RP11E

DRIVE POSITIONING TEST

MD-11-DZRPZ-C

EP-DZRPZ-C-DL -C

COPYRIGHT © 75-77

FICHE 1 OF 1

DEC 1977

digital

MADE IN USA

.REM @

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

PRODUCT CODE: MAINDEC-11-DZRPZ-C-D

PRODUCT NAME: RP11E DRIVE POSITIONING TEST

DATE CREATED: SEPTEMBER 1977

MAINTAINER: DIAGNOSTIC GROUP

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.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

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

COPYRIGHT (C) 1975, 1977 BY DIGITAL EQUIPMENT CORPORATION.

CONTENTS

1. ABSTRACT

2. REQUIREMENTS

2.1 EQUIPMENT
2.2 STORAGE
2.3 PRELIMINARY PROGRAMS

3. LOADING PROCEDURE

4. STARTING PROCEDURE

4.1 STARTING ADDRESS
4.2 OPERATOR ACTION
4.3 PROGRAM ACTION
4.4 'CONTROL C' OPERATION

5. OPERATING PROCEDURE

SOFTWARE SWITCH REGISTER
5.1 OPERATIONAL SWITCH SETTINGS

6. ERRORS

6.1 ERROR TYPES
6.2 ERROR RECOVERY

7. RESTRICTIONS

7.1 SYSTEMS WITH MIXED DRIVE TYPES
7.2 FORMATTED DISK PACKS

8. MISCELLANEOUS

8.1 EXECUTION TIME
8.2 STACK POINTER
8.3 TIMING TEST (TEST10-TEST12) PRINTOUTS
8.4 END OF TEST
8.5 DEFAULT ADDRESSES, TEST SEQUENCE, AND PARAMETERS
8.6 TIMING TEST RESTRICTIONS
8.7 DISK POSITIONING CHECK

9. PROGRAM DESCRIPTION

9.1 TEST DESCRIPTIONS

10. PROGRAM LISTING

1. ABSTRACT

THIS PROGRAM CONTAINS A SERIES OF TESTS THAT WILL ENSURE THAT THE DISK DRIVE IS CAPABLE OF PERFORMING SEEKS, THAT THE ACCESS TIMES ARE WITHIN TOLERANCE, AND THAT THE CYLINDER ADDRESSING CAPABILITY OF THE DRIVE IS FUNCTIONING PROPERLY.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 COMPUTER WITH 8K OF MEMORY (WITH OR WITHOUT HARDWARE SWITCH REGISTER); CONSOLE TELETYPE; RP11E DISK CONTROLLER;
1 TO 8 RP02, RPR02, OR RP03 DISK DRIVES WITH FORMATTED PACKS.

2.2 STORAGE

THIS PROGRAM WILL LOAD AND RUN IN 8K.

2.3 PRELIMINARY PROGRAMS

MAINDEC-11-DZRPW - RP11E DISKLESS LOGIC TEST

3. LOADING PROCEDURE

*** BEFORE STARTING, REFER TO SECTION 5.***
THE PROGRAM MAY BE LOADED FROM EITHER PAPER TAPE OR AN 'XXDP' DEVICE. REFER TO EITHER THE STANDARD PROCEDURES FOR LOADING 'ABS' FORMAT PAPER TAPES OR TO THE 'XXDP' SYSTEM REFERENCE DOCUMENT.

THIS PROGRAM MAY BE INCLUDED IN AN 'XXDP' CHAIN. IF THE PROGRAM IS PART OF A CHAIN AND IS LOADED FROM AN RP02, RPR02, OR AN RP03, THE PROGRAM WILL BYPASS TESTING DRIVE 0.

4. STARTING PROCEDURE

4.1 STARTING ADDRESSES

200	NORMAL STARTING ADDRESS
204	SELECT OPERATING PARAMETERS
210	SELECT RP11 ADDRESS
214	COMBINATION OF 204 AND 210

4.1.1 START FROM LOCATION 200

WHEN THE PROGRAM IS STARTED FROM LOCATION 200, THE PROGRAM WILL TEST ALL AVAILABLE DRIVES (SEE SECTION 3. FOR DRIVE 0 EXCEPTION) USING TESTS 0 - 10, & 12, WITH DEFAULT PARAMETERS. PREVIOUS PARAMETER CHANGES ARE OVERLAYED.

4.1.2 START FROM LOCATION 204

WHEN THE PROGRAM IS STARTED FROM LOCATION 204, THE PARAMETERS ARE RESET TO THEIR DEFAULT VALUES, AND THE PROGRAM ENTERS CONVERSATIONAL MODE. REFER TO SECTION 4.3.2.

4.1.3 START FROM LOCATION 210

WHEN THE PROGRAM IS STARTED FROM LOCATION 210, OPERATION IS THE SAME AS THE START FROM 200, EXCEPT THAT THE PROGRAM ASKS FOR A NEW RP11E ADDRESS, VECTOR ADDRESS, AND PRIORITY. THE OPERATOR MAY ENTER NEW ADDRESS VALUES OR RETAIN THE OLD VALUES. THE TEST PARAMETERS ARE RESET TO THE DEFAULT VALUES. REFER TO SECTION 4.3.1.

4.1.4 START FROM LOCATION 214

PROGRAM START FROM LOCATION 214 IS THE SAME AS THE START FROM LOCATION 204 EXCEPT THAT THE OPERATOR MAY ENTER NEW RP11E ADDRESS, VECTOR ADDRESS, OR PRIORITY. THE TEST PARAMETERS ARE RESET TO THE DEFAULT VALUES. REFER TO SECTIONS 4.3.1 & 4.3.2.

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.
4. SET 'FORMAT' AND 'MAINTENANCE' SWITCHES ON THE RP11E CONTROL PANEL TO 'NORMAL'.
5. LOAD ADDRESS 200, 204, 210, OR 214.
6. SET SWITCHES (SEE SECTION 5.
7. PRESS START.

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

. <CR PERIOD

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

.. <CR> PERIOD PERIOD

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

. <CR> COMMA

A CONTINUATION INDICATOR AND FIELD SEPARATOR:

1) THIS IS THE CONSTRUCTION USED WHEN NOT USING A PERIOD; THAT IS TO SAY THE COMMA SAYS "TERMINATE THE LINE BUT NOT THE INPUT FOR PARAMETER CHANGES".

2) SEPARATE MULTIPLE ENTRIES ON THE SAME TEXT STRING, E.G., A,B,C,D.

SLASH

A MODIFICATION INDICATOR: AT ANY GIVEN PARAMETER STOP IF A SLASH IS TYPED IT SAYS "OPEN THIS PARAMETER FOR CHANGES".

^U CONTROL-U

DUMP THE PRESENT INPUT STRING AND START A NEW LINE. TYPED BY DEPRESSING THE "CONTROL KEY" (CTRL) AND THEN STRIKING THE "U".

RUBOUT

DELETE THE LAST CHARACTER FROM THE INPUT STRING. TYPED BY STRIKING THE "RUBOUT" KEY. WHICH WILL BE ECHOED BY A BACKSLASH (\) FOLLOWED BY THE CHARACTER DELETED.

4.3.1 ADDRESS SELECTION

STARTING THE PROGRAM FROM 200 (8) WILL RESULT IN AUTOMATIC SELECTION OF THE DEFAULT RP11E VALUES OF BUS ADDRESS, VECTOR ADDRESS, AND PRIORITY LEVEL. IF THE DEFAULT VALUE OF THE BUS ADDRESS DOES NOT RESPOND (TIMES OUT) TO ADDRESSING, IT IS REPORTED AS AN ERROR. AFTER THE ERROR IS REPORTED, ONE OF TWO COURSES OF ACTION WILL BE TAKEN:

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

STARTING THE PROGRAM AT EITHER LOCATION 210 OR LOCATION 214 ALLOWS THE OPERATOR TO CHANGE THE RP11E BUS ADDRESS, THE VECTOR ADDRESS, AND/OR THE PRIORITY.

4.3.1.1 ADDRESS SELECTION EXAMPLES

EXAMPLE #1

```
RPDS=176710 176300..
```

EXAMPLE #2

```
RPDS=176710 176300.
RHVEC=254 260.
RHPRIO=5 6..
```

EXAMPLE #3

```
RPDS=177200,
RHVEC=254 260..
```

EXAMPLE #4

```
RP11 FAILED TO RESPOND TO ADDRESSING
RPDS    ERR PC
176710  XXXXXX
RPDS=176710 176300..
```

EXAMPLE #5

```
RPDS=176710 177667 6300.
RHVEC=254.
RHPRIO=5,
RPDS=176300
```

4.3.2 DRIVE, TEST, AND PARAMETER SELECTION

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

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

4.3.2.1 DRIVE, TEST, AND PARAMETER SELECTION DESCRIPTION

THE FOLLOWING IS A TABLE OF TERMS USED IN THE DESCRIPTION.

```
"R"     REPEATS (ITERATIONS)
"FC"    FIRST CYLINDER ADDRESS
"LC"    LAST CYLINDER ADDRESS
"IC"    INCREMENT CYLINDER
"TK"    TRACK ADDRESS
```

SPECIAL CASES OF CONTROL CHARACTERS

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

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

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

THE PROGRAM WILL RESPOND WITH:

```
DRIVE(S)=
```

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

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

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

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

IT NOW LOOKS LIKE THIS

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

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

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

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

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

NOTICE THIS SAYS SELECT TEST ONE, CONTINUE(,); SELECT TEST 3, OPEN(:
SELECT TESTS 4-7, CONTINUE(,); SELECT TEST 10, OPEN(; SELECT TEST
11, END OF INPUT(.).

