5808 031514 012203      MOV    (R2)+, R3      ; PICKUP THE LOW LIMIT
5809 031516 011202      MOV    (R2), R2        ; AND THE HIGH LIMIT
5810 031520 012704 001356      MOV    #TIM.UP, R4     ; PARAMETER POINTER
5811 031524 104401      1$:  TYPE      ; TYPE THE MESSAGE
5812 031526 000000      2$:  .WORD    0        ; ASCIZ MESSAGE POINTER GOES HERE
5813 031530 005764 000014      TST    14(R4)         ; DID ANY COUNTS OCCUR?
5814 031534 001527      BEQ    6$              ; NO--BRANCH
5815 031536 104401 046001      TYPE      MSGMIN      ; "MIN="
5816 031542 012446      MOV    (R4)+, -(SP)    ; PUT (R4)+ ON THE STACK
5817 031544 004737 025242      JSR    PC, @$$SB2D    ; CHANGE (R4)+ TO DECIMAL ASCIZ
5818 031550 004737 025472      JSR    PC, @$$SUPRS   ; TYPE WITHOUT LEADING ZEROS
5819 031554 104401 046026      TYPE      MSGOUS      ; "0 US"
5820 031560 005724      TST    (R4)+          ; ANY SEEKS BELOW THE LOW LIMIT
5821 031562 001421      BEQ    3$              ; NO--BRANCH
5822 031564 104401 046141      TYPE      MSG.SP      ; " "
5823 031570 016446 177776      MOV    -2(R4), -(SP)  ; PUT -2(R4) ON THE STACK
5824 031574 004737 025242      JSR    PC, @$$SB2D    ; CHANGE -2(R4) TO DECIMAL ASCIZ
5825 031600 004737 025472      JSR    PC, @$$SUPRS   ; TYPE WITHOUT LEADING ZEROS
5826 031604 104401 046033      TYPE      MBELOW      ; "BELOW THE MINIMUM OF"
5827 031610 010346      MOV    R3, -(SP)     ; PUT R3 ON THE STACK
5828 031612 004737 025242      JSR    PC, @$$SB2D    ; CHANGE R3 TO DECIMAL ASCIZ
5829 031616 004737 025472      JSR    PC, @$$SUPRS   ; TYPE WITHOUT LEADING ZEROS
5830 031622 104401 046026      TYPE      MSGOUS      ; "0 US"
5831 031626 104401 046010      3$:  TYPE      MSGMAX      ; "MAX="
5832 031632 012446      MOV    (R4)+, -(SP)  ; PUT (R4)+ ON THE STACK
5833 031634 004737 025242      JSR    PC, @$$SB2D    ; CHANGE (R4)+ TO DECIMAL ASCIZ
5834 031640 004737 025472      JSR    PC, @$$SUPRS   ; TYPE WITHOUT LEADING ZEROS
5835 031644 104401 046026      TYPE      MSGOUS      ; "0 US"
5836 031650 005724      TST    (R4)+          ; ANY SEEKS ABOVE THE HIGH LIMIT
5837 031652 000421      BR     4$              ; NO--BRANCH
5838 031654 104401 046141      TYPE      MSG.SP      ; YES--REPORT HOW MANY
5839 031660 016446 177776      MOV    -2(R4), -(SP)  ; PUT -2(R4) ON THE STACK
5840 031664 004737 025242      JSR    PC, @$$SB2D    ; CHANGE -2(R4) TO DECIMAL ASCIZ

```



```

5841 031670 004737 025472      JSR    PC, @$$SUPRS      ;TYPE WITHOUT LEADING ZEROS
5842 031674 104401 046062      TYPE   MABOVE           ;"ABOVE THE MAXIMUM OF"
5843 031700 010246                MOV    R2, -(SP)        ;PUT R2 ON THE STACK
5844 031702 004737 025242      JSR    PC, @$$SB2D      ;CHANGE R2 TO DECIMAL ASCIZ
5845 031706 004737 025472      JSR    PC, @$$SUPRS      ;TYPE WITHOUT LEADING ZEROS
5846 031712 104401 046026                TYPE   ,MSGOUS
5847 031716 104401 046017      4$:   TYPE   ,MSGAVG      ;"AVG="
5848 031722 012446                MOV    (R4)+, -(SP)     ;FORM THE AVERAGE
5849 031724 012446                MOV    (R4)+, -(SP)
5850 031726 012446                MOV    (R4)+, -(SP)
5851 031730 004737 025634      JSR    PC, @$$DIV
5852 031734 006126                ROL    (SP)+            ;IS THE REMAINDER OVER HALF?
5853 031736 100001                BPL    $               ;NO--BRANCH
5854 031740 005216                INC    (SP)            ;YES--ROUND UP
5855 031742                5$:
5856 031742 004737 025242      JSR    PC, @$$SB2D      ;CHANGE TO DECIMAL ASCIZ
5857 031746 004737 025472      JSR    PC, @$$SUPRS      ;TYPE WITHOUT LEADING ZEROS
5858 031752 104401 046026                TYPE   ,MSGOUS
5859 031756 104401 046141                TYPE   ,MSG.SP
5860 031762 016446 177776                MOV    -2(R4), -(SP)    ;PUT -2(R4) ON THE STACK
5861 031766 004737 025242      JSR    PC, @$$SB2D      ;CHANGE -2(R4) TO DECIMAL ASCIZ
5862 031772 004737 025472      JSR    PC, @$$SUPRS      ;TYPE WITHOUT LEADING ZEROS
5863 031776 104401 046111                TYPE   ,MSGNUM
5864 032002 010537 031526                MOV    R5, 2$          ;NEXT MESSAGE POINTER
5865 032006 001404                BEQ    7$              ;IF NONE EXIT
5866 032010 005005                CLR    R5              ;NO MORE THAN 2
5867 032012 000644                BR     1$
5868 032014 104401 046126      6$:   TYPE   ,MSGNON
5869 032020 000200      7$:   RTS    R0          ;EXIT
5870
5871                ; THIS SUBROUTINE WILL INCREMENT THE TRACK
5872                ; NUMBER (R2) BY THE AMOUNT SPECIFIED BY 'IT'.
5873                ; CALL
5874                ; JSR    RO, @#INCTRK
5875                ; RETURN1
5876                ; RETURN2
5877                ; TRACK NUMBER GREATER THAN LT15
5878                ; TRACK NUMBER INCREMENTED
5879 032022 020237 001602      INCTRK: CMP    R2, @#LT      ;LAST TRACK COMPLETED?
5880 032026 001410                BEQ    2$              ;YES--EXIT
5881 032030 063702 001604                ADD    @#IT, R2        ;NO--UPDATE TRACK
5882 032034 020237 001602                CMP    R2, @#LT
5883 032040 003402                BLE    1$              ;TRACK TO BIG?
5884 032042 013702 001602                MOV    @#LT, R2
5885 032046 005720      1$:   TST    (R0)+        ;YES--SET TRACK TO LAST TRACK
5886 032050 000200      2$:   RTS    R0          ;ADJUST FOR RETURN 2
5887                ; RETURN
5888                ; THIS SUBROUTINE WILL INCREMENT THE CYLINDER
5889                ; NUMBER (R1) BY THE AMOUNT SPECIFIED BY 'IC'.
5890                ; CALL
5891                ; JSR    RO, @#INCCYL
5892                ; RETURN1
5893                ; RETURN2
5894                ; CYLINDER NUMBER GREATER THAN LC15
5895                ; CYLINDER NUMBER INCREMENTED
5896 032052 020137 001574      INCCYL: CMP    R1, @#LC
5897 032056 001410                BEQ    2$              ;LAST CYLINDER COMPLETED?
5898                ; YES--EXIT

```



```

5897 032060 063701 001576      ADD    @#IC,R1      ;NO--UPDATE CYLINDER
5898 032064 020137 001574      CMP    R1,@#LC     ;CYLINDER TO BIG?
5899 032070 003402              BLE    1$          ;NO--EXIT
5900 032072 013701 001574      MOV    @#LC,R1     ;YES--SET CYLINDER TO LAST CYLINDER
5901 032076 005720              1$:   TST    (R0)+   ;ADJUST FOR RETURN 2
5902 032100 000200              2$:   RTS    R0     ;RETURN
5903
5904      ;THIS ROUTINE DECREASES THE SECTOR ADDRESS.
5905      ;CALL
5906      ;
5907      ;   CLR    -(SP)      ;CLEAR THE STACK
5908      ;   JSR    PC,DECSEC ;SUBROUTINE ENTRY
5909      ;   RETURN
5910 032102 113766 004314 000002  DECSEC: MOV    RM.REG+RMDA,2(SP) ;PUT THE SECTOR ADDRESS ON THE STACK
5911 032110 005366 000002              DEC    2(SP)      ;DECREMENT THE ADDRESS
5912 032114 100003              BPL    1$          ;BR IF NOT CORRECTION NEEDED
5913 032116 013766 001716 000002  1$:   MOV    PRMLMT+22.,2(SP) ;OVERFLOW OCCURED, FORCE TO MAXIMUM ADDRESS
5914 032124 000207              2$:   RTS    PC     ;RETURN
5915
5916      ;THIS SUBROUTINE IS USED TO FILL THE DATA BUFFER
5917      ;WITH ADDRESSES FROM 0 TO 21 WITH EACH ADDRESS
5918      ;BEING STORED IN 256 CONSECUTIVE LOCATIONS
5919      ;CALL
5920      ;
5921      ;   JSR    PC,@#FILBUF
5922      ;   RETURN
5923 032126 104412              FILBUF: SAVREG
5924 032130 005000              CLR    R0          ;SAVE R0 - R5
5925 032132 012701 051776      MOV    #BUFFER,R1 ;FIRST DISK ADDRESS
5926 032136 012702 000400      1$:   MOV    #256,R2   ;START FILLING HERE
5927 032142 010021              2$:   MOV    R0,(R1)+ ;DO 256 WORDS
5928 032144 005302              DEC    R2          ;STORE
5929 032146 003375              BGT    2$          ;MORE?
5930 032150 005200              INC    R0          ;YES--BRANCH
5931 032152 023700 001712      CMP    PRMLMT+22,R0 ;NO--UPDATE DISK ADDRESS
5932 032156 103367              BHS    1$          ;DONE?
5933 032160 104413              RESREG
5934 032162 000207              RTS    PC          ;NO--BRANCH
5935                          ;RESTORE R0 - R5
5936                          ;RETURN
5937
5938      ;THIS ROUTINE WILL CLEAR THE BUFFER BY
5939      ;SETTING EACH WORD TO "177400".
5940      ;CALL
5941      ;
5942      ;   JSR    R0,@#CLRBUF
5943      ;   RETURN
5944 032164 104412              CLRBUF: SAVREG
5945 032166 012701 177400      MOV    #177400,R1 ;SAVE R0 - R5
5946 032172 012702 051776      MOV    #BUFFER,R2 ;WORD TO FILL BUFFER WITH
5947 032176 012703 077776      MOV    #BUFFER+512.*22.,R3 ;FIRST ADDRESS OF BUFFER
5948 032202 010122              1$:   MOV    R1,(R2)+   ;LAST ADDRESS+2 OF BUFFER
5949 032204 010122              MOV    R1,(R2)+   ;FILL WORDS 1, 9,...249,...5625
5950 032206 010122              MOV    R1,(R2)+   ;FILL WORDS 2,10,...250,...5626
5951 032210 010122              MOV    R1,(R2)+   ;FILL WORDS 3,11,...251,...5627
5952 032212 010122              MOV    R1,(R2)+   ;FILL WORDS 4,12,...252,...5628
5953 032214 010122              MOV    R1,(R2)+   ;FILL WORDS 5,13,...253,...5629
5954 032216 010122              MOV    R1,(R2)+   ;FILL WORDS 6,14,...254,...5630
5955                          MOV    R1,(R2)+   ;FILL WORDS 7,15,...255,...5631

```



```

5953 032220 010122      MOV      R1,(R2)+      ;FILL WORDS 8,16,...256,...5632
5954 032222 020203      CMP      R2,R3        ;DONE?
5955 032224 103766      BLO      1$           ;NO--BRANCH
5956 032226 104413      RESREG   RO           ;RESTORE RO - R5
5957 032230 000200      RTS      RO           ;RETURN FROM CALL
5958
5959
5960      ; THIS ROUTINE IS USED TO CHECK THE DATA BUFFER
5961      ; FOR ADDRESSES 0 THROUGH 21 WITH EACH ADDRESS
5962      ; BEING STORED IN 256 CONSECUTIVE LOCATIONS
5963      ; CALL
5964      ; JSR      RO,@CKSCTR
5965      ; RETURN
5966 032232 104412      CKSCTR: SAVREG      ;SAVE RO - R5
5967 032234 162706 000004      SUB      #4,SP        ;RESERVE TEMP. STORAGE AREA
5968 032240 005001      CLR      R1           ;FIRST SECTOR
5969 032242 012716 051776      MOV      #BUFFER,(SP) ;FIRST ADDRESS OF DATA BUFFER
5970 032246 005066 000002      CLR      2(SP)        ;NO ERRORS
5971 032252 012702 000020      1$:      MOV      #16.,R2 ;LOOP COUNT (16*16=256)
5972 032256 011603      MOV      (SP),R3      ;GET 1ST ADDRESS OF THIS SECTORS DATA
5973 032260
5974 032260 020123      2$:      CMP      R1,(R3)+      ;WORD 1
5975 032262 001063      BNE      7$           ;BRANCH IF BAD
5976 032264 020123      CMP      R1,(R3)+      ;WORD 2
5977 032266 001061      BNE      7$           ;BRANCH IF BAD
5978 032270 020123      CMP      R1,(R3)+      ;WORD 3
5979 032272 001057      BNE      7$           ;BRANCH IF BAD
5980 032274 020123      CMP      R1,(R3)+      ;WORD 4
5981 032276 001055      BNE      7$           ;BRANCH IF BAD
5982 032300 020123      CMP      R1,(R3)+      ;WORD 5
5983 032302 001053      BNE      7$           ;BRANCH IF BAD
5984 032304 020123      CMP      R1,(R3)+      ;WORD 6
5985 032306 001051      BNE      7$           ;BRANCH IF BAD
5986 032310 020123      CMP      R1,(R3)+      ;WORD 7
5987 032312 001047      BNE      7$           ;BRANCH IF BAD
5988 032314 020123      CMP      R1,(R3)+      ;WORD 8
5989 032316 001045      BNE      7$           ;BRANCH IF BAD
5990 032320 020123      CMP      R1,(R3)+      ;WORD 9
5991 032322 001043      BNE      7$           ;BRANCH IF BAD
5992 032324 020123      CMP      R1,(R3)+      ;WORD 10
5993 032326 001041      BNE      7$           ;BRANCH IF BAD
5994 032330 020123      CMP      R1,(R3)+      ;WORD 11
5995 032332 001037      BNE      7$           ;BRANCH IF BAD
5996 032334 020123      CMP      R1,(R3)+      ;WORD 12
5997 032336 001035      BNE      7$           ;BRANCH IF BAD
5998 032340 020123      CMP      R1,(R3)+      ;WORD 13
5999 032342 001033      BNE      7$           ;BRANCH IF BAD
6000 032344 020123      CMP      R1,(R3)+      ;WORD 14
6001 032346 001031      BNE      7$           ;BRANCH IF BAD
6002 032350 020123      CMP      R1,(R3)+      ;WORD 15
6003 032352 001027      BNE      7$           ;BRANCH IF BAD
6004 032354 020123      CMP      R1,(R3)+      ;WORD 16
6005 032356 001025      BNE      7$           ;BRANCH IF BAD
6006 032360 005302      DEC      R2           ;FINISHED WITH THIS SECTORS DATA?
6007 032362 001336      BNE      2$          ;NO--BRANCH
6008 032364 062716 001000      3$:      ADD      #512.,(SP)    ;YES--FIRST ADDRESS OF NEXT SECTOR
    
```



```

6009 032370 005201          INC      R1          ; MOVE TO NEXT SECTOR
6010 032372 023701 001712  CMPB    PRMLT+22,R1 ; DONE?
6011 032376 103325          BHS     1$          ; NO--BRANCH
6012 032400 005766 000002 4$:      TST     2(SP)      ; ERROR OCCUR?
6013 032404 001406          BLO     6$          ; NO--BRANCH
6014 032406 123737 001444 001103  CMPB    2*ERR.CT,2*SERFLG ; MAX. ERROR OCCURRED?
6015 032414 101002          BHI     6$          ; NO--BRANCH
6016 032416 013700 001332 5$:      MOV     2*BYPASS,R0 ; TAKE ERROR EXIT
6017 032422 062706 000004 6$:      ADD     4,SP        ; FREE TEMP. AREA
6018 032426 104413          RESREG ; RESTORE R0 - R5
6019 032430 000200          RTS     R0         ; RETURN FROM CALL
6020 032432 010304 7$:      MOV     R3,R4      ; FORM WORD NUMBER AND
6021 032434 161604          SUB     (SP),R4    ; ADDRESS TO CONTINUE FROM
6022 032436 010405          MOV     R4,R5
6023 032440 006204          ASR    R4          ; WORD NUMBER
6024 032442 042705 177740  BIC    4*C37,R5
6025 032446 001002          BNE    8$          ; BRANCH IF NOT A MULTIPLE OF 16
6026 032450 012705 000040 8$:      MOV     40,R5     ; SET TO WORD 16
6027 032454 006305          ASL    R5
6028 032456 062705 032260  ADD     25,R5     ; ADDRESS
6029 032462 016337 177776 001126  MOV    -2(R3),2*SBDDAT ; SAVE BAD DATA
6030 032470 005766 000002  TST     2(SP)     ; FIRST ERROR?
6031 032474 001015          BNE    10$        ; NO--BRANCH
6032 032476 013737 004300 001350  MOV    2*DTADPB+12,2*CYL.DS ; CYLINDER NUMBER
6033 032504 113737 004277 001354  MOV    2*DTADPB+11,2*TRK.DS ; TRACK NUMBER
6034 032512 012737 032522 001206  MOV    9$,SESCAPE ; ESCAPE TO 9$ ON ERROR
6035 032520 104021          ERROR  21         ; REPORT THE ERROR
6036 032522 105166 000002 9$:      COMB   2(SP)     ; SET ERROR SWITCH
6037 032526 000404          BR     11$
6038 032530          10$:
6039 032530 012737 032540 001206  MOV    11$,SESCAPE ; ESCAPE TO 11$ ON ERROR
6040 032536 104022          ERROR  22         ; REPORT THE ERROR
6041 032540 032777 001000 146372 11$:   BIT    5*SW09,2*SWR ; LOOP ON ERROR?
6042 032546 001323          BNE    5$        ; YES
6043 032550 032777 000002 146362  BIT    5*SW01,2*SWR ; STOP DATA COMPARE?
6044 032556 001310          BNE    4$        ; YES--BRANCH
6045 032560 123737 001444 001103  CMPB    2*ERR.CT,2*SERFLG ; MAX. ERRORS?
6046 032566 101713          BLOS   5$        ; YES--BRANCH
6047 032570 032777 000040 146342  BIT    5*SW05,2*SWR ; REPORT ONLY 1ST ERROR PER SECTOR?
6048 032576 001272          BNE    3$        ; YES--BRANCH
6049 032600 000115          JMP    (R5)
6050
6051          ; THIS ROUTINE WILL MOVE THE 16 WORDS OF THE
6052          ; DESIRED PATTERN INTO THE DATA BUFFER.
6053          ; CALL
6054          ;
6055          ;      MOV     #NX,R0          ; PATTERN NUMBER INDEX TO R0
6056          ;      JSR     PC,2*SETBUF
6057 032602 104412  SETBUF: SAVREG ; SAVE R0 - R5
6058 032604 012701 051776  MOV     #BUFFER,R1 ; FIRST ADDRESS
6059 032610 013702 004272  MOV     2*DTADPB+4,R2 ; WORD COUNT
6060 032614 016003 003146 1$:     MOV     PAT.PT(R0),R3 ; PICKUP PATTERN POINTER
6061 032620 012321  MOV     (R3)+,(R1)+ ; MOVE WORD 1 INTO DATA BUFFER
6062 032622 012321  MOV     (R3)+,(R1)+ ; MOVE WORD 2 INTO DATA BUFFER
6063 032624 012321  MOV     (R3)+,(R1)+ ; MOVE WORD 3 INTO DATA BUFFER
6064 032626 012321  MOV     (R3)+,(R1)+ ; MOVE WORD 4 INTO DATA BUFFER

```



```

6121 032776 162321 SUB (R3)+,(R1)+ ;CHECK WORD 14
6122 033000 001012 BNE 4$ ;BRANCH IF DIFFERENT
6123 033002 162321 SUB (R3)+,(R1)+ ;CHECK WORD 15
6124 033004 001010 BNE 4$ ;BRANCH IF DIFFERENT
6125 033006 162321 SUB (R3)+,(R1)+ ;CHECK WORD 16
6126 033010 001006 BNE 4$ ;BRANCH IF DIFFERENT
6127 033012 062702 000020 ADD #16.,R2 ;DONE ?
6128 033016 001333 BNE 1$ ;NO--BRANCH
6129 033020 005726 3$: TST (SP)+ ;YES -- CLEAN UP STACK
6130 033022 104413 RESREG ;RESTORE R0 - R5
6131 033024 000207 RTS PC
6132 033026 010104 4$: MOV R1,R4 ;FORM THE WORD NUMBER
6133 033030 162704 051776 SUB #BUFFER,R4
6134 033034 006204 ASR R4 ;WORD NUMBER
6135 033036 010305 MOV R3,R5 ;FORM ADDRESS TO CONTINUE FROM
6136 033040 166005 003146 SUB PAT.PT(R0),R5
6137 033044 006305 ASL R5
6138 033046 062705 032712 ADD #2$,R5 ;ADDRESS
6139 033052 064341 ADD -(R3),-(R1) ;RECONSTRUCT THE BAD WORD
6140 033054 010137 001122 MOV R1,@#$BDADR ;SAVE THE ERROR INFORMATION
6141 033060 010337 001120 MOV R3,@#$GDADR
6142 033064 012137 001126 MOV (R1)+,@#$BDDAT
6143 033070 012337 001124 MOV (R3)+,@#$GDDAT
6144 033074 005716 TST (SP) ;1ST DATA COMPARE ERROR?
6145 033076 001023 BNE 6$ ;NO--BRANCH
6146 033100 013737 004300 001350 MOV @#DTADPB+12,@#CYL.DS ;CYLINDER
6147 033106 113737 004277 001354 MOVB @#DTADPB+11,@#TRK.DS ;TRACK
6148 033114 113737 004276 001352 MOVB @#DTADPB+10,@#SEC.DS ;SECTOR
6149 033122 016600 000016 MOV 16(SP),R0 ;GET TEST PC+4
6150 033126 012737 033136 001206 MOV #5$,SE$CAPE ;ESCAPE TO 5$ ON ERROR
6151 033134 104013 ERROR 13 ;REPORT THE ERROR
6152 033136 016600 000014 5$: MOV 14(SP),R0 ;PATTERN NUMBER INDEX
6153 033142 105116 COMB (SP) ;SET THE ERROR SWITCH
6154 033144 000404 BR 7$
6155 033146 6$: MOV #7$,SE$CAPE ;ESCAPE TO 7$ ON ERROR
6156 033146 012737 033156 001206 ERROR 14 ;REPORT THE ERROR
6157 033154 104014 7$: BIT #SW01,@$WR ;STOP DATA COMPARE?
6158 033156 032777 000002 145754 BNE 3$ ;YES--EXIT
6159 033164 001315 CMPB @#ERR.CT,@#$ERFLG ;MAX. ERRORS?
6160 033166 123737 001444 001103 BHI 8$ ;NO--BRANCH
6161 033174 101004 MOV @#BYPASS,16(SP) ;YES--ERROR EXIT
6162 033176 013766 001332 000016 BR 3$
6163 033204 000705 8$: JMP (R5) ;NO--CONTINUE AT NEXT WORD
6164 033206 000115
6165
6166 ; THIS ROUTINE WILL FILL THE DATA BUFFER (256*22 WORDS) WITH
6167 ; A RANDOM PATTERN. THE FIRST TWO WORDS OF EVERY 256 WILL
6168 ; BE THE BASE OF THE RANDOM NUMBER GENERATOR FOR THE
6169 ; NEXT 254 WORDS.
6170 ; NOTE: THIS ROUTINE DESTROYS R1 AND R2
6171 ; CALL
6172 ; JSR R0,@#FILRAN
6173 ; RETURN
6174
6175 033210 012701 051776 FILRAN: MOV #BUFFER,R1
6176 033214 013702 001712 MOV PRMLMT+22,R2 ;MAXIMUM NUMBER OF SECTORS

```

6177 033220 004037 033430
6178 033224 005302
6179 033226 100374
6180 033230 000200
6181
6182
6183
6184
6185
6186
6187
6188
6189
6190 033232 013746 025630
6191 033236 013746 025632
6192 033242 012702 052776
6193 033246 012701 053776
6194 033252 010103
6195 033254 011237 025632
6196 033260 016237 000002 025630
6197 033266 004037 033430
6198 033272 012637 025632
6199 033276 012637 025630
6200 033302 005046
6201 033304 162322
6202 033306 001441
6203 033310 012737 033362 001206
6204 033316 064342
6205 033320 010237 001122
6206 033324 010337 001120
6207 033330 012237 001126
6208 033334 012337 001124
6209 033340 010204
6210 033342 162704 052776
6211 033346 006204
6212 033350 005716
6213 033352 001002
6214 033354 105116
6215 033356 104015
6216 033360 104016
6217 033362 032777 001000 145550
6218 033370 001012
6219 033372 123737 001444 001103
6220 033400 101406
6221 033402 032777 000002 145530
6222 033410 001002
6223 033412 020103
6224 033414 101333
6225 033416 005726
6226 033420 001402
6227 033422 013700 001332
6228 033426 000200
6229
6230
6231
6232

```
1$: JSR RO,@#RANPAT
    DEC R2
    BPL 1$
    RTS RO

; THIS ROUTINE USES THE FIRST TWO WORDS OF THE
; READ BUFFER TO GENERATED A RANDOM PATTERN. THEN
; THE READ BUFFER IS COMPARED TO THE PATTERN GENERATED.
; NOTE: THIS ROUTINE DESTROYS R1-R4
; CALL
; JSR RO,@#RANCK
; RETURN

RANCK: MOV @#$HINUM,-(SP) ;SAVE THE PRESENT RANDOM NUMBER
      MOV @#$LONUM,-(SP)
      MOV #BUFFER+512.,R2 ;READ BUFFER ADDRESS
      MOV #BUFFER+1024.,R1 ;RANDOM PATTERN ADDRESS
      MOV R1,R3 ;COPY IT INTO R3 FOR LATER USE
      MOV (R2),@#$LONUM ;PRIME THE RANDOM NUMBER GENERATOR
      MOV 2(R2),@#$HINUM
      JSR RO,@#RANPAT ;GENERATE A RANDOM PATTERN
      MOV (SP)+,@#$LONUM ;RESTORE PRESENT RANDOM NUMBER
      MOV (SP)+,@#$HINUM
      CLR -(SP) ;NO ERRORS
1$: SUB (R3)+,(R2)+ ;ARE THESE TWO WORDS DIFFERENT?
    BEQ 4$ ;NO--BRANCH
    MOV #3$,$ESCAPE ;ESCAPE TO 3$ ON ERROR
    ADD -(R3),-(R2) ;RECREATE THE BAD WORD
    MOV R2,@#$BDADR ;ADDRESS OF BAD DATA
    MOV R3,@#$GDADR ;ADDRESS OF GOOD DATA
    MOV (R2)+,@#$BDADR ;BAD DATA
    MOV (R3)+,@#$GDADR ;GOOD DATA
    MOV R2,R4 ;FORM WORD NUMBER (1 TO 256)
    SUB #BUFFER+512.,R4
    ASR R4
    TST (SP) ;FIRST ERROR
    BNE 2$ ;NO--BRANCH
    COMB (SP) ;YES--SET ERROR SWITCH
    ERROR 15 ;REPORT THE ERROR
    ERROR 16 ;REPORT THE ERROR
3$: BIT #SW09,@SWR ;LOOP ON ERROR?
    BNE 5$ ;YES--BRANCH
    CMPB @#ERR.CT,@#SERFLG ;MAX. ERRORS OCCURRED?
    BLOS 5$ ;YES--BRANCH
    BIT #SW01,@SWR ;STOP COMPARING?
    BNE 5$ ;YES--BRANCH
4$: CMP R1,R3 ;ALL DATA BEEN COMPARED?
    BHI 1$ ;NO--BRANCH
5$: TST (SP)+ ;ERROR OCCUR?
    BEQ 6$ ;NO--BRANCH
6$: MOV @#BYPASS,RO ;TAKE ERROR EXIT
    RTS RO ;EXIT
```

; THIS ROUTINE FILLS A 256 WORD BUFFER WITH A RANDOM
; PATTERN OF WHICH THE FIRST TWO WORDS ARE THE BASE
; OF THE PATTERN.


