

# RP04/5/6

RP04/5/6 RD/WT  
CZRJAD0

AH 9182D MC

NOV 1979

COPYRIGHT 76 79

**digital**

FICHE 1 OF 2

MADE IN USA

# RP04/5/6

RP04/5/6 RD/WT  
CZRJAD0

AH 9182D MC

COPYRIGHT 76 79  
FICHE 2 OF 2

NOV 1979

**digital**  
MADE IN USA

.REM @

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

PRODUCT CODE: AC-9180D-MC  
PRODUCT NAME: CZRJAD0 RP04/5/6 MECHANICAL AND READ-WRITE TEST  
DATE: MAY, 1979  
MAINTAINER: DIAGNOSTIC ENGINEERING  
AUTHOR: C. HESS

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.  
DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1976,1979 DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

CONTENTS

- 1. ABSTRACT
- 2. REQUIREMENTS
  - 2.1 EQUIPMENT
  - 2.2 PRELIMINARY PROGRAMS
  - 2.3 MEDIA
  - 2.4 PROGRAMMABLE DRIVES
- 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 RH11 - RH70 ADDRESS SELECTION
    - 4.3.3 DRIVE AND PARAMETER SELECTION
- 5. OPERATING PROCEDURE
  - 5.1 OPERATIONAL SWITCH SETTINGS
  - 5.2 CONTROL SWITCH SETTINGS
- 6. ERRORS
  - 6.1 ERROR TYPES
  - 6.2 ERROR RECOVERY
- 7. RESTRICTIONS
- 8. MISCELLANEOUS
  - 8.1 EXECUTION TIME
  - 8.2 STACK POINTER
  - 8.3 TIMING TEST (TESTS 12 - 15) PRINTOUTS
  - 8.4 END OF TEST
- 9. PROGRAM DESCRIPTION
- 10. PROGRAM LISTING

1. ABSTRACT  
-----

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

2. REQUIREMENTS  
-----

2.1 EQUIPMENT

PDP-11 PROCESSOR  
16K MEMORY  
TELETYPE  
PROGRAM LOADING DEVICE  
KW11-L OR KW11-P (THE KW11-P IS REQUIRED FOR THE TIMING TESTS)  
RH11 OR RH70 WITH 1 - 8 RPO4/5/6 DISK DRIVES

2.2 PRELIMINARY PROGRAMS

RPO4/5/6 DISKLESS CONTROLLER TEST  
PART 1 (MAINDEC-11-DZRJG)  
PART 2 (MAINDEC-11-DZRJH)

RPO4/5/6 FUNCTIONAL CONTROLLER TEST  
PART 1 (MAINDEC-11-DZRJI)  
PART 2 (MAINDEC-11-DZRJJ)

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.

2.4 PROGRAMMABLE DRIVES (DUAL PORT ENABLED)

THIS REV INCORPORATES A SAFEGUARD TO PREVENT INADVERTENT CORRUPTION OF DISK PACKS IN PROGRAMMABLE DRIVES. THIS IS A POTENTIAL HAZARD IN RUNNING THIS PROGRAM IN A MULTIPROCESSOR SYSTEM. FOR THE STANDARD STARTING ADDRESS OF 200 THE PROGRAM HAS BEEN MODIFIED TO PREVENT INITIALIZING DRIVES FOUND TO BE PROGRAMMABLE. THIS MODIFICATION APPLIES ONLY TO THE FIELD ENVIRONMENT (XXDP CHAIN, STANDALONE) WHERE LOCATION 42 DOES NOT EQUAL LOCATION 46. FOR THE MANUFACTURING ENVIRONMENT (WHERE LOCATION 42 EQUALS LOCATION 46) PROGRAMMABLE DRIVES WILL NOT BE INHIBITED. IF THE OPERATOR DESIRES TO RUN THIS PROGRAM USING PROGRAMMABLE DRIVES IN A FIELD ENVIRONMENT USE STARTING ADDRESS 220, WHERE 220 IS

THE SAME AS 200 WITHOUT INHIBITING PROGRAMMABLE DRIVES.  
SEE SECTION 4.1 FOR A SUMMARY OF ALL STARTING ADDRESSES.

### 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 (INHIBIT PROGRAMMABLE DRIVES-SEE SECTION 2.4)  
204 SELECT OPERATING PARAMETERS (INHIBIT PROGRAMMABLE DRIVES)  
210 SELECT RH11-RH70 ADDRESSES (DO NOT INHIBIT PROGRAMMABLE DRIVES)  
214 COMBINATION OF 204 AND 210 (DO NOT INHIBIT PROGRAMMABLE DRIVES)  
220 SAME AS 200 BUT WITH NO INHIBITIONS  
224 SAME AS 204 BUT WITH NO INHIBITIONS

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

<^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 RH11 - RH70 ADDRESS SELECTION

STARTING THE PROGRAM AT 200 WILL RESULT IN AUTOMATIC SELECT OF THE DEFAULT VALUES OF BUS ADDRESS (RPCS1), VECTOR ADDRESS, AND PRIORITY LEVEL OF THE RH11-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 RH11 OR RH70 AND THE VECTOR ADDRESS.

THE PROGRAM ALLOWS THE ADDRESSES TO BE CHANGED ON WHEN THE PROGRAM IS FIRST STARTED. STARTING ADDRESSES 210(8) AND 214(8) ARE TREATED AS ADDRESSES 200(8) OR 204(8) RESPECTIVELY.

ADDRESS SELECTION EXAMPLES

EXAMPLE #1

RPCS1=176700 / 177200.

EXAMPLE #2

RPCS1=176700 / 176300<CR>  
RHVEC=254 / 260<CR>  
RMPRIO=5 / 6.

EXAMPLE #3

RPCS1=176700<CR>  
RHVEC=254 / 260.

EXAMPLE #4

RH11/RP04 FAILED TO RESPOND TO ADDRESSING  
RPCS1 ERR PC  
176300 XXXXXX  
RPCS1=176300 / 176700.



EXAMPLE #5

RPCS1=176700 / 1776\67\6300<CR>  
RHVEC=254<CR>  
RMPRIO=5<CR>  
RPCS1=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.

EACH TEST CONTAINS TWO SETS OF CYLINDER LIMIT PARAMETERS. PARAMETERS 'LC' AND 'FC' ARE USED BY RPO4/5 DRIVES AND PARAMETERS 'LC'' AND 'FC'' ARE USED BY RPO6 DRIVES. THE PROGRAM DETERMINES WHICH DRIVE IS BEING TESTED AND SELECTS THE CORRECT SET OF CYLINDER LIMIT VALUES. IF THE PROGRAM IS BEING USED TO TEST A SUBSYSTEM WHICH CONTAINS BOTH RPO4/5 AND RPO6 DRIVES, THE OPERATOR MUST CHANGE BOTH SETS OF CYLINDER LIMITS IF THE TESTS ARE TO BE MODIFIED FOR ALL DRIVES TESTED.

4.3.3.1 DRIVE AND PARAMETER SELECTION DESCRIPTION

THE FOLLOWING IS A TABLE OF TERMS USED BY THE PS1.

'R''	REPEATS (ITERATIONS)
'FC''	FIRST CYLINDER ADDRESS FOR RPO4/5'S
'LC''	LAST CYLINDER ADDRESS FOR RPO4/5'S
'FC'''	FIRST CYLINDER ADDRESS FOR RPO6'S
'LC'''	LAST CYLINDER ADDRESS FOR RPO6'S
'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)

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>

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=410 /
```

THE PROGRAM RESPONDS WITH THE PREVIOUSLY ASSIGNED PARAMETER FOR LAST CYLINDER ADDRESS IN THIS CASE USING 410 AS THE EXAMPLE. THIS IS WHAT THE 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= 410 / 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= 410 / 20.<CR>
TEST 10
R=1 / 10.<CR>
```

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

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

DRIVE(S)=

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

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

#### 4.3.3.2 DRIVE AND PARAMETER SELECTION EXAMPLES

##### EXAMPLE #1

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

##### EXAMPLE #2

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

##### EXAMPLE #3

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

TEST 7
  R=1 / 50<CR>      ;50 ITERATIONS, MOVE TO NEXT PARAMETER
  FC=0 / <CR>       ;DO NOT CHANGE 'FC' CYLINDER ADDRESS BUT CONTINUE
  LC=410 / 50.<CR>  ;TEST 10 IS STILL PENDING AND WILL BE
```

;RETAIN ITS PRESENT PARAMETERS.

EXAMPLE #4

-----

DRIVE=0<CR>  
TEST=S,E.<CR>

;SELECT DRIVE #0 AND MAKE CHANGES  
;RUN ALL SEEK TESTS AND THE EXERCISER

EXAMPLE #5

-----

DRIVE=1<CR>  
TEST=S/D<CR>  
TEST 0  
R=10 / <CR>  
FC=0 / 10..<CR>

;RUN ALL SEEK TESTS (OPEN FOR CHANGES) AND  
;THE DATA TEST (WITH DEFAULT PARAMETERS).  
;RUN WITH 10 ITERATIONS  
;CHANGE FIRST CYLINDER ADDRESS  
;AND START EXECUTION  
;TESTS 1 - 10 WILL RETAIN THEIR PREVIOUSLY  
;ASSIGNED PARAMETERS.

EXAMPLE #6

-----

DRIVE=1<CR>  
TEST=S/<CR>  
TEST 0  
R=10 / 100.<CR>  
TEST 1  
R=100 / 1000.<CR>  
TEST 2  
R=1 / 10<CR>  
FC=0 / 50<CR>  
LC=410 / 51.<CR>  
TEST 3  
R=1.<CR>  
TEST 4  
R=1..<CR>

;OPEN THE SEEK TESTS (TESTS 0-10)  
;CHANGE TO 100 ITERATIONS, TO TO THE NEXT TEST  
;CHANGE 'R' TO 1000 ITERATIONS, MOVE TO NEXT TEST  
;CHANGE 'R' TO 10 ITERATIONS, GO TO NEXT PARAMETER  
;CHANGE 'FC' TO 50, GO TO NEXT PARAMETER  
;CHANGE 'LC' TO 51, GO TO THE NEXT TEST  
;MOVE TO NEXT TEST  
;USE TEST 4'S PARAMETERS AND START PROGRAM EXECUTION

EXAMPLE #7

-----

DRIVE=1<CR>  
TEST=D/<CR>  
TEST 20  
R=1 / 1000<CR>  
FC=0 / 10<CR>  
LC=410 / 10<CR>  
FC'=0 / <CR>  
LC'=814 / <CR>  
IC=64 / 0<CR>  
FT=0 / 2<CR>  
LT=18 / 2<CR>  
IT=1 / <CR>  
FS=0 / 4<CR>  
LS=22 / 4<CR>

;SELECT AND OPEN THE DATA TEST  
;DO 1000 ITERATION OF TEST PATTERN  
;#8 ON CYLINDER 10, TRACK 2, SECTOR 4  
;RPO6 PARAMETER  
;RPO6 PARAMETER

PAT=177777 / 400..<CR> ;RUN WITH PATTERN #8

EXAMPLE #8

-----  
DRIVE=1<CR> ;USE THE SAME PARAMETERS AS IN EXAMPLE  
TEST=D/<CR> ;#7, BUT ALSO SPECIFY A DATA PATTERN (PAT #0).  
TEST 20  
R=1000 / <CR>  
FC=10 / <CR>  
LC=10 / <CR>  
FC'=0 / <CR>  
LC'=814 / <CR>  
IC=0 / <CR>  
FT=2 / <CR>  
LT=2 / <CR>  
IT=1 / <CR>  
FS=4 / <CR>  
LS=4 / <CR>  
PAT=000400 / 401<CR> ;RUN WITH PATTERNS #8 & #0 (0=OPERATOR INPUT)  
WD1=165555 / 125252<CR> ;FIRST WORD OF PATTERN 0  
WD2=133333 / 52525..<CR> ;SECOND WORD OF PATTERN 0  
;<..> START EXECUTION

EXAMPLE #9

-----  
DRIVE=0,1,4<CR> ;TEST DRIVES 0,1, AND 4 IN SEQUENCE  
TEST=0-5/<CR> ;CHANGE TEST 5  
TEST 0  
R=10 / <CR>  
FC=0 / <CR>  
LC=410 / 1<CR> ;CHANGE LAST CYLINDER FROM 410 TO 1  
FC'=0 / <CR>  
LC'=814 / 2..<CR> ;CHANGE THE LAST CYLINDER FOR ALL RPO6'S TO  
;2. 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<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 RPO4/5/6 INTERRUPT. THE 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

'SWR = NNNNNN NEW ='

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

ON PROCESSORS WITH HARDWARE SWITCH REGISTERS, THE 'SOFTWARE' SWITCH REGISTER MAY BE USED. IF THE PROGRAM FINDS ALL 16 SWITCHES IN THE '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...00 IMPLIED SEEKS WITH DATA TRANSFERS  
          -1...00 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

## 6. 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:

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

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

### 6.1 ERROR TYPES

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

#### 6.1.1 DRIVER ERROR

THESE ERRORS WILL BE DETECTED BY THE RH11/RPO4/5/6 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. ACT11 AUTOMATIC MODE ASSUMES 16 BIT MODE.

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.

THERE IS A RESTRICTION ON PROGRAMMABLE DRIVES IN CERTAIN SITUATIONS SEE SECTIONS 2.4 AND 4.1.

8. MISCELLANEOUS

8.1 EXECUTION TIME

-----  
THE PROGRAM REQUIRES APPROXIMATELY 15 MINUTES TO MAKE ONE PASS WITH RPO4/5 DRIVES AND APPROXIMATELY 16.5 MINUTES TO A PASS WITH RPO6 DRIVES. THIS ASSUMES THE DEFAULT TEST SEQUENCE (TESTS 0-10, 12-20)

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.

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

### 8.3.1 TIMING TOLERANCES

#### 1. TEST 12 -- ROTATIONAL SPEED TIMES

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

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

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

MAXIMUM=10000 US  
NOMINAL=7000 US

#### 3. TEST 14 -- ACCESS TIME MEASUREMENT

MAXIMUM=30000 US  
NOMINAL=28000 US

#### 4. TEST 15 -- MAXIMUM SEEK TIMES

MAXIMUM=52000 US  
NOMINAL=50000 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 409 SEEKS TIMED  
\* REVERSE  
MIN=5140 US  
MAX=5960 US  
AVG=5430 US 410 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=49990 US  
MAX=51980 US  
AVG=51010 US 128 SEEKS TIMED  
\* REVERSE  
MIN=48120 US  
MAX=50650 US  
AVG=49340 US 128 SEEKS TIMED

EXAMPLE #2  
-----

ROTATIONAL SPEED TIMES  
MIN=16670 US  
MAX=16690 US  
AVG=16680 US 10 SEEKS TIMED

ONE CYLINDER SEEK TIMES  
\* FORWARD  
MIN=5470 US  
MAX=10940 US 3 ABOVE THE MAXIMUM OF 10000 US  
AVG=5830 US 409 SEEKS TIMED  
\* REVERSE  
MIN=5040 US  
MAX=5970 US  
AVG=5330 US 410 SEEKS TIMED

ACCESS TIME MEASUREMENTS

\* FORWARD  
MIN=29730 US  
MAX=31620 US 73 ABOVE THE MAXIMUM OF 30000 US  
AVG=30320 US 128 SEEKS TIMED  
\* REVERSE  
MIN=28620  
MAX=31230 US 128 ABOVE THE MAXIMUM OF 30000 US  
AVG=30800 US 128 SEEKS TIMED

MAXIMUM SEEK TIMES

\* FORWARD  
MIN=53510 US  
MAX=54240 US 128 ABOVE THE MAXIMUM OF 52000 US  
AVG=54020 US 128 SEEKS TIMED  
\* REVERSE  
MIN=52050 US  
MAX=54550 US 128 ABOVE THE MAXIMUM OF 52000 US  
AVG=52210 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. 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	-	410
LC'	-	814
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	-	128
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	-	410
FC	-	0
LC'	-	814
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
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	-	410
FC'	-	0
LC'	-	814
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

1 2  
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	-	410
FC'	-	0
LC'	-	814
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
NC	-	0
LC	-	410
FC	-	0
LC'	-	814
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	-	410
FC'	-	0
LC'	-	814
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 (RPLA) TO DETERMINE THE ADDRESS OF THE SECTOR ROTATING INTO POSITION. THE PROGRAM THEN ISSUES A WRITE HEADER AND DATA COMMAND FOR THAT SECTOR. IF THE DRIVE'S POSITIONER HAS NOT SETTLED DOWN OR IF THE POSITIONER IS NOT ON CYLINDER (IF THE DRIVE IS AN RPO4, THE OFF CYLINDER CONDITION MUST LAST FOR AT LEAST 800 US), THE DRIVE WILL REPORT A 'WRU' ERROR. (RPO5/6'S MAY ALSO REPORT 'NHS' ERROR UNDER ERRORS IN THIS TEST INDICATE THAT THE SERVO SYSTEM MAY NOT BE ADJUSTED CORRECTLY, THAT THE DRIVE IS MALFUNCTIONING, OR THAT A PCAK WITH MARGINAL SERVO TRACKS IS MOUNTED ON THE DRIVE.

THIS TEST USES THE EXTENTION BITS IN THE LOOK-AHEAD REGISTER TO DETERMINE WHETHER OR NOT IT CAN PICK UP THE SECTOR ROTATING INTO POSITION. THE TEST IS OPTIMIZED SUCH THAT IF THE DRIVE SIGNALS SEEK DONE WITHIN THE FIRST 80% OF THE SECTOR CURRENTLY UNDER THE HEAD, THE TEST WILL TRY TO ADDRESS THE NEXT SECTOR. BASED ON OBSERVATION, THE PROGRAM IS ABLE TO START THE OPERATION WITHOUT LOSING A REVOLUTION MOST OF THE TIME.

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	-	410
FC'	-	0
LC'	-	814
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	-	410
FC'	-	0
LC'	-	814
IC	-	1
FT	-	0
FS	-	0

9.11 TEST 12 - ROTATIONAL SPEED TIMING TEST

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

16.67 MS/REV + OR - 2% IF 60HZ  
16.67 MS/REV + OR - 2.5% IF 50HZ.

THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

R	-	1
FC	-	0
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	-	410
FC'	-	0
LC'	-	814

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 30 MS. CYLINDER 'LC' DEFAULTS TO 136 (10) FOR AN RP04/5 OR TO 255 (10) FOR AN RP06.

THE TEST USES THE FOLLOWING PARAMETERS:

R	-	1
FC	-	0
LC	-	136
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 410 (10) FOR RP04/5'S AND TO 814 (10) FOR RP06'S.

THE TEST USES THE FOLLOWING PARAMETERS:

R	-	1
FC	-	0
LC	-	410
FC'	-	0
LC'	-	814

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 - 21 ARE WRITE CHECKED. THEN SECTOR 1 IS REWRITTEN AND SECTORS 0 - 21 ARE WRITE CHECKED. THIS REWRITE AND WRITE CHECK PROCEDURE IS CONTINUED UP THROUGH REWRITE SECTOR 21 AND WRITE CHECK SECTORS 0-21.

THE TEST USES THE FOLLOWING PARAMETERS:

R	-	1
FC	-	0
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 18 IS WRITE CHECKED. THEN TRACK 1 IS REWRITTEN AND TRACK 2 THROUGH TRACK 18 IS WRITE CHECKED. THIS PROCEDURE IS CONTINUED UP THROUGH REWRITING TRACK 17 AND WRITE CHECKING TRACK 18.

THE TEST USES THE FOLLOWING PARAMETERS:

R	-	1
FC	-	0
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	177767	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	-	410
FC'	-	0
LC'	-	814
IC	-	64

FT	-	0
LT	-	18
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	-	410
FC'	-	0
LC'	-	814

9.19 TEST 22 - RP04 ACCESS TIME ADJUSTMENT TEST

THIS TEST PERFORMS SEEKS BETWEEN CYLINDERS 0 & 136 TO ALLOW THE OPERATOR TO ADJUST THE ACCESS TIME ON AN RP04 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	-	136
FC'	-	0
LC'	-	255

10. PROGRAM LISTING

-----

1449  
 1450  
 1451  
 1452  
 1453  
 1454  
 1455  
 1456  
 1457  
 1458  
 1459  
 1460  
 1461  
 1462  
 1463  
 1464  
 1465  
 1466  
 1467  
 1468  
 1469  
 1470  
 1471  
 1472  
 1473  
 1474  
 1475  
 1476  
 1477  
 1478  
 1479  
 1480  
 1481  
 1482  
 1483  
 1484  
 1485  
 1486  
 1487  
 1488  
 1489  
 1490  
 1491  
 1492  
 1493  
 1494  
 1495  
 1496  
 1497  
 1498  
 1499  
 1500  
 1501  
 1502

```
.TITLE CZRJAD RP04/5/6 MECHANICAL AND READ/WRITE TEST
.*COPYRIGHT (C) 1976,1978
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754
.*
.*PROGRAM BY C. HESS
.*
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
.*
```

.SBTTL CONTROL SWITCH SETTINGS

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

.SBTTL OPERATIONAL SWITCH SETTINGS

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

.SBTTL TRAP CATCHER



```

1503
1504      000000      .-0
1505      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,MALT"
1506      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
1507      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
1508      .-174
1509      000174      000174      DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
1510      000176      000000      SWREG:   .WORD 0      ;;SOFTWARE SWITCH REGISTER
1511
1512      .SBTTL ACT11 HOOKS
1513
1514      ;*****
1515      ;HOOKS REQUIRED BY ACT11
1516      000200      $SVPC=      ;SAVE PC
1517      000046      .-46
1518      000046      017732      $ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
1519      000052      000052      .-52
1520      000052      000000      .WORD 0      ;;2)SET LOC.52 TO ZERO
1521      000200      000200      .= $SVPC      ;; RESTORE PC
1522
1523      .SBTTL STARTING ADDRESSES
1524      000200      .-200
1525      ;*200 = NORMAL START, INHIBIT PROGRAMMABLE DRIVES STANDALONE OR XXDP CHAIN
1526      000200      000137      004674      JMP @#START1
1527      ;*204 = SELECT OPERATING PARAMETERS, INHIBIT PROGRAMMABLE DRIVES
1528      000204      000137      004730      JMP @#START2
1529      ;*210 = SELECT RH11/RP04/5/6 ADDRESSES, DO NOT INHIBIT PROGRAMMABLE DRIVES
1530      000210      000137      004656      JMP @#START3
1531      ;*214 = COMBINATION OF 204 AND 210, DO NOT INHIBIT PROGRAMMABLE DRIVES
1532      000214      000137      004712      JMP @#START4
1533      ;*220 = SAME AS 200 BUT DO NOT INHIBIT PROGRAMMABLE DRIVES
1534      000220      000137      004642      JMP @#START5
1535      ;*224 = SAME AS 204 BUT DO NOT INHIBIT PROGRAMMABLE DRIVES
1536      000224      000137      004626      JMP @#START6
1537
1538      .SBTTL BASIC DEFINITIONS
1539
1540      ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
1541      001100      STACK= 1100
1542      .EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
1543      .EQUIV !OT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
1544
1545      ;*MISCELLANEOUS DEFINITIONS
1546      000011      HT= 11      ;;CODE FOR HORIZONTAL TAB
1547      000012      LF= 12      ;;CODE FOR LINE FEED
1548      000015      CR= 15      ;;CODE FOR CARRIAGE RETURN
1549      000200      CRLF= 200      ;;CODE FOR CARRIAGE RETURN-LINE FEED
1550      177776      PS= 177776      ;;PROCESSOR STATUS WORD
1551      .EQUIV PS,PSW
1552      177774      STKLM= 177774      ;;STACK LIMIT REGISTER
1553      177772      PIRQ= 177772      ;;PROGRAM INTERRUPT REQUEST REGISTER
1554      177570      DSWR= 177570      ;;HARDWARE SWITCH REGISTER
1555      177570      DDISP= 177570      ;;HARDWARE DISPLAY REGISTER
1556
1557      ;*GENERAL PURPOSE REGISTER DEFINITIONS
1558      000000      RO= %0      ;;GENERAL REGISTER

```

1559	000001	R1=	21	:: GENERAL REGISTER
1560	000002	R2=	22	:: GENERAL REGISTER
1561	000003	R3=	23	:: GENERAL REGISTER
1562	000004	R4=	24	:: GENERAL REGISTER
1563	000005	R5=	25	:: GENERAL REGISTER
1564	000006	R6=	26	:: GENERAL REGISTER
1565	000007	R7=	27	:: GENERAL REGISTER
1566	000006	SP=	26	:: STACK POINTER
1567	000007	PC=	27	:: PROGRAM COUNTER

1568

1569 : \*PRIORITY LEVEL DEFINITIONS

1570	000000	PR0=	0	:: PRIORITY LEVEL 0
1571	000040	PR1=	40	:: PRIORITY LEVEL 1
1572	000100	PR2=	100	:: PRIORITY LEVEL 2
1573	000140	PR3=	140	:: PRIORITY LEVEL 3
1574	000200	PR4=	200	:: PRIORITY LEVEL 4
1575	000240	PR5=	240	:: PRIORITY LEVEL 5
1576	000300	PR6=	300	:: PRIORITY LEVEL 6
1577	000340	PR7=	340	:: PRIORITY LEVEL 7

1578

1579 : \*"SWITCH REGISTER" SWITCH DEFINITIONS

1580	100000	SW15=	100000
1581	040000	SW14=	40000
1582	020000	SW13=	20000
1583	010000	SW12=	10000
1584	004000	SW11=	4000
1585	002000	SW10=	2000
1586	001000	SW09=	1000
1587	000400	SW08=	400
1588	000200	SW07=	200
1589	000100	SW06=	100
1590	000040	SW05=	40
1591	000020	SW04=	20
1592	000010	SW03=	10
1593	000004	SW02=	4
1594	000002	SW01=	2
1595	000001	SW00=	1
1596		.EQUIV	SW09,SW9
1597		.EQUIV	SW08,SW8
1598		.EQUIV	SW07,SW7
1599		.EQUIV	SW06,SW6
1600		.EQUIV	SW05,SW5
1601		.EQUIV	SW04,SW4
1602		.EQUIV	SW03,SW3
1603		.EQUIV	SW02,SW2
1604		.EQUIV	SW01,SW1
1605		.EQUIV	SW00,SW0

1606

1607 : \*DATA BIT DEFINITIONS (BIT00 TO BIT15)

1608	100000	BIT15=	100000
1609	040000	BIT14=	40000
1610	020000	BIT13=	20000
1611	010000	BIT12=	10000
1612	004000	BIT11=	4000
1613	002000	BIT10=	2000
1614	001000	BIT09=	1000

```

1615      000400      BIT08= 400
1616      000200      BIT07= 200
1617      000100      BIT06= 100
1618      000040      BIT05= 40
1619      000020      BIT04= 20
1620      000010      BIT03= 10
1621      000004      BIT02= 4
1622      000002      BIT01= 2
1623      000001      BIT00= 1
1624      .EQUIV BIT09,BIT9
1625      .EQUIV BIT08,BIT8
1626      .EQUIV BIT07,BIT7
1627      .EQUIV BIT06,BIT6
1628      .EQUIV BIT05,BIT5
1629      .EQUIV BIT04,BIT4
1630      .EQUIV BIT03,BIT3
1631      .EQUIV BIT02,BIT2
1632      .EQUIV BIT01,BIT1
1633      .EQUIV BIT00,BIT0
1634
1635      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
1636      000004      ERRVEC= 4          ;;TIME OUT AND OTHER ERRORS
1637      000010      RESVEC= 10         ;;RESERVED AND ILLEGAL INSTRUCTIONS
1638      000014      TBITVEC=14        ;;"T" BIT
1639      000014      TRIVEC= 14         ;;TRACE TRAP
1640      000014      BPTVEC= 14         ;;BREAKPOINT TRAP (BPT)
1641      000020      IOTVEC= 20         ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
1642      000024      PWRVEC= 24         ;;POWER FAIL
1643      000030      EMTVEC= 30         ;;EMULATOR TRAP (EMT) **ERROR**
1644      000034      TRAPVEC=34         ;;"TRAP" TRAP
1645      000060      TKVEC= 60          ;;TTY KEYBOARD VECTOR
1646      000064      TPVEC= 64         ;;TTY PRINTER VECTOR
1647      000240      PIRQVEC=240       ;;PROGRAM INTERRUPT REQUEST VECTOR
1648      ;;*****
1649
1650      .SETTL RH11 REGISTERS
1651
1652      ;;*****
1653
1654      ;CONTROL AND STATUS REGISTER 1 (RPCS1)
1655
1656      000100      IE= 100             ;INTERRUPT ENABLE (BIT #6)
1657      000200      RDY= 200           ;READY (BIT #7)
1658      000400      A16= 400           ;HIGH ORDER BUS ADDRESS BIT (BIT #8)
1659      001000      A17= 1000         ;HIGH ORDER BUS ADDRESS BIT (BIT #9)
1660      002000      PSEL= 2000        ;PORT SELECT (BIT #10)
1661      020000      MCPE= 20000       ;MASSBUSS PARITY ERROR (BIT #13)
1662      040000      TRE= 40000        ;TRANSFER ERROR (BIT #14)
1663      ;SC= 100000                   ;SPECIAL CONDITION (BIT #15)
1664
1665      ;WORD COUNT REGISTER (RPWC)
1666      ;(EACH BIT IS CALLED BY BIT NUMBER)
1667
1668      ;BUS ADDRESS REGISTER (RPBA)
1669      ;(EACH BIT IS CALLED BY BIT NUMBER)
1670
    
```

```

1671                                     ;CONTROL AND STATUS REGISTER 2 (RPCS2)
1672
1673         000001        US1=      1          ;UNIT SELECT (BIT #0)
1674         000002        US2=      2          ;UNIT SELECT (BIT #1)
1675         000004        US4=      4          ;UNIT SELECT (BIT #2)
1676         000010        BAI=     10          ;BUS ADDRESS INCREMENT INHIBIT (BIT #3)
1677                                     ;PAT=     20          ;MASSBUS PARITY TEST (BIT #4)
1678         000040        CLR=     40          ;CLEAR (BIT #5)
1679         000100        IR=     100         ;INPUT READY (BIT #6)
1680         000200        OR=     200         ;OUTPUT READY (BIT #7)
1681         000400        MPE=     400         ;MASS BUS PARITY ERROR (BIT #8)
1682         001000        MXF=    1000        ;MISSED TRANSFER ERROR (BIT #9)
1683         002000        PGE=    2000        ;PROGRAM ERROR (BIT #10)
1684         004000        NEM=    4000        ;NON EXISTENT MEMORY (BIT #11)
1685         010000        NED=   10000       ;NON EXISTENT DRIVE (BIT #12)
1686         020000        UPE=   20000       ;UNIBUS PARITY ERROR (BIT #13)
1687         040000        WCE=   40000       ;WRITE CHECK ERROR (BIT #14)
1688         100000        DLT=  100000       ;DATA LATE (BIT #15)
1689
1690                                     ;DATA BUFFER REGISTER (RPDB)
1691                                     ;(EACH BIT IS CALLED BY BIT NUMBER)
1692
1693
1694                                     ;:*****
1695
1696         .SBTTL  RP04/5/6 REGISTERS
1697
1698                                     ;:*****
1699
1700                                     ;CONTROL AND STATUS 1 REGISTER. (#00)
1701
1702         000001        GO=      1          ;GO BIT .BIT #0)
1703         000002        F1=      2          ;FUNCTION CODE BIT #1
1704         000004        F2=      4          ;FUNCTION CODE BIT #2
1705         000010        F3=     10          ;FUNCTION CODE BIT #3
1706         000020        F4=     20          ;FUNCTION CODE BIT #4
1707         000040        F5=     40          ;FUNCTION CODE BIT #5
1708         004000        DVA=    4000        ;DEVICE AVAILABLE (BIT #11)
1709
1710                                     ;DRIVE STATUS REGISTER (RPDS1) (#01)
1711
1712         ;DFS=      1          DRIVE FORWARD 5"/SEC. (BIT #0)
1713         000002        DFF20=  2          ;DRIVE FORWARD 20"/SEC. (BIT #1)
1714         000004        DIGB=   4          ;DRIVE TO INNER GUARD BAND (BIT #2)
1715         000010        GRV=    10         ;GO REVERSE (BIT #3)
1716         000020        DL64=   20         ;DIFFERENCE LESS THAN 64 (BIT #4)
1717         000040        DE1=    40         ;DIFFERENCE EQUALS 1 (BIT #5)
1718         000100        VV=    100         ;VOLUME VALID (BIT #6)
1719         000200        DRY=    200         ;DRIVE READY (BIT #7)
1720         000400        DPR=    400         ;DRIVE PRESENT (BIT #8)
1721         001000        PGM=   1000        ;PROGRAMABLE (BIT #9)
1722         002000        LST=   2000        ;LAST SECTOR TRANSFERRED (BIT #10)
1723         004000        WRL=   4000        ;WRITE LOCK (BIT #11)
1724         010000        MOL=  10000       ;MEDIUM ON-LINE (BIT #12)
1725         020000        PIP=  20000       ;POSITIONING OPERATION IN PROGRESS (BIT #13)
1726         040000        ERR=  40000       ;COMPOSITE ERROR (BIT #14)
    
```

```

1727      100000      ATA= 100000      ;ATTENTION ACTIVE (BIT #15)
1728
1729      ;ERROR REGISTER #01 (RPER1) (#02)
1730
1731      000001      ILF= 1      ;ILLEGAL FUNCTION (BIT #0)
1732      000002      ILR= 2      ;ILLEGAL REGISTER (BIT #1)
1733      000004      RMR= 4      ;REGISTER MODIFICATION REFUSED (BIT #2)
1734      000010      PAR= 10     ;PARITY ERROR (BIT #3)
1735      000020      FER= 20     ;FORMAT ERROR (BIT #4)
1736      000040      WCF= 40     ;WRITE CLOCK FAIL (BIT #5)
1737      000100      ECH= 100    ;ECC HARD ERROR (BIT #6)
1738      000200      HCE= 200    ;HEADER COMPARE ERROR (BIT #7)
1739      000400      HCRC= 400   ;HEADER CRC ERROR (BIT #8)
1740      001000      AOE= 1000   ;ADDRESS OVERFLOW ERROR (BIT #9)
1741      002000      IAE= 2000   ;INVALID ADDRESS ERROR (BIT #10)
1742      004000      WLE= 4000   ;WRITE LOCK ERROR (BIT #11)
1743      010000      DTE= 10000  ;DRIVE TIMING ERROR (BIT #12)
1744      020000      OPI= 20000  ;OPERATION INCOMPLETE (BIT #13)
1745      040000      UNS= 40000  ;DRIVE UNSAFE (BIT #14)
1746      100000      DCK= 100000 ;DATA CHECK ERROR (BIT 15)
1747
1748      ;MAINTAINABILITY REGISTER (RPMR)(#03)
1749
1750      000001      DMD= 1      ;DIAGINOSTIC MODE (BIT #0)
1751      000002      MCLK= 2     ;MAINTAINABILITY CLOCK (BIT #1)
1752      000004      MINX= 4     ;MAINTAINABILITY INDEX (BIT #2)
1753      000010      MSTCK= 10   ;MAINTAINABILITY SECTOR CLOCK (BIT #3)
1754      000020      MRD= 20     ;MAINTAINABILITY READ (BIT #4)
1755      000040      MWR= 40     ;MAINTAINABILITY WRITE (BIT #5)
1756      000200      DTSY= 200   ;MAINTAINABILITY SYNC DETECTED (BIT #7)
1757
1758      ;ATTENTION SUMMARY PSEUDO-REGISTER (RPAS) (#04)
1759
1760      000001      AT0= 1      ;DEVICE 0 (BIT #0)
1761      000002      AT1= 2      ;DEVICE 1 (BIT #1)
1762      000004      AT2= 4      ;DEVICE 2 (BIT #2)
1763      000010      AT3= 10     ;DEVICE 3 (BIT #3)
1764      000020      AT4= 20     ;DEVICE 4 (BIT #4)
1765      000040      AT5= 40     ;DEVICE 5 (BIT #5)
1766      000100      AT6= 100    ;DEVICE 6 (BIT #6)
1767      000200      AT7= 200    ;DEVICE 7 (BIT #7)
1768
1769      ;DESIRED SECTOR/TRACK ADDRESS REGISTER (RPDA) (#05)
1770      ;(EACH BIT IS CALLED BY BIT NUMBER)
1771
1772      ;DRIVE TYPE REGISTER (RPDT) (#06)
1773
1774      000001      DT00= 1     ;DRIVE TYPE NUMBER BIT 1
1775      000002      DT01= 2     ;DRIVE TYPE NUMBER BIT 2
1776      000004      DT02= 4     ;DRIVE TYPE NUMBER BIT 3
1777      000010      DT03= 10    ;DRIVE TYPE NUMBER BIT 4
1778      000020      DT04= 20    ;DRIVE TYPE NUMBER BIT 5
1779      000040      DT05= 40    ;DRIVE TYPE NUMBER BIT 6
1780      000100      DT06= 100   ;DRIVE TYPE NUMBER BIT 7
1781      000200      DT07= 200   ;DRIVE TYPE NUMBER BIT 8
1782      000400      DT08= 400   ;DRIVE TYPE NUMBER BIT 9
  
```

1783	004000	DRQ= 4000	:DRIVE REQUEST REQUIRED (BIT #11)
1784	020000	MOH= 20000	:MOVING HEAD (BIT #13)
1785	040000	TAP= 40000	:TAPE DRIVE (BIT #14)
1786	100000	NBA= 100000	:NOT BLOCK ADDRESSED (BIT #15)
1787			
1788		:LOOK-AHEAD REGISTER (RPLA) (#07)	
1789			
1790	000001	EXT1= 1	:EXTENSION 1 (BIT #0)
1791	000002	EXT2= 2	:EXTENSION 2 (BIT #1)
1792	000004	EXT4= 4	:EXTENSION 3 (BIT #2)
1793	000010	EXT10= 10	:EXTENSION 4 (BIT #3)
1794	000020	EXT20= 20	:EXTENSION 5 (BIT #4)
1795	000040	EXT40= 40	:EXTENSION 6 (BIT #5)
1796	000100	SC1= 100	:SECTOR COUNT FIELD 0 (BIT #6)
1797	000200	SC2= 200	:SECTOR COUNT FIELD 1 (BIT #7)
1798		:SC4= 400	:SECTOR COUNT FIELD 2 (BIT #8)
1799	001000	SC10= 1000	:SECTOR COUNT FIELD 3 (BIT #9)
1800	002000	SC20= 2000	:SECTOR COUNT FIELD 4 (BIT #10)
1801	004000	TRK1= 4000	:TRACK FIELD 1 (BIT #11)
1802	010000	TRK2= 10000	:TRACK FIELD 2 (BIT #12)
1803	020000	TRK4= 20000	:TRACK FIELD 3 (BIT #13)
1804	040000	TRK10= 40000	:TRACK FIELD 4 (BIT #14)
1805	100000	TRK20= 100000	:TRACK FIELD 5 (BIT #15)
1806			
1807		:RP04 ERROR REGISTER #2 (RPER2) (#10)	
1808			
1809	000001	WCU= 1	:WRITE CURRENT UNSAFE (BIT #0)
1810	000002	CSF= 2	:CURRENT SINK FAILURE (BIT #1)
1811	000004	WSU= 4	:WRITE SELECT UNSAFE (BIT #2)
1812	000010	CSU= 10	:CURRENT SWITCH UNSAFE (BIT #3)
1813	000020	MSE= 20	:MOTOR SEQUENCE ERROR (BIT #4)
1814	000040	TDF= 40	:TRANSITIONS DETECTOR FAILURE (BIT #5)
1815	000100	TUF= 100	:TRANSITIONS UNSAFE (BIT #6)
1816	000200	FEN= 200	:FAILSAFE ENABLED (BIT #7)
1817	000400	WRU= 400	:WRITE READY UNSAFE (BIT #8)
1818	001000	MHS= 1000	:MULTIPLE HEAD SELECT (BIT #9)
1819	002000	NHS= 2000	:NO HEAD SELECTION (BIT #10)
1820	004000	IXE= 4000	:INDEX ERROR (BIT #11)
1821	010000	VU30= 10000	:30VOLT UNSAFE (BIT #12)
1822	020000	PLU= 20000	:PLO UNSAFE (BIT #13)
1823	100000	ACU= 100000	:AC UNSAFE (BIT #15)
1824			
1825		:RP05/6 ERROR REGISTER #02 (RPER2) (#10)	
1826			
1827	000001	WCU= 1	:WRITE CURRENT UNSAFE (BIT #0)
1828	000002	CSF= 2	:CURRENT SINK FAILURE (BIT #1)
1829	000004	WSU= 4	:WRITE SELECT UNSAFE (BIT #2)
1830	000010	CSU= 10	:CURRENT SWITCH UNSAFE (BIT #3)
1831	000020	RAW= 20	:READ AND WRITE (BIT #4)
1832	000040	TDF= 40	:TRANSITIONS DETECTOR FAILURE (BIT #5)
1833	000100	TUF= 100	:TRANSITIONS UNSAFE (BIT #6)
1834	000200	ABS= 200	:ABNORMAL STOP (BIT #7)
1835	000400	WRU= 400	:WRITE READY UNSAFE (BIT #8)
1836	001000	MHS= 1000	:MULTIPLE HEAD SELECT (BIT #9)
1837	002000	NHS= 2000	:NO HEAD SELECTION (BIT #10)
1838	004000	IXE= 4000	:INDEX ERROR (BIT #11)

```

1839          020000          PLU=    20000          ;PLO UNSAFE (BIT #12)
1840
1841          ;OFFSET REGISTER (RPOF) (#11)
1842
1843          000001          OF25=   1          ;OFFSET 25 MICRO INCHES (BIT #0)
1844          000002          OF50=   2          ;OFFSET 50 MICRO INCHES (BIT #1)
1845          000004          OF100=  4          ;OFFSET 100 MICRO INCHES (BIT #2)
1846          000010          OF200= 10          ;OFFSET 200 MICRO INCHES (BIT #3)
1847          000020          OF400= 20          ;OFFSET 400 MICRO INCHES (BIT #4)
1848          000040          OF800= 40          ;OFFSET 800 MICRO INCHES (BIT #5)
1849          000200          OFREV= 200         ;OFFSET NEGATIVE (REVERSE) (BIT #5)
1850          002000          HCI=   2000        ;HEADER COMPARE INHIBIT (BIT #10)
1851          004000          ECI=   4000        ;ERROR CORRECTION CODE INHIBIT (BIT #11)
1852          010000          FMT22= 10000       ;FORMAT BIT (BIT #12)
1853
1854          ;DESIRED CYLINDER ADDRESS (RPCA) (#12)
1855          ;(EACH BIT IS CALLED BY BIT NUMBER)
1856
1857          ;CURRENT CYLINDER ADDRESS (RPCC) (#13)
1858          ;(EACH BIT IS CALLED BY BIT NUMBER)
1859
1860          ;SERIAL NUMBER REGISTER (RPSN) (#14)
1861          ;(EACH IS CALLED BY BIT NUMBER)
1862
1863          ;RPO4 ERROR REGISTER #03 (RPER3) (#15)
1864
1865          000001          PSU=    1          ;PACK SPEED UNSAFE (BIT #0)
1866          000002          VUF=    2          ;VELOCITY UNSAFE (BIT #1)
1867          000010          UWR=   10          ;ANY UNSAFE EXCEPT READ/WRITE (BIT #3)
1868          000040          ACL=   40          ;AC LOW (BIT #5)
1869          000100          DCL=  100          ;DC LOW (BIT #6)
1870          040000          SKI=  40000        ;SEEK INCOMPLETE (BIT #14)
1871          100000          OCYL= 100000       ;OFF CYLINDER (BIT #15)
1872
1873          ;RPO5/6 ERROR REGISTER #03 (RPER3) (#15)
1874
1875          000001          DCU=    1          ;DC UNSAFE (BIT #0)
1876          000002          WAO=    2          ;WRITE AND OFFSET (BIT #1)
1877          000040          ACL=   40          ;AC LOW (BIT #5)
1878          000100          DCL=  100          ;DC LOW (BIT #6)
1879          020000          OPE=  20000        ;OPERATOR PLUG ERROR (BIT #13)
1880          040000          SKI=  40000        ;SEEK INCOMPLETE (BIT #14)
1881          100000          OCYL= 100000       ;OFF CYLINDER ERROR (BIT #15)
1882
1883          ;ECC POSITION REGISTER (RPEL1) (#16)
1884          ;(EACH BIT IS CALLED BY BIT NUMBER)
1885
1886          ;ECC PATTERN REGISTER (RPEC2) (#17)
1887          ;(EACH BIT IS CALLED BY BIT NUMBER)
1888
1889          ;*****
1890
1891          ;OP CODE DEFINITIONS
1892          000101          NOOP=101
1893          000103          UNLOAD=103
1894          000105          SEEK=105
    
```

1895	000107	RECAL=107
1896	000111	DRVCLR=111
1897	000113	RELEASE=113
1898	000115	OFFSET=115
1899	000117	RTC=117
1900	000121	READIN=121
1901	000123	PACK=123
1902	000131	SEARCH=131
1903	000151	WRCKD=151
1904	000153	WRCKHD=153
1905	000161	WRITE=161
1906	000163	WRTHD=163
1907	000171	READ=171
1908	000173	READHD=173
1909	000141	GETREG=141
1910	000143	SETFORM=143
1911	000145	SELDRV=145

1912		
1913		:OTHER EQUATES
1914		
1915	177400	SCTRWC = -256.
1916	010000	FMT22=10000
1917		

:WORD COUNT FOR SECTOR  
:FORMAT 22 BIT



1918  
 1919  
 1920  
 1921  
 1922  
 1923  
 1924  
 1925  
 1926  
 1927  
 1928  
 1929  
 1930  
 1931  
 1932  
 1933  
 1934  
 1935  
 1936  
 1937  
 1938  
 1939  
 1940  
 1941  
 1942  
 1943  
 1944  
 1945  
 1946  
 1947  
 1948  
 1949  
 1950  
 1951  
 1952  
 1953  
 1954  
 1955  
 1956  
 1957  
 1958  
 1959  
 1960  
 1961  
 1962  
 1963  
 1964  
 1965  
 1966  
 1967  
 1968  
 1969  
 1970  
 1971  
 1972  
 1973

001100  
 001100 000000  
 001102 000  
 001103 000  
 001104 000000  
 001106 000000  
 001110 000000  
 001112 000000  
 001114 000  
 001115 001  
 001116 000000  
 001120 000000  
 001122 000000  
 001124 000000  
 001126 000000  
 001130 000000  
 001132 000000  
 001134 000  
 001135 000  
 001136 000000  
 001140 177570  
 001142 177570  
 001144 177560  
 001146 177562  
 001150 177564  
 001152 177566  
 001154 000  
 001155 002  
 001156 012  
 001157 000  
 001160 000000  
 001162 000000  
 001164 000000  
 001166 000000  
 001170 000000  
 001172 000000  
 001174 000000  
 001176 000000  
 001200 000000  
 001202 000000  
 001204 000000  
 001206 000000  
 001210 177607  
 001214 077  
 001215 015  
 001216 000012  
 000C15

000377

.SBTTL COMMON TAGS

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

. = 1100  
 \$CMTAG: .WORD 0  
 \$PASS: .WORD 0  
 \$STNM: .BYTE 0  
 \$ERFLG: .BYTE 0  
 \$ICNT: .WORD 0  
 \$LPADR: .WORD 0  
 \$LPERR: .WORD 0  
 \$ERTTL: .WORD 0  
 \$ITEMB: .BYTE 0  
 \$ERMAX: .BYTE 1  
 \$ERRPC: .WORD 0  
 \$GDADR: .WORD 0  
 \$BDADR: .WORD 0  
 \$GDDAT: .WORD 0  
 \$BDDAT: .WORD 0  
 .WORD 0  
 \$AUTOB: .BYTE 0  
 \$INTAG: .BYTE 0  
 .WORD 0  
 \$SWR: .WORD DSWR  
 \$DISPLAY: .WORD DDISP  
 \$TKS: 177560  
 \$TKB: 177562  
 \$TPS: 177564  
 \$TPB: 177566  
 \$NULL: .BYTE 0  
 \$FILLS: .BYTE 2  
 \$FILLC: .BYTE 12  
 \$TPFLG: .BYTE 0  
 \$REGAD: .WORD 0  
 \$REG0: .WORD 0  
 \$REG1: .WORD 0  
 \$REG2: .WORD 0  
 \$REG3: .WORD 0  
 \$REG4: .WORD 0  
 \$REG5: .WORD 0  
 \$TMP0: .WORD 0  
 \$TMP1: .WORD 0  
 \$TMP2: .WORD 0  
 \$TIMES: 0  
 \$ESCAPE: 0  
 \$BELL: .ASCIZ <207><377><377>  
 \$QUES: .ASCII /?/  
 \$CRLF: .ASCII <15>  
 \$LF: .ASCIZ <12>  
 \*\*\*\*\*  
 CR = 15

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

1974		000012		LF = 12	
1975	001220	000000		C.SWR: .WORD 0	:CONTROL SWITCHES
1976	001222	000000		SAVCSW: .WORD 0	:PREVIOUS CONTENTS OF 'C.SWR'
1977	001224	000000		CNTRLC: .WORD 0	:CONTROL 'C' FLAG
1978	001226	000000		BUSADR: .WORD 0	:GET ADDRESSES FROM THE TTY FLAG (0=NO, -1=YES)
1979	001230	000000		LPTAVL: .WORD 0	:LPT AVAILABLE STATUS (0=NO,1=YES)
1980	001232	000000		DRVSEL: .WORD 0	:DRIVES SELECTED FOR TESTING
1981	001234	037777	000000	TSTNMS: .WORD 37777.0	:RUN TESTS 0-15
1982	001240	000000	000000	OPNFLG: .WORD 0,0	:MODIFY TEST PARAMETER FLAGS
1983	001244	000000		CLKSTA: .WORD 0	:CLOCK STATUS (0=NO CLOCK,+1=KW11-P, AND -1=KW11-L)
1984					:16 MILLISECONDS PER CLOCK TICK
1985	001246	000020		TICKMS: .WORD 16.	:16666 MICROSECONDS PER CLOCK TICK
1986	001250	040432		TICKUS: .WORD 16666.	
1987	001252	000000		BYPASS: .WORD 0	
1988	001254	000000		CHKDRV: .WORD 0	:DRIVE UNDER TEST
1989	001256	000000		DRVMSK: .WORD 0	:DRIVE MASK BIT
1990	001260	000000		SVSTAT: .WORD 0	:STATUS/ERROR INDICATOR IS SAVED HERE ON AN ERROR
1991					:CYLINDER READ
1992	001262	000000		CYL.RD: .WORD 0	:TRACK READ
1993	001264	000000		TRK.RD: .WORD 0	:SECTOR READ
1994	001266	000000		SEC.RD: .WORD 0	:CYLINDER DESIRED
1995	001270	000000		CYL.DS: .WORD 0	:SECTOR DESIRED
1996	001272	000000		SEC.DS: .WORD 0	:TRACK DESIRED
1997	001274	000000		TRK.DS: .WORD 0	:MINIMUM TIME
1998	001276	000000		TIM.UP: .WORD 0	:NUMBER OF COUNTS BELOW MIN. LIMIT
1999	001300	000000		.WORD 0	:MAXIMUM TIME
2000	001302	000000		.WORD 0	:NUMBER OF COUNTS ABOVE MAX. LIMIT
2001	001304	000000		.WORD 0	:TOTAL TIME OF ALL SEEKS
2002	001306	000000	000000	.WORD 0,0	:NUMBER OF SEEKS PERFORMED
2003	001312	000000		.WORD 0	:MINIMUM TIME
2004	001314	000000		TIM.DN: .WORD 0	:NUMBER OF COUNTS BELOW MIN. LIMIT
2005	001316	000000		.WORD 0	:MAXIMUM TIME
2006	001320	000000		.WORD 0	:NUMBER OF COUNTS ABOVE MAX. LIMIT
2007	001322	000000		.WORD 0	:TOTAL TIME OF ALL SEEKS
2008	001324	000000	000000	.WORD 0,0	:NUMBER OF SEEKS PERFORMED
2009	001330	000000		.WORD 0	:POINTS TO TABLE OF TIMES
2010	001332	000000		TIM.PT: .WORD 0	:FATAL WRITE CHECK ERROR FLAG (TEST 20)
2011	001334	000000		WCEFLG: .WORD 0	:VARIABLE STALL (TEST 4)
2012	001336	000000		STALLO: .WORD 0	:SAVE DISK ADDRESS (TEST 22)
2013	001340	000000	000000	SVADR: .WORD 0,0	:SEEK TIMER (TEST 10)
2014	001344	000000		SEKTR: .WORD 0	:SEEK COUNTER
2015	001346	000000		SEKCN: .WORD 0	:TESTING RANGE FOR SERVO SETTLE DOWN TEST
2016	001350	000000		DELTA: .WORD 0	:WORD COUNT FOR A FULL TRACK IN 16 BIT MODE
2017	001352	165000		TRCKWC: .WORD -<256.*22.>	:10 MILLISECONDS STALL (TEST 0-11)
2018	001354	000012		STALL1: .WORD 10.	:10 MILLISECONDS STALL (TEST 16-21)
2019	001356	000012		STALL2: .WORD 10.	:5 SEC STALL (TEST 22)
2020	001360	011610		STALL3: .WORD 5000.	:MAX. INCREMENTING STALL ALLOWED IN TEST 4
2021	001362	000031		MXSTAL: .WORD 25.	:NUMBER OF ERRORS ALLOWED IN TESTS 16 - 21
2022	001364	144		ERR.CT: .BYTE 100.	:BEFORE GOING TO THE NEXT TEST
2023					:RESERVED
2024	001365	000		.BYTE 0	
2025					
2026				:ADDRESSES AND VECTORS	
2027	001366	176700		RH.ADR: .WORD 176700	:RH11-RH70 UNIBUS ADDRESS
2028	001370	000254	000240	RHVEC: .WORD 254,5*32.	:RH11-RH70 VECTOR ADDRESS AND PRIORITY
2029	001374	000104	000106	PKV: .WORD 104,106	:KW11-P VECTOR ADDRESS

2030	001400	172540		PKCS:	.WORD	172540		:KW11-P CONTROL AND STATUS REG.
2031	001402	172542		PKB:	.WORD	172542		:KW11-P COUNT SET BUFFER
2032	001404	172544		PKC:	.WORD	172544		:KW11-P COUNTER
2033	001406	000100	000102	LKV:	.WORD	100,102		:KW11-L VECTOR ADDRESS
2034	001412	177546		LKS:	.WORD	177546		:KW11-L STATUS REGISTER
2035	001414	177564		TPS:	.WORD	177564		:TTY PRINTER STATUS
2036	001416	177566		TPB:	.WORD	177566		:TTY PRINTER BUFFER
2037	001420	177514		LPS:	.WORD	177514		:LINE PRINTER STATUS
2038	001422	177516		LPB:	.WORD	177516		:LINE PRINTER BUFFER

2039								
2040								
2041	001424	000001						
2042	001426	000002						
2043	001430	000004						
2044	001432	000010						
2045	001434	000020						
2046	001436	000040						
2047	001440	000100						
2048	001442	000200						
2049	001444	000400						
2050	001446	001000						
2051	001450	002000						
2052	001452	004000						
2053	001454	010000						
2054	001456	020000						
2055	001460	040000						
2056	001462	100000						
2057	001464	000001						
2058	001466	000002						
2059	001470	000004						
2060	001472	000010						
2061	001474	000020						
2062	001476	000040						
2063	001500	000100						
2064	001502	000200						

: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

2065								
2066								
2067								
2068	001504	000000						
2069	001506	000000						
2070	001510	000000						
2071	001512	000000						
2072	001514	000000						
2073	001516	000000						
2074	001520	000000						
2075	001522	000000						
2076	001524	000000						
2077	001526	000000						
2078	001530	000000						
2079	001532	000000						
2080	001534	000000						

:COMMON STORAGE FOR TEST PARAMETER  
PRM: .WORD 0  
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

2081								
2082								
2083	001536	002330						
2084	001540	002344						
2085	001542	002372						

:TABLE OF PARAMETER POINTERS  
PRMPT: .WORD PRM0  
.WORD PRM1  
.WORD PRM2

2086	001544	002414	.WORD	PRM3	
2087	001546	002436	.WORD	PRM4	
2088	001550	002460	.WORD	PRM5	
2089	001552	002502	.WORD	PRM6	
2090	001554	002524	.WORD	PRM7	
2091	001556	002544	.WORD	PRM10	
2092	001560	002564	.WORD	PRM11	
2093	001562	002606	.WORD	PRM12	
2094	001564	002622	.WORD	PRM13	
2095	001566	002636	.WORD	PRM14	
2096	001570	002652	.WORD	PRM15	
2097	001572	002666	.WORD	PRM16	
2098	001574	002700	.WORD	PRM17	
2099	001576	002712	.WORD	PRM20	
2100	001600	002744	.WORD	PRM21	
2101	001602	002760	.WORD	PRM22	
2102	001604	000000	.WORD	0	; TERMINATOR
2103					
2104					
2105	001606	032767	; PARAMETER UPPER LIMIT		
2106	001610	000632	PRMLMT: .WORD	32767	;'R''
2107	001612	000632	.WORD	410.	;'FC''
2108	001614	001456	.WORD	410.	;'LC''
2109	001616	001456	.WORD	814.	;'FC''
2110	001620	001456	.WORD	814.	;'LC''
2111	001622	000022	.WORD	814.	;'IC''
2112	001624	000022	.WORD	18.	;'FT''
2113	001626	000022	.WORD	18.	;'LT''
2114	001630	000025	.WORD	18.	;'IT''
2115	001632	000025	.WORD	21.	;'FS''
2116	001634	177777	.WORD	21.	;'LS''
2117					;'PAT''
2118					
2119	001636	043146	; TABLE OF MESSAGE POINTERS		
2120	001640	043150	PRMMSG: .WORD	MSG.R	
2121	001642	043153	.WORD	MSG.FC	
2122	001644	043156	.WORD	MSG.LC	
2123	001646	043162	.WORD	MSG.FC	
2124	001650	043166	.WORD	MSG.LC	
2125	001652	043171	.WORD	MSG.IC	
2126	001654	043174	.WORD	MSG.FT	
2127	001656	043177	.WORD	MSG.LT	
2128	001660	043202	.WORD	MSG.IT	
2129	001662	043205	.WORD	MSG.FS	
2130			.WORD	MSG.LS	
2131					
2132					
2133	001664	001125	; STATUS/ERROR INDICATOR MESSAGES POINTER TABLE		
2134	001672	001456	; DEFAULT VALUES OF TEST PARAMETERS		
2135	001700	003377	DFLT: .WORD	1125,200.,410.,814.,0,0	; RECAL/SEEK (T0)
2136	001706	000200	.WORD	3377,100.,0,128.,0,256.,0,0,0,0,0	; SEEK/SEEK (T1)
2137	001714	000000			
2138	001722	000000			
2139	001726	001177	.WORD	1177,1,0,410.,0,814.,1,0,0	; INCREMENT SEEK (T2)
2140	001734	000632			
2141	001742	000001			

```

2142 001750 001177 000010 000000 .WORD 1177,10,0,256.,0,256.,1,0,0 ;STEPPING SEEK (T3)
2143 001756 000400 000000 000400
2144 001764 000001 000000 000000
2145 001772 001177 000001 000000 .WORD 1177,1,0,410.,0,814.,1,0,0 ;OSCILLATING SEEK (T4)
2146 002000 000632 000000 001456
2147 002006 000001 000000 000000
2148 002014 001177 000001 000000 .WORD 1177,1,0,410.,0,814.,1,0,0 ;CONVERGING/DIVERGING SEEK (T5)
2149 002022 000632 000000 001456
2150 002030 000001 000000 000000
2151 002036 001177 000001 000000 .WORD 1177,1,0,410.,0,814.,1,0,0 ;SERVO ADDRESSING LOGIC NOISE (T6)
2152 002044 000632 000000 001456
2153 002052 000001 000000 000000
2154 002060 000337 011610 000000 .WORD 337,5000.,0,410.,0,814.,0,18. ;RANDOM SEEK TEST (T7)
2155 002066 000632 000000 001456
2156 002074 000000 000022
2157 002100 000177 000001 000000 .WORD 177,1,0,410.,0,814.,100.,0 ;SERVO SETTLE DOWN TEST (T10)
2158 002106 000632 000000 001456
2159 002114 000144 000000
2160 002120 001177 000001 000000 .WORD 1177,1,0,410.,0,814.,1,0,0 ;ALL SEEKS TEST (T11)
2161 002126 000632 000000 001456
2162 002134 000001 000000 000000
2163 002142 001113 000001 000000 .WORD 1113,1,0,0,0,0 ;ROTATIONAL SPEED TIMING TEST (T12)
2164 002150 000000 000000 000000
2165 002156 000037 000001 000000 .WORD 37,1,0,410.,0,814. ;ONE CYLINDER SEEK TIMING TEST (T13)
2166 002164 000632 000000 001456
2167 002172 000037 000001 000000 .WORD 37,1,0,136.,0,255. ;ACCESS TIME MEASUREMENT TEST (T14)
2168 002200 000210 000000 000377
2169 002206 000037 000001 000000 .WORD 37,1,0,410.,0,814. ;MAXIMUM SEEK TIMING TEST (T15)
2170 002214 000632 000000 001456
2171 002222 000113 000001 000000 .WORD 113,1,0,0,0 ;SECTOR ADDRESSING TEST (T16)
2172 002230 000000 000000
2173 002234 001013 000001 000000 .WORD 1013,1,0,0,0 ;TRACK ADDRESSING TEST (T17)
2174 002242 000000 000000
2175 002246 007777 000001 000000 .WORD 7777,1,0,410.,0,814.,64.,0,18.,1,1,0,17777 ;DATA TEST (T20)
2176 002254 000632 000000 001456
2177 002262 000100 000000 000022
2178 002270 000001 000001 000000
2179 002276 177777
2180 002300 000037 047040 000000 .WORD 37,20000.,0,410.,0,814. ;EXERCISER (T21)
2181 002306 000632 000000 001456
2182 002314 000037 011610 000000 .WORD 37,5000.,0,136.,0,255. ;RP04 ACCESS TIME ADJUSTMENT TEST (T22)
2183 002322 000210 000000 000377

```

;PARAMETER TABLES

```

;RECAL/SEEK (T0)
PRM0: .WORD 1125
      .WORD 200.
      .WORD 410.
      .WORD 814.
      .WORD 0
      .WORD 0

```

```

;SEEK/SEEK (T1)
PRM1: .WORD 3377
      .WORD 100.

```

```

2184
2185
2186
2187
2188 002330 001125
2189 002332 000310
2190 002334 000632
2191 002336 001456
2192 002340 000000
2193 002342 000000
2194
2195
2196 002344 003377
2197 002346 000144

```

2198	002350	000000	.WORD	0
2199	002352	000200	.WORD	128.
2200	002354	000000	.WORD	0
2201	002356	000400	.WORD	256.
2202	002360	000000	.WORD	0
2203	002362	000000	.WORD	0
2204	002364	000000	.WORD	0
2205	002366	000000	.WORD	0
2206	002370	000000	.WORD	0

; INCREMENT SEEK (T2)

2209	002372	001177	PRM2: .WORD	1177
2210	002374	000001	.WORD	1
2211	002376	000000	.WORD	0
2212	002400	000632	.WORD	410.
2213	002402	000000	.WORD	0
2214	002404	001456	.WORD	814.
2215	002406	000001	.WORD	1
2216	002410	000000	.WORD	0
2217	002412	000000	.WORD	0

; STEPPING SEEK (T3)

2220	002414	001177	PRM3: .WORD	1177
2221	002416	000001	.WORD	1
2222	002420	000000	.WORD	0
2223	002422	000400	.WORD	256.
2224	002424	000000	.WORD	0
2225	002426	001000	.WORD	512.
2226	002430	000001	.WORD	1
2227	002432	000000	.WORD	0
2228	002434	000000	.WORD	0

; OSCILLATING SEEK (T4)

2231	002436	001177	PRM4: .WORD	1177
2232	002440	000001	.WORD	1
2233	002442	000000	.WORD	0
2234	002444	000632	.WORD	410.
2235	002446	000000	.WORD	0
2236	002450	001456	.WORD	814.
2237	002452	000001	.WORD	1
2238	002454	000000	.WORD	0
2239	002456	000000	.WORD	0

; CONVERGING/DIVERGING SEEK (T5)

2242	002460	001177	PRM5: .WORD	1177
2243	002462	000001	.WORD	1
2244	002464	000000	.WORD	0
2245	002466	000632	.WORD	410.
2246	002470	000000	.WORD	0
2247	002472	001456	.WORD	814.
2248	002474	000001	.WORD	1
2249	002476	000000	.WORD	0
2250	002500	000000	.WORD	0

; SERVO ADDRESSING LOGIC NOISE GENERATOR (T6)

2253	002502	001177	PRM6: .WORD	1177
------	--------	--------	-------------	------

2254	002504	000001	.WORD	1
2255	002506	000000	.WORD	0
2256	002510	000632	.WORD	410.
2257	002512	000000	.WORD	0
2258	002514	001456	.WORD	814.
2259	002516	000001	.WORD	1
2260	002520	000000	.WORD	0
2261	002522	000000	.WORD	0
2262				
2263			;RANDOM SEEK TEST (T7)	
2264	002524	000337	PRM7: .WORD	337
2265	002526	011610	.WORD	5000.
2266	002530	000000	.WORD	0
2267	002532	000632	.WORD	410.
2268	002534	000000	.WORD	0
2269	002536	001456	.WORD	814.
2270	002540	000000	.WORD	0
2271	002542	000022	.WORD	18.
2272				
2273			;SERVO SETTLE DOWN TEST (T10)	
2274	002544	000177	PRM10: .WORD	177
2275	002546	000001	.WORD	1
2276	002550	000000	.WORD	0
2277	002552	000632	.WORD	410.
2278	002554	000000	.WORD	0
2279	002556	001456	.WORD	814.
2280	002560	000144	.WORD	100.
2281	002562	000000	.WORD	0
2282				
2283			;ALL SEEKS TEST (T11)	
2284	002564	001177	PRM11: .WORD	1177
2285	002566	000001	.WORD	1
2286	002570	000000	.WORD	0
2287	002572	000632	.WORD	410.
2288	002574	000000	.WORD	0
2289	002576	001456	.WORD	814.
2290	002600	000001	.WORD	1
2291	002602	000000	.WORD	0
2292	002604	000000	.WORD	0
2293				
2294			;ROTATIONAL SPEED TIMING TEST (T12)	
2295	002606	001113	PRM12: .WORD	1113
2296	002610	000001	.WORD	1
2297	002612	000000	.WORD	0
2298	002614	000000	.WORD	0
2299	002616	000000	.WORD	0
2300	002620	000000	.WORD	0
2301				
2302			;ONE CYLINDER SEEK TIMING TEST (T13)	
2303	002622	000037	PRM13 .WORD	37
2304	002624	000001	.WORD	1
2305	002626	000000	.WORD	0
2306	002630	000632	.WORD	410.
2307	002632	000000	.WORD	0
2308	002634	001456	.WORD	814.
2309				

```
2310 ;ACCESS TIME MEASUREMENT TEST (T14)
2311 002636 000037 PRM14: .WORD 37
2312 002640 000001 .WORD 1
2313 002642 000000 .WORD 0
2314 002644 000210 .WORD 136.
2315 002646 000000 .WORD 0
2316 002650 000377 .WORD 255.
2317
2318 ;MAXIMUM SEEK TIMING TEST (T15)
2319 002652 000037 PRM15: .WORD 37
2320 002654 000001 .WORD 1
2321 002656 000000 .WORD 0
2322 002660 000632 .WORD 410.
2323 002662 000000 .WORD 0
2324 002664 001456 .WORD 814.
2325
2326 ;SECTOR ADDRESSING TEST (T16)
2327 002666 000113 PRM16: .WORD 113
2328 002670 000001 .WORD 1
2329 002672 000000 .WORD 0
2330 002674 000000 .WORD 0
2331 002676 000000 .WORD 0
2332
2333 ;TRACK ADDRESSING TEST (T17)
2334 002700 001013 PRM17: .WORD 1013
2335 002702 000001 .WORD 1
2336 002704 000000 .WORD 0
2337 002706 000000 .WORD 0
2338 002710 000000 .WORD 0
2339
2340 ;DATA TEST (T20)
2341 002712 007777 PRM20: .WORD 7777
2342 002714 000001 .WORD 1
2343 002716 000000 .WORD 0
2344 002720 000632 .WORD 410.
2345 002722 000000 .WORD 0
2346 002724 001456 .WORD 814.
2347 002726 000100 .WORD 64.
2348 002730 000000 .WORD 0
2349 002732 000022 .WORD 18.
2350 002734 000001 .WORD 1
2351 002736 000001 .WORD 1
2352 002740 000000 .WORD 0
2353 002742 177777 PTRN15: .WORD 177777
2354
2355 ;EXERCISER (T21)
2356 002744 000037 PRM21: .WORD 37
2357 002746 047040 .WORD 20000.
2358 002750 000000 .WORD 0
2359 002752 000632 .WORD 410.
2360 002754 000000 .WORD 0
2361 002756 001456 .WORD 814.
2362
2363 ;RP04 ACCESS TIME ADJUSTMENT TEST (T22)
2364 002760 000037 PRM22: .WORD 37
2365 002762 011610 .WORD 5000.
```



2366 002764 000000  
 2367 002766 000210  
 2368 002770 000000  
 2369 002772 000377  
 2370  
 2371  
 2372  
 2373  
 2374  
 2375 002774 043710  
 2376 002776 000000  
 2377 003000 003142  
 2378 003002 003244  
 2379  
 2380 003004 043710  
 2381 003006 000000  
 2382 003010 003131  
 2383 003012 003255  
 2384  
 2385 003014 043742  
 2386 003016 044011  
 2387 003020 000000  
 2388 003022 001750  
 2389  
 2390 003024 044026  
 2391 003026 044074  
 2392 003030 000000  
 2393 003032 012574  
 2394  
 2395 003034 044111  
 2396 003036 044153  
 2397 003040 000000  
 2398 003042 012574  
 2399  
 2400 003044 003104  
 2401 003046 003144  
 2402 003050 003204  
 2403 003052 003244  
 2404 003054 003304  
 2405 003056 003344  
 2406 003060 003404  
 2407 003062 003444  
 2408 003064 003504  
 2409 003066 003544  
 2410 003070 003604  
 2411 003072 003644  
 2412 003074 003704  
 2413 003076 003744  
 2414 003100 004004  
 2415 003102 004044  
 2416  
 2417  
 2418  
 2419 003104 165555  
 2420 003106 133333  
 2421 003110 165555

.WORD 0  
 .WORD 136.  
 .WORD 0  
 .WORD 255.  
  
 ;SEEK TIMING LIMITS  
 T7A: .WORD MSG7  
 .WORD 0  
 .WORD 1634. ;(16.67-2%)\*2  
 .WORD 1700. ;(16.67+2%)\*2  
  
 T7B: .WORD MSG7  
 .WORD 0  
 .WORD 1625. ;(16.67-2.5%)\*2  
 .WORD 1709. ;(16.67+2.5%)\*2  
  
 T10: .WORD MSG10A  
 .WORD MSG10B  
 .WORD 0 ;NO LOWER LIMIT  
 .WORD 1000. ;(7\*3)\*2  
  
 T11: .WORD MSG11A  
 .WORD MSG11B  
 .WORD 0 ;NO LOWER LIMIT  
 .WORD 5500. ;(28\*2)\*2  
  
 T12: .WORD MSG12A  
 .WORD MSG12B  
 .WORD 0 ;NO LOWER LIMIT  
 .WORD 5500. ;(50\*2)\*2  
  
 PAT.PT: .WORD PAT0 ;TABLE OF POINTERS WHICH POINT TO THE  
 .WORD PAT1 ;PATTERNS USED BY THE DATA TEST  
 .WORD PAT2  
 .WORD PAT3  
 .WORD PAT4  
 .WORD PAT5  
 .WORD PAT6  
 .WORD PAT7  
 .WORD PAT8  
 .WORD PAT9  
 .WORD PAT10  
 .WORD PAT11  
 .WORD PAT12  
 .WORD PAT13  
 .WORD PAT14  
 .WORD PAT15  
  
 ;PATTERNS 0 THRU 15  
 PAT0: .WORD 165555 ;PATTERN 0  
 .WORD 133333  
 .WORD 165555

2422	003112	133333		.WORD	133333	
2423	003114	165555		.WORD	165555	
2424	003116	133333		.WORD	133333	
2425	003120	165555		.WORD	165555	
2426	003122	133333		.WORD	133333	
2427	003124	165555		.WORD	165555	
2428	003126	133333		.WORD	133333	
2429	003130	165555		.WORD	165555	
2430	003132	133333		.WORD	133333	
2431	003134	165555		.WORD	165555	
2432	003136	133333		.WORD	133333	
2433	003140	165555		.WORD	165555	
2434	003142	133333		.WORD	133333	
2435						
2436	003144	000001	PAT1:	.WORD	000001	:PATTERN 1
2437	003146	000003		.WORD	000003	
2438	003150	000007		.WORD	000007	
2439	003152	000017		.WORD	000017	
2440	003154	000037		.WORD	000037	
2441	003156	000077		.WORD	000077	
2442	003160	000177		.WORD	000177	
2443	003162	000377		.WORD	000377	
2444	003164	000777		.WORD	000777	
2445	003166	001777		.WORD	001777	
2446	003170	003777		.WORD	003777	
2447	003172	007777		.WORD	007777	
2448	003174	017777		.WORD	017777	
2449	003176	037777		.WORD	037777	
2450	003200	077777		.WORD	077777	
2451	003202	177777		.WORD	177777	
2452						
2453	003204	177776	PAT2:	.WORD	177776	:PATTERN 2
2454	003206	177774		.WORD	177774	
2455	003210	177770		.WORD	177770	
2456	003212	177760		.WORD	177760	
2457	003214	177740		.WORD	177740	
2458	003216	177700		.WORD	177700	
2459	003220	177600		.WORD	177600	
2460	003222	177400		.WORD	177400	
2461	003224	177000		.WORD	177000	
2462	003226	176000		.WORD	176000	
2463	003230	174000		.WORD	174000	
2464	003232	170000		.WORD	170000	
2465	003234	160000		.WORD	160000	
2466	003236	140000		.WORD	140000	
2467	003240	100000		.WORD	100000	
2468	003242	000000		.WORD	000000	
2469						
2470	003244	000000	PAT3:	.WORD	000000	:PATTERN 3
2471	003246	000000		.WORD	000000	
2472	003250	000000		.WORD	000000	
2473	003252	177777		.WORD	177777	
2474	003254	177777		.WORD	177777	
2475	003256	177777		.WORD	177777	
2476	003260	000000		.WORD	000000	
2477	003262	000000		.WORD	000000	

2478	003264	177777	.WORD	177777	
2479	003266	177777	.WORD	177777	
2480	003270	000000	.WORD	000000	
2481	003272	177777	.WORD	177777	
2482	003274	000000	.WORD	000000	
2483	003276	177777	.WORD	177777	
2484	003300	000000	.WORD	000000	
2485	003302	177777	.WORD	177777	
2486					
2487	003304	000000	PAT4: .WORD	000000	;PATTERN 4
2488	003306	010421	.WORD	010421	
2489	003310	021042	.WORD	021042	
2490	003312	031463	.WORD	031463	
2491	003314	042104	.WORD	042104	
2492	003316	052525	.WORD	052525	
2493	003320	063146	.WORD	063146	
2494	003322	073567	.WORD	073567	
2495	003324	104210	.WORD	104210	
2496	003326	114631	.WORD	114631	
2497	003330	125252	.WORD	125252	
2498	003332	135673	.WORD	135673	
2499	003334	146314	.WORD	146314	
2500	003336	156735	.WORD	156735	
2501	003340	167356	.WORD	167356	
2502	003342	177777	.WORD	177777	
2503					
2504	003344	052525	PAT5: .WORD	052525	;PATTERN 5
2505	003346	052525	.WORD	052525	
2506	003350	052525	.WORD	052525	
2507	003352	125252	.WORD	125252	
2508	003354	125252	.WORD	125252	
2509	003356	125252	.WORD	125252	
2510	003360	052525	.WORD	052525	
2511	003362	052525	.WORD	052525	
2512	003364	125252	.WORD	125252	
2513	003366	125252	.WORD	125252	
2514	003370	052525	.WORD	052525	
2515	003372	125252	.WORD	125252	
2516	003374	052525	.WORD	052525	
2517	003376	125252	.WORD	125252	
2518	003400	052525	.WORD	052525	
2519	003402	125252	.WORD	125252	
2520					
2521	003404	007417	PAT6: .WORD	007417	;PATTERN 6
2522	003406	007417	.WORD	007417	
2523	003410	007417	.WORD	007417	
2524	003412	170360	.WORD	170360	
2525	003414	170360	.WORD	170360	
2526	003416	170360	.WORD	170360	
2527	003420	007417	.WORD	007417	
2528	003422	007417	.WORD	007417	
2529	003424	170360	.WORD	170360	
2530	003426	170360	.WORD	170360	
2531	003430	007417	.WORD	007417	
2532	003432	170360	.WORD	170360	
2533	003434	007417	.WORD	007417	

2534	003436	170360	.WORD	170360	
2535	003440	007417	.WORD	007417	
2536	003442	170360	.WORD	170360	
2537					
2538	003444	026455	PAT7: .WORD	026455	;PATTERN 7
2539	003446	026455	.WORD	026455	
2540	003450	026455	.WORD	026455	
2541	003452	151322	.WORD	151322	
2542	003454	151322	.WORD	151322	
2543	003456	151322	.WORD	151322	
2544	003460	026455	.WORD	026455	
2545	003462	026455	.WORD	026455	
2546	003464	151322	.WORD	151322	
2547	003466	151322	.WORD	151322	
2548	003470	026455	.WORD	026455	
2549	003472	151322	.WORD	151322	
2550	003474	026455	.WORD	026455	
2551	003476	151322	.WORD	151322	
2552	003500	026455	.WORD	026455	
2553	003502	151322	.WORD	151322	
2554					
2555	003504	165555	PAT8: .WORD	165555	;PATTERN 8
2556	003506	133333	.WORD	133333	
2557	003510	165555	.WORD	165555	
2558	003512	133333	.WORD	133333	
2559	003514	165555	.WORD	165555	
2560	003516	133333	.WORD	133333	
2561	003520	165555	.WORD	165555	
2562	003522	133333	.WORD	133333	
2563	003524	165555	.WORD	165555	
2564	003526	133333	.WORD	133333	
2565	003530	165555	.WORD	165555	
2566	003532	133333	.WORD	133333	
2567	003534	165555	.WORD	165555	
2568	003536	133333	.WORD	133333	
2569	003540	165555	.WORD	165555	
2570	003542	133333	.WORD	133333	
2571					
2572	003544	000001	PAT9: .WORD	000001	;PATTERN 9
2573	003546	000002	.WORD	000002	
2574	003550	000004	.WORD	000004	
2575	003552	000010	.WORD	000010	
2576	003554	000020	.WORD	000020	
2577	003556	000040	.WORD	000040	
2578	003560	000100	.WORD	000100	
2579	003562	000200	.WORD	000200	
2580	003564	000400	.WORD	000400	
2581	003566	001000	.WORD	001000	
2582	003570	002000	.WORD	002000	
2583	003572	004000	.WORD	004000	
2584	003574	010000	.WORD	010000	
2585	003576	020000	.WORD	020000	
2586	003600	040000	.WORD	040000	
2587	003602	100000	.WORD	100000	
2588					
2589	003604	177776	PAT10: .WORD	177776	;PATTERN 10

2590	003606	177775	.WORD	177775	
2591	003610	177773	.WORD	177773	
2592	003612	177767	.WORD	177767	
2593	003614	177757	.WORD	177757	
2594	003616	177737	.WORD	177737	
2595	003620	177677	.WORD	177677	
2596	003622	177577	.WORD	177577	
2597	003624	177377	.WORD	177377	
2598	003626	176777	.WORD	176777	
2599	003630	175777	.WORD	175777	
2600	003632	173777	.WORD	173777	
2601	003634	167777	.WORD	167777	
2602	003636	157777	.WORD	157777	
2603	003640	137777	.WORD	137777	
2604	003642	077777	.WORD	077777	
2605					
2606	003644	172666	PAT11: .WORD	172666	;PATTERN 11
2607	003646	155555	.WORD	155555	
2608	003650	172666	.WORD	172666	
2609	003652	155555	.WORD	155555	
2610	003654	172666	.WORD	172666	
2611	003656	155555	.WORD	155555	
2612	003660	172666	.WORD	172666	
2613	003662	155555	.WORD	155555	
2614	003664	172666	.WORD	172666	
2615	003666	155555	.WORD	155555	
2616	003670	172666	.WORD	172666	
2617	003672	155555	.WORD	155555	
2618	003674	172666	.WORD	172666	
2619	003676	155555	.WORD	155555	
2620	003700	172666	.WORD	172666	
2621	003702	155555	.WORD	155555	
2622					
2623	003704	077777	PAT12: .WORD	077777	;PATTERN 12
2624	003706	137777	.WORD	137777	
2625	003710	157777	.WORD	157777	
2626	003712	167777	.WORD	167777	
2627	003714	173777	.WORD	173777	
2628	003716	175777	.WORD	175777	
2629	003720	176777	.WORD	176777	
2630	003722	177377	.WORD	177377	
2631	003724	177577	.WORD	177577	
2632	003726	177677	.WORD	177677	
2633	003730	177737	.WORD	177737	
2634	003732	177757	.WORD	177757	
2635	003734	177767	.WORD	177767	
2636	003736	177773	.WORD	177773	
2637	003740	177775	.WORD	177775	
2638	003742	177776	.WORD	177776	
2639					
2640	003744	153333	PAT13: .WORD	153333	;PATTERN 13
2641	003746	066667	.WORD	066667	
2642	003750	153333	.WORD	153333	
2643	003752	066667	.WORD	066667	
2644	003754	153333	.WORD	153333	
2645	003756	066667	.WORD	066667	