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

RESPONDS: (FOR CLARITY ENTIRE TRANSACTION REPEATED)

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

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

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

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

```
DRIVE(S)=4.<CR>
TEST=1,3 4-7,10 11.<CR>
  TEST 3
    R=1 <CR>              ;DO NOT ALTER THIS LINE BUT CONTINUE
    FC=0 <CR>             ;DO NOT ALTER THIS LINE BUT CONTINUE
    LC=202
```

THE PROGRAM RESPONDS WITH THE PREVIOUSLY ASSIGNED PARAMETER FOR LAST CYLINDER ADDRESS IN THIS CASE USING "202" AS THE EXAMPLE. THIS IS WHAT THE USER INTENDED TO MODIFY AND IS WHY HE OPENED TEST #3. HE WANTS TO CHANGE IT TO CYLINDER 20. THEREFORE HE TYPES <> (OPEN LINE FOR CHANGE), "20" (THE NEW VALUE), <.> (THIS IS THE END OF CHANGES FOR THIS DRIVE), <CR> (TAKE IT).
THE TOTAL TRANSACTION AND RESPONSE:

```
DRIVE(S)=4.<CR>
TEST=1,3 4-7,10 11. <CR>
  TEST 3
    R= 1.<CR>
    FC=0.<CR>
    LC= 202 20.<CR>
  TEST 10
    R=1
```

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

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

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

```
DRIVE(S)=
```

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

4.3.3.2 DRIVE, TEST, 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 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=202/50..<CR>        ;TEST 10 IS STILL PENDING
                           ;AND WILL THEREFORE BE
                           ;SET TO THE SAME PARAMETERS
                           ;AS TEST 7 -- START EXECUTION
```

EXAMPLE #4