```

6233 ;CALL
6234 ;   MOV   #ADR,R1      ;ADDRESS OF THE BUFFER
6235 ;   JSR   RO,@#RANPAT
6236 ;   RETURN
6237
6238 RANPAT: MOV   R2,-(SP)   ;SAVE R2
6239 033430 010246 000200   MOV   #256./2.,R2    ;GENERATE 256 WORDS
6240 033432 012702 000402   BR    2$
6241 033436 000402 025532   1$: JSR   PC,@#SRAND   ;GENERATE A RANDOM NUMBER
6242 033440 004737 025632   2$: MOV   @#$LONUM,(R1)+ ;PUT LOW WORD IN BUFFER
6243 033444 013721 025630   MOV   @#$HINUM,(R1)+  ;PUT HIGH WORD IN BUFFER
6244 033450 013721 025630   DEC   R2              ;DONE?
6245 033454 005302         BGT   1$              ;NO--BRANCH
6246 033456 003370         MOV   (SP)+,R2       ;RESTORE R2
6247 033460 012602         RTS    RO            ;EXIT
6248 033462 000200
6249
6250 ;THIS ROUTINE GENERATES RANDOM CYLINDER, TRACK, AND SECTOR
6251 ;ADDRESSES AND SAVES THEM IN THE DPB (DTADPB+10, 11 & DTADPB+12).
6252 ;NOTE: THIS ROUTINE DESTROYS R1-R3
6253 ;CALL
6254 ;   JSR   RO,@#RANADR
6255 ;   RETURN
6256
6257 RANADR: JSR   PC,@#SRAND   ;GENERATE A RANDOM NUMBER
6258 033470 113701 025632   MOVB  @#$LONUM,R1     ;FORM SECTOR IN R1
6259 033474 113702 025633   MOVB  @#$LONUM+1,R2   ;FORM TRACK IN R2
6260 033500 013703 025630   MOV   @#$HINUM,R3    ;FORM CYLINDER IN R3
6261 033504 105701         TSTB  R1              ;ENSURE THE SECTOR IS BETWEEN 0 AND 31
6262 033506 002403         BLT   2$
6263 033510 123701 001712   1$: CMPB  PRMLT+22,R1   ;CHECK MAXIMUM SECTOR ADDRESS
6264 033514 103003         BHIS  3$
6265 033516 000241         CLC
6266 033520 106001         RORB  R1
6267 033522 000772         BR    1$
6268 033524 105702         TSTB  R2              ;ENSURE THE TRACK IS BETWEEN 0 AND 4
6269 033526 002403         BLT   5$
6270 033530 122702 000005   4$: CMPB  #5,R2
6271 033534 003003         BGT   6$
6272 033536 000241         CLC
6273 033540 106002         RORB  R2
6274 033542 000772         BR    4$
6275 033544 023703 001572   6$: CMP   @#FC,R3        ;ENSURE THE CYLINDER IS BETWEEN FC AND LC
6276 033550 003413         BLE  7$
6277 033552 000241         CLC
6278 033554 006003         ROR   R3
6279 033556 005503         ADC   R3
6280 033560 001371         BNE  6$
6281 033562 010103         MOV   R1,R3
6282 033564 000303         SWAB R3
6283 033566 060203         ADD  R2,R3
6284 033570 005203         INC  R3
6285 033572 003364         BGT  6$
6286 033574 005403         NEG  R3
6287 033576 000762         BR   6$
6288 033600 023703 001574   7$: CMP   @#LC,R3
6289 033604 002003         BGE  8$
    
```

```

6289 033606 000241          CLC
6290 033610 006003          ROR      R3
6291 033612 000772          BR       7$
6292 033614 023703 001572    8$:     CMP     @#FC,R3
6293 033620 003403          BLE     9$
6294 033622 005203          INC     R3
6295 033624 000303          SWAB   R3
6296 033626 000764          BR       7$
6297 033630 110137 004276    9$:     MOVB   R1,@#DTADPB+10 ;SAVE SECTOR ADDRESS
6298 033634 110237 004277    MOVB   R2,@#DTADPB+11 ;SAVE TRACK ADDRESS
6299 033640 010337 004300    MOV     R3,@#DTADPB+12 ;SAVE CYLINDER ADDRESS
6300 033644 000200          RTS     R0 ;RETURN
6301
6302 ; THIS ROUTINE IS USED TO INPUT THE "CONTROL SWITCHES".
6303 ; IF SWR<07>=1 THE PRESENT SETTING WILL BE TYPED AND THE NEW
6304 ; SETTING IS READ AND STORED.
6305 ; NOTE: THIS ROUTINE DESTROYS R3 AND R4
6306 ; CALL
6307 ; JSR     PC,@#GETSWR
6308 ; RETURN ; (C.SWR)=DESIRED CONTROL SWITCHES
6309
6310 033646 032777 000200 145264 GETSWR: BIT     #SW07,@SWR ;READ CONTROL SWITCHES?
6311 033654 001430          BEQ     2$ ;NO--BRANCH
6312 033656 104401 033664          TYPE   65$ ;TYPE ASCIZ STRING
6313 033662 000410          BR     64$ ;GET OVER THE ASCIZ
6314 ;:65$: .ASCIZ <CR><LF>/SET SWR<07>=0/
6315 033704          64$:
6316 033704 012703 045154    1$:     MOV     #MSG.CS,R3 ;"CONTROL SWITCHES="
6317 033710 013704 001300    MOV     @#C.SWR,R4 ;PRESENT CONTROL SWITCH SETTINGS
6318 033714 004037 033740          JSR     R0,@#GETNUM ;GET THE NEW SWITCH SETTINGS
6319 033720 000771          BR     1$ ;COMMA
6320 033722 000240          NOP    ;PERIOD
6321 033724 013737 001300 001302    MOV     C.SWR,SAVCSW ;SAVE PREVIOUS VALUE
6322 033732 010437 001300    MOV     R4,@#C.SWR ;DOUBLE PERIOD--SAVE NEW SWITCH SETTING
6323 033736 000207    2$:     RTS     PC ;RETURN FROM CALL
6324
6325 ; THIS ROUTINE WILL TYPE AN ASCIZ MESSAGE AND THEN
6326 ; INPUT AN ASCIZ STRING AND CHANGE THE STRING TO OCTAL
6327 ; IF REQUIRED.
6328 ; NOTE: THIS ROUTINE DESTROYS R1
6329 ; CALL
6330 ; MOV     #ADR,R3 ;ADDRESS OF ASCIZ MESSAGE
6331 ; MOV     #NUM,R4 ;OCTAL NUMBER
6332 ; JSR     R0,@#GETNUM
6333 ; RETURN1 ;INPUT TERMINATED WITH A COMMA
6334 ; RETURN2 ;WITH A PERIOD
6335 ; RETURN3 ;WITH A DOUBLE PERIOD
6336 ; R4=INPUT NUMBER AND
6337 ; R2=R4*32 FOR ALL
6338 ; THREE RETURNS
6339
6340 033740 010337 033746    GETNUM: MOV     R3,2$ ;SAVE MESSAGE POINTER
6341 033744 104401          1$:     TYPE   ;TYPE THE MESSAGE
6342 033746 000000          2$:     .WORD 0 ;MESSAGE POINTER GOES HERE
6343 033750 010446          MOV     R4,-(SP) ;SAVE R4 FOR TYPEOUT
6344 033752 104402          TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
    
```



```

6345 033754 104401 045200      TYPE      ,SLASH      ;' / '
6346 033760 104411      RDLIN      ;READ AN ASCIZ STRING
6347 033762 012601      MOV        (SP)+,R1    ;ADDRESS OF FIRST CHARACTER
6348 033764 004037 036210      JSR        RO, @#CK.CHR ;CHECK ONE CHARACTER
6349 033770 033744      1$        ;ILLEGAL CHARACTER
6350 033772 033744      1$        ;CARRIAGE RETURN
6351 033774 034006      3$        ;"/
6352 033776 034032      7$        ;"
6353 034000 034040      8$        ;"
6354 034002 034004      11$       ;DIGIT 0-9
6355 034004 005301      11$: DEC    R1      ;DECREMENT THE INPUT POINTER
6356 034006      3$:
6357 034006 004037 036450      JSR        RO, @#CK.NUM ;CHECK THE NUMBER
6358 034012 033744      1$        ;ILLEGAL INPUT
6359 034014 034026      6$        ;TERMINATED WITH A "," OR "CR"
6360 034016 034024      5$        ;TERMINATED WITH A "."
6361 034020 034022      4$        ;TERMINATED WITH A ".."
6362 034022 005720      4$: TST    (RO)+    ;DOUBLE PERIOD
6363 034024 005720      5$: TST    (RO)+    ;SINGLE PERIOD
6364 034026 010204      6$: MOV    R2,R4    ;COMMA--SAVE INPUT NUMBER
6365 034030 000414      BR        10$       ;GO TO EXIT
6366 034032 105711      7$: TSTB   (R1)      ;TERMINATOR AFTER A COMMA?
6367 034034 001343      BNE     1$        ;NO--LOOP
6368 034036 000411      BR        10$       ;YES--EXIT
6369 034040 105711      8$: TSTB   (R1)      ;TERMINATOR AFTER A PERIOD?
6370 034042 001406      BEQ     9$        ;YES--EXIT
6371 034044 122721 000056      CMPB   #'.,(R1)+    ;NO--DOUBLE PERIOD?
6372 034050 001335      BNE     1$        ;NO--LOOP
6373 034052 105711      TSTB   (R1)      ;YES--TERMINATOR?
6374 034054 001333      BNE     1$        ;NO--LOOP
6375 034056 005720      TST    (RO)+    ;DOUBLE PERIOD
6376 034060 005720      9$: TST    (RO)+    ;PERIOD
6377 034062 010402      10$: MOV   R4,R2    ;COMMA--POSITION THE
6378 034064 000302      SWAB   R2        ;NUMBER IN CASE IT
6379 034066 006202      ASR    R2        ;IS THE PRIORITY LEVEL
6380 034070 006202      ASR    R2
6381 034072 006202      ASR    R2
6382 034074 000200      RTS     R0        ;EXIT
6383
6384      ;THIS ROUTINE IS USED TO CHANGE OR MODIFY
6385      ;THE TEST PARAMETERS. IT GIVES THE OPERATOR
6386      ;THE CAPABILITY OF SPECIFYING WHICH DRIVES TO TEST, WHICH
6387      ;TESTS TO RUN AND HOW MANY TIMES TO
6388      ;REPEAT EACH TEST
6389
6390 034076 104412      GT.PRM: SAVREG      ;SAVE R0 - R5
6391 034100 005037 001312      GT.PRI: CLR        DRVSEL ;NO DRIVE SELECTED
6392 034104 104401 034112      TYPE     65$      ;TYPE ASCIZ STRING
6393 034110 000406      BR        64$      ;GET OVER THE ASCIZ
6394
6395      ;:65$: .ASCIZ <CR><LF>/DRIVE(S)=/
6396      64$:
6397 034126 104411      RDLIN      ;READ TTY
6398 034130 012601      MOV        (SP)+,R1    ;ADDRESS OF ASCIZ STRING
6399 034132 004037 036210      JSR        RO, @#CK.CHR ;CHECK ONE CHARACTER
6400 034136 034100      GT.PRI     ;ILLEGAL CHARACTER
6400 034140 034100      GT.PRI     ;CARRIAGE RETURN

```

```

6401 034142 034100 GT.PRI
6402 034144 034100 GT.PRI
6403 034146 034100 GT.PRI
6404 034150 034152 1$ :
6405 034152 005301 2$ : DEC R1 ;DIGIT 0-9
6406 034154
6407 034154 012702 000007 MOV #7,R2 ;UPPER LIMIT OF INPUT
6408 034160 004037 036264 JSR RO,#CK.DIG ;CHECK THE DIGIT(S)
6409 034164 034100 GT.PRI ;ILLEGAL INPUT
6410 034166 034100 GT.PRI ;INPUT TO LARGE
6411 034170 034176 3$ ;TERMINATED WITH A ". " OR "CR"
6412 034172 034222 4$ ;TERMINATED WITH A ". "
6413 034174 034222 4$ ;TERMINATED WITH A ". "
6414 034176 156237 036724 001312 3$ : BISB ATABIT(R2),DRVSEL ;SET THE DRIVE SELECTED BIT
6415 034204 105741 TSTB -(R1) ;WAS THE LINE TERMINATED?
6416 034206 001362 BNE 2$ ;NO-GET THE NEXT DRIVE
6417 034210 005037 001314 CLR #TSTNMS ;DESELECT ALL TESTS
6418 034214 005037 001316 CLR TSTNMS+2
6419 034220 000405 BR GTTST1 ;YES--SELECT TEST
6420 034222 156237 036724 001312 4$ : BISB ATABIT(R2),DRVSEL ;SET THE SELECTED DRIVE BITS
6421 034230 104413 GT.PR2: RESREG ;RESTORE RO - RS
6422 034232 000207 RTS PC ;EXIT
6423
6424 034234 GTTST1:
6425 034234 104401 034242 TYPE 65$ ;:TYPE ASCIZ STRING
6426 034240 070403 BR 64$ ;:GET OVER THE ASCIZ
6427 ;:65$: .ASCIZ /TEST=/
6428 64$:
6429 034250 104411 RDLIN ;:READ AN ASCIZ STRING
6430 034252 012601 MOV (SP)+,R1 ;:POINTER TO R1
6431 034254 122711 000056 CMPB #'.,(R1) ;:DOUBLE PERIOD?
6432 034260 001007 BNE 1$ ;:NO--BRANCH
6433 034262 122761 000056 000001 CMPB #'.,1(R1)
6434 034270 001003 BNE 1$
6435 034272 105761 000002 TSTB 2(R1) ;:"CR"?
6436 034276 001754 BEQ GT.PR2 ;:YES--EXIT
6437 034300 005037 001314 1$ : CLR TSTNMS ;:NO TEST SELECTED
6438 034304 005037 001316 CLR TSTNMS+2
6439 034310 005037 001320 CLR OPNFLG ;:NO TESTS TO BE OPENED
6440 034314 005037 001322 CLR OPNFLG+2
6441 034320 121127 000123 GTTST2: CMPB (R1),#'S ;:ALL SEEK TESTS?
6442 034324 001004 BNE 1$ ;:NO--BRANCH
6443 034326 052737 000777 001314 BIS #777,TSTNMS ;:YES--SELECT TESTS 0-10
6444 034334 000552 BR GTTST3
6445 034336 121127 000124 1$ : CMPB (R1),#'T ;:ALL TIMING TESTS?
6446 034342 001004 BNE 2$ ;:NO--BRANCH
6447 034344 052737 036000 001314 BIS #36000,TSTNMS ;:YES--SELECT TESTS 12-15
6448 034352 000543 BR GTTST3
6449 034354 121127 000101 2$ : CMPB (R1),#'A ;:ALL ADDRESSING TESTS?
6450 034360 001004 BNE 3$ ;:NO--BRANCH
6451 034362 052737 140000 001314 BIS #140000,TSTNMS ;:YES--SELECT TESTS 16 & 17
6452 034370 000534 BR GTTST3
6453 034372 121127 000104 3$ : CMPB (R1),#'D ;:DATA TEST?
6454 034376 001004 BNE 4$ ;:NO--BRANCH
6455 034400 052737 000001 001316 BIS #1,TSTNMS+2 ;:YES--SELECT TEST 20
6456 034406 000525 BR GTTST3

```



```

6457 034410 121127 000105 4$: CMPB (R1),#'E ;EXERCISER TEST?
6458 034414 001004 BNE 5$ ;NO--BRANCH
6459 034416 052737 000002 001316 BIS #2,TSTNMS+2 ;YES--SELECT TEST 21
6460 034424 000516 BR GTTST3
6461 034426 004037 036134 5$: JSR R0,@#CK.OCT ;OCTAL DIGIT?
6462 034432 000514 BR GTTST4 ;NO--BRANCH
6463 034434 010205 MOV R2,R5 ;YES--SAVE IT
6464 034436 005201 INC R1 ;MOVE TO NEXT CHARACTER
6465 034440 004037 036134 JSR R0,@#CK.OCT ;OCTAL DIGIT
6466 034444 000405 BR 6$ ;NO--BRANCH
6467 034446 005201 INC R1 ;MOVE TO NEXT CHARACTER
6468 034450 006305 ASL R5 ;SCALE HIGH DIGIT
6469 034452 006305 ASL R5
6470 034454 006305 ASL R5
6471 034456 060502 ADD R5,R2 ;COMBINE HIGH & LOW DIGITS
6472 034460 020227 000022 6$: CMP R2,#$TN-1 ;VALID TEST NUMBER?
6473 034464 0032E3 BGT GTTST1 ;NO--BRANCH
6474 034466 0102J7 034660 MOV R2,9$ ;SAVE THE TEST NUMBER
6475 034472 010204 MOV R2,R4 ;CONVERT TEST NUMBER INTO AN INDEX
6476 034474 042704 000017 BIC #17,R4 ;CLEAR UNWANTED BITS
6477 034500 006204 ASR R4 ;SHIFT THE BITS
6478 034502 006204 ASR R4 ;SHIFT THE BITS
6479 034504 006204 ASR R4 ;SHIFT THE BITS
6480 034506 006302 ASL R2
6481 034510 056264 001506 001314 BIS BITS(R2),TSTNMS(R4) ;SELECT TEST
6482 034516 121127 000055 CMPB (R1),#'E ;TEST STRING?
6483 034522 001060 BNE GTTST4 ;NO--BRANCH
6484 034524 005201 INC R1 ;YES--MOVE TO NEXT CHARACTER
6485 034526 004037 036134 JSR R0,@#CK.OCT ;OCTAL DIGIT?
6486 034532 000640 BR GTTST1 ;NO--BRANCH
6487 034534 010205 MOV R2,R5 ;YES--SAVE IT
6488 034536 005201 INC R1 ;MOVE TO NEXT CHARACTER
6489 034540 004037 036134 JSR R0,@#CK.OCT ;OCTAL DIGIT?
6490 034544 000405 BR 7$ ;NO--BRANCH
6491 034546 005201 INC R1 ;YES--MOVE TO NEXT CHARACTER
6492 034550 006305 ASL R5 ;SCALE HIGH DIGIT
6493 034552 006305 ASL R5
6494 034554 006305 ASL R5
6495 034556 060502 ADD R5,R2 ;COMBINE HIGH & LOW DIGIT
6496 034560 020227 000022 7$: CMP R2,#$TN-1 ;VALID TEST NUMBER?
6497 034564 003223 BGT GTTST1 ;NO--BRANCH
6498 034566 023702 034660 CMP 9$,R2 ;IS THE FIRST NUMBER OF THE
6499 ;STRING SMALLER THAN THE LAST?
6500 BGE GTTST1 ;NO--BRANCH
6501 034574 010246 MOV R2,-(SP) ;SAVE ENDING TEST NUMBER
6502 034576 013702 034660 MOV 9$,R2 ;GET STARTING TEST NUMBER
6503 034602 012637 034660 MOV (SP)+,9$ ;STORE ENDING TEST NUMBER
6504 034606 006337 034660 ASL 9$ ;SHIFT ENDING TEST NUMBER
6505 034612 006302 ASL R2 ;SHIFT TEST NUMBER
6506 034614 010204 8$: MOV R2,R4 ;COPY TEST NUMBER INTO R4
6507 034616 042704 000037 BIC #37,R4 ;CLEAR LOWER BITS
6508 034622 006204 ASR R4 ;SHIFT THE TEST NUMBER
6509 034624 006204 ASR R4 ;SHIFT THE TEST NUMBER
6510 034626 006204 ASR R4 ;SHIFT THE TEST NUMBER
6511 034630 006204 ASR R4 ;SHIFT THE TEST NUMBER
6512 034632 056264 001506 001314 BIS BITS(R2),TSTNMS(R4) ;SELECT THE TEST

```



```

6513 034640 062702 000002          ADD      #2,R2          ; INCREMENT THE TEST NUMBER
6514 034644 020237 034660          CMP      R2,R5          ; SEE IF FINISHED
6515 034650 101761          BLOS     8$             ; BR IF NOT
6516 034652 162702 000002          SUB      #2,R2          ; CORRECT TEST NUMBER
6517 034656 000402          BR      GTTST4         ; CONTINUE
6518 034660 000000          .WORD   0              ; STORE TEST NUMBER HERE
6519 034662 005201          GTTST3: INC     R1          ; MOVE TO NEXT CHARACTER
6520 034664 121127 000056          GTTST4: CMPB   (R1),#'.' ; "PERIOD"?
6521 034670 001441          BEQ     GTTST5         ; YES--BRANCH
6522 034672 005737 001314          TST     TSTNMS         ; ANY TEST SELECTED THIS CYCLE?
6523 034676 001005          BNE     1$             ; BR IF YES
6524 034700 005737 001316          TST     TSTNMS+2       ; ANY TEST SELECTED THIS CYCLE ?
6525 034704 001002          BNE     1$             ; BR IF YES
6526 034706 000137 034234          JMP     GTTST1         ; NO
6527 034712 121127 000057          1$:    CMPB   (R1),#'/' ; "OPEN"?
6528 034716 001004          BNE     2$             ; NO--BRANCH
6529 034720 056264 001506 001320          BIS     BITS(R2), OPNFLG(R4) ; YES--SET BITS FOR TEST TO OPEN
6530 034726 000405          BR      3$             ;
6531 034730 121127 000054          2$:    CMPB   (R1),#',' ; "COMMA"?
6532 034734 001402          BEQ     3$             ; BR IF YES
6533 034736 000137 034234          JMP     GTTST1         ; NO
6534 034742 005201          3$:    INC     R1          ; MOVE TO NEXT CHARACTER
6535 034744 105711          TSTB   (R1)           ; "CR"?
6536 034746 001402          BEQ     4$             ; BR IF 'CR'
6537 034750 000137 034320          JMP     GTTST2         ; NO--GO GET NEXT CHARACTER
6538 034754 005737 001320          4$:    TST     OPNFLG     ; ANY TESTS TO OPEN ?
6539 034760 001042          BNE     OPNTST         ; BR IF YES
6540 034762 005737 001322          TST     OPNFLG+2       ; ANY TESTS TO OPEN ?
6541 034766 001037          BNE     OPNTST         ; BR IF YES
6542 034770 000137 034234          JMP     GTTST1         ; NO--START AGAIN
6543 034774 005201          GTTST5: INC     R1          ; MOVE TO NEXT CHARACTER
6544 034776 121127 000056          CMPB   (R1),#'.' ; "PERIOD"?
6545 035002 001414          BEQ     GTTST6         ; YES--BRANCH
6546 035004 105711          TSTB   (R1)           ; "CR"?
6547 035006 001402          BEQ     1$             ; YES--BRANCH
6548 035010 000137 034234          JMP     GTTST1         ;
6549 035014 005737 001320          1$:    TST     OPNFLG     ; ANY TESTS TO OPEN ?
6550 035020 001022          BNE     OPNTST         ; BR IF YES
6551 035022 005737 001322          TST     OPNFLG+2       ; ANY TESTS TO OPEN ?
6552 035026 001017          BNE     OPNTST         ; BR IF YES
6553 035030 000137 034230          JMP     GT.PR2         ; NO--GO START TESTING
6554 035034 005201          GTTST6: INC     R1          ; MOVE TO NEXT CHARACTER
6555 035036 105711          TSTB   (R1)           ; "CR"?
6556 035040 001402          BEQ     1$             ; YES--BRANCH
6557 035042 000137 034234          JMP     GTTST1         ; NO--GO ASK FOR TEST
6558 035046 005737 001320          1$:    TST     OPNFLG     ; ANY TESTS TO OPEN ?
6559 035052 001005          BNE     OPNTST         ; BR IF YES
6560 035054 005737 001322          TST     OPNFLG+2       ; ANY TESTS TO OPEN ?
6561 035060 001002          BNE     OPNTST         ; BR IF YES
6562 035062 000137 034230          JMP     GT.PR2         ; NO--GO START TESTING
6563
6564 ; OPEN THE SELECTED TEST FOR CHANGES
6565
6566 035066 104412          OPNTST: SAVREG          ; SAVE R0 - R5
6567 035070 005027          CLR     (PC)+          ; START WITH TEST 0
6568 035072 000000          OPN.CT: .WORD 0        ; COUNT STORED HERE

```