```

2646 003760 153333 .WORD 153333
2647 003762 066667 .WORD 066667
2648 003764 153333 .WORD 153333
2649 003766 066667 .WORD 066667
2650 003770 153333 .WORD 153333
2651 003772 066667 .WORD 066667
2652 003774 153333 .WORD 153333
2653 003776 066667 .WORD 066667
2654 004000 153333 .WORD 153333
2655 004002 066667 .WORD 066667
2656
2657 004004 00J000 PAT14: .WORD 000000 ;PATTERN 14
2658 004006 177777 .WORD 177777
2659 004010 177777 .WORD 177777
2660 004012 177777 .WORD 177777
2661 004014 177777 .WORD 177777
2662 004016 177777 .WORD 177777
2663 004020 177777 .WORD 177777
2664 004022 177777 .WORD 177777
2665 004024 177777 .WORD 177777
2666 004026 177777 .WORD 177777
2667 004030 177777 .WORD 177777
2668
2669 004032 177777 .WORD 177777
2670 004034 177777 .WORD 177777
2671 004036 177777 .WORD 177777
2672 004040 177777 .WORD 177777
2673
2674 004042 177777 .WORD 177777
2675
2676 004044 177777 PAT15: .WORD 177777 ;PATTERN 15
2677 004046 000000 .WORD 000000
2678 004050 000000 .WORD 000000
2679 004052 000000 .WORD 000000
2680 004054 000000 .WORD 000000
2681 004056 000000 .WORD 000000
2682 004060 000000 .WORD 000000
2683 004062 000000 .WORD 000000
2684 004064 000000 .WORD 000000
2685 004066 000000 .WORD 000000
2686 004070 000000 .WORD 000000
2687 004072 000000 .WORD 000000
2688 004074 000000 .WORD 000000
2689 004076 000000 .WORD 000000
2690 004100 000000 .WORD 000000
2691 004102 000000 .WORD 000000
2692
2693 :DPB (DATA PARAMETER BLOCK)
2694
2695 004104 000 DPB.A .BYTE 0 ;(0) DRIVE NUMBER
2696 004105 000 .BYTE 0 ;(1) OFFSET VALUE OR FMT22, ECI, AND HCI
2697 004106 000 .BYTE 0 ;(2) COMMAND
2698 004107 000 .BYTE 0 ;(3) PSEL AND A17 AND A16
2699 004110 000000 .WORD 0 ;(4) WORD COUNT (MUST BE NEG.)
2700 004112 050202 .WORD BUFFER ;(6) BUFFER ADDRESS OR
2701 ;REGISTER TABLE POINTER
  
```

```

;(0) DRIVE NUMBER
;(1) OFFSET VALUE OR FMT22, ECI, AND HCI
;(2) COMMAND
;(3) PSEL AND A17 AND A16
;(4) WORD COUNT (MUST BE NEG.)
;(6) BUFFER ADDRESS OR
;REGISTER TABLE POINTER
  
```

2702	004114	000		.BYTE	0	;(10) SECTOR ADDRESS OR
2703						:FIRST REG. INDEX
2704	004115	000		.BYTE	0	;(11) TRACK ADDRESS OR
2705						:LAST REG. INDEX
2706	004116	000000		.WORD	0	;(12) CYLINDER ADDRESS
2707	004120	004204		.WORD	RP.REG	;(14) ERROR TABLE POINTER
2708						:POINTS TO THE FIRST OF TWENTY
2709						:LOCATIONS OF WHERE THE DRIVER
2710						:IS TO STORE THE RH11/RP04
2711						:REGISTERS ON AN ERROR. IF LEFT
2712						:ZERO REGISTERS ARE NOT SAVED.
2713	004122	000000		.WORD	0	;(16) STATUS/ERROR INDICATOR
2714						:BIT15=1=>ERROR OCCURRED
2715						:BIT07=1=>DONE
2716						:BIT14-BIT09 AND BIT06-BIT03
2717						:INDICATE TYPE OF ERROR
2718						
2719	004124	000	DPB.B:	.BYTE	0	;(0) DRIVE NUMBER
2720	004125	000		.BYTE	0	;(1) OFFSET VALUE OR FMT22, ECI, AND HCI
2721	004126	000		.BYTE	0	;(2) COMMAND
2722	004127	000		.BYTE	0	;(3) PSEL AND A17 AND A16
2723	004130	177776		.WORD	-2	;(4) WORD COUNT (MUST BE NEG.)
2724	004132	050202		.WORD	BUFFER	;(6) BUFFER ADDRESS OR
2725						:REGISTER TABLE POINTER
2726	004134	000		.BYTE	0	;(10) SECTOR ADDRESS OR
2727						:FIRST REG. INDEX
2728	004135	000		.BYTE	0	;(11) TRACK ADDRESS OR
2729						:LAST REG. INDEX
2730	004136	000000		.WORD	0	;(12) CYLINDER ADDRESS
2731	004140	004204		.WORD	RP.REG	;(14) ERROR TABLE POINTER
2732						:POINTS TO THE FIRST OF TWENTY
2733						:LOCATIONS OF WHERE THE DRIVER
2734						:IS TO STORE THE RH11/RP04
2735						:REGISTERS ON AN ERROR. IF LEFT
2736						:ZERO REGISTERS ARE NOT SAVED.
2737	004142	000000		.WORD	0	;(16) STATUS/ERROR INDICATOR
2738						:BIT15=1=>ERROR OCCURRED
2739						:BIT07=1=>DONE
2740						:BIT14-BIT09 AND BIT06-BIT03
2741						:INDICATE TYPE OF ERROR
2742						
2743	004144	000	DPB.C:	.BYTE	0	;(0) DRIVE NUMBER
2744	004145	000		.BYTE	0	;(1) OFFSET VALUE OR FMT22, ECI, AND HCI
2745	004146	000		.BYTE	0	;(2) COMMAND
2746	004147	000		.BYTE	0	;(3) PSEL AND A17 AND A16
2747	004150	177776		.WORD	-2	;(4) WORD COUNT (MUST BE NEG.)
2748	004152	050202		.WORD	BUFFER	;(6) BUFFER ADDRESS OR
2749						:REGISTER TABLE POINTER
2750	004154	000		.BYTE	0	;(10) SECTOR ADDRESS OR
2751						:FIRST REG. INDEX
2752	004155	000		.BYTE	0	;(11) TRACK ADDRESS OR
2753						:LAST REG. INDEX
2754	004156	000000		.WORD	0	;(12) CYLINDER ADDRESS
2755	004160	004204		.WORD	RP.REG	;(14) ERROR TABLE POINTER
2756						:POINTS TO THE FIRST OF TWENTY
2757						:LOCATIONS OF WHERE THE DRIVER





2814 004252 000000  
2815  
2816  
2817 004254 045212  
2818 004256 045254  
2819 004260 045305  
2820 004262 045327  
2821 004264 045355  
2822 004266 045400  
2823 004270 045437  
2824 004272 045501  
2825 004274 045545  
2826 004276 045615  
2827 004300 045635  
2828 004302 045705  
2829 004304 045755

.WORD 0

;RPEC2 (776746) ECC PATTERN

;STATUS/ERROR MESSAGE POINTER TABLE

STATBL: .WORD MSGB14 ;OFFLINE OR UNSAFE DRIVE REQUESTED  
.WORD MSGB13 ;UNLOAD DRIVE REQUESTED  
.WORD MSGB12 ;PERSISTENT UNSAFE  
.WORD MSGB11 ;PARITY ERROR OCCURRED  
.WORD MSGB10 ;FATAL PARITY ERROR  
.WORD MSGB09 ;SOFTWARE TIMEOUT ON THIS DRIVE  
.WORD MSGB08 ;SOFTWARE TIMEOUT ON ANOTHER DRIVE  
.WORD MSGB06 ;ERROR OCCURRED DURING I/O OPERATION  
.WORD MSGB05 ;ERROR OCCURRED DURING NON-I/O OPERATION  
.WORD MSGB04 ;UNSAFE OCCURRED  
.WORD MSGB03 ;AUTOMATIC RECALIBRATE SEQUENCE OCCURRED  
.WORD MSGB02 ;DRIVE HAS NOT RESPONDED TO PORT REQUEST  
.WORD MSGB01 ;DRIVE HAS BECOME NONEXISTENT

2830  
2831  
2832  
2833  
2834  
2835  
2836  
2837  
2838  
2839  
2840  
2841  
2842  
2843  
2844  
2845  
2846  
2847  
2848  
2849  
2850  
2851  
2852  
2853  
2854  
2855  
2856  
2857  
2858  
2859  
2860  
2861  
2862  
2863  
2864  
2865  
2866  
2867  
2868  
2869  
2870  
2871  
2872  
2873  
2874  
2875  
2876  
2877  
2878  
2879  
2880  
2881  
2882  
2883  
2884  
2885

004306

004306 044333  
004310 046013  
004312 047374  
004314 050026  
  
004316 044376  
004320 046030  
004322 047400  
004324 050032  
  
004326 044434  
004330 046116  
004332 047416  
004334 050036

```
.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 ASCII 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
;* RH11 INTERRUPT OCCURED (RPAS - 0)
;* ERR PC RPAS
;* $ERRPC $REG3

EM1
DH1
DT1
DF1

;* ERROR ITEM 2
;* UNEXPECTED ATTENTION OCCURRED
;* ERR PC DRIVE RPAS RPDS1 RPER1 RPER2 RPER3
;* $ERRPC $REG1 $REG3 RPERRS RPERRS+2 RPERRS+4 RPERRS+6

EM2
DH2
DT2
DF2

;* ERROR ITEM 3
;* MASSBUS PARITY ERROR (MCPE-1)
;* TEST ERR PC ADDRESS DATA
;* $TMP0 $ERRPC RD.ADR RD.WRD

EM3
DH3
DT3
DF3

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

```

2886      :* $TMP0 $ERRPC WRT.ADR WRT.WD RD.WRD
2887
2888      :*          EM4
2889      :*          DH4
2890      :*          DT4
2891      :*          DF4
2892
2893      :* ERROR ITEM 5
2894      :* ADDRESS PLUG CHANGE BIT SET
2895      :* ERR PC DRIVE RPAS RPDS1 RPER1 RPER2 RPER3
2896      :* $ERRPC $REG1 $REG3 RPERRS RPERRS+2 RPERRS+4 RPERRS+6
2897
2898      :*          EM5
2899      :*          DH2
2900      :*          DT2
2901      :*          DF2
2902
2903      :* ERROR ITEM 6 -- NOT USED
2904
2905      :*          0
2906      :*          0
2907      :*          0
2908      :*          0
2909
2910      :* ERROR ITEM 7 -- NOT USED
2911
2912      :*          0
2913      :*          0
2914      :*          0
2915      :*          0
2916
2917      :* ERROR ITEM 10
2918      :* RH11/RPO4/5/6 FAILED TO RESPOND TO ADDRESSING
2919      :* RPCS1 ERR PC
2920      :* RH.ADR $ERRPC
2921
2922      :*          EM10
2923      :*          DH10
2924      :*          DT10
2925      :*          DF10
2926
2927      :* ERROR ITEM 11
2928      :* DRIVE SELECTED IS NOT ONLINE
2929      :* DRIVE ERR PC
2930      :* $REG2 $ERRPC
2931
2932      :*          EM11
2933      :*          DH11
2934      :*          DT11
2935      :*          DF11
2936
2937      :* ERROR ITEM 12
2938      :* IMPROPER HEADER DATA
2939      :* TEST ERR PC TST PC DRIVE CYLNR TRACK SECTOR
2940      :* $TMP0 $ERRPC $REG0 CHKDRV CYL.DS TRK.DS SEC.DS
2941      :* GDCYL GDTRK GDSCR BDCYL BDRK BDSCTR
  
```

```

2942          : *      CYL.DS TRK.DS SEC.DS CYL.RD TRK.RD SEC.RD
2943          : *      CYLNDR, TRACK, AND SECTOR ARE DECIMAL
2944
2945 004416    044674      EM12
2946 004420    046260      DH12
2947 004422    047470      DT12
2948 004424    050056      DF12
2949
2950          : * ERROR ITEM 13
2951          : * DATA COMPARE FAILURE
2952          : * TEST   ERR PC TST PC DRIVE  CYLNDR TRACK  SECTOR
2953          : * $TMPO  $ERRPC $REGO  CHKDRV CYL.DS TRK.DS SEC.DS
2954          : *      GDDAT  BDDAT  WRDCNT GDADR  BDADR
2955          : *      $GDDAT $BDDAT $REG4  $GDADR $BDADR
2956          : *      CYLNDR, TRACK, SECTOR, AND WRDCNT ARE DECIMAL
2957
2958 004426    044721      EM13
2959 004430    046260      DH12
2960 004432    047522      DT13
2961 004434    050066      DF13
2962
2963          : * ERROR ITEM 14 -- FOLLOWS #13
2964          : *      $GDDAT $BDDAT $REG4  $GDADR $BDADR
2965
2966 004436    000000      0
2967 004440    000000      0
2968 004442    047540      DT13A
2969 004444    050076      DF14
2970
2971          : * ERROR ITEM 15
2972          : * DATA COMPARE FAILURE
2973          : * TEST   ERR PC TST PC DRIVE  CYLNDR TRACK  SECTOR
2974          : * $TMPO  $ERRPC $REGO  CHKDRV CYL.DS TRK.DS SEC.DS
2975          : *      GDDAT  BDDAT  WRDCNT GDADR  BDADR
2976          : *      $GDDAT $BDDAT $REG4  $GDADR $BDADR
2977          : *      CYLNDR, TRACK, SECTOR, AND WRDCNT ARE DECIMAL
2978
2979 004446    044721      EM13
2980 004450    046260      DH12
2981 004452    047522      DT13
2982 004454    050066      DF13
2983
2984          : * ERROR ITEM 16 -- FOLLOWS #15
2985          : *      $GDDAT $BDDAT $REG4  $GDADR $BDADR
2986
2987 004456    000000      0
2988 004460    000000      0
2989 004462    047540      DT13A
2990 004464    050076      DF14
2991
2992          : * ERROR ITEM 17
2993          : * DISK ERROR IN TIMING TEST
2994          : * TEST   ERR PC DRIVE  RPCS1  RPDS1  RPER1  RPER2  RPER3
2995          : * $TMPO  $ERRPC (CHKDRV RP.REG  RP.REG+12 RP.REG+14 RP.REG+40 RP.REG+62
2996
2997 004466    044746      EM17

```

2998 004470 046474  
2999 004472 047552  
3000 004474 050102

DH17  
DI17  
DF17

3001  
3002  
3003  
3004  
3005  
3006

:\* ERROR ITEM 20  
:\* CLOCK (KW11-P) OVERFLOW IN TIMING TEST  
:\* TEST ERR PC DRIVE RPCS1 RPDS1 RPER1 RPER2 RPER3  
:\* \$TMPO \$ERRFC \$CHKDRV \$P.REG \$P.REG+12 \$P.REG+14 \$P.REG+40 \$P.REG+42

3007 004476 045000  
3008 004500 046474  
3009 004502 047552  
3010 004504 050102

EM20  
DH17  
DI17  
DF17

3011  
3012  
3013  
3014  
3015  
3016  
3017  
3018  
3019

:\* ERROR ITEM 21  
:\* DATA COMPARE FAILURE  
:\* TEST ERR PC TST FC DRIVE CYLNR TRACK  
:\* \$TMPO \$ERRPC \$REGO \$CHKDRV \$CYL.DS \$TRK.DS  
:\* \$GDDAT \$BDDAT \$WRDCNT \$SECTOR  
:\* \$REG1 \$BDDAT \$REG4 \$REG1  
:\* CYLINDER, TRACK, WRDCNT, AND SECTOR ARE DECIMAL

3020 004506 044721  
3021 004510 046572  
3022 004512 047572  
3023 004514 050106

EM13  
DH21  
DI21  
DF21

3024  
3025  
3026

:\* ERROR ITEM 22--FOLLOWS #21  
:\* \$REG1 \$BDDAT \$REG4 \$REG1

3027  
3028 004516 000000  
3029 004520 000000  
3030 004522 047606  
3031 004524 050116

0  
0  
DI21A  
DF22

3032  
3033

:\* ERROR ITEM 23  
:\* DISK ERROR DURING SEEK  
:\* TEST ERR PC DRIVE CYLNR RPCS1 RPCS2 RPDS1  
:\* \$TMPO \$ERRPC \$CHKDRV \$CYL.DS \$P.REG \$P.REG+10 \$P.REG+12  
:\* RPER1 RPER2 RPER3 \$RPCA \$RPCC  
:\* \$P.REG+14 \$P.REG+40 \$P.REG+42 \$P.REG+34 \$P.REG+36

3034  
3035  
3036  
3037  
3038  
3039

EM23  
DH23  
DI23  
DF23

3040 004526 045047  
3041 004530 046707  
3042 004532 047616  
3043 004534 050122

3044  
3045  
3046  
3047  
3048  
3049  
3050

:\* ERROR ITEM 24  
:\* SEEK NOT COMPLETE WITHIN 120 MS  
:\* TEST ERR PC DRIVE CYLNR RPCS1 RPCS2 RPDS1  
:\* \$TMPO \$ERRPC \$CHKDRV \$CYL.DS \$P.REG \$P.REG+10 \$P.REG+12  
:\* RPER1 RPER2 RPER3 \$RPCA \$RPCC  
:\* \$P.REG+14 \$P.REG+40 \$P.REG+42 \$P.REG+34 \$P.REG+36

3051  
3052 004536 045076  
3053 004540 046707

EM24  
DH23

3054 004542 047616  
 3055 004544 050122

DT23  
 DF23

3056  
 3057  
 3058  
 3059  
 3060  
 3061  
 3062  
 3063  
 3064  
 3065  
 3066  
 3067  
 3068  
 3069  
 3070  
 3071  
 3072  
 3073  
 3074  
 3075  
 3076  
 3077

```

.....
.....
* ERROR ITEMS 23-40 NOT USED
* ERROR ITEMS 41-46 WILL HAVE AN EM THAT
* VARIES DEPENDING ON THE ERROR, IT WILL BE IN THE FORM:
* RH11/RPO4/5/6 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
  
```

3078 004546

ITEM41:

3079  
 3080  
 3081  
 3082  
 3083  
 3084  
 3085  
 3086  
 3087  
 3088

```

* ERROR ITEM 41
* RH11/RPO4/5/6 ERROR (MESSAGE)
* TEST ERR PC TST PC DRIVE
* $TMPO $ERRPC $REGO CHKDRV
  
```

3085 004546 045136  
 3086 004550 047042  
 3087 004552 047646  
 3088 004554 050132

EM41  
 DH41  
 DT41  
 DF41

3089  
 3090  
 3091  
 3092  
 3093

```

* ERROR ITEM 42
* RH11/RPO4/5/6 ERROR (MESSAGE)
* TEST ERR PC TST PC DRIVE RPCS1 RPCS2 RPDS1
* $TMPO $ERRPC $REGO CHKDRV RP.REG RP.REG+10 RP.REG+12
  
```

3095 004556 045136  
 3096 004560 047100  
 3097 004562 047656  
 3098 004564 050136

EM41  
 DH42  
 DT42  
 DF42

3099  
 3100  
 3101  
 3102  
 3103  
 3104  
 3105

```

* ERROR ITEM 43
* RH11/RPO4/5/6 ERROR (MESSAGE)
* TEST ERR PC TST PC DRIVE RPCS1 RPCS2 RPDS1
* $TMPO $ERRPC $REGO CHKDRV RP.REG RP.REG+10 RP.REG+12
* RPER1 RPER2 RPER3
* RP.REG+14 RP.REG+40 RP.REG+42
  
```

3106  
 3107 004566 045136  
 3108 004570 047100  
 3109 004572 047674

EM41  
 DH42  
 DT43



```

3157
3158          .SBTTL  START OF PROGRAM
3159
3160
3161 004626 005037 001226          START6: CLR    @#BUSADR      ;CLR BUSADR FLAG
3162 004632 012737 000001 035446      MOV    #1, TSTPGM    ;ALLOW PROGRAMMABLE DRIVES
3163 004640 000437                    BR     STRT2A        ;
3164 004642 005037 001226          START5: CLR    @#BUSADR      ;CLEAR BUSADR FLAG
3165 004646 012737 000001 035446      MOV    #1, TSTPGM    ;ALLOW PROGRAMMABLE DRIVES
3166 004654 000413                    BR     STRT1A        ;
3167 004656 012737 177777 001226          START3: MOV   #-1,@#BUSADR   ;GET BUSADR FLAG
3168 004664 012737 000001 035446      MOV    #1, TSTPGM    ;ALLOW PROGRAMMABLE DRIVES
3169 004672 000404                    BR     STRT1A        ;
3170 004674 005037 001226          START1: CLR    @#BUSADR      ;CLR BUSADR FLAG
3171 004700 005037 035446          CLR    TSTPGM        ;DISABLE PROGRAMMABLE DRIVES
3172 004704 005037 001224          STRT1A: CLR   @#CNTRLC     ;NO CONTROL 'C'
3173 004710 000416                    BR     START         ;
3174 004712 012737 177777 001226          START4: MOV   #-1,@#BUSADR   ;SET BUSADR FLAG
3175 004720 012737 000001 035446      MOV    #1, TSTPGM    ;ALLOW PROGRAMMABLE DRIVES
3176 004726 000404                    BR     STRT2A        ;
3177 004730 005037 001226          START2: CLR    @#BUSADR      ;CLR BUSADR FLAG
3178 004734 005037 035446          CLR    TSTPGM        ;DISABLE PROGRAMMABLE DRIVES
3179 004740 012737 177777 001224          STRT2A: MOV   #-1,@#CNTRLC   ;SET CONTROL 'C' FLAG
3180 004746 000005          START:  RESET
3181          .SBTTL  INITIALIZE THE COMMON TAGS
3182          ;;CLEAR THE COMMON TAGS (%CMTAG) AREA
3183 004750 012706 001100          MOV    #%CMTAG,R6     ;;FIRST LOCATION TO BE CLEARED
3184 004754 005026                    CLR    (R6)+          ;;CLEAR MEMORY LOCATION
3185 004756 022706 001140          CMP    #SWR,R6      ;;DONE?
3186 004762 001374                    BNE   -6              ;;LOOP BACK IF NO
3187 004764 012706 001100          MOV    #STACK,SP     ;;SETUP THE STACK POINTER
3188          ;;INITIALIZE A FEW VECTORS
3189 004770 012737 022724 000020          MOV    #SCOPE,@#IOTVEC  ;;IOT VECTOR FOR SCOPE ROUTINE
3190 004776 012737 000340 000022          MOV    #340,@#IOTVEC+2 ;;LEVEL 7
3191 005004 012737 017752 000030          MOV    #ERROR,@#EMTVEC  ;;EMT VECTOR FOR ERROR ROUTINE
3192 005012 012737 000340 000032          MOV    #340,@#EMTVEC+2 ;;LEVEL 7
3193 005020 012737 023252 000034          MOV    #STRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
3194 005026 012737 000340 000036          MOV    #340,@#TRAPVEC+2;LEVEL 7
3195 005034 012737 176543 023724          MOV    #176543,$HINUM   ;;PRIME THE RANDOM NUMBER GENERATOR
3196 005042 012737 123456 023726          MOV    #123456,$LONUM   ;;BOTH HIGH AND LOW WORDS
3197 005050 005037 001204          CLR    $TIMES         ;;INITIALIZE NUMBER OF ITERATIONS
3198 005054 005037 001206          CLR    $ESCAPE        ;;CLEAR THE ESCAPE ON ERROR ADDRESS
3199 005060 112737 000001 001115          MOV    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
3200 005066 012737 005066 001106          MOV    #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
3201 005074 012737 005074 001110          MOV    #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
3202          ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
3203          ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
3204 005102 013746 000004          MOV    @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
3205 005106 012737 005142 000004          MOV    #64,$@#ERRVEC  ;;SET UP ERROR VECTOR
3206 005114 012737 177570 001140          MOV    #DSWR,$SWR     ;;SETUP FOR A HARDWARE SWICH REGISTER
3207 005122 012737 177570 001142          MOV    #DDISP,$DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
3208 005130 022777 177777 174002          CMP    #-1,$SWR       ;;TRY TO REFERENCE HARDWARE SWR
3209 005136 001012                    BNE   66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
3210          ;;AND THE HARDWARE SWR IS NOT -1
3211 005140 000403                    BR    65$             ;;BRANCH IF NO TIMEOUT
3212 005142 012716 005150          64$:  MOV    #65,$(SP)   ;;SET UP FOR TRAP RETURN
    
```



```

3213 005146 000002          RTI
3214 005150 012737 000176 001140 65$: MOV #SWREG,SWR ;:POINT TO SOFTWARE SWR
3215 005156 012737 000174 001142      MOV #DISPREG,DISPLAY
3216 005164 012637 000004      66$: MOV (SP)+,@#ERRVEC ;:RESTORE ERROR VECTOR
3217
3218 005170 012700 001160          MOV #SREGAD,RO ;:FIRST ADDRESS
3219 005174 005020          1$: CLR (RO)+ ;:CLEAR VARIABLE STORAGE
3220 005176 022700 001210          CMP #SBELL,RO ;:DONE?
3221 005202 001374          BNE 1$ ;:NO--BRANCH
3222 005204 013737 001414 001150      MOV @#TPS,@#STPS ;:SETUP THE STATUS AND BUFFER REG'S
3223 005212 013737 001416 001152      MOV @#TPB,@#STPB ;:FOR THE TYPE ROUTINE
3224 005220 005227 177777          INC #-1 ;:FIRST START ?
3225 005224 001043          BNE 3$ ;:BR IF NOT
3226 005226 023737 000042 000046      CMP @#42,@#46 ;:ACT11 AUTOMATIC MODE?
3227 005234 001004          BNE 4$ ;:BRANCH IF NO
3228 005236 012737 000001 035446      MOV #1, TSTPGM ;:ENABLE PROGRAMMABLE DRIVES
3229 005244 000402          BR 5$
3230 005246 104401 050202          4$: TYPE ,TITLE ;:TYPE THE PROGRAM'S TITLE
3231 005252 005737 035446          5$: TST TSTPGM ;:CAN WE USE PROGRAMMABLE DRIVES?
3232 005256 001402          BEQ 6$ ;:BRANCH IF NO
3233 005260 104401 043242          TYPE ,USE ;:TYPE MSG
3234 005264          6$:
3235 005264 005737 000042          TST 42 ;:AUTO ACCEPT OR CHAIN MODE ?
3236 005270 001006          BNE 2$ ;:BR IF EITHER
3237 005272 122737 000011 000041      CMPB #11,41 ;:LOADED FROM AN RPO4/5/6 ?
3238 005300 001002          BNE 2$ ;:BR IF NOT
3239 005302 104401 050266          TYPE ,LOADRV ;:INSTRUCT THE OPERATOR TO REMOVE THE PACK
3240          ;:ON DRIVE 0 IF DRIVE 0 IS TO BE TESTED
3241 005306 105737 000041          2$: TSTB @#41 ;:LOADED FROM PAPER TAPE ?
3242 005312 001410          BEQ 3$ ;:BR IF NOT
3243 005314 004737 050562          JSR PC,$SIZE ;:SIZE THE MEMORY
3244 005320 023727 050656 100000      CMP $LSTAD,#100000 ;:16K OR MORE ON THE SYSTEM ?
3245 005326 103002          BHS 3$ ;:BR IF YES
3246 005330 104401 050464          TYPE ,NOLOAD ;:INFORM THE OPEATOR THAT THE 'XXDP' LOADER
3247          ;:WILL BE OVERWRITTEN
3248 005334 004737 021462          3$: JSR PC,$TKINT ;:TURN ON THE TTY KEYBOARD INTERRUPT
3249          .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
3250 005340 005737 000042          TST @#42 ;:ARE WE RUNNING UNDER XXDP/ACT?
3251 005344 001006          BNE 67$ ;:BRANCH IF YES
3252 005346 023727 001140 000176      CMP SWR,#SWREG ;:SOFTWARE SWITCH REG SELECTED?
3253 005354 001005          BNE 68$ ;:BRANCH IF NO
3254 005356 104406          GTSWR ;:GET SOFT-SWR SETTINGS
3255 005360 000403          BR 68$
3256 005362 112737 000001 001134      67$: MOVB #1,$AUTOB ;:SET AUTO-MODE INDICATOR
3257 005370          68$:
3258 005370 005227 177777          INC #-1 ;:SEE IF FIRST START
3259 005374 001002          BNE SRTINT ;:BR IF NOT
3260 005376 004737 050660          JSR PC,GETADR ;:GET OR CHECK THE RH11 ADDRESS
3261 005402 104401 001215          SRTINT: TYPE ,$CRLF ;:CR-LF
3262 005406 004737 024150          JSR PC,@#LP.AVL ;:CHECK FOR A LINE PRINTER
3263 005412 005037 177776          CLR @#PS ;:ENSURE THE PRIORITY = 0
3264 005416 012737 000001 001104      MOV #1,$ICNT ;:SET ITERATION COUNT TO 1
3265 005424 004737 031542          JSR PC,@#GETSWR ;:GO CHECK FOR CONTROL SWITCHES
3266 005430 004737 024212          JSR PC,@#ST.CLK ;:INITIALIZE THE CLOCK
3267 005434 004737 034650          SETVEC: JSR PC,RPINI1 ;:CHECK THE DRIVE STATUS
3268 005440 012737 177777 034572      MOV #-1,$SAVEFG ;:SET THE SAVE REGISTERS FLAG
    
```

```

3269 005446 062727 177777 000000 ADD #-1,#0 ;FIRST START ?
3270 005454 103003 BCC 11$ ;BR IF YES
3271 005456 005737 001224 TST CNTRLC ;CONTROL 'C' SWITCH SET ?
3272 005462 001111 BNE SRTDRV ;CONTINUE IF YES
3273 005464 012737 000340 177776 11$: MOV #PR7,PS ;SET PRIORITY TO 7
3274 005472 005004 CLR R4 ;DRIVE TABLE POINTER
3275 005474 104401 001215 TYPE ,$CRLF ;CR-LF
3276 005500 104401 043370 TYPE ,SYSTAT ;TYPE STATUS HEADING
3277 005504 1$:
3278 005504 010446 MOV R4,-(SP) ;;SAVE R4 FOR TYPEOUT
3279 ;:TYPE DRIVE NUMBER
3280 005506 104403 TYPOS ;:GO TYPE--OCTAL ASC!
3281 005510 002 .BYTE 2 ;:TYPE 2 DIGIT(S)
3282 005511 000 .BYTE 0 ;:SUPPRESS LEADING ZEROS
3283 005512 104401 044330 TYPE ,MSG.SP ;SPACES
3284 005516 104401 044330 TYPE ,MSG.SP ;SPACES
3285 005522 105764 034504 TSTB DRVSTA(R4) ;CHECK DRIVE'S STATUS
3286 005526 100425 BMI 4$ ;BR IF UNSAFE
3287 005530 001027 BNE 5$ ;BR IF ONLINE
3288 005532 105764 034514 TSTB DRVSTYP(R4) ;SEE IF OFFLINE OR NONEXISTENT
3289 005536 001404 BEQ 2$ ;BR IF NONEXISTENT
3290 005540 100006 BPL 3$ ;BR IF OFFLINE
3291 005542 104401 043464 TYPE ,NOTRP ;DRIVE NOT AN RP04/5/6
3292 005546 000447 BR 9$ ;CHECK NEXT DRIVE
3293 005550 104401 043437 2$: TYPE ,NOTPRS ;DRIVE NOT PRESENT
3294 005554 000444 BR 9$ ;CHECK NEXT DRIVE
3295 005556 132764 000010 034514 3$: BITB #BIT03,DRVSTYP(R4) ;DRIVE PROGRAMMABLE?
3296 005564 001403 BEQ 12$ ;BRANCH IF NO
3297 005566 104401 043317 TYPE ,NOUSE ;PRINT MSG
3298 005572 000410 BR 6$ ;PRINT DRIVE TYPE
3299 005574 104401 043416 12$: TYPE ,UNTOFF ;DRIVE OFFLINE
3300 005600 000405 BR 6$ ;PRINT DRIVE TYPE
3301 005602 104401 043454 4$: TYPE ,NOTSAF ;DRIVE UNSAFE
3302 005606 000402 BR 6$ ;PRINT DRIVE TYPE
3303 005610 104401 043427 5$: TYPE ,UNTON ;DRIVE ONLINE
3304 005614 104401 044330 6$: TYPE ,MSG.SP ;SPACES
3305 005620 012737 043502 005664 MOV #RP04B,8$ ;ADDRESS OF RP04 MESSAGE
3306 005626 132764 000001 034514 BITB #BIT00,DRVSTYP(R4) ;RP04 ?
3307 005634 001012 BNE 7$ ;BR IF YES
3308 005636 012737 043507 005664 MOV #RP05,8$ ;ADDRESS OF RP05 MESSAGE
3309 005644 132764 000002 034514 BITB #BIT01,DRVSTYP(R4) ;RP05 ?
3310 005652 001003 BNE 7$ ;BR IF YES
3311 005654 012737 043514 005664 MOV #RP06,8$ ;ADDRESS OF RP06 MESSAGE
3312 005662 104401 7$: TYPE ;TYPE THE DRIVE TYPE MESSAGE
3313 005664 000000 8$: .WORD 0 ;MESSAGE ADDRESS HERE
3314 005666 104401 001215 9$: TYPE ,$CRLF ;CR-LF
3315 005672 005204 INC R4 ;INCREMENT DRIVE NUMBER/TABLE POINTER
3316 005674 020427 000010 CMP R4,#8. ;FINISHED ?
3317 005700 001301 BNE 1$ ;BR IF NOT
3318 005702 005037 177776 CLR PS ;SET PRIORITY BACK TO '0'
3319 005706 005737 001224 SRTDRV: TST @CNTRLC ;CONTROL 'C' START/RESTART?
3320 005712 001417 BEQ 1$ ;NO--BRANCH
3321 005714 013746 001222 MOV SAVCSW,-(SP) ;GET THE PREVIOUS 'C.SWR' CONTENTS
3322 005720 063716 001220 ADD C.SWR,(SP) ;SET UP TO SEE IF 'BIT00' IS DIFFERENT
3323 005724 032726 000001 BIT #BIT00,(SP)+ ;IS 'BIT00' DIFFERENT ?
3324 005730 001405 BEQ 9$ ;BR IF NOT

```

```

3325 005732 013737 001220 001222      MOV      C.SWR,SAVCSW      ;STORE PRESENT 'C.SWR' VALUE
3326 005740 004737 024470              JSR      PC,LODFLT        ;RESET PARAMETERS TO THEIR DEFAULT VALUES
3327 005744 004737 031772      9$:     JSR      PC,@#GT.PRM    ;GET PARAMETERS
3328 005750 000420              BR       4$
3329 005752 004737 024470      1$:     JSR      PC,LODFLT        ;SETUP DEFAULT PARAMETERS
3330 005756 005037 001232      CLR      DRVSEL          ;NO DRIVES SELECTED
3331 005762 005000      CLR      R0              ;DETERMINE THE DRIVES THAT
3332 005764 012701 000001      MOV      #1,R1           ;ARE AVAILABLE FOR TESTING
3333 005770 105760 034504      2$:     TSTB     DRVSTA(R0)
3334 005774 003403              BLE     3$
3335 005776 156037 034620 001232      BISB     ATABIT(R0),@#DRVSEL
3336 006004 005200      3$:     INC      R0
3337 006006 106301              ASLB    R1
3338 006010 001367              BNE     2$
3339 006012 005037 034574      4$:     CLR      @#SEEKFG        ;CLEAR SEEK FLAG
3340 006016 032737 000400 001220      BIT      #SW08,@#C.SWR    ;DO SEEK BEFORE DATA TRANSFER?
3341 006024 001002              BNE     5$                ;YES--BRANCH
3342 006026 005137 034574      COM      @#SEEKFG        ;NO
3343 006032 122737 000011 000041 5$:     CMPB     #11,41 ;LOADED FROM AN RP04/5/6 ?
3344 006040 001003              BNE     10$              ;BR IF NOT
3345 006042 042737 000001 001232      BIC      #BIT00,DRVSEL    ;CLEAR THE DRIVE 0 SELECTION BIT
3346 006050 104401 043521      10$:    TYPE     ,DRIVES      ;'DRIVES(S) TO BE TESTED'
3347 006054 005037 017664      CLR      @#SENDCT        ;DETERMINE PASSES TO MAKE AND
3348 006060 005000      CLR      R0              ;THE DRIVES TO BE TESTED
3349 006062 013701 001232      MOV      @#DRVSEL,R1     ;ANY DRIVES SELECTED?
3350 006066 001004              BNE     6$                ;YES--BRANCH
3351 006070 104401 043552      TYPE     ,NONE           ;'NONE'
3352
3353 006074 000137 017502              JMP      @#SEOP          ;GO TO END OF PROGRAM
3354 006100 006201      6$:     ASR      R1           ;REPORT THE DRIVES TO BE TESTED
3355 006102 103011              BCC     7$
3356 006104 005237 017664      INC      @#SENDCT        ;GIVE THIS DRIVE A PASS
3357 006110 010046      MOV      R0,-(SP)        ;SAVE R0 FOR TYPEOUT
3358 006112 104403              TYPOS   ;GO TYPE--OCTAL ASCII
3359 006114      001      .BYTE 1          ;TYPE 1 DIGIT(.)
3360 006115      000      .BYTE 0          ;SUPPRESS LEADING ZEROS
3361 006116 005701              TST     R1               ;MORE DRIVES?
3362 006120 001404              BEQ     8$                ;NO--BRANCH
3363 006122 104401 043557      TYPE     ,COMMA          ;
3364 006126 005200      7$:     INC      R0           ;FORM DRIVE NUMBER
3365 006130 000763              BR       6$
3366 006132 013737 017664 017656 8$:     MOV      @#SENDCT,@#SEOPCT
3367 006140 005737 001244              TST     @#CLKSTA        ;KW11-P AVAILABLE
3368 006144 003006              BGT     RSTRT1          ;YES--BRANCH
3369 006146 032737 036000 001234      BIT      #36000,@#TSTNMS ;NO--ANY TIMING TESTS TO BE PERFORMED?
3370 006154 001402              BEQ     RSTRT1          ;NO--BRANCH
3371 006156 104401 043561      TYPE     ,NOCLOCK       ;'NO KW11-P CLOCK, TIMING TESTS WILL NOT BE PERFORMED'
3372 006162 005737 001232      RSTRT1: TST     DRVSEL    ;ANY DRIVES SELECTED ?
3373 006166 001002              BNE     1$                ;BR IF YES
3374 006170 000137 004730              JMP     START2          ;GET DRIVE SELECTION ENTRY
3375 006174 005037 001254      1$:     CLR      CHKDRV     ;INIT. THE CHECK DRIVE KEY
3376 006200 012737 000001 001256      MOV      #1,DRVMSK      ;START TO CHECK DESIRED DRIVES
3377 006206 033737 001256 001232      RSTRT2: BIT     DRVMSK,DRVSEL ;IS THIS DRIVE SELECTED?
3378 006214 001010              BNE     DRVOK           ;YES--GO CHECK IF DRIVE IS READY FOR TESTING
3379 006216 012706 001100      RESTART: MOV     #STACK,SP ;SETUP THE STACK POINTER
3380 006222 005237 001254              INC     CHKDRV          ;MOVE TO NEXT DRIVE NUMBER

```

```

3381 006226 106337 001256      ASLB   DRVMSK      ;POSITION THE MASK
3382 006232 103753              BCS    RSTR1     ;BRANCH IF THE DRIVE NUMBER NEEDS INITIALIZED
3383 006234 000764              BR     RSTR2
3384
3385 006236 013702 001254      DRVOK: MOV    CHKDRV,R2 ;PICKUP THE DRIVE NUMBER
3386 006242 105762 034504      TSTB  DRVSTA(R2) ;IS DESIRED DRIVE ON-LINE?
3387 006246 003005              BGT    1$        ;YES, BRANCH
3388 006250 104011              ERROR  11        ;DRIVE SELECTED IS NOT ONLINE
3389 006252 043737 001256 001232      BIC    DRVMSK,DRVSEL ;CLEAR DRIVE'S SELECTION BIT
3390 006260 000756              BR     RESTART   ;RETURN
3391 006262 010237 004104      1$:  MOV    R2,@#DPB.A ;SET THE DRIVE NUMBER INTO THE DPB'S
3392 006266 010237 004124      MOV    R2,@#DPB.B
3393 006272 010237 004144      MOV    R2,@#DPB.C
3394 006276 010237 004164      MOV    R2,@#DTADPB
3395 006302 004737 025104      JSR    PC,@#LDCMD ;LOAD COMMAND INTO DPB.B AND DPB.C
3396 006306 012737 017502 001252      MOV    #SEOP,@#BYPASS ;IF ERROR GO TO END OF PROGRAM
3397 006314 112737 000020 004105      MOVB  #FMT22/256.,DPB.A+1 ;ASSUME 16 BIT FORMAT
3398 006322 032737 000001 001220      BIT   #BIT00,C.SWR ;16 BIT FORMAT REQUESTED ?
3399 006330 001402              BEQ    2$        ;BR IF YES
3400 006332 105037 004105      CLRB  DPB.A+1    ;CLEAR THE 'FMT22' BIT
3401 006336 112737 000143 004106 2$:  MOVB  #SFTFORM,DPB.A+2 ;SET THE FORMAT BIT PER DPB.A+1
3402 006344 004037 025150      JSR    R0,@#CALL.A ;GO EXECUTE THE COMMAND
3403 006350 112737 000107 004106      MOVB  #RECAL,@#DPB.A+2 ;RECAL=COMMAND
3404 006356 004037 025150      JSR    R0,@#CALL.A ;GO EXECUTE THE COMMAND
3405 006362 104401 043647      TYPE  ,TESTNG    ;'TESTING DRIVE '
3406 006366 010246      MOV    R2,-(SP) ;SAVE R2 FOR TYPEOUT
3407 006370 104403      TYPOS ;GO TYPE--OCTAL ASCII
3408 006372 001      .BYTE 1 ;TYPE 1 DIGIT(S)
3409 006373 000      .BYTE 0 ;SUPPRESS LEADING ZEROS
3410 006374 104401 044330      TYPE  ,MSG.SP   ;TYPE SPACES
3411 006400 104401 043671      TYPE  ,SERIAL   ;'SERIAL NUMBER '
3412 006404 012700 000004      MOV    #4,R0    ;FOUR DIGITS TO TYPE
3413 006410 013701 004234      MOV    RP.REG+30,R1 ;SERIAL NUMBER
3414 006414 005002              CLR    R2       ;ZERO
3415 006416 006101              ROL   R1        ;PUT THE NEXT DIGIT
3416 006420 006102              ROL   R2        ;INTO R2
3417 006422 006101              ROL   R1
3418 006424 006102              ROL   R2
3419 006426 006101              ROL   R1
3420 006430 006102              ROL   R2
3421 006432 006101              ROL   R1
3422 006434 006102              ROL   R2
3423 006436 062702 000060      ADD    #'0,R2   ;MAKE IT ASCII
3424 006442 010227              MOV    R2,(PC)+ ;SAVE IT
3425 006444 000000      4$:  .WORD 0
3426 006446 104401 006444      TYPE  ,4$      ;TYPE
3427 006452 005300      DEC   R0        ;ALL DIGITS TYPED?
3428 006454 003357              BGT   3$        ;NO -- BRANCH
3429 006456 104401 001215      TYPE  ,$CRLF
3430 006462 113737 001364 001115      MOVB  ERR.CT,$ERMAX ;SETUP MAX ERROR COUNT

```



3487  
3488  
3489  
3490  
3491  
3492  
3493  
3494  
3495  
3496  
3497  
3498  
3499  
3500  
3501  
3502  
3503  
3504  
3505  
3506  
3507  
3508  
3509  
3510  
3511  
3512  
3513  
3514  
3515  
3516  
3517  
3518  
3519  
3520  
3521  
3522  
3523  
3524  
3525  
3526  
3527  
3528  
3529  
3530  
3531  
3532  
3533  
3534  
3535  
3536  
3537  
3538  
3539  
3540  
3541  
3542

006470  
006470 000240  
006472 033737 001424 001234  
006500 001002  
006502 000137 006632  
006506 012737 000000 001102  
006514 004737 024726  
006520 012737 006614 001110  
006526 013777 001102 172406  
006534 013737 001506 001204  
006542 112737 000031 001115  
006550 112737 000107 004106  
006556 113737 001524 004134  
006564 113737 001516 004135  
006572 013737 001512 004136  
006600 012737 006630 001252  
006606 012737 006614 001106  
006614 012706 001100  
006620 004037 025150  
006624 004037 025262  
006630 000004

```

: * 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.
: *****
TST0:
NOP
BIT @#BITS+<0*2>,TSTNMS ;DO THIS TEST?
BNE 64$ ;YES--BRANCH
JMP TST1 ;NO--GO TO THE NEXT TEST
64$: MOV #0,@#TSTNM ;SET UP TEST NUMBER AND
;CLEAR THE ERROR FLAG ($ERFLG)
JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
MOV #TEST0,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
MOV $TSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
MOV @#RPT,$TIMES ;GET THE ITERATION COUNT
MOVB #25,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
MOVB #RECAL,@#DPB.A+2 ;RFCAL=COMMAND
MOVB @#FS,@#DPB.B+10 ;FS
MOVB @#FT,@#DPB.B+11 ;FT
MOV @#LC,@#DPB.B+12 ;LC
MOV #EXIT0,@#BYPASS ;GO TO EXIT0 ON ERROR
MOV #TEST0,$LPADR ;SETUP LOOP ADDRESS
TEST0: MOV #STACK,SP ;SET UP STACK POINTER
JSR R0,@#CALL.A ;GO EXECUTE THE COMMAND
JSR R0,@#CALL.B ;GO EXECUTE THE COMMAND
EXIT0: SCOPE ;LOOP
: *****
: *TEST 1 SEEK/SEEK TEST
: * THIS TEST WILL CAUSE THE DRIVE TO EXECUTE A FORWARD SEEK
: * CYCLE TO 'LC', 'LT', 'LS' FOLLOWED BY A REVERSE SEEK CYCLE TO
: * 'FC', 'FT', 'FS'. AT THE COMPLETION OF EACH SEEK, THE PROPER
: * INDICATORS ARE EXAMINED TO ENSURE PROPER OPERATION.
: * 'LC' WILL DEFAULT TO 128 AND 'FC', 'FT', 'LT', 'FS', AND 'LS'
: * WILL DEFAULT TO 0
: *****
TST1:
NOP
BIT @#BITS+<1*2>,TSTNMS ;DO THIS TEST?
BNE 64$ ;YES--BRANCH
JMP TST2 ;NO--GO TO THE NEXT TEST
64$: MOV #1,@#TSTNM ;SET UP TEST NUMBER AND
;CLEAR THE ERROR FLAG ($ERFLG)
JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
MOV #TEST1,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
MOV $TSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
MOV @#RPT,$TIMES ;GET THE ITERATION COUNT
MOVB #25,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
MOVB @#FS,@#DPB.B+10 ;FS
MOVB @#LS,@#DPB.C+10 ;LS
MOVB @#FT,@#DPB.B+11 ;FT

```

```

3543 006734 113737 001520 004155      MOV      @#LT,@#DPB.C+11 ;LT
3544 006742 013737 001510 004136      MOV      @#FC,@#DPB.B+12 ;FC
3545 006750 013737 001512 004156      MOV      @#LC,@#DPB.C+12 ;LC
3546 006756 012737 007006 001252      MOV      #EXIT1,@#BYPASS ;GO TO EXIT1 ON ERROR
3547 006764 012737 006772 001106      MOV      #TEST1,$LPADR ;SETUP LOOP ADDRESS
3548 006772 012706 001100      TEST1:  MOV      #STACK,SP ;SET THE STACK POINTER
3549 006776 004037 025452      JSR      RO,@#CALL.C ;GO EXECUTE THE COMMAND
3550 007002 004037 025262      JSR      RO,@#CALL.B ;GO EXECUTE THE COMMAND
3551 007006 000004      EXIT1:  SCOPE ;LOOP
3552
3553      ;:*****
3554      ;*TEST 2          INCREMENT/SEEK TEST
3555
3556      ;*
3557      ;* THIS TEST WILL COMMAND FORWARD SEEK CYCLES TO ADVANCE THE
3558      ;* CYLINDER ADDRESS FROM 'FC' TO 'LC' BY THE INCREMENT 'IC'.
3559      ;* WHEN THE RESULTANT CYLINDER ADDRESS (NC) EXCEEDS
3560      ;* 'LC' REVERSE SEEK CYCLES ARE INITIATED; STARTING
3561      ;* AT THE LAST LEGAL 'NC' AND DECREMENTING BY 'IC'
3562      ;* UNTIL 'NC' IS LESS THAN 'FC'. AT THE COMPLETION OF EACH
3563      ;* SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO
3564      ;* ENSURE PROPER OPERATION.
3565      ;:*****
3566      TST2:
3567      NOP
3568      BIT      @#BITS+<2*2> ,TSTNMS ;DO THIS TEST?
3569      BNE      64$ ;YES--BRANCH
3570      JMP      TST3 ;NO--GO TO THE NEXT TEST
3571      007026 012737 000002 001102 64$:  MOV      #2,@#TSTNM ;SET UP TEST NUMBER AND
3572      ;CLEAR THE ERROR FLAG ($ERFLG)
3573      JSR      PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
3574      MOV      #TEST2,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
3575      MOV      $TSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
3576      MOV      @#RPT,$TIMES ;GET THE ITERATION COUNT
3577      MOV      #25,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
3578      MOV      #1,$LPADR ;SETUP LOOP ADDRESS
3579      007076 113737 001524 004134 1$:  MOV      @#FS,@#DPB.B+10 ;FS
3580      MOV      @#FT,@#DPB.B+11 ;FT
3581      MOV      #EXIT2,@#BYPASS ;GO TO EXIT2 ON ERROR
3582      007120 013737 001510 004136 TEST2: MOV      @#FC,@#DPB.B+12 ;FC
3583      MOV      #,$SLPERR ;SETUP THE ERROR LOOP ADDRESS
3584      MOV      #STACK,SP ;LOAD THE STACK POINTER
3585      INCSK:
3586      JSR      RO,@#CALL.B ;GO EXECUTE THE COMMAND
3587      ADD      @#IC,@#DPB.B+12 ;MOVE TO NEXT CYLINDER
3588      CMP      @#LC,@#DPB.B+12 ;OUT OF CYLINDERS?
3589      BGE      INCSK ;NO--BRANCH
3590      MOV      @#LC,@#DPB.B+12
3591      MOV      #,$SLPERR ;SETUP THE ERROR LOOP ADDRESS
3592      MOV      #STACK,SP ;LOAD THE STACK POINTER
3593      DECSK:
3594      JSR      RO,@#CALL.B ;GO EXECUTE THE COMMAND
3595      SUB      @#IC,@#DPB.B+12
3596      CMP      @#FC,@#DPB.B+12
3597      BLE      DECSK
3598      EXIT2: SCOPE ;LOOP

```

3599  
3600  
3601  
3602  
3603  
3604  
3605  
3606  
3607  
3608  
3609  
3610  
3611  
3612  
3613  
3614  
3615  
3616  
3617  
3618  
3619  
3620

007226  
007226 000240  
007230 033737 001432 001234  
007236 001002  
007240 000137 007422  
007244 012737 000003 001102  
007252 004737 024726  
007256 012737 007336 001110  
007264 013777 001102 171650  
007272 013737 001506 001204  
007300 112737 000031 001115

.....  
: 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 ENSURE PROPER  
: OPERATION.  
:.....

TST3:  
NOP  
BIT @#BITS+<3\*2>,TSTNMS ;DO THIS TEST?  
BNE 648 ;YES--BRANCH  
JMP TST4 ;NO--GO TO THE NEXT TEST  
648: MOV #3,@#TSTNM ;SET UP TEST NUMBER AND  
;CLEAR THE ERROR FLAG (\$ERFLG)  
JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST  
MOV #TST3,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS  
MOV \$TSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER  
MOV @#RPT,\$TIMES ;GET THE ITERATION COUNT  
MOVB #25, \$ERMAX ;MAX ERRORS ALLOWED FOR TEST



```

3621 007306 012737 007314 001106      MOV      #18,$LPADR      ;SETUP TEST LOOP ADDRESS
3622 007314 113737 001524 004134 18:      MOV      @#FS,@#DPB.B+10 ;FS
3623 007322 113737 001516 004135      MOV      @#FT,@#DPB.B+11 ;FT
3624 007330 012737 007420 001252      MOV      #EXIT3,@#BYPASS ;GO TO BYPASS ON ERROR
3625 007336 013737 001510 004136 TEST3:  MOV      @#FC,@#DPB.B+12 ;FC
3626 007344 012737 007344 001110      MOV      #,$LPERR       ;SETUP THE ERROR LOOP ADDRESS
3627 007352 012706 001100      MOV      #STACK,SP      ;LOAD THE STACK POINTER
3628 007356 004037 025262      JSR      RO,@#CALL.B     ;GO EXECUTE THE COMMAND
3629 007362 013701 001514      MOV      IC,R1          ;CYLINDER 1
3630 007366 012737 007366 001110      MOV      #,$LPERR       ;SETUP THE ERROR LOOP ADDRESS
3631 007374 012706 001100      MOV      #STACK,SP      ;LOAD THE STACK POINTER
3632 007400 010137 004136 18:      MOV      R1,@#DPB.B+12  ;DESIRED CYLINDER
3633 007404 004037 025262      JSR      RO,@#CALL.B     ;GO EXECUTE THE COMMAND
3634 007410 006301      ASL      R1             ;MOVE TO NEXT CYLINDER
3635 007412 020137 001512      LMP      R1,@#LC        ;DONE?
3636 007416 003770      BLE     18             ;NO--LOOP
3637 007420 000004      EXIT3:  SCOPE
3638

```

3639  
3640  
3641  
3642  
3643  
3644  
3645  
3646  
3647  
3648  
3649  
3650  
3651  
3652  
3653  
3654  
3655  
3656  
3657  
3658  
3659  
3660  
3661  
3662  
3663  
3664  
3665  
3666  
3667  
3668  
3669  
3670  
3671  
3672  
3673  
3674  
3675  
3676  
3677  
3678  
3679  
3680  
3681  
3682  
3683  
3684  
3685  
3686  
3687  
3688  
3689  
3690  
3691  
3692  
3693  
3694

007422  
007422 000240  
007424 033737 001434 001234  
007432 001002  
007434 000137 010010  
007440 012737 000004 001102  
007446 004737 024726  
007452 012737 007546 001110  
007460 013777 001102 171454  
007466 013737 001506 001204  
007474 112737 000031 001115  
007502 012737 007510 001106  
007510 113737 001524 004134  
007516 113737 001516 004135  
007524 012737 010006 001252  
007532 005002  
007534 032737 010000 001220  
007542 001401  
007544 005102  
007546 013701 001510  
007552 005037 001336  
007556 012737 007556 001110  
007564 012706 001100  
007570 010137 004136  
007574 004037 025262  
007600 005702  
007602 001403  
007604 004037 026502  
007610 001336  
007612 013737 001510 004136  
007620 004037 025262  
007624 005702  
007626 001413  
007630 004037 026502  
007634 001336  
007636 005237 001336  
007642 023737 001362 001336  
007650 003347  
007652 005037 001336  
007656 063701 001514  
007662 020137 001512  
007666 003740  
007670 013701 001512  
007674 012737 007674 001110

```

:*****
: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.
:*****
TST4:
NOP
BIT    @#BITS+<4*2>,TSTNMS ;DO THIS TEST?
BNE    64$ ;YES--BRANCH
JMP    TST5 ;NO--GO TO THE NEXT TEST
64$:  MOV    #4,@#TSTNM ;SET UP TEST NUMBER AND
;CLEAR THE ERROR FLAG (SERFLG)
JSR    PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
MOV    #TST4,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
MOV    $TSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
MOV    @#RPT,$TIMES ;GET THE ITERATION COUNT
MOVB   #25,$SERMAX ;MAX ERRORS ALLOWED FOR TEST
MOV    #18,$LPADR ;SETUP LOOP ADDRESS
1$:  MOVB   @#FS,@#DPB.B+10 ;FS
MOVB   @#FT,@#DPB.B+11 ;FT
MOV    #EXIT4,@#BYPASS ;GO TO EXIT4 ON ERROR
CLR    R2 ;CLEAR STALL SWITCH (NO STALL)
BIT    #SW12,@#C.SWR ;STALL REQUIRED?
BEQ    TEST4 ;NO--BRANCH
COM    R2 ;YES--SET SWITCH
TEST4: MOV    @#FC,R1 ;SET NC TO FC
CLR    @#STALLO ;START AT ZERO IF STALLS REQUIRED
MOV    #,$SLPERR ;SETUP THE ERROR LOOP ADDRESS
MOV    #STACK,SP ;LOAD THE STACK POINTER
1$:  MOV    R1,@#DPB.B+12 ;NC
JSR    R0,@#CALL.B ;GO EXECUTE THE COMMAND
TST    R2 ;STALL?
BEQ    2$ ;NO--BRANCH
JSR    R0,@#STALL ;YES--GO TO STALL ROUTINE
WORD   STALLO ;TIME POINTER
2$:  MOV    FC,@#DPB.B+12 ;FC
JSR    R0,@#CALL.B ;GO EXECUTE THE COMMAND
TST    R2 ;STALL?
BEQ    3$ ;NO--BRANCH
JSR    R0,@#STALL ;YES--GO TO STALL ROUTINE
WORD   STALLO ;TIME POINTER
INC    @#STALL) ;UPDATE THE TIME
CMP    @#MXSTAL,@#STALLO ;TIME TO BIG?
BGT    1$ ;NO--BRANCH
CLR    @#STALLO ;YES--START OVER AT ZERO
ADD    @#IC,R1 ;MOVE TO NEXT CYLINDER
CMP    R1,@#LC ;LAST CYLINDER COMPLETED?
BLE    1$ ;NO--BRANCH
MOV    @#LC,R1 ;SET NC TO LC
MOV    #,$SLPERR ;SETUP THE ERROR LOOP ADDRESS

```

```

3695 007702 012706 001100      MOV    #STACK,SP      ;LOAD THE STACK POINTER
3696 007706 010137 004136      MOV    R1,@#DPB.B+12 ;NC
3697 007712 004037 025262      JSR    R0,@#CALL.B   ;GO EXECUTE THE COMMAND
3698 007716 005702              TST    R2             ;STALL?
3699 007720 001403              BEQ    5$            ;NO--BRANCH
3700 007722 004037 026502      JSR    R0,@#STALL    ;YES--GO TO STALL ROUTINE
3701 007726 001336              .WORD  STALLO        ;TIME POINTER
3702 007730 013737 001512 004136 5$:  MOV    @#LC,@#DPB.B+12 ;LC
3703 007736 004037 025262      JSR    R0,@#CALL.B   ;GO EXECUTE THE COMMAND
3704 007742 005702              TST    R2             ;STALL?
3705 007744 001413              BEQ    6$            ;NO--BRANCH
3706 007746 004037 026502      JSR    R0,@#STALL    ;YES--GO TO STALL ROUTINE
3707 007752 001336              .WORD  STALLO        ;TIME POINTER
3708 007754 005237 001336      INC    @#STALLO      ;UPDATE STALL TIME
3709 007760 023737 001362 001336  CMP    @#MXSTAL,@#STALLO ;TIME TOO BIG?
3710 007766 003347              BGT    4$            ;NO--BRANCH
3711 007770 005037 001336      CLR    @#STALLO      ;YES--SET STALL TIME BACK TO ZERO
3712 007774 163701 001514      SUB    @#IC,R1       ;NEXT CYLINDER
3713 010000 020137 001510      CMP    R1,@#FC       ;DONE?
3714 010004 002340              BGE    4$            ;NO--BRANCH
3715 010006 000004      EXIT4: SCOPE        ;LOOP
3716
3717
3718 ;*****
3719 ;*TEST 5          CONVERGING/DIVERGING SEEK TEST
3720
3721 ;*      THIS TEST WILL CAUSE THE DRIVE TO EXECUTE FORWARD AND REVERSE
3722 ;*      SEEKS FROM "NC1" AND "NC2" RESPECTIVELY, "NC1" WILL BE INCREMENTED
3723 ;*      BY "IC" AND "NC2" WILL BE DECREMENTED BY "IC" UNTIL "NC1" IS
3724 ;*      GREATER THAN THE INITIAL VALUE OF "NC2" AND "NC2" IS
3725 ;*      LESS THAN THE INITIAL VALUE OF "NC1".  AT THE COMPLETION OF
3726 ;*      EACH SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO
3727 ;*      ENSURE PROPER OPERATION.  "NC1" AND "NC2" DEFAULT TO
3728 ;*      "FC" AND "LC" RESPECTIVELY.
3729 ;*****
3730 010010      TST5:
3731 010010 000240      NOP
3732 010012 033737 001436 001234  BIT    @#BITS+<5*2>,TSTNMS ;DO THIS TEST?
3733 010020 001002      BNE    64$          ;YES--BRANCH
3734 010022 000137 010210      JMP    TST6         ;NO--GO TO THE NEXT TEST
3735 010026 012737 000005 001102 64$:  MOV    #5,@#TSTNM   ;SET UP TEST NUMBER AND
3736                          ;CLEAR THE ERROR FLAG ($ERFLG)
3737 010034 004737 024726      JSR    PC,LODPRM    ;LOAD THE PARMETERS FOR THE TEST
3738 010040 012737 010120 001110  MOV    #TST5,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
3739 010046 013777 001102 171066  MOV    $TSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
3740 010054 013737 001506 001204  MOV    @#RPT,$TIMES  ;GET THE ITERATION COUNT
3741 010062 112737 000031 001115  MOVB   #25,,$ERMAX   ;MAX ERRORS ALLOWED FOR TEST
3742 010070 012737 010076 001106  MOV    #1,$LPADR     ;SETUP LOOP ADDRESS
3743 010076 113737 001524 004134 1$:   MOVB   @#FS,@#DPB.B+10 ;FS
3744 010104 113737 001516 004135  MOVB   @#FT,@#DPB.B+11 ;FT
3745 010112 012737 010206 001252  MOV    #EXIT5,@#BYPASS ;GO TO EXITS ON ERROR
3746 010120 013701 001510      TEST5: MOV    @#FC,R1     ;START NC1 AT FC
3747 010124 013702 001512  MOV    @#LC,R2       ;START NC2 AT LC
3748 010130 012737 010130 001110  MOV    #,,$LPERR     ;SETUP THE ERROR LOOP ADDRESS
3749 010136 012706 001100      MOV    #STACK,SP    ;LOAD THE STACK POINTER
3750 010142 010137 004136      1$:   MOV    R1,@#DPB.B+12 ;NC1
    
```

```
3751 010146 004037 025262 JSR R0,@CALL.B ;GO EXECUTE THE COMMAND
3752 010152 010237 004136 MOV R2,@DPB.B+12 ;NC2
3753 010156 004037 025262 JSR R0,@CALL.B ;GO EXECUTE THE COMMAND
3754 010162 063701 001514 ADD @#IC,R1 ;NEXT NC1
3755 010166 163702 001514 SUB @#IC,R2 ;NEXT NC2
3756 010172 020137 001512 CMP R1,@#LC ;DONE?
3757 010176 003003 BGT EXIT5 ;YES--BRANCH
3758 010200 020237 001510 CMP R2,@#FC ;?
3759 010204 002356 BGE 1$ ;NO--BRANCH
3760 010206 000004 EXIT5: SCOPE ;LOOP
```

```
*****
;*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.
*****
```

```
3772
3773 010210 TST6:
3774 010210 000240 NOP
3775 010212 033737 001440 001234 BIT @#BITS+<6*2>,TSTNMS ;DO THIS TEST?
3776 010220 001002 BNE 64$ ;YES--BRANCH
3777 010222 000137 010454 JMP TST7 ;NO--GO TO THE NEXT TEST
3778 010226 012737 000006 001102 64$: MOV #6,@#TSTNM ;SET UP TEST NUMBER AND
3779 ;CLEAR THE ERROR FLAG ($ERFLG)
3780 010234 004737 024726 JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
3781 010240 012737 010320 001110 MOV #TEST6,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
3782 010246 013777 001102 170666 MOV $TSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
3783 010254 013737 001506 001204 MOV @#RPT,$TIMES ;GET THE ITERATION COUNT
3784 010262 112737 000031 001115 MOVB #25,,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
3785 010270 012737 010276 001106 MOV #1$, $LPADR ;SETUP LOOP ADDRESS
3786 010276 113737 001524 004134 1$: MOVB @#FS,@#DPB.B+10 ;FS
3787 010304 113737 001516 004135 MOVB @#FT,@#DPB.B+11 ;FT
3788 010312 012737 010452 001252 MOV #EXIT6,@#BYPASS ;GO TO EXIT6 ON ERROR
3789 010320 013701 001510 TEST6: MOV @#FC,R1 ;PICKUP 'FC'
3790 010324 013702 001512 MOV @#LC,R2 ;FORM LAST CYLINDER THAT
3791 010330 162702 000005 SUB #5,R2 ;IS AVAILABLE FOR TESTING
3792 010334 012737 010334 001110 MOV #, $LPERR ;SETUP THE ERROR LOOP ADDRESS
3793 010342 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
3794 010346 020102 1$: CMP R1,R2 ;LAST CYLINDER
3795 010350 003040 BGT EXIT6 ;YES--BRANCH
3796 010352 010137 004136 MOV R1,@#DPB.B+12 ;NC
3797 010356 004037 025262 JSR R0,@CALL.B ;GO EXECUTE THE COMMAND
3798 010362 062737 000004 004136 ADD #4,@#DPB.B+12 ;NC+4
3799 010370 004037 025262 JSR R0,@CALL.B ;GO EXECUTE THE COMMAND
3800 010374 162737 000003 004136 SUB #3,@#DPB.B+12 ;NC+1
3801 010402 004037 025262 JSR R0,@CALL.B ;GO EXECUTE THE COMMAND
3802 010406 062737 000002 004136 ADD #2,@#DPB.B+12 ;NC+3
3803 010414 004037 025262 JSR R0,@CALL.B ;GO EXECUTE THE COMMAND
3804 010420 162737 000001 004136 SUB #1,@#DPB.B+12 ;NC+2
3805 010426 004037 025262 JSR R0,@CALL.B ;GO EXECUTE THE COMMAND
3806 010432 062737 000003 004136 ADD #3,@#DPB.B+12 ;NC+5
```

3807 010440 004037 025262  
3808 010444 063701 001514  
3809 010450 000736  
3810 010452 000004

JSR RO,@#CALL.B ;GO EXECUTE THE COMMAND  
ADD @#IC,R1  
BR 1\$  
EXIT6: SCOPE ;LOOP

3811  
3812  
3813  
3814  
3815  
3816  
3817  
3818  
3819  
3820  
3821  
3822

\*\*\*\*\*  
: \*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'.  
\*\*\*\*\*

3823 010454  
3824 010454 000240  
3825 010456 033737 001442 001234  
3826 010464 001002  
3827 010466 000137 011034  
3828 010472 012737 000007 001102  
3829  
3830 010500 004737 024726  
3831 010504 012737 010576 001110  
3832 010512 013777 001102 170422  
3833 010520 013737 001506 001204  
3834 010526 112737 000031 001115  
3835 010534 113737 001516 004135  
3836 010542 112737 000105 004106  
3837 010550 112737 000173 004126  
3838 010556 013704 034632  
3839 010562 012737 011032 001252  
3840 010570 012737 010576 001106  
3841 010576 012706 001100  
3842 010602 013737 001510 004136  
3843 010610 023737 001510 001512  
3844 010616 001422  
3845 010620 004737 023626  
3846 010624 013746 023724  
3847 010630 005046  
3848 010632 013746 001512  
3849 010636 005216  
3850 010640 163716 001510  
3851 010644 004737 023730  
3852 010650 062637 004136  
3853 010654 005726  
3854 010656 013737 004136 004116  
3855 010664  
3856 010664 012737 010664 001110  
3857 010672 012706 001100  
3858 010676 004037 025150  
3859 010702 012737 010702 001110  
3860 010710 012706 001100  
3861 010714 113764 004104 000010  
3862 010722 016446 000020

TST7:  
NOP  
BIT @#BITS+<7\*2>,TSTNMS ;DO THIS TEST?  
BNE 64\$ ;YES--BRANCH  
JMP TST10 ;NO--GO TO THE NEXT TEST  
64\$: MOV #7,@#TSTNM ;SET UP TEST NUMBER AND  
;CLEAR THE ERROR FLAG (SERFLG)  
JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST  
MOV #TST7,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS  
MOV \$TSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER  
MOV @#RPT,\$TIMES ;GET THE ITERATION COUNT  
MOVB #25, \$ERMAX ;MAX ERRORS ALLOWED FOR TEST  
MOVB FT,DPB.B+11 ;LOAD STARTING TRACK ADDRESS  
MOVB #SEEK,@#DPB.A+2 ;SEEK=COMMAND  
MOVB #READHD,DPB.B+2 ;READ HEADER & DATA COMMAND  
MOV RPADR,R4 ;UNIBUS ADDRESS OF THE RH11  
MOV #EXIT7,BYPASS ;ERROR TERMINATION ADDRESS  
MOV #TST7,\$LPADR ;SETUP THE LOOP ON TEST ADDRESS  
TEST7: MOV #STACK,SP ;SETUP THE STACK POINTER  
MOV FC,DPB.B+12 ;INITIAL CYLINDER ADDRESS  
CMP FC,LC ;CYLINDER LIMITS THE SAME ?  
BEQ 1\$ ;BR IF THEY ARE  
JSR PC,\$RAND ;CYCLE THE RANDOM NUMBER GENERATOR  
MOV \$HINUM,-(SP) ;USE THE HIGH RANDOM NUMBER  
CLR -(SP) ;UPPER DIVIDEND  
MOV LC,-(SP) ;FORM THE DIVISOR  
INC (SP) ;INCREMENT  
SUB FC,(SP) ;SUBTRACT THE LOWER LIMIT  
JSR PC,\$DIV ;DIVIDE  
ADD (SP)+,DPB.B+12 ;ADD THE REMAINDER TO THE INITIAL CYLINDER  
TST (SP)+ ;DISCARD THE QUOTIENT  
MOV DPB.B+12,DPB.A+12 ;COPY NEW CYLINDER ADDRESS  
1\$: MOV #,\$LPERR ;SETUP THE ERROR LOOP ADDRESS  
MOV #STACK,SP ;LOAD THE STACK POINTER  
JSR RO,@#CALL.A ;GO EXECUTE THE COMMAND  
MOV #,\$LPERR ;SETUP THE ERROR LOOP ADDRESS  
MOV #STACK,SP ;LOAD THE STACK POINTER  
MOVB DPB.A,RPCS2(R4) ;SELECT THE DRIVE  
MOV RPLA(R4),-(SP) ;GET THE LOOK AHEAD REGISTER

3863	010726	006316		ASL	(SP)		;ALIGN THE SECTOR ADDRESS
3864	010730	006316		ASL	(SP)		;ALIGN THE SECTOR ADDRESS
3865	010732	000316		SWAB	(SP)		;PUT ADDRESS IN LOWER BYTE
3866	010734	105766	000001	TSTB	1(SP)		;IN THE 1ST 20% OF SECTOR ?
3867	010740	001401		BEQ	2\$		;BR IF YES
3868	010742	105216		INCB	(SP)		;INCREMENT THE SECTOR ADDRESS
3869	010744	105216		INCB	(SP)	2\$:	;INCREMENT THE SECTOR ADDRESS
3870	010746	112637	004174	MOVB	(SP)+,DTADPB+10		;LOAD THE DPB
3871	010752	013746	001630	MOV	PRMLT+22,-(SP)		;PUT LAST SECTOR ADDRESS ON THE STACK
3872	010756	005216		INC	(SP)		;INCREMENT IT
3873	010760	122637	004174	CMPB	(SP)+,DTADPB+10		;NEW SECTOR ADDRESS TOO LARGE ?
3874	010764	103007		BHIS	4\$		;BR IF NOT
3875	010766	103403		BLO	3\$		;BR IF ADDRESS IS 2 GREATER
3876	010770	105037	004174	CLRB	DTADPB+10		;RESET TO SECTOR ADDRESS 0
3877	010774	000403		BR	4\$		;CONTINUE
3878	010776	112737	000001 004174	MOVB	#1,DTADPB+10	3\$:	;RESET ADDRESS TO SECTOR 1
3879	011004					4\$:	
3880	011004	004037	025262	JSR	RO,#CALL.B		;GO EXECUTE THE COMMAND
3881	011010	105237	004135	INCB	DPB.B+11		;INCREMENT THE TRACK ADDRESS
3882	011014	123737	004135 001520	CMPB	DPB.B+11,LT		;MAXIMUM ?
3883	011022	101403		BLOS	EXIT7		;BR IF NOT
3884	011024	113737	001516 004135	MOVB	FT,DPB.B+11		;RELOAD STARTING TRACK ADDRESS
3885	011032	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 'NC +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 (RPLA) TO DETERMINE THE ADDRESS OF THE SECTOR ROTATING INTO  
 ;\* POSITION. THE PROGRAM THEN ISSUES A WRITE HEADER AND DATA COMMAND  
 ;\* FOR THAT SECTOR. IF THE DRIVE'S POSITIONER HAS NOT SETTLED DOWN OR  
 ;\* IF THE POSITIONER IS NOT ON CYLINDER (IF THE DRIVE IS AN RPO4, THE  
 ;\* OFF CYLINDER CONDITION MUST LAST FOR AT LEAST 800 US), THE DRIVE  
 ;\* WILL REPORT A 'WRU' ERROR. (RPO5/6'S MAY ALSO REPORT 'NHS' ERROR UNDER  
 ;\* 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 USES THE EXTENTION BITS IN THE LOOK-AHEAD REGISTER TO DETERMINE  
 ;\* WHETHER OR NOT IT CAN PICK UP THE SECTOR ROTATING INTO POSITION. THE  
 ;\* TEST IS OPTIMIZED SUCH THAT IF THE DRIVE SIGNALS SEEK DONE WITHIN  
 ;\* THE FIRST 80% OF THE SECTOR CURRENTLY UNDER THE HEAD, THE TEST WILL  
 ;\* TRY TO ADDRESS THE NEXT SECTOR. BASED ON OBSERVATION, THE PPROGRAM  
 ;\* IS ABLE TO START THE OPERATION WITHOUT LOSING A REVOLUTION MOST OF  
 ;\* THE TIME.