```
DRIVE=0,1,4.               ;TEST DRIVES 0,1, AND 4 IN SEQUENCE
TEST=0-5/                  ;CHANGE TEST 0-5
  TEST 0
    R=10.
    FC=0.
    LC=202 1..             ;CHANGE LAST CYLINDER FROM 202
                           ;TO 1 FOR TEST #0-5; START TESTING
```

4.4 'CONTROL C' OPERATION

THE OPERATOR MAY TERMINATE THE PROGRAM AT ANY TIME BY TYPING A 'CONTROL C'. 'CONTROL C' RETURNS THE PROGROM TO THE PARAMETER ENTRY ROUTINE. THE TEST PARAMETERS ARE NOT RESTORED TO THE DEFAULT VALUES BY THE 'CONTROL C' RETURN.
CONTROL G OPERATION REFER TO SECTION 5.

5. OPERATING PROCEDURE

IF THE DIAGNOSTIC IS RUN ON A CPU WITHOUT A SWITCH REGISTER THEN A SOFTWARE SWITCH REGISTER IS USED WHICH ALLOWS THE USER THE SAME SWITCH OPTIONS AS THE HARDWARE SWITCH REGISTER. IF THE HARDWARE SWITCH REGISTER DOES NOT EXIST OR IF ONE DOES AND IT CONTAINS ALL ONES (177777) THEN THE SOFTWARE SWITCH REGISTER (LOC. 176) IS USED.

CONTROL:

THIS PROGRAM ALSO SUPPORTS THE DYNAMIC LOADING OF THE SOFTWARE SWITCH REGISTER (LOC. 176) FROM THE TTY. THIS CAN BE ACCOMPLISHED BY DOING THE FOLLOWING:

1) TYPE CONTROL G (↑G); THIS WILL ALLOW THE TTY TO ENTER DATA INTO LOC. 176 AT SELECTED POINTS WITHIN THE PROGRAM.

2) THE MACHINE WILL THEN TYPE: SWR=XXXXXXNEW= (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER.)

3) AFTER THE ''NEW='' HAS BEEN TYPED THEN THE OPERATOR CAN DO ONE OF THE FOLLOWING AT THE TTY:

   A) TYPE A NUMBER TO BE LOADED INTO LOC. 176 FOLLOWED BY A (CR). (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED AND ONLY 6 NUMBERS WILL BE ALLOWED)
      IF A (CR) IS THE FIRST KEY DEPRESSED THE SOFTWARE SWITCH REGISTER CONTENTS WILL NOT BE CHANGED.

   B) IF A CONTROL U (↑U) IS DEPRESSED THEN THE PROGRAM WILL SEND YOU BACK TO STEP 2.

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(06)=1...TYPE ALL RP11 REGISTERS IF ERROR
SW(04)=1...TYPE THE LONG SEEK TIME GRAPH (TEST #12
SW(03)=1...PERFORM STALL BETWEEN SEEK ORDERS
SW(02)=1...USE RANDOM STALL VALUE BETWEEN SEEK ORDERS
SW(01)=1...PERFORM INCREMENTING STALL IN TEST 4
SW(00)=1...DO NOT CHECK POSITION AFTER SEEK ORDERS

6. ERRORS

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

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

6.1 ERROR TYPES

THE ERRORS THAT OCCUR IN THIS PROGRAM FALL INTO TWO (2) CATEGORIES DEFINED AND EXPLAINED AS FOLLOWS:

6.1.1 NON-FATAL ERROR

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

6.1.2 FATAL ERROR

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

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

6.2 ERROR RECOVERY

6.2.1 NON-FATAL ERROR

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

6.2.2 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

7.1 SUBSYSTEMS WITH MIXED DRIVE TYPES

THE PROGRAM ASSUMES THAT ONLY ONE KIND OF DRIVE IS ATTACHED TO A SYSTEM (E.G. RP02'S & RPR02'S OR RP03'S). IF BOTH RP02'S

(AND RPR02'S) OR RP03'S ARE ON THE SAME SYSTEM, THE PROGRAM WILL ASSUME THAT ALL OF THE DRIVES ARE RP03'S. TO TEST DRIVES ON A MIXED SYSTEM, TEST ALL OF THE SAME KIND OF DRIVE AT ONE TIME. THE OTHER DRIVES ON THE CONTROLLER MUST BE POWERED DOWN.

7.2 FORMATTED PACKS

EACH OF THE DRIVES BEING TESTED MUST HAVE A FORMATTED PACK. (UNLESS THE PROGRAM IS RUN WITH SWITCH (00) SET.)

8. MISCELLANEOUS

8.1 EXECUTION TIME

TO MAKE ONE PASS OF THE PROGRAM (TESTS 0 - 7) TAKES APPROXIMATELY 1.8 HOURS FOR EACH RP02 OR RPR02 TESTED AND APPROXIMATELY 6.5 HOURS FOR EACH RP03 TESTED.
NOTE THAT TEST 7 REQUIRES 1.3 HOURS FOR RP02 DRIVES AND 5.4 HOURS FOR RP03 DRIVES. IN SPITE OF THE LENGTHY RUN TIME OF TEST 7, IT IS RECOMMENDED THAT TEST 7 BE RUN OCCASIONLLY TO VERIFY THE SUBTRACTER IN THE DISK DRIVE. TEST 7 HAS ALSO PROVEN TO BE A VALUABLE LONG TERM SEEK VERIFIER, ALSO.

8.2 STACK POINTER

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

8.3 TIMING TEST PRINTOUTS

8.3.1 TIMING TEST (TESTS 10 & 11) PRINTOUT EXAMPLES

EXAMPLE #1

ONE CYLINDER SEEK TIMES
(SEEK TIME IS IN MICRO SECONDS)

FORWARD # OF SEEKS	FORWARD SEEK TIME	REVERSE # OF SEEKS	REVERSE SEEK TIME
193	10000	60	10000
9	12500	142	12500

EXAMPLE #2

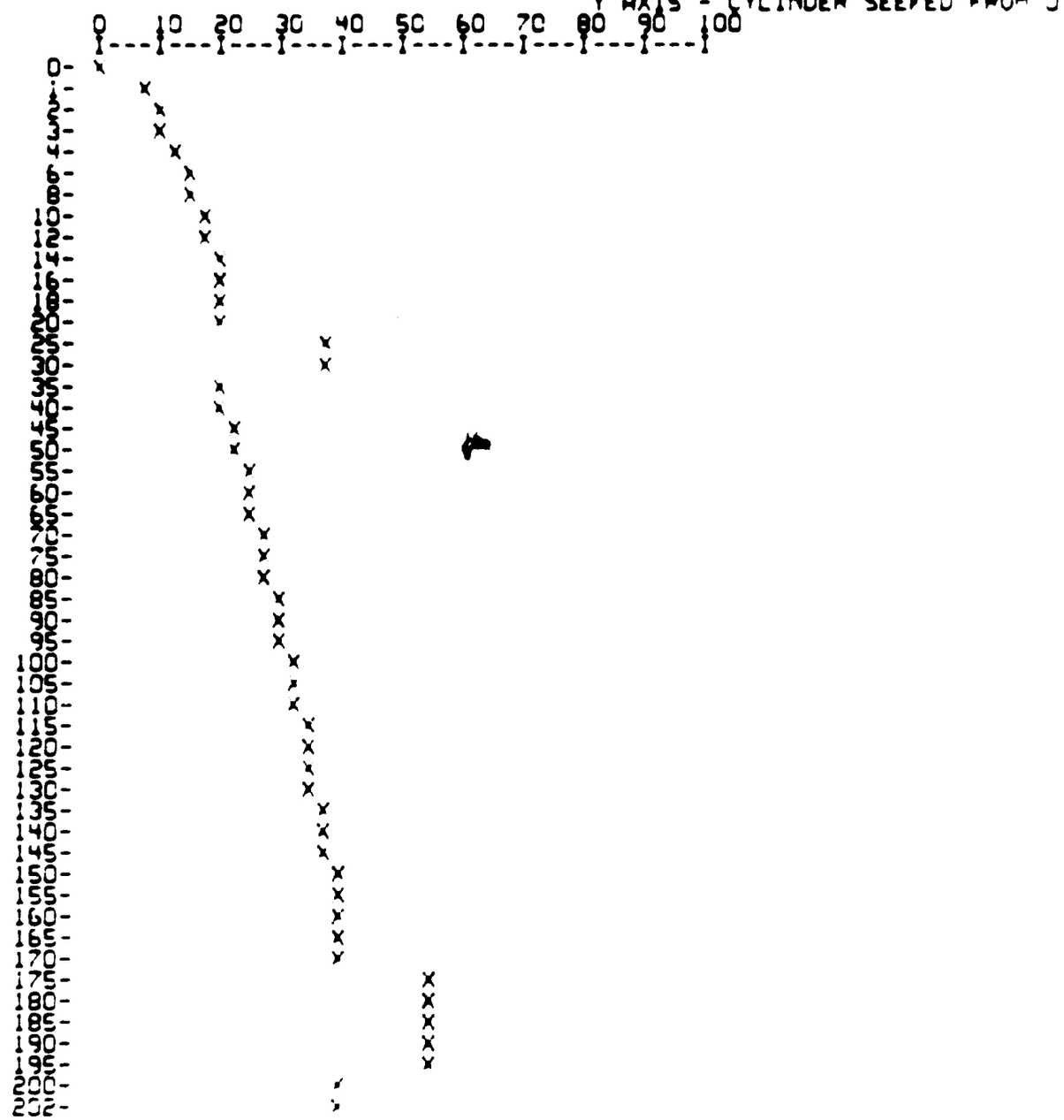
SEEK TIMES BETWEEN CYL 0 & 202
(SEEK TIME IS IN MICRO SECONDS

FORWARD # OF SEEKS	FORWARD SEEK TIME	REVERSE # OF SEEKS	REVERSE SEEK TIME
200	45000	156	45000
		44	60000

8.3.2 SEEK GRAPH TEST (TEST #12)
EXAMPLE: SMALL SEEK GRAPH (SWITCH (04) = 0

SEEK TIME GRAPH

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



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.

8.5 DEFAULT ADDRESSES, TEST SEQUENCE, AND PARAMETERS

8.5.1 DEFAULT ADDRESS

THE PROGRAM IS LOADED FOR THE FOLLOWING ADDRESSES:
RP11E BUS ADDRESS - 176710 RP11E VECTOR ADDRESS - 254 RP11E PRIORITY - 5

TO ALTER THESE ADDRESSES, START THE PROGRAM AT LOCATION 210 OR 214.

8.5.2 DEFAULT TEST SEQUENCE

THE DEFAULT TEST SEQUENCE IS TESTS 0 - 7. HIS SEQUENCE IS RESTORED ANY TIME A START FROM 200, 204, 210, OR 214 IS INITIATED. 'CONTROL C' RESTART DOES NOT ALTER THE TEST SEQUENCE.

8.5.3 DEFAULT TEST PARAMETERS

THE DEFAULT TEST PARAMETERS ARE GIVEN BELOW. REFER TO SECTION 4.3.1 FOR A GLOSSARY OF THE MNEMONICS USED TO IDENTIFY THE PARAMETERS. THE DEFAULT TEST PARAMETERS ARE RESTORED WHEN A START AT 200, 204, 210, OR 214 IS INITIATED. THE 'CONTROL C' RESTART DOES NOT ALTER THE CURRENT PARAMETER VALUES.

		PARAMETERS			
TEST #	'R'	'FC'	'LC'	'IC'	'TK'
TEST 0	100	-	-	-	0
TEST 1	1000	0	MAXCYL	-	0
TEST 2	10	0	MAXCYL	1	0
TEST 3	40	0	NOTE 3	1	0
TEST 4	10	0	MAXCYL	1	0
TEST 5	10	0	MAXCYL	1	0
TEST 6	10	0	MAXCYL	1	0
TEST 7	1	0	MAXCYL	1	0
TEST 10	(SEE BELOW - NOTE 1)				
TEST 11	1	0	MAXCYL	-	-
TEST 12	(SEE BELOW - NOTE 2)				

NOTE 1: TEST 10 ASSUMES THE FOLLOWING DEFAULT PARAMETERS. THESE PARAMETERS CANNOT BE ALTERED BY THE OPERATOR.

1	0	MAXCYL	1	-

NOTE 2: TEST 12 ASSUMES THE FOLLOWING DEFAULT PARAMETERS. THESE PARAMETERS CANNOT BE ALTERED BY THE OPERATOR.

1	0	MAXCYL	1	-

NOTE 3: 'MAXCYL' USED ABOVE IS USED TO DENOTE THE MAXIMUM CYLINDER ADDRESS USED BY THE PROGRAM. THE VALUE OF 'MAXCYL' IS DETERMINED BY THE TYPE OF DRIVES USED ON THE SYSTEM. 'MAXCYL' IS 405 (10) IF THE DRIVES ARE RP03'S AND IS 202 (10) IF THE DRIVES ARE RP02'S. IN TEST 3, 'LC' WILL BE 128 (10) IF THE DRIVES ARE RP02'S AND 256 (10) IF THE DRIVES ARE RP03'S.

8.6 TIMING TEST RESTRICTIONS

THE TIMING TESTS (TESTS 10, 11, & 12) ARE NOT INTENDED TO BE 'PASS-FAIL' TESTS. BECAUSE A HIGH RESOLUTION CLOCK IS NOT USED, THE TIMES ARE APPROXIMATE AND ARE INTENDED TO BE A GENERAL INDICATION OF THE CONDITION OF THE DRIVE'S SERVO SYSTEM. NO ATTEMPT SHOULD BE MADE TO ADJUST THE SERVO SYSTEM USING THE OUTPUT FROM THESE TESTS.

8.7 POSITIONING CHECK

IF SW(00) IS NOT SET, THE PROGRAM WILL CHECK THE DISK'S POSITION AFTER EACH SEEK (EXCEPT IN TESTS 10, 11, & 12).

THE PROGRAM CHECKS POSITION IN THE FOLLOWING MANNER:

USING THE 'SOT' COUNTER TO DETERMINE THE ADDRESS, THE PROGRAM READS THE HEADER FROM THE NEXT SECTOR AND COMPARES THE CYLINDER ADDRESS IN THE HEADER AGAINST THE CYLINDER THE PROGRAM EXPECTED; IF THE CYLINDER ADDRESS IN THE HEADER DOESN'T AGREE WITH THE EXPECTED CYLINDER ADDRESS, THE PROGRAM REPORTS A POSITIONING ERROR AND ISSUES A HOME SEEK TO THE DRIVE. (THE PROGRAM ALSO CHECKS POSITION AFTER A HOME SEEK OPERATION.

9. PROGRAM DESCRIPTION

THE RP11E DRIVE POSITIONING TEST CONTAINS 11 TESTS. TESTS 0 - 6 TEST THE OPERATION OF THE DRIVE'S SERVO SYSTEM. POSITIONING ACCURACY IS CHECKED BY READING A HEADER FROM THE DESTINATION CYLINDER USING A 'READ WITHOUT IMPLIED SEEK' INSTRUCTION AFTER THE SEEK TERMINATES.

TESTS 10, & 11, 12 MEASURE SEEK TIME. TEST 10 & 11 MEASURE THE TIME FOR SPECIFIC SEEK OPERATIONS; TEST 12 MEASURES THE TIME REQUIRED TO SEEK FROM CYLINDER 0 TO ALL OTHER CYLINDERS. TESTS 10 & 11 TYPE OUT THE MEASURED SEEK TIME IN MICROSECONDS WHILE TEST 12 PRODUCES A SEEK TIME GRAPH. IN THE SEEK TIMING TESTS, POSITIONING ACCURACY IS NOT CHECKED.

SEEK TIME IN TESTS 10, 11, & 12 IS DETERMINED BY COUNTING THE NUMBER OF SECTOR PULSES OCCURING BETWEEN START OF SEEK AND SEEK DONE. THE 'SOT' COUNTER IS USED AS THE REFERENCE IN THESE TESTS. THE RESOLUTION OF THE TIMING TESTS IS 2.5 MS.

THE PROGRAM WILL START BY IDENTIFING ITSELF AND DETERMINING ALL DRIVES THAT ARE AVAILABLE FOR TESTING. THEN BEGINNING WITH THE LOWEST NUMBERICAL DRIVE AND PROCEEDING IN SEQUENCIAL ORDER ALL OF THE DRIVES WILL BE TESTED. ONE PASS THROUGH THE TEST SEQUENCE (TESTS 0 - 7) WILL BE PERFORMED ON EACH DRIVE BEFORE MOVING TO THE NEXT DRIVE IN SEQUENCE. THE DRIVE TO BE TESTED WILL BE TYPED AT THE BEGINNING OF EACH PASS, AN "END OF PASS" MESSAGE WILL BE TYPED AT THE COMPLETION OF EACH PASS, AND AN "END OF TEST" MESSAGE WILL BE TYPED AFTER TESTING ALL DRIVES.

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

9.1 TEST DESCRIPTIONS

IN THE DESCRIPTIONS OF THE FOLLOWING TESTS, THE VARIABLES USED AND THEIR DEFAULT VALUES (UNLESS SPECIFIED OTHERWISE) ARE.

MNEMONIC	VARIABLE
FC	FIRST CYLINDER ADDRESS
LC	LAST CYLINDER ADDRESS
IC	INCREMENT VALUE
NC OR NC1	NEW OR MODIFIED CYLINDER (FC+IC) ADDRRESS
NC2	NEW OR MODIFIED CYLINDER (LC-IC) ADDRESS
TK	TRACK ADDRESS

9.1.1 TEST 0 - HOME SEEK TEST

THIS TEST WILL CAUSE THE DRIVE TO EXECUTE A HOME SEEK COMMAND CYCLE. AT THE COMPLETION OF THE COMMAND, THE STATUS INDICATORS ARE CHECKED TO VERIFY THAT NO ERRORS HAVE OCCURED.

9.1.2 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 INSURE PROPER OPERATION.

9.1.3 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 INSURE PROPER OPERATION.

9.1.4 TEST 3 - STEPPING SEEK TEST

THIS TEST WILL COMMAND SEEK CYCLES TO CYLINDER 0,1,2,4, 8, 16, 32, 64, AND 128 (AND 256 IF AN RP03). AT THE COMPLETION OF EACH SEEK COMMAND, THE PROPER INDICATORS ARE EXAMINED TO VERIFY PROPER OPERATION.

9.1.5 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 VERIFY PROPER OPERATION.

9.1.6 TEST 5 - CONVERGING/DIVERGING SEEK TEST

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

9.1.7 TEST 6 - SERVO ADDRESSING LOGIC NOSE GENERATION

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 UNITL "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 INSURE PROPER OPERATION.

9.1.8 TEST 7 - CYLINDER ADDRESSING TEST

THIS TEST WILL CAUSE THE DRIVE TO SEEK FROM EACH CYLINDER TO EVERY OTHER CYLINDER BETWEEN CYLINDERS "FC" AND "LC". SEEK CYCLES ARE COMMANDED FROM CYLINDER "NC" TO CYLINDER "NC1" AND BACK TO CYLINER "NC". "NC" AND "NC1" START AT "FC". "NC1" IS INCREMENTED BY "IC" UNTIL "NC1" IS EQUAL TO "LC". WHEN "NC1" IS EQUAL TO "LC", IT IS RESET TO "FC" AND "NC" IS INCREMENTED BY "IC". THE CYCLE IS COMPLETE WHEN "NC" AND "NC1" BOTH EQUAL "LC". AT THE COMPLETION OF EACH SEEK COMMAND, THE INDICATORS ARE EXAMINED TO VERIFY PROPER OPERATION.

9.1.9 TEST 10 - ONE CYLINDER SEEK TIMER TEST

THIS TEST PERFORMS ONE CYLINDER SEEKS FROM CYLINDER 0 TO THE MAXIMUM CYLINDER FOR THE DRIVE (E.G. 202 FOR RP02'S AND 405 FOR RP03'S). THE SEEK TIMES ARE MEASURED RELATIVE TO THE SECTOR PULSES FROM THE DRIVE. THE TEST DISPLAYS THE FORWARD AND REVERSE TIMES.

9.1.10 TEST 11 - SEEK-SEEK TIMER TEST

THIS TEST IS A GENERAL PURPOSE TIMING TEST WHICH PERFORMS 200 ITERATIONS BETWEEN THE CYLINDER IN 'FC' AND THE CYLINDER IN 'LC'. BOTH FORWARD AND REVERSE SEEKS ARE TIMED AND DISPLAYED. THE SEEK TIMES DISPLAYED ARE RELATIVE TO THE SECTOR COUNTER. 'FC' DEFAULTS TO 0; 'LC' DEFAULTS TO 'MAXCYL' (202 IF RP02'S OR 405 IF RP03'S).

9.1.11 TEST 12 - SEEK TIME GRAPH TEST

THIS TEST PRODUCES TWO SEEK TIME GRAPHS. THE FIRST GRAPH IS TYPED IS SWITCH <04> IS 0. THIS GRAPH IS A PLOT OF SEEK TIME AGAINST CYLINDERS SEEKED TO FROM CYLINDER 0. THE CYLINDERS PLOTTED ARE: 0,1,2,3,4,6,8,10,12,14,16,18,20,25...100,105...MAXCYL. THE OTHER GRAPH WHICH IS PRODUCED (SWITCH <04> SET) IS TIME TO SEEK TO ALL CYLINDERS FROM CYLINDER 0.

IN BOTH SEEK GRAPHS, ONLY THE FORWARD SEEK TIMES ARE MEASURED.

THIS TEST USES THE SECTOR COUNTER ('SOT') IN THE RP11 AS THE TIME BASE FOR THE MEASURMENTS. USE OF THE 'SOT' PROVIDES A RESOLUTION OF 2.5 MS.

.0. PROGRAM LISTING