```

6569 035074 000411 BR OPN.2 ;SKIP THE INCREMENT
6570 035076 005237 035072 OPN.1: INC OPN.CT ;MOVE TO THE NEXT TEST
6571 035102 022737 000022 035072 CMP #STN-1,OPN.CT ;TEST NUMBER TOO BIG?
6572 035110 002003 BGE OPN.2 ;NO--OPEN THE NEXT TEST
6573 035112 104413 RESREG ;RESTORE R0 - R5
6574 035114 000137 034234 JMP GTTST1 ;YES--GO ASK FOR MORE TESTS
6575 035120 013705 035072 OPN.2: MOV OPN.CT,R5 ;SETUP TO USE THE
6576 035124 006305 ASL R5 ;TEST NUMBER AS AN INDEX
6577 035126 013703 035072 MOV OPN.CT,R3 ;GET INDEX
6578 035132 042703 000017 BIC #17,R3 ;CLEAR LOWER TEST BITS
6579 035136 006203 ASR R3 ;SHIFT TEST NUMBER
6580 035140 006203 ASR R3 ;SHIFT TEST NUMBER
6581 035142 006203 ASR R3 ;SHIFT TEST NUMBER
6582 035144 036563 001506 001320 BIT BITS(R5),OPNFLG(R3) ;OPEN THIS TEST?
6583 035152 001751 BEQ OPN.1 ;NO--MOVE TO NEXT TEST
6584 035154 104401 035162 TYPE ,65$ ;TYPE ASCIZ STRING
6585 035160 000404 BR ,64$ ;GET OVER THE ASCIZ
6586 ;:65$: .ASCIZ / TEST /
6587 035172 ;64$:
6588 035172 013746 035072 MOV OPN.CT,-(SP) ;SAVE OPN.CT FOR TYPEOUT
6589 ;TEST NUMBER
6590 035176 104403 TYPOS ;GO TYPE--OCTAL ASCII
6591 035200 002 .BYTE 2 ;TYPE 2 DIGIT(S)
6592 035201 000 .BYTE 0 ;SUPPRESS LEADING ZEROS
6593 035202 104401 001215 TYPE ,SCRFL ;TYPE "CR" & "LF"
6594 035206 016500 001620 MOV PRMPT(R5),R0 ;PICKUP PARAMETER POINTER
6595 035212 011046 MOV (R0),-(SP) ;SAVE THE VARIABLE INDICATOR
6596 035214 012702 001566 MOV #PRM,R2 ;FIRST ADDRESS OF TABLE
6597 035220 000405 BR 2$
6598 035222 006216 1$: ASR (SP) ;CHECK FOR A VARIABLE
6599 035224 103403 BCS 2$ ;GO MOVE THIS ONE
6600 035226 001404 BEQ OPNPRM ;DONE
6601 035230 005722 TST (R2)+ ;BUMP THE POINTER
6602 035232 000773 BR 1$
6603 035234 012022 2$: MOV (R0)+,(R2)+ ;MOVE THIS VARIABLE INTO THE
6604 035236 000771 BR 1$ ;COMMON AREA
6605 035240 013716 001566 OPNPRM: MOV #PRM,(SP) ;GET THE VARIABLE INDICATOR
6606 035244 005004 CLR R4 ;ZERO THE INDEX
6607 035246 006216 1$: ASR (SP) ;CHECK FOR A VARIABLE
6608 035250 103403 BCS 3$ ;GO GET IT
6609 035252 001772 BEQ OPNPRM ;OUT OF VARIABLES
6610 035254 005724 2$: TST (R4)+ ;UPDATE THE INDEX
6611 035256 000773 BR 1$
6612 035260 005764 001670 3$: TST PRMLMT(R4) ;IS THE MAX. MAGNITUDE NEG?
6613 035264 100466 BMI OPNPAT ;YES--THEN IT IS THE PATTERN
6614 035266 104401 046141 TYPE ,MSG.SP ;TYPE SPACES
6615 035272 016437 001720 035302 MOV PRMMSG(R4),4$ ;TYPE THE NAME OF THIS VARIABLE
6616 035300 104401 TYPE
6617 035302 000000 4$: .WORD 0
6618 035304 104401 TYPE ,MSG.EQ ;TYPE "="
6619 035310 016446 001570 MOV RPT(R4),-(SP) ;PUT RPT(R4) ON THE STACK
6620 035314 004737 025242 JSR PC,#$$S20 ;CHANGE RPT(R4) TO DECIMAL ASCII
6621 035320 004737 025472 JSR PC,#$$SUPRS ;TYPE WITHOUT LEADING ZEROS
6622 035324 104401 045200 TYPE ,SLASH ;' / '
6623 035330 104411 RDLIN
6624 035332 012601 MOV (SP)+,R1 ;READ AN ASCIZ STRING

```


6625	035334	004037	036210		JSR	RO, @#CK.CHR	:CHECK ONE CHARACTER
6626	035340	035260			3\$:ILLEGAL CHARACTER
6627	035342	035254			2\$:CARRIAGE RETURN
6628	035344	035412			8\$:"/
6629	035346	035354			5\$:""
6630	035350	035362			6\$:""
6631	035352	035410			7\$:DIGIT 0-9
6632	035354	105711		5\$:	TSTB	(R1)	:"CR"?
6633	035356	001340			BNE	3\$:NO--STAY ON THIS VARIABLE
6634	035360	000735			BR	2\$:YES--MOVE TO NEXT VARIABLE
6635	035362	105711		6\$:	TSTB	(R1)	:IS THERE A "CR" AFTER THE PERIOD?
6636	035364	001002			BNE	64\$:NO
6637	035366	000137	036002		JMP	OPN.N2	:YES--GO CLOSE THIS TEST
6638	035372	122721	000056	64\$:	CMPB	#'.,(R1)+	:DOUBLE PERIOD?
6639	035376	001330			BNE	3\$:NO--GO ASK FOR THIS VARIABLE
6640	035400	105711			TSTB	(R1)	:YES--IS A "CR" AFTER THE DOUBLE PERIOD?
6641	035402	001326			BNE	3\$:NO--ASK FOR THIS VARIABLE AGAIN
6642	035404	000137	036020		JMP	OPN.X2	:YES--CLOSE ALL TEST
6643	035410	005301		7\$:	DEC	R1	:BACK THE POINTER UP BY ONE
6644	035412			8\$:			
6645	035412	016402	001670		MOV	PRMLMT(R4) R2	:UPPER LIMIT OF INPUT
6646	035416	004037	036264		JSR	RO, @#CK.DIG	:CHECK THE DIGIT(S)
6647	035422	035260			3\$:ILLEGAL INPUT
6648	035424	035260			3\$:INPUT TOO LARGE
6649	035426	035434			9\$:TERMINATED WITH A "," OR "CR"
6650	035430	035776			OPN.N1		:TERMINATED WITH A ","
6651	035432	036014			OPN.X1		:TERMINATED WITH A "..."
6652	035434	010264	001570	9\$:	MOV	R2, RPT(R4)	:SAVE THIS VARIABLE
6653	035440	000705			BR	2\$:MOVE TO NEXT VARIABLE
6654	035442	104401	046141	OPNPAT:	TYPE	,MSG.SP	:TYPE SPACES
6655	035446	104401	045146		TYPE	,MSG.PAT	:TYPE "PAT"
6656	035452	104401	045152		TYPE	,MSG.EQ	:TYPE "="
6657	035456	016446	001570		MOV	RPT(R4), -(SP)	:SAVE RPT(R4) FOR TYPEOUT
6658	035462	104402			TYPOC		:GO TYPE--OCTAL ASCII(ALL DIGITS)
6659	035464	104401	046142		TYPE	,MSG.SP+1	:TYPE ONE SPACE
6660	035470	104411			RDLIN		:READ ASCIZ STRING
6661	035472	012601			MOV	(SP)+, R1	:PICKUP POINTER
6662	035474	004037	036210		JSR	RO, @#CK.CHR	:CHECK ONE CHARACTER
6663	035500	035442			OPNPAT		:ILLEGAL CHARACTER
6664	035502	035240			OPNPRM		:CARRIAGE RETURN
6665	035504	035536			3\$:"/
6666	035506	035240			OPNPRM		:""
6667	035510	035514			1\$:""
6668	035512	035534			2\$:DIGIT 0-9
6669	035514	105711		1\$:	TSTB	(R1)	:"CR" AFTER THE PERIOD?
6670	035516	001531			BEQ	OPN.N2	:YES--GO CLOSE THIS TEST
6671	035520	122721	000056		CMPB	#'.,(R1)+	:NO--PERIOD?
6672	035524	001346			BNE	OPNPAT	:NO--LOOP
6673	035526	105711			TSTB	(R1)	:"CR" AFTER A DOUBLE PERIOD?
6674	035530	001533			BEQ	OPN.X2	:YES--GO START TESTING
6675	035532	000743			BR	OPNPAT	:NO--LOOP
6676	035534	005301		2\$:	DEC	R1	:BACKUP THE ASCII POINTER
6677	035536			3\$:			
6678	035536	004037	036450		JSR	RO, @#CK.NUM	:CHECK THE NUMBER
6679	035542	035442			OPNPAT		:ILLEGAL INPUT
6680	035544	035552			4\$:TERMINATED WITH A "," OR "CR"

6681	035546	035776			OPN.N1		: TERMINATED WITH A "."
6682	035550	036014			OPN.X1		: TERMINATED WITH A "..."
6683	035552	010264	001570	4\$:	MOV R2,RPT(R4)		: SAVE THE INPUT NUMBER
6684	035556	006002			ROR R2		: OPEN PATTERN 0?
6685	035560	103227			BCC OPNPRM		: NO--START AT BEGINNING OF PARAMETER TABLE
6686	035562	104412			SAVREG		: SAVE R0 - R5
6687	035564	005000		OPNWDS:	CLR R0		: START WITH WORD 0
6688	035566	012704	003206		MOV #PATO,R4		
6689	035572			1\$:			
6690	035572	104401	035600		TYPE ,65\$: : TYPE ASCIZ STRING
6691	035576	000403			BR ,64\$: : GET OVER THE ASCIZ
6692				::65\$:	.ASCIZ / WD/		
6693	035606			64\$:			
6694	035606	010046			MOV R0,-(SP)		: PUT R0 ON THE STACK
6695	035610	004737	025242		JSR PC,@#SSB2D		: CHANGE R0 TO DECIMAL ASCIZ
6696	035614	004737	025472		JSR PC,@#SSUPRS		: TYPE WITHOUT LEADING ZEROS
6697	035620	104401	045152		TYPE MSG.EQ		: TYPE "="
6698	035624	011446			MOV (R4),-(SP)		: SAVE (R4) FOR TYPEOUT
6699	035626	104402			TYP0C		: GO TYPE--OCTAL ASCII(ALL DIGITS)
6700	035630	104411			RDLIN		: READ ASCIZ STRING
6701	035632	012601			MOV (SP)+,R1		: PICKUP THE POINTER
6702	035634	004037	036210		JSR R0,@#CK.CHR		: CHECK ONE CHARACTER
6703	035640	035572			1\$: ILLEGAL CHARACTER
6704	035642	035674			4\$: CARRIAGE RETURN
6705	035644	035656			2\$		"/"
6706	035646	035674			4\$		"/"
6707	035650	035710			5\$		"/"
6708	035652	035654			10\$: DIGIT 0-9
6709	035654	005301		10\$:	DEC R1		: BACKUP THE ASCII POINTER
6710	035656			2\$:			
6711	035656	004037	036450		JSR R0,@#CK.NUM		: CHECK THE NUMBER
6712	035662	035572			1\$: ILLEGAL INPUT
6713	035664	035672			3\$: TERMINATED WITH A "." OR "CR"
6714	035666	035730			6\$: TERMINATED WITH A "..."
6715	035670	035742			8\$: TERMINATED WITH A "..."
6716	035672	010214		3\$:	MOV R2,(R4)		: SAVE THE INPUT
6717	035674	005724		4\$:	TST (R4)+		: MOVE TO NEXT WORD
6718	035676	005200			INC R0		: INCREMENT THE COUNT
6719	035700	022700	000020		CMP #16.,R0		: COUNT TO LARGE?
6720	035704	003332			1\$: NO--BRANCH
6721	035706	000726			BR OPNWDS		: YES--BRANCH
6722	035710	105711		5\$:	TSTB (R1)		: "CR" AFTER THE PERIOD?
6723	035712	001407			BEQ 7\$: YES--GO CLOSE THIS TEST
6724	035714	122721	000056		CMPB #'.,(R1)+		: NO--PERIOD?
6725	035720	001324			BNE 1\$: NO--BRANCH ILLEGAL INPUT STRING
6726	035722	105711			TSTB (R1)		: "CR" AFTER THE "PERIOD-PERIOD"?
6727	035724	001407			BEQ 9\$: YES--GO START TESTING
6728	035726	000721			BR 1\$: NO--LOOP
6729	035730	010224		6\$:	MOV R2,(R4)+		: SAVE THE INPUT
6730	035732	004737	035754	7\$:	JSR PC,@#CLSWDS		: CLOSE THE DATA PATTERN
6731	035736	104413			RESREG		: RESTORE R0 - R5
6732	035740	000420			BR OPN.N2		: MOVE TO NEXT TEST
6733	035742	010224		8\$:	MOV R2,(R4)+		: SAVE THE INPUT
6734	035744	004737	035754	9\$:	JSR PC,@#CLSWDS		: CLOSE THE DATA PATTERN
6735	035750	104413			RESREG		: RESTORE R0 - R5
6736	035752	000422			BR OPN.X2		: START TESTING

6737	035754	012701	003206	CLSWDS:	MOV	#PATO,R1	: FIRST ADDRESS OF DATA PATTERN
6738	035760	005200		1\$:	INC	RO	: COUNT THE LAST WORD THAT WAS STORED
6739	035762	022700	000017		CMP	#15.,RO	: END OF TABLE
6740	035766	002402			BLT	2\$: YES--EXIT
6741	035770	012124			MOV	(R1)+,(R4)+	: COPY
6742	035772	000772			BR	1\$: LOOP
6743	035774	000207		2\$:	RTS	PC	: RETURN
6744	035776	010264	001570	OPN.N1:	MOV	R2,RPT(R4)	: SAVE THIS VARIABLE
6745	036002	005726		OPN.N2:	TST	(SP)+	: CLEAN OFF THE STACK
6746	036004	004737	036054		JSR	PC,CLOSE	: CLOSE THIS TEST
6747	036010	000137	035076		JMP	OPN.1	: GO OPEN THE NEXT TEST
6748	036014	010264	001570	OPN.X1:	MOV	R2,RPT(R4)	: SAVE THIS VARIABLE
6749	036020	005726		OPN.X2:	TST	(SP)+	: CLEAN OFF THE STACK
6750	036022	004737	036054	1\$:	JSR	PC,CLOSE	: CLOSE THIS TEST
6751	036026	005725		2\$:	TST	(R5)+	: UPDATE THE INDEX
6752	036030	020527	000034		CMP	R5,#16*2	: INDEX TO BIG?
6753	036034	002403			BLT	3\$: NO--BRANCH
6754	036036	104413			RESREG		: RESTORE RO - R5
6755	036040	000137	034230		JMP	GT.PR2	: GO TO EXIT
6756	036044	036503	001506	3\$:	BIT	BITS(R5),R3	: IS THIS TEST OPEN FOR CHANGE?
6757	036050	001364			1\$: YES--GO CLOSE IT
6758	036052	000765			BR	2\$: NO--MOVE TO NEXT TEST
6759	036054	104412		CLOSE:	SAVREG		: SAVE RO - R5
6760	036056	012700	001566		MOV	#PRM,RO	: "FROM" ADDRESS
6761	036062	016501	001620		MOV	PRMP+(R5),R1	: "TO" ADDRESS
6762	036066	012002			MOV	(RO)+,R2	: "FROM" INDICATOR
6763	036070	012103			MOV	(R1)+,R3	: "TO" INDICATOR
6764	036072	012704	000001		MOV	#1,R4	: TEST BIT START A "RPT"
6765	036076	030402		1\$:	BIT	R4,R2	: PARAMETER TO BE MOVED?
6766	036100	001403			BEQ	2\$: NO--BRANCH
6767	036102	030403			BIT	R4,R3	: A PLACE TO PUT IT?
6768	036104	001404			BEQ	3\$: NO--BRANCH
6769	036106	011011			MOV	(RO),(R1)	: YES--MOVE "FROM" TO "TO"
6770	036110	030403		2\$:	BIT	R4,R3	: "TO" PARAMETER?
6771	036112	001401			BEQ	3\$: NO--BRANCH
6772	036114	005721			TST	(R1)+	: YES--UPDATE THE POINTER
6773	036116	005720		3\$:	TST	(RO)+	: UPDATE FROM POINTER
6774	036120	006304			ASL	R4	: POSITION THE TEST BIT
6775	036122	032704	002000		BIT	#BIT10,R4	: DONE?
6776	036126	001763			BEQ	1\$: NO--BRANCH
6777	036130	104413			RESREG		: RESTORE RO - R5
6778	036132	000207			RTS	PC	: RETURN
6779							: THIS ROUTINE IS USED TO CHECK IF AN
6780							: ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
6781					CALL		
6782					MOV	#ADR,R1	: ADDRESS OF ASCII CHARACTER
6783					JSR	RO,CHK.OCT	: CHECK THE CHARACTER
6784					RETURN1		: CHARACTER IS NOT BETWEEN 0-7
6785					RETURN2		: CHARACTER IS IN R2 AS A
6786							: OCTAL DIGIT
6787							
6788	036134	121127	000060	CK.OCT:	CMPB	(R1),#'0	: LESS THAN ZERO?
6789	036140	103407			1\$: YES -- BRANCH
6790	036142	121127	000067		CMPB	(R1),#'7	: GREATER THAN SEVEN?
6791	036146	101004			BHI	1\$: YES -- BRANCH
6792	036150	111102			MOVB	(R1),R2	: GET THE CHARACTER

6793 036152 042702 177770
 6794 036156 005720
 6795 036160 000200
 6796
 6797
 6798
 6799
 6800
 6801
 6802
 6803
 6804
 6805
 6806 036162 121127 000060
 6807 036166 103407
 6808 036170 121127 000071
 6809 036174 101004
 6810 036176 111102
 6811 036200 042702 000060
 6812 036204 005720
 6813 036206 000200
 6814
 6815
 6816
 6817
 6818
 6819
 6820
 6821
 6822
 6823
 6824
 6825
 6826
 6827
 6828 036210 105711
 6829 036212 001420
 6830 036214 121127 000057
 6831 036220 001414
 6832 036222 121127 000054
 6833 036226 001410
 6834 036230 121127 000056
 6835 036234 001404
 6836 036236 004037 036162
 6837 036242 000406
 6838 036244 005720
 6839 036246 005720
 6840 036250 005720
 6841 036252 005720
 6842 036254 005720
 6843 036256 005201
 6844 036260 011000
 6845 036262 000200
 6846
 6847
 6848

```

        BIC      #1C7,R2      ;STRIP AWAY THE ASCII
        TST      (R0)+        ;ADJUST FOR RETURN
1$:     RTS      RO          ;RETURN

;THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
;AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
;CALL
        MOV      #ADR,R1     ;ADDRESS OF ASCII CHARACTER
        JSR      RO,@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      RO          ;RETURN

;THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
;DETERMINE WHAT IT IS.
;CALL
        MOV      #ADR,R1     ;ADDRESS OF ASCII CHARACTER
        JSR      RO,@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     RO,@CK.DEC   ;"DIGIT"?
        BR      5$          ;NO -- BRANCH
1$:     TST     (R0)+        ;DIGIT BETWEEN 0-9
2$:     TST     (R0)+        ;PERIOD
3$:     TST     (R0)+        ;COMMA
4$:     TST     (R0)+        ;SLASH
5$:     TST     (R0)+        ;CARRIAGE RETURN
        INC     R1           ;MOVE POINTER TO NEXT CHARACTER
        MOV     (R0),RO      ;UNKNOWN CHARACTER
        RTS      RO          ;RETURN

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

```

6849      :CALL
6850      :      MOV      #ADR,R1      ;ADDRESS OF ASCIZ STRING
6851      :      MOV      #NUM,R2      ;MAX. MAGNITUDE OF INPUT NUMBER
6852      :      JSR      RO,@#CK.DIG  ;CHECK DIGITS
6853      :      RETURN   ADR1         ;ILLEGAL CHARACTER -- R2=?
6854      :      RETURN   ADR2         ;INPUT NUMBER TO LARGE -- R2=?
6855      :      RETURN   ADR3         ;"COMMA" -- R2 = NUMBER
6856      :      RETURN   ADR4         ;"PERIOD" -- R2 = NUMBER
6857      :      RETURN   ADR5         ;"PERIOD-PERIOD" -- R2 = NUMBER
6858
6859      036264 010446      CK.DIG: MOV      R4,-(SP)      ;SAVE R4
6860      036266 010346      MOV      R3,-(SP)      ;SAVE R3
6861      036270 010246      MOV      R2,-(SP)      ;SAVE THE MAX. SIZE ON THE STACK
6862      036272 005002      CLR      R2            ;START WITH 0
6863      036274 005003      CLR      R3
6864      036276 005004      CLR      R4
6865      036300 004037 036210 JSR      RO,@#CK.CHR  ;CHECK ONE CHARACTER
6866      036304 036434      B$      ;ILLEGAL CHARACTER
6867      036306 036434      B$      ;CARRIAGE RETURN
6868      036310 036434      B$      ;"/"
6869      036312 036434      B$      ;"."
6870      036314 036434      B$      ;" "
6871      036316 036320      1$:     ASL      R3            ;DIGIT 0-9
6872      036320 006303      MOV      R3,-(SP)      ;SAVE #2
6873      036322 010346      ASL      R3            ;4
6874      036324 006303      ASL      R3            ;8
6875      036326 006303      ADD      (SP)+,R3       ;(*8)+(*2)=*10.
6876      036330 062603      ADD      R2,R3         ;UPDATE THE INPUT NUMBER
6877      036332 060203      JSR      RO,@#CK.CHR  ;CHECK ONE CHARACTER
6878      036334 004037 036210 JSR      RO,@#CK.CHR  ;ILLEGAL CHARACTER
6879      036340 036434      B$      ;CARRIAGE RETURN
6880      036342 036354      9$:     B$      ;"/"
6881      036344 036434      B$      ;"."
6882      036346 036362      3$:     B$      ;" "
6883      036350 036360      1$:     ASL      R3            ;DIGIT 0-9
6884      036352 036320      DEC      R1            ;BACKUP THE CHARACTER POINTER
6885      036354 005301      BR      3$            ;CONTINUE
6886      036356 000401      2$:     TST      (R4)+     ;"PERIOD"
6887      036360 005724      3$:     TST      (R4)+     ;"COMMA" OR "CR"
6888      036362 005724      JSR      RO,@#CK.CHR  ;CHECK ONE CHARACTER
6889      036364 004037 036210 JSR      RO,@#CK.CHR  ;ILLEGAL CHARACTER
6890      036370 036434      B$      ;CARRIAGE RETURN
6891      036372 036424      6$:     B$      ;"/"
6892      036374 036434      8$:     B$      ;"."
6893      036376 036434      4$:     B$      ;" "
6894      036400 036404      5$:     TST      (R4)+     ;DIGIT 0-9
6895      036402 036414      TSTB   (R1)          ;"PERIOD-PERIOD"
6896      036404 005724      BEQ     8$            ;"CR"?
6897      036406 105711      BR      8$            ;YES--BRANCH
6898      036410 001405      BR      8$
6899      036412 000410      BR      8$
6900      036414 126127 177776 000054 5$:     CMPB   -2(R1),#',     ;WAS CHARACTER BEFORE THE DIGIT A COMMA?
6901      036422 001004      BNE     8$            ;NO--EXIT
6902      036424 020316      6$:     CMP      R3,(SP)    ;INPUT TO LARGE?
6903      036426 101001      BHI     7$            ;YES -- BRANCH
6904      036430 060400      ADD     R4,RO        ;ADJUST RETURN ADDRESS

```



```

6905 036432 005720      7$:   TST      (R0)+
6906 036434 010302      8$:   MOV      R3,R2      ;NUMBER TO R2
6907 036436 005726      TST      (SP)+      ;CLEAN MAX. SIZE OFF OF STACK
6908 036440 012603      MOV      (SP)+,R3    ;RESTORE R3
6909 036442 012604      MOV      (SP)+,R4    ;RESTORE R4
6910 036444 011000      MOV      (R0),R0     ;GET RETURN ADDRESS
6911 036446 000200      RTS       R0         ;RETURN
6912
6913 ; THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
6914 ; AND FORMS AN OCTAL NUMBER IN R2
6915 ; CALL:
6916 ;       MOV      #ADR,R1      ;ADDRESS OF ASCIZ STRING
6917 ;       JSR      R0,#CK.NUM    ;GO FORM THE NUMBER
6918 ;       RETURN   ADR1         ;ILLEGAL CHARACTER IN THE INPUT STRING
6919 ;       RETURN   ADR2         ;"COMMA" OR "CR"--R2=NUMBER
6920 ;       RETURN   ADR3         ;"PERIOD"--R2=NUMBER
6921 ;       RETURN   ADR4         ;"PERIOD-PERIOD"--R2=NUMBER
6922
6923 036450 010346      CK.NUM:  MOV      R3,-(SP)   ;SAVE R3
6924 036452 005003      CLR      R3          ;START NUMBER AT ZERO
6925 036454 004037 036134  JSR      R0,#CK.OCT    ;OCTAL DIGIT?
6926 036460 000440      BR       6$         ;NO--BRANCH
6927 036462 005201      1$:   INC      R1          ;MOVE TO NEXT CHARACTER
6928 036464 006303      ASL     R3          ;FOR THE OCTAL NUMBER IN R3
6929 036466 103435      BCS     6$         ;DON'T LET IT GET TO BIG
6930 036470 006303      ASL     R3
6931 036472 103433      BCS     6$
6932 036474 006303      ASL     R3
6933 036476 103431      BCS     6$
6934 036500 060203      ADD     R2,R3
6935 036502 004037 036134  JSR      R0,#CK.OCT    ;IS THIS AN OCTAL DIGIT?
6936 036506 000401      BR       2$         ;NO--FIND OUT WHAT IT IS
6937 036510 000764      BR       1$         ;YES--MAKE IT PART OF THE NUMBER
6938 036512 010302      2$:   MOV      R3,R2     ;SAVE THE OCTAL NUMBER
6939 036514 005003      CLR     R3          ;START WITH ZERO INDEX
6940 036516 004037 036210  JSR      R0,#CK.CHR    ;CHECK ONE CHARACTER
6941 036522 036562      6$    ;ILLEGAL CHARACTER
6942 036524 036552      5$    ;CARRIAGE RETURN
6943 036526 036562      6$    ;"/"
6944 036530 036552      5$    ;"."
6945 036532 036536      3$    ;"?"
6946 036534 036562      6$    ;DIGIT 0-9
6947 036536 005723      3$:   TST      (R3)+     ;"PERIOD"
6948 036540 12:127 000056  CMPB    (R1),#'.     ;"PERIOD-PERIOD"?
6949 036544 00:002      BNE     5$         ;NO--BRANCH
6950 036546 005201      INC     R1          ;YES--ADVANCE THE POINTER
6951 036550 005723      4$:   TST      (R3)+     ;"PERIOD-PERIOD"
6952 036552 005723      5$:   TST      (R3)+     ;"COMMA"
6953 036554 105711      TSTB   (R1)        ;"CR"?
6954 036556 001001      BNE     6$         ;NO--BRANCH
6955 036560 060300      ADD     R3,R0       ;YES--SAVE THE OCTAL NUMBER
6956 036562 012603      MOV     (SP)+,R3    ;RESTORE R3
6957 036564 011000      MOV     (R0),R0     ;PICKUP EXIT ADDRESS
6958 036566 000200      RTS       R0         ;RETURN

```

6959
6960
6961
6962
6963
6964
6965
6966
6967
6968
6969
6970
6971
6972
6973
6974
6975
6976
6977
6978
6979
6980
6981
6982
6983
6984
6985
6986
6987
6988
6989
6990
6991
6992
6993
6994
6995
6996
6997
6998
6999
7000
7001
7002
7003
7004
7005
7006
7007
7008
7009
7010
7011
7012
7013
7014