;\* THIS TEST IS VALID ONLY IF THE OPERATION IS STARTED WITHIN A FEW  
 ;\* HUNDRED MICRO-SECONDS AFTER SEEK DONE OCCURS. THE NECESSARY TIME

3886  
 3887  
 3888  
 3889  
 3890  
 3891  
 3892  
 3893  
 3894  
 3895  
 3896  
 3897  
 3898  
 3899  
 3900  
 3901  
 3902  
 3903  
 3904  
 3905  
 3906  
 3907  
 3908  
 3909  
 3910  
 3911  
 3912  
 3913  
 3914  
 3915  
 3916  
 3917  
 3918

3919 :\* TIME DEPENDENT PARAMETERS OCCUR FREQUENTLY ENOUGH WITHIN THE REQUIRED  
3920 :\* RANGE TO PERMIT THIS TEST TO BE EFFECTIVE.

3921 :  
3922 :\*\*\*\*\*

```

3923 011034 TST10:
3924 011034 000240 NOP
3925 011036 033737 001444 001234 BIT @#BITS+<10*2>,TSTNMS ;DO THIS TEST?
3926 011044 001002 BNE 64$ ;YES--BRANCH
3927 011046 000137 012140 JMP TST11 ;NO--GO TO THE NEXT TEST
3928 011052 012737 000010 001102 64$: MOV #10,@#STSTNM ;SET UP TEST NUMBER AND
3929 ;CLEAR THE ERROR FLAG ($ERFLG)
3930 011060 004737 024726 JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
3931 011064 012737 011242 001110 MOV #TEST10,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
3932 011072 013777 001102 170042 MOV $TSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
3933 011100 013737 001506 001204 MOV @#RPT,$TIMES ;GET THE ITERATION COUNT
3934 011106 112737 000031 001115 MOV #25, $ERMAX ;MAX ERRORS ALLOWED FOR TEST
3935 011114 012737 011122 001106 MOV #1$, $LPADR ;SETUP THE LOOP ADDRESS
3936 011122 1$:
3937 011122 112737 000105 004106 MOV #SEEK,@#DPB.A+2 ;SEEK=COMMAND
3938 011130 112737 000161 004166 MOV #WRITE,DTADPB+2 ;COMMAND
3939 011136 113737 001516 004175 MOV FT,DTADPB+11 ;TRACK ADDRESS FOR THE WRITE
3940 011144 013737 001510 004116 MOV FC,DPB.A+12 ;CYLINDER ADDRESS FOR THE SEEK
3941 011152 013737 001510 004176 MOV FC,DTADPB+12 ;CYLINDER ADDRESS FOR THE WRITE
3942 011160 013737 001510 001532 MOV FC,NC1 ;STARTING CYLINDER
3943 011166 013737 001514 001350 MOV IC,DELTA ;CYLINDER INCREMENT VALUE
3944 011174 012737 176000 004170 MOV #-<256.*4.>,DTADPB+4 ;WORD COUNT
3945 011202 012737 050202 004172 MOV #BUFFER,DTADPB+6 ;BUFFER ADDRESS
3946 011210 005000 CLR R0 ;PATTERN POINTER (WC PATTERN)
3947 011212 004737 030476 JSR PC,SETBUF ;LOAD THE WRITE BUFFER
3948 011216 005001 CLR R1 ;CLEAR REGISTER
3949 011220 113701 004104 MOV DPB.A,R1 ;LOAD DRIVE ADDRESS
3950 011224 013704 034632 MOV RPADR,R4 ;UNIBUS ADDRESS OF THE RH11
3951 011230 004737 042664 JSR PC,CLRQUE ;CLEAR THE OPERATION QUEUES
3952 011234 012737 012136 001252 MOV #EXIT10,BYPASS ;ERROR EXIT FROM TEST
3953 011242 TEST10:
3954 011242 012737 011242 001110 MOV #, $LPERR ;SETUP THE ERROR LOOP ADDRESS
3955 011250 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
3956 011254 012737 000340 177776 MOV #PR7,@#PS ;SET PRIORITY TO 7
3957 011262 005737 001244 TST CLKSTA ;SEE WHICH CLOCK ON SYSTEM
3958 011266 001415 BEQ 3$ ;BR IF NO CLOCK
3959 011270 100405 BMI 1$ ;BR IF KW11-L CLOCK
3960 011272 017746 170076 MOV @PKV,-(SP) ;SAVE THE VECTOR
3961 011276 013746 001374 MOV PKV,-(SP) ;SAVE THE VECTOR ADDRESS
3962 011302 000404 BR 2$ ;CONTINUE
3963 011304 017746 170076 1$: MOV @LKV,-(SP) ;SAVE THE 'L' CLOCK VECTOR
3964 011310 013746 001406 MOV LKV,-(SP) ;SAVE THE VECTOR ADDRESS
3965 011314 012776 012072 000000 2$: MOV #TST10B,@(SP) ;CHANGE THE VECTOR
3966 011322 012777 027170 023304 3$: MOV #DORT1,@RVEC ;CHANGE THE RPO4/RPO5 VECTOR
3967 011330 012737 000010 001344 MOV #8, SEKTR ;LOAD THE SEEK TIMER
3968 011336 012764 000040 000010 MOV #BIT05,RPCS2(R4) ;INIT THE MASSBUS
3969 011344 110164 000010 MOV R1,RPCS2(R4) ;RESELECT THE DRIVE
3970 011350 013764 004116 000034 MOV DPB.A+12,RPCA(R4) ;LOAD THE CYLINDER ADDRESS
3971 011356 013737 004116 001270 MOV DPB.A+12,CYL.DS ;CYLINDER ADDRESS FOR ERROR MESSAGE
3972 011364 112764 000105 000000 MOV #SEEK,RPCS1(R4) ;START THE SEEK
3973 011372 005037 177776 CLR @#PS ;CLEAR THE PRIORITY
3974 011376 105764 000012 4$: TSTB RPDS1(R4) ;HAS THE DRIVE FINISHED ?

```

```

3975 011402 100402      BMI      5$      ;BR IF IT HAS
3976 011404 00000i     WAIT      ;WAIT FOR THE OPERATION TO COMPLETE
3977 011406 000773     BR        4$      ;CONTINUE
3978 011410 012737 000340 177776 5$:  MOV      #PR7,@#PS  ;CHANGE PRIORITY TO MAX
3979 011416 032764 040000 000012  BIT      #BIT14,RPDS1(R4) ;ERROR ?
3980 011424 001412     BEQ      6$      ;BR IF NOT
3981 011426 012702 004104     MOV      #DPB.A,R2  ;DPB POINTER
3982 011432 004737 042202     JSR      PC,SVRH11  ;SAVE THE REGISTERS
3983 011436 104023     ERROR    23      ;ERROR DURING SEEK
3984 011440 012764 000040 000010  MOV      #BIT05,RPCS2(R4) ;INIT THE MASSBUS
3985 011446 110164 000010     MOV      R1,RPCS2(R4) ;RESELECT THE DRIVE
3986 011452 012777 037544 023154 6$:  MOV      #ISR,@RPVEC  ;SETUP THE RPO4/RPO5 VECTOR
3987 011460 005737 001244     TST      CLKSTA    ;WHICH CLOCK
3988 011464 001405     BEQ      TST10A   ;BR IF NONE
3989 011466 016676 000002 000000  MOV      2(SP),@ (SP) ;RELOAD THE CLOCK VECTOR
3990 011474 062706 000004     ADD      #4,SP    ;CORRECT THE STACK POINTER
3991 011500     TST10A:
3992 011500 012737 011500 001110  MOV      #.,$LPERR  ;SETUP THE ERROR LOOP ADDRESS
3993 011506 012706 001100     MOV      #STACK,SP ;LOAD THE STACK POINTER
3994 011512 110164 000010     MOV      R1,RPCS2(R4) ;SELECT THE DRIVE
3995 011516 016446 000020     MOV      RPLA(R4),-(SP) ;GET THE LOOK AHEAD REGISTER
3996 011522 006316     ASL      (SP)      ;ALIGN THE SECTOR ADDRESS
3997 011524 006316     ASL      (SP)      ;ALIGN THE SECTOR ADDRESS
3998 011526 000316     SWAB     (SP)      ;PUT ADDRESS IN LOWER BYTE
3999 011530 122766 000300 000001  CMPB     #300,1(SP) ;IN THE LAST 20% OR SECTOR ?
4000 011536 001001     BNE      2$      ;BR IF NOT
4001 011540 105216     INCB     (SP)      ;INCREMENT THE SECTOR ADDRESS
4002 011542 105216     INCB     (SP)      ;INCREMENT THE SECTOR ADDRESS
4003 011544 112637 004174     MOV      (SP)+,DTADPB+10 ;LOAD THE DPB
4004 011550 013746 001630     MOV      PRMLMT+22,-(SP) ;PUT MAXIMUM SECTOR ADDRESS ON THE STACK
4005 011554 005216     INC      (SP)      ;INCREMENT PAST THE MAXIMUM ADDRESS
4006 011556 122637 004174     CMPB     (SP)+,DTADPB+10 ;NEW SECTOR ADDRESS TOO LARGE ?
4007 011562 101007     BHI      4$      ;BR IF NOT
4008 011564 103403     BLO      3$      ;BR IF ADDRESS IS 2 GREATER THAN MAXIMUM
4009 011566 105037 004174     CLRB     DTADPB+10 ;RESET TO SECTOR ADDRESS 0
4010 011572 000403     BR       4$      ;CONTINUE
4011 011574 112737 000001 004174 3$:  MOV      #1,DTADPB+10 ;RESET ADDRESS TO SECTOR 1
4012 011602 012703 004170     4$:  MOV      #DTADPB+4,R3 ;POINTER
4013 011606 012764 000111 000000  MOV      #DRVCLR,RPCS1(R4) ;CLEAR THE DRIVE
4014 011614 012364 000002     MOV      (R3)+,RPWC(R4) ;LOAD THE WORD COUNT
4015 011620 012364 000004     MOV      (R3)+,RPBA(R4) ;LOAD THE BUFFER ADDRESS
4016 011624 012364 000006     MOV      (R3)+,RPDA(R4) ;LOAD THE TRACK/SECTOR ADDR
4017 011630 005037 004202     CLR      DTADPB+16 ;RESET 'DONE' INDICATOR
4018 011634 012737 004164 034544  MOV      #DTADPB,TRNSWT ;LOAD 'TRANSFER' DPB ADDRESS
4019 011642 010137 034616     MOV      R1,DTUW   ;ADDRESS OF DRIVE TRANSFERING
4020 011646 112761 000001 034474  MOV      #1,DRVACT(R1) ;SET DRIVE ACTIVE INDICATOR
4021 011654 006301     ASL      R1       ;SHIFT DRIVE ADDRESS
4022 011656 012761 001750 034576  MOV      #1000.,TIMER(R1) ;SETUP THE OPERATION TIMER
4023 011664 006201     ASR      R1       ;RESTORE R1
4024 011666 013764 004166 000000  MOV      DTADPB+2,RPCS1(R4) ;START THE OPERATION
4025 011674 005037 177776     CLR      @#PS     ;CLEAR THE PRIORITY
4026 011700 004037 025662     JSR      RO,DRVCL1 ;WAIT FOR OPERATION TO COMPLETE
4027 011704 023727 001346 001750 5$:  CMP      SEKCNT,#1000. ;FINISHED SEEKS ?
4028 011712 001026     BNE      6$      ;BR IF NOT
4029 011714 005037 001346     CLR      SEKCNT   ;CLEAR THE SEEK COUNT
4030 011720 063737 001514 001532  ADD      IC,NC1   ;ADD THE INCREMENT

```





```

4087 012164 004737 024726 JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
4088 012170 012737 012300 001110 MOV #TEST11,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
4089 012176 013777 001102 166736 MOV $STINM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
4090 012204 013737 001506 001204 MOV @#RPT,$TIMES ;GET THE ITERATION COUNT
4091 012212 112737 000031 001115 MOV #25, $SERMAX ;MAX ERRORS ALLOWED FOR TEST
4092 012220 012737 012226 001106 MOV #1$, $LPADR ;SETUP THE LOOP ADDRESS
4093 012226 113737 001524 004134 1$: MOVB FS,DPB.B+10 ;SECTOR ADDRESS
4094 012234 113737 001524 004154 MOVB FS,DPB.C+10 ;SECTOR ADDRESS
4095 012242 113737 001516 004135 MOVB FT,DPB.B+11 ;TRACK ADDRESS
4096 012250 113737 001516 004155 MOVB FT,DPB.C+11 ;TRACK ADDRESS
4097 012256 013737 001510 004136 MOV FC,DPB.B+12 ;STARTING CYLINDER ADDRESS
4098 012264 013737 001510 004156 MOV FC,DPB.C+12 ;STARTING CYLINDER ADDRESS
4099 012272 012737 012356 001252 MOV #EXIT11,BYPASS ;TEST ABORT EXIT
4100 012300 012706 001100 TEST11: MOV #STACK,SP ;SETUP THE STACK POINTER
4101 012304 1$:
4102 012304 004037 025452 JSR RO,@#CALL.C ;GO EXECUTE THE COMMAND
4103 012310 004037 025262 JSR RO,@#CALL.B ;GO EXECUTE THE COMMAND
4104 012314 063737 001514 004156 ADD IC,DPB.C+12 ;INCREMENT THE ENDING CYLINDER ADDRESS
4105 012322 023737 001512 004156 CMP LC,DPB.C+12 ;CHECK IF EXCEEDING MAXIMUM
4106 012330 002365 BGE 1$ ;BR IF NOT
4107 012332 013737 001510 004156 MOV FC,DPB.C+12 ;RESET ENDING CYLINDER ADDRESS
4108 012340 063737 001514 004136 ADD IC,DPB.B+12 ;INCREMENT THE STARTING ADDRESS
4109 012346 023737 001512 004136 CMP LC,DPB.B+12 ;EXCEEDING MAXIMUM ?
4110 012354 002353 BGE 1$ ;BR IF NOT
4111 012356 000004 EXIT11: SCOPE ;LOOP ?
4112
    
```



4169	012516	012737	013114	001206		MOV	#EXIT12,\$ESCAPE	;;ESCAPE TO EXIT12 ON ERROR
4170	012524	005005				CLR	R5	;;COUNT UP
4171	012526	012703	002774			MOV	#T7A,R3	;;60HZ PARAMETERS
4172	012532	032737	000100	001220		BIT	#SW06,@#C.SWR	;;60 HZ?
4173	012540	001402				BEQ	TEST12	;;YES--BRANCH
4174	012542	012703	003004			MOV	#T7B,R3	;;NO--50 HZ PARAMETERS
4175	012546	012706	001100		TEST12:	MOV	#STACK,SP	;;SETUP STACK
4176	012552	012701	000012			MOV	#10.,R1	;;TIME 10 SEARCHES
4177	012556	004737	027172			JSR	PC,@#STRMR	;;INITIALIZE THE TIMERS
4178	012562	012777	012770	166604		MOV	#7\$,@PKV	;;SETUP VECTOR IN CASE OF OVERFLOW
4179	012570	012777	027170	022036		MOV	#DORT1,@RPVEC	;;SETUP RP04/5/6 VECTOR
4180	012576	005077	166600		1\$:	CLR	@PKB	;;START COUNTING AT ZERO
4181	012602	012777	000131	166570		MOV	#131,@PKCS	;;INT.EN., COUNT UP AT 100KHZ
4182	012610	012714	000131			MOV	#SEARCH,(R4)	;;START A SEARCH
4183	012614	000001				WAIT		;;WAIT ON INTERRUPT
4184	012616	042777	000101	166554		BIC	#101,@PKCS	;;STOP THE CLOCK
4185	012624	032764	040000	000012		BIT	#BIT14,RPDS1(R4)	;;ERROR?
4186	012632	001415				BEQ	2\$	;;NO--BRANCH
4187	012634	104412				SAVREG		;;SAVE R0-R5
4188	012636	012702	004164			MOV	#DTADPB,R2	;;DPB POINTER
4189	012642	004737	042202			JSR	PC,@#SVRH11	;;SAVE ALL THE RH11/RP04 REGISTERS
4190	012646	012764	000040	000010		MOV	#BIT05,RPCS2(R4)	;;MASSBUS CLEAR
4191	012654	013764	004164	000010		MOV	@#DTADPB,RPCS2(R4)	;;SELECT DRIVE
4192	012662	104413				RESREG		;;RESTORE R0-R5
4193	012664	104017				ERROR	17	
4194	012666	005077	166510		2\$:	CLR	@PKB	;;START THE COUNT AT ZERO
4195	012672	012714	000131			MOV	#SEARCH,(R4)	;;START A SEARCH
4196	012676	012777	000131	166474		MOV	#131,@PKCS	;;START THE CLOCK
4197	012704	000001				WAIT		;;WAIT ON INTERRUPT
4198	012706	042777	000101	166464		BIC	#101,@PKCS	;;STOP THE CLOCK
4199	012714	032764	040000	000012		BIT	#BIT14,RPDS1(R4)	;;IS 'ERR=1'?
4200	012722	001415				BEQ	3\$	;;NO--BRANCH
4201	012724	104412				SAVREG		;;SAVE R0-R5
4202	012726	012702	004164			MOV	#DTADPB,R2	;;DPB POINTER
4203	012732	004737	042202			JSR	PC,@#SVRH11	;;SAVE ALL THE RH11/RP04 REGISTERS
4204	012736	012764	000040	000010		MOV	#BIT05,RPCS2(R4)	;;MASSBUS CLEAR
4205	012744	013764	004164	000010		MOV	@#DTADPB,RPCS2(R4)	;;SELECT DRIVE
4206	012752	104413				RESREG		;;RESTORE R0-R5
4207	012754	104017				ERROR	17	;;DISK ERROR OCCURRED
4208	012756	004737	027236		3\$:	JSR	PC,@#COUNT	;;UPDATE THE COUNT
4209	012762	005301				DEC	R1	;;DONE?
4210	012764	003304				BGT	1\$	;;NO--BRANCH
4211	012766	000424				BR	8\$	;;YES--GO TO THE EXIT
4212	012770	042777	000101	166402	7\$:	BIC	#101,@PKCS	;;STOP THE CLOCK
4213	012776	005037	177776			CLR	@#PS	;;DROP THE PRIORITY
4214	013002	012600				MOV	(SP)+,R0	;;PC OF WAIT+2
4215	013004	005726				TST	(SP)+	;;POP THE PS FROM THE STACK
4216	013006	104412				SAVREG		;;SAVE R0-R5
4217	013010	012702	004164			MOV	#DTADPB,R2	;;DPB POINTER
4218	013014	004737	042202			JSR	PC,@#SVRH11	;;SAVE ALL THE RH11/RP04 REGISTERS
4219	013020	012764	000040	000010		MOV	#BIT05,RPCS2(R4)	;;MASSBUS CLEAR
4220	013026	013764	004164	000010		MOV	@#DTADPB,RPCS2(R4)	;;SELECT DRIVE
4221	013034	104413				RESREG		;;RESTORE R0-R5
4222	013036	104020				ERROR	20	;;CLOCK OVERFLOWED
4223	013040				8\$:			
4224	013040	012764	000040	000010		MOV	#BIT05,RPCS2(R4)	;;MASSBUS INIT.

```

4225 013046 013764 004164 000010      MOV    @#DTADPB,RPCS2(R4) ;SELECT DRIVE
4226 013054 004737 024212              JSR    PC,@#ST.CLK       ;INITIALIZE THE CLOCK
4227 013060 012777 037544 021546      MOV    #ISR,@RPVEC      ;RESTORE RH11/RP04/5/6 INT. VECTOR
4228 013066 032737 000100 001220      BIT    #SW06,@#C.SWR    ;60 HZ?
4229 013074 001004              BNF    9$               ;NO -- BRANCH
4230 013076 004037 027370              JSR    RO,@#TYPTIM      ;GO TYPE THE TIMES
4231 013102 002774              T7A                      ;POINTER
4232 013104 000403              BR     EXIT12           ;GO TO EXIT
4233 013106                      9$:
4234 013106 004037 027370              JSR    RO,@#TYPTIM      ;GO TYPE THE TIMES
4235 013112 003004              T7B                      ;POINTER
4236 013114 000004      EXIT12: SCOPE           ;LOOP ?
4237
4238
4239
4240      ;*****
4241      ;*TEST 13      ONE CYLINDER SEEK TIMING TEST
4242
4243      ;*      THIS TEST WILL COMMAND FORWARD SEEK CYCLES TO ADVANCE THE
4244      ;*      CYLINDER BY ONE UNTIL THE INCREMENT IS GREATER THAN THE
4245      ;*      CYLINDER 'LC', THEN REVERSE SEEK TO CYLINDER 'FC'. THE
4246      ;*      TIME TO PERFORM EACH SEEK IS CHECKED TO ENSURE IT DOES NOT
4247      ;*      EXCEED THE MAXIMUM TIME PERMITTED FOR A ONE CYLINDER SEEK.
4248      ;*      THE TIME MUST BE LESS THAN 10MS.
4249      ;*****
4250      TST13:
4251 013116 000240      NOP
4252 013120 033737 001452 001234      BIT    @#BITS+<13*2>,TSTNMS ;DO THIS TEST?
4253 013126 001002              BNE    64$             ;YES--BRANCH
4254 013130 000137 013562              JMP    TST14           ;NO--GO TO THE NEXT TEST
4255 013134 012737 000013 001102      64$: MOV    #13,@#TSTNM     ;SET UP TEST NUMBER AND
4256                      ;CLEAR THE ERROR FLAG ($ERFLG)
4257 013142 004737 024726              JSR    PC,LODPRM       ;LOAD THE PARMETERS FOR THE TEST
4258 013146 012737 013116 001110      MOV    #TST13,@#SLPERR ;SETUP THE ERROR LOOP ADDRESS
4259 013154 013777 001102 165760      MOV    $TSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
4260 013162 013737 001506 001204      MOV    @#RPT,$TIMES    ;GET THE ITERATION COUNT
4261 013170 112737 000031 001115      MOVB   #25,$ERMAX      ;MAX ERRORS ALLOWED FOR TEST
4262 013176 005737 001244              TST    @#CLKSTA        ;KW11-P CLOCK?
4263 013202 003002              BGT    1$              ;YES--START TEST
4264 013204 000137 013562              JMP    TST14           ;NO--GO TO NEXT TEST
4265 013210 012737 013210 001106      1$:  MOV    #1$,$LPADR      ;SETUP THE LOOP ADDRESS
4266 013216 004037 027006              JSR    RO,@#SRCH00     ;DO A MASSBUS INIT. AND RECAL
4267 013222 000402              BR     2$              ;NO ERROR RETURN
4268 013224 000137 013560              JMP    EXIT13         ;ERROR RETURN--SCOPE LOOP CALL
4269 013230 012703 003014              2$:  MOV    #T10,R3        ;PARAMETER POINTER
4270 013234 012737 013560 001206      MOV    #EXIT13,$ESCAPE ;ESCAPE TO EXIT13 ON ERROR
4271 013242 012706 001100      TEST13: MOV    #STACK,SP      ;SETUP STACK
4272 013246 013737 001510 004176      MOV    FC,@#DTADPB+12  ;START WITH BEGINNING CYLINDER
4273 013254 005237 004176              INC    DTADPB+12       ;INCREMENT THE BEGINNING CYLINDER
4274 013260 005005              CLR    R5              ;SET THE UP/DOWN SWITCH TO UP
4275 013262 004737 027172              JSR    PC,@#STRIMR     ;INITIALIZE THE TIMERS
4276 013266 012777 013454 166100      MOV    #7$,@PKV        ;SETUP INCASE OF OVERFLOW
4277 013274 012777 027170 021332      MOV    #DORT1,@RPVEC   ;SET RP04/5/6 VECTOR
4278 013302 005077 166074              1$:  CLR    @PKB           ;START THE COUNTER AT ZERO
4279 013306 013764 004176 000034      MOV    @#DTADPB+12,RPCA(R4) ;LOAD DESIRED CYLINDER
4280 013314 012714 000105      MOV    #SEEK,(R4)      ;START A SEEK

```

```

4281 013320 012777 000131 166052      MOV      #131,@PKCS      ;START THE CLOCK
4282 013326 000001                WAIT                ;WAIT ON INTERRUPT
4283 013330 042777 000101 166042      BIC      #101,@PKCS     ;STOP THE CLOCK
4284 013336 032764 040000 000012      BIT      #BIT14,RPDS1(R4) ;ANY DISK ERRORS?
4285 013344 001415                BEQ      2$           ;NO--BRANCH
4286 013346 104412                SAVREG             ;SAVE R0-R5
4287 013350 012702 004164      MOV      #DTADPB,R2     ;DPB POINTER
4288 013354 004737 042202      JSR      PC,@#SVRH1     ;SAVE ALL THE RH11/RP04 REGISTERS
4289 013360 012764 000040 000010      MOV      #BIT05,RPCS2(R4) ;MASSBUS CLEAR
4290 013366 013764 004164 000010      MOV      @#DTADPB,RPCS2(R4) ;SELECT DRIVE
4291 013374 104413                RESREG            ;RESTORE R0-R5
4292 013376 104017                ERROR 17           ;REPORT THE ERROR
4293 013400 004737 027236      JSR      PC,@#COUNT   ;COUNT THIS SEEKS TIME
4294 013404 004737 026564      JSR      PC,@#TWOMS    ;STALL FOR 2 MILLISECONDS
4295 013410 005705                TST      R5           ;UP OR DOWN?
4296 013412 001011                BNE      4$           ;DOWN--BRANCH
4297 013414 005237 004176      INC      @#DTADPB+12    ;MOVE TO NEXT CYLINDER
4298 013420 023737 004176 001512      CMP      @#DTADPB+12,LC ;OUT OF CYLINDERS?
4299 013426 002725                BLT      1$           ;NO--GO DO THE NEXT SEEK
4300 013430 012705 177777      MOV      #-1,R5        ;SET UP/DOWN SWITCH TO DOWN
4301 013434 000722                BR       1$           ;GO DO THE NEXT SEEK
4302 013436 005337 004176      DEC      @#DTADPB+12    ;MOVE TO NEXT CYLINDER
4303 013442 023727 004176 000000      CMP      @#DTADPB+12,#0 ;OUT OF CYLINDERS?
4304 013450 003314                BGT      1$           ;NO--GO DO THE NEXT SEEK
4305 013452 000424                BR       8$           ;GO TO THE EXIT
4306 013454 042777 000101 165716      BIC      #101,@PKCS     ;STOP THE CLOCK
4307 013462 005037 177776      CLR      @#PS          ;DROP THE PRIORITY
4308 013466 012600      MOV      (SP)+,R0      ;PC OF WAIT+2
4309 013470 005726      TST      (SP)+        ;POP THE PS FROM THE STACK
4310 013472 104412                SAVREG            ;SAVE R0-R5
4311 013474 012702 004164      MOV      #DTADPB,R2     ;DPB POINTER
4312 013500 004737 042202      JSR      PC,@#SVRH1     ;SAVE ALL THE RH11/RP04 REGISTERS
4313 013504 012764 000040 000010      MOV      #BIT05,RPCS2(R4) ;MASSBUS CLEAR
4314 013512 013764 004164 000010      MOV      @#DTADPB,RPCS2(R4) ;SELECT DRIVE
4315 013520 104413                RESREG            ;RESTORE R0-R5
4316 013522 104020                ERROR 20           ;REPORT CLOCK OVERFLOW
4317 013524                8$:
4318 013524 012764 000040 000010      MOV      #BIT05,RPCS2(R4) ;MASSBUS INIT.
4319 013532 013764 004164 000010      MOV      @#DTADPB,RPCS2(R4) ;SELECT DRIVE
4320 013540 004737 024212      JSR      PC,@#ST.CLK    ;INITIALIZE THE CLOCK
4321 013544 012777 037544 021062      MOV      #ISR,@RPVEC    ;RESTORE RH11/RP04/5/6 INT. VECTOR
4322 013552 004037 027370      JSR      R0,@#TYPTIM    ;GO TYPE THE TIMES
4323 013556 003014                T10              ;POINTER
4324 013560 000004      EXIT13: SCOPE        ;LOOP ?

```

```

4325
4326
4327
4328
4329
4330
4331
4332
4333
4334
4335
4336

```

```

:*****
:*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 RPO4/5 OR TO 255 (10)

```

```

4337          ;*      FOR AN RP06.
4338
4339          ;*****
4340          TST14:
4341          NOP
4342          BIT      @#BITS+<14*2>,TSTNMS ;DO THIS TEST?
4343          BNE      64$      ;YES--BRANCH
4344          JMP      TST15     ;NO--GO TO THE NEXT TEST
4345          MOV      #14,@#TSTNM ;SET UP TEST NUMBER AND
4346          JSR      PC,LODPRM ;CLEAR THE ERROR FLAG ($ERFLG)
4347          MOV      #TST14,@#SLPERR ;LOAD THE PARMETERS FOR THE TEST
4348          MOV      $TSTNM,@DISPLAY ;SETUP THE ERROR LOOP ADDRESS
4349          MOV      @#RPT,$TIMES ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
4350          MOV      #25,$ERMAX ;GET THE ITERATION COUNT
4351          TST      @#CLKSTA ;MAX ERRORS ALLOWED FOR TEST
4352          BGT      1$      ;KW11-P CLOCK?
4353          JMP      TST15     ;YES--START TEST
4354          MOV      #1$,SLPADR ;NO--GO TO NEXT TEST
4355          JSR      RO,@#SRCH00 ;SET THE LOOP ADDRESS
4356          BR      2$      ;DO A MASSBUS INIT & RECAL
4357          JMP      EXIT14   ;RETURN HERE IF NO ERROR
4358          MOV      #T11,R3  ;RETURN HERE ON ERROR
4359          MOV      #EXIT14,$ESCAPE ;PARAMETER POINTER
4360          MOV      #STACK,SP ;ESCAPE TO EXIT14 ON ERROR
4361          MOV      #128.,R1  ;SETUP STACK
4362          JSR      PC,@#STRMTR ;REPEAT '0-136-0' 128 TIMES
4363          MOV      #7$,@PKV  ;INIT. THE COUNTERS
4364          MOV      #DORT1,@RPVEC ;SET UP VECTOR IN CASE OF OVERFLOW
4365          CLR      @PKB     ;SETUP RP04/5/6 VECTOR
4366          MOV      LC,RPCA(R4) ;START COUNT AT ZERO
4367          MOV      #SEEK,RPCS1(R4) ;'MIDDLE' CYLINDER
4368          MOV      #131,@PKCS ;START A SEEK
4369          WAIT    ;START THE CLOCK
4370          BIC     #101,@PKCS ;WAIT ON INTERRUPT
4371          BIT     #BIT14,RPDS1(R4) ;STOP CLOCK
4372          BEQ    2$      ;ERR=1?
4373          SAVREG ;NO--BRANCH
4374          MOV    #DTADPB,R2 ;SAVE R0-R5
4375          JSR   PC,@#SVRH11 ;DPB POINTER
4376          MOV  #BIT05,RPCS2(R4) ;SAVE ALL THE RH11/RP04 REGISTERS
4377          MOV  @#DTADPB,RPCS2(R4) ;MASSBUS CLEAR
4378          RESRFG ;SELECT DRIVE
4379          ERROR 17 ;RESTORE R0-R5
4380          CLR  R5
4381          JSR  PC,@#COUNT ;SET UP/DOWN SWITCH TO UP
4382          JSR  PC,@#TWOMS  ;UPDATE THE COUNT
4383          CLR  @PKB       ;STALL FOR 2 MILLISECONDS
4384          MOV  FC,RPCA(R4) ;START THE COUNT AT ZERO
4385          MOV  #SEEK,RPCS1(R4) ;BEGINNING CYLINDER
4386          MOV  #131,@PKCS ;START A SEEK
4387          WAIT ;START THE CLOCK
4388          BIC  #101,@PKCS ;WAIT ON INTERRUPT
4389          BIT  #BIT14,RPDS1(R4) ;STOP THE CLOCK
4390          BEQ  3$      ;ERR=1?
4391          SAVREG ;NO--BRANCH
4392          ;SAVE R0-R5

```

```
4393 014120 012702 004164      MOV      #DTADPB,R2      ;DPB POINTER
4394 014124 004737 042202      JSR      PC,@#SVRH11    ;SAVE ALL THE RH11/RP04 REGISTERS
4395 014130 012764 000040 000010  MOV      #BIT05,RPCS2(R4) ;MASSBUS CLEAR
4396 014136 013764 004164 000010  MOV      @#DTADPB,RPCS2(R4) ;SELECT DRIVE
4397 014144 104413      RESREG      ;RESTORE R0-R5
4398 014146 104017      ERROR      17
4399 014150 012705 177777      3$: MOV      #-1,R5      ;SET UP/DOWN SWITCH TO DOWN
4400 014154 004737 027236      JSR      PC,@#COUNT    ;UPDATE THE COUNT
4401 014160 004737 026564      JSR      PC,@#TWOMS     ;STALL FOR 2 MILLISECONDS
4402 014164 005301      DEC      R1      ;DONE?
4403 014166 003263      BGT      1$      ;NO--BRANCH
4404 014170 000424      BR      8$      ;YES--EXIT
4405 014172 042777 000101 165200 7$: BIC      #101,@PKCS    ;STOP THE CLOCK
4406 014200 005037 177776      CLR      @#PS      ;DROP THE PRIORITY
4407 014204 012600      MOV      (SP)+,R0     ;PC OF WAIT+2
4408 014206 005726      TST      (SP)+      ;POP THE PS FROM THE STACK
4409 014210 104412      SAVREG      ;SAVE R0-R5
4410 014212 012702 004164      MOV      #DTADPB,R2    ;DPB POINTER
4411 014216 004737 042202      JSR      PC,@#SVRH11    ;SAVE ALL THE RH11/RP04 REGISTERS
4412 014222 012764 000040 000010  MOV      #BIT05,RPCS2(R4) ;MASSBUS CLEAR
4413 014230 013764 004164 000010  MOV      @#DTADPB,RPCS2(R4) ;SELECT DRIVE
4414 014236 104413      RESREG      ;RESTORE R0-R5
4415 014240 104020      ERROR      20      ;CLOCK OVERFLOWED
4416 014242      8$:
4417 014242 012764 000040 000010  MOV      #BIT05,RPCS2(R4) ;MASSBUS INIT.
4418 014250 013764 004164 000010  MOV      @#DTADPB,RPCS2(R4) ;SELECT DRIVE
4419 014256 004737 024212      JSR      PC,@#ST.CLK    ;INITIALIZE THE CLOCK
4420 014262 012777 037544 020344  MOV      #ISR,@RPVEC    ;RESTORE RH11/RP04/5/6 INT. VECTOR
4421 014270 004037 027370      JSR      R0,@#TYPTIM    ;GO TYPE THE TIMES
4422 014274 003024      T11      ;POINTER
4423 014276 000004      EXIT14: SCOPE      ;LOOP ?
4424
4425      ;:*****
4426      ;*TEST 15      MAXIMUM SEEK TIMING TEST
4427
4428      ;*      THIS TEST WILL COMMAND A FORWARD SEEK FROM CYLINDER 0 TO
4429      ;*      CYLINDER 'LC', THEN A REVERSE SEEK FROM CYLINDER 'LC' TO
4430      ;*      CYLINDER 0. BOTH SEEKS ARE TIMED AND CHECKED TO ENSURE
4431      ;*      THEY ARE WITHIN THE TOLERANCE ALLOWED FOR THE MAXIMUM SEEK
4432      ;*      TIME. THIS SEQUENCE IS REPEATED 128 TIMES (FOR
4433      ;*      A TOTAL OF 256 SEEKS). THE MAXIMUM SEEK TIME MUST BE LESS THAN
4434      ;*      54 MS. 'LC' DEFAULTS TO 410 (10) FOR RP04/5'S AND 10 814 (10)
4435      ;*      FOR RP06'S.
4436
4437      ;:*****
4438      TST15:
4439 014300 000240      NOP
4440 014302 033737 001456 001234  BIT      @#BITS+<15*2>,TSTNMS ;DO THIS TEST?
4441 014310 001002      BNE      64$      ;YES--BRANCH
4442 014312 000137 015016      JMP      TST16     ;NO--GO TO THE NEXT TEST
4443 014316 012737 000015 001102 64$: MOV      #15,@#STSTNM    ;SET UP TEST NUMBER AND
4444      ;CLEAR THE ERROR FLAG ($ERFLG)
4445 014324 004737 024726      JSR      PC,LODPRM    ;LOAD THE PARMETERS FOR THE TEST
4446 014330 012737 014300 001110  MOV      #TST15,@#SLPERR ;SETUP THE ERROR LOOP ADDRESS
4447 014336 013777 001102 164576  MOV      $TSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
4448 014344 013737 001506 001204  MOV      @#RPT,$TIMES  ;GET THE ITERATION COUNT
```



```

4449 014352 112737 000031 001115      MOVB   #25.,SERMAX      ;MAX ERRORS ALLOWED FOR TEST
4450 014360 005737 001244      TST    @#CLKSTA        ;KW11-P CLOCK
4451 014364 003002      BGT    1$              ;YES--START TEST
4452 014366 000137 015016      JMP    TST16           ;NO--GO TO NEXT TEST
4453 014372 012737 014372 001106 1$:    MOV    #1$,SLPADR      ;SETUP THE LOOP ADDRESS
4454 014400 004037 027006      JSR    RO,@#SRCH00     ;DO A MASSBUS INIT & RECAL
4455 014404 000402      BR     2$              ;RETURN HERE IF NO ERROR
4456 014406 000137 015014      JMP    EXIT15          ;RETURN HERE ON ERROR
4457 014412 012703 003034      MOV    #T12,R3         ;PARAMETER POINTER
4458 014416 012737 015014 001206 2$:    MOV    #EXIT15,$ESCAPE ;:ESCAPE TO EXIT15 ON ERROR
4459 014424 012706 001100      TEST15: MOV   #STACK,SP       ;SETUP STACK
4460 014430 012701 000200      MOV    #128.,R1        ;REPEAT 'O-'LC'-O' 128 TIMES
4461 014434 004737 027172      JSR    PC,@#STRIMR     ;INIT. THE TIMERS
4462 014440 012777 014710 164726      MOV    #7$,@PKV        ;SETUP VECTOR IN CASE OF OVERFLOW
4463 014446 012777 027170 020160      MOV    #DORTI,@RPVEC  ;SETUP RPO4/5/6 VECTOR
4464 014454 005077 164722      1$:    CLR    @PKB            ;START COUNTING FROM ZERO
4465 014460 013764 001512 000034      MOV    LC,RPCA(R4)     ;MAXIMUM CYLINDER
4466 014466 012764 000105 000000      MOV    #SEEK,RPCS1(R4) ;START A SEEK
4467 014474 012777 000131 164676      MOV    #131,@PKCS     ;START THE CLOCK
4468 014502 000001      WAIT                    ;WAIT ON INTEPRUPT
4469 014504 042777 000101 164666      BIC    #101,@PKCS     ;STOP THE CLOCK
4470 014512 032764 040000 000012      BIT    #BIT14,RPDS1(R4) ;ERR=1?
4471 014520 001415      BEQ    2$              ;NO--BRANCH
4472 014522 104412      SAVREG                  ;SAVE R0-R5
4473 014524 012702 004164      MOV    #DTADPB,R2     ;DPB POINTER
4474 014530 004737 042202      JSR    PC,@#SVRH1     ;SAVE ALL THE RH11/RP04 REGISTERS
4475 014534 012764 000040 000010      MOV    #BIT05,RPCS2(R4) ;MASSBUS CLEAR
4476 014542 013764 004164 000010      MOV    @#DTADPB,RPCS2(R4) ;SELECT DRIVE
4477 014550 104413      RESREG                  ;RESTORE R0-R5
4478 014552 104017      ERROR 17
4479 014554 005005      2$:    CLR    R5              ;SET THE UP/DOWN SWITCH TO UP
4480 014556 004737 027236      JSR    PC,@#COUNT    ;UP THE COUNT
4481 014562 004737 026564      JSR    PC,@#TWOMS     ;STALL FOR 2 MILLISECONDS
4482 014566 005077 164610      CLR    @PKB            ;START COUNT AT ZERO
4483 014572 013764 001510 000034      MOV    FC,RPCA(R4)    ;BEGINNING CYLINDER
4484 014600 012764 000105 000000      MOV    #SEEK,RPCS1(R4) ;START A SEEK
4485 014606 012777 000131 164564      MOV    #131,@PKCS     ;START THE CLOCK
4486 014614 000001      WAIT                    ;WAIT ON INTEPRUPT
4487 014616 042777 000101 164554      BIC    #101,@PKCS     ;STOP THE CLOCK
4488 014624 032764 040000 000012      BIT    #BIT14,RPDS1(R4) ;'ERR'=1?
4489 014632 001415      BEQ    3$              ;NO--BRANCH
4490 014634 104412      SAVREG                  ;SAVE R0-R5
4491 014636 012702 004164      MOV    #DTADPB,R2     ;DPB POINTER
4492 014642 004737 042202      JSR    PC,@#SVRH1     ;SAVE ALL THE RH11/RP04 REGISTERS
4493 014646 012764 000040 000010      MOV    #BIT05,RPCS2(R4) ;MASSBUS CLEAR
4494 014654 013764 004164 000010      MOV    @#DTADPB,RPCS2(R4) ;SELECT DRIVE
4495 014662 104413      RESREG                  ;RESTORE R0-R5
4496 014664 104017      ERROR 17
4497 014666 012705 177777      3$:    MOV    #-1,R5         ;SET THE UP/DOWN SWITCH TO DOWN
4498 014672 004737 027236      JSR    PC,@#COUNT    ;UPDATE THE COUNT
4499 014676 004737 026564      JSR    PC,@#TWOMS     ;STALL FOR 2 MILLISECONDS
4500 014702 005301      DEC    R1              ;DONE?
4501 014704 003263      BGT    1$              ;NO--BRANCH
4502 014706 000424      BR     8$              ;YES--EXIT
4503 014710 042777 000101 164462 7$:    BIC    #101,@PKCS     ;STOP THE CLOCK
4504 014716 005037 177776      CLR    @#PS           ;DROP THE PRIORITY

```

```

4505 014722 012600      MOV      (SP)+,R0      ;PC OF WAIT+2
4506 014724 005726      TST      (SP)+        ;POP THE PS FROM THE STACK
4507 014726 104412      SAVREG                    ;SAVE R0-R5
4508 014730 012702 004164  MOV      #DTADPB,R2    ;DPB POINTER
4509 014734 004737 042202  JSR      PC,@#SVRH1    ;SAVE ALL THE RH11/RP04 REGISTERS
4510 014740 012764 000040 000010  MOV      #BIT05,RPCS2(R4) ;MASSBUS CLEAR
4511 014746 013764 004164 000010  MOV      @#DTADPB,RPCS2(R4) ;SELECT DRIVE
4512 014754 104413      RESREG                    ;RESTORE R0-R5
4513 014756 104020      ERROR 20                ;CLOCK OVERFLOWED
4514 014760
4515 014760 012764 000040 000010 8$: MOV      #BIT05,RPCS2(R4) ;MASSBUS INIT.
4516 014766 013764 004164 000010  MOV      @#DTADPB,RPCS2(R4) ;SELECT DRIVE
4517 014774 004737 024212  JSR      PC,@#ST.CLK    ;INITIALIZE THE CLOCK
4518 015000 012777 037544 017626  MOV      #ISR,@RPVEC    ;RESTORE RH11/RP04/5/6 INT. VECTOR
4519 015006 004037 027370  JSR      R0,@#TYPTIM    ;GO TYPE THE TIMES
4520 015012 003034      T12                      ;POINTER
4521 015014 000004      EXIT15: SCOPE          ;LOOP ?

```

4522  
4523  
4524

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

:/\:  
:/\:  
:/\:

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

:/\:  
:/\:  
:/\:

4538  
4539  
4540  
4541  
4542  
4543  
4544  
4545  
4546  
4547  
4548  
4549  
4550  
4551

\*\*\*\*\*  
:\*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 - 21 ARE WRITE CHECKED. THEN  
:\* SECTOR 1 IS REWRITTEN AND SECTORS 0 - 21 ARE WRITE CHECKED.  
:\* THIS REWRITE AND WRITE CHECK PROCEDURE IS CONTINUED UP THROUGH  
:\* REWRITE SECTOR 21 AND WRITE CHECK SECTORS 0-21.

4552  
4553  
4554  
4555  
4556  
4557  
4558  
4559  
4560  
4561  
4562  
4563  
4564  
4565  
4566  
4567  
4568  
4569  
4570  
4571  
4572  
4573  
4574  
4575  
4576  
4577

015016  
015016 000240  
015020 033737 001460 001234  
015026 001002  
015030 000137 015414  
015034 012737 000016 001102  
015042 004737 024726  
015046 012737 015112 001110  
015054 013777 001102 164060  
015062 013737 001506 001204  
015070 113737 001364 001115  
015076 012737 015412 001252  
015104 012737 015112 001106  
015112 012706 001100  
015116 004737 030022  
015122 013737 001510 004176  
015130 113737 001516 004175  
015136 105037 004174  
015142 013737 001352 004170  
015150 012737 050202 004172  
015156 012737 015156 001110  
015164 012706 001100  
015170 012737 000161 004166  
015176 004037 025642  
015202 012737 000151 004166

\*\*\*\*\*  
TST16:  
\*\*\*\*\*

NOP  
BIT @#BITS+<16\*2>,TSTNMS ;DO THIS TEST?  
BNE 64\$ ;YES--BRANCH  
JMP TST17 ;NO--GO TO THE NEXT TEST  
64\$: MOV #16,@#STSTNM ;SET UP TEST NUMBER AND  
;CLEAR THE ERROR FLAG (\$ERFLG)  
JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST  
MOV #TEST16,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS  
MOV \$TSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER  
MOV @#RPT,\$TIMES ;GET THE ITERATION COUNT  
MOVB ERR.CT,\$ERMAX ;MAX ERRORS ALLOWED FOR TEST  
MOV #EXIT16,@#BYPASS  
MOV #TEST16,\$LPADR ;SETUP THE LOOP ADDRESS  
TEST16: MOV #STACK,SP ;SET THE STACK POINTER  
JSR PC,@#FILBUF ;FILL THE BUFFER WITH SECTOR ADDRESSES  
MOV @#FC,@#DTADPB+12 ;CYLINDER  
MOVB @#FT,@#DTADPB+11 ;TRACK  
CLRB @#DTADPB+10 ;SECTOR  
MOV TRCKWC,@#DTADPB+4 ;WORD COUNT  
MOV #BUFFER,@#DTADPB+6 ;BUFFER ADDRESS  
MOV #,\$SLPERR ;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

```

4578 015210 012737 015210 001110      MOV    #,$LPERR      ;SETUP THE ERROR LOOP ADDRESS
4579 015216 012706 001100      MOV    #STACK,SP    ;LOAD THE STACK POINTER
4580 015222 004037 025642      JSR    RO,@#DRVCAL  ;START A DATA TRANSFER
4581 015226 012737 015226 001110      MOV    #,$LPERR      ;SETUP THE ERROR LOOP ADDRESS
4582 015234 012706 001100      MOV    #STACK,SP    ;LOAD THE STACK POINTER
4583 015240 004037 030060      JSR    RO,@#CLRBUF  ;CLEAR BUFFER
4584 015244 012737 000171 004166      MOV    #READ,@#DTADPB+2 ;COMMAND = READ
4585 015252 004037 025642      JSR    RO,@#DRVCAL  ;START A DATA TRANSFER
4586 015256 004037 030126      JSR    RO,@#CKSCTR  ;CHECK THE SECTOR DATA READ
4587 015262 012700 050202      MOV    #BUFFER,RO   ;BUFFER ADDRESS
4588 015266 005001              CLR    R1           ;FIRST SECTOR
4589 015270 012737 015270 001110      MOV    #,$LPERR      ;SETUP THE ERROR LOOP ADDRESS
4590 015276 012706 001100      MOV    #STACK,SP    ;LOAD THE STACK POINTER
4591 015302 012737 000161 004166 1$:  MOV    #WRITE,@#DTADPB+2 ;COMMAND=WRITE DATA
4592 015310 012737 177400 004170      MOV    #SCTRWC,@#DTADPB+4 ;WORD COUNT
4593 015316 010037 004172      MOV    RO,@#DTADPB+6 ;BUFFER ADDRESS
4594 015322 110137 004174      MOV    R1,@#DTADPB+10 ;SECTOR
4595 015326 004037 025642      JSR    RO,@#DRVCAL  ;START A DATA TRANSFER
4596 015332 012737 015332 001110      MOV    #,$LPERR      ;SETUP THE ERROR LOOP ADDRESS
4597 015340 012706 001100      MOV    #STACK,SP    ;LOAD THE STACK POINTER
4598 015344 012737 000151 004166      MOV    #WRCKD,@#DTADPB+2 ;COMMAND=WRITE CHECK DATA
4599 015352 013737 001352 004170      MOV    TRCKWC,@#DTADPB+4 ;WORD COUNT
4600 015360 012737 050202 004172      MOV    #BUFFER,@#DTADPB+6 ;BUFFER ADDRESS
4601 015366 105037 004174      CLRB  @#DTADPB+10   ;SECTOR -
4602 015372 004037 025642      JSR    RO,@#DRVCAL  ;START A DATA TRANSFER
4603 015376 062700 001000      ADD   #512.,RO     ;MOVE TO NEXT SECTOR
4604 015402 005201              INC   R1
4605 015404 023701 001630      CMP   PRMLMT+22,R1 ;DONE?
4606 015410 103334              BHS  1$           ;NO--BRANCH
4607 015412 000004              EXIT16: SCOPE    ;LOOP ?
4608
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4620
4621
4622
4623 015414
4624 015414 000240
4625 015416 033737 001462 001234
4626 015424 001002
4627 015426 000137 016034
4628 015432 012737 000017 001102 64$:
4629
4630 015440 004737 024726
4631 015444 012737 015510 001110
4632 015452 013777 001102 163462
4633 015460 013737 001506 001204

```

```

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

```

```

TST17:
NOP
BIT @#BITS+<17*2>,TSTNMS ;DO THIS TEST?
BNE 64$ ;YES--BRANCH
JMP TST20 ;NO--GO TO THE NEXT TEST
MOV #17,@#SISTNM ;SET UP TEST NUMBER AND
;CLEAR THE ERROR FLAG ($ERFLG)
JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
MOV #TEST17,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
MOV $SISTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
MOV @#RPT,$TIMES ;GET THE ITERATION COUNT

```

```

4634 015466 113737 001364 001115      MOV      ERR.CT, SERMAX      ;MAX ERRORS ALLOWED FOR TEST
4635 015474 012737 016032 001252      MOV      #EXIT17, @#BYPASS
4636 015502 012737 015510 001106      MOV      #TEST17, $LPADR    ;SETUP THE LOOP ADDRESS
4637 015510 012706 001100      TEST17: MOV     #STACK, SP    ;SET THE STACK POINTER
4638 015514 004737 030022      JSR      PC, @#FILBUF       ;FILL THE BUFFER WITH TRACK ADDRESS
4639 015520 012737 000161 004166      MOV      #WRITE, @#DTADPB+2 ;COMMAND=WRITE DATA
4640 015526 013737 001510 004176      MOV      @#FC, @#DTADPB+12  ;CYLINDER
4641 015534 113737 001524 004174      MOV      @#FS, @#DTADPB+10  ;SECTOR
4642 015542 012737 177400 004170      MOV      #SCTRUC, @#DTADPB+4 ;WORD COUNT
4643 015550 012737 050202 004172      MOV      #BUFFER, @#DTADPB+6 ;BUFFER ADDRESS
4644 015556 005000      CLR      R0                ;TRACK=0
4645 015560 012737 015560 001110      MOV      #., $LPERR        ;SETUP THE ERROR LOOP ADDRESS
4646 015566 012706 001100      MOV      #STACK, SP        ;LOAD THE STACK POINTER
4647 015572 110037 004175      1$:  MOV      RO, @#DTADPB+11 ;TRACK ADDRESS
4648 015576 004037 025642      JSR      RO, @#DRVCAL       ;START A DATA TRANSFER
4649 015602 062737 001000 004172      ADD      #256.*2., @#DTADPB+6 ;UPDATE BUFFER ADDRESS
4650 015610 005200      INC      R0                ;UPDATE TRACK NUMBER
4651 015612 022700 000023      CMP      #19., R0          ;OUT OF TRACKS?
4652 015616 003365      BGT      1$                ;NO--BRANCH
4653 015620 012737 050202 004172      MOV      #BUFFER, @#DTADPB+6 ;BUFFER ADDRESS
4654 015626 005000      CLR      R0
4655 015630 012737 015630 001110      MOV      #., $LPERR        ;SETUP THE ERROR LOOP ADDRESS
4656 015636 012706 001100      MOV      #STACK, SP        ;LOAD THE STACK POINTER
4657 015642 012737 000151 004166      MOV      #WRCKD, @#DTADPB+2 ;COMMAND=WRITE CHECK
4658 015650 110037 004175      2$:  MOV      RO, @#DTADPB+11 ;TRACK ADDRESS
4659 015654 004037 025642      JSR      RO, @#DRVCAL       ;START A DATA TRANSFER
4660 015660 062737 001000 004172      ADD      #256.*2., @#DTADPB+6 ;UPDATE BUFFER ADDRESS
4661 015666 005200      INC      R0                ;UPDATE TRACK NUMBER
4662 015670 022700 000023      CMP      #19., R0          ;OUT OF TRACKS?
4663 015674 003365      BGT      2$                ;NO--BRANCH
4664 015676 005000      CLR      R0                ;FIRST TRACK ADDRESS
4665 015700 110037 004175      3$:  MOV      RO, @#DTADPB+11 ;TRACK
4666 015704 010001      MOV      RO, R1            ;FORM BUFFER ADDRESS
4667 015706 012737 050202 004172      MOV      #BUFFER, @#DTADPB+6 ;BUFFER ADDRESS
4668 015714 005301      4$:  DEC      R1
4669 015716 002411      BLT      5$
4670 015720 062737 001000 004172      ADD      #256.*2., @#DTADPB+6
4671 015726 000772      BR      4$
4672 015730 012737 015730 001110      MOV      #., $LPERR        ;SETUP THE ERROR LOOP ADDRESS
4673 015736 012706 001100      MOV      #STACK, SP        ;LOAD THE STACK POINTER
4674 015742 012737 000161 004166      5$:  MOV      #WRITE, @#DTADPB+2 ;COMMAND=WRITE DATA
4675 015750 004037 025642      JSR      RO, @#DRVCAL       ;START A DATA TRANSFER
4676 015754 062737 001000 004172      6$:  ADD      #256.*2., @#DTADPB+6 ;UPDATE BUFFER ADDRESS
4677 015762 105237 004175      INCB    @#DTADPB+11        ;MOVE TO NEXT TRACK
4678 015766 012737 015766 001110      MOV      #., $LPERR        ;SETUP THE ERROR LOOP ADDRESS
4679 015774 012706 001100      MOV      #STACK, SP        ;LOAD THE STACK POINTER
4680 016000 012737 000151 004166      MOV      #WRCKD, @#DTADPB+2 ;COMMAND=WRITE CHECK DATA
4681 016006 004037 025642      JSR      RO, @#DRVCAL       ;START A DATA TRANSFER
4682 016012 122737 000022 004175      CMP      #18., @#DTADPB+11 ;OUT OF TRACKS?
4683 016020 003355      BGT      6$                ;NO--BRANCH
4684 016022 005200      INC      R0                ;NEXT TRACK TO WRITE
4685 016024 022700 000022      CMP      #18., R0          ;OUT OF TRACKS?
4686 016030 003323      BGT      3$                ;NO--BRANCH
4687 016032 000004      EXIT17: SCOPE

```

4688  
4689  
4690  
4691  
4692  
4693  
4694  
4695  
4696  
4697  
4698  
4699  
4700  
4701  
4702  
4703  
4704  
4705  
4706  
4707  
4708  
4709  
4710  
4711  
4712  
4713  
4714  
4715  
4716  
4717  
4718  
4719  
4720  
4721  
4722  
4723  
4724  
4725  
4726  
4727  
4728  
4729  
4730  
4731  
4732  
4733  
4734  
4735  
4736  
4737  
4738  
4739  
4740  
4741  
4742  
4743

.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 17777 (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

```

4744      :*      133333 004000 173777 155555 177757 066667 177777 000000
4745      :*      165555 010000 167777 172666 177767 153333 177777 000000
4746      :*      133333 020000 157777 155555 177773 066667 177777 000000
4747      :*      165555 040000 137777 172666 177775 153333 177777 000000
4748      :*      133333 100000 077777 155555 177776 066667 177777 000000
4749      :*
4750      :*
4751      :*
4752      :*
4753      :*
4754      :*
4755      :*
4756      :*
4757      :*
4758      :*
4759      :*
4760      :*
4761      :*
4762      :*
4763      :*
4764      :*
4765      :*
4766      :*
4767      :*
4768      :*
4769      :*
4770      :*
4771      :*
4772      :*
4773      :*
4774      :*
4775      :*
4776      :*
4777      :*
4778      :*
4779      :*
4780      :*
4781      :*
4782      :*
4783      :*
4784      :*
4785      :*
4786      :*
4787      :*
4788      :*
4789      :*
4790      :*
4791      :*
4792      :*
4793      :*
4794      :*
4795      :*
4796      :*
4797      :*
4798      :*
4799      :*
    
```

```

4752      016034      :*.....
4753      016034 000240      TST20:      NOP
4754      016036 033737 001424 001236      BIT      BITS+<20*2-40>,TSTNMS+2 ;DO THIS TEST ?
4755      016044 001002      BNE      64$      ;YES--BRANCH
4756      016046 000137 016554      JMP      TST21      ;NO--GO TO THE NEXT TEST
4757      016052 012737 000020 001102 64$:      MOV      #20,@#TSTNM      ;SET UP TEST NUMBER AND
4758      :*      ;CLEAR THE ERROR FLAG ($ERFLG)
4759      016060 004737 024726      JSR      PC,LODPRM      ;LOAD THE PARMETERS FOR THE TEST
4760      016064 012737 016236 001110      MOV      #TEST20,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
4761      016072 013777 001102 163042      MOV      $TSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
4762      016100 013737 001506 001204      MOV      @#RPT,$TIMES ;GET THE ITERATION COUNT
4763      016106 113737 001364 001115      MOV      ERR.CT,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
4764      016114 012737 016114 001106      MOV      #.,$LPADR ;SETUP THE LOOP ADDRESS
4765      016122 005000      CLR      R0 ;CLEAR SWITCH
4766      016124 005004      CLR      R4 ;FORM WORD COUNT IN R4
4767      016126 013701 001526      MOV      @#LS,R1
4768      016132 163701 001524      SUB      @#FS,R1
4769      016136 002004      BGE      1$ ;BRANCH IF FS < OR = LS
4770      016140 063701 001630      ADD      PRMLMT+22,R1 ;ADD MAXIMUM SECTOR ADDRESS TO
4771      016144 005201      INC      R1 ;MAKE THE DIFFERENCE POSITIVE
4772      016146 005100      COM      R0 ;SET SWITCH
4773      016150 062704 000400 1$:      ADD      #256.,R4
4774      016154 005301      DEC      R1
4775      016156 002374      BGE      1$
4776      016160 005404      NEG      R4
4777      016162 010405      MOV      R4,R5 ;COPY NORMAL WORD COUNT INTO SMALL WC
4778      016164 005700      TST      R0 ;SWITCH SET?
4779      016166 001412      BEQ      3$ ;NO--BRANCH
4780      016170 005005      CLR      R5 ;FORM WORD COUNT FOR LS < FS
4781      016172 013701 001630      MOV      PRMLMT+22,R1
4782      016176 163701 001524      SUB      @#FS,R1
4783      016202 062705 000400 2$:      ADD      #256.,R5
4784      016206 005301      DEC      R1
4785      016210 002374      BGE      2$
4786      016212 005405      NEG      R5
4787      016214 113737 001524 004174 3$:      MOV      @#FS,@#DTADPB+10 ;SECTOR
4788      016222 012737 050202 004172      MOV      #BUFFER,@#DTADPB+6 ;DATA BUFFER
4789      016230 012737 016552 001252      MOV      #EXIT20,@#BYPASS
4790      016236 012706 001100      TEST20: MOV      #STACK,SP ;LOAD THE STACK POINTER
4791      016242 005037 001334      CLR      @#WCEFLG ;CLEAR THE WRITE CHECK ERRUR FLAG
4792      016246 013701 001510      MOV      @#FC,R1 ;PICKUP FIRST CYLINDER
4793      016252 000407      BR      2$
4794      016254 005720 1$:      TST      (R0)+ ;MOVE TO NEXT DATA PATTERN
4795      016256 022700 000040      CMP      #16.*2.,R0 ;OUT OF PATTERNS?
4796      016262 003004      BGT      3$ ;NO--BRANCH
4797      016264 004037 027746      JSR      R0,@#INCCYL ;MOVE TO NEXT CYLINDER
4798      016270 000530      BR      EXIT20 ;OUT OF CYLINDERS
4799      016272 005000 2$:      CLR      R0 ;START WITH PATTERN 0
    
```

```

4800 016274 036037 001424 001530 38: BIT BITS(RO),@#PAT ;THIS PATTERN SELECTED?
4801 016302 001764 BEQ 18 ;NO--BRANCH
4802 016304 013702 001516 MOV @#FT,R2 ;FIRST TRACK
4803 016310 010137 004176 MOV R1,@#DTADPB+12 ;CYLINDER
4804 016314 110237 004175 48: MOV R2,@#DTADPB+11 ;TRACK
4805 016320 010437 004170 MOV R4,@#DTADPB+4 ;WORD COUNT
4806 016324 023701 001512 CMP LC,R1 ;LAST DISK CYLINDER?
4807 016330 003005 BGT 58 ;NO--BRANCH
4808 016332 022702 000022 CMP #18.,R2 ;LAST DISK TRACK?
4809 016336 003002 BGT 58 ;NO--BRANCH
4810 016340 010537 004170 MOV R5,@#DTADPB+4 ;SHORT WORD COUNT
4811 016344 017703 162570 58: MOV @SW2,R3 ;INHIBIT WRITE AND
4812 016350 005103 COM R3 ;WRITE CHECK?
4813 016352 032703 000030 BIT #SW04!SW03,R3
4814 016356 001436 BEQ 78 ;YES--BRANCH
4815 016360 004737 030476 JSR PC,@#SETBUF ;MOVE DATA PATTERN INTO THE BUFFER
4816 016364 032777 000020 162546 BIT #SW04,@SWR ;INHIBIT WRITE?
4817 016372 001012 BNE 68 ;YES--BRANCH
4818 016374 012737 016374 001110 MOV #.,$LPERR ;SETUP THE ERROR LOOP ADDRESS
4819 016402 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
4820 016406 012737 000161 004166 MOV #WRITE,@#DTADPB-2 ;COMMAND=WRITE DATA
4821 016414 004037 025642 JSR RO,@#DRVCAL ;START A DATA TRANSFER
4822 016420 032777 000010 162512 68: BIT #SW03,@SWR ;INHIBIT WRITE CHECK?
4823 016426 001012 BNE 78 ;YES--BRANCH
4824 016430 012737 016430 001110 MOV #.,$LPERR ;SETUP THE ERROR LOOP ADDRESS
4825 016436 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
4826 016442 012737 000151 004166 MOV #WRCKD,@#DTADPB-2 ;COMMAND=WRITE CHECK DATA
4827 016450 004037 025642 JSR RO,@#DRVCAL ;START A DATA TRANSFER
4828 016454 005737 001334 78: TST @#WCEFLG ;WRITE CHECK ERROR FLAG SET?
4829 016460 001404 BEQ 88 ;NO--BRANCH
4830 016462 032777 000001 162450 BIT #SW00,@SWR ;PERFORM READ AFTER FATAL 'WCE'
4831 016470 001424 BEQ 98 ;NO--BRANCH
4832 016472 032777 000004 162440 88: BIT #SW02,@SWR ;INHIBIT READ DATA AND SOFTWARE COMPARE?
4833 016500 001020 BNE 98 ;YES--BRANCH
4834 016502 012737 016502 001110 MOV #.,$LPERR ;SETUP THE ERROR LOOP ADDRESS
4835 016510 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
4836 016514 012737 000171 004166 MOV #READ,@#DTADPB+2 ;COMMAND=READ
4837 016522 004037 025642 JSR RO,@#DRVCAL ;START A DATA TRANSFER
4838 016526 032777 000002 162404 BIT #SW01,@SWR ;COMPARE THE DATA?
4839 016534 001002 BNE 98 ;NO--BRANCH
4840 016536 004737 030566 JSR PC,@#DATCMP ;YES--DO IT
4841 016542 004037 027716 98: JSR RO,@#INCRK ;MOVE TO NEXT TRACK
4842 016546 000642 BR 18 ;OUT OF TRACKS GO TO NEXT PATTERN
4843 016550 000661 BR 48 ;LOOP
4844 016552 000004 EXIT20: SCOPE ;SCOPE LOOP

```



4845  
4846  
4847  
4848  
4849  
4850  
4851  
4852  
4853  
4854  
4855  
4856  
4857  
4858  
4859  
4860  
4861  
4862  
4863  
4864  
4865  
4866  
4867  
4868  
4869  
4870  
4871  
4872  
4873  
4874  
4875  
4876  
4877  
4878  
4879  
4880  
4881  
4882  
4883  
4884  
4885  
4886  
4887  
4888  
4889  
4890  
4891  
4892  
4893  
4894  
4895  
4896  
4897  
4898  
4899  
4900

016554  
016554 000240  
016556 033737 001426 001236  
016564 001002  
016566 000137 017332  
016572 012737 000021 001102  
016600 004737 024726  
016604 012737 017024 001110  
016612 013777 001102 162322  
016620 013737 001506 001204  
016626 113737 001364 001115  
016634 012737 016634 001106  
016642 012737 017330 001252  
016650 012737 176543 023724  
016656 012737 123456 023726  
016664 013737 001510 004176  
016672 013737 001352 004170  
016700 012737 050202 004172  
016706 012737 000161 004166  
016714 032737 100000 001220  
016722 001027  
016724 004037 031104  
016730 005037 004174  
016734 012737 016734 001110  
016742 012706 001100  
016746  
016746 004037 025642  
016752 105237 004175

.SBTTL \*\*\* EXERCISE TEST \*\*\*

\*\*\*\*\*  
: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

\*\*\*\*\*  
TST21:

```

NOP
BIT BITS+<21*2-40>,STNMS+2 ;DO THIS TEST ?
BNE 64$ ;YES--BRANCH
JMP TST22 ;NO--GO TO THE NEXT TEST
64$: MOV #21,@#STSTNM ;SET UP TEST NUMBER AND
;CLEAR THE ERROR FLAG ($ERFLG)
JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
MOV #TST21,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
MOV $STSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
MOV @#RPT,$TIMES ;GET THE ITERATION COUNT
MOVB ERR.CT,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
MOV #,$SLPADR ;SETUP THE LOOP ADDRESS
MOV #EX1T21,@#BYPASS
MOV #176543,@#$HINUM ;PRIME THE RANDOM NUMBER GENERATOR
MOV #123456,@#$LONUM
MOV @#FC,@#DTADPB+12 ;CYLINDER
MOV TRCKWC,@#DTADPB+4 ;WORD COUNT
MOV #BUFFER,@#DTADPB+6 ;BUFFER ADDRESS
MOV #WRITE,@#DTADPB+2 ;COMMMAND
BIT #SW'5,@#C.SWR ;WRITE THE DISK PACK BEFORE TESTING?
BNE 3$ ;NO--BRANCH
JSR R0,@#FILRAN ;FILL DATA BUFFER WITH RANDOM DATA
4$: CLR @#DTADPB+10 ;SECTOR AND TRACK
MOV #,$SLPERR ;SETUP THE ERROR LOOP ADDRESS
MOV #STACK,$P ;LOAD THE STACK POINTER
5$: JSR R0,@#DRVCL ;START A DATA TRANSFER
INCB @#DTADPB+11 ;NEXT TRACK

```

4901	016756	122737	000023	004175		CMPB	#19.,@#DTADPB+11 ;TIME FOR NEXT CYLINDER
4902	016764	003370				BGT	2\$ ;NO--BRANCH
4903	016766	005237	004176			INC	@#DTADPB+12
4904	016772	023737	001512	004176		CMP	@#LC,@#DTADPB+12 ;OUT OF CYLINDERS?
4905	017000	002353				BGE	1\$ ;NO--BRANCH
4906	017002	012737	177400	004170	3\$:	MOV	#SCTRWC,@#DTADPB+4 ;WORD COUNT
4907	017010	012737	017024	001106		MOV	#TEST21,@#SLPADR
4908	017016	012737	017024	001110		MOV	#TEST21,@#SLPEHR
4909	017024	012706	001100		TEST21:	MOV	#STACK,SP ;SET STACK POINTER
4910	017030	004037	031360			JSR	RO,@#RANADR ;GENERATE A RANDOM ADDRESS
4911	017034	013737	004174	001340		MOV	@#DTADPB+10,@#SVADR ;SAVE THE TRACK/SECTOR
4912	017042	013737	004176	001342		MOV	@#DTADPB+12,@#SVADR+2 ;SAVE THE CYLINDER
4913	017050	012737	000161	004166		MOV	#WRITE,@#DTADPB+2 ;COMMAND=WRITE DATA
4914	017056	012701	050202			MOV	#BUFFER,R1 ;BUFFER ADDRESS
4915	017062	010137	004172			MOV	R1,@#DTADPB+6

```

4916 017066 004037 031324 JSR RO,@#RANPAT ;GENERATE RANDOM PATTERN
4917 017072 012737 017072 001110 MOV #,$LPERR ;SETUP THE ERROR LOOP ADDRESS
4918 017100 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
4919 017104 004037 025642 JSR RO,@#DRVCAL ;START A DATA TRANSFER
4920 017110 004037 031360 JSR RO,@#RANADR
4921 017114 012737 000171 004166 MOV #READ,@#DTADPB+2 ;COMMAND=READ DATA
4922 017122 012737 051202 004172 MOV #BUFFER+512,@#DTADPB+6 ;BUFFER ADDRESS
4923 017130 012737 017130 001110 MOV #,$LPERR ;SETUP THE ERROR LOOP ADDRESS
4924 017136 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
4925 017142 004037 025642 JSR RO,@#DRVCAL ;START A DATA TRANSFER
4926 017146 004037 031126 JSP RO,@#RANCK ;CHECK THE DATA
4927 017152 013737 001340 004174 MOV @#SVADR,@#DTADPB+10 ;GET ADDRESS OF WHERE THE LAST
4928 017160 013737 001342 004176 MOV @#SVADR+2,@#DTADPB+12 ;WRITE WAS PERFORMED
4929 017166 012737 000151 004166 MOV #WRCKD,@#DTADPB+2 ;COMMAND=WRITE CHECK DATA
4930 017174 012737 050202 004172 MOV #BUFFER,@#DTADPB+6 ;DATA BUFFER ADDRESS
4931 017202 012737 017202 001110 MOV #,$LPERR ;SETUP THE ERROR LOOP ADDRESS
4932 017210 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
4933 017214 004037 025642 JSR RO,@#DRVCAL ;START A DATA TRANSFER
4934 017220 004037 031360 JSR RO,@#RANADR ;GENERATE A RANDOM ADDRESS
4935 017224 012737 000171 004166 MOV #READ,@#DTADPB+2 ;COMMAND=READ
4936 017232 012737 051202 004172 MOV #BUFFER+512,@#DTADPB+6 ;DATA BUFFER ADDRESS
4937 017240 012737 017240 001110 MOV #,$LPERR ;SETUP THE ERROR LOOP ADDRESS
4938 017246 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
4939 017252 004037 025642 JSR RO,@#DRVCAL ;START A DATA TRANSFER
4940 017256 004037 031126 JSR RO,@#RANCK ;CHECK THE DATA
4941 017262 013737 001340 004174 MOV @#SVADR,@#DTADPB+10 ;GET DISK ADDRESS OF THE
4942 017270 013737 001342 004176 MOV @#SVADR+2,@#DTADPB+12 ;LAST WRITE
4943 017276 012737 000151 004166 MOV #WRCKD,@#DTADPB+2 ;COMMAND=WRITE CHECK DATA
4944 017304 012737 050202 004172 MOV #BU:FER,@#DTADPB+6 ;DATA BUFFER ADDRESS
4945 017312 012737 017312 001110 MOV #,$LPERR ;SETUP THE ERROR LOOP ADDRESS
4946 017320 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
4947 017324 004037 025642 JSR RO,@#DRVCAL ;START A DATA TRANSFER
4948 017330 000004 EXIT21: SCOPE ;LOOP ?
4949
4950 .SBTTL *** RPO4 ACCESS TIME ADJUSTMENT TEST ***
4951
4952 ;:*****
4953 ;*TEST 22 RPO4 ACCESS TIME ADJUSTMENT TEST
4954
4955 ;* THIS TEST PERFORMS SEEKS BETWEEN CYLINDERS 0 & 136 TO ALLOW THE
4956 ;* OPERATOR TO ADJUST THE ACCESS TIME ON AN RPO4 USING THE
4957 ;* DDU. THE PROGRAM STALLS APPROXIMATELY 5 SECONDS BETWEEN SEEKS
4958 ;* SO THAT THE ACCESS TIME INDICATORS ON THE DDU MAY BE OBSERVED.
4959
4960 ;:*****
4961 TST22:
4962 017332 000240 NOP
4963 017334 033737 001430 001236 BIT BITS+<22*2-40>,TSTNMS+2 ;DO THIS TEST ?
4964 017342 001002 BNE 64$ ;YES--BRANCH
4965 017344 000137 017502 JMP $EOP ;NO--GO TO THE END OF THE PROGRAM
4966 017350 012737 000022 001102 64$: MOV #22,@#STSTNM ;SET UP TEST NUMBER AND
4967 ;CLEAR THE ERROR FLAG ($ERFLG)
4968 017356 004737 024726 JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
4969 017362 012737 017420 001110 MOV #TEST22,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
4970 017370 013777 001102 161544 MOV $STSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
4971 017376 013737 001506 001204 MOV @#RP*,$TIMES ;GET THE ITERATION COUNT

```

```

4972 017404 112737 000144 001115      MOVB    #100, $ERMAX      ;MAX ERRORS ALLOWED FOR TEST
4973 017412 012737 017420 001106      MOV     #TEST22,$LPADR   ;SETUP THE LOOP ADDRESS
4974 017420 012706 001100      TEST22: MOV    #STACK,SP  ;SETUP THE STACK POINTER
4975 017424 013737 001512 004116      MOV     LC,DPB.A+12     ;ENDING CYLINDER
4976 017432 112737 000105 004106      MOVB    #SEEK,@#DPR.A+2 ;SEEK=COMMAND
4977 017440 004037 025150      JSR     RO,@#CALL.A     ;GO EXECUTE THE COMMAND
4978 017444 004037 026502      JSR     RO,STALL       ;STALL
4979 017450 001360      .WORD   STALL3         ;ADDRESS OF STALL VALUE
4980 017452 013737 001510 004116      MOV     FC,DPB.A+12     ;STARTING CYLINDER
4981 017460 112737 000105 004106      MOVB    #SEEK,@#DPB.A+2 ;SEEK=COMMAND
4982 017466 004037 025150      JSR     RO,@#CALL.A     ;GO EXECUTE THE COMMAND
4983 017472 004037 026502      JSR     RO,STALL       ;STALL
4984 017476 001360      .WORD   STALL3         ;ADDRESS OF STALL VALUE
4985 017500 000004      EXIT22: SCOPE          ;LOOP ?
4986
4987
4988
4989
4990
4991
4992
4993
4994
4995
4996
4997 017502
4998 017502 104401 017510
4999 017506 000410
5000
5001 017530
5002 017530 005737 001232
5003 017534 001434
5004 017536 104401 017544
5005 017542 000405
5006
5007 017556
5008 017556 013746 001254
5009 017562 104403
5010 017564 002
5011 017565 000
5012 017566 104401 017574
5013 017572 000412
5014
5015 017620
5016 017620 013746 001112
5017 017624 104402
5018 017626 005037 001112
5019 017632 005037 001102
5020 017636 005037 001204
5021 017642 005237 001100
5022 017646 042737 100000 001100
5023 017654 005327
5024 017656 000010
5025 017660 003030
5026 017662 012737
5027 017664 000010

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

$EOP:
      TYPE    ,65$      ;: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     @#$ERTTL,-(SP) ;:SAVE @#$ERTTL FOR TYPEOUT
      TYPOC   ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1$:
      CLR     @#$ERTTL  ;:ZERO ERROR TCTAL
      CLR     $TSTNM    ;: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?
$EOPCT: .WORD 8.
      BGT     $DOAGN    ;:YES
      MOV     (PC)+,@(PC)+ ;:RESTORE COUNTER
$ENDCT: .WORD 8.