```
                                .TITLE  MD-11-DZRPZ-C, RP11E DRIVE POSITIONING TEST
                                ;*COPYRIGHT (C) 1975,1977
                                ;*DIGITAL EQUIPMENT CORP.
                                ;*MAYNARD, MASS. 01754
                                ;*
                                ;*PROGRAM BY C. HESS/F. ROEMER
                                ;*
                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
                                ;*

                                .SBTTL  OPERATIONAL SWITCH SETTINGS
                                ;*
                                ;*      SWITCH                  USE
                                ;*      ------          --------------------
                                ;*        15            HALT ON ERROR
                                ;*        14            LOOP ON TEST
                                ;*        13            INHIBIT ERROR TYPEOUTS
                                ;*        11            INHIBIT ITERATIONS
                                ;*        10            BELL ON ERROR
                                ;*         9            LOOP ON ERROR
                                ;*         6            TYPE ALL THE RP11 REGISTERS IF ERROR
                                ;*         4            TYPE THE LONG SEEK TIME GRAPH
                                ;*         3            PERFORM STALL BETWEEN SEEK ORDERS
                                ;*         2            USE RANDOM STALL VALUE BETWEEN SEEK ORDERS
                                ;*         1            PERFORM INCREMENTING STALL IN TEST 4
                                ;*         0            DO NOT CHECK POSITION AFTER SEEK ORDERS

                                .SBTTL  TRAP CATCHER

        000000                          .=0
                                ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
        000174                          .=174
000174  000000                  DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
000176  000000                  SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER

                                .SBTTL  ACT11 HOOKS

                                ;;*******************************************************************
                                ;HOOKS REQUIRED BY ACT11
        000200                          $SVPC=.                 ;SAVE PC
        000046                          .=46
000046  007722                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
        000052                          .=52
000052  000000                          .WORD   0               ;;2)SET LOC.52 TO ZERO
        000200                          .=$SVPC                 ;; RESTORE PC

                                .SBTTL  STARTING ADDRESSES
        000200                  .=200
                                ;*200 = NORMAL START
000200  000137  002246                  JMP     @#START1
                                ;*204 = SELECT OPERATING PARAMETERS
```

```
000204  000137  002270                JMP     @#START2
                                ;*210 = SELECT RP11 ADDRESSES
000210  000137  002236                JMP     @#START3
                                ;*214 = COMBINATION OF 204 AND 210
000214  000137  002260                JMP     @#START4

                                .SBTTL  BASIC DEFINITIONS

                                ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
        001100                  STACK=  1100
                                .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                                .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL

                                ;*MISCELLANEOUS DEFINITIONS
        000011                  HT=     11              ;;CODE FOR HORIZONTAL TAB
        000012                  LF=     12              ;;CODE FOR LINE FEED
        000015                  CR=     15              ;;CODE FOR CARRIAGE RETURN
        000200                  CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
        177776                  PS=     177776          ;;PROCESSOR STATUS WORD
                                .EQUIV  PS,PSW
        177774                  STKLMT= 177774          ;;STACK LIMIT REGISTER
        177772                  PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
        177570                  DSWR=   177570          ;;HARDWARE SWITCH REGISTER
        177570                  DDISP=  177570          ;;HARDWARE DISPLAY REGISTER

                                ;*GENERAL PURPOSE REGISTER DEFINITIONS
        000000                  R0=     %0              ;;GENERAL REGISTER
        000001                  R1=     %1              ;;GENERAL REGISTER
        000002                  R2=     %2              ;;GENERAL REGISTER
        000003                  R3=     %3              ;;GENERAL REGISTER
        000004                  R4=     %4              ;;GENERAL REGISTER
        000005                  R5=     %5              ;;GENERAL REGISTER
        000006                  R6=     %6              ;;GENERAL REGISTER
        000007                  R7=     %7              ;;GENERAL REGISTER
        000006                  SP=     %6                STACK POINTER
        000007                  PC=     %7              ;;PROGRAM COUNTER

                                ;*PRIORITY LEVEL DEFINITIONS
        000000                  PR0=    0               ;;PRIORITY LEVEL 0
        000040                  PR1=    40              ;;PRIORITY LEVEL 1
        000100                  PR2=    100             ;;PRIORITY LEVEL 2
        000140                  PR3=    140             ;;PRIORITY LEVEL 3
        000200                  PR4=    200             ;;PRIORITY LEVEL 4
        000240                  PR5=    240             ;;PRIORITY LEVEL 5
        000300                  PR6=    300             ;;PRIORITY LEVEL 6
        000340                  PR7=    340             ;;PRIORITY LEVEL 7

                                ;*"SWITCH REGISTER" SWITCH DEFINITIONS
        100000                  SW15=   100000
        040000                  SW14=   40000
        020000                  SW13=   20000
        010000                  SW12=   10000
        004000                  SW11=   4000
        002000                  SW10=   2000
        001000                  SW09=   1000
        000400                  SW08=   400
```

```
000200                  SW07=   200
000100                  SW06=   100
000040                  SW05=   40
000020                  SW04=   20
000010                  SW03=   10
000004                  SW02=   4
000002                  SW01=   2
000001                  SW00=   1
                        .EQUIV  SW09,SW9
                        .EQUIV  SW08,SW8
                        .EQUIV  SW07,SW7
                        .EQUIV  SW06,SW6
                        .EQUIV  SW05,SW5
                        .EQUIV  SW04,SW4
                        .EQUIV  SW03,SW3
                        .EQUIV  SW02,SW2
                        .EQUIV  SW01,SW1
                        .EQUIV  SW00,SW0

                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                  BIT15=  100000
040000                  BIT14=  40000
020000                  BIT13=  20000
010000                  BIT12=  10000
004000                  BIT11=  4000
002000                  BIT10=  2000
001000                  BIT09=  1000
000400                  BIT08=  400
000200                  BIT07=  200
000100                  BIT06=  100
000040                  BIT05=  40
000020                  BIT04=  20
000010                  BIT03=  10
000004                  BIT02=  4
000002                  BIT01=  2
000001                  BIT00=  1
                        .EQUIV  BIT09,BIT9
                        .EQUIV  BIT08,BIT8
                        .EQUIV  BIT07,BIT7
                        .EQUIV  BIT06,BIT6
                        .EQUIV  BIT05,BIT5
                        .EQUIV  BIT04,BIT4
                        .EQUIV  BIT03,BIT3
                        .EQUIV  BIT02,BIT2
                        .EQUIV  BIT01,BIT1
                        .EQUIV  BIT00,BIT0

                        ;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
000014                  TBITVEC=14              ;;"T" BIT
000014                  TRTVEC= 14              ;;TRACE TRAP
000014                  BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
000020                  IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024                  PWRVEC= 24              ;;POWER FAIL
000030                  EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
```

```
000034          TRAPVEC=34          ;;"TRAP" TRAP
000060          TKVEC=  60          ;;TTY KEYBOARD VECTOR
000064          TPVEC=  64          ;;TTY PRINTER VECTOR
000240          PIRQVEC=240         ;;PROGRAM INTERRUPT REQUEST VECTOR

                ;OP CODE DEFINITIONS

000001          CLEAR=1                     ;CONTROLLER CLEAR
000003          WRTSEK=3                    ;WRITE WITH IMPLIED SEEK AND HEAD SELECTION
000005          RDSEK=5                     ;READ WITH IMPLIED SEEK AND HEAD SELECT
000007          WCKSEK=7                    ;WRITE CHECK WITH IMPLIIED SEEK AND HEAD SELECT
000011          SEEK=11                     ;SEEK
000013          WRITE=13                    ;WRITE (NO IMPLIED SEEK OR HEAD SELECT)
000015          HOMSEK=15                   ;HOME SEEK
000017          READ=17                     ;READ (NO IMPLIED SEEK OR HEAD SELECT)

                ;RP11 REGISTER EQUATES

                ;RPDS (DRIVE STATUS REGISTER)