;;*****

.SBTTL SINGLE/DUAL PORT RH70/RMO3 DRIVER (REV 5.1)-24-AUG-77
;NEW DRIVE TYPE ID FOR RMO2 *****
; 10-AUG-77 *****

;COPYRIGHT (C) 1977
;DIGITAL EQUIPMENT CORP.
;MAYNARD, MA 01754
;AUTHOR(S): JIM LACEY/CHUCK HESS/C. CHEN

;;*****

;STORAGE FOR RMDs, RMER1, RMER2, AND RMMR2 ON AN ERROR "2"
;RMERRS = RMDs
;RMERRS+2 = RMER1
;RMERRS+4 = RMER2
;RMERRS+6 = RMMR2

036570 000000 000000 000000 RMERRS: .WORD 0,0,0,0
036576 000000

;TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
;DRVACT=0 IF DRIVE IS IDLE
;DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
;DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION

036600 000 ;DRIVE 0
036601 000 ;DRIVE 1
036602 000 ;DRIVE 2
036603 000 ;DRIVE 3
036604 000 ;DRIVE 4
036605 000 ;DRIVE 5
036606 000 ;DRIVE 6
036607 000 ;DRIVE 7

;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXISTENT
;DRVSTA>0 IF DRIVE IS ONLINE
;DRVSTA<0 IF DRIVE IS UNSAFE

036610 000 ;DRIVE 0
036611 000 ;DRIVE 1
036612 000 ;DRIVE 2
036613 000 ;DRIVE 3
036614 000 ;DRIVE 4
036615 000 ;DRIVE 5
036616 000 ;DRIVE 6
036617 000 ;DRIVE 7

;TABLE OF DRIVE TYPES (DRV TYP=8 BYTES)
;DRV TYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
;DRV TYP=5 IF DRIVE IS RMO2 *****
;DRV TYP=4 IF DRIVE IS RMO3
;DRV TYP=-1 IF NOT RMO3


```

7015
7016 036620 000
7017 036621 000
7018 036622 000
7019 036623 000
7020 036624 000
7021 036625 000
7022 036626 000
7023 036627 000
7024
7025
7026
7027
7028
7029 036630 000
7030 036631 000
7031 036632 000
7032 036633 000
7033 036634 000
7034 036635 000
7035 036636 000
7036 036637 000
7037
7038
7039
7040
7041
7042 036640 000
7043 036641 000
7044 036642 000
7045 036643 000
7046 036644 000
7047 036645 000
7048 036646 000
7049 036647 000
7050
7051
7052
7053
7054
7055 036650 000000
7056
7057
7058
7059
7060
7061
7062
7063 036652 000000
7064
7065
7066
7067
7068
7069 036654 000
7070

```

```

DRVTYP: .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 DUAL PORT INITIALIZATION INDICATORS
;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
;DPINT<0 IF INITIALIZATION IS IN PROGRESS

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

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.

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.

SRCHWT: .WORD 0

;RMO3 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
;ACTDRV=0 IF DRIVER IS INACTIVE
;ACTDRV>0 IF DRIVER IS ACTIVE

ACTDRV: .BYTE 0

```

```

7071
7072
7073
7074
7075 036655 000
7076
7077
7078
7079
7080
7081
7082 036656 000
7083 036657 000
7084 036658 000
7085 036661 000
7086 036662 000
7087 036663 000
7088 036664 000
7089 036665 000
7090
7091
7092
7093
7094 036666 000
7095 036667 000
7096 036670 000
7097 036671 000
7098 036672 000
7099 036673 000
7100 036674 000
7101 036675 000
7102
7103
7104
7105
7106
7107
7108
7109 036676 000000
7110
7111
7112
7113
7114
7115
7116
7117 036700 177777
7118
7119
7120
7121
7122 036702 177777
7123 036704 177777
7124 036706 177777
7125 036710 177777
7126 036712 177777

```

```

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

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/RM03 REGISTERS WHEN THE
;OPERATION IS COMPLETED AS PER (DPB+14).
;SAVEFG=0 IF SAVE THE RH70/RM03 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

```



```

7127 036714 177777 .WORD -1 ;DRIVE 5
7128 036716 177777 .WORD -1 ;DRIVE 6
7129 036720 177777 .WORD -1 ;DRIVE 7
7130
7131 ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
7132 ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
7133 ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
7134
7135 036722 177777 DTUW: .WORD -1
7136
7137 ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
7138 ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
7139 ;ATTENTION BIT
7140
7141 036724 001 ATABIT: .BYTE 1 ;DRIVE 0
7142 036725 002 .BYTE 2 ;DRIVE 1
7143 036726 004 .BYTE 4 ;DRIVE 2
7144 036727 010 .BYTE 10 ;DRIVE 3
7145 036730 020 .BYTE 20 ;DRIVE 4
7146 036731 040 .BYTE 40 ;DRIVE 5
7147 036732 100 .BYTE 100 ;DRIVE 6
7148 036733 200 .BYTE 200 ;DRIVE 7
7149
7150 ;FSRM03 TO RH70 "MASSBUS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE
7151 ;CALLING IT FATAL (MCPEMX=1 WORD)
7152
7153 036734 000003 MCPEMX: .WORD 3
7154
7155 ;STORAGE FOR RMADR (THE FIRST ADDRESS (776700) OF THE RH70/RM03),
7156 ;RMVEC (THE VECTOR ADDRESS (254)), AND RMVEC+2 (THE BR LEVEL (5)).
7157
7158 036736 176700 000240 RMADR: .WORD 176700
7159 036740 000254 000240 RMVEC: .WORD 254,5*32.
7160
7161 ;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)
7162
7163 036744 000004 MXLACT: .WORD 4
7164 ;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)
7165
7166 036746 001000 MXDLTA: .WORD 8.*64.
7167 ;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)
7168
7169 036750 000200 MNDLTA: .WORD 2*64.
7170 ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)
7171
7172 036752 000005 MXWNDW: .WORD 5
7173
7174 ;DEFINITIONS OF THE RH70/RM03 ADDRESS INDEXES
7175
7176 000000 RMCS1=0 ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
7177 000002 RMWC=2 ;WORD COUNT REGISTER (NOT A DRIVE REG)
7178 000004 RMBA=4 ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
7179 000006 RMDA=6 ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
7180 000010 RMCS2=10 ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
7181 000012 RMDS=12 ;DRIVE STATUS REGISTER (DRIVE REG 01)
7182 000014 RMER1=14 ;ERROR REGISTER #1 (DRIVE REG. 02)
  
```

7183	000016	RMA5=16	:ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
7184	000020	RMLA=20	:LOOK AHEAD REGISTER (DRIVE REG. 07)
7185	000022	RMOB=22	:DATA BUFFER REGISTER (NOT A DRIVE REG.)
7186	000024	RMMR1=24	:MAINTAINABILITY REGISTER (DRIVE REG. 03)
7187	000026	RMDT=26	:DRIVE TYPE REGISTER (DRIVE REG. 06)
7189	000030	RMSN=30	:SERIAL NUMBER REGISTER (DRIVE REG. 10)
7189	000032	RMOF=32	:OFFSET REGISTER (DRIVE REG. 11)
7190	000034	RMDC=34	:DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
7191	000036	RMHR=36	:DUMMY ADDRESS REGISTER (DRIVE REG. 13)
7192	000040	RMMR2=40	:MAINTENANCE REGISTER #2
7193	000042	RMER2=42	:ERROR REGISTER #2 (DRIVE REG. 15)
7194	000044	RMEC1=44	:ECC POSITION REGISTER (DRIVE REG. 16)
7195	000046	RMEC2=46	:ECC PATTERN REGISTER (DRIVE REG. 17)

;RH70/RM03 DRIVER INITIALIZATION CODE

;THIS ROUTINE WILL DETERMINE WHICH RM03 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

7209				RMINIT: SAVREG	:SAVE R0 - R5
7210	036754	104412		MOV @#PS,-(SP)	:SAVE THE PRESENT PROCESSOR STATUS
7211	036756	013746	177776	MOV #(<5*32.>,@#PS	:CHANGE THE PRIORITY TO 5
7212	036762	012737	000240	PC,CLIQUE	:CLEAR ALL REQUEST QUEUES
7213	036770	004737	044616	JSR #RMERRS,R1	:FIRST ADDRESS TO BE CLEARED
7214	036774	012701	036570	MOV #SEEKFG,R2	:LAST ADDRESS TO BE CLEARED
7215	037000	012702	036700	1\$: CLR (R1)+	:CLEAR
7216	037004	005021		CMP R1,R2	:ARE WE DONE?
7217	037006	020102		BLO 1\$:BRANCH IF NO
7218	037010	103775		MOV #DTUW,R2	:LAST ADDRESS
7219	037012	012702	036722	2\$: MOV #-1,(R1)+	:INITIALIZE
7220	037016	012721	177777	CMP R1,R2	:DONE?
7221	037022	020102		BLOS 2\$:LOOP IF NO
7222	037024	101774		CLR DRVSTA	:SET ALL DRIVES TO OFFLINE
7223	037026	005037	036610	CLR DRVSTA+2	
7224	037032	005037	036612	CLR DRVSTA+4	
7225	037036	005037	036614	CLR DRVSTA+6	
7226	037042	005037	036616	MOV RMVEC,R3	:SETUP THE RH70/RM03 VECTOR
7227	037046	013703	036740	MOV #ISR,(R3)+	
7228	037052	012723	041566	MOV RMVEC+2,(R3)	
7229	037056	013713	036742	MOV RMADR,R4	:FIRST ADDRESS OF RH70/RM03
7230	037062	013704	036736	MOV #BIT05,RMCS2(R4)	:MASSBUS INIT
7231	037066	012764	000040	000010	:START WITH DRIVE 0
7232	037074	005001		CLR R1	:INIT THE DRIVE
7233	037076	004037	037166	3\$: JSR R0,DRVINT	: 'DVA' NOT SET OR PARITY ERROR
7234	037102	000401		BR 4\$:NORMAL RETURN
7235	037104	000402		BR 5\$:SET DRIVE STATUS TO OFFLINE
7236	037106	105061	036610	4\$: CLRB DRVSTA(R1)	:GO TO NEXT DRIVE
7237	037112	005201		5\$: INC R1	:MASK OUT UNUSED BITS
7238	037114	042701	177770	BIC #1C7,R1	

K13

```

7239 037120 001366      BNE      3$          ;BR IF MORE DRIVES TO GO
7240 037122 012701 000007  MOV      #7,R1      ;START WITH DRIVE 7
7241 037126 005037 177776  CLR      @#PS       ;CLEAR THE PROCESSOR STATUS
7242 037132 105761 036630 6$: TSTB   DPINT(R1)  ;WAITING FOR DRIVE TO SWITCH PORTS ?
7243 037136 001405      BEQ      8$          ;BR NOT WAITING
7244 037140 004737 044252  JSR      PC,SET.IE  ;SET INTERRUPT
7245 037144 105761 036630 7$: TSTB   DPINT(R1)  ;DRIVE SWITCHED PORTS ?
7246 037150 001375      BNE      7$          ;BR IF NOT
7247 037152 005301 8$: DEC    R1         ;GO TO THE NEXT DRIVE
7248 037154 100366      BPL      6$          ;CHECK NEXT DRIVE
7249 037156 012637 177776  MOV      (SP)+,@#PS ;RESTORE THE PROCESSOR STATUS
7250 037162 104413      RESREG   PC         ;RESTORE R0 - R5
7251 037164 000207      RTS      PC         ;BYE-BYE
7252
7253 ;DRIVE INITIALIZATION ROUTINE
7254 ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
7255 ;AN RMO3. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT22
7256 ;IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
7257 ;INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
7258 ;DRVSTA IS SET TO THE PROPER CONDITION.
7259
7260 ;CALL
7261 :
7262 :
7263 :
7264 :
7265 :
7266 :
7267 037166 010546      MOV      #DRVNUM,R1 ;DRIVE NUMBER TO R1
7268 037170 105061 036610  MOV      RMADR,R4   ;UNIBUS ADDRESS OF RH70/RMO3 (RMCS1)
7269 037174 105061 036620  JSR      R0,DRVINT  ;CALLED BY A JSR
7270 037200 105061 036656  RETURN1  ;ERROR OCCURRED (PARITY)
7271 037204 010164 000010  RETURN2  ;NORMAL RETURN
7272 037210 112764 000111 000000  DRVINT: MOV      R5,-(SP) ;SAVE R5
7273 037216 032764 010000 000010  DULP:  CLRB   DRVSTA(R1) ;START DRIVE STATUS AS OFFLINE
7274 037224 001403      CLRB   DRVSTYP(R1) ;CLEAR THE DRIVE TYPE INDICATOR
7275 037226 004737 044252  CLRB   ULDFLG(R1)  ;CLEAR THE UNLOAD FLAG
7276 037232 000507      MOV      R1,RMCS2(R4) ;SELECT A DRIVE
7277 037234 105061 036610  MOVB   #111,RMCS1(R4) ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
7278 037240 032764 004000 000000  BIT    #BIT12,RMCS2(R4) ;NONEXISTENT DRIVE?
7279 037246 001750      BEQ    1$          ;NO---BRANCH
7280 037250 004037 043562  JSR      PC,SET.IE  ;GO SET "IE" WITHOUT A "TRE"
7281 037254 000026      BR     6$          ;LEAVE THIS ROUTINE
7282 037256 037476 1$: CLRB   DRVSTA(R1) ;SET DRIVE STATUS TO OFFLINE
7283 037260 012605      BIT    #BIT11,RMCS1(R4) ;SEE IF DRIVE AVAILABLE
7284 037262 112761 000004 036620  BEQ    DULP        ;BR IF DRIVE NOT AVAILABLE
7285 037270 022705 020024      JSR      R0,RD.RM  ;READ THE DRIVE TYPE REG.
7286 037274 001420      RMDT   8$          ;ERROR RETURN ADDRESS
7287 037276 022705 024024  MOV      (SP)+,R5   ;PUT DRIVE TYPE IN R5
7288 037302 001415      MOVB   #4,DRVSTYP(R1) ;SET RMO3 INDICATOR
7289 037304 112761 000005 036620  CMP    #20024,R5   ;SINGLE PORT RMO3 ?
7290 037312 022705 020025      BEQ    2$          ;BR IF YES
7291 037316 001407      CMP    #24024,R5   ;DUAL PORT RMO3 ?
7292 037320 022705 024025  BEQ    2$          ;BR IF YES
7293 037324 001404      MOVB   #5,DRVSTYP(R1) ;SET RMO2 INDICATOR
7294 037326 112761 177777 036620  CMP    #20025,R5   ;SINGLE PPRT RMO2 ?
                                BEQ    2$          ;BRANCH IF SO
                                CMP    #24025,R5   ;DUAL PORT RMO2 ?
                                BEQ    2$          ;BRANCH IF SO
                                MOVB   #-1,DRVSTYP(R1) ;SET INDICATOR TO 'OTHER'

```



```

7295 037334 000446          BR      6$          ;EXIT
7296 037336 012746 000121    2$:  MOV      #121, -(SP) ;DO A "READ-IN PRESET"
7297 037342 004037 043742    JSR      RO, WRT.RM
7298 037346 000000          RMCS1
7299 037350 037476          8$
7300 037352 012746 010000    MOV      #BIT12, -(SP) ;SET FMT22=1
7301 037356 004037 043742    JSR      RO, WRT.RM
7302 037362 000032          RMOF
7303 037364 037476          8$
7304 037366 004037 043562    JSR      RO, RD.RM ;READ RMD5
7305 037372 000012          RMD5
7306 037374 037476          8$
7307 037376 012605          MOV      (SP)+, R5 ;AND SAVE IT IN R5
7308 037400 100015          BPL      4$          ;BRANCH IF ATA=0
7309 037402 116164 036724 000016    MOVB    ATABIT(R1), RMA5(R4) ;CLEAR ATTENTION BIT
7310 037410 004037 043562    JSR      RO, RD.RM ;FIND OUT WHY ATA=1
7311 037414 000014          RMER1
7312 037416 037476          8$
7313 037420 006126          ROL      (SP)+ ;IS IT UNSAFE?
7314 037422 100004          BPL      4$          ;BR IF NOT
7315 037424 112761 177777 036610    MOVB    #-1, DRVSTA(R1) ;SET UNSAFE INDICATOR
7316 037432 000407          BR      6$          ;EXIT
7317 037434 005105          4$:  COM      R5 ;CHECK MOL, DPR, DRY, AND VV
7318 037436 042705 167077    BIC      #1<BIT12!BIT08!BIT07!BIT06>, R5
7319 037442 001003          BNE     6$          ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
7320 037444 112761 000001 036610    MOVB    #1, DRVSTA(R1) ;SET DRIVE STATUS TO ONLINE
7321 037452 005720          TST     (R0)+ ;STEP OVER THE ERROR RETURN
7322 037454 000410          BR      8$          ;EXIT
7323 037456 006301          7$:  ASL      R1 ;CHANGE INDEX TO ADDRESS WORDS
7324 037460 012761 060000 036702    MOV      #60000, TIMER(R1) ;START 2 SEC TIMER
7325 037466 006201          ASR      R1 ;RESTORE R1
7326 037470 112761 177777 036630    MOVB    #-1, DPINT(R1) ;SET PORT INITIALIZE INIDICATOR
7327 037476 012605          8$:  MOV      (SP)+, R5 ;RESTORE R5
7328 037500 000200          RTS      RO ;EXIT
7329
7330 ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
7331 ;CALL
7332 ;
7333 ;
7334 ; JSR      RO, @#RMO3 ;CALL THE RMO3 DRIVER
7335 ; PNTADR ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
7336 ; RETURN1 ;RETURN HERE IF QUEUE IS FULL
7337 ; RETURN2 ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
7338 ; ;IS AN ERROR CONDITION
7339
7340 037502 013746 177776          RMO3: MOV      @#PS, -(SP) ;SAVE THE CALLING STATUS
7341 037506 013737 036742 177776    MOV      RMVEC+2, @#PS ;DON'T ALLOW ANY RMO3 INTERRUPTS
7342 037514 112737 000001 036654    MOVB    #1, ACTDRV ;SET "ACTIVE DRIVER" FLAG
7343 037522 104412          SAVREG ;SAVE R0 - R5
7344 037524 011002          MOV      (R0), R2 ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
7345 037526 005062 000016    CLR      16(R2) ;CLEAR THE STATUS/ERROR INDICATOR
7346 037532 111201          MOVB    (R2), R1 ;PICKUP THE DRIVE NUMBER
7347 037534 013704 036736    MOV      RMADR, R4 ;UNIBUS ADDRESS OF RMCS1
7348 037540 105761 036610    TSTB    DRVSTA(R1) ;CHECK DRIVES STATUS
7349 037544 003014          BGT      1$          ;BRANCH IF ONLINE
7350 037546 105761 036656    TSTB    ULDFLG(R1) ;UNLOAD COMMAND IN QUEUE?
    
```


M13

```

7351 037552 001036          BNE      3$          ;BRANCH IF YES
7352 037554 105761 036630  TSTB    DPINT(R1)  ;TRYING TO INIT THE DRIVE
7353 037560 001042          BNE      5$          ;BR IF YES
7354 037562 004037 037166  JSR     RO,DRVINT  ;GO INIT. THE DRIVE
7355 037566 000434          BR       4$          ;ERROR RETURN
7356 037570 105761 036610  TSTB    DRVSTA(R1) ;IS DRIVE STATUS ONLINE?
7357 037574 003445          BLE      6$          ;BR IF NOT
7358 037576 105761 036640  1$:    TSTB    DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
7359 037602 001031          BNE      5$          ;BR IF YES
7360 037604 010164 000010  MOV     R1,RMCS2(R4) ;SELECT THE DRIVE
7361 037610 004037 044714  JSR     RO,DRVQUE  ;PUT THIS REQUEST IN QUEUE
7362 037614 000460          BR       9$          ;QUEUE IS FULL
7363 037616 122762 000103 000002  CMPB    #103,2(R2) ;IS THIS REQ. FOR AN UNLOAD?
7364 037624 001003          BNE      2$          ;BR IF NO
7365 037626 112761 177777 036656  MOVB   #-1,ULDFLG(R1) ;SET THE "UNLOAD IN QUEUE" FLAG
7366 037634 105761 036600  2$:    TSTB    DRVACT(R1) ;IS THIS DRIVE ACTIVE?
7367 037640 001043          BNE      8$          ;BR IF YES
7368 037642 004737 037774  JSR     PC,OPT     ;CALL THE OPTIMIZER
7369 037646 000440          BR       8$
7370 037650 012762 120000 000016  3$:    MOV     #BIT15:BIT13,16(R2) ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
7371 037656 000434          BR       8$          ;EXIT
7372 037660 004737 041054          JSR     PC,C17     ;GO HANDLE THE PARITY ERROR
7373 037664 000431          BR       8$
7374 037666 004037 044714          JSR     RO,DRVQUE  ;PUT REQUEST IN QUEUE
7375 037672 000431          BR       9$          ;QUEUE IS FULL
7376 037674 032714 000100          BIT     #BIT06,(R4) ;IE BIT SET ?
7377 037700 001023          BNE      8$          ;YES
7378 037702 004737 044252          JSR     PC,SET.IE ;SET THE INTERRUPT
7379 037706 000420          BR       8$          ;RETURN
7380 037710 105761 036610  6$:    TSTB    DRVSTA(R1) ;SEE IF DRIVE OFFLINE OR UNSAFE
7381 037714 002412          BLT     7$          ;BR IF UNSAFE
7382 037716 012762 140000 000016  MOV     #BIT15:BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
7383 037724 105761 036620  TSTB    DRVSTYP(R1) ;SEE IF OFFLINE OR NONEXISTENT
7384 037730 001007          BNE      8$          ;BR IF OFFLINE
7385 037732 012762 100002 000016  MOV     #BIT15:BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
7386 037740 000403          BR       8$          ;GO TO EXIT
7387 037742 012762 110000 000016  7$:    MOV     #BIT15:BIT12,16(R2) ;DRIVE IS UNSAFE
7388 037750 104413  8$:    RESREG  ;RESTORE R0 - R5
7389 037752 005720          TST     (R0)+      ;SETUP FOR NORMAL RETURN
7390 037754 000401          BR       10$       ;FINISH UP, THEN EXIT
7391 037756 104413  9$:    RESREG  ;RESTORE R0 - R5
7392 037760 005720 10$:   TST     (R0)+      ;CORRECT THE RETURN ADDRESS
7393 037762 105037 036654  CLRB   ACTDRV     ;CLEAR "ACTIVE DRIVER" FLAG
7394 037766 012637 177776  MOV     (SP)+,2#PS ;RETURN "PS" TO USER LEVEL
7395 037772 000200          RTS     RO        ;RETURN TO CALLER
7396
7397          ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
7398
7399          ;CALL
7400          MOV     #DRVNUM,R1 ;DRIVE NUMBER TO R1
7401          JSR     PC,OPT     ;SETUP A COMMAND
7402
7403 037774 104412  OPT:   SAVREG  ;SAVE R0 - R5
7404 037776 013746 177776  MOV     2#PS,-(SP) ;SAVE PROC. STATUS
7405 040002 146137 036724 036652  BICB   ATABIT(R1),SRCHWT ;CLEAR LA SEACH FLAG
7406 040010 105061 036640  CLRB   DPRQS(R1)  ;RESET THE PORT REQ FLAG ****

```


7407	040014	004737	044770		JSR	PC,GETREQ	:GET "DPB" POINTER OF REQUEST	
7408	040020	005702			TST	R2	:IS THERE A REQUEST IN QUEUE?	
7409	040022	001472			BEQ	7\$:NO--BRANCH TO EXIT	
7410	040024	010164	000010		MOV	R1, RMCS2(R4)	:LOAD THE DRIVE ADDRESS *****	
7411	040030	012764	000111	000000	MOV	#111, RMCS1(R4)	:CLEAR THE DRIVE	
7412	040036	032764	004000	000000	BIT	#BIT11, RMCS1(R4)	:DVA SET ?	
7413	040044	001446			BEQ	5\$:TO PROT REQUEST, IF NOT	
7414	040046	105761	036610	10\$:	TSTB	DRVSTA(R1)	:IS DRIVE ONLINE?	
7415	040052	003014			BGT	1\$:YES--BRANCH	
7416	040054	004737	045012		JSR	PC,POPQUE	:NO--REMOVE REQUEST FROM QUEUE	
7417	040060	012762	140000	000016	MOV	#BIT15:BIT14,16(R2)	:SET OFFLINE STATUS/ERROR INDICATOR	
7418	040066	105761	036610		TSTB	DRVSTA(R1)	:IS DRIVE UNSAFE ?	
7419	040072	100053			BPL	8\$:BR TO EXIT IF NOT	
7420	040074	012762	110000	000016	MOV	#BIT15:BIT12,16(R2)	:SET UNSAFE STATUS/ERROR INDICATOR	
7421	040102	000447			BR	8\$:BRANCH TO EXIT	
7422	040104			1\$:				
7423				:	MOV	#111, -(SP)	:LOAD COMMAND ONTO THE STACK	
7424				:	JSR	RD, WRT.RM	:LOAD THE REGISTER	
7425				:	JSR	RMCS1	:REGISTER INCREMENT	
7426				:	B\$:ERROR RETURN ADDRESS	
7427				:	BIT	#BIT11, (R4)	:DRIVE AVAILABLE ?	
7428				:	BEQ	9\$:BR IF NOT	
7429	040104	122762	000150	000002	CMPB	#150,2(R2)	:IS THE REQUEST FOR I/O?	
7430	040112	002403			BLT	2\$:YES--BRANCH	
7431	040114	004737	040440		JSR	PC, CI4	:CALL THE COMMAND INITIATOR	
7432	040120	000440			BR	8\$:BRANCH TO EXIT	
7433	040122	005737	036722	2\$:	TST	DTUW	:DATA TRANSFER UNDERWAY?	
7434	040126	002012			BGE	4\$:YES--GO START A SEARCH	
7435	040130	005737	036700		TST	SEEKFG	:DO IMPLIED SEEKS?	
7436	040134	100404			BMI	3\$:YES---BRANCH	
7437	040136	004037	041412		JSR	RD, LA	:NO--DO LOOK AHEAD	
7438	040142	000427			BR	8\$:RETURN HERE ON A PARITY ERROR	
7439	040144	000403			BR	4\$:GO START A SEARCH	
7440	040146	004737	040232	3\$:	JSR	PC, CI1	:START A DATA TRANSFER	
7441	040152	000423			BR	8\$		
7442	040154	004737	040340	4\$:	JSR	PC, CI3	:START A SEARCH	
7443	040160	000420			BR	8\$:GO TO THE EXIT	
7444	040162	112761	177777	036640	5\$:	MOVB	#-1, DPRQS(R1)	:SET PORT REQUEST INDICATOR
7445	040170	010103			MOV	R1, R3	:SET UP TO ADDRESS WORDS	
7446	040172	006303			ASL	R3	:CONVERT TO WORD INDEX	
7447	040174	012763	060000	036702	MOV	#60000, TIMER(R3)	:START 10 SEC TIMER	
7448	040202	000402			BR	7\$:EXIT	
7449	040204	004737	041054	6\$:	JSR	PC, CI7	:PROCESS THE PARITY ERROR	
7450	040210	032714	000100	7\$:	BIT	#BIT06, (R4)	:SEE IF 'IE' ALREADY SET	
7451	040214	001002			BNE	8\$:BR IF SET	
7452	040216	004737	044252		JSR	PC, SET.IE	:SET "IE" WITHOUT A "TRE"	
7453	040222	012637	177776	8\$:	MOV	(SP)+, @#PS	:RESTORE PROC. STATUS	
7454	040226	104413			RESREG		:RESTORE R0 - R5	
7455	040230	000207			RTS	PC		
7456								
7457								
7458								
7459								
7460								
7461								
7462								