```

5028	017666	017656				SEOPCT			
5029	017670	104401	017676			TYPE	.658		::TYPE ASCIZ STRING
5030	017674	000410				BR	648		::GET OVER THE ASCIZ
5031					::658:	.ASCIZ	<CR><LF>/END OF	TEST /	
5032	017716				648:				
5033	017716	104401	017746			TYPE	.SENULL		::TYPE NULL CHARACTER
5034	017722	013700	000042		\$GET42:	MOV	@#42,R0		::GET MONITOR ADDRESS
5035	017726	001405				BEQ	\$DOAGN		::BRANCH IF NO MONITOR
5036	017730	000005				RESET			::CLEAR THE WORLD
5037	017732	004710			\$ENDAD:	JSR	PC,(R0)		::GO TO MONITOR
5038	017734	000240				NOP			::SAVE ROOM
5039	017736	000240				NOP			::FOR
5040	017740	000240				NOP			::ACT11
5041	017742				\$DOAGN:				
5042	017742	000137				JMP	@(PC)+		::RETURN
5043	017744	006216			\$RTNAD:	.WORD	RESTART		
5044	017746	377	377	000	\$ENULL:	.BYTE	-1,-1,0		::NULL CHARACTER STRING
5045		017752				.EVEN			

```

5046
5047      .SBTTL *** SYSMAC SUBROUTINES ***
5048
5049      .SBTTL ERROR HANDLER ROUTINE
5050
5051      ;*****
5052      ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
5053      ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
5054      ;*AND GO TO TYPERR ON ERROR
5055      ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
5056      ;*SW15=1      HALT ON ERROR
5057      ;*SW13=1      INHIBIT ERROR TYPEOUTS
5058      ;*SW10=1      BELL ON ERROR
5059      ;*SW09=1      LOOP ON ERROR
5060      ;*CALL
5061      ;*      ERROR      N      ;;ERROR=EMT AND N-ERROR ITEM NUMBER
5062
5063      $ERROR:
5064      017752      104407      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
5065      017752      032777      000400      161156      BIT      #SW08,@SWR      ;;SEND ERROR MESSAGE TO TTY?
5066      017754      001411      BEQ      7$      ;;YES--BRANCH
5067      017762      005737      001230      TST      @#LPTAVL      ;;IS THERE A LINE PRINTER AVAILABLE?
5068      017764      001406      BEQ      7$      ;;NO--BRANCH
5069      017770      013737      001420      001150      MOV      @#LPS,@#STPS      ;;YES--SETUP STATUS
5070      020000      013737      001422      001152      MOV      @#LPB,@#STPB      ;;AND BUFFER REG.'S FOR LINE PRINTER
5071      020006      105237      001103      7$:      INCB      $ERFLG      ;;SET THE ERROR FLAG
5072      020012      001775      BEQ      7$      ;;DON'T LET THE FLAG GO TO ZERO
5073      020014      013777      001102      161120      MOV      $STNM,@DISPLAY      ;;DISPLAY TEST NUMBER AND ERROR FLAG
5074      020022      032777      002000      161110      BIT      #BIT10,@SWR      ;;BELL ON ERROR?
5075      020030      001402      BEQ      1$      ;;NO - SKIP
5076      020032      104401      001210      TYPE      ,SBELL      ;;RING BELL
5077      020036      005237      001112      1$:      INC      $ERTTL      ;;COUNT THE NUMBER OF ERRORS
5078      020042      011637      001116      MOV      (SP),$ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
5079      020046      162737      000002      001116      SUB      #2,$ERRPC
5080      020054      117737      161036      001114      MOV      @ERRPC,$ITEMB      ;;STRIP AND SAVE THE ERROR ITEM CODE
5081      020062      032777      020000      161050      BIT      #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
5082      020070      001004      BNE      20$      ;;SKIP TYPEOUTS
5083      020072      004737      020172      JSR      PC,TYPERR      ;;GO TO USER ERROR ROUTINE
5084      020076      104401      001215      TYPE      ,SCLRF
5085      020102      20$:
5086      020102      005777      161032      2$:      TST      @SWR      ;;HALT ON ERROR
5087      020106      100002      BPL      3$      ;;SKIP IF CONTINUE
5088      020110      000000      HALT      ;;HALT ON ERROR
5089      020112      104407      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
5090      020114      032777      001000      161016      3$:      BIT      #BIT09,@SWR      ;;LOOP ON ERROR SWITCH SET?
5091      020122      001402      BEQ      4$      ;;BR IF NO
5092      020124      013716      001110      MOV      $LPERR,(SP)      ;;FUDGE RETURN FOR LOOPING
5093      020130      005737      001206      4$:      TST      $ESCAPE      ;;CHECK FOR AN ESCAPE ADDRESS
5094      020134      001402      BEQ      5$      ;;BR IF NONE
5095      020136      013716      001206      MOV      $ESCAPE,(SP)      ;;FUDGE RETURN ADDRESS FOR ESCAPE
5096      020142      5$:
5097      020142      023737      000042      000046      CMP      @#42,@#46      ;;ACT11 AUTOMATIC MODE?
5098      020150      001001      BNE      6$      ;;NO, CONTINUE
5099      020152      000240      NOP
5100      020154      013737      001414      001150      6$:      MOV      @#TPS,@#STPS      ;;SET STATUS AND BUFFER REG.'S
5101      020162      013737      001416      001152      MOV      @#TPB,@#STPB      ;;FOR TTY
  
```

```

5102 020170 000002          RTI          ;RETURN FROM ERROR CALL
5103
5104                      ;*****
5105                      ;SBTTL  TYPERR - TYPE ERROR ROUTINE
5106                      ;THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE
5107                      ;WHICH ERROR IS TO BE REPORTED, IT THEN OBTAINS, FROM THE "ERROR
5108                      ;TABLE" ($ERRTB), AND REPORTS THE APPROPRIATE INFORMATION
5109                      ;CONCERNING THE ERROR.
5110                      ;CALL
5111                      ;      JSR      PC,@#TYPERR
5112                      ;      RETURN
5113
5114 020172 113737 001102 001176 TYPERR: MOVB  @#STSTNM,@#STMPO ;SAVE THE TEST NUMBER
5115 020200 104412          SAVREG          ;SAVE R0 - R5
5116 020202 162700 000004  SUB      #4,RC          ;FORM TEST PC
5117 020206 010037 001162  MOV      R0,@#SREG0      ;COPY R0-R5 IN $REG0-$REG5
5118 020212 010137 001164  MOV      R1,@#SREG1
5119 020216 010237 001166  MOV      R2,@#SREG2
5120 020222 010337 001170  MOV      R3,@#SREG3
5121 020226 010437 001172  MOV      R4,@#SREG4
5122 020232 010537 001174  MOV      R5,@#SREG5
5123 020236 113700 001114  MOVB   @#$ITEMB,R0      ;PICKUP ERROR ITEM NUMBER
5124 020242 010001          MOV      R0,R1          ;AND COPY IT INTO R1
5125 020244 005300          DEC      R0          ;FORM INDEX FOR ERROR TABLE
5126 020246 106300          ASLB   R0
5127 020250 106300          ASLB   R0
5128 020252 106300          ASLB   R0
5129 020254 103002          BCC    1$          ;IS ERROR > 37?
5130 020256 062700 000240  ADD     #ITEM41-$ERRTB,R0 ;YES--FORM OFFSET
5131 020262 062700 004306 1$:    ADD     #$ERRTB,R0  ;FORM ADDRESS
5132 020266 012037 020302  MOV     (R0)+,2$      ;GET ERROR MESSAGE (EM) POINTER
5133 020272 001447          BEQ    7$          ;BRANCH IF THERE ISN'T ONE
5134 020274 104401 001215  TYPE   ,SCLRF      ;"CARRIAGE RETURN - LINE FEED
5135 020300 104401          TYPE
5136 020302 000000 2$:    .WORD  0          ;"EM" POINTER GOES HERE
5137 020304 162701 000041  SUB     #41,R1        ;SPECIAL ERROR ITEM NUMBER?
5138 020310 100440          BMI    7$          ;NO--BRANCH
5139 020312 013701 001260  MOV     @#SVSTAT,R1   ;GET STATUS/ERROR INDICATOR
5140 020316 106301          ASLB  " R1          ;STRIP "DONE" BIT (BIT07)
5141 020320 006301          ASL   R1          ;STRIP "ERROR" BIT (BIT15)
5142 020322 012702 004254  MOV     #STATBL,R2    ;1ST ADDRESS ON STATUS MESSAGE POINTERS
5143 020326 005003          CLR   R3          ;CARRIAGE RETURN-LINE FEED SWITCH
5144 020330 104401 020336  TYPE   ,65$        ;:TYPE ASCIZ STRING
5145 020334 000402          BR    64$        ;:GET OVER THE ASCIZ
5146                      ;:65$: .ASCIZ / (/
5147                      64$:
5148 020342 012237 020364 3$:    MOV     (R2)+,5$      ;MESSAGE POINTER
5149 020346 006301          ASL   R1          ;TYPE THIS MESSAGE?
5150 020350 103013          BCC   6$          ;NO--BRANCH
5151 020352 005103          COM   R3          ;YES--TYPE A "CR" & "LF"?
5152 020354 001002          BNE   4$          ;NO--BRANCH
5153 020356 104401 001215  TYPE   ,SCLRF      ;YES
5154 020362 104401 4$:    TYPE
5155 020364 000000 5$:    .WORD  0          ;MESSAGE POINTER GOES HERE
5156 020366 005701          TST   R1          ;MORE TO TYPE?
5157 020370 001403          BEQ   6$          ;NO--BRANCH
    
```

```

5158 020372 104401 044330          TYPE      ,MSG.SP      ;YES--SPACES
5159 020376 000761          BR          3$          ;LOOP
5160 020400 001360          6$: BNE      3$          ;BRANCH IF NOT FINISHED
5161 020402 104401 020410          TYPE      ,67$        ;:TYPE ASCIZ STRING
5162 020406 000401          BR          66$        ;:GET OVER THE ASCIZ
5163          ;:67$: .ASCIZ  (/)
5164 020412          66$:
5165 020412 012037 020426          7$: MOV      (R0)+,8$    ;PICK UP DATA HEADER (DH) POINTER
5166 020416 001404          BEQ      9$          ;BRANCH IF NONE
5167 020420 104401 001215          TYPE      ,%CRLF      ;CARRIAGE RETURN-LINE FEED
5168 020424 104401          TYPE
5169 020426 000000          8$: .WORD    0          ;"DH" POINTER GOES HERE
5170 020430 012001          9$: MOV      (R0)+,R1    ;PICKUP DATA TABLE (DT) POINTER
5171 020432 001450          BEQ      20$         ;BRANCH IF NONE
5172 020434 005005          CLR      R5          ;SET INDENT SWITCH
5173 020436 012000          MOV      (R0)+,R0    ;DATA FORMAT (DF) POINTER
5174 020440 012002          MOV      (R0)+,R2    ;NUMBER OF DH'S TO TYPE
5175 020442 001441          BEQ      17$         ;BRANCH IF DH NUMBER IS 0
5176 020444 005105          COM      R5          ;NO INDENT
5177 020446 104401 001215          TYPE      ,%CRLF      ;CARRIAGE RETURN-LINE FEED
5178 020452 112003          10$: MOVB   (R0)+,R3    ;NUMBER OF DATA WORDS TO TYPE
5179 020454 112004          MOVB   (R0)+,R4    ;AND HOW TO TYPE THEM
5180 020456 006004          11$: ROR      R4          ;OCTAL OR DECIMAL?
5181 020460 103403          BCS      12$         ;DECIMAL--BRANCH
5182 020462 013146          MOV      @(R1)+,-(SP) ;:SAVE @(R1)+ FOR TYPEOUT
5183 020464 104402          TYPOC
5184 020466 000402          BR          13$
5185 020470          12$:
5186 020470 013146          MOV      @(R1)+,-(SP) ;:SAVE @(R1)+ FOR TYPEOUT
5187 020472 104405          TYPDS
5188 020474 005303          13$: DEC      R3          ;:GO TYPE--DECIMAL ASCII WITH SIGN
5189 020476 001403          BEQ      14$         ;MORE NUMBERS TO TYPE?
5190 020500 104401 044330          TYPE      ,MSG.SP      ;YES--TYPE SEPERATORS
5191 020504 000764          BR          11$         ;LOOP
5192 020506 005302          14$: DEC      R2          ;MORE DH'S?
5193 020510 003421          BLE      20$         ;NO--BRANCH
5194 020512 104401 001215          TYPE      ,%CRLF      ;YES--START A NEW LINE
5195 020516 005105          COM      R5          ;INDENT?
5196 020520 001002          BNE      15$         ;NO--BRANCH
5197 020522 104401 044330          TYPE      ,MSG.SP      ;YES--TYPE SPACES
5198 020526 012037 020534          15$: MOV      (R0)+,16$   ;GET NEXT DH
5199 020532 104401          TYPE
5200 020534 000000          16$: .WORD    0          ;DH POINTER GOES HERE
5201 020536 104401 001215          TYPE      ,%CRLF      ;CARRIAGE RETURN-LINE FEED
5202 020542 005705          TST      R5          ;INDENT?
5203 020544 001342          BNE      10$         ;NO--BRANCH
5204 020546 104401 044330          17$: TYPE      ,MSG.SP      ;YES--TYPE SPACES
5205 020552 000737          BR          10$         ;LOOP
5206 020554 104413          20$: RESREG
5207 020556 000207          RTS      PC          ;RETURN

```

.SBTTL TYPE ROUTINE

5208  
5209  
5210  
5211  
5212  
5213

```

;*****
;*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.

```



```

5214      ;*NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
5215      ;*NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
5216      ;*NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
5217      ;*
5218      ;*CALL:
5219      ;*1) USING A TRAP INSTRUCTION
5220      ;*      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
5221      ;*OR
5222      ;*      TYPE
5223      ;*      MESADR
5224      ;*
5225
5226 020560 105737 001157      $TYPE:  TSTB      $TFPLG      ;;IS THERE A TERMINAL?
5227 020564 100002      BPL      1$      ;;BR IF YES
5228 020566 000000      HALT      ;;HALT HERE IF NO TERMINAL
5229 020570 000407      BR      3$      ;;LEAVE
5230 020572 010046      1$:  MOV      RO,-(SP)      ;;SAVE RO
5231 020574 017600 000002      MOV      @2(SP),RO      ;;GET ADDRESS OF ASCIZ STRING
5232 020600 112046      2$:  MOVB     (RO)+,-(SP)      ;;PUSH CHARACTER TO BE TYPED ONTO STACK
5233 020602 001005      BNE      4$      ;;BR IF IT ISN'T THE TERMINATOR
5234 020604 005726      TST     (SP)+      ;;IF TERMINATOR POP IT OFF THE STACK
5235 020606 012600      60$:  MOV     (SP)+,RO      ;;RESTORE RO
5236 020610 062716 000002      3$:  ADD     #2,(SP)      ;;ADJUST RETURN PC
5237 020614 000002      RTI
5238 020616 122716 000011      4$:  CMPB     #HT,(SP)      ;;BRANCH IF <HT>
5239 020622 001430      BEQ      8$
5240 020624 122716 000200      CMPB     #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
5241 020630 001006      BNE      5$
5242 020632 005726      TST     (SP)+      ;;POP <CR><LF> EQUIV
5243 020634 104401      TYPE
5244 020636 001215      $CRLF
5245 020640 105037 020774      CLRB     $CHARCNT      ;;CLEAR CHARACTER COUNT
5246 020644 000755      BR      2$      ;;GET NEXT CHARACTER
5247 020646 004737 020730      5$:  JSR     PC,$TYPEPC      ;;GO TYPE THIS CHARACTER
5248 020652 123726 001156      6$:  CMPB     $FILLC,(SP)+      ;;IS IT TIME FOR FILLER CHARS.?
5249 020656 001350      BNE      2$      ;;IF NO GO GET NEXT CHAR.
5250 020660 013746 001154      MOV     $NULL,-(SP)      ;;GET # OF FILLER CHARS. NEEDED
5251      ;;AND THE NULL CHAR.
5252 020664 105366 000001      7$:  DECB     1(SP)      ;;DOES A NULL NEED TO BE TYPED?
5253 020670 002770      BLT      6$      ;;BR IF NO--GO POP THE NULL OFF OF STACK
5254 020672 004737 020730      JSR     PC,$TYPEPC      ;;GO TYPE A NULL
5255 020676 105337 020774      DECB     $CHARCNT      ;;DO NOT COUNT AS A COUNT
5256 020702 000770      BR      7$      ;;LOOP
5257
5258      ;HORIZONTAL TAB PROCESSOR
5259
5260 020704 112716 000040      8$:  MOVB     #' ,(SP)      ;;REPLACE TAB WITH SPACE
5261 020710 004737 020730      9$:  JSR     PC,$TYPEPC      ;;TYPE A SPACE
5262 020714 132737 000007 020774      BITB     #?,$CHARCNT      ;;BRANCH IF NOT AT
5263 020722 001372      BNE      9$      ;;TAB STOP
5264 020724 005726      TST     (SP)+      ;;POP SPACE OFF STACK
5265 020726 000724      BR      2$      ;;GET NEXT CHARACTER
5266 020730 105777 160214      $TYPEPC: TSTB     @2$TPS      ;;WAIT UNTIL PRINTER IS READY
5267 020734 100375      BPL     $TYPEPC
5268 020736 116677 000002 160206      MOVB     2(SP),@2$TPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
5269 020744 122766 000015 000002      CMPB     #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?

```

```

5270 020752 001003          BNE      1$          ;;BRANCH IF NO
5271 020754 105037 020774  CLRB     $CHARCNT    ;;YES--CLEAR CHARACTER COUNT
5272 020760 000406          BR       $TYPEX      ;;EXIT
5273 020762 122766 000012 000002 1$:  CMPB     #LF,2(SP)    ;;IS CHARACTER A LINE FEED?
5274 020770 001402          BEQ      $TYPEX      ;;BRANCH IF YES
5275 020772 105227          INCB     (PC)+       ;;COUNT THE CHARACTER
5276 020774 000000          $CHARCNT: .WORD    0    ;;CHARACTER COUNT STORAGE
5277 020776 000207          $TYPEX:  RTS      PC
5278
5279

```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

5280
5281
5282
5283
5284
5285
5286
5287
5288
5289
5290
5291
5292
5293
5294
5295
5296
5297
5298
5299
5300
5301
5302
5303
5304
5305
5306
5307
5308
5309
5310
5311
5312
5313
5314
5315
5316
5317
5318
5319
5320
5321
5322
5323
5324
5325

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*   MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPOS    ;;CALL FOR TYPEOUT
*   .BYTE   N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*   .BYTE   M              ;;M=1 OR 0
*                               ;;1=TYPE LEADING ZEROS
*                               ;;0=SUPPRESS LEADING ZEROS
*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
*   MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPON    ;;CALL FOR TYPEOUT
*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*   MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPOC    ;;CALL FOR TYPEOUT
5305 021000 017646 000000  $TYPOS: MOV     @ (SP),-(SP)    ;;PICKUP THE MODE
5306 021004 116637 000001 021223  MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
5307 021012 112637 021225  MOVB    (SP)+,$SOMODE+1 ;;NUMBER OF DIGITS TO TYPE
5308 021016 062716 000002  ADD     #2,(SP)        ;;ADJUST RETURN ADDRESS
5309 021022 000406          BR       $TYPON
5310 021024 112737 000001 021223  $TYPOC: MOVB   #1,$OFILL    ;;SET THE ZERO FILL SWITCH
5311 021032 112737 000006 021225  MOVB    #6,$SOMODE+1    ;;SET FOR SIX(6) DIGITS
5312 021040 112737 000005 021222  $TYPON: MOVB   #5,$OCNT    ;;SET THE ITERATION COUNT
5313 021046 010346          MOV     R3,-(SP)      ;;SAVE R3
5314 021050 010446          MOV     R4,-(SP)      ;;SAVE R4
5315 021052 010546          MOV     R5,-(SP)      ;;SAVE R5
5316 021054 113704 021225  MOVB    $SOMODE+1,R4   ;;GET THE NUMBER OF DIGITS TO TYPE
5317 021060 005404          NEG     R4
5318 021062 062704 000006  ADD     #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
5319 021066 110437 021224  MOVB    R4,$SOMODE     ;;SAVE IT FOR USE
5320 021072 113704 021223  MOVB    $OFILL,R4     ;;GET THE ZERO FILL SWITCH
5321 021076 016605 000012  MOV     12(SP),R5     ;;PICKUP THE INPUT NUMBER
5322 021102 005003          CLR     R3            ;;CLEAR THE OUTPUT WORD
5323 021104 006105          1$:  ROL     R5          ;;ROTATE MSB INTO "C"
5324 021106 000404          BR      3$           ;;GO DO MSB
5325 021110 006105          2$:  ROL     R5          ;;FORM THIS DIGIT

```

```

5326 021112 006105          ROL    R5
5327 021114 006105          ROL    R5
5328 021116 010503          MOV    R5,R3
5329 021120 006103          3$:   ROL    R3          ;;GET LSB OF THIS DIGIT
5330 021122 105337 021224   DECB   $OMODE          ;;TYPE THIS DIGIT?
5331 021126 100016          BPL    7$              ;;BR IF NO
5332 021130 042703 177770   BIC    #177770,R3      ;;GET RID OF JUNK
5333 021134 001002          BNE    4$              ;;TEST FOR 0
5334 021136 005704          TST    R4              ;;SUPPRESS THIS 0?
5335 021140 001403          BEQ    5$              ;;BR IF YES
5336 021142 005204          4$:   INC    R4          ;;DON'T SUPPRESS ANYMORE 0'S
5337 021144 052703 000060   BIS    #'0,R3          ;;MAKE THIS DIGIT ASCII
5338 021150 052703 000040   5$:   BIS    #' ,R3      ;;MAKE ASCII IF NOT ALREADY
5339 021154 110337 021220   MOVB   R3,8$          ;;SAVE FOR TYPING
5340 021160 104401 021220   TYPE   ,8$            ;;GO TYPE THIS DIGIT
5341 021164 105337 021222   7$:   DECB   $OCNT          ;;COUNT BY 1
5342 021170 003347          BGT    2$              ;;BR IF MORE TO DO
5343 021172 002402          BLT    6$              ;;BR IF DONE
5344 021174 005204          INC    R4              ;;INSURE LAST DIGIT ISN'T A BLANK
5345 021176 000744          BR     2$              ;;GO DO THE LAST DIGIT
5346 021200 012605          6$:   MOV    (SP)+,R5      ;;RESTORE R5
5347 021202 012604          MOV    (SP)+,R4      ;;RESTORE R4
5348 021204 012603          MOV    (SP)+,R3      ;;RESTORE R3
5349 021206 016666 000002 000004  MOV    2(SP),4(SP)    ;;SET THE STACK FOR RETURNING
5350 021214 012616          MOV    (SP)+,(SP)
5351 021216 000002          RTI                    ;;RETURN
5352 021220 000          8$:   .BYTE  0          ;;STORAGE FOR ASCII DIGIT
5353 021221 000          .BYTE  0          ;;TERMINATOR FOR TYPE ROUTINE
5354 021222 000          $OCNT: .BYTE  0     ;;OCTAL DIGIT COUNTER
5355 021223 000          $OFILL: .BYTE  0   ;;ZERO FILL SWITCH
5356 021224 000000          $OMODE: .WORD  0   ;;NUMBER OF DIGITS TO TYPE

```

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

5357
5358
5359
5360
5361
5362
5363
5364
5365
5366
5367
5368
5369
5370 021226
5371 021226 010046
5372 021230 010146
5373 021232 010246
5374 021234 010346
5375 021236 010546
5376 021240 012746 020200
5377 021244 016605 000020
5378 021250 100004
5379 021252 005405
5380 021254 112766 000055 000001
5381 021262 005000          1$:   CLR    R0          ;;ZERO THE CONSTANTS INDEX

;*****
;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
;*REPLACED WITH SPACES.
;*CALL:
;*      MOV    NUM,-(SP)          ;;PUT THE BINARY NUMBER ON THE STACK
;*      TYPDS          ;;GO TO THE ROUTINE

$TYPDS:
MOV    R0,-(SP)          ;;PUSH R0 ON STACK
MOV    R1,-(SP)          ;;PUSH R1 ON STACK
MOV    R2,-(SP)          ;;PUSH R2 ON STACK
MOV    R3,-(SP)          ;;PUSH R3 ON STACK
MOV    R5,-(SP)          ;;PUSH R5 ON STACK
MOV    #20200,-(SP)      ;;SET BLANK SWITCH AND SIGN
MOV    20(SP),R5         ;;GET THE INPUT NUMBER
BPL    1$                ;;BR IF INPUT IS POS.
NEG    R5                 ;;MAKE THE BINARY NUMBER POS.
MOVB   #'-,1(SP)         ;;MAKE THE ASCII NUMBER NEG.

```

```

5382 021264 012703 021442      MOV    #SDBLK,R3      ;;SETUP THE OUTPUT POINTER
5383 021270 112723 000040      MOVVB  #' ,(R3)+     ;;SET THE FIRST CHARACTER TO A BLANK
5384 021274 005002           2$: CLR    R2          ;;CLEAR THE BCD NUMBER
5385 021276 016001 021432      MOV    $DTBL(R0),R1  ;;GET THE CONSTANT
5386 021302 160105           3$: SUB    R1,R5      ;;FORM THIS BCD DIGIT
5387 021304 002402           BLT    4$           ;;BR IF DONE
5388 021306 005202           INC    R2          ;;INCREASE THE BCD DIGIT BY 1
5389 021310 000774           BR     3$
5390 021312 060105           4$: ADD    R1,R5      ;;ADD BACK THE CONSTANT
5391 021314 005702           TST    R2          ;;CHECK IF BCD DIGIT=0
5392 021316 001002           BNE    5$          ;;FALL THROUGH IF 0
5393 021320 105716           TSTB   (SP)        ;;STILL DOING LEADING 0'S?
5394 021322 100407           BMI    7$          ;;BR IF YES
5395 021324 106316           5$: ASLB   (SP)        ;;MSD?
5396 021326 103003           BCC    6$          ;;BR IF NO
5397 021330 116663 000001 177777  MOVVB  1(SP),-1(R3)  ;;YES--SET THE SIGN
5398 021336 052702 000060 6$: BIS    #'0,R2     ;;MAKE THE BCD DIGIT ASCII
5399 021342 052702 000040 7$: BIS    #' ,R2     ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
5400 021346 110223           MOVVB  R2,(R3)+     ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
5401 021350 005720           TST    (R0)+       ;;JUST INCREMENTING
5402 021352 020027 000010      CMP    R0,#10      ;;CHECK THE TABLE INDEX
5403 021356 002746           BLT    2$          ;;GO DO THE NEXT DIGIT
5404 021360 003002           BGT    8$          ;;GO TO E'IT
5405 021362 010502           MOV    R5,R2       ;;GET THE LSD
5406 021364 000764           BR     6$          ;;GO CHANGE TO ASCII
5407 021366 105726           8$: TSTB   (SP)+     ;;WAS THE LSD THE FIRST NON-ZERO?
5408 021370 100003           BPL    9$          ;;BR IF NO
5409 021372 116663 177777 177776 9$: MOVVB  -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
5410 021400 105013           CLRB   (R3)        ;;SET THE TERMINATOR
5411 021402 012605           MOV    (SP)+,R5    ;;POP STACK INTO R5
5412 021404 012603           MOV    (SP)+,R3    ;;POP STACK INTO R3
5413 021406 012602           MOV    (SP)+,R2    ;;POP STACK INTO R2
5414 021410 012601           MOV    (SP)+,R1    ;;POP STACK INTO R1
5415 021412 012600           MOV    (SP)+,R0    ;;POP STACK INTO R0
5416 021414 104401 021442      TYPE   ,SDBLK      ;;NOW TYPE THE NUMBER
5417 021420 016666 000002 000004  MOV    2(SP),4(SP)  ;;ADJUST THE STACK
5418 021426 012616           MOV    (SP)+,(SP)
5419 021430 000002           RTI
5420 021432 023420           $DTBL: 10000.
5421 021434 001750           1000.
5422 021436 000144           100.
5423 021440 000012           10.
5424 021442 000004           $DBLK: .BLKW 4
5425
5426           .SBTTL TTY INPUT ROUTINE
5427
5428           ;:*****
5429           .ENABL  LSB
5430 021452 000000      $TKCNT: .WORD 0      ;;NUMBER OF ITEMS IN QUEUE
5431 021454 000000      $TKQIN: .WORD 0     ;;INPUT POINTER
5432 021456 000000      $TKQOUT: .WORD 0    ;;OUTPUT POINTER
5433 021460 000002      $TKQSRT: .BLKB 2    ;;TTY KEYBOARD QUEUE
5434           $TKQEND=.
5435
5436           ;*TK INITIALIZE ROUTINE
5437           ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
    
```

```

5438      ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
5439
5440      ;*CALL:
5441      ;*      JSR      PC,$TKINT
5442      ;*      RETURN
5443
5444      $TKINT: CLR      $TKCNT      ;;CLEAR COUNT OF ITEMS IN QUEUE
5445      MOV      #$TKQSR, $TKQIN    ;;MOVE THE STARTING ADDRESS OF THE
5446      MOV      $TKQIN, $TKQOUT    ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
5447      MOV      #$TKSRV, @TKVEC   ;;INITIALIZE THE KEYBOARD VECTOR
5448      MOV      #200, @TKVEC+2    ;;'BR' LEVEL 4
5449      TST      @TKB              ;;CLEAR DONE FLAG
5450      MOV      #100, @TKS        ;;ENABLE TTY KEYBOARD INTERRUPT
5451      RTS      PC                ;;RETURN TO CALLER
5452
5453      ;*TK SERVICE ROUTINE
5454      ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
5455      ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
5456      ;*IT IN THE QUEUE.
5457      ;*IF THE CHARACTER IS A "CONTROL-C" (^C) $TKINT IS CALLED AND
5458      ;*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (START2)
5459
5460      $TKSRV: MOVB   @TKB, -(SP)    ;;PICKUP THE CHARACTER
5461      BIC      #^C177, (SP)        ;;STRIP THE JUNK
5462      CMP      (SP), #3            ;;IS IT A CONTROL C?
5463      BNE     1$                  ;;BRANCH IF NO
5464      TYPE    , $CNTLC            ;;TYPE A CONTROL-C (^C)
5465      JSR     PC, $TKINT          ;;INIT THE KEYBOARD
5466      TST     (SP)+              ;;CLEAN UP STACK
5467      JMP     START2             ;;CONTROL C RESTART
5468      1$:    CMP      (SP), #7      ;;IS IT A CONTROL G?
5469      BNE     2$                  ;;BRANCH IF NO
5470      CMP      #SWREG, SWR        ;;IS SOFT-SWR SELECTED?
5471      BEQ     6$                  ;;GO TO SWR CHANGE
5472
5473      2$:    CMP      #2, $TKCNT    ;;IS THE QUEUE FULL?
5474      BNE     3$                  ;;BRANCH IF NO
5475      TYPE    , $BELL            ;;RING THE TTY BELL
5476      TST     (SP)+              ;;CLEAN CHARACTER OFF OF STACK
5477      BR      5$                  ;;EXIT
5478      3$:    CMP      (SP), #23    ;;IS IT A CONTROL-S?
5479      BNE     32$                 ;;BRANCH IF NO
5480      CLR     @TKS                ;;DISABLE TTY KEYBOARD INTERRUPTS
5481      TST     (SP)+              ;;CLEAN CHAR OFF STACK
5482      31$:   TSTB    @TKS          ;;WAIT FOR A CHAR
5483      BPL     31$                 ;;LOOP UNTIL ITS THERE
5484      MOVB   @TKB, -(SP)        ;;GET THE CHARACTER
5485      BIC      #^C177, (SP)      ;;MAKE IT 7-BIT ASCII
5486      CMP      (SP)+, #21        ;;IS IT A CONTROL-Q?
5487      BNE     31$                 ;;BRANCH IF NO
5488      MOV     #100, @TKS         ;;REENABLE TTY KEYBOARD INTERRUPTS
5489      RTI                      ;;RETURN
5490      32$:   INC      $TKCNT      ;;COUNT THIS CHARACTER
5491      CMP      (SP), #140        ;;IS IT UPPER CASE?
5492      BLT     4$                  ;;BRANCH IF YES
5493      4$:

```

```

5494 021706 021627 000175          CMP      (SP),#175      ;;IS IT A SPECIAL CHAR?
5495 021712 003002          BGT      4$            ;;BRANCH IF YES
5496 021714 042716 000040          BIC      #40,(SP)      ;;MAKE IT UPPER CASE
5497 021720 112677 177530          4$:     MOVVB   (SP)+,@$TKQIN  ;;AND PUT IT IN QUEUE
5498 021724 005237 021454          INC      $TKQIN        ;;UPDATE THE POINTER
5499 021730 023727 021454 021462    CMP      $TKQIN,#$TKQEND ;;GO OFF THE END?
5500 021736 001003          BNE      5$            ;;BRANCH IF NO
5501 021740 012737 021460 021454    MOV      #$TKQSRT,$TKQIN ;;RESET THE POINTER
5502 021746 000002          5$:     RTI              ;;RETURN
5503
5504
5505
5506
5507
5508
5509 021750 022737 000176 001140    $CKSWR: CMP      #SWREG,SWR  ;;IS THE SOFT-SWR SELECTED
5510 021756 001124          BNE      15$          ;;EXIT IF NOT
5511 021760 105777 157160          TSTB    @$TKS        ;;IS A CHAR WAITING?
5512 021764 100121          BPL      15$          ;;IF NOT, EXIT
5513 021766 117746 157154          MOVVB   @$TKB,-(SP)   ;;YES
5514 021772 042716 177600          BIC      #^C177,(SP)  ;;MAKE IT 7-BIT ASCII
5515 021776 021627 000007          CMP      (SP),#7      ;;IS IT A CONTROL-G?
5516 022002 001300          BNE      2$            ;;IF NOT, PUT IT IN THE TTY QUEUE
5517
5518
5519
5520
5521
5522
5523 022004 123727 001134 000001    6$:     CMPB    $AUTOB,#1  ;;ARE WE RUNNING IN AUTO-MODE?
5524 022012 001674          BEQ      2$            ;;BRANCH IF YES
5525 022014 005726          TST      (SP)+        ;;CLEAR CONTROL-G OFF STACK
5526 022016 004737 021462          JSR      PC,$TKINT    ;;FLUSH THE TTY INPUT QUEUE
5527 022022 005077 157116          CLR      @$TKS        ;;DISABLE TTY KEYBOARD INTERRUPTS
5528 022026 112737 000001 001135    MOVVB   #1,$INTAG     ;;SET INTERRUPT MODE INDICATOR
5529
5530 022034 104401 022674          $GTSWR: TYPE    , $CNTLG  ;;ECHO THE CONTROL-G (^G)
5531 022040 104401 022701          TYPE    , $MSWR      ;;TYPE CURRENT CONTENTS
5532 022044 013746 000176          MOV      SWREG,-(SP)  ;;SAVE SWREG FOR TYPEOUT
5533 022050 104402          TYPOC   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
5534 022052 104401 022712          TYPE    , $MNEW      ;;PROMPT FOR NEW SWR
5535 022056 005046          19$:    CLR      -(SP)     ;;CLEAR COUNTER
5536 022060 005046          CLR      -(SP)     ;;THE NEW SWR
5537 022062 105777 157056          7$:     TSTB    @$TKS        ;;CHAR THERE?
5538 022066 100375          BPL      7$          ;;IF NOT TRY AGAIN
5539
5540 022070 117746 157052          MOVVB   @$TKB,-(SP)  ;;PICK UP CHAR
5541 022074 042716 177600          BIC      #^C177,(SP)  ;;MAKE IT 7-BIT ASCII
5542
5543 022100 021627 000003          CMP      (SP),#3      ;;IS IT A CONTROL-C?
5544 022104 001015          BNE      9$            ;;BRANCH IF NOT
5545 022106 104401 022662          TYPE    , $CNTLC     ;;YES, ECHO CONTROL-C (^C)
5546 022112 062706 000006          ADD      #6,SP        ;;CLEAN UP STACK
5547 022116 123727 001135 000001    CMPB    $INTAG,#1    ;;REENABLE TTY KEYBOARD INTERRUPTS?
5548 022124 001003          BNE      8$            ;;BRANCH IF NO
5549 022126 012777 000100 157010    MOV      #100,@$TKS   ;;ALLOW TTY KEYBOARD INTERRUPTS

```

```

5550 022134 000137 004730      8$:   JMP      START2          ;;CONTROL-C RESTART
5551
5552
5553 022140 021627 000025      9$:   CMP      (SP),#25        ;;IS IT A CONTROL-U?
5554 022144 001005                BNE      10$                ;;BRANCH IF NOT
5555 022146 104401 022667        TYPE    ,%CNTLU            ;;YES, ECHO CONTROL-U (^U)
5556 022152 062706 000006      20$:  ADD      #6,SP            ;;IGNORE PREVIOUS INPUT
5557 022156 000737                BR       19$                ;;LET'S TRY IT AGAIN
5558
5559
5560 022160 021627 000015      10$:  CMP      (SP),#15         ;;IS IT A <CR>?
5561 022164 001022                BNE      16$                ;;BRANCH IF NO
5562 022166 005766 000004        TST     4(SP)              ;;YES, IS IT THE FIRST CHAR?
5563 022172 001403                BEQ     11$                ;;BRANCH IF YES
5564 022174 016677 000002 156736  MOV     2(SP),@SWR         ;;SAVE NEW SWR
5565 022202 062706 000006      11$:  ADD      #6,SP            ;;CLEAR UP STACK
5566 022206 104401 001215      14$:  TYPE    ,%CRLF           ;;ECHO <CR> AND <LF>
5567 022212 123727 001135 000001  CMPB   $INTAG,#1         ;;RE-ENABLE TTY KBD INTERRUPTS?
5568 022220 001003                BNE     15$                ;;BRANCH IF NOT
5569 022222 012777 000100 156714  MOV     #100,@$TKS        ;;RE-ENABLE TTY KBD INTERRUPTS
5570 022230 000002                RTI
5571 022232 004737 020730      16$:  JSR     PC,$TYPEC         ;;ECHO CHAR
5572 022236 021627 000060        CMP     (SP),#60          ;;CHAR < 0?
5573 022242 002420                BLT     18$                ;;BRANCH IF YES
5574 022244 021627 000067        CMP     (SP),#67          ;;CHAR > 7?
5575 022250 003015                BGT     18$                ;;BRANCH IF YES
5576 022252 042726 000060        BIC     #60,(SP)+         ;;STRIP-OFF ASCII
5577 022256 005766 000002        TST     2(SP)             ;;IS THIS THE FIRST CHAR
5578 022262 001403                BEQ     17$                ;;BRANCH IF YES
5579 022264 006316                ASL     (SP)              ;;NO, SHIFT PRESENT
5580 022266 006316                ASL     (SP)              ;;CHAR OVER TO MAKE
5581 022270 006316                ASL     (SP)              ;;ROOM FOR NEW ONE.
5582 022272 005266 000002      17$:  INC     2(SP)             ;;KEEP COUNT OF CHAR
5583 022276 056616 177776        BIS     -2(SP), (SP)      ;;SET IN NEW CHAR
5584 022302 000667                BR      7$                ;;GET THE NEXT ONE
5585 022304 104401 001214      18$:  TYPE    ,%QUES           ;;TYPE ?<CR><LF>
5586 022310 000720                BR      20$                ;;SIMULATE CONTROL-U
5587      .DSABL  LSB
5588
5589
5590      ;*****
5591      ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
5592      ;*CALL:
5593      ;*   RDCHR          ;;GET A CHARACTER FROM THE QUEUE
5594      ;*   RETURN HERE   ;;CHARACTER IS ON THE STACK
5595      ;*                ;;WITH PARITY BIT STRIPPED OFF
5596      ;*
5597      ;*
5598 022312 011646 000004 000002 $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
5599 022314 016666 000004                MOV     4(SP),2(SP)       ;;THE PS
5600 022322 005066 000004                CLR     4(SP)            ;;GET READY FOR A CHARACTER
5601 022326 005046                CLR     -(SP)           ;;PUT NEW PS ON STACK
5602 022330 012746 022336                MOV     #64$,-(SP)       ;;PUT NEW PC ON STACK
5603 022334 000002                RTI                    ;;POP NEW PC AND PS
5604 022336
5605 022336 005737 021452      64$:  TST     $TKCNT           ;;WAIT ON A CHARACTER
1$:

```

```

5606 022342 001775          BEQ      1$
5607 022344 005337 021452    DEC      $TKCNT      ;;DECREMENT THE COUNTER
5608 022350 117766 177102 000004  MOVB    @TKQOUT,4(SP) ;;GET ONE CHARACTER
5609 022356 005237 021456          INC      $TKQOUT      ;;UPDATE THE POINTER
5610 022362 023727 021456 021462  CMP     $TKQOUT,$TKQEND ;;DID IT GO OFF OF THE END?
5611 022370 001003          BNE     2$          ;;BRANCH IF NO
5612 022372 012737 021460 021456  MOV     #$TKQSR1,$TKQOUT ;;RESET THE POINTER
5613 022400 000002          2$:     RTI          ;;RETURN
5614                                     ;*****
5615                                     ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
5616                                     ;*CALL:
5617                                     ;*   RDLIN          ;;INPUT A STRING FROM THE TTY
5618                                     ;*   RETURN HERE    ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
5619                                     ;*                                     ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
5620
5621 022402 010346          $RDLIN: MOV    R3,-(SP)      ;;SAVE R3
5622 022404 005046          CLR     -(SP)          ;;CLEAR THE RUBOUT KEY
5623 022406 012703 022636    1$:     MOV    #$TTYIN,R3    ;;GET ADDRESS
5624 022412 022703 022662    2$:     CMP    #$TTYIN+20.,R3 ;;BUFFER FULL?
5625 022416 101456          BLOS   4$            ;;BR IF YES
5626 022420 104410          RDCHR          ;;GO READ ONE CHARACTER FROM THE TTY
5627 022422 112613          MOVB   (SP)+,(R3)     ;;GET CHARACTER
5628 022424 122713 000177    10$:    CMPB   #177,(R3)     ;;IS IT A RUBOUT
5629 022430 001022          BNE   5$            ;;BR IF NO
5630 022432 005716          TST   (SP)          ;;IS THIS THE FIRST RUBOUT?
5631 022434 001007          BNE   6$            ;;BR IF NO
5632 022436 112737 000134 022634  MOVB   #' \,9$       ;;TYPE A BACK SLASH
5633 022444 104401 022634    TYPE   ,9$
5634 022450 012716 177777    MOV    #-1,(SP)      ;;SET THE RUBOUT KEY
5635 022454 005303          6$:     DEC    R3          ;;BACKUP BY ONE
5636 022456 020327 022636    CMP    R3,$TTYIN     ;;STACK EMPTY?
5637 022462 103434          BLO   4$            ;;BR IF YES
5638 022464 111337 022634    MOVB   (R3),9$       ;;SETUP TO TYPEOUT THE DELETED CHAR.
5639 022470 104401 022634    TYPE   ,9$
5640 022474 000746          BR     2$            ;;GO READ ANOTHER CHAR.
5641 022476 005716          5$:     TST   (SP)          ;;RUBOUT KEY SET?
5642 022500 001406          BEQ   7$            ;;BR IF NO
5643 022502 112737 000134 022634  MOVB   #' \,9$       ;;TYPE A BACK SLASH
5644 022510 104401 022634    TYPE   ,9$
5645 022514 005016          CLR   (SP)          ;;CLEAR THE RUBOUT KEY
5646 022516 122713 000025    7$:     CMPB   #25,(R3)     ;;IS CHARACTER A CTRL U?
5647 022522 001003          BNE   8$            ;;BR IF NO
5648 022524 104401 022667    TYPE   ,%CNTLU      ;;TYPE A CONTROL 'U'
5649 022530 000726          BR     1$            ;;GO START OVER
5650 022532 122713 000022    8$:     CMPB   #22,(R3)     ;;IS CHARACTER A '^R'?
5651 022536 001011          BNE   3$            ;;BRANCH IF NO
5652 022540 105013          CLRB  (R3)          ;;CLEAR THE CHARACTER
5653 022542 104401 001215    TYPE   ,%CRLF       ;;TYPE A 'CR' & 'LF'
5654 022546 104401 022636    TYPE   ,TTYIN       ;;TYPE THE INPUT STRING
5655 022552 000717          BR     2$            ;;GO PICKUP ANOTHER CHACTER
5656 022554 104401 001214    4$:     TYPE   ,%QUES     ;;TYPE A '?'
5657 022560 000712          BR     1$            ;;CLEAR THE BUFFER AND LOOP
5658 022562 111337 022634    3$:     MOVB   (R3),9$     ;;ECHO THE CHARACTER
5659 022566 104401 022634    TYPE   ,9$
5660 022572 122723 000015    CMPB   #15,(R3)+    ;;CHECK FOR RETURN
5661 022576 001305          BNE   2$            ;;LOOP IF NOT RETURN

```



```

5662 022600 105063 177777      CLR B    -1(R3)          ;;CLEAR RETURN (THE 15)
5663 022604 104401 001216      TYP E    $LF           ;;TYPE A LINE FEED
5664 022610 005726             TST     (SP)+          ;;CLEAN RUBOUT KEY FROM THE STACK
5665 022612 012603             MOV     (SP)+,R3       ;;RESTORE R3
5666 022614 011646             MOV     (SP)-,(SP)     ;;ADJUST THE STACK AND PUT ADDRESS OF THE
5667 022616 016666 000004 000002 MOV     4(SP),2(SP)    ;; FIRST ASCII CHARACTER ON IT
5668 022624 012766 022636 000004 MOV     #STTYIN,4(SP)
5669 022632 000002             RTI                    ;;RETURN
5670 022634 000          9$: .BYTE 0          ;;STORAGE FOR ASCII CHAR. TO TYPE
5671 022635 000          .BYTE 0          ;;TERMINATOR
5672 022636 000024          $TTYIN: .BLKB 20.       ;;RESERVE 20. BYTES FOR TTY INPUT
5673 022662 041536 005015 000  $CNTLC: .ASCIZ /^C/<15><12>  ;;CONTROL "C"
5674 022667 0136 006525 000012 $CNTLU: .ASCIZ /^U/<15><12>  ;;CONTROL "U"
5675 022674 043536 005015 000  $CNTLG: .ASCIZ /^G/<15><12>  ;;CONTROL "G"
5676 022701 015 051412 051127 $MSWR: .ASCIZ <15><12>/SWR = /
5677 022706 036440 000040          $MNEW: .ASCIZ / NEW = /
5678 022712 020040 042516 020127
5679 022720 020075 000          .EVEN
5680 022724
5681
5682          .SBTTL SCOPE HANDLER ROUTINE
5683
5684          ;;*****
5685          ;;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
5686          ;;*AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
5687          ;;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
5688          ;;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
5689          ;;*SW14=1 LOOP ON TEST
5690          ;;*SW11=1 INHIBIT ITERATIONS
5691          ;;*SW09=1 LOOP ON ERROR
5692          ;;*CALL
5693          ;;* SCOPE          ;;SCOPE=10T
5694
5695          $SCOPE:
5696 022724 104407          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
5697 022726 032777 040000 156204 1$: BIT #BIT14,@SWR          ;;LOOP ON PRESENT TEST?
5698 022734 001101          BNE $OVER          ;;YES IF SW14=1
5699          ;#####START OF CODE FOR THE XOR TESTER#####
5700 022736 000416          $XTSTR: BR 6$          ;;IF RUNNING ON THE "XOR" TESTER CHANGE
5701          ;;THIS INSTRUCTION TO A "NOP" (NOP 240)
5702 022740 013746 000004          MOV @#ERRVEC,-(SP)    ;;SAVE THE CONTENTS OF THE ERROR VECTOR
5703 022744 012737 022764 000004          MOV #5$,@#ERRVEC    ;;SET FOR TIMEOUT
5704 022752 005737 177060          TST @#177060        ;;TIME OUT ON XOR?
5705 022756 012637 000004          MOV (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
5706 022762 000453          BR $SVLAD           ;;GO TO THE NEXT TEST
5707 022764 022626          5$: CMP (SP)+,(SP)+   ;;CLEAR THE STACK AFTER A TIME OUT
5708 022766 012637 000004          MOV (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
5709 022772 000413          BR 7$              ;;LOOP ON THE PRESENT TEST
5710 022774          6$;#####END OF CODE FOR THE XOR TESTER#####
5711 022774 105737 001103          2$: TSTB $ERFLG      ;;HAS AN ERROR OCCURRED?
5712 023000 001421          BEQ 3$              ;;BR IF NO
5713 023002 123737 001115 001103          CMPB $ERMAX,$ERFLG  ;;MAX. ERRORS FOR THIS TEST OCCURRED?
5714 023010 101015          BHI 3$              ;;BR IF NO
5715 023012 032777 001000 156120          BIT #BIT09,@SWR     ;;LOOP ON ERROR?
5716 023020 001404          BEQ 4$              ;;BR IF NO
5717 023022 013737 001110 001106          7$: MOV $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE

```

```

5718 023030 000443          BR      $OVER
5719 023032 105037 001103 4$:     CLR    $ERFLG      ;; ZERO THE ERROR FLAG
5720 023036 005037 001204          CLR    $TIMES      ;; CLEAR THE NUMBER OF ITERATIONS TO MAKE
5721 023042 000415          BR      1$          ;; ESCAPE TO THE NEXT TEST
5722 023044 032777 004000 156066 3$:     BIT    #BIT11,@SWP  ;; INHIBIT ITERATIONS?
5723 023052 001011          BNE    1$          ;; BR IF YES
5724 023054 005737 001100          TST    $PASS      ;; IF FIRST PASS OF PROGRAM
5725 023060 001406          BEQ    1$          ;; INHIBIT ITERATIONS
5726 023062 005237 001104          INC    $ICNT      ;; INCREMENT ITERATION COUNT
5727 023066 023737 001204 001104          CMP    $TIMES,$ICNT ;; CHECK THE NUMBER OF ITERATIONS MADE
5728 023074 002021          BGE    $OVER      ;; BR IF MORE ITERATION REQUIRED
5729 023076 012737 000001 001104 1$:     MOV    #1,$ICNT    ;; REINITIALIZE THE ITERATION COUNTER
5730 023104 013737 023154 001204          MOV    $MXCNT,$TIMES ;; SET NUMBER OF ITERATIONS TO DO
5731 023112 105237 001102          $SVLAD: INCR   $STNM      ;; COUNT TEST NUMBERS
5732 023116 011637 001106          MOV    (SP),$LPADR ;; SAVE SCOPE LOOP ADDRESS
5733 023122 011637 001110          MOV    (SP),$LPERR ;; SAVE ERROR LOOP ADDRESS
5734 023126 005037 001206          CLR    $ESCAPE    ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
5735 023132 112737 000001 001115          MOV    #1,$ERMAX  ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
5736 023140 013777 001102 155774 $OVER: MOV    $STNM,@DISPLAY ;; DISPLAY TEST NUMBER
5737 023146 013716 001106          MOV    $LPADR,(SP) ;; FUDGE RETURN ADDRESS
5738 023152 000002          RTI             ;; FIXES PS
5739 023154 000001          $MXCNT: 1        ;; MAX. NUMBER OF ITERATIONS

```

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

```

5740
5741
5742
5743
5744
5745
5746
5747
5748
5749
5750
5751
5752
5753
5754
5755
5756
5757

```

```

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

```

```

5758 023156          $SAVREG:
5759 023156 010046          MOV    R0,-(SP)    ;; PUSH R0 ON STACK
5760 023160 010146          MOV    R1,-(SP)    ;; PUSH R1 ON STACK
5761 023162 010246          MOV    R2,-(SP)    ;; PUSH R2 ON STACK
5762 023164 010346          MOV    R3,-(SP)    ;; PUSH R3 ON STACK
5763 023166 010446          MOV    R4,-(SP)    ;; PUSH R4 ON STACK
5764 023170 010546          MOV    R5,-(SP)    ;; PUSH R5 ON STACK
5765 023172 016646 000022          MOV    22(SP),-(SP) ;; SAVE PS OF MAIN FLOW
5766 023176 016646 000022          MOV    22(SP),-(SP) ;; SAVE PC OF MAIN FLOW
5767 023202 016646 000022          MOV    22(SP),-(SP) ;; SAVE PS OF CALL
5768 023206 016646 000022          MOV    22(SP),-(SP) ;; SAVE PC OF CALL
5769 023212 000002          RTI

```

```

5770
5771
5772
5773

```

```

;*RESTORE R0-R5
;*CALL
;* RESREG

```

5774 023214  
 5775 023214 012666 000022  
 5776 023220 012666 000022  
 5777 023224 012666 000022  
 5778 023230 012666 000022  
 5779 023234 012605  
 5780 023236 012604  
 5781 023240 012603  
 5782 023242 012602  
 5783 023244 012601  
 5784 023246 012600  
 5785 023250 000002

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

5786  
 5787  
 5788  
 5789  
 5790  
 5791  
 5792  
 5793

.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.

5794  
 5795 023252 010046  
 5796 023254 016600 000002  
 5797 023260 005740  
 5798 023262 111000  
 5799 023264 006300  
 5800 023266 016000 023306  
 5801 023272 000200

\$TRAP: MOV R0,-(SP) ;;SAVE R0  
 MOV 2(SP),R0 ;;GET TRAP ADDRESS  
 TST -(R0) ;;BACKUP BY 2  
 MOVB (R0),R0 ;;GET RIGHT BYTE OF TRAP  
 ASL R0 ;;POSITION FOR INDEXING  
 MOV \$TRPAD(R0),R0 ;;INDEX TO TABLE  
 RTS R0 ;;GO TO ROUTINE

5802  
 5803  
 5804  
 5805

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

5806 023274 011646  
 5807 023276 016666 000004 000002  
 5808 023304 000002

\$TRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN  
 MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN  
 RTI ;;RESTORE THE PSW

5809  
 5810  
 5811  
 5812  
 5813  
 5814

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

5815  
 5816  
 5817 023306 023274  
 5818 023310 020560  
 5819 023312 021024  
 5820 023314 021000  
 5821 023316 021040  
 5822 023320 021226  
 5823  
 5824 023322 022040  
 5825  
 5826 023324 021750  
 5827 023326 022312  
 5828 023330 022402  
 5829 023332 023156

ROUTINE  
 -----  
 \$TRPAD: .WORD \$TRAP2  
 \$TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE  
 \$TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)  
 \$TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)  
 \$TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)  
 \$TYPDS ;;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)  
 \$GTSWR ;;CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING  
 \$CKSWR ;;CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR  
 \$RDCHR ;;CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE  
 \$RDLIN ;;CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE  
 \$SAVREG ;;CALL=SAVREG TRAP+12(104412) SAVE R0-R5 ROUTINE

```

5830 023334 023214          $RESREG ;;CALL=RESREG  TRAP+13(104413) RESTORE R0-R5 ROUTINE
5831
5832          .SBTTL  SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE
5833
5834          ;*****
5835          ;*THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
5836          ;*UNSIGNED DECIMAL ASCIZ NUMBER.
5837          ;*CALL
5838          ;*      MOV      NUMBER,-(SP)      ;;PUT BINARY NUMBER ON THE STACK
5839          ;*      JSR      PC,@#$SB2D      ;;CALL
5840          ;*      RETURN                      ;;ADDRESS OF THE 1ST ASCIZ CHAR.IS ON THE STACK
5841
5842
5843 023336 016637 000002 023366 $SB2D:  MOV      2(SP),1$      ;;SAVE BINARY NUMBER
5844 023344 012746 023366      MOV      #1$,-(SP)      ;;SET POINTER
5845 023350 004737 023372      JSR      PC,@#$DB2D      ;;CALL DOUBLE LENGTH CONVERT
5846 023354 062716 000005      ADD      #5,(SP)        ;;ONLY ALLOW FIVE CHARACTERS
5847 023360 012666 000002      MOV      (SP)+,2(SP)    ;;PICKUP POINTER
5848 023364 000207                RTS      PC              ;;RETURN
5849 023366 000000 000000      1$:      .WORD      0,0
5850
5851          .SBTTL  DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
5852
5853          ;*****
5854          ;*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
5855          ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
5856          ;*POSITIVE.
5857          ;*CALL
5858          ;*      MOV      #PNTR,-(SP)      ;;POINTER TO LOW WORD OF BINARY NUMBER
5859          ;*      JSR      PC,@#$DB2D      ;;CALL
5860          ;*      RETURN                      ;;THE FIRST ADDRESS OF ASCII
5861          ;*                                  ;;IS ON THE STACK
5862
5863
5864 023372 104412                $DB2D:  SAVREG                      ;;SAVE REGISTERS
5865 023374 016602 000002      MOV      2(SP),R2      ;;PICKUP THE DATA POINTER
5866 023400 012700 023552      MOV      #$DECVL,R0    ;;GET ADDRESS OF "$DECVL" STRING
5867 023404 010066 000002      MOV      R0,2(SP)     ;;PUT ADDRESS OF ASCII STRING ON STACK
5868 023410 012201                MOV      (R2)+,R1      ;;PICKUP THE BINARY NUMBER
5869 023412 012202                MOV      (R2)+,R2
5870 023414 012737 000012 023470      MOV      #10.,4$      ;;SET UP TO DO 10 CONVERSIONS
5871 023422 012704 023502      MOV      #$INPWR,R4    ;;ADDRESS OF TEN POWER
5872 023426 012705 023504      MOV      #$INPWR+2,R5
5873 023432 005003                1$:      CLR      R3              ;;CLEAR PARTIAL
5874 023434 161401                2$:      SUB      (R4),R1      ;;SUBTRACT TEN POWER
5875 023436 005602                SBC      R2
5876 023440 161502                SUB      (R5),R2
5877 023442 002402                BLT      3$            ;;BR IF TEN POWER TO LARGE
5878 023444 005203                INC      R3              ;;ADD 1 TO PARTIAL
5879 023446 000772                BR       2$            ;;LOOP
5880 023450 062401                3$:      ADD      (R4)+,R1      ;;RESTORE SUBTRACTED VALUE
5881 023452 005502                ADC      R2
5882 023454 062402                ADD      (R4)+,R2
5883 023456 022525                CMP      (R5)+,(R5)+    ;;MOVE TO NEXT TEN POWER
5884 023460 052703 000060      BIS      #'0,R3        ;;CHANGE PARTIAL TO ASCII
5885 023464 110320                MOVB     R3,(R0)+      ;;SAVE IT

```

```

5886 023466 005327          DEC      (PC)+          ;;DONE?
5887 023470 000000          4$: .WORD      0
5888 023472 001357          BNE     1$              ;;BR IF NO
5889 023474 105020          CLR    (RO)+          ;;TERMINATOR
5890 023476 104413          RESREG          ;;RESTORE REGISTERS
5891 023500 000207          RTS     PC            ;;RETURN
5892 023502 145000          $INPWR: 145000        ;;1.0E09
5893 023504 035632          35632
5894 023506 160400          160400              ;;1.0E08
5895 023510 002765          2765
5896 023512 113200          113200              ;;1.0E07
5897 023514 000230          230
5898 023516 041100          041100              ;;1.0E06
5899 023520 000017          17
5900 023522 103240          103240              ;;1.0E05
5901 023524 000001          1
5902 023526 023420          23420               ;;1.0E04
5903 023530 000000          0
5904 023532 001750          1750                ;;1.0E03
5905 023534 000000          0
5906 023536 000144          144                 ;;1.0E02
5907 023540 000000          0
5908 023542 000012          12                  ;;1.0E01
5909 023544 000000          0
5910 023546 000001          1                    ;;1.0E00
5911 023550 000000          0
5912 023552 000014          $DECVL: .BLKB 12.   ;;RESERVE STORAGE FOR ASCII STRING

```

```

5913
5914 .SBTTL TYPE NUMERICAL ASCII STRING SUPPRESS LEADING ZEROS
5915

```

```

5916 .....
5917 *THIS ROUTINE IS USED TO TYPE AN ASCII NUMBER SUPPRESSING THE
5918 *LEADING NUMBERS.
5919 *CALL
5920 *      MOV      #NUMADR,-(SP)  ;;FIRST ADDRESS OF ASCII STRING
5921 *      JSR      PC,@#$SUPRS
5922

```

```

5923
5924 $SUPRS: MOV      R0,-(SP)      ;;SAVE R0
5925         MOV      4(SP),R0    ;;PICKUP THE POINTER
5926 1$:     TSTB     (R0)        ;;TERMINATOR?
5927         BEQ     2$          ;;BR IF YES
5928         CMPB    #'0,(R0)+    ;;IS THIS AN ASCII '0' ?
5929         BEQ     1$          ;;BR IF YES
5930 2$:     DEC     R0           ;;BACKUP BY '1'
5931         MOV     R0,3$       ;;SAVE FOR TYPING
5932         TYPE    3$          ;;GO TYPE
5933 3$:     .WORD    0           ;;ASCII POINTER GOES HERE
5934         MOV     (SP)+,R0     ;;RESTORE R0
5935         MOV     (SP)+,(SP)   ;;RESTORE THE STACK
5936         RTS     PC          ;;RETURN

```

```

5937
5938 .SBTTL RANDOM NUMBER GENERATOR ROUTINE
5939

```

```

5940 .....
5941 *THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR

```

```

5942 ;*WITH A RANGE OF 0 TO 2(+33)-1.
5943 ;*CALL:
5944 ;* JSR PC,$RAND ;:CALL THE ROUTINE
5945 ;* RETURN ;:RETURN HERE THE RANDOM
5946 ;* ;:NUMBER WILL BE IN
5947 ;* ;:$HINUM,$LONUM
5948 ;*

```

```

5949 023626 $RAND:
5950 023626 010046 MOV RO,-(SP) ;:PUSH RO ON STACK
5951 023630 010146 MOV R1,-(SP) ;:PUSH R1 ON STACK
5952 023632 010246 MOV R2,-(SP) ;:PUSH R2 ON STACK
5953 023634 013700 023726 MOV $LONUM,R0 ;:SET R0 WITH LOW
5954 023640 013701 023724 MOV $HINUM,R1 ;:SET R1 WITH HIGH
5955 023644 012702 177771 MOV #-7,R2 ;:SET SHIFT COUNT
5956 023650 006300 1$: ASL RO ;:SHIFT RO LEFT AND
5957 023652 006101 ROL R1 ;:ROTATE CARRY INTO R1 AND
5958 023654 005202 INC R2 ;:CHECK FOR DONE
5959 023656 001374 BNE 1$ ;:CONTINUE SHIFT LOOP
5960 023660 063700 023726 ADD $LONUM,R0 ;:ADD NUMBER TO MAKE X 129
5961 023664 005501 ADC R1 ;:PROPOGATE CARRY
5962 023666 063701 023724 ADD $HINUM,R1 ;:ADD NUMBER TO MAKE X 129
5963 023672 062700 001057 ADD #1057,R0 ;:ADD LOW CONSTANT
5964 023676 005501 ADC R1 ;:PROPOGATE CARRY
5965 023700 062701 047401 ADD #47401,R1 ;:ADD HIGH CONSTANT
5966 023704 010037 023726 MOV RO,$LONUM ;:SAVE RO
5967 023710 010137 023724 MOV R1,$HINUM ;:SAVE R1
5968 023714 012602 MOV (SP)+,R2 ;:POP STACK INTO R2
5969 023716 012601 MOV (SP)+,R1 ;:POP STACK INTO R1
5970 023720 012600 MOV (SP)+,RO ;:POP STACK INTO RO
5971 023722 000207 RTS PC ;:RETURN
5972 023724 176543 $HINUM: .WORD 176543
5973 023726 123456 $LONUM: .WORD 123456

```

.SBTTL INTEGER DIVIDE ROUTINE

```

5974 ;:*****
5975 ;*THIS ROUTINE WILL DIVIDE A 32-BIT TWO'S COMPLEMENT INTEGER
5976 ;*DIVIDEND BY A 16-BIT TWO'S COMPLEMENT INTEGER DIVISOR GIVING
5977 ;*A 16-BIT TWO'S COMPLEMENT INTEGER QUOTIENT AND A 16-BIT REMAINDER.
5978 ;*DIVISION WILL BE PERFORMED SO THAT THE REMAINDER IS OF THE
5979 ;*SAVE SIGN AS THE DIVIDEND.
5980 ;*CALL:
5981 ;* MOV LOW DIVIDEND,-(SP) ;:THE HIGH DIVIDEND MUST BE < 1/2
5982 ;* MOV HIGH DIVIDEND,-(SP); AS LARGE AS THE DIVISOR
5983 ;* MOV DIVISOR,-(SP)
5984 ;* JSR PC,$DIV
5985 ;* RETURN ;:QUOTIENT & REMAINDER ARE ON THE STACK
5986 ;* 'V'=0 IMPLIES NO ERROR
5987 ;* 'V'=1 IMPLIES ERROR OCCURRED
5988 ;* 'C'=0 DIVIDE OVERFLOW OCCURRED
5989 ;* 'C'=1 ATTEMPTED TO DIVIDE BY ZERO
5990 ;*
5991 ;*
5992 ;*
5993 ;*
5994 ;*
5995 ;* STACK NO ERROR OVERFLOW DIVIDE BY ZERO
5996 ;* -----
5997 ;* TOP REMAINDER ALL ZEROS ALL ONES