                ;ATTENTION BITS ARE REFERENCED BY BIT NUMBER (BIT00 - BIT07)
000400          SUWP=400                    ;SELECTED UNIT WRITE PROTECTED
001000          SUFU=1000                   ;SELECTED UNIT FILE UNSAFE
002000          SUSU=2000                   ;SELECTED UNIT SEEK UNDERWAY
004000          SUSI=4000                   ;SELECTED UNIT SEEK INCOMPLETE
010000          HNF=10000                   ;HEADER NOT FOUND
020000          SURP03=20000                ;SELECTED UNIT IS AN RP03
040000          SUOL=40000                  ;SELECTED UNIT IS ONLINE
100000          SURDY=100000                ;SELECTED UNIT IS READY

                ;RPER (ERROR REGISTER)

000001          DSKERR=1                    ;DISK ERROR
000002          EOP=2                       ;END OF PACK
000004          NXME=4                      ;NON-EXISTENT MEMORY
000010          WCE=10                      ;WRITE CHECK ERROR
000020          TIMEE=20                    ;TIMING ERROR
000040          CSME=40                     ;CHECKSUM ERROR
000100          WPE=100                     ;WORD PARITY ERROR
000200          LPE=200                     ;LONGITUDINAL PARITY ERROR
000400          MODERR=400                  ;MODE ERROR
001000          FMTE=1000                   ;FORMAT ERROR
002000          PROG=2000                   ;PROGRAM ERROR
004000          NXS=4000                    ;NON-EXISTENT SECTOR
010000          NXT=10000                   ;NON-EXISTENT TRACK
020000          NXC=20000                   ;NON-EXSITENT CYLINDER
040000          FUV=40000                   ;FILE UNSAFE VIOLATION
100000          WPV=100000                  ;WRITE PROTECT VIOLATION

                ;RPCS  (CONTROL REGISTER)

                ;'GO' BIT AND FUNCTION BITS ARE LOADED TOGETHER AND ARE DEFINED ELSEWHERE.
                ;EXTENDED MEMORY ADDRESS BITS ARE REFERED TO BY BIT NUMBER
000100          IDE=100                     ;INTERRUPT ON DONE
000200          RDY=200                     ;RP11 READY
                ;DRIVE SELECT BITS ARE REFERED TO BY BIT NUMBER
```

```
004000          HDR=4000                ;HEADER
010000          MODE=10000              ;MODE
020000          AIE=20000               ;ATTENTION INTERRUPT ENABLE
040000          HE=40000                ;HARD ERROR
100000          ERR=100000              ;ERROR
```

```
                                        .SBTTL  COMMON TAGS

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

                 001100                         .=1100
001100                          $CMTAG:                         ;;START OF COMMON TAGS
001100   000000                 $PASS:   .WORD   0              ;;CONTAINS PASS COUNT
001102       000                 $TSTNM:  .BYTE   0              ;;CONTAINS THE TEST NUMBER
001103       000                 $ERFLG:  .BYTE   0              ;;CONTAINS ERROR FLAG
001104   000000                 $TCNT:   .WORD   0              ;;CONTAINS SUBTEST ITERATION COUNT
001106   000000                 $LPADR:  .WORD   0              ;;CONTAINS SCOPE LOOP ADDRESS
001110   000000                 $LPERR:  .WORD   0              ;;CONTAINS SCOPE RETURN FOR ERRORS
001112   000000                 $ERTTL:  .WORD   0              ;;CONTAINS TOTAL ERRORS DETECTED
001114       000                 $ITEMB:  .BYTE   0              ;;CONTAINS ITEM CONTROL BYTE
001115       001                 $ERMAX:  .BYTE   1              ;;CONTAINS MAX. ERRORS PER TEST
001116   000000                 $ERRPC:  .WORD   0              ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120   000000                 $GOADR:  .WORD   0              ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122   000000                 $BOADR:  .WORD   0              ;;CONTAINS ADDRESS OF 'BAD' DATA
001124   000000                 $GODAT:  .WORD   0              ;;CONTAINS 'GOOD' DATA
001126   000000                 $BODAT:  .WORD   0              ;;CONTAINS 'BAD' DATA
001130   000000                          .WORD   0              ;;RESERVED--NOT TO BE USED
001132   000000                          .WORD   0
001134       000                 $AUTOB:  .BYTE   0              ;;AUTOMATIC MODE INDICATOR
001135       000                 $INTAG:  .BYTE   0              ;;INTERRUPT MODE INDICATOR
001136   000000                          .WORD   0
001140   177570                 SWR:     .WORD   DSWR           ;;ADDRESS OF SWITCH REGISTER
001142   177570                 DISPLAY:  .WORD  DDISP          ;;ADDRESS OF DISPLAY REGISTER
001144   177560                 $TKS:    177560                 ;;TTY KBD STATUS
001146   177562                 $TKB:    177562                 ;;TTY KBD BUFFER
001150   177564                 $TPS:    177564                 ;;TTY PRINTER STATUS REG. ADDRESS
001152   177566                 $TPB:    177566                 ;;TTY PRINTER BUFFER REG. ADDRESS
001154       000                 $NULL:   .BYTE   0              ;;CONTAINS NULL CHARACTER FOR FILLS
001155       002                 $FILLS:  .BYTE   2              ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156       012                 $FILLC:  .BYTE   12             ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157       000                 $TPFLG:  .BYTE   0              ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES
001160   000000                 $REGAD:  .WORD   0              ;;CONTAINS THE ADDRESS FROM
                                                                 ;;WHICH  ($REG0) WAS OBTAINED
001162   000000                 $REG0:   .WORD   0              ;;CONTAINS (($REGAD)+0)
001164   000000                 $REG1:   .WORD   0              ;;CONTAINS (($REGAD)+2)
001166   000000                 $REG2:   .WORD   0              ;;CONTAINS (($REGAD)+4)
001170   000000                 $REG3:   .WORD   0              ;;CONTAINS (($REGAD)+6)
001172   000000                 $REG4:   .WORD   0              ;;CONTAINS (($REGAD)+10)
001174   000000                 $REG5:   .WORD   0              ;;CONTAINS (($REGAD)+12)
001176   000000                 $REG6:   .WORD   0              ;;CONTAINS (($REGAD)+14)
001200   000000                 $REG7:   .WORD   0              ;;CONTAINS (($REGAD)+16)
001202   000000                 $REG10:  .WORD   0              ;;CONTAINS (($REGAD)+20)
001204   000000                 $TMP0:   .WORD   0              ;;USER DEFINED
001206   000000                 $TIMES:  0                      ;;MAX. NUMBER OF ITERATIONS
001210   000000                 $ESCAPE: 0                      ;;ESCAPE ON ERROR ADDRESS
001212   177607   000377        $BELL:   .ASCIZ  <207><377><377>;;CODE FOR BELL
001216       077                 $QUES:   .ASCII  /?/            ;;QUESTION MARK
001217       015                 $CRLF:   .ASCII  <15>           ;;CARRIAGE RETURN
001220   000012                 $LF:     .ASCIZ  <12>           ;;LINE FEED
                                        ;;*****************************************************************
```

```
001222  000000          CNTRLC: .WORD   0               ;CONTROL "C" FLAG
001224  000000          BUSADR: .WORD   0               ;GET ADDRESSES FROM THE TTY FLAG  0=NO, -1=YES
001226  000000          INADR:  .WORD   0               ;CONTAINS INNER ADDRESS FOR TIMING TEST
001230  000000          OUTADR: .WORD   0               ;CONTAINS OUTER ADDRESS FOR TIMING TEST
001232  000000          DRVSEL: .WORD   0               ;DRIVES SELECTED FOR TESTING
001234  000777          TSTNUM: .WORD   777             ;RUN TESTS 0-10
001236  000000          MAXCYL: .WORD   0               ;MAXIMUM CYLINDER ADDRESS
001240  000000          DRVTYP: .WORD   0               ;CONTAINS A BIT FOR EACH RP03 DRIVE
                                                        ;ON THE SYSTEM.  BIT00=DRIVE 0, ETC
001242  000000          SPSAV:  .WORD   0               ;SAVE THE STACK POINTER HERE
001244  000000          BYPASS: .WORD   0               ;BYPASS CURRENT TEST ADDRESS
001246  000000          CHKDRV: .WORD   0               ;DRIVE UNDER TEST
001250  000000          DRVMSK: .WORD   0               ;DRIVE MASK BIT
001252  000000          DRVBAD: .WORD   0               ;CONTAINS BITS FOR UNSAFE OR OFFLINE DRIVES
001254  000000          CYL.CR: .WORD   0               ;CURRENT CYLINDER
001256  000000          TRK.RD: .WORD   0               ;TRACK READ
001260  000000          SEC.RD: .WORD   0               ;SECTOR READ
001262  000000          CYL.DS: .WORD   0               ;CYLINDER DESIRED
001264  000000          STR20X: .WORD   0               ;START FROM 200, 204, 210, 214 FLAG

                        ;CONSTANTS STORAGE
001266  176710          RPADR:  .WORD   176710          ;RP11 ADDRESS
001270  000254          RPVEC:  .WORD   254             ;RP11 VECTOR ADDRESS
001272  000240          RPPRIO: .WORD   5*32.           ;RP11 PRIORITY
001274  000000          STALL0: .WORD   0               ;VARIABLE STALL (TEST 4)
001276  000031          STALL1: .WORD   25.             ;25 MILLISECONDS STALL (TEST 0-6
001300  000000          STALLG: .WORD   0               ;GENERAL STALL VALUE (VARIABLE
001302  000062          MXSTAL: .WORD   50.             ;MAX INCREMENTING STALL ALLOWED IN TEST 4
                                .EVEN

                        ;*TABLE OF DRIVE STATUS INDICATORS
                                ;*DRVSTA>0 - DRIVE IS ONLINE
                                ;*DRVSTA<0 - DRIVE IS OFFLINE, UNSAFE, OR AN RP03

001304  000             DRVSTA: .BYTE   0               ;DRIVE 0
001305  000                     .BYTE   0               ;DRIVE 1
001306  000                     .BYTE   0               ;DRIVE 2
001307  000                     .BYTE   0               ;DRIVE 3
001310  000                     .BYTE   0               ;DRIVE 4
001311  000                     .BYTE   0               ;DRIVE 5
001312  000                     .BYTE   0               ;DRIVE 6
001313  000                     .BYTE   0               ;DRIVE 7

                        ;ATTENTION BIT TABLE

001314  001             ATABIT: .BYTE   1               ;DRIVE 0
001315  002                     .BYTE   2               ;DRIVE 1
001316  004                     .BYTE   4               ;DRIVE 2
001317  010                     .BYTE   10              ;DRIVE 3
001320  020                     .BYTE   20              ;DRIVE 4
001321  040                     .BYTE   40              ;DRIVE 5
001322  100                     .BYTE   100             ;DRIVE 6
001323  200                     .BYTE   200             ;DRIVE 7