:COMMAND INITIATOR

:CALL

MOV	#DRVNUM, R1	:DRIVE NUMBER
MOV	#DPB, R2	:ADDRESS OF DPB
JSR	PC, CI?	:CI? = CI1, CI3, OR CI4

7519	040466	004037	043742		JSR	RO,WRT.RM	
7520	040472	000006			RMDA		
7521	040474	041054			CI7		
7522	040476	000403			BR	2\$:GO LOAD CYLINDER
7523	040500	122703	000105	1\$:	CMPB	#105,R3	:IS IT A SEEK COMMAND
7524	040504	001007			BNE	3\$:BRANCH IF NO
7525	040506	016246	000012	2\$:	MOV	12(R2),-(SP)	:LOAD DESIRED CYLINDER
7526	040512	004037	043742		JSR	RO,WRT.RM	
7527	040516	000034			RMDC		
7528	040520	041054			CI7		
7529	040522	000546			BR	CI6	
7530	040524	122703	000115	3\$:	CMPB	#115,R3	:IS IT AN "OFFSET" COMMAND?
7531	040530	001013			BNE	4\$:BR IF NO
7532	040532	004037	043562		JSR	RO,RO.RM	:MERGE THE OFFSET VALUE INTO RMOF
7533	040536	000032			RMOF		:BUT DON'T CHANGE THE UPPER
7534	040540	041054			CI7		
7535	040542	116216	000001		MOVB	1(R2),(SP)	:BYTE WHEN LOADING THE
7536	040546	004037	043742		JSR	RO,WRT.RM	:REGISTER (RMOF)
7537	040552	000032			RMOF		
7538	040554	041054			CI7		
7539	040556	000530			BR	CI6	:GO START THE COMMAND
7540	040560	122703	000107	4\$:	CMPB	#107,R3	:IS IT A "RECALIBRATE" COMMAND?
7541	040564	001525			BEQ	CI6	:BRANCH IF YES
7542	040566	122703	000117		CMPB	#117,R3	:IS IT A RETURN TO CENTER?
7543	040572	001522			BEQ	CI6	:BRANCH IF YES
7544	040574	122703	000103		CMPB	#103,R3	:IS IT AN "UNLOAD" COMMAND?
7545	040600	001016			BNE	5\$:BRANCH IF NO
7546	040602	112761	000001	036600	MOVB	#1,DRVACT(R1)	:SET THE DRIVE ACTIVE INDICATOR
7547	040610	105061	036610		CLRB	DRVSTA(R1)	:PUT DRIVE STATUS TO OFFLINE
7548	040614	112761	000001	036656	MOVB	#1,ULDFLG(R1)	:SET "UNLOAD IN PROGRESS" FLAG
7549	040622	010346			MOV	R3,-(SP)	:START THE "UNLOAD" COMMAND
7550	040624	004037	043742		JSR	RO,WRT.RM	
7551	040630	000000			RMCS1		
7552	040632	041054			CI7		
7553	040634	000207			RTS	PC	:RETURN TO USER
7554	040636	122703	000143	5\$:	CMPB	#143,R3	:IS IT A "SET FORMAT" COMMAND?
7555	040642	001014			BNE	6\$:BRANCH IF NO
7556	040644	004037	043562		JSR	RO,RO.RM	:READ THE OFFSET REGISTER
7557	040650	000032			RMOF		
7558	040652	041054			CI7		
7559	040654	116266	000001	000001	MOVB	1(R2),1(SP)	:COMBINE "FMT22" "ECI" AND "HCI"
7560	040662	004037	043742		JSR	RO,WRT.RM	:LOAD "FMT22", "ECI", AND/OR "HCI".
7561	040666	000032			RMOF		
7562	040670	041054			CI7		
7563	040672	000436			BR	12\$	
7564	040674	122703	000141	6\$:	CMPB	#141,R3	:IS IT A "GET REGISTER" COMMAND?
7565	040700	001023			BNE	10\$:BRANCH IF NO
7566	040702	016203	000006	7\$:	MOV	6(R2),R3	:POINTS TO 1ST ADDRESS OF WHERE
7567							:TO PUT THE REGISTER(S)
7568	040706	116237	000010	040724	MOVB	10(R2),9\$:INIT. THE INDEX FOR THE FIRST REG.
7569	040714	116205	000011		MOVB	11(R2),R5	:INDEX OF LAST REG. TO MOVE
7570	040720	004037	043562	8\$:	JSR	RO,RO.RM	:READ RH70/RMO3 REGISTER
7571	040724	000000		9\$:	RMCS1		:INDEX OF REG. TO READ
7572	040726	041054			CI7		
7573	040730	012623			MOV	(SP)+,(R3)+	:GET THE CONTENTS OF RH70/RMO3 REG.
7574	040732	023705	040724		CMP	9\$,R5	:LAST REG. BEEN READ?

7575	040736	001414			BEQ	12\$:GET OUT IF YES
7576	040740	062737	000002	040724	ADD	#2,9\$:INCREASE THE INDEX BY 2
7577	040746	000764			BR	8\$:LOOP--MORE TO READ
7578	040750	122703	000145		10\$: CMPB	#145,R3		:IS IT A "SELECT DRIVE" COMMAND?
7579	040754	001405			BEQ	12\$:BRANCH IF YES
7580	040756	010346			11\$: MOV	R3,-(SP)		:LOAD THE COMMAND
7581	040760	004037	043742		JSR	RO,WRT.RM		
7582	040764	000000			RMCS1			
7583	040766	041054			CI7			
7584	040770	004737	045012		12\$: JSR	PC,POPQUE		:REMOVE REQ. FROM QUEUE
7585	040774	052762	000200	000016	BIS	#BIT07,16(R2)		:SET THE "DONE" BIT
7586	041002	005737	036676		TST	SAVEFG		:SAVE THE RH70/RMO3 REGISTERS?
7587	041006	100002			BPL	13\$:BRANCH IF NO
7588	041010	004737	044134		JSR	PC,SVRH70		:YES--GO SAVE THE REGISTERS
7589	041014	000207			13\$: RTS	PC		:RETURN TO USER
7590	041016	006301			CI5: ASL	R1		
7591	041020	012761	060000	036702	MOV	#60000,TIMER(R1)		:SET A ONE SECOND TIMER
7592	041026	006201			ASR	R1		
7593	041030	112761	000001	036600	MOVB	#1,DRVACT(R1)		:SET THE DRIVE ACTIVE
7594	041036	000207			RTS	PC		:RETURN TO THE USER
7595	041040	010346			16\$: MOV	R3,-(SP)		:LOAD THE COMMAND
7596	041042	004037	043742		JSR	RO,WRT.RM		
7597	041046	000000			RMCS1			
7598	041050	041054			CI7			
7599	041052	000761			BR	CI5		
7600	041054	032764	010000	000010	CI7: BIT	#BIT12,RMCS2(R4)		:DRIVE NON-EXISTENT ?
7601					BNE	CI8		:BR IF YES
7602	041062	005702			15: TST	R2		:ANYTHING IN QUEUE ?
7603					;	BEQ	CI7B	:BR IF NOT
7604	041064	001001			BNE	2\$:BRANCH IF QUEUE IS THERE
7605	041066	000207			RTS	PC		:OTHERWISE EXIT
7606	041070	012762	104000	000016	2\$: MOV	#BIT15!BIT11,16(R2)		:SET "PARITY" ERROR INDICATOR
7607					JSR	PC,SVRH70		:GO SAVE THE RH70/RMO3 REGISTERS
7608	041076	012746	000111		CI7B: MOV	#11,-(SP)		:DO A "DRIVE CLEAR"
7609	041102	004037	043742		JSR	RO,WRT.RM		
7610	041106	000000			RMCS1			
7611	041110	041154			CI8			
7612	041112	004737	044674		2\$: JSR	PC,EMPTYQ		:EMPTY THE QUEUE
7613	041116	105061	036640		CLRB	DPRQS(R1)		:CLEAR THE PORT REQUEST FLAG
7614	041122	105061	036656		CLRB	ULDFLG(R1)		:CLEAR THE UNLOAD IN QUEUE FLAG
7615	041126	105061	036600		CLRB	DRVACT(R1)		:DRIVE IS IDLE
7616	041132	020237	036650		CMP	R2,TRNSWT		:IF THIS DRIVE HAD AN I/O REQUEST
7617					;	R1,DTUW		:IF THIS DRIVE HAD AN I/O REQUEST
7618	041136	001005			BNE	1\$:IN PROGRESS CLEAR ALL OF THE FLAGS
7619	041140	005037	036650		CLR	TRNSWT		
7620	041144	012737	177777	036722	MOV	#-1,DTUW		
7621	041152	000207			1\$: RTS	PC		
7622	041154	104412			CI8: SAVREG			:SAVE R0 - R5
7623	041156	032764	010000	000010	BIT	#BIT12,RMCS2(R4)		:IS 'NED' SET ?
7624					;	1\$:BR IF YES
7625	041164	005001			CLR	R1		
7626	041166	005003			CLR	R3		
7627	041170	105761	036600		1\$: TSTB	DRVACT(R1)		:DRIVE ACTIVE?
7628					;	5\$:BRANCH IF NO
7629	041174	001003			BNE	22\$:BRANCH IF IN ACTIVE
7630	041176	105761	036640		TSTB	DPRQS(R1)		:PORT REQUEST

E14

```

7631 041202 001443          BEQ      5$          ;BRANCH IF NOT
7632 041204 013702 036650 22$:  MOV      TRANSW,R2    ;GET THE "TRANSFER WAIT" QUEUE
7633 041210 020137 036722    CMP      R1,DTUW     ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
7634 041214 001402          BEQ      2$          ;BRANCH IF YES
7635 041216 004737 044770    JSR      PC,GETREQ   ;GET THE DPB POINTER
7636 041222 005702          TST      R2          ;QUEUE ENTRY FOR DRIVE ?
7637 041224 001413          BEQ      4$          ;BR IF NOT
7638 041226 032764 010000 000010 BIT      #BIT12,RMCS2(R4) ;'NED' SET ?
7639 041234 001404          BEQ      3$          ;BR IF NOT
7640 041236 012762 100002 000016 MOV      #BIT15!BIT01,16(R2) ;SET 'DRIVE NON-EXISTENT' INDICATOR
7641 041244 000403          BR       4$          ;CONTINUE
7642 041246 012762 102000 000016 3$:  MOV      #BIT15!BIT10,16(R2) ;SET "NON-CLEARABLE PARITY" ERROR INDICATOR
7643                                     JSR      PC,SVRH70   ;SAVE RH70/RMO3 REGISTERS
7644 041254 012763 177777 036702 4$:  MOV      #-1,TIMER(R3) ;STOP THE TIMER
7645 041262 105061 036600    CLRB    DRVACT(R1)   ;SET "DRIVE ACTIVE" TO IDLE
7646 041266 105061 036640    CLRB    DPRQS(R1)   ;CLEAR PORT REQUEST FLAG
7647 041272 020137 036722    CMP      R1,DTUW     ;IS THIS DRIVE SETUP FOR A TRANSFER
7648 041276 001005          BNE     5$          ;BR IF NOT
7649 041300 012737 177777 036722    MOV      #-1,DTUW    ;RESET THE INDICATOR
7650 041306 005037 036650    CLR     TRNSWT       ;CLEAR THE TRANSFER QUEUE
7651 041312 105061 036656 5$:  CLRB    ULDFLG(R1)   ;CLEAR UNLOAD FLAG
7652 041316 032764 010000 000010 BIT      #BIT12,RMCS2(R4) ;'NED' SET ?
7653                                     BNE     6$          ;BR IF YES
7654 041324 005201          INC     R1          ;MOVE TO THE NEXT DRIVE
7655 041326 062703 000002    ADD     #2,R3
7656 041332 042701 177770    BIC     #1<7,R1
7657 041336 001314          BNE     1$          ;BRANCH IF MORE DRIVES
7658 041340 012737 177777 036722    MOV      #-1,DTUW    ;NO DATA TRANSFERS UNDERWAY
7659 041346 005037 036650    CLR     TRNSWT       ;CLEAR THE 'TRANSFER WAIT' QUEUE
7660 041352 004737 044616    JSR      PC,CLRQUE   ;CLEAR ALL OF THE REQUEST QUEUES
7661 041356 012764 000040 000010 MOV      #BIT05,RMCS2(R4) ;DO A MASSBUS INIT.
7662 041364 000406          BR       7$          ;CONTINUE
7663 041366 004737 044674 6$:  JSR      PC,EMPTYQ   ;CLEAR THE DRIVE'S QUEUE
7664 041372 105061 036610    CLRB    DRVSTA(R1)  ;SET DRIVE TO OFFLINE
7665 041376 105061 036620    CLRB    DRVSTYP(R1) ;CLEAR THE DRIVE TYPE INDICATOR
7666 041402 004737 044252 7$:  JSR      PC,SET.IE   ;SET "IE" WITHOUT "TRE"
7667 041406 104413          RESREG ;RESTORE R0 - R5
7668 041410 000207          RTS     PC          ;RETURN
7669
7670                                     ;LOOK AHEAD ROUTINE
7671                                     ;CALL
7672                                     ;
7673                                     ;
7674                                     ;
7675                                     ;
7676                                     ;
7677                                     ;
7678                                     ;
7679                                     ;
7680 LA:  MOV      RMADR,R4 ;GET RMCS1'S ADDRESS
7681     MOV      R1,RMCS2(R4) ;SELECT DRIVE
7682     JSR      RO,RO.RM ;READ DRIVE STATUS
7683     RMDS
7684     4$
7685     BIC     #1<020200,(SP) ;ON CYLINDER ?
7686     CMP     #200,(SP)+ ;PIP=0,DRY=1?
  
```



```

7687 041442 001044          BNE      3$          ;NO
7688 041444 105261 036666    INCB     LACNT(R1)    ;INCREMENT THE LOOK AHEAD COUNT
7689 041450 126137 036666 036744    CMPB    LACNT(R1),MXLACT ;EXCEED MAX?
7690 041456 003033          BGT     2$          ;BRANCH IF YES
7691 041460 116203 000010    MOVB    10(R2),R3   ;GET DESIRED SECTOR ADDRESS AND
7692 041464 000303          SWAB    R3          ;MULT. BY 64--ALIGN WITH
7693 041466 006203          ASR     R3          ;LOOK AHEAD REGISTER
7694 041470 006203          ASR     R3
7695 041472 012737 000340 177776    MOV     #340,R#PS   ;PRIORITY LEVEL "7"
7696 041500 004037 043562 6$:      JSR     R0,RD.RM    ;READ LOOK AHEAD REGISTER
7697 041504 000020          RMLA
7698 041506 041560          4$
7699 041510 021664 000020    CMP     (SP),RMLA(R4) ;CORRECT LA NUMBER ?
7700 041514 001402          BEQ     7$          ;YES
7701 041516 005726          TST     (SP)+       ;NO,CLEAR STACK
7702 041520 000415          BR      3$
7703 041522 162603          7$:     SUB     (SP)+,R3   ;CALCULATE THE DELTA
7704 041524 002002          BGE     1$
7705 041526 062703 004000    ADD     #(<32.*64.>),R3 ;MAKE THE DELTA POSITIVE
7706 041532 023703 036746 1$:     CMP     MXDLTA,R3   ;CHECK THE DELTA TO SEE
7707 041536 002406          BLT     3$          ;IF IT IS WITHIN THE
7708 041540 023703 036750    CMP     MNDLTA,R3   ;WINDOW---IF YES, ZERO
7709 041544 002003          BGE     3$          ;THE LOOK AHEAD COUNT
7710 041546 105061 036666 2$:     CLRB    LACNT(R1)  ;AND TAKE THE I/O EXIT
7711 041552 005720          TST     (R0)+
7712 041554 005720 3$:     TST     (R0)+       ;ADJUST THE RETURN ADDRESS
7713 041556 000402          BR      5$
7714 041560 004737 041054 4$:     JSR     PC,C17
7715 041564 000200 5$:     RTS     R0
7716
7717 ;INTERRUPT SERVICE ROUTINE
7718
7719 041566 112737 000001 036654 ISR:    MOVB    #1,ACTDRV   ;SET "ACTIVE DRIVER" FLAG
7720 041574 104412          SAVREG   ;SAVE R0 - R5
7721 041576 013704 036736    MOV     RMADR,R4    ;ADDRESS OF RHSCS1
7722 041602 013701 036722    MOV     DTUW,R1     ;GET "DATA TRANSFER UNDERWAY" INDICATOR
7723 041606 002403          BLT     1$          ;BRANCH IF NO DATA TRANSFER UNDERWAY
7724 041610 004737 041632    JSR     PC,TD
7725 041614 000402          BR      2$
7726 041616 004737 042002 1$:     JSR     PC,SC
7727 041622 104413 2$:     RESREG
7728 041624 105037 036654    CLRB    ACTDRV
7729 041630 000002          RTI
7730
7731 ;TRANSFER DONE ROUTINE
7732
7733 041632 105061 036600 7733:  CLRB    DRVACT(R1)  ;SET DRIVE ACTIVE INDICATOR TO IDLE
7734 041636 012737 177777 036722    MOV     #-1,DTUW    ;NO DATA TRANSFERS UNDERWAY
7735 041644 006301          ASL     R1
7736 041646 012761 177777 036702    MOV     #-1,TIMER(R1) ;CANCEL TIMEOUT
7737 041654 006201          ASR     R1
7738 041656 013702 036650    MOV     TRNSWT,R2   ;GET "DPB" ADDRESS FROM THE
7739 041662 005037 036650    CLR     TRNSWT      ;TRANSFER WAIT QUEUE--CLEAR QUEUE
7740 041666 052762 000200 000016  BIS     #BIT07,16(R2) ;SET DONE
7741 041674 010164 000010    MOV     R1,RMC$2(R4) ;SELECT THE DRIVE
7742 041700 004037 043562    JSR     R0,RD.RM    ;TRANSFER ERROR(TRE=1)?
    
```


Line	Address	Offset	Value	Label	Instruction	Comment
7799	042114	105761	036610	2\$:	TSTB DRVSTA(R1)	;CHECK THE DRIVE STATUS
7800	042120	003025			BGT 5\$;BRANCH IF ONLINE
7801	042122	105761	036656		TSTB ULDFLG(R1)	;UNLOAD IN PROGRESS?
7802	042126	003422			BLE 5\$;BRANCH IF NOT
7803	042130	004737	044770		JSR PC,GETREQ	;GET DPB POINTER
7804	042134	004737	044134		JSR PC,SVRH70	;SAVE THE RH70/RM03 REGISTERS
7805	042140	004737	042750		JSR PC,SC12	;SAVE RMD5, RMER1, RMER2, AND RMMR2
7806						;ALSO DO A DRIVE INIT (DRVINT)
7807	042144	105761	036610		TSTB DRVSTA(R1)	;DID DRIVE COME ONLINE?
7808	042150	003416			BLE 6\$;NO---BRANCH
7809	042152	032737	040000	036570	BIT #BIT14,RMERRS	;WAS THERE AN ERROR?
7810	042160	001002			BNE 3\$;BR IF ERROR
7811	042162	000137	042640		JMP SC11	;NO ERROR
7812	042166	013705	036572	3\$:	MOV RMERRS+2,R5	;YES -- PICKUP RMER1 AND
7813	042172	000504			BR SC6A	;GO PROCESS THE ERROR
7814	042174	105761	036600	5\$:	TSTB DRVACT(R1)	;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
7815	042200	001033			BNE SC6	;BR IF EITHER
7816	042202	004737	042750		JSR PC,SC12	;SAVE RMD5, RMER1, RMER2, AND RMMR2
7817						;ALSO DO A 'DRVINT'
7818	042206	105761	036630	6\$:	TSTB DPINT(R1)	;TRYING TO INIT THE DRIVE ?
7819	042212	001321			BNE SC4	;BR IF YES, CHECK ON MORE DRIVES
7820	042214	105761	036610		TSTB DRVSTA(R1)	;CHECK ON DRIVE'S STATUS
7821	042220	100412			BMI 7\$;BR IF UNSAFE
7822	042222	032737	020000	036574	BIT #BIT13,RMERRS+4	;ADDRESS PLUG CHANGED ?
7823	042230	001013			BNE 8\$;BR IF YES
7824					MOV #113,-(SP)	;RELEASE COMMAND
7825	042232	012746	000111		MOV #111,-(SP)	;DRIVE CLEAR
7826	042236	004037	043742		JSR RD,WRT.RM	;WRITE THE COMMAND INTO RMCS1
7827	042242	000000			RMCS1	;REGISTER INDEX
7828	042244	042610			SCB	;PARITY EXIT ADDRESS
7829	042246	011605		7\$:	MOV (SP),R5	;PICKUP (RMAS) BEFORE THE ERROR CALL
7830	042250	004037	045060		JSR RD,ES.SAV	;SAVE THE ADDRESS IN 'SESCAPE'
7831	042254	104002			ERROR 2	;REPORT THE UNEXPECTED ATTENTION
7832	042256	000677			BR SC4	;GO CHECK FOR MORE ATA'S
7833	042260			8\$:		
7834	042260	004037	045060		JSR RD,ES.SAV	;SAVE THE ADDRESS IN 'SESCAPE'
7835	042264	104005			ERROR 5	;REPORT THE ADDRESS PLUG CHANGE
7836	042266	000673			BR SC4	;CHECK FOR MORE DRIVES
7837	042270	006301		SC6:	ASL R1	;SETUP TO ADDRESS WORDS
7838	042272	012761	177777	036702	MOV #-1,TIMER(R1)	;STOP THE TIMER
7839	042300	006201			ASR R1	;RESTORE THE DRIVE ADDRESS
7840	042302	004737	044770		JSR PC,GETREQ	;GET THE DPB POINTER FROM THE QUEUE
7841	042306	010164	000010		MOV R1,RMCS2(R4)	;SELECT DRIVE
7842	042312	000137	042640		JMP SC11	;PROCESS THE SEARCH
7843	042316	004037	043562		JSR RD,RD.RM	;READ THE RM03'S STATUS REG.
7844	042322	000012			RMD5	
7845	042324	042610			SCB	
7846	042326	011605			MOV (SP),R5	;AND PUT IT IN R5
7847	042330	006126			ROL (SP)+	;WAS THERE AN ERROR?
7848	042332	100407			BMI 1\$;BR IF ERROR
7849	042334	105761	036600		TSTB DRVACT(R1)	;CHECK DRIVE'S STATE
7850	042340	003137			BGT SC11	;BR IF DRIVE ACTIVE WITH ORDER
7851	042342	052762	100210	000016	BIS #BIT15:BIT07:BIT03,16(R2)	;INFORM USER OF ERROR RECOVER COMPLETION
7852	042350	000470			BR SC7	
7853	042352	004037	043562	1\$:	JSR RD,RD.RM	;READ ERROR REGISTER #1
7854	042356	000014			RMER1	

7855	042360	042610				SC8			
7856	042362	012605				MOV	(SP)+,R5		:AND SAVE IT IN R5
7857	042364	004737	044134			JSR	PC,SVRH70		:SAVE RH70/RM03 REGISTERS
7858	042370	012746	000111			MOV	#111,-(SP)		:ISSUE A DRIVE CLEAR
7859	042374	004037	043742			JSR	RO,WRT.RM		
7860	042400	000000				RMCS1			
7861	042402	042610				SC8			
7862	042404	006105			SC6A:	ROL	R5		:WAS "UNSAFE" CONDITION =1?
7863	042406	100406				BMI	1\$:BRANCH IF YES
7864	042410	005702				TST	R2		:ANYTHING IN QUEUE ?
7865	042412	001447				BEQ	SC7		:BR IF NOT
7866	042414	052762	100240	000016		BIS	#BIT15!BIT07!BIT05,16(R2)		:INFORM USER OF ERROR
7867	042422	000443				BR	SC7		
7868	042424	004037	043562		1\$:	JSR	RO,RD.RM		:READ DRIVE STATUS REG. #1
7869	042430	000012				RMDS			
7870	042432	042610				SC8			
7871	042434	011605				MOV	(SP),R5		:SAVE RMDS IN R5
7872	042436	006126				ROL	(SP)+		: "ERR" = 1 ?
7873	042440	100011				BPL	2\$:BR IF NO--UNSAFE CLEARED
7874	042442	112761	177777	036610		MOVB	#-1,DRVSTA(R1)		:DRIVE IS UNSAFE
7875	042450	004737	044134			JSR	PC,SVRH70		:SAVE RH70/RM03 REGISTERS
7876	042454	052762	110000	000016		BIS	#BIT15!BIT12,16(R2)		:INFORM USER OF UNSAFE ERROR
7877	042462	000423				BR	SC7		
7878	042464	032705	010000		2\$:	BIT	#BIT12,R5		: "MOL" = 1 ?
7879	042470	001015				BNE	3\$:BR IF YES
7880	042472	112761	177777	036600		MOVB	#-1,DRVACT(R1)		:ACTIVE ERROR RECOVER
7881	042500	112761	000001	036610		MOVB	#1,DRVSTA(R1)		:ONLINE
7882	042506	006301				ASL	R1		
7883	042510	012761	072460	036702		MOV	#30000.,TIMER(R1)		:START 30 SECOND TIMER
7884	042516	006201				ASR	R1		
7885	042520	000137	042056			JMP	SC4		
7886	042524	052762	100220	000016	3\$:	BIS	#BIT15!BIT07!BIT04,16(R2)		:INFORM USER OF ERROR
7887	042532	105061	036600		SC7:	CLRB	DRVACT(R1)		:DRIVE IS IDLE
7888					:	JSR	PC,EMPTYQ		:DUMP THE QUEUE
7889	042536	004737	045012			JSR	PC,POPQUE		:REMOVE THE QUEUE
7890	042542	105761	036656			TSTB	ULDFLG(R1)		:UNLOAD IN RMOGRESS OR QUEUE?
7891	042546	003002				BGT	1\$:BR IF NOT
7892	042550	105061	036656			CLRB	ULDFLG(R1)		:CLEAR UNLOAD FLAG
7893	042554	116164	036724	000016	1\$:	MOVB	ATABIT(R1),RMAS(R4)		:CLEAR ATTENTION BIT
7894	042562	105761	036610			TSTB	DRVSTA(R1)		:IS THE DRIVE UNSAFE ?
7895	042566	100406			2\$:	BMI	2\$:BR IF IT IS
7896					:	MOV	#113,-(SP)		:RELEASE COMMAND
7897	042570	012746	000111			MOV	#111,-(SP)		:DRIVE CLEAR COMMAND
7898	042574	004037	043742			JSR	RO,WRT.RM		:WRITE THE COMMAND INTO RPCS1
7899	042600	000000				RMCS1			:REGISTER INDEX
7900	042602	042610				SC8			:PARITY EXIT ADDRESS
7901	042604	000137	042056		2\$:	JMP	SC4		:CHECK FOR MORE DRIVES
7902	042610	105761	036600		SC8:	TSTB	DRVACT(R1)		:IS DRIVE IDLE?
7903	042614	001405				BEQ	1\$:YES--BRANCH
7904	042616	004737	044770			JSR	PC,GETREQ		:GET DPB POINTER
7905	042622	004737	041054			JSR	PC,CI7		:PROCESS THE PARITY ERROR
7906	042626	000402				BR	2\$:CONTINUE
7907	042630				1\$:				
7908					:	JSR	PC,CI7		:PROCESS THE PARITY ERROR
7909	042630	004737	041076			JSR	PC,CI7B		:PROCESS THE UNCORRECTABLE PARITY ERROR
7910	042634	000137	042056		2\$:	JMP	SC4		:CHECK MORE DRIVES