```

```

5998      ;*      +2      QUOTIENT      ALL ZEROS      ALL ONES
5999
6000      023730      $DIV:
6001      023730      104400      TRAP      ;;PUSH OLD PSW AND PC ON STACK
6002      023732      042716      000017      BIC      #17,(SP)      ;;STRIP AWAY CONDITION CODES
6003      023736      010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
6004      023740      010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
6005      023742      010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
6006      023744      010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
6007      023746      005046      CLR      -(SP)      ;;SAVE A PLACE FOR SIGNS
6008      023750      012746      000021      MOV      #17,-(SP)      ;;SETUP THE ITERATION COUNTER
6009      023754      016601      000024      MOV      24(SP),R1      ;;PICKUP THE DIVIDEND
6010      023760      016600      000022      MOV      22(SP),R0
6011      023764      100005      BPL      1$      ;;CHECK THE SIGN
6012      023766      105366      000003      DECB     3(SP)      ;;KEEP TRACK OF THE SIGN
6013      023772      005400      NEG      R0      ;;AND NEGATE THE ORIGINAL
6014      023774      005401      NEG      R1      ;;NUMBER
6015      023776      005600      SBC      R0
6016      024000      016602      000020      1$:      MOV      20(SP),R2      ;;PICKUP THE DIVISOR
6017      024004      002407      BLT      2$      ;;CHECK THE SIGN
6018      024006      003011      BGT      3$      ;;DIVISOR OF 0 IS A NO-NO
6019      024010      052766      000003      000014      BIS      #3,14(SP)      ;;SET 'V' & 'C'
6020      024016      012700      177777      MOV      #-1,R0      ;;SET REMAINDER TO ALL ONES
6021      024022      000424      BR      7$      ;;EXIT
6022      024024      005266      000002      2$:      INC      2(SP)      ;;KEEP TRACK OF DIVISORS SIGN
6023      024030      000401      BR      4$
6024      024032      005402      3$:      NEG      R2      ;;NEGATE THE ORIGINAL NUMBER
6025      024034      000241      4$:      CLC      ;;CLEAR 'C'
6026      024036      000405      BR      6$      ;;START FORMING QUOTIENT
6027      024040      006100      5$:      ROL      R0      ;;POSITION MSB'S
6028      024042      010003      MOV      R0,R3      ;;COPY
6029      024044      060203      ADD      R2,R3      ;;COMPARE DIVIDEND & DIVISOR
6030      024046      103001      BCC      6$      ;;BR IF DIVIDEND > DIVISOR
6031      024050      010300      MOV      R3,R0      ;;REMAINDER AFTER THIS LOOP
6032      024052      006101      6$:      ROL      R1      ;;QUOTIENT BIT ENTERS HERE
6033      024054      005316      DEC      (SP)      ;;DONE?
6034      024056      001370      BNF      5$      ;;BR IF NO
6035      024060      005701      TST      R1      ;;OVERFLOW?
6036      024062      100005      BPL      8$      ;;BR IF NO
6037      024064      052766      000002      000014      BIS      #2,14(SP)      ;;SET 'V' IN RETURN STATUS WORD
6038      024072      005000      CLR      R0      ;;SET REMAINDER TO ALL ZEROS
6039      024074      010001      7$:      MOV      R0,R1      ;;COPY REMAINDER INTO QUOTIENT
6040      024076      005726      8$:      TST      (SP)+      ;;CLEAR COUNTER FROM STACK
6041      024100      005716      TST      (SP)      ;;REMAINDER SIGN CORRECTION NEEDED?
6042      024102      002004      BGE      9$      ;;BR IF NO
6043      024104      005400      NEG      R0      ;;NEGATE REMAINDER
6044      024106      105066      000001      CLRB     1(SP)      ;;CLEAR SIGN
6045      024112      005316      DEC      (SP)      ;;BUT DON'T FORGET QUOTIENT
6046      024114      005726      9$:      TST      (SP)+      ;;QUOTIENT SIGN CORRECTION NEEDED?
6047      024116      001401      BEQ      10$      ;;BR IF NO
6048      024120      005401      NEG      R1      ;;NEGATE QUOTIENT
6049      024122      010166      000020      10$:      MOV      R1,20(SP)      ;;RETURN QUOTIENT AND
6050      024126      010066      000016      MOV      R0,16(SP)      ;;REMAINDER TO USER
6051      024132      012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
6052      024134      012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
6053      024136      012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
    
```

```

6054 024140 012600          MOV      (SP)+,R0          ;;POP STACK INTO R0
6055 024142 012666 000002  MOV      (SP)+,2(SP)      ;;SETUP TO RETURN CONDITION CODES
6056 024146 000002          RTI                          ;;RETURN
6057
6058          .SBTTL  *** PROGRAM SUBROUTINES ***
6059
6060          ;SET 'LPTAVL' TO THE PROPER STATE.
6061          ; LPTAVL = 0 IF NO LINE PRINTER AVAILABLE
6062          ; LPTAVL = 1 IF LINE PRINTER IS AVAILABLE
6063          ;CALL
6064          ;       JSR      PC,@#LP.AVL
6065          ;       RETURN
6066
6067 024150 005037 001230  LP.AVL: CLR      @#LPTAVL          ;START WITH NO PRINTER AVAILABLE
6068 024154 012737 024200 000004  MOV      #1$,@#ERRVEC      ;SETUP THE TIMEOUT VECTOR
6069 024162 005037 000006          CLR      @#ERRVEC+2
6070 024166 005777 155226          TST      @LPS              ;IS THERE A LINE PRINTER?
6071 024172 005237 001230          INC      @#LPTAVL          ;YES--SET AVAILABLE SWITCH
6072 024176 000401          BR      2$
6073 024200 022626          1$:  CMP      (SP)+,(SP)+      ;NO--POP STACK
6074 024202 012737 000006 000004  2$:  MOV      #ERRVEC+2,@#ERRVEC ;RESTORE TIMEOUT VECTOR
6075 024210 000207          RTS      PC                ;RETURN
6076
6077          ;THIS ROUTINE WILL DETERMINE IF THERE IS A CLOCK ON THE SYSTEM
6078          ;AND IF THERE IS IT WILL SETUP THE VECTOR AND START THE CLOCK
6079          ;"CLKSTA" WILL INDICATE THE CLOCK TYPE
6080          ; 0= NO CLOCK
6081          ; +1= KW11-P
6082          ; -1= KW11-L
6083          ;THIS ROUTINE WILL ALSO SETUP "TICKMS" (TIME
6084          ;PER CLOCK TICK IN MILLISECONDS) AND "TICKUS"
6085          ;(TIME PER CLOCK TICK IN MICROSECONDS) AS
6086          ;PER SW00.
6087          ;SW00=0 -- 60HZ
6088          ;SW00=1 -- 50HZ
6089          ;CALL
6090          ;       JSR      PC,@#ST.CLK
6091          ;       RETURN
6092
6093 024212 010146          ST.CLK: MOV      R1,-(SP)          ;SAVE R1
6094 024214 012701 000006          MOV      #ERRVEC+2,R1      ;SAVE AND SETUP TIMEOUT VECTOR
6095 024220 011146          MOV      (R1),-(SP)
6096 024222 005011          CLR      (R1)              ;LEVEL 0
6097 024224 014146          MOV      -(R1),-(SP)
6098 024226 012711 024256          MOV      #1$,(R1)          ;GO TO 1$ ON TIMEOUT
6099 024232 005037 001244          CLR      CLKSTA           ;SET CLOCK STATUS TO NO CLOCK
6100 024236 005777 155136          TST      @PKCS            ;IS THERE A KW11-P?
6101 024242 012737 000001 001244  MOV      #1,CLKSTA          ;YES--SET STATUS TO KW11-P
6102 024250 004737 024360          JSR      PC,ST.PCLK        ;START THE KW11-P
6103 024254 000414          BR      3$                 ;GO TO EXIT
6104 024256 022626          1$:  CMP      (SP)+,(SP)+      ;CLEAN UP THE STACK
6105 024260 012711 024304          MOV      #2$,(R1)          ;IF TIMEOUT GO TO 2$
6106 024264 005777 155122          TST      @LKS              ;IS THERE A KW11-L?
6107 024270 012737 177777 001244  MOV      #-1,CLKSTA         ;YES-- SET STATUS TO KW11-L
6108 024276 004737 024422          JSR      PC,ST.LCLK        ;START THE KW11-L
6109 024302 000401          BR      3$                 ;EXIT

```



```

6110 024304 022626      2$:  CMP      (SP)+,(SP)+      ;CLEAN UP THE STACK
6111 024306 01262i     3$:  MOV      (SP)+,(R1)+      ;RESTORE THE TIMEOUT VECTOR
6112 024310 012621      MOV      (SP)+,(R1)+
6113 024312 012601      MOV      (SP)+,R1          ;RESTORE R1
6114 024314 032737 000100 001220  BIT      #SW06,@#C.SWR      ;50HZ OR 60HZ?
6115 024322 001407      BEQ      4$                ;BRANCH IF 60
6116 024324 012737 000020 001246  MOV      #20,@#TICKMS      ;SETUP TIME PER
6117 024332 012737 047040 001250  MOV      #20000.,@#TICKUS  ;TICK FOR 50HZ
6118 024340 000406      BR       5$
6119 024342 012737 000016 001246  4$:  MOV      #16,@#TICKMS    ;SETUP TIME PER
6120 024350 012737 040432 001250  MOV      #16666.,@#TICKUS  ;TICK FOR 60HZ
6121 024356 000207      5$:  RTS      PC              ;RETURN
6122
6123 024360      ST.PCLK:
6124 024360 032737 000040 001220  BIT      #SW05,@#C.SWR      ;ALLOW SOFTWARE TIMEOUTS?
6125 024366 001014      BNE      1$                ;NO--BRANCH
6126 024370 012777 024456 154776  MOV      #SRVCLK,@PKV      ;SETUP THE KW11-P VECTOR
6127 024376 012777 000300 154772  MOV      #300,@PKV+2
6128 024404 012777 000001 154770  MOV      #1,@PKB           ;COUNT ONE TICK
6129 024412 012777 000115 154760  MOV      #115,@PKCS        ;'INT.EN.',COUNT DOWN'', 'MODE 1 (REPEAT)'',
6130                                     ;'LINE FREQ'', AND 'RUN''
6131 024420 000207      1$:  RTS      PC              ;RETURN
6132
6133 024422      ST.LCLK:
6134 024422 032737 000040 001220  BIT      #SW05,@#C.SWR      ;ALLOW SOFTWARE TIMEOUTS?
6135 024430 001011      BNE      1$                ;NO--BRANCH
6136 024432 012777 024456 154746  MOV      #SRVCLK,@LKV      ;SETUP THE KW11-L VECTOR
6137 024440 012777 000300 154742  MOV      #300,@LKV+2
6138 024446 012777 000100 154736  MOV      #100,@LKS         ;START THE KW11-L
6139 024454 000207      1$:  RTS      PC              ;RETURN
6140
6141 024456 013746 001246  SRVCLK: MOV      @#TICKMS,-(SP) ;TIME PER TICK IN MILLISECONDS
6142 024462 004737 041102  JSR      PC,@#RPTMR        ;COUNT THE ELAPSED TIME
6143 024466 000002      RTI      ;RETURN AFTER INTERRUPT
6144
6145                                     ;THIS ROUTINE SETS UP DEFAULT PARAMETER VALUES WHEN THE PROGRAM IS
6146                                     ;STARTED OR WHEN THE VALUE OF BIT00 IN 'C.SWR' IS CHANGED.
6147                                     ;CALL
6148                                     ; JSR      PC,LODFLT
6149                                     ; RETURN
6150
6151 024470      LODFLT:
6152 024470 010046      MOV      R0,-(SP)          ;;PUSH R0 ON STACK
6153 024472 010146      MOV      R1,-(SP)          ;;PUSH R1 ON STACK
6154 024474 010246      MOV      R2,-(SP)          ;;PUSH R2 ON STACK
6155 024476 010346      MOV      R3,-(SP)          ;;PUSH R3 ON STACK
6156 024500 012737 176777 001234  MOV      #176777,TSTNMS    ;SELECT TESTS 0-10, 12-17
6157 024506 012737 000001 001236  MOV      #1,TSTNMS+2       ;SET SELECT BIT FOR TEST 20
6158 024514 012700 001664      MOV      #DFLT,R0         ;DEFAULT PARAMETERS POINTER
6159 024520 012701 002330      MOV      #PRMO,R1        ;TABLE POINTER
6160 024524 010102      MOV      R1,R2           ;STOP ADDRESS
6161 024526 012021      1$:  MOV      (R0)+,(R1)+      ;MOVE DEFAULT PARAMETERS INTO
6162 024530 020002      CMP      R0,R2          ;RUN TIME TABLES ** DONE?
6163 024532 103775      BLO      1$             ;NO--BRANCH
6164 024534 012700 003504      MOV      #PAT8,R0        ;PATO DEFAULTS TO PATTERN 8
6165 024540 012701 003104      MOV      #PATO,R1

```

```

6166 024544 012021      2$:  MOV      (R0)+,(R1)+
6167 024546 020027 003544    CMP      R0,#PAT9
6168 024552 103774      BLO      2$
6169 024554 032737 000001 001220    BIT      #BIT00,C.SWR      ;16 BIT MODE ?
6170 024562 001012      BNE      3$                ;BR IF 18
6171 024564 012737 000025 001630    MOV      #21.,PRMLMT+22    ;SET 'FS' LIMIT TO 21.
6172 024572 012737 000025 001632    MOV      #21.,PRMLMT+24    ;SET 'LS' LIMIT TO 21.
6173 024600 012737 165000 001352    MOV      #-<256.*22.>,TRCKWC ;WORD COUNT FOR A 16 BIT TRACK
6174 024606 000411      BR       4$                ;CONTINUE
6175 024610 012737 000023 001630    3$:  MOV      #19.,PRMLMT+22    ;SET 'FS' LIMIT TO 19.
6176 024616 012737 000023 001632    MOV      #19.,PRMLMT+24    ;SET 'LS' LIMIT TO 19.
6177 024624 012737 166000 001352    MOV      #-<256.*20.>,TRCKWC ;WORD COUNT FOR COUNT FOR AN 18 BIT TRACK
6178 024632 012701 001536    4$:  MOV      #PRMPT,R1        ;ADDRESS OF PARAMETER POINTER TABLE
6179 024636 005711    5$:  TST      (R1)            ;END OF THE TABLE ?
6180 024640 001425      BEQ      8$                ;BR IF END
6181 024642 032731 002000    BIT      #BIT10,@(R1)+     ;'LS' SELECTED ?
6182 024646 001773      BEQ      5$                ;BR IF NOT
6183 024650 016102 177776    MOV      -2(R1),R2        ;PARAMETER TABLE ADDRESS
6184 024654 011246    MOV      (R2),-(SP)       ;PARAMETER ALLOCATION BITS
6185 024656 012703 000013    MOV      #11.,R3          ;NUMBER OF PARAMETERS (MAXIMUM) BEFORE 'LS'
6186 024662 006216    6$:  ASR      (SP)            ;COUNT THE PARAMETER
6187 024664 103002      BCC      7$                ;BR IF NOT USED
6188 024666 062702 000002    ADD      #2,R2            ;INCREMENT THE PARAMETER TABLE ADDRESS
6189 024672 005303    7$:  DEC      R3              ;COUNT THE PARAMETER
6190 024674 001372      BNE      6$                ;BR IF NOT THERE YET
6191 024676 005726      TST      (SP)+            ;CORRECT THE STACK POINTER
6192 024700 021237 001630    CMP      (R2),PRMLMT+22    ;IS 'LS' TOO LARGE FOR THE MODE SELECTED ?
6193 024704 101754      BLOS     5$                ;BR IF NOT
6194 024706 013712 001630    MOV      PRMLMT+22,(R2)    ;RESET VALUE FOR MODE USED
6195 024712 000751      BR       5$                ;CONTINUE
6196 024714    8$:
6197 024714 012603      MOV      (SP)+,R3         ;;POP STACK INTO R3
6198 024716 012602      MOV      (SP)+,R2         ;;POP STACK INTO R2
6199 024720 012601      MOV      (SP)+,R1         ;;POP STACK INTO R1
6200 024722 012600      MOV      (SP)+,R0         ;;POP STACK INTO R0
6201 024724 000207      RTS      PC               ;RETURN
6202
6203      ;THIS ROUTINE FILLS THE PARAMETER TABLE THE CURRENT TEST.
6204      ;CALL
6205      ;      MOV      #TESTNUM,$STNM ;LOAD THE TEST NUMBER
6206      ;      JSR      PC,LODPRM
6207      ;      RETURN
6208
6209      LODPRM:
6210 024726 010146      MOV      R1,-(SP)         ;;PUSH R1 ON STACK
6211 024730 010246      MOV      R2,-(SP)         ;;PUSH R2 ON STACK
6212 024732 010346      MOV      R3,-(SP)         ;;PUSH R3 ON STACK
6213 024734 010446      MOV      R4,-(SP)         ;;PUSH R4 ON STACK
6214 024736 005004      CLR      R4               ;CLEAR R4
6215 024740 113704 001102    MOVB     $STNM,R4         ;GET THE TEST NUMBER
6216 024744 006304      ASL      R4               ;SETUP TO ADDRESS WORDS
6217 024746 016401 001536    MOV      PRMPT(R4),R1     ;GET THE TEST'S PARAMETER TABLE ADDRESS
6218 024752 012702 001504    MOV      #PRM,R2         ;PARAMETER EXECUTION TABLE
6219 024756 005003      CLR      R3               ;R3 IS USED AS A COUNTER
6220 024760 013704 001254    MOV      CHKDRV,R4        ;DRIVE'S ADDRESS
6221 024764 012122      MOV      (R1)+,(R2)+     ;PARAMETER SPECIFIER

```

```

6222 024766 006237 001504      1$: ASR PRM ;THIS PARAMETER USED IN THE TEST ?
6223 024772 103002             BCC 2$ ;BR IF NOT
6224 024774 012122             MOV (R1)+,(R2)+ ;LOAD THE VALUE
6225 024776 000401             BR 3$ ;CONTINUE
6226 025000 005022             2$: CLR (R2)+ ;CLEAR THE UNUSED PARAMETER LOCATION
6227 025002 005203             3$: INC R3 ;COUNT THE POSITION IN THE OUTPUT TABLE
6228 025004 020327 000014     CMP R3,#12. ;FINISHED ?
6229 025010 001430             BEQ 6$ ;BR IF YES
6230 025012 020327 000003     CMP R3,#3 ;DOING THE CYLINDER ADDRESSES ?
6231 025016 001363             BNE 1$ ;BR IF NOT
6232 025020 132764 000004 034514 BITB #BIT02,DRV TYP(R4) ;RP06 ?
6233 025026 001016             BNE 5$ ;BR IF IT IS
6234 025030 062703 000002     ADD #2,R3 ;COUNT THE BYPASSED PARAMETERS (FC' & LC')
6235 025034 006237 001504     ASR PRM ;SHIFT THE COUNTER
6236 025040 103002             BCC 4$ ;BR IF FC' IS NOT USED
6237 025042 062701 000002     ADD #2,R1 ;MOVE THE INPUT POINTER
6238 025046 006237 001504     4$: ASR PRM ;COUNT THE PARAMETER
6239 025052 103345             BCC 1$ ;BR IF LC' NOT USED
6240 025054 062701 000002     ADD #2,R1 ;MOVE THE INPUT PINTER
6241 025060 000742             BR 1$ ;KEEP GOING
6242 025062 000741             BR 1$ ;KEEP GOING
6243 025064 162702 000004     5$: SUB #4,R2 ;BACKUP THE OUTPUT POINTER
6244 025070 000736             BR 1$ ;KEEP GOING
6245 025072             6$:
6246 025072 012604             MOV (SP)+,R4 ;;POP STACK INTO R4
6247 025074 012603             MOV (SP)+,R3 ;;POP STACK INTO R3
6248 025076 012602             MOV (SP)+,R2 ;;POP STACK INTO R2
6249 025100 012601             MOV (SP)+,R1 ;;POP STACK INTO R1
6250 025102 000207             RTS PC ;RETURN

```

```

6251
6252 ;THIS ROUTINE LOADS A READ HEADER AND DATA COMMAND OR A SEEK COMMAND.
6253 ;INTO DPB.B+2 AND DPB.C+2, DEPENDING ON THE STATE OF 'CONTROL SWITCH'
6254 ;BIT07.
6255 ;CALL
6256 ;
6257 ; JSR PC,@#LDCMD
6258 ; RETURN

```

```

6259 025104 032737 000200 001220 LDCMD: BIT #SW07,@#C.SWR ;DO EXPLICIT SEEKS?
6260 025112 001007             BNE 1$ ;YES--BRANCH
6261 025114 012737 000173 004126 MOV #READHD,@#DPB.B+2 ;NO--SET UP FOR READ HEADER AND
6262 025122 012737 000173 004146 MOV #READHD,@#DPB.C+2 ;DATA COMMAND
6263 025130 000406             BR 2$
6264 025132 012737 000105 004126 1$: MOV #SEEK,@#DPB.B+2 ;SETUP FOR SEEK COMMAND
6265 025140 012737 000105 004146 MOV #SEEK,@#DPB.C+2
6266 025146 000207             2$: RTS PC

```

```

6267
6268 ;THIS ROUTINE WILL CALL THE RP04/5/6 DRIVER AND THEN WAIT ON THE FUNCTION
6269 ;TO COMPLETE. IF AN ERROR OCCURS IT IS REPORTED.
6270 ;CALL
6271 ;
6272 ; FILL 'DPB' WITH COMMAND INFORMATION
6273 ; JSR RC,@#CALL.A
6274 ; RETURN

```

```

6275 025150 005037 001206 CALL.A: CLR @#$ESCAPE ;NO ESCAPE ADDRESS
6276 025154 004037 035450 JSR R0,@#RP04 ;CALL RP04 DRIVER
6277 025160 004104 DPB.A

```

```

6278 025162 000772
6279 025164 005737 004122 1$: BR CALL.A
6280 025170 001775 BEQ 1$ ;DONE?
6281 025172 100032 BPL 3$ ;NO--LOOP
6282 025174 012737 025250 001206 MOV #2$, $ESCAPE ;BRANCH IF NO ERROR
6283 025202 013737 004116 001270 MOV @#DPB.A+12,@#CYL.DS ;ESCAPE TO 2$ ON ERROR
6284 025210 113737 004115 001274 MOV @#DPB.A+11,@#TRK.DS ;CYLINDER
6285 025216 113737 004114 001272 MOV @#DPB.A+10,@#SEC.DS ;TRACK
6286 025224 012746 004122 MOV #DPB.A+16,-(SP) ;TRACK
6287 025230 004737 026364 JSR PC,@#ERINDX ;SECTOR
6288 025234 062607 ADD (SP)+,PC ;STATUS/ERROR INDICATOR ADDRESS
6289 025236 104041 ERROR 41 ;FORM DISPATCH INDEX
6290 025240 104042 ERROR 42 ;REPORT PROPER ERROR
6291 025242 104043 ERROR 43 ;PARITY ERROR
6292 025244 104044 ERROR 44 ;UNSAFE ERROR
6293 025246 104045 ERROR 45 ;NON-I/O ERROR
6294 025250 013746 004122 2$: MOV DPB.A+16,-(SP) ;I/O ERROR
6295 025254 004737 026324 JSR PC,LOP.CK ;STATUS WORD
6296 025260 000200 3$: RTS RO ;SEE IF LOOP, ABORT, OR CONTINUE
;RETURN

```

```

;THIS ROUTINE IS THE SAME AS "CALL.A" EXCEPT FOR THE DPB USED AND IF
;THE COMMAND IS A READ HEADER AND DATA THE HEADER (CYLINDER, TRACK,
;AND SECTOR) READ IS CHECKED FOR VALIDITY.

```

```

6297
6298 ;CALL
6299 ; FILL DPB
6300 ; JSR RO,@#CALL.B
6301 ;
6302 ;
6303 ;
6304 ;
6305 ;

```

```

6306 025262 005037 001206 CALL.B: CLR @#$ESCAPE ;NO ESCAPE ADDRESS
6307 025266 004037 035450 JSR RO,@#RP04 ;CALL RP04 DRIVER
6308 025272 004124 DPB.B
6309 025274 000772 BR CALL.B
6310 025276 005737 004142 1$: TST DPB.B+16 ;DONE?
6311 025302 001775 BEQ 1$ ;NO--BRANCH
6312 025304 100042 BPL 4$ ;BRANCH IF NO ERROR
6313 025306 012737 025400 001206 MOV #3$, $ESCAPE ;ESCAPE TO 3$ ON ERROR
6314 025314 013737 004136 001270 MOV @#DPB.B+12,@#CYL.DS ;CYLINDER
6315 025322 113737 004135 001274 MOV @#DPB.B+11,@#TRK.DS ;TRACK
6316 025330 113737 004134 001272 MOV @#DPB.B+10,@#SEC.DS ;SECTOR
6317 025336 012746 004142 MOV #DPB.B+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
6318 025342 004737 026364 JSR PC,@#ERINDX ;FORM DISPATCH INDEX
6319 025346 062607 ADD (SP)+,PC ;REPORT PROPER ERROR
6320 025350 104041 ERROR 41 ;
6321 025352 104042 ERROR 42 ;PARITY ERROR
6322 025354 104043 ERROR 43 ;UNSAFE ERROR
6323 025356 104044 ERROR 44 ;NON-I/O ERROR
6324 025360 005737 004220 TST RP.REG+RPER1 ;DRIVE ERROR ?
6325 025364 001404 BEQ 2$ ;BR IF NOT
6326 025366 032737 177677 004220 BIT #^C100,RP.REG+RPER1 ;SEE IF ONLY 'HCE' SET
6327 025374 001406 BEQ 4$ ;BR IF IT IS
6328 025376 104045 2$: ERROR 45 ;I/O ERROR
6329 025400 013746 004142 3$: MOV DPB.B+16,-(SP) ;STATUS WORD
6330 025404 004737 026324 JSR PC,LOP.CK ;SEE IF LOOP, ABORT, OR CONTINUE
6331 025410 000410 BR 5$ ;CHECK FOR STALL
6332 025412 123727 004126 000173 4$: CMPB @#DPB.B+2,#READHD ;DOING IMPLIED SEEKS?
6333 025420 001004 BNE 5$ ;NO--BRANCH

```

```

6334 025422 004037 026644      JSR    RO,@#VERIFY      ;YES--GO CHECK THE DATA
6335 025426 004134      DPB.B+10
6336 025430 000407      BR     6$              ;ERROR DURING VERIFY
6337 025432 032737 040000 001220 5$:  BIT    #SW14,@#C.SWR   ;STALL?
6338 025440 001403      BEQ    6$              ;NO--BRANCH
6339 025442 004037 026502      JSR    RO,@#STALL      ;YES--CALL STALL ROUTINE
6340 025446 001354      .WORD  STALL1          ;STALL TIME POINTER
6341 025450 000200      6$:   RTS    RO        ;RETURN
6342
6343      ;THIS ROUTINE IS THE SAME AS "CALL.B" EXCEPT FOR THE DPB USED.
6344      ;CALL
6345      ;
6346      ;   FILL DPB
6347      ;   JSR    RO,@#CALL.C
6348      ;   RETURN
6349 025452 005037 001206      CALL.C: CLR    @#SESCAPE ;NO ESCAPE ADDRESS
6350 025456 004037 035450      JSR    RO,@#RPO4      ;CALL RPO4 DRIVER
6351 025462 004144      DPB.C
6352 025464 000772      BR     CALL.C
6353 025466 005737 004162      1$:   TST    @#DPB.C+16  ;DONE?
6354 025472 001775      BEQ    1$              ;NO--LOOP
6355 025474 100042      BPL    4$              ;YES--BRANCH IF NO ERROR
6356 025476 012737 025570 001206  MOV    #3$,SESCAPE    ;:ESCAPE TO 3$ ON ERROR
6357 025504 013737 004156 001270  MOV    @#DPB.C+12,@#CYL.DS ;CYLINDER
6358 025512 113737 004155 001274  MOVB   @#DPB.C+11,@#TRK.DS ;TRACK
6359 025520 113737 004154 001272  MOVB   @#DPB.C+10,@#SEC.DS ;SECTOR
6360 025526 012746 004162      MOV    #DPB.C+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
6361 025532 004737 026364      JSR    PC,@#ERINDX   ;FORM DISPATCH INDEX
6362 025536 062607      ADD    (SP)+,PC      ;REPORT PROPER ERROR
6363 025540 104041      ERROR  41            ;
6364 025542 104042      ERROR  42            ;PARITY ERROR
6365 025544 104043      ERROR  43            ;UNSAFE ERROR
6366 025546 104044      ERROR  44            ;NON-I/O ERROR
6367 025550 005737 004220      TST    RP.REG+RPER1  ;DRIVE ERROR ?
6368 025554 001404      BEQ    2$              ;BR IF NOT
6369 025556 032737 177677 004220  BIT    #^C100,RP.REG+RPER1 ;SEE IF ONLY 'HCE' SET
6370 025564 001406      BEQ    4$              ;BR IF IT IS
6371 025566 104045      2$:   ERROR  45            ;I/O ERROR
6372 025570 013746 004162      3$:   MCV    DPB.C+16,-(SP) ;STATUS WORD
6373 025574 004737 026324      JSR    PC,LOP.CK     ;SEE IF LOOP, ABORT, OR CONTINUE
6374 025600 000410      BR     5$
6375 025602 123727 004146 000173 4$:   CMPB   @#DPB.C+2,#READHD ;DOING IMPLIED SEEK?
6376 025610 001004      BNE    5$              ;NO--EXIT
6377 025612 004037 026644      JSR    RO,@#VERIFY   ;YES--CHECK THE DATA
6378 025616 004154      DPB.C+10
6379 025620 000407      BR     6$              ;ERROR DURING VERIFY
6380 025622 032737 040000 001220 5$:  BIT    #SW14,@#C.SWR   ;STALL?
6381 025630 001403      BEQ    6$              ;NO--BRANCH
6382 025632 004037 026502      JSR    RO,@#STALL    ;YES--CALL STALL ROUTINE
6383 025636 001354      .WORD  STALL1        ;STALL TIME POINTER
6384 025640 000200      6$:   RTS    RO
6385
6386
6387      ;THIS ROUTINE IS THE SAME AS "CALL.A" EXCEPT FOR THE DPB USED AND
6388      ;ON AN ERROR LOCATION "ERR.CT" IS EXAMINED. IF ERR.CT IS EQUAL TO
6389      ;SERFLG EXIT IS TO THE NEXT TEST.

```

```

6390      :CALL
6391      :      FILL DPB
6392      :      JSR      RO,@#DRVCAL
6393      :      RETURN
6394
6395 025642 005037 001206  DRVCAL: CLR      @#$ESCAPE      ;NO ESCAPE ADDRESS
6396 025646 005037 001334  CLR      @#WCEFLG      ;CLEAR WRITE CHECK ERROR FLAG
6397 025652 004037 035450  JSR      RO,@#RP04      ;CALL RP04 DRIVER
6398 025656 004164  DTADPB
6399 025660 000770  BR      DRVCAL
6400 025662 005737 004202  DRVCL1: TST     @#DTADPB+16 ;DONE
6401 025666 001775  BEQ     DRVCL1          ;NO--LOOP
6402 025670 100402  BMI     1$             ;BR IF ERRORS
6403 025672 000137 026304  JMP     10$           ;NO ERRORS
6404 025676
6405 025676 012737 025752 001206 1$:  MOV     #2$, $ESCAPE    ;; ESCAPE TO 2$ ON ERROR
6406 025704 013737 004176 001270  MOV     @#DTADPB+12,@#CYL.DS ;CYLINDER
6407 025712 113737 004175 001274  MOVB   @#DTADPB+11,@#TRK.DS ;TRACK
6408 025720 113737 004174 001272  MOVB   @#DTADPB+10,@#SEC.DS ;SECTOR
6409 025726 012746 004202  MOV     #DTADPB+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
6410 025732 004737 026364  JSR     PC,@#ERINDX      ;FORM DISPATCH INDEX
6411 025736 062607  ADD     (SP)+,PC         ;REPORT PROPER ERROR
6412 025740 104041  ERROR  41              ;
6413 025742 104042  ERROR  42              ;PARITY ERROR
6414 025744 104043  ERROR  43              ;UNSAFE ERROR
6415 025746 104044  ERROR  44              ;NON-I/O ERROR
6416 025750 104045  ERROR  45              ;I/O ERROR
6417 025752 122737 000020 001102 2$:  CMPB   #20,@#$TSTNM     ;TEST 20?
6418 025760 001137  BNE     8$             ;NO--BRANCH
6419 025762 013746 004202  MOV     DTADPB+16,-(SP) ;STATUS WORD
6420 025766 004737 026324  JSR     PC,LOP.CK       ;SEE IF LOOP, ABORT, OR CONTINUE
6421 025772 122737 000151 004166  CMPB   #WRCKD,@#DTADPB+2 ;DOING A WRITE CHECK?
6422 026000 001133  BNE     12$           ;NO--BRANCH
6423 026002 032737 040000 004214  BIT     #BIT14,@#RF.REG+10 ;IS 'WCE'=1?
6424 026010 001527  BEQ     12$           ;NO--BRANCH
6425 026012 032777 000020 153120  BIT     #SW04,@SWR      ;INHIBIT WRITES?
6426 026020 001123  BNE     12$           ;YES--BRANCH
6427 026022 112737 000161 004166  MOVB   #WRITE,@#DTADPB+2 ;SETUP FOR A WRITE
6428 026030 005037 001206  CLR     @#$ESCAPE      ;NO ESCAPE ADDRESS
6429 026034 004037 035450  JSR     RO,@#RP04      ;DO THE WRITE
6430 026040 004164  DTADPB
6431 026042 000240  NOP
6432 026044 005737 004202 3$:  TST     @#DTADPB+16     ;DONE?
6433 026050 001775  BEQ     3$             ;NO--LOOP
6434 026052 100026  BPL     4$             ;YES--BRANCH IF NO ERROR
6435 026054 012737 026260 001206  MOV     #8$, $ESCAPE    ;; ESCAPE TO 8$ ON ERROR
6436 026062 013737 004176 001270  MOV     @#DTADPB+12,@#CYL.DS ;CYLINDER
6437 026070 113737 004175 001274  MOVB   @#DTADPB+11,@#TRK.DS ;TRACK
6438 026076 113737 004174 001272  MOVB   @#DTADPB+10,@#SEC.DS ;SECTOR
6439 026104 012746 004202  MOV     #DTADPB+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
6440 026110 004737 026364  JSR     PC,@#ERINDX      ;FORM DISPATCH INDEX
6441 026114 062607  ADD     (SP)+,PC         ;REPORT PROPER ERROR
6442 026116 104041  ERROR  41              ;
6443 026120 104042  ERROR  42              ;PARITY ERROR
6444 026122 104043  ERROR  43              ;UNSAFE ERROR
6445 026124 104044  ERROR  44              ;NON-I/O ERROR

```

```

6446 026126 104045          ERROR 45          ;I/O ERROR
6447 026130 112737 000151 004166 4$:  MOVB  #WRCKD,@#DTADPB+2 ;COMMAND=WRITE CHECK DATA
6448 026136 004037 035450          JSR   RO,@#RPO4      ;DO THE WRITE CHECK
6449 026142 004164          DTADPB
6450 026144 000240          NOP
6451 026146 005737 004202          5$:  TST   @#DTADPB+16    ;DONE?
6452 026152 001775          BEQ   5$            ;NO--LOOP
6453 026154 100410          BMI   7$            ;YES--BRANCH IF ERROR
6454 026156 004037 035450          JSR   RO,@#RPO4      ;DO A 2ND WRITE CHECK
6455 026162 004164          DTADPB
6456 026164 000240          NOP
6457 026166 005737 004202          6$:  TST   @#DTADPB+16    ;DONE?
6458 026172 001775          BEQ   6$            ;NO--LOOP
6459 026174 100043          BPL   10$           ;YES--BRANCH IF NO ERROR
6460 026176 012737 000001 001334 7$:  MOV   #1,@#WCEFLG    ;SET THE WRITE CHECK ERROR FLAG
6461 026204 012737 026260 001206      MOV   #8$, $ESCAPE   ;ESCAPE TO 8$ ON ERROR
6462 026212 013737 004176 001270      MOV   @#DTADPB+12,@#CYL.DS ;CYLINDER
6463 026220 113737 004175 001274      MOVB  @#DTADPB+11,@#TRK.DS ;TRACK
6464 026226 113737 004174 001272      MOVB  @#DTADPB+10,@#SEC.DS ;SECTOR
6465 026234 012746 004202          MOV   #DTADPB+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
6466 026240 004737 026364          JSR   PC,@#ERINDX   ;FORM DISPATCH INDEX
6467 026244 062607          ADD   (SP)+,PC      ;REPORT PROPER ERROR
6468 026246 104041          ERROR 41           ;
6469 026250 104042          ERROR 42           ;PARITY ERROR
6470 026252 104043          ERROR 43           ;UNSAFE ERROR
6471 026254 104044          ERROR 44           ;NON-I/O ERROR
6472 026256 104046          ERROR 46           ;FATAL WRITE CHECK
6473 026260 013746 004202          8$:  MOV   DTADPB+16,-(SP) ;STATUS WORD
6474 026264 004737 026324          JSR   PC,LOP.CK     ;SEE IF LOOP, ABORT, OR CONTINUE
6475 026270 123737 001364 001103 12$:  CMPB  @#ERR.CT,@#SERFLG ;GO TO NEXT TEST?
6476 026276 101002          BHI   10$           ;NO--BRANCH
6477 026300 013700 001252          9$:  MOV   @#BYPASS,RO   ;YES--GET EXIT ADDRESS
6478 026304 032737 040000 001220 10$:  BIT   #SW14,@#C.SWR ;STALL?
6479 026312 001403          BEQ   11$           ;NO--BRANCH
6480 026314 004037 026502          JSR   RO,@#STALL    ;YES--CALL STALL ROUTINE
6481 026320 001356          .WORD STALL2       ;STALL TIME POINTER
6482 026322 000200          11$: RTS   RO
6483
6484          ;THIS SUBROUTINE CHECK FOR LOOP, ABORT, OR CONTINUE SWITCHES AFTER
6485          ;ERRORS 41, 42, 43, 44, 45, AND 46.
6486          ;CALL
6487          ;   MOV   DTA+16,-(SP) ;STATUS WORD FROM DPB IN USE
6488          ;   JSR   PC,LOP.CK
6489          ;   RETURN
6490
6491 026324 032777 001000 152606 LOP.CK: BIT   #SW9,@SWR    ;LOOP ON ERROR
6492 026332 001402          BEQ   1$            ;BR IF NOT
6493 026334 000177 152550          JMP   @#LPPER       ;START AT THE LOOP ADDRESS
6494 026340 005037 001206          1$:  CLR   $ESCAPE       ;CLEAR ERROR ESCAPE FLAG
6495 026344 032766 072006 000002      BIT   #BIT14:BIT13:BIT12:BIT10:BIT02:BIT01,2(SP) ;CHECK ERROR TYPE
6496 026352 001402          BEQ   2$            ;BR IF DRIVE NOT OFFLINE, UNLOADED, OR
6497          ;PERSISTENT UNSAFE OR FATAL MASSBUS PARITY
6498 026354 000137 017502          JMP   $EOP          ;TERMINATE DRIVE
6499 026360 012616          2$:  MOV   (SP)+,(SP)   ;ADJUST RETURN ADDRESS
6500 026362 000207          RTS   PC
6501

```

```

6502      ;THIS ROUTINE FORMS AN INDEX THAT WILL BE USED TO DISPATCH
6503      ;TO THE PROPER ERROR CALL.  THE INDEX IS FORMED BY EXAMINING
6504      ;THE STATUS/ERROR INDICATOR OF THE APPLICABLE DPB.
6505      ;INDEX          STATUS/ERROR
6506      ;-----
6507      ; 0 BIT14!BIT13!BIT08!BIT01
6508      ; 2 BIT11!BIT10!BIT02
6509      ; 4 BIT12!BIT04
6510      ; 6 BIT05!BIT03!<BIT09 & COMMAND=NON-I/O>
6511      ;10 BIT06!<BIT09 & COMMAND=I/O>
6512      ;CALL
6513      ;      JSR      #DPB+16,-(SP)      ;ADDRESS OF STATUS/ERROR INDICATOR
6514      ;      JSR      PC,@#ERINDX      ;FORM INDEX
6515      ;      RETURN     ;INDEX IS ON THE STACK
6516
6517      026364 010046      ERINDX: MOV      R0,-(SP)      ;SAVE R0
6518      026366 010146      MOV      R1,-(SP)      ;SAVE R1
6519      026370 016600 000006  MOV      6(SP),R0      ;GET STATUS/ERROR INDICATOR POINTER
6520      026374 011037 001260  MOV      (R0),@#SVSTAT ;SAVE THE STATUS/ERROR INDICATOR
6521      026400 005001      CLR      R1      ;START INDEX AT ZERO
6522      026402 032710      BIT      (PC)+,(R0)    ;FORM INDEX OF 0?
6523      026404 020402      .WORD   BIT13!BIT08!BIT01
6524      026406 001027      BNE     5$      ;YES--BRANCH
6525      026410 032710      BIT      (PC)+,(R0)    ;FORM PARITY ERROR OR PORT REQUEST INDEX (2)?
6526      026412 006004      .WORD   BIT11!BIT10!BIT02
6527      026414 001023      BNE     4$      ;YES--BRANCH
6528      026416 032710      BIT      (PC)+,(R0)    ;FORM UNSAFE INDEX (4)?
6529      026420 050020      .WORD   BIT14!BIT12!BIT04
6530      026422 001017      BNE     3$      ;YES--BRANCH
6531      026424 032710      BIT      (PC)+,(R0)    ;FORM NON-I/O ERROR INDEX (6)?
6532      026426 000050      .WORD   BIT05!BIT03
6533      026430 001013      BNE     2$      ;YES--BRANCH
6534      026432 032710      BIT      (PC)+,(R0)    ;FORM I/O ERRCR INDEX (10)?
6535      026434 000100      .WORD   BIT06
6536      026436 001007      BNE     1$      ;YES--BRANCH
6537      026440 032710      BIT      (PC)+,(R0)    ;SOFTWARE TIMEOUT?
6538      026442 001000      .WORD   BIT09
6539      026444 001410      BEQ     5$      ;NO--FORM INDEX OF 0
6540      026446 122760 000150 177762  CMPB   #150,-16(R0)    ;YES--I/O?
6541      026454 003001      BGT     2$      ;NO--BRANCH
6542      026456 005201      1$:   INC     R1      ;INDEX=10---ERROR=45 OR 46
6543      026460 005201      2$:   INC     R1      ;INDEX=6---ERROR=44
6544      026462 005201      3$:   INC     R1      ;INDEX=4---ERROR=43
6545      026464 005201      4$:   INC     R1      ;INDEX=2---ERROR=42
6546      026466 006301      5$:   ASL     R1      ;INDEX=0---ERROR=41
6547      026470 010166 000006  MOV      R1,6(SP)      ;RETURN INDEX TO USER
6548      026474 012601      MOV      (SP)+,R1      ;RESTORE R1
6549      026476 012600      MOV      (SP)+,R0      ;RESTORE R0
6550      026500 000207      RTS     PC      ;RETURN FROM CALL
6551
6552      ;THIS ROUTINE WILL PROVIDE A STALL IN MILLISECONDS FOR A SPECIFIC
6553      ;AMOUNT OF TIME IF BIT13 OF C.SWR = 0 OR A RANDOM AMOUNT OF TIME
6554      ;IF BIT 13 OF C.SWR = 1.
6555      ;STALL1 CONTAINS SPECIFIED TIME FOR TESTS 0 - 7, AND STALL2
6556      ;CONTAINS THE TIME FOR TESTS 16-21.
6557      ;CALL
  
```



```

6558      :      JSR      RO,@#STALL
6559      :      TIME POINTER      ;WHERE TO FIND THE STALL TIME
6560
6561 026502 013046      STALL:  MOV      @(RO)+,-(SP)      ;PICKUP STALL TIME
6562 026504 032737 020000 001220  BIT      #SW13,@#C.SWR      ;USE A RANDOM TIME?
6563 026512 001406      BEQ      1$      ;NO--BRANCH
6564 026514 004737 023626  JSR      PC,@#$RAND      ;YES--FORM RANDOM NUMBER
6565 026520 013716 023726  MOV      @#$LONUM,(SP)      ;AND USE IT FOR THE STALL TIME
6566 026524 042716 177700  BIC      #^C77,(SP)      ;BUT NEVER > 64 MILLISECONDS
6567 026530 005046      1$:  CLR      -(SP)      ;CLEAR TEMP. LOCATION
6568 026532 162766 000001 000002  2$:  SUB      #1,2(SP)      ;MORE STALL REQUIRED?
6569 026540 103407      BLO      4$      ;NO--BRANCH
6570 026542 012716 000144  MOV      #100.,(SP)      ;STALL FOR ABOUT 1 MILLISECOND
6571 026546 005700      3$:  TST      RO      ;NOP TO KILL TIME
6572 026550 005366 000000  DEC      0(SP)      ;COUNT
6573 026554 001374      BNE      3$      ;LOOP IF MORE COUNTS NEEDED
6574 026556 000765      BR      2$
6575 026560 022626      4$:  CMP      (SP)+,(SP)+      ;CLEAN OFF THE STACK
6576 026562 000200      RTS      RO      ;EXIT
6577
6578
6579      ;ROUTINE TO PROVIDE A 2 MS STALL AFTER A SEEK OPERATION IN THE SEEK TIMING
6580      ;TESTS. THIS STALL IS REQUIRED TO COMPENSATE FOR THE 'ACCESS READY' DELAY
6581      ;IN THE RP04. THIS STALL TIME IS NOT INCLUDED IN THE CALCULATED SEEK TIMES.
6582      ;CALL
6583      :      JSR      PC,@#TWOMS
6584      :      RETURN
6585
6586 026564 013746 177776      TWOMS:  MOV      @#PS,-(SP)      ;SAVE THE PRESENT PROCESSOR STATUS
6587 026570 012737 000240 177776  MOV      #<5*32.>,@#PS      ;SET THE PROCESSOR PRIORITY TO 5
6588 026576 017746 152572  MOV      @PKV,-(SP)      ;SAVE THE OLD CLOCK VECTOR ADDRESS
6589 026602 012777 026626 152564  MOV      #1$,@PKV      ;SETUP NEW VECTOR ADDRESS
6590 026610 012777 000310 152564  MOV      #200.,@PKB      ;LOAD THE CLOCK BUFFER
6591 026616 012777 000101 152554  MOV      #101,@PKCS      ;START THE CLOCK
6592 026624 000001      WAIT      ;WAIT FOR 2 MS
6593 026626 062706 000004  1$:  ADD      #4,SP      ;INCREMENT STACK FOR RETURN
6594 026632 012677 152536  MOV      (SP)+,@PKV      ;RESTORE OLD CLOCK VECTOR
6595 026636 012637 177776  MOV      (SP)+,@#PS      ;RESTORE THE OLD PROCESSOR STATUS
6596 026642 000207      RTS      PC      ;RETURN
6597
6598      ;ROUTINE TO SOFTWARE COMPARE HEADER ON IMPLIED SEEKS
6599      ;CALL
6600      :      JSR      RO,@#VERIFY
6601      :      ADR POINTER      ;ADDRESS OF DPB+10 (SECTOR NUMBER)
6602      :      RETURN
6603
6604 026644 010146      VERIFY:  MOV      R1,-(SP)      ;SAVE R1
6605 026646 012001      MOV      (RO)+,R1      ;GET ADDRESS OF DPB+10
6606 026650 042737 010000 050202  BIC      #FMT22,@#BUFFER      ;STRIP FORMAT BIT FROM CYLINDER NUMBER
6607 026656 023761 050202 000002  CMP      @#BUFFER,2(R1)      ;CYLINDER NUMBER OK?
6608 026664 001003      BNE      1$      ;NO--BRANCH
6609 026666 023711 050204  CMP      @#BUFFER+2,(R1)      ;YES--HOW ABOUT TRACK/SECTOR?
6610 026672 001441      BEQ      3$      ;BRANCH IF GOOD
6611 026674 013737 050202 001262  1$:  MOV      @#BUFFER,@#CYL.RD      ;SAVE THE EXPECTED AND THE
6612 026702 113737 050205 001264  MOV      @#BUFFER+3,@#TRK.RD      ;RECEIVED CYLINDER, TRACK,
6613 026710 113737 050204 001266  MOV      @#BUFFER+2,@#SEC.RD      ;AND SECTOR

```

```

6614 026716 112137 001272      MOV      (R1)+,@#SEC.DS
6615 026722 112137 001274      MOV      (R1)+,@#TRK.DS
6616 026726 011137 001270      MOV      (R1),@#CYL.DS
6617 026732 012737 026744 001206  MOV      #2$, $ESCAPE      ;; ESCAPE TO 2$ ON ERROR
6618 026740 005740          TST      -(R0)              ;MAKE IT TEST PC+4
6619 026742 104012          ERROR    12                ;REPORT THE ERROR
6620 026744 012737 000107 004106 2$:  MOV      #RECAL,DPB.A+2    ;LOAD RECALIBRATE ORDER CODE
6621 026752 004037 025150          JSR      RO,@#CALL.A      ;GO EXECUTE THE COMMAND
6622 026756 005037 001206          CLR      $ESCAPE          ;CLEAR ERROR ESCAPE FLAG
6623 026762 032777 001000 152150  BIT      #SW9,@SWR         ;LOOP ON ERROR ?
6624 026770 001404          BEQ      4$               ;BR IF NOT
6625 026772 000177 152112          JMP      @SLPERR          ;RETURN TO ERROR LOOP ADDRESS
6626 026776 062700 000002 3$:  ADD      #2,R0             ;INCREMENT RETURN ADDRESS
6627 027002 012601 4$:  MOV      (SP)+,R1         ;RESTORE R1
6628 027004 000200          RTS      RO               ;EXIT
6629
6630          ; THIS ROUTINE WILL PERFORM A 'MASSBUS' INIT. FOLLOWED BY
6631          ; A 'RECALIBRATE' ON THE DRIVE UNDER TEST.
6632          ; NOTE: THIS ROUTINE DESTROYS R1 AND R4
6633          ; CALL
6634          ; JSR      RO,SRCH00      ; DO A MASSBUS INIT. AND RECAL
6635          ; RETURN1              ; RETURN HERE IF NO ERROR
6636          ; RETURN2              ; RETURN HERE ON ERROR
6637
6638 027006 005001          SRCH00: CLR      R1          ; INCASE OF ERROR (TYPTIM)
6639 027010 005037 177776          CLR      @#PS
6640 027014 012777 037544 005612  MOV      #ISR,@RPVEC      ; SETUP INTERRUPT VECTOR
6641 027022 013704 034632          MOV      @#RPADR,R4      ; PICKUP ADDRESS OF RPCS1
6642 027026 012764 000040 000010  MOV      #BIT05,RPCS2(R4) ; MASSBUS INIT.
6643 027034 005037 004174          CLR      @#DTADPB+10     ; TRACK=0; SECTOR=0
6644 027040 005037 004176          CLR      @#DTADPB+12     ; CYLINDER =0
6645 027044 012737 000107 004166  MOV      #RECAL,@#DTADPB+2 ; COMMAND = RECALIBRATE
6646 027052 005037 001206          CLR      @#$ESCAPE       ; NO ESCAPE ADDRESS
6647 027056 004037 035450          JSR      RO,@#RPO4       ; CALL THE DRIVER
6648 027062 004164          DTADPB
6649 027064 000440          BR      4$               ; QUEUE IS FULL
6650 027066 005737 004202 1$:  TST      DTADPB+16        ; WAIT ON DONE
6651 027072 001775          BEQ      1$
6652 027074 100030          BPL      3$               ; TAKE NORMAL EXIT IF NO ERROR
6653 027076 012737 027152 001206  MOV      #2$, $ESCAPE     ;; ESCAPE TO 2$ ON ERROR
6654 027104 013737 004176 001270  MOV      @#DTADPB+12,@#CYL.DS ; CYLINDER
6655 027112 113737 004175 001274  MOV      @#DTADPB+11,@#TRK.DS ; TRACK
6656 027120 113737 004174 001272  MOV      @#DTADPB+10,@#SEC.DS ; SECTOR
6657 027126 012746 004202          MOV      #DTADPB+16,-(SP) ; STATUS/ERROR INDICATOR ADDRESS
6658 027132 004737 026364          JSR      PC,@#ERINDX     ; FORM DISPATCH INDEX
6659 027136 062607          ADD      (SP)+,PC        ; REPORT PROPER ERROR
6660 027140 104041          ERROR    41
6661 027142 104042          ERROR    42              ; PARITY ERROR
6662 027144 104043          ERROR    43              ; UNSAFE ERROR
6663 027146 104044          ERROR    44              ; NON-I/O ERROR
6664 027150 104045          ERROR    45              ; I/O ERROR
6665 027152 005720 2$:  TST      (R0)+           ; ADJUST FOR ERROR EXIT
6666 027154 000404          BR      4$               ; GO TO THE EXIT
6667 027156 005064 000006 3$:  CLR      RPDA(R4)        ; TRACK AND SECTOR = 0
6668 027162 005064 000034          CLR      RPCA(R4)        ; CYLINDER = 0
6669 027166 000200 4$:  RTS      RO               ; RETURN
    
```

```

6670
6671      ;THIS IS AN RTI WHICH IS USED BY THE TIMING TESTS & THE SERVO SETTLE DOWN TEST
6672
6673 027170 000002 DORTI: RTI      ;RETURN FROM INTERRUPT
6674
6675      ;THIS ROUTINE WILL INITIALIZE THE TIMERS USED BY THE "TIMING ROUTINES
6676      ;CALL
6677      ;
6678      JSR      PC,@#STRTMR
6679      RETURN
6680 027172 104412 STRTMR: SAVREG      ;SAVE R0-R5
6681 027174 012700 001276 MOV      #TIM.UP,R0      ;START AT TIM.UP (MINIMUM)
6682 027200 012701 001332 MOV      #TIM.PT,R1      ;STOP AT TIM.PT
6683 027204 005020 1$: CLR      (R0)+      ;CLEAR
6684 027206 020001 CMP      R0,R1      ;DONE?
6685 027210 103775 BLO      1$      ;NO--BRANCH
6686 027212 012710 050202 MOV      #BUFFER,(R0)      ;SETUP POINTER
6687 027216 012737 077777 001276 MOV      #^CBIT15,@#TIM.UP ;SET MINIMUM TIME TO MAXIMUM
6688 027224 012737 077777 001314 MOV      #^CBIT15,@#TIM.DN ;POSITIVE NUMBER
6689 027232 104413 RESREG      ;RESTORE R0-R5
6690 027234 000207 RTS      PC      ;RETURN
6691
6692      ;THIS ROUTINE WILL ADD THE ELAPSED TIME TO THE AVERAGE COUNTER AND
6693      ;MAINTAIN THE MINIMUM AND MAXIMUM TIMES.
6694      ;NOTE: THIS ROUTINE DESTROYS R2
6695      ;CALL
6696      ;
6697      MOV      #TP,R3      ;PARAMETER POINTER
6698      MOV      FLAG,R5      ;FLAG=0=COUNT UP
6699      ;
6700      ;
6701      ;
6702      ;
6703      ;
6704      ;
6705      ;
6706      ;
6707      ;
6708      ;
6709      ;
6710      ;
6711      ;
6712      ;
6713      ;
6714      ;
6715      ;
6716      ;
6717      ;
6718      ;
6719      ;
6720      ;
6721      ;
6722      ;
6723      ;
6724      ;
6725      ;

```

```

6726 027360 062737 000002 001332      ADD    #2,@#TIM.PT      ;ADVANCE THE POINTER
6727 027366 000207                6$:   RTS      PC      ;RETURN
6728
6729                                     ;THIS ROUTINE IS USED TO TYPE THE MINIMUM,
6730                                     ;MAXIMUM, AND AVERAGE TIMES FOR THE TIMING TESTS
6731                                     ;IT WILL ALSO CHECK THE TIMES TO ENSURE
6732                                     ;THEY ARE WITHIN TOLERANCE AND IF NOT FLAG THE BAD TIMES.
6733                                     ;NOTE: THIS ROUTINE DESTROYS R2-R5
6734                                     ;CALL
6735                                     JSR    RO,@#TYPTIM      ;GO REPORT THE TIMES
6736                                     TABLE      ;POINT TO THE PROPER TABLE
6737                                     RETURN
6738
6739                                     ;TABLE:MSGADR1      ;ADDRESS OF ASCIZ MESSAGE NUMBER 1
6740                                     MSGADR2      ;ADDRESS OF ASCIZ MESSAGE NUMBER 2
6741                                     MIN.ALLOWED    ;MINIMUM TIME ALLOWED
6742                                     MAX.ALLOWED    ;MAXIMUM TIME ALLOWED
6743
6744 027370 012002                TYPTIM: MOV    (R0)+,R2      ;PICKUP THE TABLE POINTER
6745 027372 032777 000100 151540      BIT    #SW06,@SWR      ;INHIBIT TIME REPORTS?
6746 027400 001145                BNE    7$              ;YES--BRANCH
6747 027402 012237 027422                MOV    (R2)+,2$        ;ADDRESS OF MESSAGE NUMBER 1
6748 027406 012205                MOV    (R2)+,R5        ;ADDRESS OF MESSAGE NUMBER 2
6749 027410 012203                MOV    (R2)+,R3        ;PICKUP THE LOW LIMIT
6750 027412 011202                MOV    (R2),R2         ;AND THE HIGH LIMIT
6751 027414 012704 001276                MOV    #TIM.UP,R4      ;PARAMETER POINTER
6752 027420 104401                1$:   TYPE      ;TYPE THE MESSAGE
6753 027422 000000                2$:   .WORD    0        ;ASCIZ MESSAGE POINTER GOFS HERE
6754 027424 005764 000014                TST    14(R4)         ;DID ANY COUNTS OCCUR?
6755 027430 001527                BEQ    6$              ;NO--BRANCH
6756 027432 104401 044170                TYPE    ,MSGMIN      ;"MIN="
6757 027436 012446                MOV    (R4)+,-(SP)     ;PUT (R4)+ ON THE STACK
6758 027440 004737 023336                JSR    PC,@#$$SB2D    ;CHANGE (R4)+ TO DECIMAL ASCIZ
6759 027444 004737 023566                JSR    PC,@#$$SUPRS   ;TYPE WITHOUT LEADING ZEROS
6760 027450 104401 044215                TYPE    ,MSGOUS      ;"O US"
6761 027454 005724                TST    (R4)+          ;ANY SEEKS BELOW THE LOW LIMIT
6762 027456 001421                BEQ    3$              ;NO--BRANCH
6763 027460 104401 044330                TYPE    ,MSG.SP      ;
6764 027464 016446 177776                MOV    -2(R4),-(SP)   ;PUT -2(R4) ON THE STACK
6765 027470 004737 023336                JSR    PC,@#$$SB2D    ;CHANGE -2(R4) TO DECIMAL ASCIZ
6766 027474 004737 023566                JSR    PC,@#$$SUPRS   ;TYPE WITHOUT LEADING ZEROS
6767 027500 104401 044222                TYPE    ,M BELOW     ;"BELOW THE MINIMUM OF"
6768 027504 010346                MOV    R3,-(SP)      ;PUT R3 ON THE STACK
6769 027506 004737 023336                JSR    PC,@#$$SB2D    ;CHANGE R3 TO DECIMAL ASCIZ
6770 027512 004737 023566                JSR    PC,@#$$SUPRS   ;TYPE WITHOUT LEADING ZEROS
6771 027516 104401 044215                TYPE    ,MSGOUS      ;
6772 027522 104401 044177                3$:   TYPE    ,MSGMAX  ;"MAX="
6773 027526 012446                MOV    (R4)+,-(SP)   ;PUT (R4)+ ON THE STACK
6774 027530 004737 023336                JSR    PC,@#$$SB2D    ;CHANGE (R4)+ TO DECIMAL ASCIZ
6775 027534 004737 023566                JSR    PC,@#$$SUPRS   ;TYPE WITHOUT LEADING ZEROS
6776 027540 104401 044215                TYPE    ,MSGOUS      ;
6777 027544 005724                TST    (R4)+          ;ANY SEEKS ABOVE THE HIGH LIMIT
6778 027546 001421                BEQ    4$              ;NO--BRANCH
6779 027550 104401 044330                TYPE    ,MSG.SP      ;YES--REPORT HOW MANY
6780 027554 016446 177776                MOV    -2(R4),-(SP)   ;PUT -2(R4) ON THE STACK
6781 027560 004737 023336                JSR    PC,@#$$SB2D    ;CHANGE -2(R4) TO DECIMAL ASCIZ

```

```

6782 027564 004737 023566      JSR    PC,@#%SUPRS      ;TYPE WITHOUT LEADING ZEROS
6783 027570 104401 044251      TYPE   ,MABOVE         ;"ABOVE THE MAXIMUM OF"
6784 027574 010246                MOV    R2,-(SP)         ;PUT R2 ON THE STACK
6785 027576 004737 023336      JSR    PC,@#%SB2D      ;CHANGE R2 TO DECIMAL ASCII
6786 027602 004737 023566      JSR    PC,@#%SUPRS      ;TYPE WITHOUT LEADING ZEROS
6787 027606 104401 044215      TYPE   ,MSGOUS
6788 027612 104401 044206      4$:   TYPE   ,MSGAVG     ;"AVG="
6789 027616 012446                MOV    (R4)+,-(SP)     ;FORM THE AVERAGE
6790 027620 012446                MOV    (R4)+,-(SP)
6791 027622 012446                MOV    (R4)+,-(SP)
6792 027624 004737 023730      JSR    PC,@#%DIV
6793 027630 006126                ROL    (SP)+           ;IS THE REMAINDER OVER HALF?
6794 027632 100001                BPL    5$              ;NO--BRANCH
6795 027634 005216                INC    (SP)            ;YES--ROUND UP
6796 027636                5$:
6797 027636 004737 023336      JSR    PC,@#%SB2D      ;CHANGE TO DECIMAL ASCII
6798 027642 004737 023566      JSR    PC,@#%SUPRS      ;TYPE WITHOUT LEADING ZEROS
6799 027646 104401 044215      TYPE   ,MSGOUS
6800 027652 104401 044330      TYPE   ,MSG.SP
6801 027656 016446 177776      MOV    -2(R4),-(SP)    ;PUT -2(R4) ON THE STACK
6802 027662 004737 023336      JSR    PC,@#%SB2D      ;CHANGE -2(R4) TO DECIMAL ASCII
6803 027666 004737 023566      JSR    PC,@#%SUPRS      ;TYPE WITHOUT LEADING ZEROS
6804 027672 104401 044300      TYPE   ,MSGNUM         ;"SEEKS TIMED"
6805 027676 010537 027422      MOV    R5,2$          ;NEXT MESSAGE POINTER
6806 027702 001404                BEQ    7$              ;IF NONE EXIT
6807 027704 005005                CLR    R5              ;NO MORE THAN 2
6808 027706 000644                BR     1$
6809 027710 104401 044315      6$:   TYPE   ,MSGNON
6810 027714 000200      7$:   RTS     R0        ;EXIT

;THIS SUBROUTINE WILL INCREMENT THE TRACK
;NUMBER (R2) BY THE AMOUNT SPECIFIED BY 'IT'.
;CALL
;
;      JSR    R0,@#INCTRK
;      RETURN1
;      RETURN2
;
;      ;TRACK NUMBER GREATER THAN LT15
;      ;TRACK NUMBER INCREMENTED
INCTRK: CMP    R2,@#LT      ;LAST TRACK COMPLETED?
        BEQ    2$          ;YES--EXIT
        ADD    @#IT,R2     ;NO--UPDATE TRACK
        CMP    R2,@#LT     ;TRACK TO BIG?
        BLE    1$          ;NO--EXIT
        MOV    @#LT,R2     ;YES--SET TRACK TO LAST TRACK
1$:     TST    (R0)+       ;ADJUST FOR RETURN 2
2$:     RTS     R0        ;RETURN

;THIS SUBROUTINE WILL INCREMENT THE CYLINDER
;NUMBER (R1) BY THE AMOUNT SPECIFIED BY 'IC'.
;CALL
;
;      JSR    R0,@#INCCYL
;      RETURN1
;      RETURN2
;
;      ;CYLINDER NUMBER GREATER THAN LC15
;      ;CYLINDER NUMBER INCREMENTED
INCCYL: CMP    R1,@#LC
        BEQ    2$          ;YES--EXIT

```

```

6838 027754 063701 001514      ADD    @#IC,R1      ;NO--UPDATE CYLINDER
6839 027760 020137 001512      CMP    R1,@#LC     ;CYLINDER TO BIG?
6840 027764 003402              BLE    1$          ;NO--EXIT
6841 027766 013701 001512      MOV    @#LC,R1     ;YES--SET CYLINDER TO LAST CYLINDER
6842 027772 005720              1$:   TST    (R0)+   ;ADJUST FOR RETURN 2
6843 027774 000200              2$:   RTS     RO    ;RETURN
6844
6845      ;THIS ROUTINE DECREMENTS THE SECTOR ADDRESS.
6846      ;CALL
6847      ;
6848      ;   CLR    -(SP)      ;CLEAR THE STACK
6849      ;   JSR    PC,DECSEC ;SUBROUTINE ENTRY
6850      ;   RETURN
6851 027776 113766 004212 000002  DECSEC: MOV    RP,REG+RPDA,2(SP) ;PUT THE SECTOR ADDRESS ON THE STACK
6852 030004 005366 000002          DEC    2(SP)      ;DECREMENT THE ADDRESS
6853 030010 100003              BPL    1$          ;BR IF NOT CORRECTION NEEDED
6854 030012 013766 001634 000002  MOV    PRMLMT+22.,2(SP) ;OVERFLOW OCCURED, FORCE TO MAXIMUM ADDRESS
6855 030020 000207              1$:   RTS     PC    ;RETURN
6856
6857      ;THIS SUBROUTINE IS USED TO FILL THE DATA BUFFER
6858      ;WITH ADDRESSES FROM 0 TO 21 WITH EACH ADDRESS
6859      ;BEING STORED IN 256 CONSECUTIVE LOCATIONS
6860      ;CALL
6861      ;   JSR    PC,@#FILBUF
6862      ;   RETURN
6863
6864 030022 104412          FILBUF: SAVREG      ;SAVE RO - R5
6865 030024 005000          CLR    RO          ;FIRST DISK ADDRESS
6866 030026 012701 050202  MOV    #BUFFER,R1   ;START FILLING HERE
6867 030032 012702 000400  1$:   MOV    #256.,R2 ;DO 256 WORDS
6868 030036 010021          2$:   MOV    RO,(R1)+   ;STORE
6869 030040 005302          DEC    R2          ;MORE?
6870 030042 003375          BGT    2$          ;YES--BRANCH
6871 030044 005200          INC    RO          ;NO--UPDATE DISK ADDRESS
6872 030046 023700 001630  CMP    PRMLMT+22,RO ;DONE?
6873 030052 103367          BHS    1$          ;NO--BRANCH
6874 030054 104413          RESREG      ;RESTORE RO - R5
6875 030056 000207          RTS     PC        ;RETURN
6876
6877      ;THIS ROUTINE WILL CLEAR THE BUFFER BY
6878      ;SETTING EACH WORD TO "177400".
6879      ;CALL
6880      ;   JSR    RO,@#CLRBUF
6881      ;   RETURN
6882
6883 030060 104412          CLRBUF: SAVREG     ;SAVE RO - R5
6884 030062 012701 177400  MOV    #177400,R1   ;WORD TO FILL BUFFER WITH
6885 030066 012702 050202  MOV    #BUFFER,R2   ;FIRST ADDRESS OF BUFFER
6886 030072 012703 076202  MOV    #BUFFER+<512.*22.>,R3 ;LAST ADDRESS+2 OF BUFFER
6887 030076 010122          1$:   MOV    R1,(R2)+   ;FILL WORDS 1, 9,...249,...5625
6888 030100 010122          MOV    R1,(R2)+   ;FILL WORDS 2,10,...250,...5626
6889 030102 010122          MOV    R1,(R2)+   ;FILL WORDS 3,11,...251,...5627
6890 030104 010122          MOV    R1,(R2)+   ;FILL WORDS 4,12,...252,...5628
6891 030106 010122          MOV    R1,(R2)+   ;FILL WORDS 5,13,...253,...5629
6892 030110 010122          MOV    R1,(R2)+   ;FILL WORDS 6,14,...254,...5630
6893 030112 010122          MOV    R1,(R2)+   ;FILL WORDS 7,15,...255,...5631

```

```

6894 030114 010122      MOV      R1,(R2)+      :FILL WORDS 8,16,...256,...5632
6895 030116 020203      CMP      R2,R3        :DONE?
6896 030120 103766      BLO     1$            :NO--BRANCH
6897 030122 104413      RESREG                    :RESTORE R0 - R5
6898 030124 000200      RTS      R0           :RETURN FROM CALL
6899
6900                      :THIS ROUTINE IS USED TO CHECK THE DATA BUFFER
6901                      :FOR ADDRESSES 0 THROUGH 21 WITH EACH ADDRESS
6902                      :BEING STORED IN 256 CONSECUTIVE LOCATIONS
6903                      :CALL
6904                      :
6905                      :      JSR      R0,@#CKSCTR
6906                      :      RETURN
6907 030126 104412      CKSCTR: SAVREG                    :SAVE R0 - R5
6908 030130 162706 000004  SUB      #4,SP          :RESERVE TEMP. STORAGE AREA
6909 030134 005001      CLR      R1           :FIRST SECTOR
6910 030136 012716 050202  MOV      #BUFFER,(SP)   :FIRST ADDRESS OF DATA BUFFER
6911 030142 005066 000002  CLR      2(SP)         :NO ERRORS
6912 030146 012702 000020  1$: MOV      #16.,R2 ;LOOP COUNT (16*16=256)
6913 030152 011603      MOV      (SP),R3      :GET 1ST ADDRESS OF THIS SECTORS DATA
6914 030154
6915 030154 020123      2$: CMP      R1,(R3)+   :WORD 1
6916 030156 001063      BNE     7$            :BRANCH IF BAD
6917 030160 020123      CMP      R1,(R3)+   :WORD 2
6918 030162 001061      BNE     7$            :BRANCH IF BAD
6919 030164 020123      CMP      R1,(R3)+   :WORD 3
6920 030166 001057      BNE     7$            :BRANCH IF BAD
6921 030170 020123      CMP      R1,(R3)+   :WORD 4
6922 030172 001055      BNE     7$            :BRANCH IF BAD
6923 030174 020123      CMP      R1,(R3)+   :WORD 5
6924 030176 001053      BNE     7$            :BRANCH IF BAD
6925 030200 020123      CMP      R1,(R3)+   :WORD 6
6926 030202 001051      BNE     7$            :BRANCH IF BAD
6927 030204 020123      CMP      R1,(R3)+   :WORD 7
6928 030206 001047      BNE     7$            :BRANCH IF BAD
6929 030210 020123      CMP      R1,(R3)+   :WORD 8
6930 030212 001045      BNE     7$            :BRANCH IF BAD
6931 030214 020123      CMP      R1,(R3)+   :WORD 9
6932 030216 001043      BNE     7$            :BRANCH IF BAD
6933 030220 020123      CMP      R1,(R3)+   :WORD 10
6934 030222 001041      BNE     7$            :BRANCH IF BAD
6935 030224 020123      CMP      R1,(R3)+   :WORD 11
6936 030226 001037      BNE     7$            :BRANCH IF BAD
6937 030230 020123      CMP      R1,(R3)+   :WORD 12
6938 030232 001035      BNE     7$            :BRANCH IF BAD
6939 030234 020123      CMP      R1,(R3)+   :WORD 13
6940 030236 001033      BNE     7$            :BRANCH IF BAD
6941 030240 020123      CMP      R1,(R3)+   :WORD 14
6942 030242 001031      BNE     7$            :BRANCH IF BAD
6943 030244 020123      CMP      R1,(R3)+   :WORD 15
6944 030246 001027      BNE     7$            :BRANCH IF BAD
6945 030250 020123      CMP      R1,(R3)+   :WORD 16
6946 030252 001025      BNE     7$            :BRANCH IF BAD
6947 030254 005302      DEC     R2           :FINISHED WITH THIS SECTORS DATA?
6948 030256 001336      BNE     2$            :NO--BRANCH
6949 030260 062716 001000  3$: ADD      #512.,(SP)   :YES--FIRST ADDRESS OF NEXT SECTOR

```

```

6950 030264 005201          INC      R1          ;MOVE TO NEXT SECTOR
6951 030266 023701 001630  CMP      PRMLMT+22,R1 ;DONE?
6952 030272 103325          BHS      1$          ;NO--BRANCH
6953 030274 005766 000002 4$:      TST      2(SP)    ;ERROR OCCUR?
6954 030300 001406          BEQ      6$          ;NO--BRANCH
6955 030302 123737 001364 001103  CMPB    @#ERR.CT,@#SERFLG ;MAX. ERROR OCCURRED?
6956 030310 101002          BHI      6$          ;NO--BRANCH
6957 030312 013700 001252 5$:      MOV      @#BYPASS,R0   ;TAKE ERROR EXIT
6958 030316 062706 000004 6$:      ADD      #4,SP        ;FREE TEMP. AREA
6959 030322 104413          RESREG                    ;RESTORE R0 - R5
6960 030324 000200          RTS                     ;RETURN FROM CALL
6961 030326 010304          7$:      MOV      R3,R4        ;FORM WORD NUMBER AND
6962 030330 161604          SUB      (SP),R4       ;ADDRESS TO CONTINUE FROM
6963 030332 010405          MOV      R4,R5
6964 030334 006204          ASR      R4            ;WORD NUMBER
6965 030336 042705 177740          BIC      #^C37,R5
6966 030342 001002          BNE      8$          ;BRANCH IF NOT A MULTIPLE OF 16
6967 030344 012705 000040          MOV      #40,R5       ;SET TO WORD 16
6968 030350 006305          8$:      ASL      R5
6969 030352 062705 030154          ADD      #2$,R5       ;ADDRESS
6970 030356 016337 177776 001126  MOV      -2(R3),@#SBDDAT ;SAVE RAD DATA
6971 030364 005766 000002          TST      2(SP)        ;FIRST ERROR?
6972 030370 001015          BNE      10$         ;NO--BRANCH
6973 030372 013737 004176 001270  MOV      @#DTADPB+12,@#CYL.DS ;CYLINDER NUMBER
6974 030400 113737 004175 001274  MOVB     @#DTADPB+11,@#TRK.DS ;TRACK NUMBER
6975 030406 012737 030416 001206  MOV      #9$, $ESCAPE  ;:ESCAPE TO 9$ ON ERROR
6976 030414 104021          ERROR   21            ;REPORT THE ERROR
6977 030416 105166 000002 9$:      COMB    2(SP)        ;SET ERROR SWITCH
6978 030422 000404          BR      11$
6979 030424          10$:
6980 030424 012737 030434 001206  MOV      #11$, $ESCAPE ;:ESCAPE TO 11$ ON ERROR
6981 030432 104022          ERROR   22            ;REPORT THE ERROR
6982 030434 032777 001000 150476 11$:    BIT      #SW09,@SWR    ;LOOP ON ERROR?
6983 030442 001323          BNE      5$          ;YES
6984 030444 032777 000002 150466  BIT      #SW01,@SWR    ;STOP DATA COMPARE?
6985 030452 001310          BNE      4$          ;YES--BRANCH
6986 030454 123737 001364 001103  CMPB    @#ERR.CT,@#SERFLC ;MAX. ERRORS?
6987 030462 101713          BLOS    5$          ;YES--BRANCH
6988 030464 032777 000040 150446  BIT      #SW05,@SWR    ;REPORT ONLY 1ST ERROR PER SECTOR?
6989 030472 001272          BNE      3$          ;YES--BRANCH
6990 030474 000115          JMP      (R5)

```

:THIS ROUTINE WILL MOVE THE 16 WORDS OF THE  
 :DESIRED PATTERN INTO THE DATA BUFFER.

```

:CALL
:      MOV      #NX,R0          ;PATTERN NUMBER INDEX TO R0
:      JSR      PC,@#SETBUF

```

```

SETBUF: SAVREG                    ;SAVE R0 - R5
        MOV      #BUFFER,R1     ;FIRST ADDRESS
        MOV      @#DTADPB+4,R2  ;WORD COUNT
        1$:      MOV      PAT.PT(R0),R3 ;PICKUP PATTERN POINTER
        MOV      (R3)+,(R1)+    ;MOVE WORD 1 INTO DATA BUFFER
        MOV      (R3)+,(R1)+    ;MOVE WORD 2 INTO DATA BUFFER
        MOV      (R3)+,(R1)+    ;MOVE WORD 3 INTO DATA BUFFER
        MOV      (R3)+,(R1)+    ;MOVE WORD 4 INTO DATA BUFFER

```

```

6998 030476 104412          050202
6999 030500 012701 004170
7000 030504 013702 003044
7001 030510 016003
7002 030514 012321
7003 030516 012321
7004 030520 012321
7005 030522 012321