                        ;BIT TABLE
001324  000000          BITS:   .WORD   BIT00
```

```
001326  000002                  .WORD   BIT01
001330  000004                  .WORD   BIT02
001332  000010                  .WORD   BIT03
001334  000020                  .WORD   BIT04
001336  000040                  .WORD   BIT05
001340  000100                  .WORD   BIT06
001342  000200                  .WORD   BIT07
001344  000400                  .WORD   BIT08
001346  001000                  .WORD   BIT09
001350  002000                  .WORD   BIT10
001352  004000                  .WORD   BIT11
001354  010000                  .WORD   BIT12
001356  020000                  .WORD   BIT13
001360  040000                  .WORD   BIT14
001362  100000                  .WORD   BIT15

                        ;DRIVE OPERATION PARAMETER BLOCK

001364     000          DPB:    .BYTE   0               ;OP CODE
001365     000                  .BYTE   0               ;DRIVE ADDRESS
001366  000000                  .WORD   0               ;CYLINDER ADDRESS
001370     000                  .BYTE   0               ;SECTOR ADDRESS
001371     000                  .BYTE   0               ;TRACK ADDRESS

                        ;COMMON STORAGE FOR TEST PARAMETER
001372  000000          PRM:    .WORD   0
001374  000000          RPT:    .WORD   0               ;REPEAT COUNTS FOR ALL TESTS
001376  000000          FC:     .WORD   0               ;FIRST CYLINDER
001400  000000          LC:     .WORD   0               ;LAST CYLINDER
001402  000000          IC:     .WORD   0               ;INCREMENT CYLINDER
001404  000000          TK:     .WORD   0               ;TRACK ADDRESS

                        ;TABLE OF PARAMETER POINTERS
001406  001632          PRMPT:  .WORD   PRM0
001410  001640                  .WORD   PRM1
001412  001654                  .WORD   PRM2
001414  001670                  .WORD   PRM3
001416  001704                  .WORD   PRM4
001420  001720                  .WORD   PRM5
001422  001734                  .WORD   PRM6
001424  001750                  .WORD   PRM7
001426  001764                  .WORD   PRM10
001430  001770                  .WORD   PRM11
001432  002000                  .WORD   PRM12

                        ;PARAMETER UPPER LIMIT
001434  077777          PRMLMT: .WORD   32767.          ;"R"
001436  000312                  .WORD   202.            ;"FC"
001440  000312                  .WORD   202.            ;"LC"
001442  000312                  .WORD   202.            ;"IC"
001444  000023                  .WORD   19.             ;"TK"

                        ;TABLE OF MESSAGE POINTERS
001446  021420          PRMMSG: .WORD   MSG.R
001450  021422                  .WORD   MSG.FC
```

```
001452  021425                          .WORD   MSG.LC
001454  021430                          .WORD   MSG.IC
001456  021433                          .WORD   MSG.TK

                                ;DEFAULT VALUES OF TESTS PARAMETERS
001460  000021  000144  000000  DFLT:   .WORD   21,100.,0          ;HOME SEEK (T0)
001466  000037  001750  000000          .WORD   37,1000.,0,202.,0,0  ;SEEK/SEEK (T1)
001474  000312  000000  000000
001502  000037  000010  000000          .WORD   37,10.,0,202.,1,0  ;INCREMENT SEEK (T2)
001510  000312  000001  000000
001516  000037  000010  000000          .WORD   37,10.,0,128.,1,0  ;STEPPING SEEK (T3)
001524  000200  000001  000000
001532  000037  000010  000000          .WORD   37,10.,0,202.,1,0  ;OSCILLATING SEEK (T4)
001540  000312  000001  000000
001546  000037  000010  000000          .WORD   37,10.,0,202.,1,0  ;CONVERGING/DIVERGING SEEK (T5)
001554  000312  000001  000000
001562  000037  000010  000000          .WORD   37,10.,0,202.,1,0  ;SERVO ADDRESSING LOGIC NOISE (T6)
001570  000312  000001  000000
001576  000037  000001  000000           WORD   37,1,0,202.,1,0  ;CYLINDER ADDRESSING TEST (T7)
001604  000312  000001  000000
001612  000001  000001                  .WORD   1,1                ;ONE CYLINDER SEEK TIMER TEST (T10)
001616  000037  000001  000000          .WORD   37,1,0,202         ;SEEK-SEEK TIMER TEST (T11)
001624  000312
001626  000001  000001                  .WORD   1,1                ;SEEK TIME GRAPH TEST (T12)

                                ;PARAMETER TABLES

                                ;HOME SEEK (T0)
001632  000021                  PRM0:   .WORD   21
001634  000144                  RPT0:   .WORD   100.
001636  000000                  TK0:    .WORD   0

                                ;SEEK-SEEK (T1)
001640  000037                  PRM1:   .WORD   37
001642  001750                  RPT1:   .WORD   1000.
001644  000000                  FC1:    .WORD   0
001646  000312                  LC1:    .WORD   202.
001650  000000                  IC1:    .WORD   0
001652  000000                  TK1:    .WORD   0

                                ;INCREMENT SEEK (T2)
001654  000037                  PRM2:   .WORD   37
001656  000012                  RPT2:   .WORD   10.
001660  000000                  FC2:    .WORD   0
001662  000312                  LC2:    .WORD   202.
001664  000001                  IC2:    .WORD   1
001666  000000                  TK2:    .WORD   0

                                ;STEPPING SEEK (T3)
001670  000037                  PRM3:   .WORD   37
001672  000012                  RPT3:   .WORD   10.
001674  000000                  FC3:    .WORD   0
001676  000200                  LC3:    .WORD   128.
001700  000001                  IC3:    .WORD   1
001702  000000                  TK3:    .WORD   0
```

```
                                        ;OSCILLATING SEEK (T4
001704  000037                          PRM4:   .WORD   37
001706  000012                          RPT4:   .WORD   10.
001710  000000                          FC4:    .WORD   0
001712  000312                          LC4:    .WORD   202.
001714  000001                          IC4:    .WORD   1
001716  000000                          TK4:    .WORD   0

                                        ;CONVERGING/DIVERGING SEEK (T5)
001720  000037                          PRM5:   .WORD   37
001722  000012                          RPT5:   .WORD   10.
001724  000000                          FC5:    .WORD   0
001726  000312                          LC5:    .WORD   202.
001730  000001                          IC5:    .WORD   1
001732  000000                          TK5:    .WORD   0

                                        ;SERVO ADDRESSING LOGIC NOISE GENERATOR (T6
001734  000037                          PRM6:   .WORD   37
001736  000012                          RPT6:   .WORD   10.
001740  000000                          FC6:    .WORD   0
001742  000312                          LC6:    .WORD   202.
001744  000001                          IC6:    .WORD   1
001746  000000                          TK6:    .WORD   0

                                        ;CYLINDER ADDRESSING (T7)
001750  000037                          PRM7:   .WORD   37
001752  000001                          RPT7:   .WORD   1
001754  000000                          FC7:    .WORD   0
001756  000312                          LC7:    .WORD   202.
001760  000001                          IC7:    .WORD   1
001762  000000                          TK7:    .WORD   0

                                        ;ONE CYLINDER SEEK TIMER TEST (T10
001764  000001                          PRM10:  .WORD   1
001766  000001                          RPT10:  .WORD   1

                                        ;SEEK-SEEK TIMING TEST (T11)
001770  000007                          PRM11:  .WORD   7
001772  000001                          RPT11:  .WORD   1
001774  000000                          FC11:   .WORD   0
001776  000312                          LC11:   .WORD   202.

                                        ;SEEK TIME GRAPH TEST (T12
002000  000001                          PRM12:  .WORD   1
002002  000001                          RPT12:  .WORD   1

                                        ;SAVE THE RP11E REGISTERS HERE

002004  000000                          $RPDS:  .WORD   0               ;DRIVE STATUS REGISTER
002006  000000                          $RPER:  .WORD   0               ;ERROR REGISTER
002010  000000                          $RPCS:  .WORD   0               ;COMMAND & STATUS REGISTER
002012  000000                          $RPWC:  .WORD   0               ;WORD COUNT REGISTER
002014  000000                          $RPBA:  .WORD   0               ;BUFFER ADDRESS REGISTER
002016  000000                          $RPCA:  .WORD   0               ;CURRENT CYLINDER ADDRESS REGISTER
002020  000000                          $RPDA:  .WORD   0               ;TRACK-SECTOR ADDRESS REGISTER
002022  000000                          $RPM1:  .WORD   0               ;MAINTENANCE REGISTER #1
```

```
002024  000000                  $SUCA:  .WORD   0               ;SELECTED UNIT CYLINDER ADDRESS REGISTER
002026  000000                  $SILO:  .WORD   0               ;SILO REGISTER

                                ;REGISTER INDEX VALUES

        000000                  RPDS=00                         ;DRIVE STATUS REGISTER
        000002                  RPER=02                         ;ERROR REGISTER
        000004                  RPCS=04                         ;CONTROL REGISTER
        000006                  RPWC=06                         ;WORD COUNT REGISTER
        000010                  RPBA=10                         ;BUFFER ADDRESS REGISTER
        000012                  RPCA=12                         ;CYLINDER ADDRESS REGISTER
        000014                  RPDA=14                         ;SECTOR/TRACK ADDRESS REGISTER
        000016                  RPM1=16                         ;MAINTENANCE REGISTER #1
        000020                  PPM2=20                         ;MAINTENANCE REGISTER #2
        000022                  RPM3=22                         ;MAINTENANCE REGISTER #3
        000024                  SUCA=24                         ;SELECTED UNIT CYLINDER ADDRESS REGISTER
        000026                  SILO=26                         ;SILO REGISTER

002030  020040      040         BLNK13: .ASCII
002033      040                 BLNK10: .ASCII
002034      040                 BLNK59: .ASCII
002035      040                 BLNK58: .ASCII
002036      040                 BLNK57: .ASCII
002037      040                 BLNK56: .ASCII
002040      040                 BLNK55: .ASCII
002041      040                 BLNK54: .ASCII
002042      040                 BLNK53: .ASCII
002043      040                 BLNK52: .ASCII
002044  000040                  BLNK51: .ASCIZ