7911	042640	105761	036656		SC11:	TSTB	ULDFLG(R1)	:"UNLOAD IN PROGRESS"?
7912	042644	003402				BLE	1\$:BRANCH IF NO
7913	042646	105061	036656			CLRB	ULDFLG(R1)	:CLEAR UNLOAD FLAG
7914	042652	105061	036600		1\$:	CLRB	DRVACT(R1)	:SET DRIVE IDLE
7915	042656	136137	036724	036652		BITB	ATABIT(R1),SRCHWT	:DOING A SEARCH OPERATION FOR :AN I/O COMMAND?
7916								
7917	042664	001012				BNE	2\$:BRANCH IF YES
7918	042666	004737	045012			JSR	PC,POPQUE	:REMOVE REQUEST FROM QUEUE
7919	042672	052762	000200	000016		BIS	#BIT07,16(R2)	:SET "DONE" BIT
7920	042700	005737	036676			TST	SAVEFG	:SAVE THE REGISTERS?
7921	042704	100002				BPL	2\$:BRANCH IF NO
7922	042706	004737	044134			JSR	PC,SVRH70	:YES--SAVE ALL OF THE RH70/RMO3 REG'S
7923	042712	116164	036724	000016	2\$:	MOVVB	ATABIT(R1),RMAS(R4)	:CLEAR ATTENTION BIT
7924	042720	146137	036724	036652		BICB	ATABIT(R1),SRCHWT	:CLEAR IMPLIED SEEK SET
7925	042726	006301				ASL	R1	:WORD INDEX
7926	042730	012761	177777	036702		MOV	#-1,TIMER(R1)	:STOP CLOCK
7927	042736	006201				ASR	R1	:RESTORE R1
7928	042740	004737	037774			JSR	PC,OPT	:START A REQUEST
7929	042744	000137	042056			JMP	SC4	:CHECK FOR MORE DRIVES
7930	042750	010164	000010		SC12:	MOV	R1,RMCS2(R4)	:SELECT DRIVE
7931	042754	016437	000012	036570		MOV	RMDS(R4),RMERRS	:SAVE THE FOUR REGISTERS THAT
7932	042762	016437	000014	036572		MOV	RMER1(R4),RMERRS+2	:WILL TELL US SOMETHING
7933	042770	016437	000042	036574		MOV	RMER2(R4),RMERRS+4	
7934	042776	016437	000040	036576		MOV	RMMR2(R4),RMERRS+6	
7935						JSR	RD,DRVINT	:INIT. THE STATE OF THE DRIVE
7936						BR	1\$:TAKE ERROR EXIT
7937	043004	000207				RTS	PC	:RETURN
7938	043006	005726			1\$:	TST	(SP)+	:POP PC OFF OF THE STACK
7939	043010	000677				BR	SC8	:PROCESS THE PARITY ERROR
7940	043012	006301			SC13:	ASL	R1	:SETUP TO ADDRESS WORDS
7941	043014	012761	177777	036702		MOV	#-1,TIMER(R1)	:STOP THE TIMER
7942	043022	006201				ASR	R1	
7943	043024	010164	000010			MOV	R1,RMCS2(R4)	:SELECT THE DRIVE
7944	043030	116164	036724	000016		MOVVB	ATABIT(R1),RMAS(R4)	:CLEAR THE ATTENTION BIT
7945	043036	105761	036630		1\$:	TSTB	DPINT(R1)	:INITIALIZING THE DRIVE ?
7946	043042	001424				BEQ	2\$:BR IF NOT
7947	043044	105061	036630			CLRB	DPINT(R1)	:CLEAR THE INIT INDICATOR
7948	043050	004037	037166			JSR	RD,DRVINT	:GO INIT THE DRIVE
7949	043054	000240				NOP		:DUMMY PARITY ERROR RETURN
7950	043056	105761	036610			TSTB	DRVSTA(R1)	:DRIVE ONLINE ?
7951	043062	003014				BGT	2\$:BR IF YES -- START ORDER
7952	043064	005702				TST	R2	:QUEUE ENTRY FOR THE DRIVE
7953	043066	001426				BEQ	3\$:BR IF NOT
7954	043070	004737	044770			JSR	PC,GETREQ	:GET DPB ADDRESS
7955	043074	052762	140000	000016		BIS	#BIT15:BIT14,16(R2)	:INFORM USER THAT DRIVE OFFLINE
7956	043102	004737	044134			JSR	PC,SVRH70	:SAVE THE REGISTERS
7957						JSR	PC,EMPTYQ	:EMPTY THE REQUEST QUEUE
7958	043106	004737	045012			JSR	PC,POPQUE	:REMOVE THE QUEUE
7959	043112	000414				BR	3\$	
7960	043114	032764	004000	000000	2\$:	BIT	#BIT11,RMCS1(R4)	:DVA SET ?
7961	043122	001006				BNE	4\$:SET THEN CALL OPT
7962	043124	006301				ASL	R1	
7963	043126	012761	060000	036702		MOV	#60000,TIMER(R1)	
7964	043134	006201				ASR	R1	
7965	043136	000402				BR	3\$	
7966	043140	004737	037774		4\$:	JSR	PC,OPT	:START THE PENDING REQUEST


```

7967 043144 000137 042056 3$: JMP SC4 ;PROCESS OTHER DRIVES
7968
7969 ;/RMO3 TIMER ROUTINE
7970 ;CALL
7971 ; MOV #TIME, -(SP) ;ELAPSED TIME IN MILLISECONDS ON THE STACK
7972 ; JSR PC, RPTMR ;CALL RMO3 TIME ROUTINE
7973
7974 043150 005737 036654 RPTMR: TST ACTDRV ;CHECK "ACTDRV & ACTSTR"
7975 043154 001027 4$ BNE ;IF NON ZERO EXIT
7976 043156 112737 000001 036655 MOVB #1, ACTSTR ;SET "ACTSTR"
7977 043164 104412 SAVREG ;SAVE R0 - R5
7978 043166 005001 CLR R1 ;START WITH DRIVE 0
7979 043170 005003 CLR R3
7980 043172 005763 036702 1$: TST TIMER(R3) ;IS THE TIMER RUNNING?
7981 043176 002406 BLT 2$ ;BRANCH IF NO
7982 043200 166663 000002 036702 SUB 2(SP), TIMER(R3) ;COUNT THE INTERVAL
7983 043206 003002 BGT 2$ ;BR IF NO SOFTWARE TIMEOUT
7984 043210 004737 043240 JSR PC, STO ;CALL SOFTWARE TIMEOUT ROUTINE
7985 043214 005201 2$: INC R1 ;MOVE TO NEXT DRIVE
7986 043216 005723 TST (R3)+
7987 043220 022701 000010 CMP #8, R1 ;OUT OF DRIVES?
7988 043224 003362 BGT 1$ ;BRANCH IF NO
7989 043226 104413 3$: RESREG ;RESTORE R0 - R5
7990 043230 105037 036655 CLRB ACTSTR ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
7991 043234 012616 4$: MOV (SP)+, (SP) ;ADJUST THE STACK
7992 043236 000207 RTS PC ;RETURN
7993
7994 ;SOFTWARE TIMEOUT ROUTINE
7995
7996 ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
7997 ;OR GREATER
7998
7999 ;CALL: STO
8000 ; MOV #DRVNUM, R1 ;DRIVE NUMBER
8001 ; JSR PC, STO ;CALL
8002 ;
8003 ; RETURN
8004 043240 010146 STO: MOV R1, -(SP) ;SAVE R1
8005 043242 010246 MOV R2, -(SP) ;SAVE R2
8006 043244 010346 MOV R3, -(SP) ;SAVE R3
8007 043246 010446 MOV R4, -(SP) ;SAVE R4
8008 043250 013704 036736 MOV RMADR, R4 ;GET ADDRESS OF "RMCS1"
8009 043254 010164 000010 MOV R1, RMCS2(R4) ;SELECT THE DRIVE
8010 043260 004037 043562 JSR R0, RD.RM ;READ "DRIVE STATUS REG"
8011 043264 000012 RMDS
8012 ;
8013 ;
8014 043266 043550 ;
8015 043270 105726 ;
8016 043272 100436 ;
8017 043274 105761 036630 ST01: TSTB DPINT(R1) ;IS "DRY"=1?
8018 043300 001033 BNE ST02 ;BR IF YES
8019 043302 105761 036640 TSTB DPRQS(R1) ;TRYING TO INTIALIZE THE DRIVE ?
8020 043310 013702 036650 BNE ST02 ;BR IF YES
8021 043314 020137 036722 MOV TRNSWT, R2 ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
8022 043320 001404 CMP R1, DTUW ;BR IF YES
; PICKUP TRANSFER WAIT QUEUE
; TRANSFER UNDERWAY ON THIS DRIVE?
; BRANCH IF YES

```



```

8023 043322 000137 043550      JMP      ST09      ;IF NOT DON'T BOTHER DRIVES
8024 043326 004737 044770      JSR      PC,GETREQ ;GET DPB ADDRESS
8025 043332 052762 101000 000016 1$:  BIS      #BIT15!BIT09,16(R2) ;SET THE ERROR FLAGS
8026 043340 004737 044134      JSR      PC,SVRH70 ;SAVE RH70/RMO3 REGISTERS
8027      ;      MOV      #BIT05, RMCS2(R4) ;"INIT" THE MASS BUS
8028 043344 105061 036600      CLR      DRVACT(R1) ;DRIVE IS IDLE
8029 043350 105061 036656      CLR      ULDFLG(R1) ;CLEAR THE UNLOAD FLAG
8030 043354 005037 036650      CLR      TRNSWT      ;CLEAR DPB ADDRESS
8031 043360 012737 177777 036722  MOV      #-1,DTUW      ;CLEAR THE TRANSFER DRIVE #
8032 043366 000470      BR       ST09      ;DON'T BOTHER OTHER DRIVES
8033 043370 116405 000016      MOV      RMAS(R4),R5 ;READ ATTENTION REG
8034 043374 136105 036724      BIT      ATABIT(R1),R5 ;IS ATTENTION FOR THIS DRIVE UP ?
8035 043400 001007      BNE      ST03      ;YES--BRANCH
8036 043402 105761 036630      TST      DPINT(R1)      ;TRYING TO INITIALIZE THE DRIVE ?
8037 043406 001021      BNE      ST06      ;BR IF YES - DRIVE NOT ONLINE
8038 043410 105761 036640      TST      DPRQS(R1)      ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
8039 043414 001035      BNE      ST07      ;BR IF YES - NO RESPONSE TO REQUEST
8040 043416 000454      BR       ST09      ;OTHER WISE EXIT
8041 043420 105761 036630      TST      DPINT(R1)      ;INITIALIZING THE DRIVE ?
8042 043424 001003      BNE      1$      ;BR IF INIT PENDING
8043 043426 105761 036640      TST      DPRQS(R1)      ;PORT REQUEST PENDING ?
8044 043432 001446      BEQ      ST09      ;BR IF NOT
8045 043434 012763 177777 036702 1$:  MOV      #-1,TIMER(R3) ;STOP THE TIMER
8046 043442 000442      BR       ST09      ;EXIT
8047 043444 004737 041154      JSR      PC,C18      ;GO HANDLE THE PARITY ERROR
8048 043450 000437      BR       ST09
8049 043452 105061 036630      CLR      DPINT(R1)      ;CLEAR THE INITIALIZE INDICATOR
8050 043456 105061 036610      CLR      DRVSTA(R1)      ;SET UNIT OFFLINE
8051 043462 012763 177777 036702  MOV      #-1,TIMER(R3) ;STOP THE TIMER
8052 043470 004737 044770      JSR      PC,GETREQ      ;GET THE DPB ADDRESS
8053 043474 005702      TST      R2      ;REQUEST IN QUEUE ?
8054 043476 001424      BEQ      ST09      ;BR IF NOT
8055 043500 052762 140000 000016  BIS      #BIT15!BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
8056 043506 000414      BR       ST08      ;FINISH
8057 043510 012763 177777 036702  MOV      #-1,TIMER(R3) ;STOP THE TIMER
8058 043516 105061 036640      CLR      DPRQS(R1)      ;CLEAR PORT REQUEST INDICATOR
8059 043522 004737 044770      JSR      PC,GETREQ      ;GET DPB ADDRESS
8060 043526 005702      TST      R2      ;QUEUE ENTRY FOR DRIVE ?
8061 043530 001407      BEQ      ST09      ;BR IF NONE
8062 043532 012762 100004 000016  MOV      #BIT15!BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
8063 043540 004737 044674      JSR      PC,EMPTYQ      ;CLEAR THE QUEUE FOR THE DRIVE
8064 043544 004737 044134      JSR      PC,SVRH70      ;SAVE THE REGISTERS
8065 043550 012604      MOV      (SP)+,R4      ;RESTORE R4
8066 043552 012603      MOV      (SP)+,R3      ;RESTORE R3
8067 043554 012602      MOV      (SP)+,R2      ;RESTORE R2
8068 043556 012601      MOV      (SP)+,R1      ;RESTORE R1
8069 043560 000207      RTS      PC      ;RETURN
8070
8071      ;ROUTINE TO READ A RH70/RMO3 REGISTER
8072
8073      ;CALL
8074      JSR      RO,RD,RM      ;GO READ A REGISTER
8075      INDEX      ;REG. INDEX FROM BASE
8076      ERRADR      ;ERROR ADDRESS--PROCESS ERROR STARTING
8077      ;AT THIS ADDRESS
8078      ;CONTENTS OF REG. IS ON THE STACK
      RETURN

```



```

8079
8080 043562 013737 036734 043730 RD.RM: MOV MCPEMX,RD.RM2 ;MAX. RETRYS ALLOWED
8081 043570 011646 MOV (SP)-,(SP) ;SAVE RD FOR RETURN
8082 043572 013737 036736 043606 MOV RMADR,RD.ADR ;FORM THE DESIRED ADDRESS
8083 043600 062037 043606 ADD (RD)+,RD.ADR ;USING THE BASE AND THE INDEX
8084 043604 013727 RD.RM1: MOV @ (PC)+,(PC)+ ;READ THE DESIRED REGISTER OF THE RM03
8085 043606 000000 RD.ADR: .WORD 0 ;ADDRESS IS FORMED HERE
8086 043610 000000 RD.WRD: .WORD 0 ;REG. CONTENTS PUT HERE
8087 043612 013766 043610 000002 MOV RD.WRD,2(SP) ;RETURN IT TO THE USER
8088 043620 013746 036736 MOV RMADR,-(SP) ;PUT THE ADDRESS ON THE STACK
8089 043624 062716 000010 ADD #RMCS2,(SP) ;FORM THE ADDRESS OF RMCS2
8090 043630 032736 010000 BIT #BIT12,@(SP)+ ;CHECK THE 'NED' BIT
8091 043634 001037 BNE RD.RM3 ;BR IF DRIVE NON-EXISTENT
8092 043636 017746 173074 MOV @RMADR,-(SP) ;READ RMCS1
8093 043642 032716 020000 BIT #BIT13,(SP) ;DID MCPE SET?
8094 043646 001002 BNE 1$ ;BRANCH IF YES
8095 043650 022620 CMP (SP)+,(RD)+ ;ADJUST FOR RETURN
8096 043652 000432 BR RD.RM4 ;EXIT
8097 043654
8098 043654 004037 045060 1$: JSR RD,ES.SAV ;SAVE THE ADDRESS IN 'SESCAPE'
8099 043660 104003 ERROR 3 ;REPORT "MCPE" ERROR
8100 043662 005737 036722 TST DTUW ;DATA TRANSFER UNDERWAY?
8101 043666 100405 BMI 2$ ;NO--BRANCH
8102 043670 032716 040000 BIT #BIT14,(SP) ;NO--"TRE"=1?
8103 043674 001402 BEQ 2$ ;NO--BRANCH
8104 043676 005726 TST (SP)+ ;YES--CLEAN OFF THE STACK AND
8105 043700 000415 BR RD.RM3 ;TAKE THE FATAL ERROR EXIT
8106 043702 052716 040000 2$: BIS #BIT14,(SP) ;CLEAR "MCPE" BY SENDING A "1" TO "TRE"
8107 043706 000316 SWAB (SP) ;POSITION BEFORE WRITING
8108 043710 013737 036736 043724 MOV RMADR,3$ ;FORM ADDRESS OF HIGH BYTE
8109 043716 005237 043724 INC 3$
8110 043722 112637 MOVWB (SP)+,@(PC)+ ;WRITE THE HIGH BYTE OF RMCS1
8111 043724 000000 3$: .WORD 0 ;ADDRESS STORAGE
8112 043726 005327 DEC (PC)+ ;EXCEEDED MAX. RETRYS
8113 043730 000003 RD.RM2: .WORD 3
8114 043732 002324 BGE RD.RM1 ;BRANCH IF NO
8115 043734 011000 RD.RM3: MOV (RD),RD ;FATAL ERROR EXIT
8116 043736 012616 MOV (SP)+,(SP)
8117 043740 000200 RD.RM4: RTS RD
8118
8119 ;ROUTINE TO WRITE A REGISTER
8120
8121 ;CALL
8122
8123 MOV DATA,-(SP) ;DATA TO BE LOADED ON THE STACK
8124 JSR RD,WRT.RM ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
8125 INDEX ;INDEX OF THE REGISTER TO BE LOADED
8126 ERRADR ;ADDRESS TO RETURN TO ON AN ERROR
8127 RETURN ;ERROR FREE RETURN
8128 043742 013737 036734 044120 WRT.RM: MOV MCPEMX,WRT.R2 ;MAX RETRYS ALLOWED
8129 043750 016637 000002 044030 MOV 2(SP),WRT.WD ;SAVE THE WORD TO WRITE
8130 043756 012616 MOV (SP)+,(SP) ;ADJUST THE STACK
8131 043760 012037 044032 MOV (RD)+,WRT.AD ;GET INDEX OF REGISTER TO BE WRITTEN
8132 043764 001015 BNE 1$ ;BRANCH IF NOT RMCS1
8133 043766 122737 000150 044030 CMPB #150,WRT.WD ;IS THE COMMAND FOR DATA TRANSFERS?
8134 043774 002411 BLT 1$ ;YES--DON'T GET THE OLD A16 & A17, & PSEL

```



```

8135 043776 004037 043562 JSR RO, RD.RM ;NO---COMBINE A16&A17, & PSEL WITH
8136 044002 000000 RMCS1 ;THE COMMAND BEFORE SENDING IT TO
8137 044004 044124 WRT.R3 ;THE RH70/RM03
8138 044006 000316 SWAB (SP)
8139 044010 042716 177770 BIC #1C7, (SP)
8140 044014 112637 044031 MOVB (SP)+, WRT.WD+1
8141 044020 063737 036736 044032 1$: ADD RMADR, WRT.AD ;FORM THE ADDRESS OF THE DISK REG.
8142 044026 012737 WRT.R1: MOV (PC)+, @ (PC)+ ;LOAD THE DESIRED REG.
8143 044030 000000 WRT.WD: .WORD 0 ;WORD TO WRITE GOES HERE
8144 044032 000000 WRT.AD: .WORD 0 ;ADDRESS IS FORMED HERE
8145 044034 013746 036736 MOV RMADR, -(SP) ;PUT THE ADDRESS ON THE STACK
8146 044040 062716 000010 ADD #RMCS2, (SP) ;FORM THE ADDRESS OF RMCS2
8147 044044 032736 010000 BIT #BIT12, @ (SP)+ ;CHECK THE 'NED' BIT
8148 044050 001025 BNE WRT.R3 ;BR IF DRIVE NON-EXISTENT
8149 044052 004037 043562 JSR RO, RD.RM ;CHECK FOR PARITY ERROR ON WRITE
8150 044056 000014 RMER1
8151 044060 044124 WRT.R3
8152 044062 032726 000010 BIT #BIT03, (SP)+ ;BRANCH IF "PAR=0"
8153 044066 001420 BEQ WRT.R4 ;PICKUP THE INDEX
8154 044070 016037 177776 044102 MOV -2(R0), 1$ ;READ THE REG.
8155 044076 004037 043562 JSR RO, RD.RM ;REG. INDEX
8156 044102 000000 1$: .WORD 0 ;RETURN TO THIS ADDRESS ON ERROR
8157 044104 044124 WRT.R3 ;SAVE THE ADDRESS IN 'SESCAPE'
8158 044106 004037 045060 JSR RO, ES.SAV ;REPORT THE PARITY ON WRITE ERROR
8159 044112 104004 ERROR 4 ;CLEAR OFF THE STACK
8160 044114 005726 TST (SP)+ ;DECREMENT THE ERROR COUNT
8161 044116 005327 DEC (PC)+ ;RETRY COUNTER
8162 044120 000003 WRT.R2: .WORD 3 ;TRY AGAIN IF NOT FINISHED
8163 044122 002341 BGE WRT.R1 ;TAKE THE "PARITY ON WRITE" ERROR EXIT
8164 044124 011000 WRT.R3: MOV (R0), R0 ;EXIT
8165 044126 000401 WRT.R4: BR WRT.R5 ;ADJUST FOR ERROR FREE EXIT
8166 044130 005720 WRT.R5: TST (R0)+
8167 044132 000200 WRT.R5: RTS R0
8168
8169 ;ROUTINE TO SAVE THE RH70/RP04/5/RM03 REGISTERS AS PER DPB+14
8170
8171 ;CALL
8172
8173 ;
8174 ;
8175 044134 104412 SVRH70: SAVREG ;SAVE R0 - R5
8176 044136 005702 TST R2 ;QUEUE ENTRY FOR THE DRIVE ?
8177 044140 001442 BEQ 6$ ;BR IF NONE
8178 044142 013704 036736 MOV RMADR, R4
8179 044146 111264 000010 MOVB (R2), RMCS2(R4) ;SELECT DRIVE
8180 044152 016203 000014 MOV 14(R2), R3 ;GET THE ERROR TABLE POINTER
8181 044156 001433 BEQ 6$ ;EXIT IF NO ADDRESS
8182 044160 005037 044214 CLR 3$ ;COUNTER & POINTER
8183 044164 023727 044214 000022 1$: CMP 3$, #RMD8 ;REACHED THE BUFFER REGISTER ?
8184 044172 001006 BNE 2$ ;BR IF NOT
8185 044174 032764 000200 000010 BIT #BIT07, RMCS2(R4) ;'OR' SET ?
8186 044202 001002 BNE 2$ ;BR IF SET
8187 044204 005023 CLR (R3)+ ;STORE RMD8 AS ZEROES
8188 044206 000405 BR 4$ ;CONTINUE
8189 044210 004037 043562 2$: JSR RO, RD.RM ;READ THE SELECTED REGISTER
8190 044214 000000 3$: .WORD 0 ;REGISTER INDEX
    
```



```

8247 044360 044456 .WORD QDRV2 ;DRIVE 2
8248 044362 044476 .WORD QDRV3 ;DRIVE 3
8249 044364 044516 .WORD QDRV4 ;DRIVE 4
8250 044366 044536 .WORD QDRV5 ;DRIVE 5
8251 044370 044556 .WORD QDRV6 ;DRIVE 6
8252 044372 044576 .WORD QDRV7 ;DRIVE 7
8253
8254 044374 044416 QSTART: .WORD QDRV0 ;DRIVE 0 START ADDRESS
8255 044376 044436 QSTOP: .WORD QDRV1 ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
8256 044400 044456 .WORD QDRV2 ;STOP DRIVE 1--START DRIVE 2
8257 044402 044476 .WORD QDRV3 ;STOP DRIVE 2--START DRIVE 3
8258 044404 044516 .WORD QDRV4 ;STOP DRIVE 3--START DRIVE 4
8259 044406 044536 .WORD QDRV5 ;STOP DRIVE 4--START DRIVE 5
8260 044410 044556 .WORD QDRV6 ;STOP DRIVE 5--START DRIVE 6
8261 044412 044576 .WORD QDRV7 ;STOP DRIVE 6--START DRIVE 7
8262 044414 044616 .WORD QTERM ;STOP DRIVE 7
8263
8264 ;DRIVE REQUEST QUEUES
8265
8266 044416 000010 QDRV0: .BLKW 10
8267 044436 000010 QDRV1: .BLKW 10
8268 044456 000010 QDRV2: .BLKW 10
8269 044476 000010 QDRV3: .BLKW 10
8270 044516 000010 QDRV4: .BLKW 10
8271 044536 000010 QDRV5: .BLKW 10
8272 044556 000010 QDRV6: .BLKW 10
8273 044576 000010 QDRV7: .BLKW 10
8274 044616 044616 QTERM=.
8275
8276 ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
8277 ;CALL
8278 ;
8279 ; JSR PC,CLRQUE
8280 ;
8281 044616 104412 CLRQUE: SAVREG ;SAVE R0 - R5
8282 044620 012702 044324 MOV #QCNT,R2 ;ZERO THE QUEUE COUNTS
8283 044624 005022 CLR (R2)+ ;DRIVES 0 & 1
8284 044626 005022 CLR (R2)+ ;DRIVES 2 & 3
8285 044630 005022 CLR (R2)+ ;DRIVES 4 & 5
8286 044632 005022 CLR (R2)+ ;DRIVES 6 & 7
8287 044634 012703 000010 MOV #8,R3 ;MOVE THE STARTING
8288 044640 012701 044374 MOV #QSTART,R1 ;ADDRESS OF THE QUEUE INTO
8289 044644 012122 1$: MOV (R1)+,(R2)+ ;THE QUEUE INPUT POINTER
8290 044646 005303 DEC R3
8291 044650 001375 BNE 1$
8292 044652 012703 000010 MOV #8,R3 ;MOVE THE STARTING ADDRESS
8293 044656 012701 044374 MOV #QSTART,R1 ;OF THE QUEUE INTO THE
8294 044662 012122 2$: MOV (R1)+,(R2)+ ;QUEUE OUTPUT POINTER
8295 044664 005303 DEC R3
8296 044666 001375 BNE 2$
8297 044670 104413 RESREG ;RESTORE R0 - R5
8298 044672 000207 RTS PC
8299
8300 ;EMPTY THE QUEUE SPECIFIED BY R1
8301 ;CALL
8302