```



7006	030524	012321	MOV	(R3)+,(R1)+	:MOVE WORD 5 INTO DATA BUFFER
7007	030526	012321	MOV	(R3)+,(R1)+	:MOVE WORD 6 INTO DATA BUFFER
7008	030530	012321	MOV	(R3)+,(R1)+	:MOVE WORD 7 INTO DATA BUFFER
7009	030532	012321	MOV	(R3)+,(R1)+	:MOVE WORD 8 INTO DATA BUFFER
7010	030534	012321	MOV	(R3)+,(R1)+	:MOVE WORD 9 INTO DATA BUFFER
7011	030536	012321	MOV	(R3)+,(R1)+	:MOVE WORD 10 INTO DATA BUFFER
7012	030540	012321	MOV	(R3)+,(R1)+	:MOVE WORD 11 INTO DATA BUFFER
7013	030542	012321	MOV	(R3)+,(R1)+	:MOVE WORD 12 INTO DATA BUFFER
7014	030544	012321	MOV	(R3)+,(R1)+	:MOVE WORD 13 INTO DATA BUFFER
7015	030546	012321	MOV	(R3)+,(R1)+	:MOVE WORD 14 INTO DATA BUFFER
7016	030550	012321	MOV	(R3)+,(R1)+	:MOVE WORD 15 INTO DATA BUFFER
7017	030552	012321	MOV	(R3)+,(R1)+	:MOVE WORD 16 INTO DATA BUFFER

```

7018 030554 062702 000020      ADD    #16.,R2 ;DONE?
7019 030560 001353              BNE    1$      :NO--BRANCH
7020 030562 104413              RESREG          :RESTORE R0 - R5
7021 030564 000207              RTS     PC      :RETURN
7022
7023                          ;THIS ROUTINE COMPARES A 16 WORD DATA PATTERN
7024                          ;AGAINST THE DATA BUFFER
7025                          ;CALL
7026                          ;
7027                          ;      MOV    #NX,R0      :PATTERN NUMBER INDFX TO R0
7028                          ;      JSR    PC,@#DATCMP
7029                          ;      RETURN
7030 030566 104412      DATCMP: SAVREG          :SAVE R0 - R5
7031 030570 012701 050202      MOV    #BUFFER,R1  :FIRST ADDRESS OF BUFFER
7032 030574 013702 004170      MOV    @#DTADPB+4,R2 :WORD COUNT
7033 030600 005046              CLR    -(SP)       :NO ERROR
7034 030602 016003 003044      1$:  MOV    PAT.PT(R0),R3 :PATTERN POINTER
7035 030606 2$:
7036 030606 162321              SUB    (R3)+,(R1)+ :CHECK WORD 1
7037 030610 001044              BNE    4$          :BRANCH IF DIFFERENT
7038 030612 162321              SUB    (R3)+,(R1)+ :CHECK WORD 2
7039 030614 001042              BNE    4$          :BRANCH IF DIFFERENT
7040 030616 162321              SUB    (R3)+,(R1)+ :CHECK WORD 3
7041 030620 001040              BNE    4$          :BRANCH IF DIFFERENT
7042 030622 162321              SUB    (R3)+,(R1)+ :CHECK WORD 4
7043 030624 001036              BNE    4$          :BRANCH IF DIFFERENT
7044 030626 162321              SUB    (R3)+,(R1)+ :CHECK WORD 5
7045 030630 001034              BNE    4$          :BRANCH IF DIFFERENT
7046 030632 162321              SUB    (R3)+,(R1)+ :CHECK WORD 6
7047 030634 001032              BNE    4$          :BRANCH IF DIFFERENT
7048 030636 162321              SUB    (R3)+,(R1)+ :CHECK WORD 7
7049 030640 001030              BNE    4$          :BRANCH IF DIFFERENT
7050 030642 162321              SUB    (R3)+,(R1)+ :CHECK WORD 8
7051 030644 001026              BNE    4$          :BRANCH IF DIFFERENT
7052 030646 162321              SUB    (R3)+,(R1)+ :CHECK WORD 9
7053 030650 001024              BNE    4$          :BRANCH IF DIFFERENT
7054 030652 162321              SUB    (R3)+,(R1)+ :CHECK WORD 10
7055 030654 001022              BNE    4$          :BRANCH IF DIFFERENT
7056 030656 162321              SUB    (R3)+,(R1)+ :CHECK WORD 11
7057 030660 001020              BNE    4$          :BRANCH IF DIFFERENT
7058 030662 162321              SUB    (R3)+,(R1)+ :CHECK WORD 12
7059 030664 001016              BNE    4$          :BRANCH IF DIFFERENT
7060 030666 162321              SUB    (R3)+,(R1)+ :CHECK WORD 13
7061 030670 001014              BNE    4$          :BRANCH IF DIFFERENT
7062 030672 162321              SUB    (R3)+,(R1)+ :CHECK WORD 14
7063 030674 001012              BNE    4$          :BRANCH IF DIFFERENT
7064 030676 162321              SUB    (R3)+,(R1)+ :CHECK WORD 15
7065 030700 0C1010              BNE    4$          :BRANCH IF DIFFERENT
7066 030702 162321              SUB    (R3)+,(R1)+ :CHECK WORD 16
7067 030704 001006              BNE    4$          :BRANCH IF DIFFERENT
7068 030706 062702 000020      ADD    #16.,R2      :DONE ?
7069 030712 001333              BNE    1$          :NO--BRANCH
7070 030714 005726      3$:  TST    (SP)+      :YES -- CLEAN UP STACK
7071 030716 104413              RESREG          :RESTORE R0 - R5
7072 030720 000207              RTS     PC
7073 030722 010104      4$:  MOV    R1,R4      :FORM THE WORD NUMBER

```

```

7074 030724 162704 050202      SUB      #BUFFER,R4
7075 030730 006204              ASR      R4                ;WORD NUMBER
7076 030732 010305              MOV      R3,R5            ;FORM ADDRESS TO CONTINUE FROM
7077 030734 166005 003044      SUB      PAT.P1(R0),R5
7078 030740 006305              ASL      R5
7079 030742 062705 030606      ADD      #28,R5           ;ADDRESS
7080 030746 064341              ADD      -(R3),-(R1)      ;RECONSTRUCT THE BAD WORD
7081 030750 010137 001122      MOV      R1,@#SBDADR      ;SAVE THE ERROR INFORMATION
7082 030754 010337 001120      MOV      R3,@#SGDADR
7083 030760 012137 001126      MOV      (R1)+,@#SDDAT
7084 030764 012337 001124      MOV      (R3)+,@#SGDDAT
7085 030770 005716              TST      (SP)             ;1ST DATA COMPARE ERROR?
7086 030772 001023              BNE      6$               ;NO--BRANCH
7087 030774 013737 004176 001270  MOV      @#DTADPB+12,@#CYL.DS ;CYLINDER
7088 031002 113737 004175 001274  MOVVB   @#DTADPB+11,@#TRK.DS ;TRACK
7089 031010 113737 004174 001272  MOVVB   @#DTADPB+10,@#SEC.DS ;SECTOR
7090 031016 016600 000026      MOV      26(SP),R0        ;GET TEST PC+4
7091 031022 012737 031032 001206  MOV      #5$,SESCAPE      ;ESCAPE TO 5$ ON ERROR
7092 031030 104013              ERROR   13                ;REPORT THE ERROR
7093 031032 016600 000020      MOV      20(SP),R0        ;PATTERN NUMBER INDEX
7094 031036 105116              COMB    (SP)              ;SET THE ERROR SWITCH
7095 031040 000404              BR      7$
7096 031042                      6$:
7097 031042 012737 031052 001206  MOV      #7$,SESCAPE      ;ESCAPE TO 7$ ON ERROR
7098 031050 104014              ERROR   14                ;REPORT THE ERROR
7099 031052 032777 000002 150060 7$:  BIT      #SW01,@SWR        ;STOP DATA COMPARE?
7100 031060 001315              BNE     3$                ;YES--EXIT
7101 031062 123737 001364 001103  CMPB   @#ERR.CT,@#SERFLG ;MAX. ERRORS?
7102 031070 101004              BHI     8$                ;NO--BRANCH
7103 031072 013766 001252 000016  MOV      @#BYPASS,16(SP) ;YES--ERROR EXIT
7104 031100 000705              BR      3$
7105 031102 000115      8$:  JMP      (R5)             ;NO--CONTINUE AT NEXT WORD
7106
7107      ;THIS ROUTINE WILL FILL THE DATA BUFFER (256*22 WORDS) WITH
7108      ;A RANDOM PATTERN. THE FIRST TWO WORDS OF EVERY 256 WILL
7109      ;BE THE BASE OF THE RANDOM NUMBER GENERATOR FOR THE
7110      ;NEXT 254 WORDS.
7111      ;NOTE: THIS ROUTINE DESTROYS R1 AND R2
7112      ;CALL
7113      ;      JSR      R0,@#FILRAN
7114      ;      RETURN
7115
7116 031104 012701 050202      FILRAN: MOV      #BUFFER,R1
7117 031110 013702 001630      MOV      PRMLMT+22,R2     ;MAXIMUM NUMBER OF SECTORS
7118 031114 004037 031324      1$:  JSR      R0,@#RANPAT
7119 031120 005302              DEC      R2
7120 031122 100374              BPL     1$
7121 031124 000200              RTS     R0
7122
7123      ;THIS ROUTINE USES THE FIRST TWO WORDS OF THE
7124      ;READ BUFFER TO GENERATED A RANDOM PATTERN. THEN
7125      ;THE READ BUFFER IS COMPARED TO THE PATTERN GENERATED.
7126      ;NOTE: THIS ROUTINE DESTROYS R1-R4
7127      ;CALL
7128      ;      JSR      R0,@#RANCK
7129      ;      RETURN
  
```

```

7130
7131 031126 013746 023724      RANCK:  MOV    @#$HINUM,-(SP)  ;SAVE THE PRESENT RANDOM NUMBER
7132 031132 013746 023726      MOV    @#$LONUM,-(SP)
7133 031136 012702 051202      MOV    #BUFFER+512.,R2 ;READ BUFFER ADDRESS
7134 031142 012701 052202      MOV    #BUFFER+1024.,R1 ;RANDOM PATTERN ADDRESS
7135 031146 010103      MOV    R1,R3 ;COPY IT INTO R3 FOR LATER USE
7136 031150 011237 023726      MOV    (R2),@#$LONUM ;PRIME THE RANDOM NUMBER GENERATOR
7137 031154 016237 000002 023724  MOV    2(R2),@#$HINUM
7138 031162 004037 031324      JSR    R0,@#RANPAT ;GENERATE A RANDOM PATTERN
7139 031166 012637 023726      MOV    (SP)+,@#$LONUM ;RESTORE PRESENT RANDOM NUMBER
7140 031172 012637 023724      MOV    (SP)+,@#$HINUM
7141 031176 005046      CLR    -(SP) ;NO ERRORS
7142 031200 162322      1$:  SUB    (R3)+,(R2)+ ;ARE THESE TWO WORDS DIFFERENT?
7143 031202 001441      BEQ    4$ ;NO--BRANCH
7144 031204 012737 031256 001206  MOV    #3$, $ESCAPE ;ESCAPE TO 3$ ON ERROR
7145 031212 064342      ADD    -(R3),-(R2) ;RECREATE THE BAD WORD
7146 031214 010237 001122      MOV    R2,@#$BDADR ;ADDRESS OF BAD DATA
7147 031220 010337 001120      MOV    R3,@#$GDADR ;ADDRESS OF GOOD DATA
7148 031224 012237 001126      MOV    (R2)+,@#$BDDAT ;BAD DATA
7149 031230 012337 001124      MOV    (R3)+,@#$GDDAT ;GOOD DATA
7150 031234 010204      MOV    R2,R4 ;FORM WORD NUMBER (1 TO 256)
7151 031236 162704 051202      SUB    #BUFFER+512.,R4
7152 031242 006204      ASR    R4
7153 031244 005716      TST    (SP) ;FIRST ERROR
7154 031246 001002      BNE    2$ ;NO--BRANCH
7155 031250 105116      COMB   (SP) ;YES--SET ERROR SWITCH
7156 031252 104015      ERROR  15 ;REPORT THE ERROR
7157 031254 104016      2$:  ERROR  16 ;REPORT THE ERROR
7158 031256 032777 001000 147654  3$:  BIT    #SW09,@$SWR ;LOOP ON ERROR?
7159 031264 001012      BNE    5$ ;YES--BRANCH
7160 031266 123737 001364 001103  CMPB   @#ERR.CT,@#$ERFLG ;MAX. ERRORS OCCURRED?
7161 031274 101406      BLOS   5$ ;YES--BRANCH
7162 031276 032777 000002 147634  BIT    #SW01,@$SWR ;STOP COMPARING?
7163 031304 001002      BNE    5$ ;YES--BRANCH
7164 031306 020103      4$:  CMP    R1,R3 ;ALL DATA BEEN COMPARED?
7165 031310 101333      BHI    1$ ;NO--BRANCH
7166 031312 005726      5$:  TST    (SP)+ ;ERROR OCCUR?
7167 031314 001402      BEQ    6$ ;NO--BRANCH
7168 031316 013700 001252  MOV    @#BYPASS,R0 ;TAKE ERROR EXIT
7169 031322 000200      6$:  RTS    R0 ;EXIT
7170
7171      ;THIS ROUTINE FILLS A 256 WORD BUFFER WITH A RANDOM
7172      ;PATTERN OF WHICH THE FIRST TWO WORDS ARE THE BASE
7173      ;OF THE PATTERN.
7174      ;CALL
7175      ;   MOV    #ADR,R1 ;ADDRESS OF THE BUFFER
7176      ;   JSR    R0,@#RANPAT
7177      ;   RETURN
7178
7179 031324 010246      RANPAT: MOV    R2,-(SP) ;SAVE R2
7180 031326 012702 000200      MOV    #256./2.,R2 ;GENERATE 256 WORDS
7181 031332 000402      BR     2$
7182 031334 004737 023626      1$:  JSR    PC,@#$SRAND ;GENERATE A RANDOM NUMBER
7183 031340 013721 023726      2$:  MOV    @#$LONUM,(R1)+ ;PUT LOW WORD IN BUFFER
7184 031344 013721 023724      MOV    @#$HINUM,(R1)+ ;PUT HIGH WORD IN BUFFER
7185 031350 005302      DEC    R2 ;DONE?

```

```

7186 031352 003370          BGT      1$          :NO--BRANCH
7187 031354 012602          MOV      (SP)+,R2     :RESTORE R2
7188 031356 000200          RTS      R0          :EXIT
7189
7190          :THIS ROUTINE GENERATES RANDOM CYLINDER, TRACK, AND SECTOR
7191          :ADDRESSES AND SAVES THEM IN THE DPB (DTADPB+10 AND DTADPB+12).
7192          :NOTE: THIS ROUTINE DESTROYS R1-R3
7193          :CALL
7194          :      JSR      R0,@#RANADR
7195          :      RETURN
7196
7197 031360 004737 023626  RANADR: JSR      PC,@#SRAND   :GENERATE A RANDOM NUMBER
7198 031364 113701 023726  MOV     @#$LONUM,R1    :FORM SECTOR IN R1
7199 031370 113702 023727  MOV     @#$LONUM+1,R2  :FORM TRACK IN R2
7200 031374 013703 023724  MOV     @#$HINUM,R3   :FORM CYLINDER IN R3
7201 031400 105701          TSTB    R1           :ENSURE THE SECTOR IS BETWEEN 0 AND 21
7202 031402 002403          BLT     2$
7203 031404 123701 001630  1$:  CMP    PRMLMT+22,R1  :CHECK MAXIMUM SECTOR ADDRESS
7204 031410 103003          BHIS   3$
7205 031412 000241          CLC
7206 031414 106001          RORB   R1
7207 031416 000772          BR     1$
7208 031420 105702          3$:  TSTB    R2           :ENSURE THE TRACK IS BETWEEN 0 AND 18
7209 031422 002403          BLT     5$
7210 031424 122702 000023  4$:  CMP    #19.,R2
7211 031430 003003          BGT     6$
7212 031432 000241          CLC
7213 031434 106002          RORB   R2
7214 031436 000772          BR     4$
7215 031440 023703 001510  6$:  CMP    @#FC,R3       :ENSURE THE CYLINDER IS BETWEEN FC AND LC
7216 031444 003413          BLE    7$
7217 031446 000241          CLC
7218 031450 006003          ROR    R3
7219 031452 005503          ADC    R3
7220 031454 001371          BNE    6$
7221 031456 010103          MOV    R1,R3
7222 031460 000303          SWAB  R3
7223 031462 060203          ADD    R2,R3
7224 031464 005203          INC    R3
7225 031466 003364          BGT    6$
7226 031470 005403          NEG    R3
7227 031472 000762          BR     6$
7228 031474 023703 001512  7$:  CMP    @#LC,R3
7229 031500 002003          BGE    8$
7230 031502 000241          CLC
7231 031504 006003          ROR    R3
7232 031506 000772          BR     7$
7233 031510 023703 001510  8$:  CMP    @#FC,R3
7234 031514 003403          BLE    9$
7235 031516 005203          INC    R3
7236 031520 000303          SWAB  R3
7237 031522 000764          BR     7$
7238 031524 110137 004174  9$:  MOV     R1,@#DTADPB+10 :SAVE SECTOR ADDRESS
7239 031530 110237 004175  MOV     R2,@#DTADPB+11 :SAVE TRACK ADDRESS
7240 031534 010337 004176  MOV     R3,@#DTADPB+12 :SAVE CYLINDER ADDRESS
7241 031540 000200          RTS      R0          :RETURN
  
```

```

7242
7243      ; THIS ROUTINE IS USED TO INPUT THE "CONTROL SWITCHES".
7244      ; IF SWR<07>=1 THE PRESENT SETTING WILL BE TYPED AND THE NEW
7245      ; SETTING IS READ AND STORED.
7246      ; NOTE: THIS ROUTINE DESTROYS R3 AND R4
7247      ; CALL
7248      ;
7249      ;       JSR      PC,@#GETSWR
7250      ;       RETURN                      ; (C.SWR)=DESIRED CONTROL SWITCHES
7251 031542 032777 000200 147370 GETSWR: BIT      #SW07,@SWR      ; READ CONTROL SWITCHES?
7252 031550 001430                      BEQ      2$          ; NO--BRANCH
7253 031552 104401 03156C                      TYPE    ,65$       ; TYPE ASCIZ STRING
7254 031556 000410                      BR      64$       ; GET OVER THE ASCIZ
7255      ;:65$: .ASCIZ <CR><LF>/SET SWR<07>=0/
7256 031600                      64$:
7257 031600 012703 043216 1$:      MOV      #MSG.CS,R3      ; "CONTROL SWITCHES="
7258 031604 013704 001220                      MOV      @#C.SWR,R4      ; PRESENT CONTROL SWITCH SETTINGS
7259 031610 004037 031634                      JSR      R0,@#GETNUM     ; GET THE NEW SWITCH SETTINGS
7260 031614 000771                      BR      1$          ; COMMA
7261 031616 000240                      NOP                          ; PERIOD
7262 031620 013737 001220 001222                      MOV      C.SWR,SAVCSW    ; SAVE PREVIOUS VALUE
7263 031626 010437 001220                      MOV      R4,@#C.SWR     ; DOUBLE PERIOD--SAVE NEW SWITCH SETTING
7264 031632 000207 2$:      RTS      PC          ; RETURN FROM CALL
7265
7266      ; THIS ROUTINE WILL TYPE AN ASCIZ MESSAGE AND THEN
7267      ; INPUT AN ASCIZ STRING AND CHANGE THE STRING TO OCTAL
7268      ; IF REQUIRED.
7269      ; NOTE: THIS ROUTINE DESTROYS R1
7270      ; CALL
7271      ;       MOV      #ADR,R3          ; ADDRESS OF ASCIZ MESSAGE
7272      ;       MOV      #NUM,R4          ; OCTAL NUMBER
7273      ;       JSR      R0,@#GETNUM
7274      ;       RETURN1                    ; INPUT TERMINATED WITH A COMMA
7275      ;       RETURN2                    ; WITH A PERIOD
7276      ;       RETURN3                    ; WITH A DOUBLE PERIOD
7277      ;       ;                          ; R4=INPUT NUMBER AND
7278      ;       ;                          ; R2=R4*32 FOR ALL
7279      ;       ;                          ; THREE RETURNS
7280
7281 031634 010337 031642 GETNUM: MOV      R3,2$      ; SAVE MESSAGE POINTER
7282 031640 104401 1$:      TYPE                      ; TYPE THE MESSAGE
7283 031642 000000 2$:      .WORD    0          ; MESSAGE POINTER GOES HERE
7284 031644 010446                      MOV      R4,-(SP)       ; SAVE R4 FOR TYPEOUT
7285 031646 104402                      TYPOC                      ; GO TYPE--OCTAL ASCII(ALL DIGITS)
7286 031650 104401 043364                      TYPE    ,SLASH         ; /
7287 031654 104411                      RDLIN                      ; READ AN ASCIZ STRING
7288 031656 012601                      MOV      (SP)+,R1       ; ADDRESS OF FIRST CHARACTER
7289 031660 004037 034104                      JSR      R0,@#CK.CHR    ; CHECK ONE CHARACTER
7290 031664 031640 1$:      1$          ; ILLEGAL CHARACTER
7291 031666 031640 1$:      1$          ; CARRIAGE RETURN
7292 031670 031702 3$:      3$          ; /
7293 031672 031726 7$:      7$          ;
7294 031674 031734 8$:      8$          ;
7295 031676 031700 11$:     11$         ; DIGIT 0-9
7296 031700 005301 11$:     11$         ; DECREMENT THE INPUT POINTER
7297 031702 3$:      3$          ;

```

```

7298 031702 004037 034344      ISR      RO,@#CK.NUM      ;CHECK THE NUMBER
7299 031706 031640              1$              ;ILLEGAL INPUT
7300 031710 031722              6$              ;TERMINATED WITH A "... OR "CR"
7301 031712 031720              5$              ;TERMINATED WITH A "...
7302 031714 031716              4$              ;TERMINATED WITH A "...
7303 031716 005720      4$: TST      (RO)+      ;DOUBLE PERIOD
7304 031720 005720      5$: TST      (RO)+      ;SINGLE PERIOD
7305 031722 010204      6$: MOV      R2,R4      ;COMMA--SAVE INPUT NUMBER
7306 031724 000414              BR      10$        ;GO TO EXIT
7307 031726 105711      7$: TSTB     (R1)      ;TERMINATOR AFTER A COMMA?
7308 031730 001343              BNE     1$         ;NO--LOOP
7309 031732 000411              BR      10$        ;YES--EXIT
7310 031734 105711      8$: TSTB     (R1)      ;TERMINATOR AFTER A PERIOD?
7311 031736 001406              BEQ     9$         ;YES--EXIT
7312 031740 122721 000056      CMPB    #'.,(R1)+    ;NO--DOUBLE PERIOD?
7313 031744 001335              BNE     1$         ;NO--LOOP
7314 031746 105711      TSTB    (R1)      ;YES--TERMINATOR?
7315 031750 001333              BNE     1$         ;NO--LOOP
7316 031752 005720              TST     (RO)+      ;DOUBLE PERIOD
7317 031754 005720      9$: TST     (RO)+      ;PERIOD
7318 031756 010402      10$: MOV    R4,R2    ;COMMA--POSITION THE
7319 031760 000302              SWAB   R2          ;NUMBER IN CASE IT
7320 031762 006202              ASR    R2          ;IS THE PRIORITY LEVEL
7321 031764 006202              ASR    R2
7322 031766 006202              ASR    R2
7323 031770 000200              RTS     RO          ;EXIT
7324
7325      ;THIS ROUTINE IS USED TO CHANGE OR MODIFY
7326      ;THE TEST PARAMETERS. IT GIVES THE OPERATOR
7327      ;THE CAPABILITY OF SPECIFYING WHICH DRIVES TO TEST, WHICH
7328      ;TESTS TO RUN AND HOW MANY TIMES TO
7329      ;REPEAT EACH TEST
7330
7331 031772 104412      GT.PRM: SAVREG      ;SAVE R0 - R5
7332 031774 005037 001272      GT.PR1: CLR      DRVSEL ;NO DRIVE SELECTED
7333 032000 104401 032006              TYPE     ,65$      ;;TYPE ASCIZ STRING
7334 032004 000406              BR      64$        ;;GET OVER THE ASCIZ
7335      ;;65$: .ASCIZ <CR><LF>/DRIVE(S)=/
7336      64$:
7337 032022 104411      RDLIN     ;READ TTY
7338 032024 012601      MOV      (SP)+,R1    ;ADDRESS OF ASCIZ STRING
7339 032026 004037 034104      JSR      RO,@#CK.CHR ;CHECK ONE CHARACTER
7340 032032 031774              GT.PR1     ;ILLEGAL CHARACTER
7341 032034 031774              GT.PR1     ;CARRIAGE RETURN
7342 032036 031774              GT.PR1     ;"/"
7343 032040 031774              GT.PR1     ;".."
7344 032042 031774              GT.PR1     ;"...
7345 032044 032046              1$        ;DIGIT 0-9
7346 032046 005301      1$: DEC     R1
7347 032050      2$:
7348 032050 012702 000007      MOV      #7,R2      ;UPPER LIMIT OF INPUT
7349 032054 004037 034160      JSR      RO,@#CK.DIG ;CHECK THE DIGIT(S)
7350 032060 031774              GT.PR1     ;ILLEGAL INPUT
7351 032062 031774              GT.PR1     ;INPUT TO LARGE
7352 032064 032072              3$        ;TERMINATED WITH A "... OR "CR"
7353 032066 032116              4$        ;TERMINATED WITH A "...

```

```

7354 032070 032116          4$          ;TERMINATED WITH A ". ."
7355 032072 156237 034620 001232 3$:  BISB  ATABIT(R2),DRVSEL ;SET THE DRIVE SELECTED BIT
7356 032100 105741          TSTB  -(R1)           ;WAS THE LINE TERMINATED?
7357 032102 001362          BNE   2$             ;NO-GET THE NEXT DRIVE
7358 032104 005037 001234  CLR   @#TSTNMS       ;DESELECT ALL TESTS
7359 032110 005037 001236  CLR   TSTNMS+2
7360 032114 000405          BR    GTTST1         ;YES--SELECT TEST
7361 032116 156237 034620 001232 4$:  BISB  ATABIT(R2),DRVSEL ;SET THE SELECTED DRIVE BITS
7362 032124 104413  GT.PR2: RESREG      ;RESTORE R0 - R5
7363 032126 000207          RTS   PC             ;EXIT
7364
7365 032130          GTTST1:
7366 032130 104401 032136  TYPE   ,65$          ;;TYPE ASCIZ STRING
7367 032134 000403          BR    64$          ;;GET OVER THE ASCIZ
7368          ;;65$: .ASCIZ /TEST=/
7369 032144 64$:
7370 032144 104411          RDLIN          ;READ AN ASCIZ STRING
7371 032146 012601          MOV   (SP)+,R1     ;POINTER TO R1
7372 032150 122711 000056  CMPB  #'.,(R1)     ;DOUBLE PERIOD?
7373 032154 001007          BNE   1$           ;NO--BRANCH
7374 032156 122761 000056 000001  CMPB  #'.,1(R1)
7375 032164 001003          BNE   1$
7376 032166 105761 000002  TSTB  2(R1)        ;'CR'?
7377 032172 001754          BEQ   GT.PR2       ;YES--EXIT
7378 032174 005037 001234 1$:  CLR   TSTNMS       ;NO TEST SELECTED
7379 032200 005037 001236  CLR   TSTNMS+2
7380 032204 005037 001240  CLR   OPNFLG       ;NO TESTS TO BE OPENED
7381 032210 005037 001242  CLR   OPNFLG+2
7382 032214 121127 000123  GTTST2: CMPB  (R1),#'S ;ALL SEEK TESTS?
7383 032220 001094          BNE   1$           ;NO--BRANCH
7384 032222 052737 000777 001234  BIS   #777,TSTNMS ;YES--SELECT TESTS 0-10
7385 032230 000552          BR    GTTST3
7386 032232 121127 000124 1$:  CMPB  (R1),#'T     ;ALL TIMING TESTS?
7387 032236 001004          BNE   2$           ;NO--BRANCH
7388 032240 052737 036000 001234  BIS   #36000,TSTNMS ;YES--SELECT TESTS 12-15
7389 032246 000543          BR    GTTST3
7390 032250 121127 000101 2$:  CMPB  (R1),#'A     ;ALL ADDRESSING TESTS?
7391 032254 001004          BNE   3$           ;NO--BRANCH
7392 032256 052737 140000 001234  BIS   #140000,TSTNM ;YES--SELECT TESTS 16 & 17
7393 032264 000534          BR    GTTST3
7394 032266 121127 000104 3$:  CMPB  (R1),#'D     ;DATA TEST?
7395 032272 001004          BNE   4$           ;NO--BRANCH
7396 032274 052737 000001 001236  BIS   #1,TSTNMS+2 ;YES--SELECT TEST 20
7397 032302 000525          BR    GTTST3
7398 032304 121127 000105 4$:  CMPB  (R1),#'E     ;EXERCISER TEST?
7399 032310 001004          BNE   5$           ;NO--BRANCH
7400 032312 052737 000002 001236  BIS   #2,TSTNMS+2 ;YES--SELECT TEST 21
7401 032320 000516          BR    GTTST3
7402 032322 004037 034030 5$:  JSR   R0,@#CK.OCT ;OCTAL DIGIT?
7403 032326 000514          BR    GTTST4       ;NO--BRANCH
7404 032330 010205          MOV   R2,R5        ;YES--SAVE IT
7405 032332 005201          INC   R1           ;MOVE TO NEXT CHARACTER
7406 032334 004037 034030  JSR   R0,@#CK.OCT ;OCTAL DIGIT
7407 032340 000405          BR    6$           ;NO--BRANCH
7408 032342 005201          INC   R1           ;MOVE TO NEXT CHARACTER
7409 032344 006305          ASL   R5           ;SCALE HIGH DIGIT

```



```

7410 032346 006305      ASL      R5
7411 032350 006305      ASL      R5
7412 032352 060502      ADD      R5,R2      ;COMBINE HIGH & LOW DIGITS
7413 032354 020227 000022    6$:  CMP      R2,#$TN-1  ;VALID TEST NUMBER?
7414 032360 003263      BGT      GTTST1     ;NO--BRANCH
7415 032362 010237 032554    MOV      R2,9$      ;SAVE THE TEST NUMBER
7416 032366 010204      MOV      R2,R4      ;CONVERT TEST NUMBER INTO AN INDEX
7417 032370 042704 000017    BIC      #17,R4     ;CLEAR UNWANTED BITS
7418 032374 006204      ASR      R4         ;SHIFT THE BITS
7419 032376 006204      ASR      R4         ;SHIFT THE BITS
7420 032400 006204      ASR      R4         ;SHIFT THE BITS
7421 032402 006302      ASL      R2
7422 032404 056264 001424 001234  BIS      BITS(R2),TSTNMS(R4) ;SELECT TEST
7423 032412 121127 000055    (MPB)   (R1),#' -    ;TEST STRING?
7424 032416 001060      BNE      GTTST4     ;NO--BRANCH
7425 032420 005201      INC      R1         ;YES--MOVE TO NEXT CHARACTER
7426 032422 004037 034030    JSR      RO,@#CK.OCT ;OCTAL DIGIT?
7427 032426 000640      BR       GTTST1     ;NO--BRANCH
7428 032430 010205      MOV      R2,R5     ;YES--SAVE IT
7429 032432 005201      INC      R1         ;MOVE TO NEXT CHARACTER
7430 032434 004037 034030    JSR      RO,@#CK.OCT ;OCTAL DIGIT?
7431 032440 000405      BR       7$        ;NO--BRANCH
7432 032442 005201      INC      R1         ;YES--MOVE TO NEXT CHARACTER
7433 032444 006305      ASL      R5         ;SCALE HIGH DIGIT
7434 032446 006305      ASL      R5
7435 032450 006305      ASL      R5
7436 032452 060502      ADD      R5,R2     ;COMBINE HIGH & LOW DIGIT
7437 032454 020227 000022    7$:  CMP      R2,#$TN-1  ;VALID TEST NUMBER?
7438 032460 003223      BGT      GTTST1     ;NO--BRANCH
7439 032462 023702 032554    CMP      9$,R2     ;IS THE FIRST NUMBER OF THE
7440                                ;STRING SMALLER THAN THE LAST?
7441                                ;NO--BRANCH
7441 032466 002220      BGE      GTTST1     ;NO--BRANCH
7442 032470 010246      MOV      R2,-(SP)  ;SAVE ENDING TEST NUMBER
7443 032472 013702 032554    MOV      9$,R2     ;GET STARTING TEST NUMBER
7444 032476 012637 032554    MOV      (SP)+,9$  ;STORE ENDING TEST NUMBER
7445 032502 006337 032554    ASL      9$        ;SHIFT ENDING TEST NUMBER
7446 032506 006302      ASL      R2        ;SHIFT TEST NUMBER
7447 032510 010204      8$:  MOV      R2,R4     ;COPY TEST NUMBER INTO R4
7448 032512 042704 000037    BIC      #37,R4    ;CLEAR LOWER BITS
7449 032516 006204      ASR      R4        ;SHIFT THE TEST NUMBER
7450 032520 006204      ASR      R4        ;SHIFT THE TEST NUMBER
7451 032522 006204      ASR      R4        ;SHIFT THE TEST NUMBER
7452 032524 006204      ASR      R4        ;SHIFT THE TEST NUMBER
7453 032526 056264 001424 001234  BIS      BITS(R2),TSTNMS(R4) ;SELECT THE TEST
7454 032534 062702 000002    ADD      #2,R2     ;INCREMENT THE TEST NUMBER
7455 032540 020237 032554    CMP      R2,9$     ;SEE IF FINISHED
7456 032544 101761      BLOS     8$        ;BR IF NOT
7457 032546 162702 000002    SUB      #2,R2     ;CORRECT TEST NUMBER
7458 032552 000402      BR       GTTST4     ;CONTINUE
7459 032554 000000      9$:  .WORD    0        ;STORE TEST NUMBER HERE
7460 032556 005201      GTTST3: INC      R1  ;MOVE TO NEXT CHARACTER
7461 032560 121127 000056    GTTST4: (MPB)   (R1),#' . ;"PERIOD"?
7462 032564 001441      BEQ      GTTST5     ;YES--BRANCH
7463 032566 005737 001234    TST      TSTNMS    ;ANY TEST SELECTED THIS CYCLE?
7464 032572 001005      BNE      1$        ;BR IF YES
7465 032574 005737 001236    TST      TSTNMS+2  ;ANY TEST SELECTED THIS CYCLE ?

```

```

7466 032600 001002          BNE 1$          ;BR IF YES
7467 032602 000137 032130    JMP GTTST1      ;NO
7468 032606 121127 000057    1$: CMPB (R1),#'/' ;'OPEN'?
7469 032612 001004          BNE 2$          ;NO--BRANCH
7470 032614 056264 001424 001240  BIS BITS(R2), OPNFLG(R4) ;YES--SET BITS FOR TEST TO OPEN
7471 032622 000405          BR 3$
7472 032624 121127 000054    2$: CMPB (R1),#'. ;'COMMA'?
7473 032630 001402          BEQ 3$          ;BR IF YES
7474 032632 000137 032130    JMP GTTST1      ;NO
7475 032636 005201          3$: INC R1       ;MOVE TO NEXT CHARACTER
7476 032640 105711          TSTB (R1)       ;'CR'?
7477 032642 001402          BEQ 4$          ;BR IF 'CR'
7478 032644 000137 032214    JMP GTTST2      ;NO--GO GET NEXT CHARACTER
7479 032650 005737 001240    4$: TST OPNFLG   ;ANY TESTS TO OPEN ?
7480 032654 001042          BNE OPNTST      ;BR IF YES
7481 032656 005737 001242    TST OPNFLG+2    ;ANY TESTS TO OPEN ?
7482 032662 001037          BNE OPNTST      ;BR IF YES
7483 032664 000137 032130    JMP GTTST1      ;NO--START AGAIN
7484 032670 005201          GTTST5: INC R1  ;MOVE TO NEXT CHARACTER
7485 032672 121127 000056    CMPB (R1),#'.' ;'PERIOD'?
7486 032676 001414          BEQ GTTST6      ;YES--BRANCH
7487 032700 105711          TSTB (R1)       ;'CR'?
7488 032702 001402          BEQ 1$          ;YES--BRANCH
7489 032704 000137 032130    JMP GTTST1
7490 032710 005737 001240    1$: TST OPNFLG   ;ANY TESTS TO OPEN ?
7491 032714 001022          BNE OPNTST      ;BR IF YES
7492 032716 005737 001242    TST OPNFLG+2    ;ANY TESTS TO OPEN ?
7493 032722 001017          BNE OPNTST      ;BR IF YES
7494 032724 000137 032124    JMP GT.PR2      ;NO--GO START TESTING
7495 032730 005201          GTTST6: INC R1  ;MOVE TO NEXT CHARACTER
7496 032732 105711          TSTB (R1)       ;'CR'?
7497 032734 001402          BEQ 1$          ;YES--BRANCH
7498 032736 000137 032130    JMP GTTST1      ;NO--GO ASK FOR TEST
7499 032742 005737 001240    1$: TST OPNFLG   ;ANY TESTS TO OPEN ?
7500 032746 001005          BNE OPNTST      ;BR IF YES
7501 032750 005737 001242    TST OPNFLG+2    ;ANY TESTS TO OPEN ?
7502 032754 001002          BNE OPNTST      ;BR IF YES
7503 032756 000137 032124    JMP GT.PR2      ;NO--GO START TESTING
7504
7505 ;OPEN THE SELECTED TEST FOR CHANGES
7506
7507 032762 104412          OPNTST: SAVREG  ;SAVE R0 - R5
7508 032764 005027          CLR (PC)+      ;START WITH TEST 0
7509 032766 000000          OPN.CT: .WORD 0 ;COUNT STORED HERE
7510 032770 000411          BR OPN.2       ;SKIP THE INCREMENT
7511 032772 005237 032766    OPN.1: INC OPN.CT ;MOVE TO THE NEXT TEST
7512 032776 022737 000022 032766  CMP #5TN-1,OPN.CT ;TEST NUMBER TOO BIG?
7513 033004 002003          BGE OPN.2      ;NO--OPEN THE NEXT TEST
7514 033006 104413          RESREG         ;RESTORE R0 - R5
7515 033010 000137 032130    JMP GTTST1      ;YES--GO ASK FOR MORE TESTS
7516 033014 013705 032766    OPN.2: MOV OPN.CT,R5 ;SETUP TO USE THE
7517 033020 006305          ASL R5         ;TEST NUMBER AS AN INDEX
7518 033022 013703 032766    MOV OPN.CT,R3  ;GET INDEX
7519 033026 042703 000017    BIC #17,R3     ;CLEAR LOWER TEST BITS
7520 033032 006203          ASR R3         ;SHIFT TEST NUMBER
7521 033034 006203          ASR R3         ;SHIFT TEST NUMBER

```

```

7522 033036 006203          ASR      R3          ;SHIFT TEST NUMBER
7523 033040 036563 001424 001240 BIT      BITS(R5),OPNFLG(R3) ;OPEN THIS TEST?
7524 033046 001751          BEQ      OPN.1        ;NO--MOVE TO NEXT TEST
7525 033050 104401 033056 TYPE     .65$        ;:TYPE ASCIZ STRING
7526 033054 000404          BR       64$        ;:GET OVER THE ASCIZ
7527          ;:65$: .ASCIZ / TEST /
7528 033066          64$:
7529 033066 013746 032766 MOV      OPN.CT,-(SP) ;:SAVE OPN.CT FOR TYPEOUT
7530          ;:TEST NUMBER
7531 033072 104403          TYPOS     ;:GO TYPE--OCTAL ASCII
7532 033074 002          .BYTE     2          ;:TYPE 2 DIGIT(S)
7533 033075 000          .BYTE     0          ;:SUPPRESS LEADING ZEROS
7534 033076 104401 001215 TYPE     ,SCLRF      ;:TYPE 'CR' & 'LF'
7535 033102 016500 001536 MOV      PRMPT(R5),R0 ;:PICKUP PARAMETER POINTER
7536 033106 011046          MOV      (R0),-(SP) ;:SAVE THE VARIABLE INDICATOR
7537 033110 012702 001504 MOV      #PRM,R2     ;:FIRST ADDRESS OF TABLE
7538 033114 000405          BR       2$
7539 033116 006216          1$: ASR      (SP)        ;:CHECK FOR A VARIABLE
7540 033120 103403          BCS      2$        ;:GO MOVE THIS ONE
7541 033122 001404          BEQ      OPNPRM    ;:DONE
7542 033124 005722          TST      (R2)+     ;:BUMP THE POINTER
7543 033126 000773          BR       1$
7544 033130 012022          2$: MOV      (R0)+,(R2)+ ;:MOVE THIS VARIABLE INTO THE
7545 033132 000771          BR       1$        ;:COMMON AREA
7546 033134 013716 001504 OPNPRM: MOV @#PRM,(SP) ;:GET THE VARIABLE INDICATOR
7547 033140 005004          CLR      R4        ;:ZERO THE INDEX
7548 033142 006216          1$: ASR      (SP)        ;:CHECK FOR A VARIABLE
7549 033144 103403          BCS      3$        ;:GO GET IT
7550 033146 001772          BEQ      OPNPRM    ;:OUT OF VARIABLES
7551 033150 005724          2$: TST      (R4)+     ;:UPDATE THE INDEX
7552 033152 000773          BR       1$
7553 033154 005764 001606          3$: TST      PRMLMT(R4) ;:IS THE MAX. MAGNITUDE NEG?
7554 033160 100466          BMI      OPNPAT    ;:YES--THEN IT IS THE PATTERN
7555 033162 104401 044330 TYPE     ,MSG.SP     ;:TYPE SPACES
7556 033166 016437 001636 033176 MOV      PRMSG(R4),4$ ;:TYPE THE NAME OF THIS VARIABLE
7557 033174 104401          TYPE
7558 033176 000000          4$: .WORD     0
7559 033200 104401 043214 TYPE     ,MSG.EQ    ;:TYPE '='
7560 033204 016446 001506 MOV      RPT(R4),-(SP) ;:PUT RPT(R4) ON THE STACK
7561 033210 004737 023336 JSR      PC,@#SSB2D ;:CHANGE RPT(R4) TO DECIMAL ASCII
7562 033214 004737 023566 JSR      PC,@#SSUPRS ;:TYPE WITHOUT LEADING ZEROS
7563 033220 104401 043364 TYPE     ,SLASH     ;: '/'
7564 033224 104411          RDLIN
7565 033226 012601          MOV      (SP)+,R1 ;:READ AN ASCII STRING
7566 033230 004037 034104 JSR      RO,@#CK.CHR ;:CHECK ONE CHARACTER
7567 033234 033154          3$          ;:ILLEGAL CHARACTER
7568 033236 033150          2$          ;:CARRIAGE RETURN
7569 033240 033306          8$          ;:
7570 033242 033250          5$          ;:
7571 033244 033256          6$          ;:
7572 033246 033304          7$          ;:DIGIT 0-9
7573 033250 105711          5$: TSTB     (R1)    ;:'CR'?
7574 033252 001340          BNE      3$        ;:NO--STAY ON THIS VARIABLE
7575 033254 000735          BR       2$        ;:YES--MOVE TO NEXT VARIABLE
7576 033256 105711          6$: TSTB     (R1)    ;:IS THERE A 'CR' AFTER THE PERIOD?
7577 033260 001002          BNE      64$      ;:NO

```

```

7578 033262 000137 033676          JMP      OPN.N2          ;YES--GO CLOSE THIS TEST
7579 033266 122721 000056    64$:  CMPB    #'.,(R1)+    ;DOUBLE PERIOD?
7580 033272 001330          BNE     3$              ;NO--GO ASK FOR THIS VARIABLE
7581 033274 105711          TSTB   (R1)            ;YES--IS A 'CR' AFTER THE DOUBLE PERIOD?
7582 033276 001326          BNE     3$              ;NO--ASK FOR THIS VARIABLE AGAIN
7583 033300 000137 033714          JMP     OPN.X2          ;YES--CLOSE ALL TEST
7584 033304 005301          7$:   DEC     R1         ;BACK THE POINTER UP BY ONE
7585 033306          8$:
7586 033306 016402 001606          MOV     PRMLMT(R4),R2   ;UPPER LIMIT OF INPUT
7587 033312 004037 034160          JSR    RO,@#CK.DIG     ;CHECK THE DIGIT(S)
7588 033316 033154          3$
7589 033320 033154          3$
7590 033322 033330          9$:   9$
7591 033324 033672          OPN.N1                  ;TERMINATED WITH A '.,' OR 'CR'
7592 033326 033710          OPN.X1                  ;TERMINATED WITH A '.,'
7593 033330 010264 001506    9$:   MOV     R2,RPT(R4)   ;SAVE THIS VARIABLE
7594 033334 000705          BR     2$              ;MOVE TO NEXT VARIABLE
7595 033336 104401 044330    OPNPAT: TYPE ,MSG.SP    ;TYPE SPACES
7596 033342 104401 043210          TYPE ,MSG.PAT          ;TYPE 'PAT'
7597 033346 104401 043214          TYPE ,MSG.EQ           ;TYPE '='
7598 033352 016446 001506          MOV     RPT(R4),-(SP)  ;;SAVE RPT(R4) FOR TYPEOUT
7599 033356 104402          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
7600 033360 104401 044331          TYPE ,MSG.SP+1        ;TYPE ONE SPACE
7601 033364 104411          RDLIN                   ;READ ASCII STRING
7602 033366 012601          MOV     (SP)+,R1       ;PICKUP POINTER
7603 033370 004037 034104          JSR    RO,@#CK.CHR    ;CHECK ONE CHARACTER
7604 033374 033336          OPNPAT                  ;ILLEGAL CHARACTER
7605 033376 033134          OPNPRM                  ;CARRIAGE RETURN
7606 033400 033432          3$
7607 033402 033134          OPNPRM                  ;
7608 033404 033410          1$:   1$
7609 033406 033430          2$:   2$
7610 033410 105711          TSTB   (R1)            ;'CR' AFTER THE PERIOD?
7611 033412 001531          BEQ    OPN.N2          ;YES--GO CLOSE THIS TEST
7612 033414 122721 000056    CMPB   #'.,(R1)+      ;NO--PERIOD?
7613 033420 001346          BNE    OPNPAT          ;NO--LOOP
7614 033422 105711          TSTB   (R1)            ;'CR' AFTER A DOUBLE PERIOD?
7615 033424 001533          BEQ    OPN.X2          ;YES--GO START TESTING
7616 033426 000743          BR     OPNPAT          ;NO--LOOP
7617 033430 005301          2$:   DEC     R1         ;BACKUP THE ASCII POINTER
7618 033432          3$:
7619 033432 004037 034344          JSR    RO,@#CK.NUM     ;CHECK THE NUMBER
7620 033436 033336          OPNPAT                  ;ILLEGAL INPUT
7621 033440 033446          4$:   4$
7622 033442 033672          OPN.N1                  ;TERMINATED WITH A '.,' OR 'CR'
7623 033444 033710          OPN.X1                  ;TERMINATED WITH A '.,'
7624 033446 010264 001506    4$:   MOV     R2,RPT(R4)   ;SAVE THE INPUT NUMBER
7625 033452 006002          ROR    R2               ;OPEN PATTERN 0?
7626 033454 103227          BCC   OPNPRM           ;NO--START AT BEGINNING OF PARAMETER TABLE
7627 033456 104412          SAVREG                   ;SAVE RO - R5
7628 033460 005000          OPNWDS: CLR    RO       ;START WITH WORD 0
7629 033462 012704 003104          MOV    #PAT0,R4
7630 033466          1$:
7631 033466 104401 033474          TYPE   ,65$            ;;TYPE ASCII STRING
7632 033472 000403          BR     64$            ;;GET OVER THE ASCII
7633          ;;65$: .ASCII / WD/

```

7634	033502		64\$:	MOV	RO,-(SP)	:PUT RO ON THE STACK
7635	033502	010046		JSR	PC,@#SSB2D	:CHANGE RO TO DECIMAL ASCII
7636	033504	004737	023336	JSR	PC,@#SSUPRS	:TYPE WITHOUT LEADING ZEROS
7637	033510	004737	023566	TYPE	,MSG.EQ	:TYPE "="
7638	033514	104401	043214	MOV	(R4),-(SP)	:SAVE (R4) FOR TYPEOUT
7639	033520	011446		TYPOC		:GO TYPE--OCTAL ASCII(ALL DIGITS)
7640	033522	104402		RDLIN		:READ ASCII STRING
7641	033524	104411		MOV	(SP)+,R1	:PICKUP THE POINTER
7642	033526	012601		JSR	RO,@#CK.CHR	:CHECK ONE CHARACTER
7643	033530	004037	034104	1\$		:ILLEGAL CHARACTER
7644	033534	033466		4\$		:CARRIAGE RETURN
7645	033536	033570		2\$		:/
7646	033540	033552		4\$		..
7647	033542	033570		5\$		...
7648	033544	033604		10\$		:DIGIT 0-9
7649	033546	033550		DEC	R1	:BACKUP THE ASCII POINTER
7650	033550	005301	10\$:			
7651	033552		2\$:	JSR	RO,@#CK.NUM	:CHECK THE NUMBER
7652	033552	004037	034344	1\$		:ILLEGAL INPUT
7653	033556	033466		3\$		:TERMINATED WITH A "... OR "CR"
7654	033560	033566		6\$		:TERMINATED WITH A "...
7655	033562	033624		8\$		:TERMINATED WITH A "...
7656	033564	033636		3\$:	MOV	R2,(R4)
7657	033566	010214		4\$:	TST	(R4)+
7658	033570	005724			INC	RO
7659	033572	005200			CMP	#16.,RO
7660	033574	022700	000020		BGT	1\$
7661	033600	003332			BR	OPNWDS
7662	033602	000726		5\$:	TSTB	(R1)
7663	033604	105711			BEQ	7\$
7664	033606	001407			CMPB	#'.,(R1)+
7665	033610	122721	000056		BNE	1\$
7666	033614	001324			TSTB	(R1)
7667	033616	105711			BEQ	9\$
7668	033620	001407			BR	1\$
7669	033622	000721		6\$:	MOV	R2,(R4)+
7670	033624	010224		7\$:	JSR	PC,@#CLSWDS
7671	033626	004737	033650		RESREG	
7672	033632	104413			BR	OPN.N2
7673	033634	000420		8\$:	MOV	R2,(R4)+
7674	033636	010224		9\$:	JSR	PC,@#CLSWDS
7675	033640	004737	033650		RESREG	
7676	033644	104413			BR	OPN.X2
7677	033646	000422		CLSWDS:	MOV	#PATO,R1
7678	033650	012701	003104	1\$:	INC	RO
7679	033654	005200			CMP	#15.,RO
7680	033656	022700	000017		BLT	2\$
7681	033662	002402			MOV	(R1)+,(R4)+
7682	033664	012124			BR	1\$
7683	033666	000772		2\$:	RTS	PC
7684	033670	000207		OPN.N1:	MOV	R2,RPT(R4)
7685	033672	010264	001506	OPN.N2:	TST	(SP)+
7686	033676	005726			JSR	PC,CLOSE
7687	033700	004737	033750		JMP	OPN.1
7688	033704	000137	032772		MOV	R2,RPT(R4)
7689	033710	010264	001506	OPN.X1:		

```

7690 033714 005726          OPN.X2: TST      (SP)+          ;CLEAN OFF THE STACK
7691 033716 004737 033750 1$: JSR      PC,CLOSE        ;CLOSE THIS TEST
7692 033722 005725          2$: TST      (R5)+          ;UPDATE THE INDEX
7693 033724 020527 000034  CMP      R5,#16*2        ;INDEX TO BIG?
7694 033730 002403          BLT          3$           ;NO--BRANCH
7695 033732 104413          RESREG      ;RESTORE R0 - R5
7696 033734 000137 032124  JMP      GT,PR2         ;GO TO EXIT
7697 033740 036503 001424  3$: BIT      BITS(R5),R3    ;IS THIS TEST OPEN FOR CHANGE?
7698 033744 001364          BNE          1$           ;YES--GO CLOSE IT
7699 033746 000765          BR          2$           ;NO--MOVE TO NEXT TEST
7700 033750 104412          CLOSE: SAVREG          ;SAVE R0 - R5
7701 033752 012700 001504  MOV      #PRM,R0        ;"FROM" ADDRESS
7702 033756 016501 001536  MOV      PRMP1(R5),R1   ;"TO" ADDRESS
7703 033762 012002          MOV      (R0)+,R2       ;"FROM" INDICATOR
7704 033764 012103          MOV      (R1)+,R3       ;"TO" INDICATOR
7705 033766 012704 000001  MOV      #1,R4          ;TEST BIT START A "RPT"
7706 033772 030402          1$: BIT      R4,R2        ;PARAMETER TO BE MOVED?
7707 033774 001403          BEQ          2$           ;NO--BRANCH
7708 033776 030403          BIT      R4,R3         ;A PLACE TO PUT IT?
7709 034000 001404          BEQ          3$           ;NO--BRANCH
7710 034002 011011          MOV      (R0),(R1)      ;YES--MOVE "FROM" TO "TO"
7711 034004 030403          2$: BIT      R4,R3         ;"TO" PARAMETER?
7712 034006 001401          BEQ          3$           ;NO--BRANCH
7713 034010 005721          TST      (R1)+          ;YES--UPDATE THE POINTER
7714 034012 005720          3$: TST      (R0)+          ;UPDATE FROM POINTER
7715 034014 006304          ASL      R4             ;POSITION THE TEST BIT
7716 034016 032704 002000  BIT      #BIT10,R4      ;DONE?
7717 034022 001763          BEQ          1$           ;NO--BRANCH
7718 034024 104413          RESREG      ;RESTORE R0 - R5
7719 034026 000207          RTS      PC            ;RETURN
7720          ;THIS ROUTINE IS USED TO CHECK IF AN
7721          ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
7722          ;CALL
7723          ;      MOV      #ADR,R1          ;ADDRESS OF ASCII CHARACTER
7724          ;      JSR      R0,@#CK.OCT    ;CHECK THE CHARACTER
7725          ;      RETURN1          ;CHARACTER IS NOT BETWEEN 0-7
7726          ;      RETURN2          ;CHARACTER IS IN R2 AS A
7727          ;      ;OCTAL DIGIT
7728
7729 034030 121127 000060  CK.OCT: CMPB      (R1),#'0    ;LESS THAN ZERO?
7730 034034 103407          BLO          1$           ;YES -- BRANCH
7731 034036 121127 000067  CMPB      (R1),#'7      ;GREATER THAN SEVEN?
7732 034042 101004          BHI          1$           ;YES -- BRANCH
7733 034044 111102          MOVB      (R1),R2       ;GET THE CHARACTER
7734 034046 042702 177770  BIC      #^C7,R2        ;STRIP AWAY THE ASCII
7735 034052 005720          TST      (R0)+          ;ADJUST FOR RETURN
7736 034054 000200          1$: RTS      R0         ;RETURN
7737
7738          ;THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
7739          ;AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
7740          ;CALL
7741          ;      MOV      #ADR,R1          ;ADDRESS OF ASCII CHARACTER
7742          ;      JSR      R0,@#CK.DEC    ;CHECK THE CHARACTER
7743          ;      RETURN1          ;NOT BETWEEN 0 AND 9
7744          ;      RETURN2          ;BETWEEN 0 AND 9
7745          ;      ;R2 - DIGIT

```

```

7746
7747 034056 121127 000060 CK.DEC: CMPB (R1),#'0 ;LESS THAN ZERO?
7748 034062 103407 BLO 1$ ;YES -- BRANCH
7749 034064 121127 000071 CMPB (R1),#'9 ;GREATER THAN NINE?
7750 034070 101004 BHI 1$ ;YES -- BRANCH
7751 034072 111102 MOVB (R1),R2 ;GET THE CHARACTER
7752 034074 042702 000060 BIC #'0,R2 ;STRIP AWAY THE ASCII
7753 034100 005720 TST (R0)+ ;ADJUST FOR RETURN
7754 034102 000200 1$: RTS R0 ;RETURN
7755
7756 ;THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
7757 ;DETERMINE WHAT IT IS.
7758 ;CALL
7759 ; MOV #ADR,R1 ;ADDRESS OF ASCII CHARACTER
7760 ; JSR R0,@#CK.CHR ;CHECK CHARACTER
7761 ; RETURN ADR1 ;UNKNOWN CHARACTER
7762 ; RETURN ADR2 ;CARRIAGE RETURN * (R1)=ADR+1
7763 ; RETURN ADR3 ;SLASH * (R1)=ADR+1
7764 ; RETURN ADR4 ;COMMA * (R1)=ADR+1
7765 ; RETURN ADR5 ;PERIOD * (R1)=ADR+1
7766 ; RETURN ADR6 ;DIGIT BETWEEN 0 AND 9.
7767 ; ;R2 = DIGIT * (R1)-ADR+1
7768
7769 034104 105711 CK.CHR: TSTB (R1) ;'CARRIAGE RETURN'?
7770 034106 001420 BEQ 4$ ;YES -- BRANCH
7771 034110 121127 000057 CMPB (R1),#'/' ;'SLASH'?
7772 034114 001414 BEQ 3$ ;YES -- BRANCH
7773 034116 121127 000054 CMPB (R1),#',' ;'COMMA'?
7774 034122 001410 BEQ 2$ ;YES -- BRANCH
7775 034124 121127 000056 CMPB (R1),# '.' ;'PERIOD'?
7776 034130 001404 BEQ 1$ ;YES -- BRANCH
7777 034132 004037 034056 JSR R0,@#CK.DEC ;'DIGIT'?
7778 034136 000406 BR 5$ ;NO -- BRANCH
7779 034140 005720 TST (R0)+ ;DIGIT BETWEEN 0-9
7780 034142 005720 1$: TST (R0)+ ;PERIOD
7781 034144 005720 2$: TST (R0)+ ;COMMA
7782 034146 005720 3$: TST (R0)+ ;SLASH
7783 034150 005720 4$: TST (R0)+ ;CARRIAGE RETURN
7784 034152 005201 INC R1 ;MOVE POINTER TO NEXT CHARACTER
7785 034154 011000 5$: MOV (R0),R0 ;UNKNOWN CHARACTER
7786 034156 000200 RTS R0 ;RETURN
7787
7788 ;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
7789 ;CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
7790 ;CALL
7791 ; MOV #ADR,R1 ;ADDRESS OF ASCII STRING
7792 ; MOV #NUM,R2 ;MAX. MAGNITUDE OF INPUT NUMBER
7793 ; JSR R0,@#CK.DIG ;CHECK DIGITS
7794 ; RETURN ADR1 ;ILLEGAL CHARACTER -- R2=?
7795 ; RETURN ADR2 ;INPUT NUMBER TOO LARGE -- R2=?
7796 ; RETURN ADR3 ;'COMMA' -- R2 = NUMBER
7797 ; RETURN ADR4 ;'PERIOD' -- R2 = NUMBER
7798 ; RETURN ADR5 ;'PERIOD-PERIOD' -- R2 NUMBER
7799
7800 034160 010446 CK.DIG: MOV R4,-(SP) ;SAVE R4
7801 034162 010346 MOV R3,-(SP) ;SAVE R3

```

```

7802 034164 010246          MOV     R2,-(SP)          ;SAVE THE MAX. SIZE ON THE STACK
7803 034166 005^02          CLR     R2              ;START WITH 0
7804 034170 005003          CLR     R3
7805 034172 005004          CLR     R4
7806 034174 004037 034104   JSR     R0,@#CK.CHR     ;CHECK ONE CHARACTER
7807 034200 034330          8$     ;ILLEGAL CHARACTER
7808 034202 034330          8$     ;CARRIAGE RETURN
7809 034204 034330          8$     ;"/"
7810 034206 034330          8$     ;".."
7811 034210 034330          8$     ;".."
7812 034212 034214          1$     ;DIGIT 0-9
7813 034214 006303          1$:   ASL     R3          ;2
7814 034216 010346          MOV     R3,-(SP)       ;SAVE *2
7815 034220 006303          ASL     R3             ;4
7816 034222 006303          ASL     R3             ;8
7817 034224 062603          ADD     (SP)+,R3       ;(*8)+(*2)=*10.
7818 034226 060203          ADD     R2,R3         ;UPDATE THE INPUT NUMBER
7819 034230 004037 034104   JSR     R0,@#CK.CHR     ;CHECK ONE CHARACTER
7820 034234 034330          8$     ;ILLEGAL CHARACTER
7821 034236 034250          9$     ;CARRIAGE RETURN
7822 034240 034330          8$     ;"/"
7823 034242 034256          3$     ;".."
7824 034244 034254          2$     ;".."
7825 034246 034214          1$     ;DIGIT 0-9
7826 034250 005301          9$:   DEC     R1        ;BACKUP THE CHARACTER POINTER
7827 034252 000401          BR     3$             ;CONTINUE
7828 034254 005724          2$:   TST     (R4)+     ;"PERIOD"
7829 034256 005724          3$:   TST     (R4)+     ;"COMMA" OR "CR"
7830 034260 004037 034104   JSR     R0,@#CK.CHR     ;CHECK ONE CHARACTER
7831 034264 034330          8$     ;ILLEGAL CHARACTER
7832 034266 034320          6$     ;CARRIAGE RETURN
7833 034270 034330          8$     ;"/"
7834 034272 034330          8$     ;".."
7835 034274 034300          4$     ;".."
7836 034276 034310          5$     ;DIGIT 0-9
7837 034300 005724          4$:   TST     (R4)+     ;"PERIOD-PERIOD"
7838 034302 105711          TSTB   (R1)           ;"CR"?
7839 034304 001405          BEQ    6$             ;YES--BRANCH
7840 034306 000410          BR     8$
7841 034310 126127 177776 000054 5$:   CMPB   -2(R1),#',     ;WAS CHARACTER BEFORE THE DIGIT A COMMA?
7842 034316 001004          BNE    8$             ;NO--EXIT
7843 034320 020316          6$:   CMP     R3,(SP)     ;INPUT TO LARGE?
7844 034322 101001          BHI    7$             ;YES -- BRANCH
7845 034324 060400          ADD     R4,R0         ;ADJUST RETURN ADDRESS
7846 034326 005720          7$:   TST     (R0)+     ;NUMBER TO R2
7847 034330 010302          8$:   MOV     R3,R2     ;CLEAN MAX. SIZE OFF OF STACK
7848 034332 005726          TST     (SP)+         ;RESTORE R3
7849 034334 012603          MOV     (SP)+,R3     ;RESTORE R4
7850 034336 012604          MOV     (SP)+,R4     ;GET RETURN ADDRESS
7851 034340 011000          MOV     (R0),R0     ;RETURN
7852 034342 000200          RTS     R0
7853
7854 ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
7855 ;AND FORMS AN OCTAL NUMBER IN R2
7856 ;CALL:
7857 ;      MOV     #ADR,R1          ;ADDRESS OF ASCIZ STRING
  
```



```

7858      :      JSR      RC,@#CK.NUM      ;GO FORM THE NUMBER
7859      :      RETURN   ADR1              ;ILLEGAL CHARACTER IN THE INPUT STRING
7860      :      RETURN   ADR2              ;"COMMA" OR "CR"--R2=NUMBER
7861      :      RETURN   ADR3              ;"PERIOD"--R2=NUMBER
7862      :      RETURN   ADR4              ;"PERIOD-PERIOD"--R2=NUMBER
7863
7864      034344 010346      (K.NUM: MOV    R3,-(SP)      ;SAVE R3
7865      034346 005003      CLR     R3              ;START NUMBER AT ZERO
7866      034350 004037      JSR     RO,@#CK.OCT    ;OCTAL DIGIT?
7867      034354 000440      BR     6$              ;NO--BRANCH
7868      034356 005201      1$:   INC     R1              ;MOVE TO NEXT CHARACTER
7869      034360 006303      ASL    R3              ;FOR THE OCTAL NUMBER IN R3
7870      034362 103435      BCS    6$              ;DON'T LET IT GET TO BIG
7871      034364 006303      ASL    R3
7872      034366 103433      BCS    6$
7873      034370 006303      ASL    R3
7874      034372 103431      BCS    6$
7875      034374 060203      ADD    R2,R3
7876      034376 004037      JSR     RO,@#CK.OCT    ;IS THIS AN OCTAL DIGIT?
7877      034402 000401      BR     2$              ;NO--FIND OUT WHAT IT IS
7878      034404 000764      BR     1$              ;YES--MAKE IT PART OF THE NUMBER
7879      034406 010302      2$:   MOV    R3,R2      ;SAVE THE OCTAL NUMBER
7880      034410 005003      CLR    R3              ;START WITH ZERO INDEX
7881      034412 004037      JSR     RO,@#CK.CHR    ;CHECK ONE CHARACTER
7882      034416 034456      6$    ;ILLEGAL CHARACTER
7883      034420 034446      5$    ;CARRIAGE RETURN
7884      034422 034456      6$    ;"/"
7885      034424 034446      5$    ;".."
7886      034426 034432      3$    ;"..."
7887      034430 034456      6$    ;DIGIT 0-9
7888      034432 005723      3$:   TST    (R3)+      ;"PERIOD"
7889      034434 121127      CMPB   (R1),#'.      ;"PERIOD-PERIOD"?
7890      034440 001002      BNE    5$              ;NO--BRANCH
7891      034442 005201      INC    R1              ;YES--ADVANCE THE POINTER
7892      034444 005723      4$:   TST    (R3)+      ;"PERIOD-PERIOD"
7893      034446 005723      5$:   TST    (R3)+      ;"COMMA"
7894      034450 105711      TSTB   (R1)          ;"CR"?
7895      034452 001001      BNE    6$              ;NO--BRANCH
7896      034454 06030J      ADD    R3,RO          ;YES--SAVE THE OCTAL NUMBER
7897      034456 012603      6$:   MOV    (SP)+,R3      ;RESTORE R3
7898      034460 011000      MOV    (RO),RO        ;PICKUP EXIT ADDRESS
7899      034462 000200      RTS    RO            ;RETURN

```

7900  
7901  
7902  
7903  
7904  
7905  
7906  
7907  
7908  
7909  
7910  
7911  
7912  
7913  
7914  
7915  
7916  
7917  
7918  
7919  
7920  
7921  
7922  
7923  
7924  
7925  
7926  
7927  
7928  
7929  
7930  
7931  
7932  
7933  
7934  
7935  
7936  
7937  
7938  
7939  
7940  
7941  
7942  
7943  
7944  
7945  
7946  
7947  
7948  
7949  
7950  
7951  
7952  
7953  
7954  
7955

034464 000000 000000 000000  
 034472 000000  
 034474 000  
 034475 000  
 034476 000  
 034477 000  
 034500 000  
 034501 000  
 034502 000  
 034503 000  
 034504 000  
 034505 000  
 034506 000  
 034507 000  
 034510 000  
 034511 000  
 034512 000  
 034513 000

```

:*****
.SBTTL SINGLE/DUAL PORT RH11/RP04/5/6 DRIVER (REV 1.0)
:COPYRIGHT (C) 1976
:DIGITAL EQUIPMENT CORP.
:MAYNARD, MA 01754
:AUTHOR(S): JIM LACEY/CHUCK HESS
:*****
:STORAGE FOR RPDS1, RPER1, RPER2, AND RPER3 ON AN ERROR "2"
:RPERRS = RPDS1
:RPERRS+2 = RPER1
:RPERRS+4 = RPER2
:RPERRS+6 = RPER3
RPERRS: .WORD 0,0,0,0

: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
DRVACT: .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 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
DRVSTA: .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 DRIVE TYPES (DRVTYP=8 BYTES)
:DRVTYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
:DRVTYP=1 IF DRIVE IS RP04
:DRVTYP=2 IF DRIVE IS RP05
:DRVTYP=4 IF DRIVE IS RP06
:DRVTYP -1 IF NOT RP04/5/6
  
```

7956 034514 000  
7957 034515 000  
7958 034516 000  
7959 034517 000  
7960 034520 000  
7961 034521 000  
7962 034522 000  
7963 034523 000

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

7964  
7965  
7966  
7967  
7968

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

7969 034524 000  
7970 034525 000  
7971 034526 000  
7972 034527 000  
7973 034530 000  
7974 034531 000  
7975 034532 000  
7976 034533 000

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

7977  
7978  
7979  
7980  
7981

;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

7982 034534 000  
7983 034535 000  
7984 034536 000  
7985 034537 000  
7986 034540 000  
7987 034541 000  
7988 034542 000  
7989 034543 000

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

7990  
7991  
7992  
7993  
7994

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

7995 034544 000000  
7996

TRNSWT: .WORD 0

7997  
7998  
7999  
8000  
8001  
8002

;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.