                                        .EVEN
```

```
                                        .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

 002046                                 $ERRTB:
                                        ;*EM AND DH ARE ASCIZ MESSAGES. DT IS A STRING OF WORDS THAT POINT TO THE
                                        ;*DATA TO BE TYPED AND DF IS A STRING OF DATA THAT TELL HOW THE DT WORDS
                                        ;*ARE TO BE TYPED.  IF ANY OF THE POINTERS ARE NOT NEEDED FOR A PARTICULAR
                                        ;*ERROR IT IS REPLACED WITH A ZERO.
                                        ;*EACH OF THE ITEMS BELOW REFER TO THE ERROR NUMBER AND INDICATE
                                        ;*THE INFORMATION THAT WILL BE TYPED WHEN THE ERROR OCCURS.
                                        ;*UNLESS STATED OTHER ALL NUMBERS ARE OCTAL


                                        ;ERROR 1

 002046  021642                                 EM1             ;RP11 FAILED TO RESPOND TO ADDRESSING
 002050  022603                                 DH1
 002052  023336                                 DT1
 002054  023472                                 DF1

                                        ;ERROR 2

 002056  021707                                 EM2             ;DRIVE UNSAFE
 002060  022620                                 DH2
 002062  023342                                 DT2
 002064  023476                                 DF2

                                        ;ERROR 3

 002066  021724                                 EM3             ;DRIVE OFFLINE
 002070  022620                                 DH2
 002072  023342                                 DT2
 002074  023476                                 DF2

                                        ;ERROR 4

 002076  021742                                 EM4             ;DISK ERROR DURING TIMING TEST
 002100  022620                                 DH2
 002102  023342                                 DT2
 002104  023476                                 DF2

                                        ;ERROR 5

 002106  022001                                 EM5             ;NO INTERRUPT FROM POSITIONING AFTER 1 SECOND
```

```
002110  022620                  DH2
002112  023342                  DT2
002114  023476                  DF2

                        ;ERROR 6

002116  022056                  EM6             ;DISK ERROR AFTER POSITIONING
002120  022620                  DH2
002122  023342                  DT2
002124  023476                  DF2

                        ;ERROR 7

002126  022113                  EM7             ;ATTN BIT NOT SET AFTER POSITIONING
002130  022620                  DH2
002132  023342                  DT2
002134  023476                  DF2

                        ;ERROR 10

002136  022166                  EM10            ;DRIVE OFFLINE AFTER POSITIONING
002140  022620                  DH2
002142  023342                  DT2
002144  023476                  DF2

                        ;ERROR 11

002146  022226                  EM11            ;'SUCA' NOT CORRECT AFTER POSITIONING
002150  022675                  DH11
002152  023356                  DT11
002154  023502                  DF11

                        ;ERROR 12

002156  022273                  EM12            ;DISK ERROR WHILE VERIFYING POSITION
002160  022762                  DH12
002162  023374                  DT12
002164  023506                  DF12

                        ;ERROR 13

002166  022337                  EM13            ;DISK POSITIONED TO WRONG CYLINDER
002170  023057                  DH13
002172  023414                  DT13
002174  023512                  DF13

                        ;ERROR 14

002176  022401                  EM14            ;NO INTERRUPT FROM I O AFTER 1 SECOND
002200  022620                  DH2
002202  023342                  DT2
002204  023476                  DF2

                        ;ERROR 15

002206  022446                  EM15            ;SEEK INCOMPLETE
```

```
002210  022620                          DH2
002212  023342                          DT2
002214  023476                          DF2

                                :ERROR 16

002216  022466                          EM16            ;DRIVE NOT READY AFTER POSITIONING
002220  022620                          DH2
002222  023342                          DT2
002224  023476                          DF2

                                :ERROR 17

002226  022530                          EM17            ;DRIVE NOT READY/OFFLINE DURING TIMING TEST
002230  022620                          DH2
002232  023342                          DT2
002234  023476                          DF2
```

```
                                        .SBTTL  START OF PROGRAM

002236  012737  177777  001224  START3: MOV     #-1,@#BUSADR     ;GET BUSADR FLAG
002244  000402                          BR      STRTIA
002246  005037  001224          START1: CLR     @#BUSADR         ;CLR BUSADR FLAG
002252  005037  001222          STRTIA: CLR     @#CNTRLC         ;NO CONTROL "C"
002256  000411                          BR      START            ;
002260  012737  177777  001224  START4: MOV     #-1,@#BUSADR     ;SET BUSADR FLAG
002266  000402                          BR      STRT2A
002270  005037  001224          START2: CLR     @#BUSADR         ;CLR BUSADR FLAG
002274  012737  177777  001222  STRT2A: MOV     #-1,@#CNTRLC     ;SET CONTROL "C" FLAG
002302  012737  177777  001264  START:  MOV     #-1,STR2OX       ;SET START FLAG
002310  000005                          RESET
                                        .SBTTL  INITIALIZE THE COMMON TAGS
                                        ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
002312  012706  001100                  MOV     #$CMTAG,R6       ;;FIRST LOCATION TO BE CLEARED
002316  005026                          CLR     (R6)+            ;;CLEAR MEMORY LOCATION
002320  022706  001140                  CMP     #SWR,R6 ;;DONE?
002324  001374                          BNE     .-6              ;;LOOP BACK IF NO
002326  012706  001100                  MOV     #STACK,SP        ;;SETUP THE STACK POINTER
                                        ;;INITIALIZE A FEW VECTORS
002332  012737  012524  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
002340  012737  000340  000022          MOV     #340,@#IOTVEC+2  ;;LEVEL 7
002346  012737  007742  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
002354  012737  000340  000032          MOV     #340,@#EMTVEC+2  ;;LEVEL 7
002362  012737  013164  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
002370  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
002376  005037  001206                  CLR     $TIMES           ;;INITIALIZE NUMBER OF ITERATIONS
002402  005037  001210                  CLR     $ESCAPE          ;;CLEAR THE ESCAPE ON ERROR ADDRESS
002406  112737  000001  001115          MOVB    #1,$ERMAX        ;;ALLOW ONE ERROR PER TEST
002414  012737  002414  001106          MOV     #.,$LPADR        ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
002422  012737  002422  001110          MOV     #.,$LPERR        ;;SETUP THE ERROR LOOP ADDRESS
                                        ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                        ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
002430  013746  000004                  MOV     @#ERRVEC,-(SP)   ;;SAVE ERROR VECTOR
002434  012737  002470  000004          MOV     #64$,@#ERRVEC    ;;SET UP ERROR VECTOR
002442  012737  177570  001140          MOV     #DSWR,SWR        ;;SETUP FOR A HARDWARE SWICH REGISTER
002450  012737  177570  001142          MOV     #DDISP,DISPLAY   ;;AND A HARDWARE DISPLAY REGISTER
002456  022777  177777  176454          CMP     #-1,@SWR         ;;TRY TO REFERENCE HARDWARE SWR
002464  001012                          BNE     66$              ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                 ;;AND  THE HARDWARE SWR IS NOT = -1
002466  000403                          BR      65$              ;;BRANCH IF NO TIMEOUT
002470  012716  002476          64$:    MOV     #65$,(SP)        ;;SET UP FOR TRAP RETURN
002474  000002                          RTI
002476  012737  000176  001140  65$:    MOV     #SWREG,SWR       ;;POINT TO SOFTWARE SWR
002504  012737  000174  001142          MOV     #DISPREG,DISPLAY
002512  012637  000004          66$:    MOV     (SP)+,@#ERRVEC   ;;RESTORE ERROR VECTOR

002516  012700  001160                  MOV     #$REGAD,R0       ;FIRST ADDRESS
002522  005020                  1$:     CLR     (R0)+            ;CLEAR VARIABLE STORAGE
002524  022700  001212                  CMP     #$BELL,R0        ;DONE?
002530  001374                          BNE     1$               ;NO--BRANCH
002532  005227  177777                  INC     #-1              ;FIRST START ?
002536  001036                          BNE     2$               ;BR IF NOT
```