```



```

8303      :      MOV      DRVNUM,R1      ;DRIVE NUMBER TO R1
8304      :      JSR      PC,EMPTYQ
8305
8306 044674 105061 044324  EMPTYQ: CLR  QCNT(R1)      ;CLEAR NUMBER OF ITEMS IN QUEUE
8307 044700 006301          ASL  R1
8308 044702 016161 044334 044354  MOV  QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
8309 044710 006201          ASR  R1
8310 044712 000207          RTS  PC
8311
8312      ;ROUTINE TO PUT A REQUEST IN QUEUE
8313      :CALL
8314      :
8315      :      MOV      #DRVNUM,R1      ;DRIVE NUMBER
8316      :      MOV      #DPB,R2      ;ADDRESS OF PARAMETER BLOCK
8317      :      JSR      RO,DRVQUE      ;GO PUT REQUEST IN QUEUE
8318      :      RETURN1      ;RETURN HERE IF QUEUE IS FULL
8319      :      RETURN2      ;RETURN HERE IF REQUEST IS IN QUEUE
8320
8321 044714 122761 000010 044324  DRVQUE: CMPB #10,QCNT(R1)      ;IS QUEUE FULL?
8322 044722 001421          BEQ  2$      ;BR IF YES-TAKE RETURN1
8323 044724 105261 044324          INCB QCNT(R1)      ;INCREMENT QUEUE COUNT
8324          ASL  R1
8325 044732 010271 044334          MOV  R2,QINPT(R1)      ;PUT THIS REQUEST IN QUEUE
8326 044736 062761 000002 044334  ADD  #2,QINPT(R1)      ;UPDATE THE QUEUE POINTER
8327 044744 026161 044334 044376  CMP  QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
8328 044752 001003          BNE  1$      ;BRANCH IF NO
8329 044754 016161 044374 044334  MOV  QSTART(R1),QINPT(R1) ;YES--RESET POINTER
8330          ASR  R1
8331 044764 005720          TST  (R0)+      ;TAKE RETURN 2
8332 044766 000200          RTS  RO      ;RETURN TO USER
8333
8334      ;ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
8335      :CALL
8336      :
8337      :      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
8338      :      JSR      PC,GETREQ      ;GO GET THE REQUEST
8339      :      RETURN      ;R2="DPB" ADDRESS OF THE REQUEST
8340      :      ;R2=0 IF NO REQUEST IN QUEUE
8341
8342 044770 005002          GETREQ: CLR  R2
8343 044772 105761 044324          TSTB QCNT(R1)      ;IS THERE ANY REQUEST IN QUEUE?
8344 044776 001404          BEQ  2$      ;NO---BRANCH
8345 045000 006301          ASL  R1
8346 045002 017102 044354          MOV  QOUTPT(R1),R2      ;PICKUP "DPB" POINTER FOR THIS DRIVE
8347 045006 006201          ASR  R1
8348 045010 000207          RTS  PC      ;RETURN TO USER
8349
8350      ;ROUTINE TO "POP" THE REQUEST FROM QUEUE
8351      :CALL
8352      :
8353      :      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
8354      :      JSR      PC,POPQUE      ;CALL TO REMOVE REQUEST
8355      :      RETURN      ;R2=ADDRESS OF DPB REMOVED
8356
8357 045012 105361 044324  POPQUE: DECB QCNT(R1)      ;DECREMENT QUEUE COUNT
8358 045016 006301          ASL  R1

```


8359 045020 017102 044354
8360 045024 005071 044354
8361 045030 062761 000002 044354
8362 045036 026161 044354 044376
8363 045044 001003
8364 045046 016161 044374 044354
8365 045054 006201
8366 045056 000207
8367
8368
8369
8370
8371
8372
8373
8374
8375
8376 045060 012037 045074
8377 045064 013746 001206
8378 045070 005037 001206
8379 045074 000000
8380 045076 012637 001206
8381 045102 000200
8382
8383
8384
8385
8386 045104 000122
8387 045106 041506 000
8388 045111 114 000103
8389 045114 041506 000047
8390 045120 041514 000047
8391 045124 041511 000
8392 045127 106 000124
8393 045132 052114 000
8394 045135 111 000124
8395 045140 051506 000
8396 045143 114 000123
8397 045146 040520 000124
8398 045152 000075
8399 045154 005015 047503 052116
8400 045162 047522 020114 053523
8401 045170 052111 044103 051505
8402 045176 000075
8403
8404 045200 027440 000040
8405 045204 047125 052111 051440
8406 045212 040524 052524 035123
8407 045220 005015 000012
8408 045224 051104 053111 000105
8409 045232 047440 043106 044514
8410 045240 042516 000
8411 045243 040 047117 044514
8412 045250 042516 000
8413 045253 040 047516 020124
8414 045260 051120 051505 047105

```

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),QSTOP(R1) ;TIME TO RESET THE POINTER?
BNE 1$ ;NO--BRANCH TO EXIT
MOV QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
1$: ASR R1
RTS PC ;RETURN TO USER

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

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

.SBTTL ASCIZ MESSAGES
MSG.R: .ASCIZ /R/
MSG.FC: .ASCIZ /FC/
MSG.LC: .ASCIZ /LC/
MSG.FCP: .ASCIZ /FC'/
MSG.LCP: .ASCIZ /LC'/
MSG.IC: .ASCIZ /IC/
MSG.FT: .ASCIZ /FT/
MSG.LT: .ASCIZ /LT/
MSG.IT: .ASCIZ /IT/
MSG.FS: .ASCIZ /FS/
MSG.LS: .ASCIZ /LS/
MSG.PAT: .ASCIZ /PAT/
MSG.EQ: .ASCIZ /=/
MSG.CS: .ASCIZ <CR><LF>/CONTROL SWITCHES=/

SLASH: .ASCIZ @ / @
SYSTAT: .ASCIZ /UNIT STATUS: /<CR><LF><LF>

UNTMSG: .ASCIZ /DRIVE/
UNTOFF: .ASCIZ / OFFLINE/
UNTON: .ASCIZ / ONLINE/
NOTPRS: .ASCIZ / NOT PRESENT/

```


8415	045266	000124				
8416	045270	052440	051516	043101	NOTSAF: .ASCIZ	/ UNSAFE/
8417	045276	000105				
8418	045300	047040	052117	051040	NOTRM: .ASCIZ	2 NOT RMO3/RMO22
8419	045306	030115	027463	046522		
8420	045314	031060	000			
8421	045317	122	030115	000062	ISRMO2: .ASCIZ	/RMO2/
8422	045324	046522	031460	000	ISRMO3: .ASCIZ	/RMO3/
8423	045331	015	042012	044522	DRIVES: .ASCIZ	<CR><LF>/DRIVE(S) TO BE TESTED /
8424	045336	042526	051450	020051		
8425	045344	047524	041040	020105		
8426	045352	042524	052123	042105		
8427	045360	000040				
8428	045362	047516	042516	000	NONE: .ASCIZ	/NONE/
8429	045367	054	000		COMMA: .ASCIZ	/ /
8430	045371	015	047012	020117	NOCLOK: .ASCIZ	<CR><LF>/NO KW11-P CLOCK, TIMING TESTS WILL NOT BE PERFORMED/
8431	045376	053513	030461	050055		
8432	045404	041440	047514	045503		
8433	045412	020054	044524	044515		
8434	045420	043516	052040	051505		
8435	045426	051524	053440	046111		
8436	045434	020114	047516	020124		
8437	045442	042502	050040	051105		
8438	045450	047506	046522	042105		
8439	045456	000				
8440	045457	015	005012	042524	TESTNG: .ASCIZ	<CR><LF><LF>/TESTING DRIVE /
8441	045464	052123	047111	020107		
8442	045472	051104	053111	020105		
8443	045500	000				
8444	045501	123	051105	040511	SERIAL: .ASCIZ	/SERIAL NUMBER /
8445	045506	020114	052516	041115		
8446	045514	051105	000040			
8447						
8448	045520	005015	051012	052117	MSG7: .ASCIZ	<CR><LF><LF>/ROTATIONAL SPEED TIMES/
8449	045526	052101	047511	040516		
8450	045534	020114	050123	042505		
8451	045542	020104	044524	042515		
8452	045550	000123				
8453	045552	005015	047412	042516	MSG10A: .ASCIZ	<CR><LF><LF>/ONE CYLINDER SEEK TIMES/<CR><LF>/ * FORWARD/
8454	045560	041440	046131	047111		
8455	045566	042504	020122	042523		
8456	045574	045505	052040	046511		
8457	045602	051505	005015	025040		
8458	045610	043040	051117	040527		
8459	045616	042122	000			
8460	045621	015	020012	020052	MSG10B: .ASCIZ	<CR><LF>/ * REVERSE/
8461	045626	042522	042526	051522		
8462	045634	000105				
8463	045636	005015	040412	041503	MSG11A: .ASCIZ	<CR><LF><LF>/ACCESS TIME MEASUREMENT/<CR><LF>/ * FORWARD/
8464	045644	051505	020123	044524		
8465	045652	042515	046440	040505		
8466	045660	052523	042522	042515		
8467	045666	052116	005015	025040		
8468	045674	043040	051117	040527		
8469	045702	042122	000			
8470	045705	015	020012	020052	MSG11B: .ASCIZ	<CR><LF>/ * REVERSE/

8471	045712	042522	042526	051522	
8472	045720	000105			
8473	045722	005015	046412	054101	MSG12A: .ASCIZ <CR><LF><LF>/MAXIMUM SEEK TIMES/<CR><LF>/ * FORWARD/
8474	045730	046511	046525	051440	
8475	045736	042505	020113	044524	
8476	045744	042515	006523	020012	
8477	045752	020052	047506	053522	
8478	045760	051101	000104		
8479	045764	005015	025040	051040	MSG12B: .ASCIZ <CR><LF>/ * REVERSE/
8480	045772	053105	051105	042523	
8481	046000	000			
8482					
8483	046001	015	046412	047111	MSGMIN: .ASCIZ <CR><LF>/MIN=/ /
8484	046006	000075			
8485	046010	005015	040515	036530	MSGMAX: .ASCIZ <CR><LF>/MAX=/ /
8486	046016	000			
8487	046017	015	040412	043526	MSGAVG: .ASCIZ <CR><LF>/AVG=/ /
8488	046024	000075			
8489	046026	020060	051525	000	MSGOUS: .ASCIZ /O US/ /
8490	046033	040	042502	047514	MBELOW: .ASCIZ / BELOW THE MINIMUM OF / /
8491	046040	020127	044124	020105	
8492	046046	044515	044516	052515	
8493	046054	020115	043117	000040	
8494	046062	040440	047502	042526	MABOVE: .ASCIZ / ABOVE THE MAXIMUM OF / /
8495	046070	052040	042510	046440	
8496	046076	054101	046511	046525	
8497	046104	047440	020106	000	
8498	046111	040	042523	045505	MSGNUM: .ASCIZ / SEEKS TIMED/ /
8499	046116	020123	044524	042515	
8500	046124	000104			
8501	046126	047040	052117	052040	MSGNON: .ASCIZ / NOT TIMED/ /
8502	046134	046511	042105	000	
8503	046141	040	000040		MSG.SP: .ASCIZ / / ; TWO (2) SPACES
8504					
8505					.SBTTL ERROR HEADER (EM) MESSAGES
8506					
8507	046144	044122	030067	044440	EM1: .ASCIZ /RH70 INTERRUPT OCCURRED (RMAS = 0)/
8508	046152	052116	051105	052522	
8509	046160	052120	047440	041503	
8510	046166	051125	042522	020104	
8511	046174	051050	040515	020123	
8512	046202	020075	024460	000	
8513	046207	125	042516	050130	EM2: .ASCIZ /UNEXPECTED ATTENTION OCCURRED/ /
8514	046214	041505	042524	020104	
8515	046222	052101	042524	052116	
8516	046230	047511	020116	041517	
8517	046236	052503	051122	042105	
8518	046244	000			
8519	046245	115	051501	041123	EM3: .ASCIZ /MASSBUS PARITY ERROR(MCPE=1)/ /
8520	046252	051525	050040	051101	
8521	046260	052111	020131	051105	
8522	046266	047522	024122	041515	
8523	046274	042520	030475	000051	
8524	046302	040515	051523	052502	EM4: .ASCIZ /MASSBUS PARITY ERROR(PAR=1)/ /
8525	046310	020123	040520	044522	
8526	046316	054524	042440	051122	

8527	046324	051117	050050	051101	
8528	046332	030475	000051		
8529	046336	042101	051104	051505	EMS: .ASCIZ /ADDRESS PLUG CHANGE BIT SET/
8530	046344	020123	046120	043525	
8531	046352	041440	040510	043516	
8532	046360	020105	044502	020124	
8533	046366	042523	000124		
8534	046372	044122	030067	051057	EM10: .ASCIZ "RH70/RMO3 FAILED TO RESPOND TO ADDRESSING"
8535	046400	030115	020063	040506	
8536	046406	046111	042105	052040	
8537	046414	020117	042522	050123	
8538	046422	047117	020104	047524	
8539	046430	040440	042104	042522	
8540	046436	051523	047111	000107	
8541	046444	051104	053111	020105	EM11: .ASCIZ /DRIVE SELECTED IS NOT ONLINE/
8542	046452	042523	042514	052103	
8543	046460	042105	044440	020123	
8544	046466	047516	020124	047117	
8545	046474	044514	042516	000	
8546	046501	111	050115	047522	EM12: .ASCIZ /IMPROPER HEADER DATA/
8547	046506	042520	020122	042510	
8548	046514	042101	051105	042040	
8549	046522	052101	000101		
8550	046526	040504	040524	041440	EM13: .ASCIZ /DATA COMPARE FAILURE/
8551	046534	046517	040520	042522	
8552	046542	043040	044501	052514	
8553	046550	042522	000		
8554	046553	104	051511	020113	EM17: .ASCIZ /DISK ERROR IN TIMING TEST/
8555	046560	051105	047522	020122	
8556	046566	047111	052040	046511	
8557	046574	047111	020107	042524	
8558	046602	052123	000		
8559	046605	103	047514	045503	EM20: .ASCIZ /CLOCK (KW11-P) OVERFLOW IN TIMING TEST/
8560	046612	024040	053513	030461	
8561	046620	050055	020051	053117	
8562	046626	051105	046106	053517	
8563	046634	044440	020116	044524	
8564	046642	044515	043516	052040	
8565	046650	051505	000124		
8566	046654	044504	045523	042440	EM23: .ASCIZ /DISK ERROR DURING SEEK/
8567	046662	051122	051117	042040	
8568	046670	051125	047111	020107	
8569	046676	042523	045505	000	
8570	046703	123	042505	020113	EM24: .ASCIZ /SEEK NOT COMPLETE WITHIN 120 MS/
8571	046710	047516	020124	047503	
8572	046716	050115	042514	042524	
8573	046724	053440	052111	044510	
8574	046732	020116	031061	020060	
8575	046740	051515	000		
8576	046743	122	033510	027460	EM41: .ASCIZ "RH70/RMO3 ERROR"
8577	046750	046522	031460	042440	
8578	046756	051122	051117	000	
8579	046763	106	052101	046101	EM46: .ASCIZ /FATAL WRITE CHECK ERROR/
8580	046770	053440	044522	042524	
8581	046776	041440	042510	045503	
8582	047004	042440	051122	051117	

8583	047012	000			
8584					
8585					.SBTTL STATUS/ERROR INDICATOR MESSAGES
8586					
8587	047013	117	043106	044514	MSGB14: .ASCIZ /OFFLINE OR UNSAFE DRIVE REQUESTED/
8588	047020	042516	047440	020122	
8589	047026	047125	040523	042506	
8590	047034	042040	044522	042526	
8591	047042	051040	050505	042525	
8592	047050	052123	042105	000	
8593	047055	125	046116	040517	MSGB13: .ASCIZ /UNLOADED DRIVE REQUESTED/
8594	047062	042504	020104	051104	
8595	047070	053111	020105	042522	
8596	047076	052521	051505	042524	
8597	047104	000104			
8598	047106	042520	051522	051511	MSGB12: .ASCIZ /PERSISTENT UNSAFE/
8599	047114	042524	052116	052440	
8600	047122	051516	043101	000105	
8601	047130	040520	044522	054524	MSGB11: .ASCIZ /PARITY ERROR OCCURRED/
8602	047136	042440	051122	051117	
8603	047144	047440	041503	051125	
8604	047152	042522	000104		
8605	047156	040506	040524	020114	MSGB10: .ASCIZ /FATAL PARITY ERROR/
8606	047164	040520	044522	054524	
8607	047172	042440	051122	051117	
8608	047200	000			
8609	047201	123	043117	053524	MSGB09: .ASCIZ /SOFTWARE TIMEOUT ON THIS DRIVE/
8610	047206	051101	020105	044524	
8611	047214	042515	052517	020124	
8612	047222	047117	052040	044510	
8613	047230	020123	051104	053111	
8614	047236	000105			
8615	047240	047523	052106	040527	MSGB08: .ASCIZ /SOFTWARE TIMEOUT ON ANOTHER DRIVE/
8616	047246	042522	052040	046511	
8617	047254	047505	052125	047440	
8618	047262	020116	047101	052117	
8619	047270	042510	020122	051104	
8620	047276	053111	000105		
8621	047302	051105	047522	020122	MSGB06: .ASCIZ "ERROR OCCURRED DURING I/O OPERATION"
8622	047310	041517	052503	051122	
8623	047316	042105	042040	051125	
8624	047324	047111	020107	027511	
8625	047332	020117	050117	051105	
8626	047340	052101	047511	000116	
8627	047346	051105	047522	020122	MSGB05: .ASCIZ "ERROR OCCURRED DURING NON-I/O OPERATION"
8628	047354	041517	052503	051122	
8629	047362	042105	042040	051125	
8630	047370	047111	020107	047516	
8631	047376	026516	027511	020117	
8632	047404	050117	051105	052101	
8633	047412	047511	000116		
8634	047416	047125	040523	042506	MSGB04: .ASCIZ /UNSAFE OCCURRED/
8635	047424	047440	041503	051125	
8636	047432	042522	000104		
8637	047436	052501	047524	040515	MSGB03: .ASCIZ /AUTOMATIC RECALIBRATE SEQUENCE OCCURRED/
8638	047444	044524	020103	042522	

8639	047452	040503	044514	051102
8640	047460	052101	020105	042523
8641	047466	052521	047105	042503
8642	047474	047440	041503	051125
8643	047502	042522	000104	
8644	047506	051104	053111	020105
8645	047514	040510	020123	047516
8646	047522	020124	042522	050123
8647	047530	047117	042504	020104
8648	047536	047524	050040	051117
8649	047544	020124	042522	052521
8650	047552	051505	000124	
8651	047556	051104	053111	020105
8652	047564	040510	020123	042502
8653	047572	047503	042515	047040
8654	047600	047117	042455	044530
8655	047606	052123	047105	000124
8656				
8657				
8658				
8659	047614	051105	020122	041520
8660	047622	020040	046522	051501
8661	047630	000		
8662	047631	105	051122	050040
8663	047636	020103	042040	044522
8664	047644	042526	020040	051040
8665	047652	040515	020123	020040
8666	047660	051040	042115	020123
8667	047666	020040	046522	051105
8668	047674	020061	020040	046522
8669	047702	051115	020062	020040
8670	047710	046522	051105	000062
8671	047716	042524	052123	020040
8672	047724	020040	051105	020122
8673	047732	041520	020040	042101
8674	047740	051104	051505	020123
8675	047746	040504	040524	000
8676	047753	124	051505	020124
8677	047760	020040	042440	051122
8678	047766	050040	020103	040440
8679	047774	042104	042522	051523
8680	050002	043440	042104	052101
8681	050010	020101	041040	042104
8682	050016	052101	000101	
8683	050022	046522	051503	020061
8684	050030	020040	051105	020122
8685	050036	041520	000	
8686	050041	104	044522	042526
8687	050046	020040	042440	051122
8688	050054	050040	000103	
8689	050060	042524	052123	020040
8690	050066	020040	051105	020122
8691	050074	041520	020040	051524
8692	050102	020124	041520	020040
8693	050110	051104	053111	020105
8694	050116	020040	054503	047114

MSGBO2: .ASCIZ /DRIVE HAS NOT RESPONDED TO PORT REQUEST/

MSGBO1: .ASCIZ /DRIVE HAS BECOME NON-EXISTENT/

.SBTTL DATA HEADER (DT) MESSAGES

DH1: .ASCIZ /ERR PC RMAS/

DH2: .ASCIZ /ERR PC DRIVE RMAS RMD5 RMER1 RMMR2 RMER2/

DH3: .ASCIZ /TEST ERR PC ADDRESS DATA/

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

DH10: .ASCIZ /RMCS1 ERR PC/

DH11: .ASCIZ /DRIVE ERR PC/

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

8751	050614	042515	031122	020040
8752	050622	051040	042115	020103
8753	050630	020040	051040	044115
8754	050636	000122		
8755	050640	042524	052123	020040
8756	050646	020040	051105	020122
8757	050654	041520	020040	051524
8758	050662	020124	041520	020040
8759	050670	051104	053111	000105
8760	050676	042524	052123	020040
8761	050704	020040	051105	020122
8762	050712	041520	020040	051524
8763	050720	020124	041520	020040
8764	050726	051104	053111	020105
8765	050734	020040	046522	051503
8766	050742	020061	020040	046522
8767	050750	051503	020062	020040
8768	050756	046522	051504	000
8769	050763	122	042515	030522
8770	050770	020040	051040	046515
8771	050776	031122	020040	051040
8772	051004	042515	031122	000
8773	051011	122	041515	030523
8774	051016	020040	051040	041515
8775	051024	031123	020040	051040
8776	051032	042115	020123	020040
8777	051040	046522	051110	020040
8778	051046	020040	046522	041504
8779	051054	020040	020040	046522
8780	051062	040504	000	
8781	051065	122	042515	030522
8782	051072	020040	051040	046515
8783	051100	031122	020040	051040
8784	051106	042515	031122	000
8785	051113	122	042515	030522
8786	051120	020040	051040	046515
8787	051126	031122	020040	051040
8788	051134	042515	031122	020040
8789	051142	051040	053515	020103
8790	051150	020040	051040	041115
8791	051156	020101	020040	051040
8792	051164	042115	000102	

DH41: .ASCIZ /TEST ERR PC TST PC DRIVE/

DH42: .ASCIZ /TEST ERR PC TST PC DRIVE RMCS1 RMCS2 RMDS/

DH43A: .ASCIZ /RMER1 RMMR2 RMER2/

DH44A: .ASCIZ /RMCS1 RMCS2 RMDS RMHR RMDC RMDA/

DH44B: .ASCIZ /RMER1 RMMR2 RMER2/

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

.EVEN

.SBTTL DATA TABLE (DT)

8799	051170	001116	001170	
8800	051174	001116	001164	001170
8801	051202	036570	036572	036574
8802	051210	036576		
8803	051212	001176	001116	043606
8804	051220	043610		
8805	051222	001176	001116	044032
8806	051230	044030	043610	

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

M15

8807	051234	001176	001116	001164	DT5:	.WORD	\$TMPO, \$ERRPC, \$REG1, \$REG5, RMERRS, RMERRS+2, RMERRS+4, RMERRS+6
8808	051242	001174	036570	036572			
8809	051250	036574	036576				
8810	051254	001450	001116		DT10:	.WORD	RH.ADR, \$ERRPC
8811	051260	001166	001116		DT11:	.WORD	\$REG2, \$ERRPC
8812	051264	001176	001116	001162	DT12:	.WORD	\$TMPO, \$ERRPC, \$REG0, CHKDRV, CYL.DS, TRK.DS, SEC.DS
8813	051272	001334	001350	001354			
8814	051300	001352					
8815	051302	001350	001354	001352	DT12A:	.WORD	CYL.DS, TRK.DS, SEC.DS, CYL.RD, TRK.RD, SEC.RD
8816	051310	001342	001344	001346			
8817	051316	001176	001116	001162	DT13:	.WORD	\$TMPO, \$ERRPC, \$REG0, CHKDRV, CYL.DS, TRK.DS, SEC.DS
8818	051324	001334	001350	001354			
8819	051332	001352					
8820	051334	001124	001126	001172	DT13A:	.WORD	\$GDDAT, \$BDDAT, \$REG4, \$GDADR, \$BDADR
8821	051342	001120	001122				
8822	051346	001176	001116	001334	DT17:	.WORD	\$TMPO, \$ERRPC, CHKDRV, RM.REG, RM.REG+12, RM.REG+14, RM.REG+40, RM.REG+42
8823	051354	004306	004320	004322			
8824	051362	004346	004350				
8825	051366	001176	001116	001162	DT21:	.WORD	\$TMPO, \$ERRPC, \$REG0, CHKDRV, CYL.DS, TRK.DS
8826	051374	001334	001350	001354			
8827	051402	001164	001126	001172	DT21A:	.WORD	\$REG1, \$BDDAT, \$REG4, \$REG1
8828	051410	001164					
8829	051412	001176	001116	001334	DT23:	.WORD	\$TMPO, \$ERRPC, CHKDRV, CYL.DS, RM.REG, RM.REG+10, RM.REG+12
8830	051420	001350	004306	004316			
8831	051426	004320					
8832	051430	004322	004346	004350	DT23A:	.WORD	RM.REG+14, RM.REG+40, RM.REG+42, RM.REG+34, RM.REG+36
8833	051436	004342	004344				
8834	051442	001176	001116	001162	DT41:	.WORD	\$TMPO, \$ERRPC, \$REG0, CHKDRV
8835	051450	001334					
8836	051452	001176	001116	001162	DT42:	.WORD	\$TMPO, \$ERRPC, \$REG0, CHKDRV, RM.REG, RM.REG+10, RM.REG+12
8837	051460	001334	004306	004316			
8838	051466	004320					
8839	051470	001176	001116	001162	DT43:	.WORD	\$TMPO, \$ERRPC, \$REG0, CHKDRV, RM.REG, RM.REG+10, RM.REG+12
8840	051476	001334	004306	004316			
8841	051504	004320					
8842	051506	004322	004346	004350	DT43A:	.WORD	RM.REG+14, RM.REG+40, RM.REG+42
8843	051514	001176	001116	001162	DT44:	.WORD	\$TMPO, \$ERRPC, \$REG0, CHKDRV, CYL.DS, TRK.DS, SEC.DS
8844	051522	001334	001350	001354			
8845	051530	001352					
8846	051532	004306	004316	004320	DT44A:	.WORD	RM.REG, RM.REG+10, RM.REG+12, RM.REG+36, RM.REG+34, RM.REG+06
8847	051540	004344	004342	004314			
8848	051546	004322	004346	004350	DT44B:	.WORD	RM.REG+14, RM.REG+40, RM.REG+42
8849	051554	001176	001116	001162	DT45:	.WORD	\$TMPO, \$ERRPC, \$REG0, CHKDRV, CYL.DS, TRK.DS, SEC.DS
8850	051562	001334	001350	001354			
8851	051570	001352					
8852	051572	004306	004316	004320	DT45A:	.WORD	RM.REG, RM.REG+10, RM.REG+12, RM.REG+36, RM.REG+34, RM.REG+06
8853	051600	004344	004342	004314			
8854	051606	004322	004346	004350	DT45B:	.WORD	RM.REG+14, RM.REG+40, RM.REG+42, RM.REG+2, RM.REG+4, RM.REG+22
8855	051614	004310	004312	004330			

.SBTTL DATA FORMAT (DF) TABLE

8859	051622	000001	DF1:	.WORD	1	: NUMBER OF DATA HEADERS
8860	051624	002		.BYTE	2	: NUMBER OF WORDS IN DATA TABLE
8861	051625	000		.BYTE	0	: ALL 3 NUMBERS ARE OCTAL
8862						