8003 034546 000000  
8004

SRCHWT: .WORD 0

8005  
8006  
8007  
8008

;RPO4/5/6 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)  
;ACTDRV=0 IF DRIVER IS INACTIVE  
;ACTDRV>0 IF DRIVER IS ACTIVE

8009 034550 000  
8010  
8011

ACTDRV: .BYTE 0

;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)

```

8012                                     ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
8013                                     ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
8014
8015 034551      000      ACTSTR: .BYTE  0
8016
8017                                     ;UNLOAD FLAG (ULDFLG=8 BYTES)
8018                                     ;ULDFLG=0 IF NO UNLOAD COMMAND
8019                                     ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
8020                                     ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
8021
8022 034552      000      ULDFLG: .BYTE  0          ;DRIVE 0
8023 034553      000      .BYTE  0          ;DRIVE 1
8024 034554      000      .BYTE  0          ;DRIVE 2
8025 034555      000      .BYTE  0          ;DRIVE 3
8026 034556      000      .BYTE  0          ;DRIVE 4
8027 034557      000      .BYTE  0          ;DRIVE 5
8028 034560      000      .BYTE  0          ;DRIVE 6
8029 034561      000      .BYTE  0          ;DRIVE 7
8030
8031                                     ;LOOK AHEAD COUNT (LACNT=8 BYTES)
8032                                     ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
8033
8034 034562      000      LACNT: .BYTE  0          ;DRIVE 0
8035 034563      000      .BYTE  0          ;DRIVE 1
8036 034564      000      .BYTE  0          ;DRIVE 2
8037 034565      000      .BYTE  0          ;DRIVE 3
8038 034566      000      .BYTE  0          ;DRIVE 4
8039 034567      000      .BYTE  0          ;DRIVE 5
8040 034570      000      .BYTE  0          ;DRIVE 6
8041 034571      000      .BYTE  0          ;DRIVE 7
8042
8043                                     ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
8044                                     ;SAVEFG <0 IF SAVE THE RH11/RP04/5/6 REGISTERS WHEN THE
8045                                     ;OPERATION IS COMPLETED AS PER (DPB+14).
8046                                     ;SAVEFG=0 IF SAVE THE RH11/RP04/5/6 REGISTERS, AS PER
8047                                     ;(DPB+14), AFTER AN ERROR.
8048
8049 034572 000000      SAVEFG: .WORD  0
8050
8051                                     ;SEEK FLAG (SEEKFG=1 WORD)
8052                                     ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
8053                                     ;FOR A DATA TRANSFER START A SEARCH COMMAND
8054                                     ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
8055                                     ;DISREGARD THE WINDOW
8056
8057 034574 000000      SEEKFG: .WORD  0
8058
8059                                     ;TIMEOUT TABLE (TIMER=8 WORDS)
8060                                     ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
8061
8062 034576 177777      TIMER:  .WORD -1          ;DRIVE 0
8063 034600 177777      .WORD -1          ;DRIVE 1
8064 034602 177777      .WORD -1          ;DRIVE 2
8065 034604 177777      .WORD -1          ;DRIVE 3
8066 034606 177777      .WORD -1          ;DRIVE 4
8067 034610 177777      .WORD -1          ;DRIVE 5

```

8068 034612 177777  
8069 034614 177777  
8070  
8071  
8072  
8073  
8074  
8075 034616 177777  
8076  
8077  
8078  
8079  
8080  
8081 034620 001  
8082 034621 002  
8083 034622 004  
8084 034623 010  
8085 034624 020  
8086 034625 040  
8087 034626 100  
8088 034627 200  
8089  
8090  
8091  
8092  
8093 034630 000003  
8094  
8095  
8096  
8097  
8098 034632 176700  
8099 034634 000254 000240  
8100  
8101  
8102  
8103 034640 000004  
8104  
8105  
8106 034642 001000  
8107  
8108  
8109 034644 000200  
8110  
8111  
8112 034646 000005  
8113  
8114  
8115  
8116 000000  
8117 000002  
8118 000004  
8119 000006  
8120 000010  
8121 000012  
8122 000014  
8123 000016

.WORD -1 ;DRIVE 6  
.WORD -1 ;DRIVE 7  
;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)  
;DTUW<0 IF NO DATA TRANSFER UNDERWAY  
;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N  
DTUW: .WORD -1  
;ATTENTION BITS TABLE (ATABIT=8 BYTES)  
;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES  
;ATTENTION BIT  
ATABIT: .BYTE 1 ;DRIVE 0  
.BYTE 2 ;DRIVE 1  
.BYTE 4 ;DRIVE 2  
.BYTE 10 ;DRIVE 3  
.BYTE 20 ;DRIVE 4  
.BYTE 40 ;DRIVE 5  
.BYTE 100 ;DRIVE 6  
.BYTE 200 ;DRIVE 7  
;RP04/5/6 TO RH11 "MASSBUS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE  
;CALLING IT FATAL (MCPEMX=1 WORD)  
MCPEMX: .WORD 3  
;STORAGE FOR RPADR (THE FIRST ADDRESS (776700) OF THE RH11/RP04/5/6),  
;RPVEC (THE VECTOR ADDRESS (254)), AND RPVEC+2 (THE BR LEVEL (5)).  
RPADR: .WORD 176700  
RPVEC: .WORD 254,5\*32.  
;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)  
MXLACT: .WORD 4  
;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)  
MXDLTA: .WORD 8.\*64.  
;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)  
MNDLTA: .WORD 2\*64.  
;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)  
MXWNDW: .WORD 5  
;DEFINITIONS OF THE RH11/RP04/5/6 ADDRESS INDEXES  
RPCS1=0 ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)  
RPWC=2 ;WORD COUNT REGISTER (NOT A DRIVE REG)  
RPBA=4 ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)  
RPDA=6 ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)  
RPCS2=10 ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)  
RPDS1=12 ;DRIVE STATUS REGISTER (DRIVE REG 01)  
RPER1=14 ;ERROR REGISTER #1 (DRIVE REG. 02)  
RPAS=16 ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)

8124	000020	RPLA=20	:LOOK AHEAD REGISTER (DRIVE REG. 07)
8125	000022	RPDB=22	:DATA BUFFER REGISTER (NOT A DRIVE REG.)
8126	000024	RPMR=24	:MAINTAINABILITY REGISTER (DRIVE REG. 03)
8127	000026	RPDT=26	:DRIVE TYPE REGISTER (DRIVE REG. 06)
8128	000030	RPSN=30	:SERIAL NUMBER REGISTER (DRIVE REG. 10)
8129	000032	RPOF=32	:OFFSET REGISTER (DRIVE REG. 11)
8130	000034	RPCA=34	:DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
8131	000036	RPCC=36	:CURRENT CYLINDER ADDRESS REGISTER (DRIVE REG. 13)
8132	000040	RPER2=40	:ERROR REGISTER #2 (DRIVE REG. 14)
8133	000042	RPER3=42	:ERROR REGISTER #3 (DRIVE REG. 15)
8134	000044	RPEC1=44	:ECC POSITION REGISTER (DRIVE REG. 16)
8135	000046	RPEC2=46	:ECC PATTERN REGISTER (DRIVE REG. 17)

```

8136
8137 ;RH11/RP04/5/6 DRIVER INITIALIZATION CODE
8138 ;THIS ROUTINE WILL DETERMINE WHICH RP04/5/6 DRIVES ARE
8139 ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
8140 ;TO THE PROPER STATE FOR EACH DRIVE.
8141 ;NOTE: THIS ROUTINE CALLS DRVINT

```

```

8142
8143 ;CALL
8144
8145 ;JSR PC,RPINIT
8146 ;RETURN
8147
8148 ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
8149

```

```

8150 034650 104412 RPINIT: SAVREG ;SAVE R0 - R5
8151 034652 013746 177776 MOV @#PS,-(SP) ;SAVE THE PRESENT PROCESSOR STATUS
8152 034656 012737 000240 177776 MOV #<5*32.>,@#PS ;CHANGE THE PRIORITY TO 5
8153 034664 004737 042664 JSR PC,CLRQUE ;CLEAR ALL REQUEST QUEUES
8154 034670 012701 034464 MOV #RPERRS,R1 ;FIRST ADDRESS TO BE CLEARED
8155 034674 012702 034574 MOV #SEEKFG,R2 ;LAST ADDRESS TO BE CLEARED
8156 034700 005021 1$: CLR (R1)+ ;CLEAR
8157 034702 020102 CMP R1,R2 ;ARE WE DONE?
8158 034704 101775 BLOS 1$ ;BRANCH IF NO
8159 034706 012702 034616 MOV #DTUW,R2 ;LAST ADDRESS
8160 034712 012721 177777 2$: MOV #-1,(R1)+ ;INITIALIZE
8161 034716 020102 CMP R1,R2 ;DONE?
8162 034720 101774 BLOS 2$ ;LOOP IF NO
8163 034722 005037 034504 CLR DRVSTA ;SET ALL DRIVES TO OFFLINE
8164 034726 005037 034506 CLR DRVSTA+2
8165 034732 005037 034510 CLR DRVSTA+4
8166 034736 005037 034512 CLR DRVSTA+6
8167 034742 013703 034634 MOV RPVEC,R3 ;SETUP THE RH11/RP04/5/6 VECTOR
8168 034746 012723 037544 MOV #ISR,(R3)+
8169 034752 013713 034636 MOV RPVEC+2,(R3)
8170 034756 013704 034632 MOV RPADR,R4 ;FIRST ADDRESS OF RH11/RP04
8171 034762 012764 000040 000010 MOV #BIT05,RPCS2(R4) ;MASSBUS INIT
8172 034770 005001 CLR R1 ;START WITH DRIVE 0
8173 034772 004037 035062 3$: JSR R0,DRVINT ;INIT THE DRIVE
8174 034776 000401 BR 4$ ;'DVA' NOT SET OR PARITY ERROR
8175 035000 000402 BR 5$ ;NORMAL RETURN
8176 035002 105061 034504 4$: CLRB DRVSTA(R1) ;SET DRIVE STATUS TO OFFLINE
8177 035006 005201 5$: INC R1 ;GO TO NEXT DRIVE
8178 035010 042701 177770 BIC #^C7,R1 ;MASK OUT UNUSED BITS
8179 035014 001366 BNE 3$ ;BR IF MORE DRIVES TO GO

```

```

8180 035016 012701 000007      MOV      #7,R1      ;START WITH DRIVE 7
8181 035022 005037 177776      CLR      @#PS      ;CLEAR THE PROCESSOR STATUS
8182 035026 105761 034524      6$: TSTB   DPINT(R1) ;WAITING FOR DRIVE TO SWITCH PORTS ?
8183 035032 001405          BEQ      8$        ;BR NOT WAITING
8184 035034 004737 042320      JSR      PC,SET.IE ;SET INTERRUPT
8185 035040 105761 034524      7$: TSTB   DPINT(R1) ;DRIVE SWITCHED PORTS ?
8186 035044 001375          BNE      7$        ;BR IF NOT
8187 035046 005301      8$: DEC     R1        ;GO TO THE NEXT DRIVE
8188 035050 100366          BPL      6$        ;CHECK NEXT DRIVE
8189 035052 012637 177776      MOV      (SP)+,@#PS ;RESTORE THE PROCESSOR STATUS
8190 035056 104413          RESREG   ;RESTORE R0 - R5
8191 035060 000207          RTS      PC        ;BYE-BYE
8192
8193      ;DRIVE INITIALIZATION ROUTINE
8194      ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
8195      ;AN RP04/5/6. IF IT IS, A 'READ-IN PRESET' IS ISSUED AND FMT22
8196      ;IS SET TO A '1'. THEN MCL, DPR, DRY, AND VV ARE CHECKED TO
8197      ;INSURE THEY ARE ALL ON A '1'. AND DEPENDING ON THEIR STATE,
8198      ;DRVSTA IS SET TO THE PROPER CONDITION.
8199
8200      ;CALL
8201      :
8202      :      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
8203      :      MOV      RPADR,R4        ;UNIBUS ADDRESS OF RH11/RP04/5/6 (RPCS1)
8204      :      JSR      R0,DRVINT      ;CALLED BY A JSR
8205      :      RETURN1      ;ERROR OCCURRED (PARITY)
8206      :      RETURN2      ;NORMAL RETURN
8207
8207 035062 010546      DRVINT: MOV     R5,-(SP)      ;SAVE R5
8208 035064 105061 034504      CLRB   DRVSTA(R1)      ;START DRIVE STATUS AS OFFLINE
8209 035070 105061 034514      CLRB   DRVSTYP(R1)     ;CLEAR THE DRIVE TYPE INDICATOR
8210 035074 105061 034552      CLRB   ULDFLG(R1)     ;CLEAR THE UNLOAD FLAG
8211 035100 010164 000010      MOV     R1,RPCS2(R4)   ;SELECT A DRIVE
8212 035104 112764 000111 000000      MOVB   #111,RPCS1(R4) ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
8213 035112 032764 010000 000010      BIT    #BIT12,RPCS2(R4) ;NONEXISTENT DRIVE?
8214 035120 001403          BEQ    1$            ;NO---BRANCH
8215 035122 004737 042320      JSR    PC,SET.IE      ;GO SET 'IE' WITHOUT A 'TRE'
8216 035126 000533          BR    6$            ;LEAVE THIS ROUTINE
8217 035130 105061 034504      1$: CLRB  DRVSTA(R1)   ;SET DRIVE STATUS TO OFFLINE
8218 035134 032764 004000 000000      BIT    #BIT11,RPCS1(R4) ;SEE IF DRIVE AVAILABLE
8219 035142 001527          BEQ    7$            ;BR IF DRIVE NOT AVAILABLE
8220 035144 004037 041630      JSR    R0,RD.RP      ;READ THE DRIVE TYPE REG.
8221 035150 000026          RPD   8$            ;ERROR RETURN ADDRESS
8222 035152 035442          MOV    (SP)+,R5      ;PUT DRIVE TYPE IN R5
8223 035154 012605          MOVB   #1,DRVSTYP(R1) ;SET RP04 INDICATOR
8224 035156 112761 000001 034514      CMP    #20020,R5      ;IS IT A SINGLE PORT RP04?
8225 035164 022705 020020          BEQ    2$            ;BRANCH IF YES
8226 035170 001431          CMP    #24020,R5     ;IS IT A DUAL PORT RP04?
8227 035172 022705 024020          BEQ    2$            ;BR IF YES
8228 035176 001426          MOVB   #2,DRVSTYP(R1) ;SET RP05 INDICATOR
8229 035200 112761 000002 034514      CMP    #20021,R5      ;SINGLE PORT RP05 ?
8230 035206 022705 020021          BEQ    2$            ;BR IF YES
8231 035212 001420          CMP    #24021,R5     ;DUAL PORT RP05 ?
8232 035214 022705 024021          BEQ    2$            ;BR IF YES
8233 035220 001415          MOVB   #4,DRVSTYP(R1) ;SET RP06 INDICATOR
8234 035222 112761 000004 034514      CMP    #20022,R5      ;SINGLE PORT RP06 ?
8235 035230 022705 020022

```

```

8236 035234 001407          BEQ      2$          ;BR IF YES
8237 035236 022705 024022  CMP      #24022,R5   ;DUAL PORT RP06 ?
8238 035242 001404          BEQ      2$          ;BR IF YES
8239 035244 112761 177777 034514  MOVB    #-1,DRVTP(R1) ;SET INDICATOR TO 'OTHER'
8240 035252 000461          BR       6$          ;EXIT
8241 035254 005737 035446      2$:  TST    TSTPGM      ;INHIBIT PROGRAMMABLE DRIVE?
8242 035260 001010          BNE     9$          ;BRANCH IF NO
8243 035262 032764 001000 000012  BIT    #BIT09, RPDS1(R4) ;IS DRIVE PROGRAMMABLE?
8244 035270 001404          BEQ     9$          ;BRANCH IF NO
8245 035272 152761 000010 034514  BISB   #BIT03, DRVTP(R1) ;SET INDICATOR
8246 035300 000446          BR       6$          ;EXIT
8247 035302 012746 000121      9$:  MOV    #121,-(SP)   ;DO A 'READ-IN PRESE''
8248 035306 004037 042010      JSR    RO,WRT.RP
8249 035312 000000          RPOF   RPC51
8250 035314 035442          8$
8251 035316 012746 010000      MOV    #BIT12,-(SP)   ;SET FMT22 1
8252 035322 004037 042010      JSR    RO,WRT.RP
8253 035326 000032          RPOF
8254 035330 035442          8$
8255 035332 004037 041630      JSR    RO,RD.RP      ;READ RPDS1
8256 035336 000012          RPDS1
8257 035340 035442          8$
8258 035342 012605          MOV    (SP)+,R5      ;AND SAVE IT IN R5
8259 035344 100015          BPL    4$          ;BRANCH IF ATA=0
8260 035346 116164 034620 000016  MOVB   ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
8261 035354 004037 041630      JSR    RO,RD.RP      ;FIND OUT WHY ATA=1
8262 035360 000014          RPER1
8263 035362 035442          8$
8264 035364 006126          ROL    (SP)+        ;IS IT UNSAFE?
8265 035366 100004          BPL    4$          ;BR IF NOT
8266 035370 112761 177777 034504  MOVB   #-1,DRVSTA(R1) ;SET UNSAFE INDICATOR
8267 035376 000407          BR     6$          ;EXIT
8268 035400 005105          4$:  COM    R5          ;CHECK MOL, DPR, DRY, AND VV
8269 035402 042705 167077      BIC    #^C<BIT12!BIT08!BIT07!BIT06>,R5
8270 035406 001003          BNE    6$          ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
8271 035410 112761 000001 034504  MOVB   #1,DRVSTA(R1) ;SET DRIVE STATUS TO ONLINE
8272 035416 005720          6$:  TST    (R0)+        ;STEP OVER THE ERROR RETURN
8273 035420 000410          BR     8$          ;EXIT
8274 035422 006301          7$:  ASL    R1          ;CHANGE INDEX TO ADDRESS WORDS
8275 035424 012761 003720 034576  MOV    #2000.,TIMER(R1) ;START 2 SEC TIMER
8276 035432 006201          ASR    R1          ;RESTORE R1
8277 035434 105161 034524      COMB   DPINT(R1)
8278 035440 005720          TST    (R0)+
8279 035442 012605          8$:  MOV    (SP)+,R5      ;RESTORE R5
8280 035444 000200          RTS    RO          ;EXIT
8281
8282          ;TEST PROGRAMMABLE DRIVE FLAG (TSTPGM=1 WORD)
8283          ;THE FLAG WILL BE SET BY THE PROGRAM UNDER
8284          ;MANUFACTURING CONDITIONS (ACT,APT) AND
8285          ;CLEARED UNDER FIELD CONDITIONS(XXDP CHAIN,
8286          ;STANDALONE) WITH STARTING ADDRESS 200.
8287          ;THE FLAG WILL BE SET UNDER ALL CONDITIONS
8288          ;WITH STARTING ADDRESS 220.
8289
8290 035446 000000      TSTPGM: .WORD 0      ; 1 TEST PROGRAMMABLE DRIVE
8291

```



```

8292          ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
8293          ;CALL
8294          ;
8295          ;
8296          ;
8297          ;
8298          ;
8299          ;
8300          ;
8301          ;
8302 035450 013746 177776 RP04: MOV @#PS,-(SP) ;SAVE THE CALLING STATUS
8303 035454 013737 034636 177776 MOV RPVEC+2,@#PS ;DON'T ALLOW ANY RP04/5/6 INTERRUPTS
8304 035462 112737 000001 034550 MOVB #1,ACTDRV ;SET "ACTIVE DRIVER" FLAG
8305 035470 104412 SAVREG ;SAVE R0 - R5
8306 035472 011002 MOV (R0),R2 ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
8307 035474 005062 000016 CLR 16(R2) ;CLEAR THE STATUS/ERROR INDICATOR
8308 035500 111201 MOVB (R2),R1 ;PICKUP THE DRIVE NUMBER
8309 035502 013704 034632 MOV RPADR,R4 ;UNIBUS ADDRESS OF RPCS1
8310 035506 105761 034504 TSTB DRVSTA(R1) ;CHECK DRIVES STATUS
8311 035512 003014 BGT 1$ ;BRANCH IF ONLINE
8312 035514 105761 034552 TSTB ULDFLG(R1) ;UNLOAD COMMAND IN QUEUE?
8313 035520 001036 BNE 3$ ;BRANCH IF YES
8314 035522 105761 034524 TSTB DPINT(R1) ;TRYING TO INIT THE DRIVE
8315 035526 001042 BNE 5$ ;BR IF YES
8316 035530 004037 035062 JSR R0,DRVINT ;GO INIT. THE DRIVE
8317 035534 000434 BR 4$ ;ERROR RETURN
8318 035536 105761 034504 TSTB DRVSTA(R1) ;IS DRIVE STATUS ONLINE?
8319 035542 003445 BLE 6$ ;BR IF NOT
8320 035544 105761 034534 1$: TSTB DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
8321 035550 001031 BNE 5$ ;BR IF YES
8322 035552 010164 000010 MOV R1,RPCS2(R4) ;SELECT THE DRIVE
8323 035556 004037 042762 JSR R0,DRVQUE ;PUT THIS REQUEST IN QUEUE
8324 035562 000460 BR 9$ ;QUEUE IS FULL
8325 035564 122762 000103 000002 CMPB #103,2(R2) ;IS THIS REQ. FOR AN UNLOAD?
8326 035572 001003 BNE 2$ ;BR IF NO
8327 035574 112761 177777 034552 MOVB #-1,ULDFLG(R1) ;SET THE "UNLOAD IN QUEUE" FLAG
8328 035602 105761 034474 2$: TSTB DRVACT(R1) ;IS THIS DRIVE ACTIVE?
8329 035606 001043 BNE 8$ ;BR IF YES
8330 035610 004737 035742 JSR PC,OPT ;CALL THE OPTIMIZER
8331 035614 000440 BR 8$
8332 035616 012762 120000 000016 3$: MOV #BIT15:BIT13,16(R2) ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
8333 035624 000434 BR 8$ ;EXIT
8334 035626 004737 037052 4$: JSR PC,C17 ;GO HANDLE THE PARITY ERROR
8335 035632 000431 BR 8$
8336 035634 004037 042762 5$: JSR R0,DRVQUE ;PUT REQUEST IN QUEUE
8337 035640 000431 BR 9$ ;QUEUE IS FULL
8338 035642 032714 000100 BIT #BIT06,(R4) ;IS 'IE' SET ALREADY ?
8339 035646 001023 BNE 8$ ;BR IF IT IS
8340 035650 004737 042320 JSR PC,SET.IE ;SET INTERRUPT
8341 035654 000420 BR 8$ ;RETURN, REQUEST IN QUEUE
8342 035656 105761 034504 6$: TSTB DRVSTA(R1) ;SEE IF DRIVE OFFLINE OR UNSAFE
8343 035662 002412 BLT 7$ ;BR IF UNSAFE
8344 035664 012762 140000 000016 MOV #BIT15:BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
8345 035672 105761 034514 TSTB DRVTP(R1) ;SEE IF OFFLINE OR NONEXISTENT
8346 035676 001007 BNE 8$ ;BR IF OFFLINE
8347 035700 012762 100002 000016 MOV #BIT15:BIT01,16(R2) ;REPORT DRIVE NONEXISTENT

```

```

8348 035706 000403          BR      8$      ;GO TO EXIT
8349 035710 012762 110000 000016 7$:  MOV     #BIT15:BIT12,16(R2) ;DRIVE IS UNSAFE
8350 035716 104413          8$:  RESREG          ;RESTORE R0 - R5
8351 035720 005720          TST     (R0)+      ;SETUP FOR NORMAL RETURN
8352 035722 000401          BR      10$      ;FINISH UP, THEN EXIT
8353 035724 104413          9$:  RESREG          ;RESTORE R0 - R5
8354 035726 005720          10$: TST     (R0)+      ;CORRECT THE RETURN ADDRESS
8355 035730 105037 034550    CLR B   ACTDRV     ;CLEAR 'ACTIVE DRIVER' FLAG
8356 035734 012637 177776    MOV     (SP)+, @#PS ;RETURN 'PS' TO USER LEVEL
8357 035740 000200          RTS     R0        ;RETURN TO CALLER
8358
8359          ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
8360          ;CALL
8361          ;
8362          MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
8363          JSR     PC,OPT          ;SETUP A COMMAND
8364          ;
8365          OPT:  SAVREG          ;SAVE R0 - R5
8366          MOV     @#PS,-(SP)      ;SAVE PROC. STATUS
8367          BIC B  ATABIT(R1),SRCHWT ;CLEAR 'SEARCH WAIT' KEY
8368          JSR     PC,GETREQ       ;GET 'DPB' POINTER OF REQUEST
8369          TST     R2             ;IS THERE A REQUEST IN QUEUE?
8370          BEQ     7$             ;NO--BRANCH TO EXIT
8371          035766 032764 004000 000000 BIT     #BIT11,RPCS1(R4) ;IS DVA STILL SET ?
8372          BEQ     10$           ;BR IF NOT
8373          035776 032764 000100 000012 BIT     #BIT6,RPDS1(R4) ;IS VV SET ?
8374          BNE     10$           ;BR IF IT IS
8375          036006 004037 035062 9$:  JSR     R0,DRVINT      ;SEE IF DRIVE STILL ONLINE ?
8376          036012 000470          BR      6$             ;PARITY OR 'DVA' NOT SET
8377          036014 105761 034504 10$: TST B   DRVSTA(R1) ;IS DRIVE ONLINE?
8378          036020 003014          BGT     1$             ;YES--BRANCH
8379          036022 004737 043060          JSR     PC,POPQUE     ;NO--REMOVE REQUEST FROM QUEUE
8380          036026 012762 140000 000016 MOV     #BIT15:BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
8381          036034 105761 034504          TST B   DRVSTA(R1) ;IS DRIVE UNSAFE ?
8382          036040 100064          BPL     8$             ;BR TO EXIT IF NOT
8383          036042 012762 110000 000016 MOV     #BIT15:BIT12,16(R2) ;SET UNSAFE STATUS/ERROR INDICATOR
8384          036050 000460          BR      8$             ;BRANCH TO EXIT
8385          036052 012746 000111 1$:  MOV     #111,-(SP) ;LOAD COMMAND ONTO THE STACK
8386          036056 004037 042010          JSR     R0,WRT.RP    ;LOAD THE REGISTER
8387          036062 000000          RPCS1          ;REGISTER INCREMENT
8388          036064 036174          6$             ;ERROR RETURN ADDRESS
8389          036066 032714 004000          BIT     #BIT11,(R4) ;DRIVE AVAILABLE ?
8390          036072 001427          BEQ     5$             ;BR IF NOT
8391          036074 122762 000150 000002 CMP B   #150,2(R2) ;IS THE REQUEST FOR I/O?
8392          036102 002403          BLT     2$             ;YES--BRANCH
8393          036104 004737 036436          JSR     PC,C14       ;CALL THE COMMAND INITIATOR
8394          036110 000440          BR      8$             ;BRANCH TO EXIT
8395          036112 005737 034616 2$:  TST     DTUW         ;DATA TRANSFER UNDERWAY?
8396          036116 002012          BGE     4$             ;YES--GO START A SEARCH
8397          036120 005737 034574          TST     SEEKFG      ;DO IMPLIED SEEKS?
8398          036124 100404          BMI     3$             ;YES---BRANCH
8399          036126 004037 037406          JSR     R0,LA        ;NO--DO LOOK AHEAD
8400          036132 000427          BR      8$             ;RETURN HERE ON A PARITY ERROR
8401          036134 000403          BR      4$             ;GO START A SEARCH
8402          036136 004737 036222 3$:  JSR     PC,C11       ;START A DATA TRANSFER
8403          036142 000423          BR      8$

```

```

8404 036144 004737 036330      4$: JSR    PC,C13      ;START A SEARCH
8405 036150 000420              BR     8$           ;GO TO THE EXIT
8406 036152 112761 177777 034534 5$: MOVB   #-1,DPROS(R1) ;SET PORT REQUEST INDICATOR
8407 036160 010103              MOV    R1,R3       ;SET UP TO ADDRESS WORDS
8408 036162 006303              ASL    R3           ;CONVERT TO WORD INDEX
8409 036164 012763 023420 034576 MOV    #10000.,TIMER(R3) ;START 10 SEC TIMER
8410 036172 000402              BR     7$           ;EXIT
8411 036174 004737 037052      6$: JSR    PC,C17      ;PROCESS THE PARITY ERROR
8412 036200 032714 000100      7$: BIT    #BIT06,(R4) ;SEE IF 'IE' ALREADY SET
8413 036204 001002              BNE   8$           ;BR IF SET
8414 036206 004737 042320      JSR    PC,SET.IE   ;SET 'IE' WITHOUT A 'TRE'
8415 036212 012637 177776      8$: MOV    (SP)+,2#PS ;RESTORE PROC. STATUS
8416 036216 104413              RESREG ;RESTORE R0 - R5
8417 036220 000207              RTS    PC
8418
8419              ;COMMAND INITIATOR
8420
8421              ;CALL
8422              ;
8423              ;CALL      MOV    #DRVNUM,R1      ;DRIVE NUMBER
8424              ;          MOV    #DPB,R2        ;ADDRESS OF DPB
8425              ;          JSR    PC,C1?        ;C1?= C11,C13, OR C14
8426              ;          ;WHERE:
8427              ;          ;C11=DATA TRANSFER
8428              ;          ;C12=SEARCH REQUESTED BY DATA XFER
8429              ;          ;C14=NOT DATA TRANSFER
8430 036222 004737 043060      C11: JSR    PC,POPQUE  ;REMOVE REQUEST FROM 'DRIVES WAIT' QUEUE
8431 036226 010237 034544      MOV    R2,TRNSWT  ;PUT REQ. IN TRANSFER WAIT QUEUE
8432 036232 010203              MOV    R2,R3      ;DPB ADDRESS TO R3
8433 036234 013704 034632      MOV    RPADR,R4   ;RPCS1 ADDRESS
8434 036240 010164 000010      MOV    R1,RPCS2(R4) ;SELECT DRIVE
8435 036244 062703 000004      ADD    #4,R3      ;DESIRED WORD COUNT
8436 036250 062704 000002      ADD    #2,R4      ;RPWC ADDRESS
8437 036254 012324              MOV    (R3)+,(R4)+ ;LOAD WORD COUNT
8438 036256 012324              MOV    (R3)+,(R4)+ ;LOAD BUFFER ADDRESS
8439 036260 012346              MOV    (R3)+,-(SP) ;LOAD SECTOR AND TRACK
8440 036262 004037 042010      JSR    R0,WRT.RP  ;CALL THE LOAD(WRITE) ROUTINE
8441 036266 000006              RPDA   ;INDEX OF REGISTER TO LOAD
8442 036270 037052              C17   ;ERROR RETURN ADDRESS
8443 036272 012346              MOV    (R3)+,-(SP) ;LOAD CYLINDER ADDRESS
8444 036274 004037 042010      JSR    R0,WRT.RP
8445 036300 000034              RPCA   ;
8446 036302 037052              C17   ;
8447 036304 016246 000002      MOV    2(R2),-(SP) ;LOAD 'COMMAND+GO', 'A17&A16', AND 'PSEL'
8448 036310 004037 042010      JSR    R0,WRT.RP
8449 036314 000000              RPCS1 ;
8450 036316 037052              C17   ;
8451 036320 010137 034616      MOV    R1,DTUW    ;SET 'DATA TRANSFER UNDERWAY'
8452 036324 000137 037014      JMP    C15        ;
8453 036330 013704 034632      C13: MOV    RPADR,R4   ;RPCS1 ADDRESS
8454 036334 010164 000010      MOV    R1,RPCS2(R4) ;SELECT DRIVE
8455 036340 016246 000012      MOV    12(R2),-(SP) ;DESIRED CYLINDER ADDRESS
8456 036344 004037 042010      JSR    R0,WRT.RP
8457 036350 000034              RPCA   ;
8458 036352 037052              C17   ;
8459 036354 116203 000010      MOVB   10(R2),R3  ;PICKUP SECTOR ADDRESS

```

8460	036360	163703	034646		SUB	MXLNDW,R3	:BACKUP BY MAX. SEARCH FOR I/O WINDOW
8461	036364	002002			BGE	1\$	
8462	036366	062703	000026		ADD	#22.,R3	
8463	036372	010346		1\$:	MOV	R3,-(SP)	:COMBINE THE ADJUSTED SECTOR WITH
8464	036374	116266	000011	000001	MOVB	11(R2),1(SP)	:THE DESIRED TRACK
8465	036402	004037	042010		JSR	RO,WRT.RP	:LOAD DESIRED TRACK & SECTOR
8466	036406	000006			RPDA		
8467	036410	037052			C17		
8468	036412	012746	000131		MOV	#131,-(SP)	:START A SEARCH
8469	036416	004037	042010		JSR	RO,WRT.RP	
8470	036422	000000			RPCS1		
8471	036424	037052			C17		
8472	036426	156137	034620	034546	BISB	ATABIT(R1),SRCHWT	:SET "SEARCH WAIT" KEY
8473	036434	000567			BR	C15	
8474	036436	013704	034632	C14:	MOV	RPADR,R4	:RPCS1 ADDRESS
8475	036442	010164	000010		MOV	R1,RPCS2(R4)	:SELECT DRIVE
8476	036446	116203	000002		MOVB	2(R2),R3	:PICKUP THE REQUESTED COMMAND
8477	036452	122703	000131		CMPB	#131,R3	:IS IT A SEARCH COMMAND?
8478	036456	001007			BNE	1\$	:BRANCH IF NO
8479	036460	016246	000010		MOV	10(R2),-(SP)	:LOAD DESIRED TRACK & SECTOR
8480	036464	004037	042010		JSR	RO,WRT.RP	
8481	036470	000006			RPDA		
8482	036472	037052			C17		
8483	036474	000403			BR	2\$	:GO LOAD CYLINDER
8484	036476	122703	000105	1\$:	CMPB	#105,R3	:IS IT A SEEK COMMAND
8485	036502	001007			BNE	3\$	:BRANCH IF NO
8486	036504	016246	000012	2\$:	MOV	12(R2),-(SP)	:LOAD DESIRED CYLINDER
8487	036510	004037	042010		JSR	RO,WRT.RP	
8488	036514	000034			RPCA		
8489	036516	037052			C17		
8490	036520	000546			BR	C16	
8491	036522	122703	000115	3\$:	CMPB	#115,R3	:IS IT AN "OFFSET" COMMAND?
8492	036526	001013			BNE	4\$	:BR IF NO
8493	036530	004037	041630		JSR	RO,RD.RP	:MERGE THE OFFSET VALUE INTO RPOF
8494	036534	000032			RPOF		:BUT DON'T CHANGE THE UPPER
8495	036536	037052			C17		
8496	036540	116216	000001		MOVB	1(R2),(SP)	:BYTE WHEN LOADING THE
8497	036544	004037	042010		JSR	RO,WRT.RP	:REGISTER (RPOF)
8498	036550	000032			RPOF		
8499	036552	037052			C17		
8500	036554	000530			BR	C16	:GO START THE COMMAND
8501	036556	122703	000107	4\$:	CMPB	#107,R3	:IS IT A "RECALIBRATE" COMMAND?
8502	036562	001525			BEQ	C16	:BRANCH IF YES
8503	036564	122703	000117		CMPB	#117,R3	:IS IT A RETURN TO CENTER?
8504	036570	001522			BEQ	C16	:BRANCH IF YES
8505	036572	122703	000103		CMPB	#103,R3	:IS IT AN "UNLOAD" COMMAND?
8506	036576	001016			BNE	5\$	:BRANCH IF NO
8507	036600	112761	000001	034474	MOVB	#1,DRVACT(R1)	:SET THE DRIVE ACTIVE INDICATOR
8508	036606	105061	034504		CLRB	DRVSTA(R1)	:PUT DRIVE STATUS TO OFFLINE
8509	036612	112761	000001	034552	MOVB	#1,ULDFLG(R1)	:SET "UNLOAD IN PROGRESS" FLAG
8510	036620	010346			MOV	R3,-(SP)	:START THE "UNLOAD" COMMAND
8511	036622	004037	042010		JSR	RO,WRT.RP	
8512	036626	000000			RPCS1		
8513	036630	037052			C17		
8514	036632	000207			RTS	PC	:RETURN TO USER
8515	036634	122703	000143	5\$:	CMPB	#143,R3	:IS IT A "SET FORMAT" COMMAND?

8516	036640	001014			BNE	6\$		:BRANCH IF NO
8517	036642	004037	041630		JSR	RO, RD.RP		:READ THE OFFSET REGISTER
8518	036646	000032			RPOF			
8519	036650	037052			C17			
8520	036652	116266	000001	000001	MOV B	1(R2), 1(SP)		:COMBINE "FMT22", "ECI", AND "HCI"
8521	036660	004037	042010		JSR	RO, WRT.RP		:LOAD "FMT22", "ECI", AND/OR "HCI".
8522	036664	000032			RPOF			
8523	036666	037052			C17			
8524	036670	000436			BR	12\$		
8525	036672	122703	000141		6\$: CMPB	#141, R3		:IS IT A "GET REGISTER" COMMAND?
8526	036676	001023			BNE	10\$		:BRANCH IF NO
8527	036700	016203	000006		7\$: MOV	6(R2), R3		:POINTS TO 1ST ADDRESS OF WHERE :TO PUT THE REGISTER(S)
8528								
8529	036704	116237	000010	036722	MV B	10(R2), 9\$		:INIT. THE INDEX FOR THE FIRST REG.
8530	036712	116205	000011		MOV B	11(R2), R5		:INDEX OF LAST REG. TO MOVE
8531	036716	004037	041630		8\$: JSR	RO, RD.RP		:READ RPO4/5/6 REGISTER
8532	036722	000000			9\$: RPCS1			:INDEX OF REG. TO READ
8533	036724	037052			C17			
8534	036726	012623			MOV	(SP)+, (R3)+		:GET THE CONTENTS OF RH11/RPO4/5/6 REG.
8535	036730	023705	036722		CMP	9\$, R5		:LAST REG. BEEN READ?
8536	036734	001414			BEQ	12\$		:GET OUT IF YES
8537	036736	062737	000002	036722	ADD	#2, 9\$		:INCREASE THE INDEX BY 2
8538	036744	000764			BR	8\$		:LOOP--MORE TO READ
8539	036746	122703	000145		10\$: CMPB	#145, P3		:IS IT A "SELECT DRIVE" COMMAND?
8540	036752	001405			BEQ	12\$		:BRANCH IF YES
8541	036754	010346			11\$: MOV	R3, -(SP)		:LOAD THE COMMAND
8542	036756	004037	042010		JSR	RO, WRT.RP		
8543	036762	000000			RPCS1			
8544	036764	037052			C17			
8545	036766	004737	043060		12\$: JSR	PC, POPQUE		:REMOVE REQ. FROM QUEUE
8546	036772	052762	000200	000016	BIS	#BIT07, 16(R2)		:SET THE "DONE" BIT
8547	037000	005737	034572		TST	SAVEFG		:SAVE THE RH11/RPO4/5/6 REGISTERS?
8548	037004	10 02			BPL	13\$		:BRANCH IF NO
8549	037006	004737	042202		JSR	PC, SVRH11		:YES--GO SAVE THE REGISTERS
8550	037012	000207			13\$: RTS	PC		:RETURN TO USER
8551	037014	006301			C15: ASL	R1		
8552	037016	012761	001750	034576	MOV	#1000., TIMER(R1)		:SET A ONE SECOND TIMER
8553	037024	006201			ASR	R1		
8554	037026	112761	000001	034474	MOV B	#1, DRVACT(R1)		:SET THE DRIVE ACTIVE
8555	037034	000207			RTS	PC		:RETURN TO THE USER
8556	037036	010346			C16: MOV	R3, -(SP)		:LOAD THE COMMAND
8557	037040	004037	042010		JSR	RO, WRT.RP		
8558	037044	000000			RPCS1			
8559	037046	037052			C17			
8560	037050	000761			BR	C15		
8561	037052	032764	010000	000010	C17: BIT	#BIT12, RPCS2(R4)		:DRIVE NON-EXISTENT ?
8562	037060	001034			BNE	C18		:BR IF YES
8563	037062	005702			1\$: TST	R2		:ANYTHING IN QUEUE ?
8564	037064	001405			BEQ	C17B		:BR IF NOT
8565	037066	012762	104000	000016	MOV	#BIT15:BIT11, 16(R2)		:SET "PARITY" ERROR INDICATOR
8566	037074	004737	042202		JSR	PC, SVRH11		:GO SAVE THE RH11/RPO4/5/6 REGISTERS
8567	037100	012746	000111		C17B: MOV	#111, -(SP)		:DO A "DRIVE CLEAR"
8568	037104	004037	042010		JSR	RO, WRT.RP		
8569	037110	000000			RPCS1			
8570	037112	037152			C18			
8571	037114	004737	042742		JSR	PC, EMPTYQ		:EMPTY THE QUEUE



```

8628      :      JSR      RO,LA      ;GO CHECK THE WINDOW
8629      :      RETURN1      ;ERROR RETURN
8630      :      RETURN2      ;START A SEARCH
8631      :      RETURN3      ;START A DATA TRANSFER
8632
8633 037406 013704 034632 LA:  MOV      RPADR,R4      ;GET RPCS1'S ADDRESS
8634 037412 010164 000010      MOV      R1,RPCS2(R4)    ;SELECT DRIVE
8635 037416 004037 041630      JSR      RO,RD.RP      ;READ CURRENT CYLINDER
8636 037422 000036      RPCC
8637 037424 037536      4$
8638 037426 022662 000012      CMP      (SP)+,12(R2)    ;ERROR RETURN ADDRESS
8639      ;IS CURRENT CYLINDER-DESIRED
8640 037432 001037      BNE      3$            ;CYLINDER?
8641 037434 105261 034562      INCB     LACNT(R1)      ;EXIT IF NO
8642 037440 126137 034562 034640      CMPB    LACNT(R1),MXLACT ;INCREMENT THE LOOK AHEAD COUNT
8643 037446 003026      BGT      2$            ;EXCEED MAX?
8644 037450 116203 000010      MOVB    10(R2),R3      ;BRANCH IF YES
8645 037454 000303      SWAB    R3            ;GET DESIRED SECTOR ADDRESS AND
8646 037456 006203      ASR     R3            ;MULT. BY 64--ALIGN WITH
8647 037460 006203      ASR     R3            ;LOOK AHEAD REGISTER
8648 037462 012737 000340 177776      MOV     #340,@#PS      ;PRIORITY LEVEL "7"
8649 037470 004037 041630      JSR     RO,RD.RP      ;READ LOOK AHEAD REGISTER
8650 037474 000020      RPLA
8651 037476 037536      4$
8652 037500 162603      SUB     (SP)+,R3      ;CALCULATE THE DELTA
8653 037502 002002      BGE     1$            ;MAKE THE DELTA POSITIVE
8654 037504 062703 002600      ADD     #<22.*64.>,R3 ;CHECK THE DELTA TO SEE
8655 037510 023703 034642 1$:  CMP     MXDLTA,R3      ;IF IT IS WITHIN THE
8656 037514 002406      BLT     3$            ;WINDOW---IF YES, ZERO
8657 037516 023703 034644      CMP     MNDLTA,R3     ;THE LOOK AHEAD COUNT
8658 037522 002003      BGE     3$            ;AND TAKE THE I/O EXIT
8659 037524 105061 034562 2$:  CLRB   LACNT(R1)
8660 037530 005720      TST    (RO)+
8661 037532 005720      TST    (RO)+
8662 037534 000402      BR     5$            ;ADJUST THE RETURN ADDRESS
8663 037536 004737 037052 4$:  JSR    PC,C17
8664 037542 000200      5$:  RTS    RO            ;EXIT
8665      ;PROCESS THE ERROR
8666      ;RETURN
8667      ;INTERRUPT SERVICE ROUTINE
8668 037544 112737 000001 034550 ISR:  MOVB   #1,ACTDRV      ;SET "ACTIVE DRIVER" FLAG
8669 037552 104412      SAVREG      ;SAVE R0 - R5
8670 037554 013704 034632      MOV     RPADR,R4      ;ADDRESS OF RHSCS1
8671 037560 013701 034616      MOV     DTUW,R1       ;GET "DATA TRANSFER UNDERWAY" INDICATOR
8672 037564 002403      BLT     1$            ;BRANCH IF NO DATA TRANSFER UNDERWAY
8673 037566 004737 037610      JSR     PC,TD         ;CALL TRANSFER DONE
8674 037572 000402      BR     2$            ;EXIT
8675 037574 004737 037750 1$:  JSR     PC,SC         ;CALL SPECIAL CONDITIONS
8676 037600 104413      2$:  RESREG      ;RESTORE R0 - R5
8677 037602 105037 034550      CLRB   ACTDRV       ;CLEAR "ACTIVE DRIVER" FLAG
8678 037606 000002      RTI
8679      ;RETURN
8680      ;TRANSFER DONE ROUTINE
8681
8682 037610 105061 034474      TD:  CLRB   DRVACT(R1) ;SET DRIVE ACTIVE INDICATOR TO IDLE
8683 037614 012737 177777 034616      MOV     #-1,DTUW      ;NO DATA TRANSFERS UNDERWAY

```

```

8684 037622 006301 ASL R1
8685 037624 01276i 177777 034576 MOV #-1,TIMER(R1) ;CANCEL T:MEOUT
8686 037632 006201 ASR R1
8687 037634 013702 034544 MOV TRNSWT,R2 ;GET "DPB" ADDRESS FROM THE
8688 037640 005037 034544 CLR TRNSWT ;TRANSFER WAIT QUEUF--CLEAR QUEUE
8689 037644 052762 000200 000016 BIS #BIT07,16(R2) ;SET DONE
8690 037652 010164 000010 MOV R1,RPCS2(R4) ;SELECT THE DRIVE
8691 037656 004037 041630 JSR R0,RD.RP ;TRANSFER ERROR(TRE 1)?
8692 037662 000000 RPCS1
8693 037664 037052 C17
8694 037666 006126 ROL (SP)+
8695 037670 100413 BMI 3$ ;BR IF YES
8696 037672 005737 034572 TST SAVEFG ;SAVE THE RM11/RP04/5/6 REGISTERS?
8697 037676 100002 BPL 1$ ;BRANCH IF NO
8698 037700 004737 042202 JSR PC,SVRH11 ;YES--SAVE THE REGISTERS
8699 037704 004737 035742 1$: JSR PC,OPT ;CALL OPTIMIZER
8700 037710 000417 BR SC ;CHECK OTHER DRIVES
8701 037712 012714 000113 2$: MOV #113,(R4) ;RELEASE THE DRIVE
8702 037716 000414 BR SC ;CHECK FOR OTHER DRIVES
8703 037720 052762 100100 000016 3$: BIS #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
8704 037726 004737 042742 JSR PC,EMPTYQ ;EMPTY THE "DRIVE'S WAIT" QUEUE
8705 037732 004737 042202 JSR PC,SVRH11 ;SAVE THE RM11/RP04/5/6 REGISTERS
8706 037736 012714 040111 MOV #40111,(R4) ;ISSUE A "DRIVE CLEAR"
8707 037742 012714 000113 MOV #113,(R4) ;ISSUE A RELEASE TO THE DRIVE
8708 037746 000400 BR SC ;CHECK FOR OTHER DRIVES
8709
8710 ;SPECIAL CONDITION ROUTINE
8711
8712 037750 116403 000016 SC: MOVB RPAS(R4),R3 ;READ "RPAS"
8713 037754 001014 BNE 2$ ;BRANCH IF ANY 'ATA' BITS SET
8714 037756 004037 041630 JSR R0,RD.RP ;READ CONTROL AND STATUS REGISTER
8715 037762 000000 RPCS1
8716 037764 037152 C18
8717 037766 106126 ROLB (SP)+ ;IS "IE"=1?
8718 037770 100405 BMI 1$ ;YES, NO DRIVES TO CHECK
8719 037772 004037 043122 JSR R0,ES.SAV ;SAVE THE ADDRESS IN '$ESCAPE'
8720 037776 104001 ERROR 1 ;REPORT AN ILLEGAL INTERRUPT
8721 040000 004737 042320 JSR PC,SET.IE ;SET INTERRUPT ENABLE
8722 040004 000207 1$: RTS PC ;RETURN
8723 040006 005046 2$: CLR -(SP) ;PROCESS ALL DRIVES THAT HAVE
8724 040010 110316 MOVB R3,(SP) ;AN "ATA"=1
8725 040012 012703 000001 MOV #1,R3
8726 040016 005001 CLR R1
8727 040020 030316 SC3: BIT R3,(SP) ;ATA=1?
8728 040022 001005 BNE SC5 ;YES--BRANCH
6729 040024 005201 SC4: INC R1 ;MOVE TO THE NEXT DRIVE
8730 040026 106303 ASLB R5
8731 040030 001373 BNE SC3 ;BRANCH IF MORE TO CHECK?
8732 040032 005726 TST (SP)+ ;CLEAN OFF THE STACK
8733 040034 000207 RTS PC ;RETURN TO USER
8734 040036 105761 034524 SC5: TSTB DPINT(R1) ;INITIALIZING THE DRIVE ?
8735 040042 001402 BEQ 1$ ;BR IF NOT
8736 040044 000137 040742 JMP SC13 ;PROCESS THE DRIVE
8737 040050 105761 034534 1$: TSTB DPRQS(R1) ;PORT REQUEST OUTSTANDING ?
8738 040054 001402 BEQ 2$ ;BR IF NOT
8739 040056 000137 040742 JMP SC13 ;START THE OUTSTANDING COMMAND

```



8740	040062	105761	034504	2\$:	TSTB	DRVSTA(R1)	:CHECK THE DRIVE STATUS
8741	040066	003025			BGT	5\$	:BRANCH IF ONLINE
8742	040070	105761	034552		TSTB	ULDFLG(R1)	:UNLOAD IN PROGRESS?
8743	040074	003422			BLE	5\$	:BRANCH IF NOT
8744	040076	004737	043036		JSR	PC,GETREQ	:GET DPB POINTER
8745	040102	004737	042202		JSR	PC,SVRH11	:SAVE THE RH11/RPO4/5/6 REGISTERS
8746	040106	004737	040672		JSR	PC,SC12	:SAVE RPDS1, RPER1, RPER2, AND RPER3
8747							:ALSO DO A DRIVE INIT (DRVINT)
8748	040112	105761	034504		TSTB	DRVSTA(R1)	:DID DRIVE COME ONLINE?
8749	040116	003416			BLE	6\$	:NO---BRANCH
8750	040120	032737	040000	034464	BIT	#BIT14,RPERRS	:WAS THERE AN ERROR?
8751	040126	001002			BNE	3\$	:BR IF ERROR
8752	040130	000137	040602		JMP	SC11	:NO ERROR
8753	040134	013705	034466	3\$:	MOV	RPERRS+2,R5	:YES -- PICKUP RPER1 AND
8754	040140	000502			BR	SC6A	:GO PROCESS THE ERROR
8755	040142	105761	034474	5\$:	TSTB	DRVACT(R1)	:DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
8756	040146	001033			BNE	SC6	:BR IF EITHER
8757	040150	004737	040672		JSR	PC,SC12	:SAVE RPDS1, RPER1, RPER2, AND RPER3
8758							:ALSO DO A DRVINT
8759	040154	105761	034524	6\$:	TSTB	DPINT(R1)	:TRYING TO INIT THE DRIVE ?
8760	040160	001321			BNE	SC4	:BR IF YES, CHECK ON MORE DRIVES
8761	040162	105761	034504		TSTB	DRVSTA(R1)	:CHECK ON DRIVE'S STATUS
8762	040166	100412			BMI	7\$	:BR IF UNSAFE
8763	040170	032737	020000	034472	BIT	#BIT13,RPERRS+6	:ADDRESS PLUG CHANGED ?
8764	040176	001013			BNE	8\$	:BR IF YES
8765	040200	012746	000113		MOV	#113,-(SP)	:RELEASE COMMAND
8766	040204	004037	042010		JSR	RO,WRT.RP	:WRITE THE COMMAND INTO RPCS1
8767	040210	000000			RPCS1		:REGISTER INDFX
8768	040212	040552			SC8		:PARITY EXIT ADDRESS
8769	040214	011605		7\$:	MOV	(SP),R5	:PICKUP (RPAS) BEFORE THE ERROR CALL
8770	040216	004037	043122		JSR	RO,ES.SAV	:SAVE THE ADDRESS IN '\$ESCAPE'
8771	040222	104002			ERROR	2	:REPORT THE UNEXPECTED ATTENTION
8772	040224	000677			BR	SC4	:GO CHECK FOR MORE ATA'S
8773	040226			8\$:			
8774	040226	004037	043122		JSR	RO,ES.SAV	:SAVE THE ADDRESS IN '\$ESCAPE'
8775	040232	104005			ERROR	5	:REPORT THE ADDRESS PLUG CHANGE
8776	040234	000673			BR	SC4	:CHECK FOR MORE DRIVES
8777	040236	006301		SC6:	ASL	R1	:SETUP TO ADDRESS WORDS
8778	040240	012761	177777	034576	MOV	#-1,TIMER(R1)	:STOP THE TIMER
8779	040246	006201			ASR	R1	:RESTORE THE DRIVE ADDRESS
8780	040250	004737	043036		JSR	PC,GETREQ	:GET THE DPB POINTER FROM THE QUEUE
8781	040254	010164	000010		MOV	R1,RPCS2(R4)	:SELECT DRIVE
8782	040260	004037	041630		JSR	RO,RD.RP	:READ THE RPO4'S STATUS REG.
8783	040264	000012			RPCS1		
8784	040266	040552			SC8		
8785	040270	011605			MOV	(SP),R5	:AND PUT IT IN R5
8786	040272	006126			ROL	(SP)+	:WAS THERE AN ERROR?
8787	040274	100407			BMI	1\$	:BR IF ERROR
8788	040276	105761	034474		TSTB	DRVACT(R1)	:CHECK DRIVE'S STATE
8789	040302	003137			BGT	SC11	:BR IF DRIVE ACTIVE WITH ORDER
8790	040304	052762	100210	000016	BIS	#BIT15:BIT07.BIT03,16(R2)	:INFORM USER OF ERROR RECOVER COMPLETION
8791	040312	000470			BR	SC7	
8792	040314	004037	041630	1\$:	JSR	RO,FD.RP	:READ ERROR REGISTER #1
8793	040320	000014			RPER1		
8794	040322	040552			SC8		
8795	040324	012605			MOV	(SP)+,R5	:AND SAVE IT IN R5

```

8796 040326 004737 042202 JSR PC,SVRH11 ;SAVE RH11/RP04/5/6 REGISTERS
8797 040332 012746 000111 MOV #111,-(SP) ;ISSUE A DRIVE CLEAR
8798 040336 004037 042010 JSR RO,WRT.RP
8799 040342 000000 RPCS1
8800 040344 040552 SC8
8801 040346 006105 SC6A: ROL R5 ;WAS 'UNSAFE' CONDITION =1?
8802 040350 100406 BMI 1$ ;BRANCH IF YES
8803 040352 005702 TST R2 ;ANYTHING IN QUEUE ?
8804 040354 001447 BEQ SC7 ;BR IF NOT
8805 040356 052762 100240 000016 BIS #BIT15!BIT07!BIT05,16(R2) ;INFORM USER OF ERROR
8806 040364 000443 BR SC7
8807 040366 004037 041630 1$: JSR RO,RD.RP ;READ DRIVE STATUS REG. #1
8808 040372 000012 RPDS1
8809 040374 040552 SC8
8810 040376 011605 MOV (SP),R5 ;SAVE RPDS1 IN R5
8811 040400 006126 ROL (SP)+ ;'ERR'=1?
8812 040402 100011 BPL 2$ ;BR IF NO--UNSAFE CLEARED
8813 040404 112761 177777 034504 MOVB #-1,DRVSTA(R1) ;DRIVE IS UNSAFE
8814 040412 004737 042202 JSR PC,SVRH11 ;SAVE RH11/RP04/5/6 REGISTERS
8815 040416 052762 110000 000016 BIS #BIT15!BIT12,16(R2) ;INFORM USER OF UNSAFE ERROR
8816 040424 000423 BR SC7
8817 040426 032705 010000 2$: BIT #BIT12,R5 ;'MOL' = 1 ?
8818 040432 001015 BNE 3$ ;BR IF YES
8819 040434 112761 177777 034474 MOVB #-1,DRVACT(R1) ;ACTIVE ERROR RECOVER
8820 040442 112761 000001 034504 MOVB #1,DRVSTA(R1) ;ONLINE
8821 040450 006301 R1
8822 040452 012761 072460 034576 MOV #30000.,TIMER(R1) ;START 30 SECOND TIMER
8823 040460 006201 ASR R1
8824 040462 000137 040024 JMP SC4
8825 040466 052762 100220 000016 3$: BIS #BIT15!BIT07!BIT04,16(R2) ;INFORM USER OF ERROR
8826 040474 105061 034474 SC7: CLRB DRVACT(R1) ;DRIVE IS IDLE
8827 040500 004737 042742 JSR PC,EMPTYQ ;DUMP THE QUEUE
8828 040504 105761 034552 TSTB ULDFLG(R1) ;UNLOAD IN PROGRESS OR QUEUE?
8829 040510 003002 BGT 1$ ;BR IF NOT
8830 040512 105061 034552 CLRB ULDFLG(R1) ;CLEAR UNLOAD FLAG
8831 040516 116164 034620 000016 1$: MOVB ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
8832 040524 105761 034504 TSTB DRVSTA(R1) ;IS THE DRIVE UNSAFE ?
8833 040530 100406 BMI 2$ ;BR IF IT IS
8834 040532 012746 000113 MOV #113,-(SP) ;RELEASE COMMAND
8835 040536 004037 042010 JSR RO,WRT.RP ;WRITE THE COMMAND INTO RPCS1
8836 040542 000000 RPCS1 ;REGISTER INDEX
8837 040544 040552 SC8 ;PARITY EXIT ADDRESS
8838 040546 000137 040024 2$: JMP SC4 ;CHECK FOR MORE DRIVES
8839 040552 105761 034474 SC8: TSTB DRVACT(R1) ;IS DRIVE IDLE?
8840 040556 001405 BEQ 1$ ;YES--BRANCH
8841 040560 004737 043036 JSR PC,GETREQ ;GET DPB POINTER
8842 040564 004737 037052 JSR PC,C17 ;PROCESS THE PARITY ERROR
8843 040570 000402 BR 2$ ;CONTINUE
8844 040572 004737 037100 1$: JSR PC,C17B ;PROCESS THE UNCORRECTABLE PARITY ERROR
8845 040576 000137 040024 2$: JMP SC4 ;CHECK MORE DRIVES
8846 040602 105761 034552 SC11: TSTB ULDFLG(R1) ;'UNLOAD IN PROGRESS'?
8847 040606 003402 BLE 1$ ;BRANCH IF NO
8848 040610 105061 034552 CLRB ULDFLG(R1) ;CLEAR UNLOAD FLAG
8849 040614 105061 034474 1$: CLRB DRVACT(R1) ;SET DRIVE IDLE
8850 040620 136137 034620 034546 BITB ATABIT(R1),SRCHWT ;DOING A SEARCH OPERATION FOR
8851 ;AN I/O COMMAND?

```

```

8852 040626 001012          BNE      2$          ;BRANCH IF YES
8853 040630 004737 043060    JSR      PC,POPQUE  ;REMOVE REQUEST FROM QUEUE
8854 040634 052762 000200 000016    BIS      #BIT07,16(R2) ;SET "DONE" BIT
8855 040642 005737 034572    TST      SAVEFG     ;SAVE THE REGISTERS?
8856 040646 100002          BPL      2$          ;BRANCH IF NO
8857 040650 004737 042202    JSR      PC,SVRH11  ;YES--SAVE ALL OF THE RH11/RP04/5/6 REG'S
8858 040654 116164 034620 000016 2$:    MOVB    ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
8859 040662 004737 035742    JSR      PC,OPT     ;START A REQUEST
8860 040666 000137 040024    JMP      SC4        ;CHECK FOR MORE DRIVES
8861 040672 010164 000010          SC12:   MOV     R1,RPCS2(R4)    ;SELECT DRIVE
8862 040676 016437 000012 034464    MOV     RPDS1(R4),RPERRS ;SAVE THE FOUR REGISTERS THAT
8863 040704 016437 000014 034466    MOV     RPER1(R4),RPERRS+2 ;WILL TELL US SOMETHING
8864 040712 016437 000040 034470    MOV     RPER2(R4),RPERRS+4
8865
8866 040720 016437 000042 034472    MOV     RPER3(R4),RPERRS+6
8867 040726 004037 035062    JSR      R0,DRVINT  ;INIT. THE STATE OF THE DRIVE
8868 040732 000401          BR      1$          ;TAKE ERROR EXIT
8869 040734 000207          RTS      PC         ;RETURN
8870 040736 005726          1$:    TST     (SP)+       ;POP PC OFF OF THE STACK
8871 040740 000704          BR      SC8        ;PROCESS THE PARITY ERROR
8872 040742 006301          SC13:   ASL     R1         ;SETUP TO ADDRESS WORDS
8873 040744 012761 177777 034576    MOV     #-1,TIMER(R1) ;STOP THE TIMER
8874 040752 006201          ASR     R1         ;
8875 040754 010164 000010          MOV     R1,RPCS2(R4) ;SELECT THE DRIVE
8876 040760 116164 034620 000016    MOVB    ATABIT(R1),RPAS(R4) ;CLEAR THE ATTENTION BIT
8877 040766 032714 004000          BIT     #BIT11,(R4)  ;DRIVE AVAILABLE ?
8878 040772 001006          BNE     1$         ;BR IF AVAILABLE
8879 040774 006301          ASL     R1         ;
8880 040776 012761 023420 034576    MOV     #10000.,TIMER(R1) ;START 10 SEC TIMER AGAIN
8881
8882 041004 006201          ASR     R1         ;
8883 041006 000433          BR      3$         ;EXIT
8884 041010 105761 034524          1$:    TSTB   DPINT(R1)  ;INITIALIZING THE DRIVE ?
8885 041014 001424          BEQ     2$         ;BR IF NOT
8886 041016 105061 034524          CLRB   DPINT(R1)  ;CLEAR THE INIT INDICATOR
8887 041022 004037 035062          JSR    R0,DRVINT  ;GO INIT THE DRIVE
8888 041026 000240          NOP                    ;DUMMY PARITY ERROR RETURN
8889 041030 105761 034504          TSTB   DRVSTA(R1)  ;DRIVE ONLINE ?
8890 041034 003014          BGT     2$         ;BR IF YES -- START ORDER
8891 041036 005702          TST     R2         ;QUEUE ENTRY FOR THE DRIVE
8892 041040 001416          BEQ     3$         ;BR IF NOT
8893 041042 004737 043036          JSR    PC,GETREQ  ;GET DPB ADDRESS
8894 041046 052762 140000 000016    BIS    #BIT15!BIT14,16(R2) ;INFORM USER THAT DRIVE OFFLINE
8895 041054 004737 042202          JSR    PC,SVRH11  ;SAVE THE REGISTERS
8896 041060 004737 042742          JSR    PC,EMPTYQ  ;EMPTY THE REQUEST QUEUE
8897 041064 000404          BR      3$         ;
8898 041066 105061 034534          2$:    CLRB   DPRQS(R1) ;CLEAR THE PORT REQUEST INDICATOR
8899 041072 004737 035742          JSR    PC,OPT     ;START THE PENDING REQUEST
8900 041076 000137 040024          3$:    JMP     SC4        ;PROCESS OTHER DRIVES
8901
8902          ;RP04/5/6 TIMER ROUTINE
8903          ;CALL
8904          ;      MOV     #TIME,-(SP) ;ELAPSED TIME IN MILLISECONDS ON THE STACK
8905          ;      JSR     PC,RPTMR  ;CALL RP04/5/6 TIME ROUTINE
8906
8907 041102 005737 034550          RPTMR: TST     ALDRV  ;CHECK "ACTDRV & ACTSTR"

```

```

8908 041106 001030      BNE      4$      ; IF NON ZERO EXIT
8909 041110 112737 000001 034551  MOVB     #1,ACTSTR ; SET "ACTSTR"
8910 041116 104412      SAVREG          ; SAVE R0 - R5
8911 041120 005001      CLR      R1      ; START WITH DRIVE 0
8912 041122 005003      CLR      R3
8913 041124 005763 034576 1$:  TST     TIMER(R3) ; IS THE TIMER RUNNING?
8914 041130 002407      BLT     2$      ; BRANCH IF NO
8915 041132 166663 000002 034576  SUB     2(SP),TIMER(R3) ; COUNT THE INTERVAL
8916 041140 003003      BGT     2$      ; BR IF NO SOFTWARE TIMEOUT
8917 041142 004737 041174  JSR     PC,STJ   ; CALL SOFTWARE TIMEOUT ROUTINE
8918 041146 000405      BR     3$      ; GO TO THE EXIT
8919 041150 005201 2$:  INC     R1      ; MOVE TO NEXT DRIVE
8920 041152 005723      TST     (R3)+
8921 041154 022701 000010  CMP     #8.,R1   ; OUT OF DRIVES?
8922 041160 003361      BGT     1$      ; BRANCH IF NO
8923 041162 104413 3$:  RESREG          ; RESTORE R0 - R5
8924 041164 105037 034551  CLRB   ACTSTR   ; ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
8925 041170 012616 4$:  MOV     (SP)+,(SP) ; ADJUST THE STACK
8926 041172 000207      RTS     PC      ; RETURN
8927
8928 ; SOFTWARE TIMEOUT ROUTINE
8929
8930 ; NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
8931 ; OR GREATER
8932
8933 ; CALL:
8934 ;     MOV     #DRVNUM,R1 ; DRIVE NUMBER
8935 ;     JSR     PC,STJ   ; CALL
8936 ;     RETURN
8937
8938 041174 010146  STO:  MOV     R1,-(SP) ; SAVE R1
8939 041176 010346      MOV     R3,-(SP) ; SAVE R3
8940 041200 013704 034632  MOV     RPADR,R4  ; GET ADDRESS OF "RPCS1"
8941 041204 010164 000010  MOV     R1,RPCS2(R4) ; SELECT THE DRIVE
8942 041210 004037 041630  JSR     R0,RD.RP  ; READ "DRIVE STATUS REG"
8943 041214 000012      RPDS1
8944 041216 041516      STOS
8945 041220 105726      TSTB   (SP)+     ; IS "DRY"=1?
8946 041222 100477      BMI    ST02     ; BR IF YES
8947 041224 105761 034524  ST01:  TSTB   DPINT(R1) ; TRYING TO INTIALIZE THE DRIVE ?
8948 041230 001074      BNE    ST02     ; BR IF YES
8949 041232 105761 034534  TSTB   DPRQS(R1) ; OUTSTANDING PORT REQUEST FOR THE DRIVE ?
8950 041236 001071      BNE    ST02     ; BR IF YES
8951 041240 013702 034544  MOV     TRNSWT,R2 ; PICKUP TRANSFER WAIT QUEUE
8952 041244 020137 034616  CMP     R1,DTUW  ; TRANSFER UNDERWAY ON THIS DRIVE?
8953 041250 001402      BEQ    1$      ; BRANCH IF YES
8954 041252 004737 043036  JSR     PC,GETREQ ; GET DPB ADDRESS
8955 041256 052762 101000 000016 1$:  BIS     #BIT15:BIT09,16(R2) ; SET THE ERROR FLAGS
8956 041264 004737 042202      JSR     PC,SVRH11 ; SAVE RH11/RP04/5/6 REGISTERS
8957 041270 012764 000040 000010  MOV     #BIT05,RPCS2(R4) ; "INIT" THE MASS BUS
8958 041276 105061 034474      CLRB   DRVACT(R1) ; DRIVE IS IDLE
8959 041302 105061 034552      CLRB   ULDFLG(R1) ; CLEAR THE UNLOAD FLAG
8960 041306 005001      CLR    R1      ; START WITH DRIVE 0
8961 041310 005003      CLR    R3
8962 041312 004037 035062 2$:  JSR     R0,DRVINT ; INIT. THIS DRIVE
8963 041316 000477      BR     STOS    ; PARITY ERROR RETURN

```

```

8964 041320 105761 034474      TSTB   DRVACT(R1)      ;DRIVE IDLE BEFORE THE INIT.?
8965 041324 001414      BEQ    48              ;YES--BRANCH
8966 041326 013702 034544      MOV    TRNSWT,R2      ;GET TRANSFER WAIT QUEUE
8967 041332 023701 034616      CMP    DTUW,R1        ;WAS THERE I/O ON THIS DRIVE?
8968 041336 001402      BEQ    38              ;YES--BRANCH
8969 041340 004737 043036      JSR    PC,GETREQ      ;GET THE DPB POINTER FROM QUEUE
8970 041344 052762 100400 000016 38:    BIS    #BIT15:BIT08,16(R2) ;INFORM USER OF INIT.
8971 041352 105061 034474      CLRB  DRVACT(R1)      ;SET DRIVE ACTIVE TO IDLE
8972 041356 105061 034552      CLRB  ULDFLG(R1)      ;NO UNLOAD
8973 041362 012763 177777 034576      MOV    #-1,TIMER(R3) ;STOP THE TIMER
8974 041370 005723      TST   (R3)+           ;UPDATE THE INDEX
8975 041372 005201      INC   R1              ;INCREMENT THE DRIVE NUMBER
8976 041374 022701 000010      CMP    #8.,R1         ;LAST DRIVE BEEN CHECKED?
8977 041400 003344      BGT   28              ;NO--LOOP
8978 041402 012737 177777 034616      MOV    #-1,DTUW       ;NO DATA TRANSFERS UNDERWAY
8979 041410 005037 034544      CLR   TRNSWT         ;CLEAR TRANSFER WAIT QUEUE
8980 041414 004737 042664      JSR   PC,CLRQUE      ;CLEAR ALL REQUEST QUEUES
8981 041420 000500      BR    ST09           ;EXIT
8982 041422 116405 000016      ST02: MOVB  RPAS(R4),R5 ;READ ATTENTION REG
8983 041426 136105 034620      BITB  ATABIT(R1),R5  ;IS ATTENTION FOR THIS DRIVE UP ?
8984 041432 001017      BNE   ST03           ;YES--BRANCH
8985 041434 105761 034524      TSTB  DPINT(R1)      ;TRYING TO INITIALIZE THE DRIVE ?
8986 041440 001031      BNE   ST06           ;BR IF YES - DRIVE NOT ONLINE
8987 041442 105761 034534      TSTB  DPRQS(R1)      ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
8988 041446 001045      BNE   ST07           ;BR IF YES - NO RESPONSE TO REQUEST
8989 041450 020137 034616      CMP   R1,DTUW        ;DATA TRANSFER UNDERWAY FOR THIS DRIVE
8990 041454 001263      BNE   ST01           ;BR IF NO
8991 041456 004037 041630      JSR   R0,RD.RP       ;YES--CHECK "RDY"
8992 041462 000000      RPCS1
8993 041464 041516      STOS
8994 041466 105726      TSTB  (SP)+
8995 041470 100255      BPL   ST01           ;BR IF "RDY"--0
8996 041472 105761 034524      ST03: TSTB  DPINT(R1) ;INITIALIZING THE DRIVE ?
8997 041476 001003      BNE   18             ;BR IF INIT PENDING
8998 041500 105761 034534      TSTB  DPRQS(R1)      ;PORT REQUEST PENDING ?
8999 041504 001446      BEQ   ST09           ;BR IF NOT
9000 041506 012763 177777 034576 18:    MOV    #-1,TIMER(R3) ;STOP THE TIMER
9001 041514 000442      BR    ST09           ;EXIT
9002 041516 004737 037152      ST05: JSR   PC,C18    ;GO HANDLE THE PARITY ERROR
9003 041522 000437      BR    ST09
9004 041524 105061 034524      ST06: CLRB  DPINT(R1) ;CLEAR THE INITIALIZE INDICATOR
9005 041530 105061 034504      CLRB  DRVSTA(R1)     ;SET UNIT OFFLINE
9006 041534 012763 177777 034576      MOV    #-1,TIMER(R3) ;STOP THE TIMER
9007 041542 004737 043036      JSR   PC,GETREQ      ;GET THE DPB ADDRESS
9008 041546 005702      TST   R2              ;REQUEST IN QUEUE ?
9009 041550 001424      BEQ   ST09           ;BR IF NOT
9010 041552 052762 140000 000016      BIS    #BIT15:BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
9011 041560 000414      BR    ST08           ;FINISH
9012 041562 012763 177777 034576  ST07: MOV    #-1,TIMER(R3) ;STOP THE TIMER
9013 041570 105061 034534      CLRB  DPRQS(R1)      ;CLEAR PORT REQUEST INDICATOR
9014 041574 004737 043036      JSR   PC,GETREQ      ;GET DPB ADDRESS
9015 041600 005702      TST   R2              ;QUEUE ENTRY FOR DRIVE ?
9016 041602 001407      BEQ   ST09           ;BR IF NONE
9017 041604 012762 100004 000016      MOV    #BIT15:BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
9018 041612 004737 042742      ST08: JSR   PC,EMPTYQ ;CLEAR THE QUEUE FOR THE DRIVE
9019 041616 004737 042202      JSR   PC,SVRW11     ;SAVE THE REGISTERS

```

```

9020 041622 012603 ST09: MOV (SP)+,R3 ;RESTORE R3
9021 041624 01260i MOV (SP)+,R1 ;RESTORE R1
9022 041626 000207 RTS PC ;RETURN
9023
9024 ;ROUTINE TO READ A RM11/RP04/5/6 REGISTER
9025 ;CALL
9026 ;CALL JSR RO,RD.RP ;GO READ A REGISTER
9027 ;INDEX ;REG. INDEX FROM BASE
9028 ;ERRADR ;ERROR ADDRESS--PROCESS ERROR STARTING
9029 ; ;AT THIS ADDRESS
9030 ; ;CONTENTS OF REG. IS ON THE STACK
9031 ;RETURN
9032
9033 041630 013737 034630 041776 RD.RP: MOV MCPEM,RD.RP2 ;MAX. RETRYS ALLOWED
9034 041636 011646 MOV (SP),-(SP) ;SAVE RO FOR RETURN
9035 041640 013737 034632 041654 MOV RPADR,RD.ADR ;FORM THE DESIRED ADDRESS
9036 041646 062037 041654 ADD (RO)+,RD.ADR ;USING THE BASE AND THE INDEX
9037 041652 013727 RD.RP1: MOV @ (PC)+,(PC)+ ;READ THE DESIRED REGISTER OF THE RP04
9038 041654 000000 RD.ADR: .WORD 0 ;ADDRESS IS FORMED HERE
9039 041656 000000 RD.WRD: .WORD 0 ;REG. CONTENTS PUT HERE
9040 041660 013766 041656 000002 MOV RD.WRD,2(SP) ;RETURN IT TO THE USER
9041 041666 013746 034632 MOV RPADR,-(SP) ;PUT THE ADDRESS ON THE STACK
9042 041672 062716 000010 ADD #RPCS2,(SP) ;FORM THE ADDRESS OF RPCS2
9043 041676 032736 010000 BIT #BIT12,@(SP)+ ;CHECK THE 'NED' BIT
9044 041702 001037 BNE RD.RP3 ;BR IF DRIVE NON-EXISTENT
9045 041704 017746 172722 MOV @RPADR,-(SP) ;READ RPCS1
9046 041710 032716 020000 BIT #BIT13,(SP) ;DID MCPE SET?
9047 041714 001002 BNE 1$ ;BRANCH IF YES
9048 041716 022620 CMP (SP)+,(RO)+ ;ADJUST FOR RETURN
9049 041720 000432 BR RD.RP4 ;EXIT
9050 041722 1$:
9051 041722 004037 043122 JSR RO,ES.SAV ;SAVE THE ADDRESS IN 'ESCAPE'
9052 041726 104003 ERROR 3 ;REPORT 'MCPE' ERROR
9053 041730 005737 034616 TST DTUW ;DATA TRANSFER UNDERWAY?
9054 041734 100405 BMI 2$ ;NO--BRANCH
9055 041736 032716 040000 BIT #BIT14,(SP) ;NO--'TRE'=1?
9056 041742 001402 BEQ 2$ ;NO--BRANCH
9057 041744 005726 TST (SP)+ ;YES--CLEAN OFF THE STACK AND
9058 041746 000415 BR RD.RP3 ;TAKE THE FATAL ERROR EXIT
9059 041750 052716 040000 2$: BIS #BIT14,(SP) ;CLEAR 'MCPE' BY SENDING A '1' TO 'TRE'
9060 041754 000316 SWAB (SP) ;POSITION BEFORE WRITING
9061 041756 013737 034632 041772 MOV RPADR,3$ ;FORM ADDRESS OF HIGH BYTE
9062 041764 005237 041772 INC 3$
9063 041770 112637 MOV (SP)+,@(PC)+ ;WRITE THE HIGH BYTE OF RPCS1
9064 041772 000000 3$: .WORD 0 ;ADDRESS STORAGE
9065 041774 005327 DEC (PC)+ ;EXCEEDED MAX. RETRYS
9066 041776 000003 RD.RP2: .WORD 3
9067 042000 002324 BGE RD.RP1 ;BRANCH IF NO
9068 042002 011000 RD.RP3: MOV (RO),RO ;FATAL ERROR EXIT
9069 042004 012616 MOV (SP)+,(SP)
9070 042006 000200 RD.RP4: RTS RO
9071
9072 ;ROUTINE TO WRITE A REGISTER
9073 ;CALL
9074 ;CALL MOV DATA,-(SP) ;DATA TO BE LOADED ON THE STACK
9075 ;

```

```

9076      :      JSR      RO,WRT.RP      ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
9077      :      INDEX     ;INDEX OF THE REGISTER TO BE LOADED
9078      :      ERRADR    ;ADDRESS TO RETURN TO ON AN ERROR
9079      :      RETURN   ;ERROR FREE RETURN
9080
9081 042010 013737 034630 042164 WRT.RP: MOV      MCPERM,WRT.R? ;MAX RETRYS ALLOWED
9082 042016 016637 000002 042076      MOV      2(SP),WRT.WD ;SAVE THE WORD TO WRITE
9083 042024 012616      MOV      (SP)+,(SP) ;ADJUST THE STACK
9084 042026 012037 042100      MOV      (RO)+,WRT.AD ;GET INDEX OF REGISTER TO BE WRITTEN
9085 042032 001015      BNE     1$ ;BRANCH IF NOT RPCS1
9086 042034 122737 000150 042076      CMPB    #150,WRT.WD ;IS THE COMMAND FOR DATA TRANSFERS?
9087 042042 002411      BLT     1$ ;YES--DON'T GET THE OLD A16 & A17, & PSEL
9088 042044 004037 041630      JSR     RO,RD.RP ;NO---COMBINE A16&A17, & PSEL WITH
9089 042050 000000      RPCS1 ;THE COMMAND BEFORE SENDING IT TO
9090 042052 042172      WRT.R3 ;THE RH11/RP04
9091 042054 000316      SWAB    (SP)
9092 042056 042716 177770      BIC     #^C7,(SP)
9093 042062 12637 042077      MOVB   (SP)+,WRT.WD+1
9094 042066 03737 034632 042100 1$:      ADD     RPADR,WRT.AD ;FORM THE ADDRESS OF THE DISK REG.
9095 042074 03737      WRT.R1: MOV     (PC)+,@(PC)+ ;LOAD THE DESIRED REG.
9096 042076 00 000      WRT.WD: .WORD 0 ;WORD TO WRITE GOES HERE
9097 042100 000000      WRT.AD: .WORD 0 ;ADDRESS IS FORMED HERE
9098 042102 013746 034632      MOV     RPADR,-(SP) ;PUT THE ADDRESS ON THE STACK
9099 042106 062716 000010      ADD     #RPCS2,(SP) ;FORM THE ADDRESS OF RPCS2
9100 042112 032736 010000      BIT     #BIT12,@(SP)+ ;CHECK THE 'NED' BIT
9101 042116 001025      BNE     WRT.R3 ;BR IF DRIVE NON-EXISTENT
9102 042120 004037 041630      JSR     RO,RD.RP ;CHECK FOR PARITY ERROR ON WRITE
9103 042124 000014      RPER1
9104 042126 042172      WRT.R3
9105 042130 032726 000010      BIT     #BIT03,(SP)+
9106 042134 001420      BEQ     WRT.R4 ;BRANCH IF "PAR-0"
9107 042136 016037 177776 042150      MOV     -2(RO),1$ ;PICKUP THE INDEX
9108 042144 004037 041630      JSR     RO,RD.RP ;READ THE REG.
9109 042150 000000      1$:      .WORD 0 ;REG. INDEX
9110 042152 042172      WRT.R3 ;RETURN TO THIS ADDRESS ON ERROR
9111 042154 004037 043122      JSR     RO,ES.SAV ;SAVE THE ADDRESS IN 'SESCAPE'
9112 042160 104004      ERROR  4 ;REPORT THE PARITY ON WRITE ERROR
9113 042162 005327      DEC     (PC)+ ;DECREMENT THE ERROR COUNT
9114 042164 000003      WRT.R2: .WORD 3 ;RETRY COUNTER
9115 042166 002342      BGE     WRT.R1 ;TRY AGAIN IF NOT FINISHED
9116 042170 005726      TST     (SP)+ ;CLEAN OFF THE STACK
9117 042172 011000      WRT.R3: MOV     (RO),RO ;TAKE THE "PARITY ON WRITE" ERROR EXIT
9118 042174 000401      BR      WRT.R5 ;EXIT
9119 042176 005720      WRT.R4: TST     (RO)+ ;ADJUST FOR ERROR FREE EXIT
9120 042200 000200      WRT.R5: RTS     RO
9121
9122      ;ROUTINE TO SAVE THE RH11/RP04/5/6 REGISTERS AS PER DPB+14
9123      :
9124      :CALL
9125      :      MOV     #DPBNUM,R2 ;DPB POINTER TO R2
9126      :      JSR     PC,SVRH11 ;SAVE THE DRIVES REG'S
9127
9128 042202 104412      SVRH11: SAVREG ;SAVE RO - R5
9129 042204 005702      TST     R2 ;QUEUE ENTRY FOR THE DRIVE ?
9130 042206 001430      BEQ     4$ ;BR IF NONE
9131 042210 013704 034632      MOV     RPADR,R4

```

```

9132 042214 111264 000010      MOVB   (R2),RPCS2(R4)  ;SELECT DRIVE
9133 042220 016203 000014      MOV    14(R2),R3      ;GET THE ERROR TABLE POINTER
9134 042224 001433              BEQ    6$              ;EXIT IF NO ADDRESS
9135 042226 005037 042262      CLR    3$              ;COUNTER & POINTER
9136 042232 023727 042262 000022 1$:  CMP    3$,#RPDB       ;REACHED THE BUFFER REGISTER ?
9137 042240 001006              BNE    2$              ;BR IF NOT
9138 042242 032764 000200 000010  BIT    #BIT07,RPCS2(R4) ;'OR' SET ?
9139 042250 001002              BNE    2$              ;BR IF SET
9140 042252 005023              CLR    (R3)+          ;STORE RPDB AS ZEROES
9141 042254 000405              BR     4$              ;CONTINUE
9142 042256 004037 041630      JSR    R0,RD.RP       ;READ THE SELECTED REGISTER
9143 042262 000000 3$:      .WORD 0               ;REGISTER INDEX
9144 042264 042310 5$:      .WORD 0               ;ERROR RETURN ADDRESS
9145 042266 012623              MOV    (SP)+,(R3)+    ;STORE THE REGISTER CONTENTS
9146 042270 023727 042262 000046 4$:  CMP    3$,#RPEC2     ;REACHED THE END ?
9147 042276 001406              BEQ    6$              ;BR IF YES
9148 042300 062737 000002 042262  ADD    #2,3$          ;INCREMENT THE REGISTER INDEX
9149 042306 000751              BR     1$              ;CONTINUE READING THE REGISTERS
9150 042310 004737 037052      JSR    PC,C17         ;PROCESS THE UNCORRECTABLE PARITY ERROR
9151 042314 104413 6$:      RESREG              ;RESTORE R0 - R5
9152 042316 000207      RTS    PC              ;RETURN
9153
9154      ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
9155      ;CALL
9156      ;      MOV    #DRVNUM,R1      ;DRIVE NUMBER TO R1
9157      ;      JSR    PC,SET.IE      ;SET "IE"
9158      ;      RETURN
9159
9160      SET.IE: MOV    R4,-(SP)      ;SAVE R4
9161      MOV    RPADR,R4      ;PICKUP ADDRESS OF RPCS1
9162      MOV    R1,RPCS2(R4)  ;SELECT DRIVE
9163      MOV    (R4),-(SP)    ;READ RPCS1
9164      BIS    #BIT14,(SP)  ;SET THE "TRE" BIT OF THE WORD READ
9165      SWAB  (SP)          ;ADJUST FOR DATO
9166      MOVB  #BIT06,(R4)   ;SET "IE"
9167      BIT    #BIT12,RPCS2(R4) ;IS "NED"=1?
9168      BNE    1$           ;YES--CLEAR "TRE"
9169      TST    (SP)+        ;CLEAN OFF THE STACK
9170      BR     2$
9171      1$:  MOVB  (SP)+,1(R4)  ;CLEAR "TRE"
9172      2$:  MOV    (SP)+,R4   ;RESTORE R4
9173      RTS    PC              ;RETURN TO CALLER
9174
9175      ;QUEUE COUNT
9176      QCNT: .BYTE 0         ;DRIVE 0
9177      .BYTE 0         ;DRIVE 1
9178      .BYTE 0         ;DRIVE 2
9179      .BYTE 0         ;DRIVE 3
9180      .BYTE 0         ;DRIVE 4
9181      .BYTE 0         ;DRIVE 5
9182      .BYTE 0         ;DRIVE 6
9183      .BYTE 0         ;DRIVE 7
9184
9185      ;QUEUE INPUT POINTERS
9186
9187      QINPT: .WORD QDRV0   ;DRIVE 0

```



```

9188 042404 042504 .WORD QDRV1 ;DRIVE 1
9189 042406 042524 .WORD QDRV2 ;DRIVE 2
9190 042410 042544 .WORD QDRV3 ;DRIVE 3
9191 042412 042564 .WORD QDRV4 ;DRIVE 4
9192 042414 042604 .WORD QDRV5 ;DRIVE 5
9193 042416 042624 .WORD QDRV6 ;DRIVE 6
9194 042420 042644 .WORD QDRV7 ;DRIVE 7
9195
9196 ;QUEUE OUTPUT POINTERS
9197
9198 042422 042464 QOUTPT: .WORD QDRV0 ;DRIVE 0
9199 042424 042504 .WORD QDRV1 ;DRIVE 1
9200 042426 042524 .WORD QDRV2 ;DRIVE 2
9201 042430 042544 .WORD QDRV3 ;DRIVE 3
9202 042432 042564 .WORD QDRV4 ;DRIVE 4
9203 042434 042604 .WORD QDRV5 ;DRIVE 5
9204 042436 042624 .WORD QDRV6 ;DRIVE 6
9205 042440 042644 .WORD QDRV7 ;DRIVE 7
9206
9207 042442 042464 QSTART: .WORD QDRV0 ;DRIVE 0 START ADDRESS
9208 042444 042504 QSTOP: .WORD QDRV1 ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
9209 042446 042524 .WORD QDRV2 ;STOP DRIVE 1--START DRIVE 2
9210 042450 042544 .WORD QDRV3 ;STOP DRIVE 2--START DRIVE 3
9211 042452 042564 .WORD QDRV4 ;STOP DRIVE 3--START DRIVE 4
9212 042454 042604 .WORD QDRV5 ;STOP DRIVE 4--START DRIVE 5
9213 042456 042624 .WORD QDRV6 ;STOP DRIVE 5--START DRIVE 6
9214 042460 042644 .WORD QDRV7 ;STOP DRIVE 6--START DRIVE 7
9215 042462 042664 .WORD QTERM ;STOP DRIVE 7
9216
9217 ;DRIVE REQUEST QUEUES
9218
9219 042464 000010 QDRV0: .BLKW 10
9220 042504 000010 QDRV1: .BLKW 10
9221 042524 000010 QDRV2: .BLKW 10
9222 042544 000010 QDRV3: .BLKW 10
9223 042564 000010 QDRV4: .BLKW 10
9224 042604 000010 QDRV5: .BLKW 10
9225 042624 000010 QDRV6: .BLKW 10
9226 042644 000010 QDRV7: .BLKW 10
9227 042664 000010 QTERM=.
9228
9229 ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
9230 ;
9231 ;CALL
9232 ; JSR PC,CLRQUE
9233
9234 042664 104412 CLRQUE: SAVREG ;SAVE R0 - R5
9235 042666 012702 042372 MOV #QCNT,R2 ;ZERO THE QUEUE COUNTS
9236 042672 005022 CLR (R2)+ ;DRIVES 0 & 1
9237 042674 005022 CLR (R2)+ ;DRIVES 2 & 3
9238 042676 005022 CLR (R2)+ ;DRIVES 4 & 5
9239 042700 005022 CLR (R2)+ ;DRIVES 6 & 7
9240 042702 012703 000010 MOV #8,R3 ;MOVE THE STARTING
9241 042706 012701 042442 MOV #QSTART,R1 ;ADDRESS OF THE QUEUE INTO
9242 042712 012122 1$: MOV (R1)+,(R2)+ ;THE QUEUE INPUT POINTER
9243 042714 005303 DEC R3

```

```

9244 042716 001375          BNE      1$
9245 042720 012703 000010  MOV      #8.,R3          ;MOVE THE STARTING ADDRESS
9246 042724 012701 042442  MOV      #QSTART,R1     ;OF THE QUEUE INTO THE
9247 042730 012122          2$: MOV      (R1)+,(R2)+     ;QUEUE OUTPUT POINTER
9248 042732 005303          DEC      R3
9249 042734 001375          BNE      2$
9250 042736 104413          RESREG          ;RESTORE R0 - R5
9251 042740 000207          RTS      PC
9252
9253          ;EMPTY THE QUEUE SPECIFIED BY R1
9254          ;
9255          ;CALL
9256          ;      MOV      DRVNUM,R1          ;DRIVE NUMBER TO R1
9257          ;      JSR      PC,EMPTYQ
9258
9259 042742 105061 042372  EMPTYQ: CLRB   QCNT(R1)          ;CLEAR NUMBER OF ITEMS IN QUEUE
9260 042746 006301          ASL      R1
9261 042750 016161 042402 042422  MOV      QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER INPUT POINTER
9262 042756 006201          ASR      R1
9263 042760 000207          RTS      PC
9264
9265          ;ROUTINE TO PUT A REQUEST IN QUEUE
9266          ;
9267          ;CALL
9268          ;      MOV      #DRVNUM,R1          ;DRIVE NUMBER
9269          ;      MOV      #DPB,R2           ;ADDRESS OF PARAMETER BLOCK
9270          ;      JSR      R0,DRVQUE        ;GO PUT REQUEST IN QUEUE
9271          ;      RETURN1                   ;RETURN HERE IF QUEUE IS FULL
9272          ;      RETURN2                   ;RETURN HERE IF REQUEST IS IN QUEUE
9273
9274 042762 122761 000010 042372  DRVQUE: CMPB   #10,QCNT(R1)      ;IS QUEUE FULL?
9275 042770 001421          BEQ      2$              ;BR IF YES-TAKE RETURN1
9276 042772 105261 042372          INCB   QCNT(R1)          ;INCREMENT QUEUE COUNT
9277 042776 006301          ASL      R1
9278 043000 010271 042402          MOV      R2,@QINPT(R1)   ;PUT THIS REQUEST IN QUEUE
9279 043004 062761 000002 042402  ADD      #2,QINPT(R1)     ;UPDATE THE QUEUE POINTER
9280 043012 026161 042402 042444  CMP      QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
9281 043020 001003          BNE      1$              ;BRANCH IF NO
9282 043022 016161 042442 042402  MOV      QSTART(R1),QINPT(R1) ;YES--RESET POINTER
9283 043030 006201          1$: ASR      R1
9284 043032 005720          TST     (R0)+           ;TAKE RETURN 2
9285 043034 000200          2$: RTS      R0          ;RETURN TO USER
9286
9287          ;ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUF
9288          ;
9289          ;CALL
9290          ;      MOV      #DRVNUM,R1          ;DRIVE NUMBER TO R1
9291          ;      JSR      PC,GETREQ        ;GO GET THE REQUEST
9292          ;      RETURN                    ;R2="DPB" ADDRESS OF THE REQUEST
9293          ;      ;R2=0 IF NO REQUEST IN QUEUE
9294
9295 043036 005002          GETREQ: CLR      R2
9296 043040 105761 042372          TSTB   QCNT(R1)          ;IS THERE ANY REQUEST IN QUEUE?
9297 043044 001404          BEQ      2$              ;NO---BRANCH
9298 043046 006301          1$: ASL      R1
9299 043050 017102 042422          MOV      @QOUTPT(R1),R2 ;PICKUP "DPB" POINTER FOR THIS DRIVE
  
```

```

9300 043054 006201          ASR      R1
9301 043056 000207      28:      RTS      PC          ;RETURN TO USER
9302
9303          ;ROUTINE TO "POP" THE REQUEST FROM QUEUE
9304          ;
9305          ;CALL
9306          ;          MOV      #DRVNUM,R1          ;DRIVE NUMBER TO R1
9307          ;          JSR      PC,POPQUE          ;CALL TO REMOVE REQUEST
9308          ;          RETURN          ;R2=ADDRESS OF DPB REMOVED
9309
9310 043060 105361 042372      POPQUE: DECB     QCNT(R1)          ;DECREMENT QUEUE COUNT
9311 043064 006301          ASL      R1
9312 043066 017102 042422      MOV      @QOUTPT(R1),R2          ;GET THE 'DPB' POINTER
9313 043072 062761 000002 042422      ADD     #2,QOUTPT(R1)          ;UPDATE THE QUEUE POINTER
9314 043100 026161 042422 042444      CMP     QOUTPT(R1),QSTOP(R1)    ;TIME TO RESET THE POINTER?
9315 043106 001003          BNE     18          ;NO--BRANCH TO EXIT
9316 043110 016161 042442 042422      MOV     QSTART(R1),QOUTPT(R1)   ;YES--RESET THE POINTER
9317 043116 006201      18:      ASR      R1
9318 043120 000207          RTS      PC          ;RETURN TO USER
9319
9320          ;ROUTINE TO SAVE THE CONTENTS OF '$ESCAPE' WHEN THE DRIVER
9321          ;REPORTS AN ERROR DIRECTLY.
9322          ;
9323          ;CALL
9324          ;          JSR      R0,ES.SAV          ;:THE ERROR CALL
9325          ;          ERROR   N          ;:THE RETURN IS PAST THE ERROR CALL
9326          ;          RETURN
9327
9328 043122 012037 043136      ES.SAV: MOV     (R0)+,18          ;GET THE ERROR CALL
9329 043126 013746 001206      MOV     $ESCAPE,-(SP)          ;SAVE THE ADDRESS IN '$ESCAPE'
9330 043132 005037 001206      CLR     $ESCAPE          ;CLEAR THE ESCAPE RETURN
9331 043136 000000      18:      .WORD   0          ;THE ERROR CALL IS MOVED HERE
9332 043140 012637 001206      MOV     (SP)+,$ESCAPE          ;RESTORE THE ESCAPE ADDRESS
9333 043144 000200          RTS      R0          ;RETURN
9334
9335
9336          .SBTTL  ASCIZ  MESSAGES
9337
9338 043146 000122          MSG.R:  .ASCIZ  /R/
9339 043150 041506          MSG.FC:  .ASCIZ  /FC/
9340 043153          114 000103      MSG.LC:  .ASCIZ  /LC/
9341 043156 041506 000047      MSG.FCP: .ASCIZ  /FC'/
9342 043162 041514 000047      MSG.LCP: .ASCIZ  /LC'/
9343 043166 041511          000      MSG.IC:  .ASCIZ  /IC/
9344 043171          106 000124      MSG.FT:  .ASCIZ  /FT/
9345 043174 052114          000      MSG.LT:  .ASCIZ  /LT/
9346 043177          111 000124      MSG.IT:  .ASCIZ  /IT/
9347 043202 051506          000      MSG.FS:  .ASCIZ  /FS/
9348 043205          114 000123      MSG.LS:  .ASCIZ  /LS/
9349 043210 040520 000124      MSG.PAT: .ASCIZ  /PAT/
9350 043214 000075          MSG.EQ:  .ASCIZ  /=/
9351 043216 005015 047503 052116      MSG.CS:  .ASCIZ  <CR><LF>/CONTROL SWITCHES=/
9352 043224 047522 020114 053523
9353 043232 052111 044103 051505
9354 043240 000075
9355 043242 005015 040527 047122      USE:    .ASCIZ  <CR><LF>/WARNING: PROGRAMMABLE DRIVES MAY BE USED/

```

9356 043250 047111 035107 050040  
 9357 043256 047522 051107 046501  
 9358 043264 040515 046102 020105  
 9359 043272 051104 053111 051505  
 9360 043300 046440 054501 041040  
 9361 043306 020105 051525 042105  
 9362 043314 005015 000  
 9363 043317 040 051120 043517  
 9364 043324 040522 046515 041101  
 9365 043332 042514 042055 044522  
 9366 043340 042526 053440 046111  
 9367 043346 020114 047516 020124  
 9368 043354 042502 052440 042523  
 9369 043362 000104  
 9370  
 9371 043364 027440 000040  
 9372 043370 047125 052111 051440  
 9373 043376 040524 052524 035123  
 9374 043404 005015 000012  
 9375 043410 051104 053111 000105  
 9376 043416 047440 043106 044514  
 9377 043424 042516 000  
 9378 043427 040 047117 044514  
 9379 043434 042516 000  
 9380 043437 040 047516 020124  
 9381 043444 051120 051505 047105  
 9382 043452 000124  
 9383 043454 052440 051516 043101  
 9384 043462 000105  
 9385 043464 047040 052117 051040  
 9386 043472 030120 027464 027465  
 9387 043500 000066  
 9388 043502 050122 032060 000  
 9389 043507 122 030120 000065  
 9390 043514 050122 033060 000  
 9391 043521 015 042012 044522  
 9392 043526 042526 051450 020051  
 9393 043534 047524 041040 020105  
 9394 043542 042524 052123 042105  
 9395 043550 000040  
 9396 043552 047516 042516 000  
 9397 043557 054 000  
 9398 043561 015 047012 020117  
 9399 043566 053513 030461 050055  
 9400 043574 041440 047514 045503  
 9401 043602 020054 044524 044515  
 9402 043610 043516 052040 051505  
 9403 043616 051524 053440 046111  
 9404 043624 020114 047516 020124  
 9405 043632 042502 050040 051105  
 9406 043640 047506 046522 042105  
 9407 043646 000  
 9408 043647 015 005012 042524  
 9409 043654 052123 047111 020107  
 9410 043662 051104 053111 020105  
 9411 043670 000

MOUSE: .ASCIZ / PROGRAMMABLE-DRIVE WILL NOT BE USED/

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

UNMSG: .ASCIZ /DRIVE/  
UNTOFF: .ASCIZ / OFFLINE/

UNTON: .ASCIZ / ONLINE/

NOTPRS: .ASCIZ / NOT PRESENT/

NOTSAF: .ASCIZ / UNSAFE/

NOTRP: .ASCIZ @ NOT RP04/5/6@

RP04B: .ASCIZ /RP04/  
RP05: .ASCIZ /RPC5/  
RP06: .ASCIZ /RP06/  
DRIVES: .ASCIZ <CR><LF>/DRIVE(S) TO BE TESTED /

NONE: .ASCIZ /NONE/  
COMMA: .ASCIZ /,/ /  
NOCLOCK: .ASCIZ <CR><LF>/NO KW11-P CLOCK, TIMING TESTS WILL NOT BE PERFORMED/

TESTNG: .ASCIZ <CR><LF><LF>/TESTING DRIVE /

9412	043671	123	051105	040511	SERIAL: .ASCIZ /SERIAL NUMBER /
9413	043676	020114	052516	041115	
9414	043704	051105	000040		
9415					
9416	043710	005015	051012	052117	MSG7: .ASCIZ <CR><LF><LF>/ROTATIONAL SPEED TIMES/
9417	043716	052101	047511	040516	
9418	043724	020114	050123	042505	
9419	043732	020104	044524	042515	
9420	043740	000123			
9421	043742	005015	047412	042516	MSG10A: .ASCIZ <CR><LF><LF>/ONE CYLINDER SEEK TIMES/<CR><LF>/ * FORWARD/
9422	043750	041440	046131	047111	
9423	043756	042504	020122	042523	
9424	043764	045505	052040	046511	
9425	043772	051505	005015	025040	
9426	044000	043040	051117	040527	
9427	044006	042122	000		
9428	044011	015	020012	020052	MSG10B: .ASCIZ <CR><LF>/ * REVERSE/
9429	044016	042522	042526	051522	
9430	044024	000105			
9431	044026	005015	040412	041503	MSG11A: .ASCIZ <CR><LF><LF>/ACCESS TIME MEASUREMENT/<CR><LF>/ * FORWARD/
9432	044034	051505	020123	044524	
9433	044042	042515	046440	040505	
9434	044050	052523	046522	047105	
9435	044056	006524	020012	020052	
9436	044064	047506	053522	051101	
9437	044072	000104			
9438	044074	005015	025040	051040	MSG11B: .ASCIZ <CR><LF>/ * REVERSE/
9439	044102	053105	051105	042523	
9440	044110	000			
9441	044111	015	005012	040515	MSG12A: .ASCIZ <CR><LF><LF>/MAXIMUM SEEK TIMES/<CR><LF>/ * FORWARD/
9442	044116	044530	052515	020115	
9443	044124	042523	045505	052040	
9444	044132	046511	051505	005015	
9445	044140	025040	043040	051117	
9446	044146	040527	042122	000	
9447	044153	015	020012	020052	MSG12B: .ASCIZ <CR><LF>/ * REVERSE/
9448	044160	042522	042526	051522	
9449	044166	000105			
9450					
9451	044170	005015	044515	036516	MSGMIN: .ASCIZ <CR><LF>/MIN=/
9452	044176	000			
9453	044177	015	046412	054101	MSGMAX: .ASCIZ <CR><LF>/MAX-/
9454	044204	000075			
9455	044206	005015	053101	036507	MSGAVG: .ASCIZ <CR><LF>/AVG=/
9456	044214	000			
9457	044215	060	052440	000123	MSGOUS: .ASCIZ /0 US/
9458	044222	041040	046105	053517	MBELOW: .ASCIZ / BELOW THE MINIMUM OF /
9459	044230	052040	042510	046440	
9460	044236	047111	046511	046525	
9461	044244	047440	020106	000	
9462	044251	040	041101	053117	MABOVE: .ASCIZ / ABOVE THE MAXIMUM OF /
9463	044256	020105	044124	020105	
9464	044264	040515	044530	052515	
9465	044272	020115	043117	000040	
9466	044300	051440	042505	051513	MSGNUM: .ASCIZ / SEEKS TIMED/
9467	044306	052040	046511	042105	

9468	044314	000			
9469	044315	040	047516	020124	MSGNON: .ASCIZ / NOT TIMED/
9470	044322	044524	042515	000104	
9471	044330	020040	000		MSG.SP: .ASCIZ / / ;TWO (2) SPACES
9472					
9473					.SBTTL ERROR HEADER (EM) MESSAGES
9474					
9475	044333	122	030510	020061	EM1: .ASCIZ /RH11 INTERRUPT OCCURRED (RPAS = 0)/
9476	044340	047111	042524	051122	
9477	044346	050125	020124	041517	
9478	044354	052503	051122	042105	
9479	044362	024040	050122	051501	
9480	044370	036440	030040	000051	
9481	044376	047125	054105	042520	EM2: .ASCIZ /UNEXPECTED ATTENTION OCCURRED/
9482	044404	052103	042105	040440	
9483	044412	052124	047105	044524	
9484	044420	047117	047440	041503	
9485	044426	051125	042522	000104	
9486	044434	040515	051523	052502	EM3: .ASCIZ /MASSBUS PARITY ERROR(MCPE-1)/
9487	044442	020123	040520	044522	
9488	044450	054524	042440	051122	
9489	044456	051117	046450	050103	
9490	044464	036505	024461	000	
9491	044471	115	051501	041123	EM4: .ASCIZ /MASSBUS PARITY ERROR(PAR-1)/
9492	044476	051525	050040	051101	
9493	044504	052111	020131	051105	
9494	044512	047522	024122	040520	
9495	044520	036522	024461	000	
9496	044525	101	042104	042522	EM5: .ASCIZ /ADDRESS PLUG CHANGE BIT SET/
9497	044532	051523	050040	052514	
9498	044540	020107	044103	047101	
9499	044546	042507	041040	052111	
9500	044554	051440	052105	000	
9501	044561	122	030510	027461	EM10: .ASCIZ "RH11/RPO4/5/6 FAILED TO RESPOND TO ADDRESSING"
9502	044566	050122	032060	032457	
9503	044574	033057	043040	044501	
9504	044602	042514	020104	047524	
9505	044610	051040	051505	047520	
9506	044616	042116	052040	020117	
9507	044624	042101	051104	051505	
9508	044632	044523	043516	000	
9509	044637	104	044522	042526	EM11: .ASCIZ /DRIVE SELECTED IS NOT ONLINE/
9510	044644	051440	046105	041505	
9511	044652	042524	020104	051511	
9512	044660	047040	052117	047440	
9513	044666	046116	047111	000105	
9514	044674	046511	051120	050117	EM12: .ASCIZ /IMPROPER HEADER DATA/
9515	044702	051105	044040	040505	
9516	044710	042504	020122	040504	
9517	044716	040524	000		
9518	044721	104	052101	020101	EM13: .ASCIZ /DATA COMPARE FAILURE/
9519	044726	047503	050115	051101	
9520	044734	020105	040506	046111	
9521	044742	051125	000105		
9522	044746	044504	045523	042440	EM17: .ASCIZ /DISK ERROR IN TIMING TEST/
9523	044754	051122	051117	044440	

9524	044762	020116	044524	044515	
9525	044770	043516	052040	051505	
9526	044776	000124			
9527	045000	046103	041517	020113	EM20: .ASCIZ /CLOCK (KW11-P) OVERFLOW IN TIMING TEST/
9528	045006	045450	030527	026461	
9529	045014	024520	047440	042526	
9530	045022	043122	047514	020127	
9531	045030	047111	052040	046511	
9532	045036	047111	020107	042524	
9533	045044	052123	000		
9534	045047	104	051511	020113	EM23: .ASCIZ /DISK ERROR DURING SEEK/
9535	045054	051105	047522	020122	
9536	045062	052504	044522	043516	
9537	045070	051440	042505	000113	
9538	045076	042523	045505	047040	EM24: .ASCIZ /SEFK NOT COMPLETE WITHIN 120 MS/
9539	045104	052117	041440	046517	
9540	045112	046120	052105	020105	
9541	045120	044527	044124	047111	
9542	045126	030440	030062	046440	
9543	045134	000123			
9544	045136	044122	030461	051057	EM41: .ASCIZ "RH11/RP04/5/6 ERROR"
9545	045144	030120	027464	027465	
9546	045152	020066	051105	047522	
9547	045160	000122			
9548	045162	040506	040524	020114	EM46: .ASCIZ /FATAL WRITE CHECK ERROR/
9549	045170	051127	052111	020105	
9550	045176	044103	041505	020113	
9551	045204	051105	047522	000122	
9552					
9553					.SBTTL STATUS/ERROR INDICATOR MESSAGES
9554					
9555	045212	043117	046106	047111	MSGB14: .ASCIZ /OFFLINE OR UNSAFE DRIVE REQUESTED/
9556	045220	020105	051117	052440	
9557	045226	051516	043101	020105	
9558	045234	051104	053111	020105	
9559	045242	042522	052521	051505	
9560	045250	042524	000104		
9561	045254	047125	047514	042101	MSGB13: .ASCIZ /UNLOADED DRIVE REQUESTED/
9562	045262	042105	042040	044522	
9563	045270	042526	051040	050505	
9564	045276	042525	052123	042105	
9565	045304	000			
9566	045305	120	051105	044523	MSGB12: .ASCIZ /PERSISTENT UNSAFE/
9567	045312	052123	047105	020124	
9568	045320	047125	040523	042506	
9569	045326	000			
9570	045327	120	051101	052111	MSGB11: .ASCIZ /PARITY ERROR OCCURRED/
9571	045334	020131	051105	047522	
9572	045342	020122	041517	052503	
9573	045350	051122	042105	000	
9574	045355	106	052101	046101	MSGB10: .ASCIZ /FATAL PARITY ERROR/
9575	045362	050040	051101	052111	
9576	045370	020131	051105	047522	
9577	045376	000122			
9578	045406	047523	052106	040527	MSGB09: .ASCIZ /SOFTWARE TIMEOUT ON THIS DRIVE/
9579	045406	042522	052040	046511	

9580 045414 047505 052125 047440  
 9581 045422 020116 044124 051511  
 9582 045430 042040 044522 042526  
 9583 045436 000  
 9584 045437 123 043117 053524  
 9585 045444 051101 020105 044524  
 9586 045452 042515 052517 020124  
 9587 045460 047117 040440 047516  
 9588 045466 044124 051105 042040  
 9589 045474 044522 042526 000  
 9590 045501 105 051122 051117  
 9591 045506 047440 041503 051125  
 9592 045514 042522 020104 052504  
 9593 045522 044522 043516 044440  
 9594 045530 047457 047440 042520  
 9595 045536 040522 044524 047117  
 9596 045544 000  
 9597 045545 105 051122 051117  
 9598 045552 047440 041503 051125  
 9599 045560 042522 020104 052504  
 9600 045566 044522 043516 047040  
 9601 045574 047117 044455 047457  
 9602 045602 047440 042520 040522  
 9603 045610 044524 047117 000  
 9604 045615 125 051516 043101  
 9605 045622 020105 041517 052503  
 9606 045630 051122 042105 000  
 9607 045635 101 052125 046517  
 9608 045642 052101 041511 051040  
 9609 045650 041505 046101 041111  
 9610 045656 040522 042524 051440  
 9611 045664 050505 042525 041516  
 9612 045672 020105 041517 052503  
 9613 045700 051122 042105 000  
 9614 045705 104 044522 042526  
 9615 045712 044040 051501 047040  
 9616 045720 052117 051040 051505  
 9617 045726 047520 042116 042105  
 9618 045734 052040 020117 047520  
 9619 045742 052122 051040 050505  
 9620 045750 042525 052123 000  
 9621 045755 104 044522 042526  
 9622 045762 044040 051501 041040  
 9623 045770 41505 046517 020105  
 9624 045776 047516 026516 054105  
 9625 046004 051511 042524 052116  
 9626 046012 000  
 9627  
 9628  
 9629  
 9630 046013 105 051122 050040  
 9631 046020 020103 051040 040520  
 9632 046026 000123  
 9633 046030 051105 020122 041520  
 9634 046036 020040 051104 053111  
 9635 046044 020105 020040 050122

MSGB08: .ASCIZ /SOFTWARE TIMEOUT ON ANOTHER DRIVE/

MSGB06: .ASCIZ "ERROR OCCURRED DURING I/O OPERATION"

MSGB05: .ASCIZ "ERROR OCCURRED DURING NON-I/O OPERATION"

MSGB04: .ASCIZ /UNSAFE OCCURRED/

MSGB03: .ASCIZ /AUTOMATIC RECALIBRATE SEQUENCE OCCURRED/

MSGB02: .ASCIZ /DRIVE HAS NOT RESPONDED TO PORT REQUEST/

MSGB01: .ASCIZ /DRIVE HAS BECOME NON-EXISTENT/

.SBTTL DATA HEADER (DT) MESSAGES

DM1: .ASCIZ /ERR PC RPAS/

DM2: .ASCIZ /ERR PC DRIVE RPAS RPDS1 RPER1 RPER2 RPER3/









9804	047634	004220	004244	004246	DT23A:	.WORD	RP.REG+14,RP.REG+40,RP.REG+42,RP.REG+34,RP.REG+36
9805	047642	004240	004242				
9806	047646	001176	001116	001162	DT41:	.WORD	\$TMPO,\$ERRPC,\$REGO,CHKDRV
9807	047654	001254					
9808	047656	001176	001116	001162	DT42:	.WORD	\$IMPO,\$ERRPC,\$REGO,CHKDRV,RP.REG,RP.REG+10,RP.REG+12
9809	047664	001254	004204	004214			
9810	047672	004216					
9811	047674	001176	001116	001162	DT43:	.WORD	\$TMPO,\$ERRPC,\$REGO,CHKDRV,RP.REG,RP.REG+10,RP.REG+12
9812	047702	001254	004204	004214			
9813	047710	004216					
9814	047712	004220	004244	004246	DT43A:	.WORD	RP.REG+14,RP.REG+40,RP.REG+42
9815	047720	001176	001116	001162	DT44:	.WORD	\$TMPO,\$ERRPC,\$REGO,CHKDRV,CYL.DS,TRK.DS,SEC.DS
9816	047726	001254	001270	001274			
9817	047734	001272					
9818	047736	004204	004214	004216	DT44A:	.WORD	RP.REG,RP.REG+10,RP.REG+12,RP.REG+36,RP.REG+34,RP.REG+06
9819	047744	004242	004240	004212			
9820	047752	004220	004244	004246	DT44B:	.WORD	RP.REG+14,RP.REG+40,RP.REG+42
9821	047760	001176	001116	001162	DT45:	.WORD	\$TMPO,\$ERRPC,\$REGO,CHKDRV,CYL.DS,TRK.DS,SEC.DS
9822	047766	001254	001270	001274			
9823	047774	001272					
9824	047776	004204	004214	004216	DT45A:	.WORD	RP.REG,RP.REG+10,RP.REG+12,RP.REG+36,RP.REG+34,RP.REG+06
9825	050004	004242	004240	004212			
9826	050012	004220	004244	004246	DT45B:	.WORD	RP.REG+14,RP.REG+40,RP.REG+42,RP.REG+2,RP.REG+4,RP.REG+22
9827	050020	004206	004210	004226			

.SBTTL DATA FORMAT (DF) TABLE

9831	050026	000001			DF1:	.WORD	1	:NUMBER OF DATA HEADERS
9832	050030	002				.BYTE	2	:NUMBER OF WORDS IN DATA TABLE
9833	050031	000				.BYTE	0	:ALL 3 NUMBERS ARE OCTAL
9834								
9835	050032	000001			DF2:	.WORD	1	
9836	050034	007				.BYTE	7	
9837	050035	000				.BYTE	0	
9838								
9839	050036	000001			DF3:	.WORD	1	
9840	050040	004				.BYTE	4	
9841	050041	000				.BYTE	0	
9842								
9843	050042	000001			DF4:	.WORD	1	
9844	050044	005				.BYTE	5	
9845	050045	000				.BYTE	0	
9846								
9847	050046	000001			DF10:	.WORD	1	
9848	050050	002				.BYTE	2	
9849	050051	000				.BYTE	0	
9850								
9851	050052	000001			DF11:	.WORD	1	
9852	050054	002				.BYTE	2	
9853	050055	000				.BYTE	0	
9854								
9855	050056	000002			DF12:	.WORD	2	:2 DM'S TO BE TYPED
9856	050060	007				.BYTE	7	:7 DATA WORDS FOLLOW THE 1ST DM
9857	050061	160				.BYTE	160	:WORDS 1-4 ARE OCTAL 5-7 ARE DECIMAL
9858	050062	046347				.WORD	DM12A	:ADDRESS OF 2ND DM
9859	050064	006				.BYTE	6	:6 DATA WORDS FOLLOW THE 2ND DM

9860	050065	000		.BYTE	0		:ALL WORDS ARE OCTAL
9861							
9862	050066	000002	DF13:	.WORD	2		
9863	050070	007		.BYTE	7		
9864	050071	160		.BYTE	160		
9865	050072	046426		.WORD	DH13A		
9866	050074	005		.BYTE	5		
9867	050075	004		.BYTE	4		:WORD 3 IS DECIMAL
9868							
9869	050076	000000	DF14:	.WORD	0		
9870	050100	005		.BYTE	5		
9871	050101	004		.BYTE	4		:WORD 3 IS DECIMAL
9872							
9873	050102	000001	DF17:	.WORD	1		
9874	050104	010		.BYTE	^D8		
9875	050105	000		.BYTE	0		
9876							
9877	050106	000002	DF21:	.WORD	2		
9878	050110	006		.BYTE	6		
9879	050111	060		.BYTE	60		
9880	050112	046650		.WORD	DH21A		
9881	050114	004		.BYTE	4		
9882	050115	014		.BYTE	14		
9883							
9884	050116	000000	DF22:	.WORD	0		
9885	050120	004		.BYTE	4		
9886	050121	014		.BYTE	14		
9887							
9888	050122	000002	DF23:	.WORD	2		
9889	050124	007		.BYTE	7		
9890	050125	010		.BYTE	10		:WORD 4 IS DECIMAL
9891	050126	046775		.WORD	DH2 <sup>7</sup> A		
9892	050130	005		.BYTE	5		
9893	050131	000		.BYTE	0		
9894							
9895							
9896	050132	000001	DF41:	.WORD	1		
9897	050134	004		.BYTE	4		
9898	050135	000		.BYTE	0		
9899							
9900	050136	000001	DF42:	.WORD	1		
9901	050140	007		.BYTE	7		
9902	050141	000		.BYTE	0		
9903							
9904	050142	000002	DF43:	.WORD	2		
9905	050144	007		.BYTE	7		
9906	050145	000		.BYTE	0		
9907	050146	047166		.WORD	DH43A		
9908	050150	003		.BYTE	3		
9909	050151	000		.BYTE	0		
9910							
9911	050152	000003	DF44:	.WORD	3		
9912	050154	007		.BYTE	7		
9913	050155	160		.BYTE	160		
9914	050156	047214		.WORD	DH44A		
9915	050160	006		.BYTE	6		

9916	050161	000			.BYTE	0
9917	050162	047271			.WORD	DH44B
9918	050164	003			.BYTE	3
9919	050165	000			.BYTE	0
9920						
9921	050166	000003		DF45:	.WORD	3
9922	050170	007			.BYTE	7
9923	050171	160			.BYTE	160
9924	050172	047214			.WORD	DH44A
9925	050174	006			.BYTE	6
9926	050175	000			.BYTE	0
9927	050176	047317			.WORD	DH45A
9928	050200	006			.BYTE	6
9929	050201	000			.BYTE	0
9930						
9931				.EVEN		
9932		050202		BUFFER=.		
9933						
9934	050202	005015	041412	051132	TITLE: .ASCII	<CR><LF><LF>/CZRJAD/<CR><LF>
9935	050210	040512	006504	012		
9936	050215	122	030120	027464	.ASCIIZ	@RPO4/5/6 MECHANICAL & READ-WRITE TEST@<CR><LF><LF>
9937	050222	027465	020066	042515		
9938	050230	044103	047101	041511		
9939	050236	046101	023040	051040		
9940	050244	040505	026504	051127		
9941	050252	052111	020105	042524		
9942	050260	052123	005015	000012		
9943	050266	005015	047524	052040	LOADRV: .ASCII	<CR><LF>/TO TEST DRIVE 0 REPLACE THE 'XXDP' PACK ON DRIVE 0/<CR><LF>
9944	050274	051505	020124	051104		
9945	050302	053111	020105	020060		
9946	050310	042522	046120	041501		
9947	050316	020105	044124	020105		
9948	050324	054047	042130	023520		
9949	050332	050040	041501	020113		
9950	050340	047117	042040	044522		
9951	050346	042526	030040	005015		
9952	050354	044527	044124	040440	.ASCII	/WITH ANOTHER PACK, CLEAR MEMORY LOCATION 40, AND RESTART/<CR><LF>
9953	050362	047516	044124	051105		
9954	050370	050040	041501	026113		
9955	050376	041440	042514	051101		
9956	050404	046440	046505	051117		
9957	050412	020131	047514	040503		
9958	050420	044524	047117	032040		
9959	050426	026060	040440	042116		
9960	050434	051040	051505	040524		
9961	050442	052122	005015			
9962	050446	044124	020105	051120	.ASCIIZ	/THE PROGRAM/<CR><LF>
9963	050454	043517	040522	006515		
9964	050462	000012				
9965	050464	005015	054523	052123	NOLOAD: .ASCIIZ	<CR><LF>'SYSTEM HAS 16K MEMORY, 'XXDP' LOADER WILL BE OVERWRITTEN'<CR><LF>
9966	050472	046505	044040	051501		
9967	050500	030440	045466	046440		
9968	050506	046505	051117	026131		
9969	050514	023440	054130	050104		
9970	050522	020047	047514	042101		
9971	050530	051105	053440	046111		

9972	050536	020114	042502	047440
9973	050544	042526	053522	044522
9974	050552	052124	047105	005015
9975	050560	000012		

9976  
9977

.EVEN



9978  
 9979  
 9980  
 9981  
 9982  
 9983  
 9984  
 9985  
 9986  
 9987 050562 010046  
 9988 050564 010146  
 9989 050566 013746 000004  
 9990 050572 013746 000006  
 9991 050576 010600  
 9992  
 9993 050600 104400  
 9994 050602 012637 000006  
 9995 050606 012737 050626 000004  
 9996 050614 012701 020000  
 9997 050620 005711  
 9998 050622 005721  
 9999 050624 000775  
 10000 050626 162701 000002  
 10001 050632 010006  
 10002 050634 012637 000006  
 10003 050640 012637 000004  
 10004 050644 010137 050656  
 10005 050650 012601  
 10006 050652 012600  
 10007 050654 000207  
 10008 050656 000000  
 10009  
 10010  
 10011  
 10012  
 10013  
 10014  
 10015  
 10016  
 10017  
 10018  
 10019  
 10020  
 10021  
 10022 050660 005737 001226  
 10023 050664 001433  
 10024 050666 005037 001226  
 10025 050672 012700 001366  
 10026 050676 012703 051046  
 10027 050702 011004  
 10028 050704 004737 051070  
 10029 050710 004037 031634  
 10030 050714 000402  
 10031 050716 000765  
 10032 050720 000414  
 10033 050722 010420

```

.SBTTL ROUTINE TO SIZE MEMORY
;*****
;CALL:
;* JSR PC,$SIZE
;* RETURN
;*$LSTAD WILL CONTAIN THE LAST AVAILABLE MEMORY LOCATION

$SIZE: MOV RO,-(SP) ;;SAVE RO ON THE STACK
        MOV R1,-(SP) ;;SAVE R1 ON THE STACK
        MOV @#ERRVEC,-(SP) ;;SAVE PRESENT ERROR VECTOR PS & PC
        MOV @#ERRVEC+2,-(SP)
        MOV SP,RO ;;SAVE THE STACK POINTER
;;SET THE ERRVEC PS TO THE PRESENT PS
        TRAP ;;PUSH OLD PSW AND PC ON STACK
        MOV (SP)+,@#ERRVEC+2 ;;SAVE THE PSW IN @#ERRVEC+2
        MOV #2,@#ERRVEC ;;SET FOR TIMEOUT
        MOV #20000,R1 ;;FIRST ADDRESS
1$: TST (R1) ;;TEST THIS ADDRESS
    TST (R1)+ ;;STEP TO NEXT ADDRESS
    BR 1$ ;;TRY ANOTHER
2$: SUB #2,R1 ;;DROP BACK
    MOV RO,SP ;;RESTORE THE STACK
    MOV (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
    MOV (SP)+,@#ERRVEC
    MOV R1,$LSTAD ;;LAST ADDRESS
    MOV (SP)+,R1 ;;RESTORE R1
    MOV (SP)+,RO ;;RESTORE RO
    RTS PC
$LSTAD: .WORD 0 ;;CONTAINS THE LAST ADDRESS
;*****
.SBTTL GETADR - GET BUS ADDRESS AND VECTOR ADDRESS
;THIS ROUTINE IS USED TO ENSURE THE BUS ADDRESS
;OF THE RH11/RP04 IS SETUP TO READ THE PROPER VALUE.
;IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
;REQUIRED.
;NOTE: THIS ROUTINE DESTROYS RO-R4
;CALL
;
; JSR PC,@#GETADR
; RETURN

GETADR: TST @#BUSADR ;;INPUT FROM TTY REQUESTED?
        BEQ 7$ ;;NO--BRANCH
        CLR @#BUSADR ;;YES--CLEAR THE REQUEST FLAG
1$: MOV #RH.ADR,RO ;;FIRST ADDRESS
    MOV #MRPCS1,R3 ;;"RPCS1="
    MOV (RO),R4 ;;PRESENT RPCS1 ADDRESS
    JSR PC,CLRF ;;CLEAR INPUT BUFFER
    JSR RO,@#GETNUM ;;GET NEW RPCS1
    BR 2$ ;;COMMA
    BR 1$ ;;PERIOD
    BR 5$ ;;DOUBLE PERIOD
2$: MOV R4,(RO)+ ;;SAVE NEW RPCS1
    
```

```

10034 050724 012703 051057      MOV      #MRHVEC,R5      ;"RHVEC="
10035 050730 011004      MOV      (R0),R4        ;PRESENT RH11 VECTOR ADDRESS
10036 050732 004737 051070      JSR      PC,CLRBF       ;CLEAR INPUT BUFFER
10037 050736 004037 031634      JSR      R0,@#GETNUM    ;GET NEW RHVEC
10038 050742 000402      BR       3$            ;COMMA
10039 050744 000752      BR       1$            ;PERIOD
10040 050746 000401      BR       5$            ;DOUBLE PERIOD
10041 050750 010420      3$:     MOV      R4,(R0)+  ;SAVE NEW RHVEC
10042 050752 010410      5$:     MOV      R4,(R0)   ;SAVE INPUT
10043 050754 013701 000004      7$:     MOV      @#ERRVEC,R1 ;SAVE THE ERROR VECTOR
10044 050760 012737 051014 000004      MOV      #8$,@#ERRVEC   ;SETUP FOR TRAP
10045 050766 005777 130374      TST      @RH.ADR        ;CHECK FOR RH11/RP04
10046 050772 010137 000004      MOV      R1,@#ERRVEC    ;RESTORE ERROR VECTOR
10047 050776 012700 001366      MOV      #RH.ADR,R0     ;FIRST ADDRESS OF NEW PARAMETERS
10048 051002 012701 034632      MOV      #RPADR,R1      ;FIRST ADDRESS OF WHERE TO PUT THEM
10049 051006 012021      MOV      (R0)+,(R1)+    ;BUS ADDRESS
10050 051010 012021      MOV      (R0)+,(R1)+    ;VECTOR ADDRESS
10051 051012 000207      RTS      PC             ;RETURN
10052 051014 010137 000004      8$:     MOV      R1,@#ERRVEC ;RESTORE ERROR VECTOR
10053 051020 022626      CMP      (SP)+,(SP)+    ;CLEAN OFF THE STACK
10054 051022 104010      ERROR   10             ;REPORT THE ERROR
10055 051024 005737 000042      TST      @#42           ;IS THERE A MONITOR?
10056 051030 001720      BEQ     1$             ;NO--GO ASK FOR ADDRESS
10057 051032 005037 001232      CLR     @#DRVSEL       ;YES--NO DRIVES SELECTED
10058 051036 005037 017656      CLR     @#SEOPCT       ;NO PASSES
10059 051042 000137 017502      JMP     @#SEOP          ;GO TO END OF PROGRAM
10060
10061 051046 005015 050122 051503 MRPCS1: .ASCIZ <CR><LF>/RPCS1=/
10062 051054 036461 000
10063 051057 015 051012 053110 MRHVEC: .ASCIZ <CR><LF>/RHVEC=/
10064 051064 041505 000075
10065
10066 ;*****
10067 ;ROUTINE TO CLEAR INPUT BUFFER FOR NEW CS1 AND VEC
10068 ;
10069 CLRBF: MOV      R0,-(SP)   ;SAVE R0
10070      MOV      R1,-(SP)
10071      MOV      #10,R1     ;COUNT 10 LOCATIONS
10072      MOV      #STYIN,R0  ;START ADR OF INPUT BUF
10073      1$:     CLR      (R0)+
10074      DEC     R1
10075      BNE    1$
10076      MOV      (SP)+,R1
10077      MOV      (SP)+,R0   ;RESTORE R0
10078      RTS     PC
10079      .END

```

ABS =	000200	1834#												
ACL =	000040	1868#	1877#											
ACTDRV	034550	8009#	8304*	8355*	8668*	8677*	8907							
ACTSTR	034551	8015#	8909*	8924*										
ACU =	100000	1823#												
AOE =	001000	1740#												
ATA =	100000	1727#												
ATABIT	034620	3335	7355	7361	8081#	8260	8367	8472	8831	8850	8858	8876	8983	
ATO =	000001	1760#												
AT1 =	000002	1761#												
AT2 =	000004	1762#												
AT3 =	000010	1763#												
AT4 =	000020	1764#												
AT5 =	000040	1765#												
AT6 =	000100	1766#												
AT7 =	000200	1767#												
A16 =	000400	1658#												
A17 =	001000	1659#												
BA1 =	000010	1676#												
BITS	001424	2041#	3496	3530	3568	3611	3652	3732	3775	3825	3925	4082	4148	4252
		4342	4440	4554	4625	4754	4800	4874	4963	7422	7453	7470	7523	7697
BIT0 =	000001	1633#												
BIT00 =	000001	1623#	1633	2041	2057	3306	3323	3345	3398	6169				
BIT01 =	000002	1622#	1632	2042	2058	3309	6495	6523	8347	8594				
BIT02 =	000004	1621#	1631	2043	2059	6232	6495	6526						
BIT03 =	000010	1620#	1630	2044	2060	3295	6532	8245	8790	9105				
BIT04 =	000020	1619#	1629	2045	2061	6529	8825							
BIT05 =	000040	1618#	1628	2046	2062	3968	3984	4060	4190	4204	4219	4224	4289	4313
		4318	4377	4395	4412	4417	4475	4493	4510	4515	6532	6642	8171	8614
		8805	8957											
BIT06 =	000100	1617#	1627	2047	2063	6535	8269	8338	8412	8703	9166			
BIT07 =	000200	1616#	1626	2048	2064	8269	8546	8689	8790	8805	8825	8854	9138	
BIT08 =	000400	1615#	1625	2049	6523	8269	8970							
BIT09 =	001000	1614#	1624	2050	5090	5715	6538	8243	8955					
BIT1 =	000002	1632#												
BIT10 =	002000	1613#	2051	5074	6181	6495	6526	7716	8596					
BIT11 =	004000	1612#	2052	5722	6526	8218	8371	8389	8565	8877				
BIT12 =	010000	1611#	2053	6495	6529	8213	8251	8269	8349	8383	8561	8580	8592	8605
		8815	8817	9043	9100	9167								
BIT13 =	020000	1610#	2054	5081	6495	6523	8332	8763	9046					
BIT14 =	040000	1609#	2055	3979	4185	4199	4284	4372	4390	4470	4488	5697	6423	6495
		6529	8344	8380	8750	8894	9010	9055	9059	9164				
BIT15 =	100000	1608#	2056	6687	6688	8332	8344	8347	8349	8380	8383	8565	8594	8596
		8703	8790	8805	8815	8825	8894	8955	8970	9010	9017			
BIT2 =	000004	1631#	9017											
BIT3 =	000010	1630#												
BIT4 =	000020	1629#												
BIT5 =	000040	1628#												
BIT6 =	000100	1627#	8373											
BIT7 =	000200	1626#												
BIT8 =	000400	1625#												
BIT9 =	001000	1624#												
BPTVEC =	000014	1640#												
BUFFER =	050202	2700	2724	2748	2772	3945	4572	4587	4600	4643	4653	4667	4788	4890
		4914	4922	4930	4936	4944	6606*	6607	6609	6611	6612	6613	6686	6723
		6866	6885	6886	6910	6999	7031	7074	7116	7133	7134	7151	9932#	



DFLT	001664	2133#	6158											
DF1	050026	2861	9831#											
DF10	050046	2925	9847#											
DF11	050052	2935	9851#											
DF12	050056	2948	9855#											
DF13	050066	2961	2982	9862#										
DF14	050076	2969	2990	9869#										
DF17	050102	3000	3010	9873#										
DF2	050032	2871	2901	9835#										
DF21	050106	3023	9877#											
DF22	050116	3031	9884#											
DF23	050122	3043	3055	9888#										
DF3	050036	2881	9839#											
DF4	050042	2891	9843#											
DF41	050132	3088	9896#											
DF42	050136	3098	9900#											
DF43	050142	3110	9904#											
DF44	050152	3125	9911#											
DF45	050166	3140	3155	9921#										
DH1	046013	2859	9630#											
DH10	046222	2923	9654#											
DH11	046241	2933	9657#											
DH12	046260	2946	2959	2980	3123	3138	3153	9660#						
DH12A	046347	9670#	9858											
DH13A	046426	9678#	9865											
DH17	046474	2998	3008	9685#										
DH2	046030	2869	2899	9633#										
DH21	046572	3021	9696#											
DH21A	046650	9704#	9880											
DH23	046707	3041	3053	9710#										
DH23A	046775	9720#	9891											
DH3	046116	2879	9642#											
DH4	046153	2889	9647#											
DH41	047042	3086	9727#											
DH42	047100	3096	3108	9732#										
DH43A	047166	9741#	9907											
DH44A	047214	9745#	9914	9924										
DH44B	047271	9753#	9917											
DH45A	047317	9757#	9927											
DIGB =	000004	1714#												
DISPLA	001142	1946#	3207*	3215*	3503*	3537*	3575*	3618*	3659*	3739*	3782*	3832*	3932*	4089*
		4155*	4259*	4349*	4447*	4561*	4632*	4761*	4881*	4970*	5073*	5736*		
DISPRE	000174	1509#	3215											
DLT =	100000	1688#												
DL64 =	000020	1716#												
DMD =	000001	1750#												
DORT1	027170	3966	4179	4277	4365	4463	6673#							
DPB.A	004104	2695#	3391*	3397*	3400*	3401*	3403*	3506*	3836*	3854*	3861	3937*	3940*	3949
		3970	3971	3981	4042*	4049*	4051	4053	4057	4975*	4976*	4980*	4981*	6277
		6279	6283	6284	6285	6286	6294	6620*						
DPB.B	004124	2719#	3392*	3507*	3508*	3509*	3540*	3542*	3544*	3579*	3580*	3582*	3587*	3588
		3590*	3595*	3596	3622*	3623*	3625*	3632*	3663*	3664*	3674*	3680*	3696*	3702*
		3743*	3744*	3750*	3752*	3786*	3787*	3796*	3798*	3800*	3802*	3804*	3806*	3835*
		3837*	3842*	3852*	3854	3881*	3882	3884*	4093*	4095*	4097*	4108*	4109	6261*
		6264*	6308	6310	6314	6315	6316	6317	6329	6332	6335			
DPB.C	004144	2743#	3393*	3541*	3543*	3545*	4094*	4096*	4098*	4104*	4105	4107*	6262*	6265*











MSGB03	045635	2827	9607#																		
MSGB04	045615	2826	9604#																		
MSGB05	045545	2825	9597#																		
MSGB06	045501	2824	9590#																		
MSGB08	045437	2823	9584#																		
MSGB09	045400	2822	9578#																		
MSGB10	045355	2821	9574#																		
MSGB11	045327	2820	9570#																		
MSGB12	045305	2819	9566#																		
MSGB13	045254	2818	9561#																		
MSGB14	045212	2817	9555#																		
MSGFCP	043156	2122	9341#																		
MSG LCP	043162	2123	9342#																		
MSGMAX	044177	6772	9453#																		
MSGMIN	044170	6756	9451#																		
MSGNON	044315	6809	9469#																		
MSGNUM	044300	6804	9466#																		
MSG.CS	043216	7257	9351#																		
MSG.EQ	043214	7559	7597	7638	9350#																
MSG.FC	043150	2120	9339#																		
MSG.FS	043202	2128	9347#																		
MSG.FT	043171	2125	9344#																		
MSG.IC	043166	2124	9343#																		
MSG.IT	043177	2127	9346#																		
MSG.LC	043153	2121	9340#																		
MSG.LS	043205	2129	9348#																		
MSG.LT	043174	2126	9345#																		
MSG.PA	043210	7596	9349#																		
MSG.R	043146	2119	9338#																		
MSG.SP	044330	3283	3284	3304	3410	5158	5170	5197	5204	6763	6779	6800	7555	7595							
		7600	9471#																		
MSGOUS	044215	6760	6771	6776	6787	6799	9457#														
MSG10A	043742	2385	9421#																		
MSG10B	044011	2386	9428#																		
MSG11A	044026	2390	9431#																		
MSG11B	044074	2391	9438#																		
MSG12A	044111	2395	9441#																		
MSG12B	044153	2396	9447#																		
MSG7	043710	2375	2380	9416#																	
MSTCK =	000010	1753#																			
MWR =	000040	1755#																			
MXDLTA	034642	8106#	8655																		
MXF =	001000	1682#																			
MXLACT	034640	8103#	8642																		
MXSTAL	001362	2021#	3687	3709																	
MXWNDW	034646	8112#	8460																		
NBA =	100000	1786#																			
NC1	001532	2079#	3942*	4030*	4031	4034	4042														
NC2	001534	2080#																			
NED =	010000	1685#																			
NEM =	004000	1684#																			
NMS =	002000	1819#	1837#																		
NOCLOK	043561	3371	9398#																		
NOLQAD	050464	3246	9965#																		
NONE	043552	3351	9396#																		
NOOP	000101	1892#																			

100000

NOTPRS	043437	3293	9380#									
NOTRP	043464	3291	9385#									
NOTSAF	043454	3301	9383#									
MOUSE	043317	3297	9363#									
OCYL =	100000	1871#	1881#									
OFFSET =	000115	1898#										
OFREV =	000200	1849#										
OF100 =	000004	1845#										
OF200 =	000010	1846#										
OF25 =	000001	1843#										
OF400 =	000020	1847#										
CF50 =	000002	1844#										
OF800 =	000040	1848#										
OPE =	020000	1879#										
OPI =	020000	1744#										
OPNFLG	001240	1982#	7380*	7381*	7470*	7479	7481	7490	7492	7499	7501	7523
OPNPAT	033336	7554	7595#	7604	7613	7616	7620					
OPNPRM	033134	7541	7546#	7550	7605	7607	7626					
OPNTST	032762	7480	7482	7491	7493	7500	7502	7507#				
OPNWDS	033460	7628#	7662									
OPN.CT	032766	7509#	7511*	7512	7516	7518	7529					
OPN.N1	033672	7591	7622	7685#								
OPN.N2	033676	7578	7611	7673	7686#							
OPN.X1	033710	7592	7623	7689#								
OPN.X2	033714	7583	7615	7677	7690#							
OPN.1	032772	7511#	7524	7688								
OPN.2	033014	7510	7513	7516#								
OPT	035742	8330	8365#	8699	8859	8899						
OR =	000200	1680#										
PACK =	000123	1901#										
PAR =	000010	1734#										
PAT	001530	2078#	4800									
PAT.PT	003044	2400#	7001	7034	7077							
PAT0	003104	2400	2419#	6165	7629	7678						
PAT1	003144	2401	2436#									
PAT10	003604	2410	2589#									
PAT11	003644	2411	2606#									
PAT12	003704	2412	2623#									
PAT13	003744	2413	2640#									
PAT14	004004	2414	2657#									
PAT15	004044	2415	2676#									
PAT2	003204	2402	2453#									
PAT3	003244	2403	2470#									
PAT4	003304	2404	2487#									
PAT5	003344	2405	2504#									
PAT6	003404	2406	2521#									
PAT7	003444	2407	2538#									
PAT8	003504	2408	2555#	6164								
PAT9	003544	2409	2572#	6167								
PGE =	002000	1683#										
PGM =	001000	1721#										
PIP =	020000	1725#										
PIRQ =	177772	1553#										
PIRQVE =	000240	1647#										
PKB	001402	2031#	4180*	4194*	4278*	4366*	4384*	4464*	4482*	6128*	6590*	
PKC	001404	2032#	6706	6708	6709	6713	6715	6716	6720	6725		

PKCS	001400	2030#	4181*	4184*	4196*	4198*	4212*	4281*	4283*	4306*	4369*	4371*	4387*	4389*
		4405*	4467*	4469*	4485*	4487*	4503*	6100	6129*	6591*				
PKV	001374	2029#	3960	3961	4178*	4276*	4364*	4462*	6126*	6127*	6588	6589*	6594*	
PLU =	020000	1822#	1839#											
POPQUE	043060	8379	8430	8545	8853	9310#								
PRM	001504	2068#	6218	6222*	6235*	6238*	7537	7546	7701					
PRMLMT	001606	2105#	3871	4004	4605	4770	4781	6171*	6172*	6175*	6176*	6192	6194	6854
		6872	6951	7117	7203	7553	7586							
PRMMSG	001636	2119#	7556											
PRMP1	001536	2083#	6178	6217	7535	7702								
PRM0	002330	2083	2188#	6159										
PRM1	002344	2084	2196#											
PRM10	002544	2091	2274#											
PRM11	002564	2092	2284#											
PRM12	002606	2093	2295#											
PRM13	002622	2094	2303#											
PRM14	002636	2095	2311#											
PRM15	002652	2096	2319#											
PRM16	002666	2097	2327#											
PRM17	002700	2098	2334#											
PRM2	002372	2085	2209#											
PRM20	002712	2099	2341#											
PRM21	002744	2100	2356#											
PRM22	002760	2101	2364#											
PRM3	002414	2086	2220#											
PRM4	002436	2087	2231#											
PRM5	002460	2088	2242#											
PRM6	002502	2089	2253#											
PRM7	002524	2090	2264#											
PR0 =	000000	1570#												
PR1 =	000040	1571#												
PR2 =	000100	1572#												
PR3 =	000140	1573#												
PR4 =	000200	1574#												
PR5 =	000240	1575#												
PR6 =	000300	1576#												
PR7 =	000340	1577#	3273	3956	3978									
PS =	177776	1550#	1551	3263*	3273*	3318*	3956*	3973*	3978*	4025*	4213*	4307*	4406*	4504*
		6586	6587*	6595*	6639*	8151	8152*	8181*	8189*	8302	8303*	8356*	8366	8415*
		8648*												
PSEL =	002000	1660#												
PSU =	000001	1865#												
PSW =	177776	1551#												
PTRN15	002742	2353#												
PWRVEC =	000024	1642#												
QCNT	042372	9176#	9235	9259*	9274	9276*	9296	9310*						
QDRV0	042464	9187	9198	9207	9219#									
QDRV1	042504	9188	9199	9208	9220#									
QDRV2	042524	9189	9200	9209	9221#									
QDRV3	042544	9190	9201	9210	9222#									
QDRV4	042564	9191	9202	9211	9223#									
QDRV5	042604	9192	9203	9212	9224#									
QDRV6	042624	9193	9204	9213	9225#									
QDRV7	042644	9194	9205	9214	9226#									
QINPT	042402	9187#	9261	9278*	9279*	9280	9282*							
QOUTPT	042422	9198#	9261*	9299	9312	9313*	9314	9316*						













T7A	002774	2375#	4171	4231										
T7B	003004	2380#	4174	4235										
ULDFLG	034552	8022#	8210*	8312	8327*	8509*	8572*	8604*	8742	8828	8830*	8846	8848*	8950*
		8972*												
UNLOAD=	000103	1893#												
UNS	= 040000	1745#												
UNMSG	043410	9375#												
UNTOFF	043416	3299	9376#											
UNTON	043427	3303	9378#											
UPE	= 020000	1686#												
USE	043242	3233	9355#											
US1	= 000001	1673#												
US2	= 000002	1674#												
US4	= 000004	1675#												
UWR	= 000010	1867#												
VERIFY	026644	6334	6377	6604#										
VUF	= 000002	1866#												
VU30	= 010000	1821#												
VV	= 000100	1718#												
WAO	= 000002	1876#												
WCE	= 040000	1687#												
WCEFLG	001334	2011#	4791*	4828	6396*	6460*								
WCF	= 000040	1736#												
WCU	= 000001	1809#	1827#											
WLF	= 004000	1742#												
WRCKD	= 000151	1903#	4577	4598	4657	4680	4826	4929	4943	6421	6447			
WRCKHD	= 000153	1904#												
WRITE	= 000161	1905#	3938	4575	4591	4639	4674	4820	4891	4913	6427			
WRL	= 004000	1723#												
WRTHD	= 000163	1906#												
WRT.AD	042100	9084*	9094*	9097#	9777									
WRT.RP	042010	8248	8252	8386	8440	8444	8448	8456	8465	8469	8480	8487	8497	8511
		8521	8542	8557	8568	8766	8798	8835	9081#					
WRT.R1	042074	9095#	9115											
WRT.R2	042164	9081*	9114#											
WRT.R3	042172	9090	9101	9104	9110	9117#								
WRT.R4	042176	9106	9119#											
WRT.R5	042200	9118	9120#											
WRT.WD	042076	9082*	9086	9093*	9096#	9777								
WRU	= 000400	1817#	1835#											
WSU	= 000004	1811#	1829#											
\$AUTOB	001134	1942#	3256*	5523	5680									
\$BDADR	001122	1937#	7081*	7146*	9792									
\$BDDAT	001126	1939#	6970*	7083*	7148*	9792	9799							
\$BELL	001210	1968#	3220	5076	5103	5476	5673							
\$CHARC	020774	5245*	5255*	5262	5271*	5276#								
\$CKSWR	021750	5509#	5826											
\$CMTAG	001100	1925#	3182	3183	3191	3197	3198							
\$CM1	= 000006	1957#	1958#	1959#	1960#	1961#	1962#	1963#						
\$CM2	= 000014	1957#	1958#	1959#	1960#	1961#	1962#	1963#						
\$CM3	= 000006	1955#	1957											
\$CM4	= 000003	1963#	1964#	1965#	1966#									
\$CNTLC	022662	5464	5545	5673#										
\$CNTLG	022674	5530	5675#											
\$CNTLU	022667	5555	5648	5674#										
\$CRLF	001215	1970#	3261	3275	3314	3429	5084	5103	5134	5153	5167	5177	5194	5201



\$PASS	001100	1926#	5021*	5022*	5044	5724	5740								
\$QUES	001214	1969#	5103	5279	5585	5656	5673								
\$RAND	023626	3845	4043	5949#	6564	7182	7197								
\$RDCHR	022312	5598#	5827												
\$RDDEC=	***** U	5829													
\$RDLIN	022402	5621#	5828												
\$RDOCT=	***** U	5829													
\$RDSZ =	000024	5614#													
\$REGAD	001160	1955#	3218												
\$REGO	001162	1957#	5117*	9784	9789	9797	9806	9808	9811	9815	9821				
\$REG1	001164	1958#	5118*	9772	9779	9799									
\$REG2	001166	1959#	5119*	9783											
\$REG3	001170	1960#	5120*	9771	9772										
\$REG4	001172	1961#	5121*	9792	9799										
\$REG5	001174	1962#	5122*	9779											
\$RESRE	023214	5774#	5830												
\$RTNAD	017744	5043#													
\$R2A =	***** U	5831													
\$SAVRE	023156	5758#	5829												
\$SB2D	023336	5843#	6758	6765	6769	6774	6781	6785	6797	6802	7561	7636			
\$SCOPE	022724	3189	5695#												
\$SETUP=	000147	3161#	3188	3189	3191	3193	3195	3197	3198	3200	3249	5019	5064	5089	
		5097	5468	5473	5474	5504	5680	5696							
		3243	9987#												
\$SIZE	050562														
\$STUP =	177777	3161#													
\$SUPRS	023566	5924#	6759	6766	6770	6775	6782	6786	6798	6803	7562	7637			
\$SVLAD	023112	5706	5731#												
\$SVPC =	000200	1516#	1521												
\$SWR =	167000	1449#	1459	1487	1488	1489	1490	1491	1492	1918#	1966	1967	1968	3197	
		3198	3200	3201	3506	3540	3578	3621	3662	3742	3785	3835	3935	4092	
		4158	4262	4352	4450	4564	4635	4764	4884	4973	4994	5020	5036	5042	
		5044	5055	5056	5057	5058	5059	5074	5081	5086	5090	5103	5688	5689	
		5690	5691	5692	5697	5709	5711	5712	5713	5720	5721	5722	5733	5736	
		5739													
\$SWRMK=	000000	5692													
\$TIMES	001204	1966#	3197*	3504*	3538*	3576*	3619*	3660*	3740*	3783*	3833*	3933*	4090*	4156*	
		4260*	4350*	4448*	4562*	4633*	4762*	4882*	4971*	5020*	5720*	5727	5730*	5739	
		1948#	5429	5449	5460	5485	5513	5540							
\$TKB	001146	5430#	5444*	5474	5491*	5605	5607*								
\$TKCNT	021452	3248	5444#	5465	5526										
\$TKINT	021462	5434#	5499	5610											
\$TKQEN=	021462	5431#	5445*	5446	5497*	5498*	5499	5501*							
\$TKQIN	021454	5432#	5446*	5608	5609*	5610	5612*								
\$TKQOU	021456	5433#	5445	5501	5612										
\$TKQSR	021460	1947#	5429	5450*	5481*	5483	5489*	5511	5527*	5537	5549*	5569*			
\$TKS	001144	5447	5460#												
\$TKSRV	021532	1963#	5114*	9775	9777	9779	9784	9789	9794	9797	9801	9806	9808	9811	
\$TMPO	001176	9815	9821												
		1964#													
\$TMP1	001200	1965#													
\$TMP2	001202	1449#	1459	3485	3494	3496	3505	3506#	3517	3528	3530	3539	3540#	3553	
\$TN =	000023	3566	3568	3577	3578#	3600	3609	3611	3620	3621#	3639	3650	3652	3661	
		3662#	3717	3730	3732	3741	3742#	3762	3773	3775	3784	3785#	3812	3823	
		3825	3854	3835#	3888	3923	3925	3934	3935#	4067	4080	4082	4091	4092#	
		4134	4146	4148	4157	4158#	4239	4250	4252	4261	4262#	4327	4340	4342	
		4351	4352#	4425	4438	4440	4449	4450#	4539	4552	4554	4563	4564#	4609	

	4623	4625	4634	4635#	4690	4752	4754	4763	4764#	4847	4872	4874	4883
\$TNPWR 023502	4884#	4952	4961	4963	4972	4973#	7413	7437	7512				
\$TPB 001152	5871	5872	5892#										
\$TPFLG 001157	1950#	3223*	5070*	5101*	5268*	5279							
\$TPS 001150	1954#	5226	5279										
\$TRAP 023252	1949#	3222*	5069*	5100*	5266	5279							
\$TRAP2 023274	3193	5795#											
\$TRP = 000014	5806#	5817											
	5810#	5819#	5820#	5821#	5822#	5823#	5824	5825#	5826	5827#	5828#	5829#	5830#
	5831#												
\$TRPAD 023306	5800	5817#											
\$STNM 001102	1927#	3499*	3503	3533*	3537	3571*	3575	3614*	3618	3655*	3659	3735*	3739
	3778*	3782	3828*	3832	3928*	3932	4085*	4089	4151*	4155	4255*	4259	4345*
	4349	4443*	4447	4557*	4561	4628*	4632	4757*	4761	4877*	4881	4966*	4970
	5019*	5073	5103	5114	5687	5731*	5736	5740	6215	6417			
	5623	5624	5636	5654	5668	5672#	10072						
\$TTYIN 022636	5823												
\$TYPBN= ***** U	5370#	5822											
\$TYPDS 021226	5226#	5810	5818										
\$TYPE 020560	5247	5254	5261	5266#	5267	5571							
\$TYPEC 020730	5272	5274	5277#										
\$TYPEX 020776	5310#	5819											
\$TYPOC 021024	5309	5312#	5821										
\$TYPON 021040	5305#	5820											
\$TYPOS 021000	5700#												
\$XTSTR 022736	5036#												
\$GET4= 000000	5306*	5310*	5320	5355#									
\$OFILL 021223	5083	5697											
\$LOCAT= ***** U	1504#	1508#	1516	1517#	1519#	1521#	1524#	1924#	1972	3186	3200	3201	3583
.	3591	3626	3630	3672	3694	3748	3792	3856	3859	3954	3992	4573	4578
	4581	4589	4596	4645	4655	4672	4678	4764	4818	4824	4834	4884	4896
	4917	4923	4931	4937	4945	5001#	5032#	5044	5045#	5103	5147#	5279	5424#
	5429	5433#	5434	5435	5672#	5673	5680#	5739	5740	5912#	7528#	9219#	922C#
	9221#	9222#	9223#	9224#	9225#	9226#	9227	9932					

CKCHR	3157#	7289	7339	7566	7603	7643	7806	7819	7830	7881					
CKDIG	3157#	7347	7585												
CKNUM	3157#	7297	7618	7651											
COMMEN	1#	1648#	3434	3469	4116	4525									
COMND	3157#	3403	3506	3836	3936	4976	4981								
DO	3157#	3402	3404	3513	3514	3549	3550	3586	3594	3628	3633	3675	3681	3697	3703
	3751	3753	3797	3799	3801	3803	3805	3807	3858	3879	4101	4103	4977	4982	6621
DODTA	3157#	4576	4580	4585	4595	4602	4648	4659	4675	4681	4821	4827	4837	4898	4919
	4925	4933	4939	4947											
DRV.IN	3157#	4223	4317	4416	4514										
ENDCOM	1#	1648#	3462	3480	4129	4534									
ENDPAS	3157#	5029													
ERRCAL	7901#	8719	8770	8773	9050	9111									
ERREND	3157#	5097													
ERROR	1542#	3388	3983	4059	4193	4207	4222	4292	4316	4380	4398	4415	4478	4496	4513
	6289	6290	6291	6292	6293	6320	6321	6322	6323	6328	6363	6364	6365	6366	6371
	6412	6413	6414	6415	6416	6442	6443	6444	6445	6446	6468	6469	6470	6471	6472
	6619	6660	6661	6662	6663	6664	6976	6981	7092	7098	7156	7157	8720	8771	8775
	9052	9112	10054												
ERRTYP	3157#	5065													
ER.NDX	3157#	6283	6314	6357	6406	6436	6462	6654							
ESCAPE	1#	1648#	4169	4270	4360	4458	6282	6313	6356	6404	6435	6461	6617	6653	6975
	6979	7091	7096	7144											
GETPRI	1#	1648#	6000	9993											
GETSWR	1#	1449#	1648#	3249											
LOOP	3157#	3583	3591	3626	3630	3672	3694	3748	3792	3855	3859	3953	3991	4573	4578
	4581	4589	4596	4645	4655	4672	4678	4818	4824	4834	4896	4917	4923	4931	4937
	4945														
MORETA	1918#	1973													
MORE.S	3157#	3494	3528	3566	3609	3650	3730	3773	3823	3923	4080	4146	4250	4340	4438
	4552	4623	4752	4872	4961										
MSG	3485#	3487	3517#	3519	3553#	3555	3600#	3602	3639#	3641	3717#	3719	3762#	3764	3812#
	3814	3887#	3890	4067#	4069	4133#	4136	4239#	4241	4327#	4329	4425#	4427	4539#	4541
	4609#	4611	4690#	4692	4847#	4849	4952#	4954							
MULT	1#	1648#													
NEWST	1#	1648#	3485	3517	3553	3600	3639	3717	3762	3812	3888	4067	4134	4239	4327
	4425	4539	4609	4690	4847	4952									
POP	1#	1648#	5411	5779	5968	6051	6196	6245							
PUSH	1#	1648#	5370	5759	5949	6003	6151	6209							
REPORT	1#	1648#	3157#	4230	4233	4322	4421	4519							
RPO4.D	2#	7901													
SAV.RH	3157#	4187	4201	4216	4286	4310	4374	4392	4409	4472	4490	4507			
SCOPE	1543#	3515	3551	3598	3637	3715	3760	3810	3885	4065	4111	4236	4324	4423	4521
	4607	4687	4844	4948	4985										
SETPRI	1#	1648#	5601												
SETTRA	5810#	5819	5820	5821	5822	5824	5826	5827	5828	5829	5830				
SETUP	1#	1648#	3181												
SET.TN	3157#	3495	3529	3567	3610	3651	3731	3774	3824	3924	4081	4147	4251	4341	4439
	4553	4624	4753	4873	4962										
SKIP	1#	1648#													
SLASH	1#	1648#													
SPACE	1648#														
STARS	1#	1514	1648#	1652	1694	1698	1889	1920	1972	3059	3485	3493	3517	3527	3553
	3565	3600	3608	3639	3649	3717	3729	3762	3772	3812	3822	3888	3922	4067	4079
	4134	4145	4239	4249	4327	4339	4425	4437	4539	4551	4609	4622	4690	4751	4847
	4871	4952	4960	4991	5051	5104	5211	5282	5360	5428	5504	5519	5590	5614	5684



.S40CA 1#  
.1170 1#

. ABS. 051120 000

ERRORS DETECTED: 0

DSKZ:CZRJAD.BIN,DSKZ:CZRJAD.LST/CRF/SOL/NL:TOC:MD:MC:CND/LI:ME=CZRJAD.SML,CZRJAD.010,CZRJAD.P11  
RUN-TIME: 77 108 7 SECONDS  
RUN-TIME RATIO: 429/194=2.2  
CORE USED: 52K (103 PAGES)