8863	051626	000001	DF2:	.WORD	1
8864	051630	007		.BYTE	7
8865	051631	000		.BYTE	0
8866					
8867	051632	000001	DF3:	.WORD	1
8868	051634	004		.BYTE	4
8869	051635	000		.BYTE	0
8870					
8871	051636	000001	DF4:	.WORD	1
8872	051640	005		.BYTE	5
8873	051641	000		.BYTE	0
8874					
8875	051642	000001	DF10:	.WORD	1
8876	051644	002		.BYTE	2
8877	051645	000		.BYTE	0
8878					
8879	051646	000001	DF11:	.WORD	1
8880	051650	002		.BYTE	2
8881	051651	000		.BYTE	0
8882					
8883	051652	000002	DF12:	.WORD	2
8884	051654	007		.BYTE	7
8885	051655	160		.BYTE	160
8886	051656	050147		.WORD	DH12A
8887	051660	006		.BYTE	6
8888	051661	000		.BYTE	0
8889					
8890	051662	000002	DF13:	.WORD	2
8891	051664	007		.BYTE	7
8892	051665	160		.BYTE	160
8893	051666	050226		.WORD	DH13A
8894	051670	005		.BYTE	5
8895	051671	004		.BYTE	4
8896					
8897	051672	000000	DF14:	.WORD	0
8898	051674	005		.BYTE	5
8899	051675	004		.BYTE	4
8900					
8901	051676	000001	DF17:	.WORD	1
8902	051700	010		.BYTE	10
8903	051701	000		.BYTE	0
8904					
8905	051702	000002	DF21:	.WORD	2
8906	051704	006		.BYTE	6
8907	051705	060		.BYTE	60
8908	051706	050447		.WORD	DH21A
8909	051710	004		.BYTE	4
8910	051711	014		.BYTE	14
8911					
8912	051712	000000	DF22:	.WORD	0
8913	051714	004		.BYTE	4
8914	051715	014		.BYTE	14
8915					
8916	051716	000002	DF23:	.WORD	2
8917	051720	007		.BYTE	7
8918	051721	010		.BYTE	10

: 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

;WORD 3 IS DECIMAL

;WORD 3 IS DECIMAL

;WORD 4 IS DECIMAL

8919	051722	050573			.WORD	DH23A
8920	051724	005			.BYTE	5
8921	051725	000			.BYTE	0
8922						
8923						
8924	051726	000001	DF41:		.WORD	1
8925	051730	004			.BYTE	4
8926	051731	000			.BYTE	0
8927						
8928	051732	000001	DF42:		.WORD	1
8929	051734	007			.BYTE	7
8930	051735	000			.BYTE	0
8931						
8932	051736	000002	DF43:		.WORD	2
8933	051740	007			.BYTE	7
8934	051741	000			.BYTE	0
8935	051742	050763			.WORD	DH43A
8936	051744	003			.BYTE	3
8937	051745	000			.BYTE	0
8938						
8939	051746	000003	DF44:		.WORD	3
8940	051750	007			.BYTE	7
8941	051751	160			.BYTE	160
8942	051752	051011			.WORD	DH44A
8943	051754	006			.BYTE	6
8944	051755	000			.BYTE	0
8945	051756	051065			.WORD	DH44B
8946	051760	003			.BYTE	3
8947	051761	000			.BYTE	0
8948						
8949	051762	000003	DF45:		.WORD	3
8950	051764	007			.BYTE	7
8951	051765	160			.BYTE	160
8952	051766	051011			.WORD	DH44A
8953	051770	006			.BYTE	6
8954	051771	000			.BYTE	0
8955	051772	051113			.WORD	DH45A
8956	051774	006			.BYTE	6
8957	051775	000			.BYTE	0
8958						
8959						
8960	051776		.EVEN			
8961			BUFFER=.			
8962	051776	005015	041412	051132	TITLE:	.ASCII <CR><LF><LF>/CZRMFB0/<CR><LF>
8963	052004	043115	030102	005015		
8964	052012	046522	031460	051057	.ASCIZ	ARM03/RMO2 EXTENDED DRIVE TEST0<CR><LF>
8965	052020	030115	020062	054105		
8966	052026	042524	042116	042105		
8967	052034	042040	044522	042526		
8968	052042	052040	051505	006524		
8969	052050	000012				
8970	052052	005015	047524	052040	LOADRV:	.ASCII <CR><LF>/TO TEST DRIVE 0 REPLACE THE 'XXDP' PACK ON DRIVE 0/<CR><LF>
8971	052060	051505	020124	051104		
8972	052066	053111	020105	020060		
8973	052074	042522	046120	041501		
8974	052102	020105	044124	020105		

8975	052110	054047	042130	023520
8976	052116	050040	041501	020113
8977	052124	047117	042040	044522
8978	052132	042526	030040	005015
8979	052140	044527	044124	040440
8980	052146	047516	044124	051105
8981	052154	050040	041501	026113
8982	052162	041440	042514	051101
8983	052170	046440	046505	051117
8984	052176	020131	047514	040503
8985	052204	044524	047117	032040
8986	052212	026060	040440	042116
8987	052220	051040	051505	040524
8988	052226	052122	005015	
8989	052232	044124	020105	051120
8990	052240	043517	040522	006515
8991	052246	000012		
8992	052250	005015	054523	052123
8993	052256	046505	044040	051501
8994	052264	030440	045466	046440
8995	052272	046505	051117	026131
8996	052300	023440	054130	050104
8997	052306	020047	047514	042101
8998	052314	051105	053440	046111
8999	052322	020114	042502	047440
9000	052330	042526	053522	044522
9001	052336	052124	047105	005015
9002	052344	000012		
9003				
9004				
9005				
9006				
9007				
9008				
9009				
9010				
9011				
9012				
9013				
9014	052346	010046		
9015	052350	010146		
9016	052352	013746	000004	
9017	052356	013746	000006	
9018	052362	010600		
9019				
9020	052364	104400		
9021	052366	012637	000006	
9022	052372	012737	052412	000004
9023	052400	012701	020000	
9024	052404	005711		
9025	052406	005721		
9026	052410	000775		
9027	052412	162701	000002	
9028	052416	010006		
9029	052420	012637	000006	
9030	052424	012637	000004	

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

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

NOLOAD: .ASCIZ <CR><LF>/SYSTEM HAS 16K MEMORY, 'XXDP' LOADER WILL BE OVERWRITTEN/<CR><LF>

.EVEN

.SBTTL ROUTINE TO SIZE MEMORY

*CALL:

* JSR PC,\$SIZE

* RETURN

*\$LSTAD WILL CONTAIN THE LAST AVAILABLE MEMORY LOCATION

```

$SIZE:  MOV    RO,-(SP)      ;;SAVE RO ON THE STACK
        MOV    R1,-(SP)      ;;SAVE R1 ON THE STACK
        MOV    @#ERRVEC-(SP) ;;SAVE PRESENT ERROR VECTOR PS & PC
        MOV    @#ERRVEC+2,-(SP)
        MOV    SP,RO        ;;SAVE THE STACK POINTER
        ;;SET THE ERRVEC PS TO THE PRESENT PS
        TRAP
        MOV    (SP)+,@#ERRVEC+2 ;;PUSH OLD PSW AND PC ON STACK
        MOV    @2$,@#ERRVEC    ;;SAVE THE PSW IN @#ERRVEC+2
        MOV    #20000,R1      ;;SET FOR TIMEOUT
        TST    (R1)           ;;FIRST ADDRESS
        BR    1$              ;;TEST THIS ADDRESS
        TST    (R1)+          ;;STEP TO NEXT ADDRESS
        BR    1$              ;;TRY ANOTHER
        SUB    #2,R1          ;;DROP BACK
        MOV    RO,SP          ;;RESTORE THE STACK
        MOV    (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
        MOV    (SP)+,@#ERRVEC

```


9031 052430 010137 052442
9032 052434 012601
9033 052436 012600
9034 052440 000207
9035 052442 000000

MOV R1,\$LSTAD ;:LAST ADDRESS
MOV (SP)+,R1 ;:RESTORE R1
MOV (SP)+,R0 ;:RESTORE R0
RTS PC
\$LSTAD: .WORD 0 ;:CONTAINS THE LAST ADDRESS

9036
9037
9038
9039
9040
9041
9042
9043
9044
9045
9046
9047
9048

:SBTTL GETADR - GET BUS ADDRESS AND VECTOR ADDRESS
:THIS ROUTINE IS USED TO ENSURE THE BUS ADDRESS
:OF THE RH70/RMO3 IS SETUP TO READ THE PROPER VALUE.
:IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
:REQUIRED.
:NOTE: THIS ROUTINE DESTROYS R0-R4
:CALL

9049 052444 005737 001306
9050 052450 001427
9051 052452 005037 001306
9052 052456 012700 001450
9053 052462 012703 052622
9054 052466 011004
9055 052470 004037 033740
9056 052474 000402
9057 052476 000767
9058 052500 000412
9059 052502 010420
9060 052504 012703 052712
9061 052510 011004
9062 052512 004037 033740
9063 052516 000402
9064 052520 000756
9065 052522 000401
9066 052524 010420
9067 052526 010410
9068 052530 013701 000004
9069 052534 012737 052570 000004
9070 052542 005777 126702
9071 052546 010137 000004
9072 052552 012700 001450
9073 052556 012701 036736
9074 052562 012021
9075 052564 012021
9076 052566 000207
9077 052570 010137 000004
9078 052574 022626
9079 052576 104010
9080 052600 005737 000042
9081 052604 001724
9082 052606 005037 001312
9083 052612 005037 021226
9084 052616 000137 021052
9085
9086 052622 005015 047105 042524

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,R0 ;:FIRST ADDRESS
MOV @#RMCS1,R3 ;:"RMCS1=" ;:PRESENT RMCS1 ADDRESS
JSR R0,@#GETNUM ;:GET NEW RMCS1
BR 2\$;:COMMA
BR 1\$;:PERIOD
BR 5\$;:DOUBLE PERIOD
2\$: MOV R4,(R0)+ ;:SAVE NEW RMCS1
MOV @#RHVEC,R3 ;:"RHVEC=" ;:PRESENT RH70 VECTOR ADDRESS
JSR R0,@#GETNUM ;:GET NEW RHVEC
BR 3\$;:COMMA
BR 1\$;:PERIOD
BR 5\$;:DOUBLE PERIOD
3\$: MOV R4,(R0)+ ;:SAVE NEW RHVEC
5\$: MOV R4,(R0) ;:SAVE INPUT
7\$: MOV @#ERRVEC,R1 ;:SAVE THE ERROR VECTOR
MOV @#ERRVEC,R1 ;:SETUP FOR TRAP
TST @RH.ADR ;:CHECK FOR RH70/RMO3
MOV R1,@#ERRVEC ;:RESTORE ERROR VECTOR
MOV @RH.ADR,R0 ;:FIRST ADDRESS OF NEW PARAMETERS
MOV @#RMADR,R1 ;:FIRST ADDRESS OF WHERE TO PUT THEM
MOV (R0)+,(R1)+ ;:BUS ADDRESS
MOV (R0)+,(R1)+ ;:VECTOR ADDRESS
RTS PC ;:RETURN
8\$: MOV R1,@#ERRVEC ;:RESTORE ERROR VECTOR
CMP (SP)+,(SP)+ ;:CLEAN OFF THE STACK
ERROR 10 ;:REPORT THE ERROR
TST @#42 ;:IS THERE A MONITOR?
BEQ 1\$;:NO--GO ASK FOR ADDRESS
CLR @#DRVSEL ;:YES--NO DRIVES SELECTED
CLR @#SEOPCT ;:NO PASSES
JMP @#SEOP ;:GO TO END OF PROGRAM

9086 052622 005015 047105 042524 MRMCS1: .ASCII <CR><LF>/ENTER SIX OCTAL DIGITS,FOLLOWED BY . AND <CR>/

E16

MD-11-DZRMF-B RM03/RM02 EXTENDED DRIVE TEST MACY11 30(1046) 22-NOV-77 08:04 PAGE 169
CZRMFB.P11 21-NOV-77 14:10 GETADR - GET BUS ADDRESS AND VECTOR ADDRESS

SEQ 0199

9087	052630	020122	044523	020130
9088	052636	041517	040524	020114
9089	052644	044504	044507	051524
9090	052652	043054	046117	047514
9091	052660	042527	020104	054502
9092	052666	026040	040440	042116
9093	052674	036040	051103	076
9094	052701	015	051012	041515
9095	052706	030523	000075	
9096	052712	005015	044122	042526
9097	052720	036503	000	
9098		000001		

.ASCIZ <CR><LF>/RMCS1=/
MRHVEC: .ASCIZ <CR><LF>/RHVEC=/
.END

AT3 = 000010	335#														
AT4 = 000020	336#														
AT5 = 000040	337#														
AT6 = 000100	338#														
AT7 = 000200	339#														
AUNIT = 000000	533	540													
AUSWR = 000000	533	547													
AVECT1 = 000000	533	572													
AVECT2 = 000000	533	573													
A16 = 000400	207#														
A17 = 001000	208#														
BAI = 000010	225#														
BASFLG = 001446	630#	2182*	3865	3884	5454*	5491*	5524*								
BITS = 001506	647#	2106	2156	2211	2270	2326	2422	2481	2547	2652	2825	2907	3043		
	3155	3269	3399	3486	3631	3706	3792	3910	6481	6512	6529	6582	6756		
BIT0 = 000001	182#														
BIT00 = 000001	172#	182	647	663	1931	1953	2008	5189							
BIT01 = 000002	171#	181	648	664	5549	5577	7385	7640							
BIT02 = 000004	170#	180	649	665	5252	5549	5580								
BIT03 = 000010	169#	179	650	666	5586	7851	8152								
BIT04 = 000020	168#	178	651	667	5583	7886									
BIT05 = 000040	167#	177	652	668	2711	2727	2803	2984	2998	3013	3018	3102	3126		
	3131	3206	3224	3241	3246	3320	3338	3355	3360	5586	5701	7231	7661		
	7866														
BIT06 = 000100	166#	176	653	669	5589	7318	7376	7450	7759	8213					
BIT07 = 000200	165#	175	654	670	7318	7585	7740	7851	7866	7886	7919	8185			
BIT08 = 000400	164#	174	655	5577	7318										
BIT09 = 001000	163#	173	656	4101	4736	5592	8025								
BIT1 = 000002	181#														
BIT10 = 002000	162#	657	4078	5201	5549	5580	6775	7642							
BIT11 = 004000	161#	658	4743	5580	7278	7412	7606	7960							
BIT12 = 010000	160#	659	5549	5583	7273	7300	7318	7387	7420	7600	7623	7638	7652		
	7876	7878	8090	8147	8214										
BIT13 = 020000	159#	660	4085	5549	5577	7370	7822	8093							
BIT14 = 040000	158#	661	2722	2979	2993	3097	3201	3219	3315	3333	4718	5463	5549		
	5583	7382	7417	7809	7955	8055	8102	8106	8211						
BIT15 = 100000	157#	662	5316	5355	5402	5452	5489	5522	5598	5746	5747	7370	7382		
	7385	7387	7417	7420	7606	7640	7642	7759	7851	7866	7876	7886	7955		
	8025	8055	8062												
BIT2 = 000004	180#														
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 = 051776	1321	1345	1369	1393	2688	3433	3448	3461	3520	3530	3544	3690	3824		
	3851	3861	3872	3880	3891	5665*	5666	5668	5670	5671	5672	5745	5782		
	5925	5944	5945	5969	6058	6090	6133	6175	6192	6193	6210	8960*			
BUSADR 001306	583#	1782*	1784*	1787*	1789*	9049	9051*								
BYPASS 001332	592#	2006*	2136*	2189*	2240*	2299*	2355*	2451*	2510*	2577*	2695*	2858*	3425*		
	3512*	3691*	3819*	5531	6016	6162	6227								
CALL.A 027054	2012	2014	2139	2596	3924	3929	5295#	5298	5680						
CALL.B 027206	2140	2193	2245	2253	2303	2308	2365	2371	2387	2393	2457	2459	2519		

E01

RD.RM2	043730	8080*	8113*											
RD.RM3	043734	8091	8105	8115*										
RD.RM4	043740	8096	8117*											
RD.WRD	043610	8086*	8087	8803	8805									
READ =	000171	442*	3445	3752	3860	3879								
RE:OMD=	000173	443*	2575	5281	5282	5361	5408							
READ:IN=	000121	435*												
RECAL =	000107	430*	2013	2132	5679	5704								
RELEAS=	000113	432*												
RESREG=	104413	2995	3000	3015	3104	3128	3208	3226	3243	3322	3340	3357	4218	4850*
		4910	5748	5933	5956	6018	6079	6130	6421	6573	6731	6735	6754	6777
		7250	7388	7391	7454	7667	7727	7989	8198	8297				
RESTAR	006234	1989*	2000	3990										
RESVEC=	000010	186*												
REX =	010000	306*												
RHVEC	001452	634*												
RM.ADR	001450	633*	8810	9052	9070	9072								
RMADR	036736	2576	2693	5700	7158*	7230	7347	7471	7491	7513	7680	7721	8008	8082
		8088	8092	8108	8141	8145	8178	8208	9073					
RMAS =	000016	7183*	7309*	7770	7893*	7923*	7944*	8033						
RMBA =	000004	2758*	7178*											
RMCS1 =	000000	2715*	2756*	2767*	3197*	3215*	3311*	3329*	7176*	7272*	7278	7298	7411*	7412
		7487	7509	7551	7571	7582	7597	7610	7743	7773	7827	7860	7899	7960
		8136												
RMCS2 =	000010	2599*	2711*	2712*	2727*	2728*	2737*	2803*	2804*	2984*	2985*	2998*	2999*	3013*
		3014*	3018*	3019*	3102*	3103*	3126*	3127*	3131*	3132*	3206*	3207*	3224*	3225*
		3241*	3242*	3246*	3247*	3320*	3321*	3338*	3339*	3355*	3356*	3360*	3361*	5701*
		7180*	7231*	7271*	7273	7360*	7410*	7472*	7492*	7514*	7600	7623	7638	7652
		7661*	7681*	7741*	7841*	7930*	7943*	8009*	8089	8146	8179*	8185	8209*	8214
RMDA =	000006	2759*	2942*	5726*	5910	7179*	7479	7505	7520					
RMDB =	000022	7185*	8183											
RMDC =	000034	2713*	2939*	3092*	3196*	3214*	3310*	3328*	5727*	7190*	7483	7495	7527	
RMDS =	000012	2717	2722	2979	2993	3097	3201	3219	3315	3333	7181*	7305	7683	7844
		7869	7931	8011										
RMDT =	000026	7187*	7281											
RMEC1 =	000044	7194*												
RMEC2 =	000046	7195*	8193											
RMERRS	036570	6979*	7214	7809	7812	7822	7931*	7932*	7933*	7934*	8800	8807		
RMER1 =	000014	5314	5350	5352	5397	5399	5450	5487	5520	5596	7182*	7311	7854	7932
		8150												
RMER2 =	000042	5316	5355	5402	5452	5489	5522	5598	7193*	7933				
RMHR =	000036	7191*												
RMINIT	036754	1879	7210*											
RMLA =	000020	2600	2738	7184*	7697	7699								
RMHR1 =	000024	7186*												
RMHR2 =	000040	7192*	7934											
RMOF =	000032	7189*	7302	7533	7537	7557	7561							
RMR =	000004	277*												
RMSN =	000030	7188*												
RMVEC	036740	2709*	2729*	2973*	3021*	3090*	3134*	3194*	3249*	3308*	3363*	5699*	7159*	7227
		7229	7341											
RMWC =	000002	2757*	7177*											
RM.REG	004306	1328	1352	1376	1400	1416*	2023	5314	5316	5350	5352	5355	5397	5399
		5402	5450	5452	5463	5487	5489	5520	5522	5596	5598	5910	8822	8829
		8832	8836	8839	8842	8846	8848	8852	8854					
RM03	037502	5296	5332	5379	5430	5469	5495	5501	5706	7340*				

CKCHR	1778#	6348	6398	6625	6662	6702	6865	6878	6889	6940					
CKDIG	1778#	6406	6644												
CKNUM	1778#	6356	6677	6710											
COMMEN	197#	2044	2079	2875	3370										
COMND	1778#	2013	2132	2574	2679	3923	3928								
DO	1778#	2012	2014	2139	2140	2192	2193	2245	2253	2303	2308	2365	2371	2387	2393
	2457	2459	2519	2521	2523	2525	2527	2529	2596	2617	2860	2862	3924	3929	5680
DODTA	1778#	3437	3441	3446	3456	3463	3525	3536	3552	3558	3737	3743	3753	3832	3857
	3864	3876	3883	3894											
DRV. IN	1778#	3017	3130	3245	3359										
ENDCOM	197#	2072	2090	2888	3379										
ENDPAS	1778#	3976													
ERRCAL	6960#	7778	7830	7833	8097	8158									
ERREND	1778#	4112													
ERROR	91#	1998	2726	2802	2987	3001	3016	3105	3129	3209	3227	3244	3323	3341	3358
	5309	5310	5311	5312	5318	5345	5346	5347	5348	5357	5392	5393	5394	5395	5404
	5445	5446	5447	5448	5456	5482	5483	5484	5485	5493	5515	5516	5517	5518	5526
	5678	5719	5720	5721	5722	5723	6035	6040	6151	6157	6215	6216	7779	7831	7835
	8099	8159	9079												
ERRTYP	1778#	4069													
ER. NDX	1778#	5303	5339	5386	5439	5476	5509	5713							
ESCAPE	197#	2943	3083	3189	3303	5302	5338	5385	5437	5475	5508	5676	5712	6034	6038
	6150	6155	6203												
GETPRI	197#	5020	9020												
GETSWR	1#	197#	1859												
LOOP	1778#	2242	2250	2301	2305	2362	2384	2454	2514	2593	2597	2696	2734	3434	3439
	3442	3450	3457	3522	3532	3549	3555	3731	3740	3750	3830	3855	3862	3874	3881
	3892														
MORETA	453#	578													
MORE. S	1778#	2104	2154	2209	2268	2324	2420	2479	2545	2650	2823	2905	3047	3153	3267
	3397	3484	3629	3790	3908										
MSG	2095#	2097	2143#	2145	2196#	2198	2259#	2261	2314#	2316	2407#	2409	2468#	2470	2534#
	2536	2625#	2628	2810#	2812	2892#	2895	3036#	3038	3140#	3142	3254#	3256	3384#	3386
	3470#	3472	3567#	3569	3763#	3765	3899#	3901							
MULT	197#														
NEWTST	197#	2095	2143	2196	2259	2314	2407	2468	2534	2626	2810	2893	3036	3140	3254
	3384	3470	3567	3763	3899										
POP	197#	4042	4043	4432	4799	4988	5071	5216	5265						
PUSH	197#	4003	4005	4026	4391	4779	4969	5023	5171	5229					
REPORT	197#	1778#	3135	3250	3364										
RMO3. D	1#	6960													
SAV. RH	1778#	2981	2995	3010	3099	3123	3203	3221	3238	3317	3335	3352			
SCOPE	92#	2141	2194	2257	2312	2405	2466	2532	2623	2808	2870	3033	3137	3252	3366
	3468	3564	3760	3895	3932										
SETPRI	197#	4622													
SETTRA	4830#	4839	4840	4841	4842	4844	4846	4847	4848	4849	4850				
SETUP	197#	1792													
SET. TN	1778#	2105	2155	2210	2269	2325	2421	2480	2546	2651	2824	2906	3048	3154	3268
	3398	3485	3630	3791	3909										
SKIP	197#														
SLASH	197#														
SPACE	197#														
STARS	67	197#	201	243	247	424	455	457	464	477	529	532	1680	2095	2103
	2143	2153	2196	2208	2259	2267	2314	2323	2407	2419	2468	2478	2534	2544	2626
	2649	2810	2822	2893	2904	3036	3046	3140	3152	3254	3266	3384	3396	3470	3483
	3567	3628	3763	3789	3899	3907	3938	3998	4055	4116	4223	4303	4381	4449	4525

SWRSU	4540	4611	4635	4705	4763	4809	4854	4873	4936	4960	4997	6959	6971	9008	9037
TRMTRP	197#	1814#													
TSTTYP	4830#														
	12#	2117	2167	2222	2281	2337	2433	2492	2558	2663	2836	2917	3060	3166	3280
	3410	3497	3642	3803											
TYPBIN	197#														
TYPB2D	1778#	5816	5823	5827	5832	5839	5843	5855	5860	6619	6694				
TYPDEC	197#	4197													
TYPEND	1778#	3945													
TYPNAM	197#														
TYPNUM	197#														
TYPOCS	197#	1889	1965	2016	2124	2174	2229	2288	2344	2440	2499	2565	2670	2843	2924
	3067	3173	3287	3417	3504	3649	3810	3955	6588						
TYPOCT	197#	3963	4194	4553	6343	6657	6698								
TYPTXT	197#	3945	3951	3959	3976	4156	4173	6312	6392	6424	6584	6689			
\$\$CHRE	475#	514	515	516	517	518	519								
\$\$CMTM	475#	520	521	522											
\$\$ESCA	197#														
\$\$NEWT	197#	2095	2143	2196	2259	2314	2407	2468	2534	2626	2810	2893	3036	3140	3254
	3384	3470	3567	3763	3899										
\$\$SET	4830#	4839	4840	4841	4842	4844	4846	4847	4848	4849	4850				
\$\$SETH	1830#														
\$\$SKIP	197#														
.EQUAT	1#	87													
.HEADE	1#														
.SETUP	1#	1782													
.SWRHI	1#	36													
.SWRLO	46#	47	48	49	50	51	52	53							
\$.ACT1	1#	65													
\$.APT8	1#	530#													
\$.APTH	1#	453													
\$.APTY	1#	3996													
\$.SCATC	1#	55													
\$.SCMTA	1#	475													
\$.SDB2D	1#	4871													
\$.SDIV	1#	4995													
\$.SEOP	1#	3936													
\$.SERRO	1#	4053													
\$.SRAND	1#	4958													
\$.SRDDE	1#														
\$.SROOC	1#														
\$.SREAD	1#	4447													
\$.SSAVE	1#	4761													
\$.SSB2D	1#	4852													
\$.SSCOP	1#	4703													
\$.SSIZE	1#	9006													
\$.SSUPR	1#	4934													
\$.STRAP	1#	4807													
\$.STYPD	1#	4379													
\$.STYPE	1#	4221													
\$.STYPO	1#	4301													

. ABS. 052723 000

002

MD-11-DZRMF-B RMD3/RMD2 EXTENDED DRIVE TEST MACY11 30(1046) 22-NOV-77 08:04 PAGE 195
CZRMFB.P11 21-NOV-77 14:10 CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0223

ERRORS DETECTED: 0

RMD3:CZRMFB,RMD3:CZRMFB.LST/NL:MD:MC:CND:TOC/SOL/CRF=RMD3:RMDRV5.P11,RMD3:CZRMFB.P11
RUN-TIME: 33 27 2 SECONDS
RUN-TIME RATIO: 558/64=8.6
CORE USED: 47K (93 PAGES)