```
002540  023737  000042  000046          CMP     @#42,@#46         ;ACT11 AUTO MODE?
002546  001432                          BEQ     2$                ;YES, SKIP TITLE PRINTOUT
002550  104401  002556                  TYPE    ,68$              ;;TYPE ASCIZ STRING
002554  000427                          BR      67$               ;;GET OVER THE ASCIZ
                                ;;68$:  .ASCIZ  <15><12>/MD-11-DZRPZ-C, RP11E DRIVE POSITIONING TEST/
002634                          67$:
002634  004737  011254          2$:     JSR     PC,$TKINT         ;SETUP THE TTY KEYBOARD
                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
002640  005737  000042                  TST     @#42              ;;ARE WE RUNNING UNDER XXDP/ACT?
002644  001006                          BNE     69$               ;;BRANCH IF YES
002646  023727  001140  000176          CMP     SWR,#SWREG        ;;SOFTWARE SWITCH REG SELECTED?
002654  001005                          BNE     70$               ;;BRANCH IF NO
002656  104406                          GTSWR                     ;;GET SOFT-SWR SETTINGS
002660  000403                          BR      70$
002662  112737  000001  001134  69$:    MOVB    #1,$AUTOB         ;;SET AUTO-MODE INDICATOR
002670                          70$:
002670  012737  000377  001234          MOV     #377,TSTNMS       ;SELECT TESTS 0-7
002676  012700  001460                  MOV     #DFLT,R0          ;DEFAULT PARAMETERS POINTER
002702  012701  001632                  MOV     #PRMD,R1          ;TABLE POINTER
002706  010102                          MOV     R1,R2             ;STOP ADDRESS
002710  012021                  3$:     MOV     (R0)+,(R1)+       ;MOVE DEFAULT PARAMETERS INTO
002712  020002                          CMP     R0,R2             ;RUN TIME TABLES ** DONE?
002714  103775                          BLO     3$                ;NO--BRANCH
002716  000406                          BR      SRTINT            ;FINISH SETUP
002720  000005                  START5: RESET                     ;'CONTROL C' RESET
002722  004737  011254                  JSR PC,$TKINT
002726  012737  177777  001222          MOV     #-1,CNTRLC        ;SET 'CONTROL C' SWITCH
002734  012737  000340  000062  SRTINT: MOV     #PR7,TKVEC+2      ;SET TTY PRIORITY TO 7
002742  005037  177776                  CLR     PS                ;SET PRIORITY TO ZERO
002746  005037  001252                  CLR     DRVBAD            ;CLEAR OFFLINE/UNSAFE DRIVE BITS
002752  005037  001206                  CLR     $TIMES            ;INITIALIZE NUMBER OF ITERATIONS
002756  005037  001210                  CLR     $ESCAPE           ;CLEAR THE ESCAPE ON ERROR ADDRESS
002762  005037  001244                  CLR     BYPASS            ;CLEAR THE BYPASS ADDRESS
002766  012737  000001  001104          MOV     #1,$ICNT          ;PRESET ITERATION COUNT TO 1
002774  112737  000001  001115          MOVB    #1,$ERMAX         ;ALLOW ONE ERROR PER TEST
003002  012737  003002  001106          MOV     #.,$LPADR         ;INITIALIZE THE LOOP ADDRESS FOR SCOPE
003010  012737  003010  001110          MOV     #.,$LPERR         ;SETUP THE ERROR LOOP ADDRESS
003016  004737  017256                  JSR     PC,GETADR         ;CHECK RP11 ADDRESS
003022  004737  013642                  JSR     PC,RPINIT         ;FIND OUT WHICH DRIVES ARE ON SYSTEM
003026  012737  000312  001236          MOV     #202.,MAXCYL      ;ASSUME RP02'S
003034  005737  001240                  TST     DRVTYP            ;WHICH DRIVES ?
003040  001403                          BEQ     1$                ;BR IF THEY REALLY WERE RP02'S
003042  012737  000625  001236          MOV     #405.,MAXCYL      ;SET MAX CYLINDER FOR RP03'S
003050  013737  001236  001436  1$:     MOV     MAXCYL,PRMLMT+2   ;UPDATE 'FC' LIMIT
003056  013737  001236  001440          MOV     MAXCYL,PRMLMT+4   ;UPDATE 'LC' LIMIT
003064  005737  001264                  TST     STR20X            ;COMING FROM A 'CONTROL C' RESTART ?
003070  001002                          BNE     9$                ;BR IF NOT
003072  000137  003442                  JMP     START6            ;BYPASS 'LC' UPDATE AND DRIVE STATUS TYPEOUT
003076                          9$:
003076  013737  001236  001646          MOV     MAXCYL,LC1        ;SETUP MAX CYLINDER FOR TEST 1
003104  013737  001236  001662          MOV     MAXCYL,LC2        ;SETUP MAX CYLINDER FOR TEST 2
003112  013737  001236  001712          MOV     MAXCYL,LC4        ;SETUP MAX CYLINDER FOR TEST 4
003120  013737  001236  001726          MOV     MAXCYL,LC5        ;SETUP MAX CYLINDER FOR TEST 5
003126  013737  001236  001742          MOV     MAXCYL,LC6        ;SETUP MAX CYLINDER FOR TEST 6
003134  013737  001236  001756          MOV     MAXCYL,LC7        ;SETUP MAX CYLINDER FOR TEST 7
003142  013737  001236  001776          MOV     MAXCYL,LC11       ;SETUP MAX CYLINDER FOR TEST 11
```

```
003150  012737  000200  001676          MOV     #128.,LC3          ;ASSUME RP02
003156  005737  001240                  TST     DRVTYP             ;RP02 OR RP03 ?
003162  001403                          BEQ     2$                 ;BR IF RP02'S
003164  012737  000400  001676          MOV     #256.,LC3          ;CHANGE VALUE FOR RP03
003172  005227  177777          2$:     INC     #-1                ;FIRST START ?
003176  001404                          BEQ     10$                ;BR IF IT IS
003200  032777  000200  175732          BIT     #SW07,@SWR         ;INHIBIT THE STATUS TYPEOUT
003206  001115                          BNE     START6             ;BR IF YES
003210                          10$:
003210  104401  003216                  TYPE    ,65$               ;;TYPE ASCIZ STRING
003214  000412                          BR      64$                ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12><12>/DRIVE STATUS:/<15><12><12>
003242                          64$:
003242  005001                          CLR     R1                 ;CLEAR TABLE POINTER
003244  012702  000010                  MOV     #8.,R2             ;COUNTER
003250                          3$:
003250  010146                          MOV     R1,-(SP)           ;;SAVE R1 FOR TYPEOUT
                                                                   ;;TYPE DRIVE NUMBER
003252  104403                          TYPOS                      ;;GO TYPE--OCTAL ASCII
003254     001                          .BYTE   1                  ;;TYPE 1 DIGIT(S)
003255     000                          .BYTE   0                  ;;SUPPRESS LEADING ZEROS
003256  105761  001304                  TSTB    DRVSTA(R1)         ;CHECK DRIVE'S STATUS
003262  001450                          BEQ     7$                 ;DRIVE IS OFFLINE OR NON-EXISTENT
003264  100412                          BMI     4$                 ;DRIVE IS UNSAFE
003266  104401  003274                  TYPE    ,67$               ;;TYPE ASCIZ STRING
003272  000406                          BR      66$                ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  /     ONLINE/
003310                          66$:
003310  000411                          BR      5$                 ;
003312                          4$:
003312  104401  003320                  TYPE    ,69$               ;;TYPE ASCIZ STRING
003316  000406                          BR      68$                ;;GET OVER THE ASCIZ
                                ;;69$:  .ASCIZ  /     UNSAFE/
003334                          68$:
003334  136137  001314  001240  5$:     BITB    ATABIT(R1),DRVTYP  ;SEE WHICH DRIVE TYPE
003342  001010                          BNE     6$                 ;BR IF RP03
003344  104401  003352                  TYPE    ,71$               ;;TYPE ASCIZ STRING
003350  000404                          BR      70$                ;;GET OVER THE ASCIZ
                                ;;71$:  .ASCIZ  /  RP02
003362                          70$:
003362  000422                          BR      8$                 ;
003364                          6$:
003364  104401  003372                  TYPE    ,73$               ;;TYPE ASCIZ STRING
003370  000404                          BR      72$                ;;GET OVER THE ASCIZ
                                ;;73$:  .ASCIZ  /  RP03/
003402                          72$:
003402  000412                          BR      8$                 ;
003404                          7$:
003404  104401  003412                  TYPE    ,75$               ;;TYPE ASCIZ STRING
003410  000407                          BR      74$                ;;GET OVER THE ASCIZ
                                ;;75$:  .ASCIZ  /     OFFLINE/
003430                          74$:
003430  104401  001217          8$:     TYPE    ,$CRLF             ;CR-LF
003434  005201                          INC     R1                 ;INCREMENT TABLE POINTER
003436  005302                          DEC     R2                 ;DECREMENT THE COUNTER
003440  001303                          BNE     3$                 ;CONTINUE
```

```
003442  005037  001264          START6: CLR     STR20X          ;CLEAR MANUAL START FLAG
003446  005737  001222                  TST     CNTRLC          ;CONTROL "C" START/RESTART?
003452  001403                          BEQ     1$              ;NO--BRANCH
003454  004737  017610                  JSR     PC,@#GT.PRM     ;YES--GET PARAMETERS
003460  000416                          BR      5$
003462  005037  001232          1$:     CLR     DRVSEL          ;NO DRIVES SELECTED
003466  005000                          CLR     R0              ;DETERMINE THE DRIVES THAT
003470  012701  000001                  MOV     #1,R1           ;ARE AVAILABLE FOR TESTING
003474  105760  001304          2$:     TSTB    DRVSTA(R0)
003500  003403                          BLE     3$
003502  156037  001314  001232          BISB    ATABIT(R0),@#DRVSEL
003510  005200                  3$:     INC     R0
003512  106301                          ASLB    R1
003514  001367                          BNE     2$
003516                          5$:
003516  104401  003524                  TYPE    ,65$            ;;TYPE ASCIZ STRING
003522  000415                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12>/DRIVE(S) TO BE TESTED /
003556                          64$:
003556  005037  007656                  CLR     @#$ENDCT        ;DETERMINE PASSES TO MAKE AND
003562  005000                          CLR     R0              ;THE DRIVES TO BE TESTED
003564  013701  001232                  MOV     @#DRVSEL,R1     ;ANY DRIVES SELECTED?
003570  001010                          BNE     9$              ;YES--BRANCH
003572  104401  003600                  TYPE    ,67$            ;;TYPE ASCIZ STRING
003576  000403                          BR      66$             ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  /NONE/
003606                          66$:
003606  000137  007474                  JMP     @#$EOP          ;GO TO END OF PROGRAM
003612  005737  001222          9$:     TST     CNTRLC          ;CONTROL "C" START/RESTART ?
003616  001010                          BNE     6$              ;BR IF NOT
003620  005737  000042                  TST     42              ;UNDER MONITOR CONTROL ?
003624  001405                          BEQ     6$              ;BR IF NOT
003626  122737  000007  000041          CMPB    #7,41           ;LOADED BY RP02/RP03 ?
003634  001001                          BNE     6$              ;BR IF NOT
003636  006201                          ASR     R1              ;EXCLUDE DRIVE 0 FROM TESTING
003640  006201                  6$:     ASR     R1              ;REPORT THE DRIVES TO BE TESTED
003642  103013                          BCC     7$
003644  005237  007656                  INC     @#$ENDCT        ;GIVE THIS DRIVE A PASS
003650  010046                          MOV     R0,-(SP)        ;;SAVE R0 FOR TYPEOUT
003652  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
003654     001                          .BYTE   1               ;;TYPE 1 DIGIT(S)
003655     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
003656  005701                          TST     R1              ;MORE DRIVES?
003660  001406                          BEQ     8$              ;NO--BRANCH
003662  104401  003670                  TYPE    ,69$            ;;TYPE ASCIZ STRING
003666  000401                          BR      68$             ;;GET OVER THE ASCIZ
                                ;;69$:  .ASCIZ  ","
003672                          68$:
003672  005200                  7$:     INC     R0              ;INCREMENT DRIVE NUMBER
003674  000761                          BR      6$
003676  013737  007656  007650  8$:     MOV     @#$ENDCT,@#$EOPCT
003704  005037  001246          RSTRT1: CLR     CHKDRV          ;INIT. THE CHECK DRIVE KEY
003710  012737  000001  001250          MOV     #1,DRVMSK       ;START TO CHECK DESIRED DRIVES
003716  033737  001250  001232  RSTRT2: BIT     DRVMSK,DRVSEL   ;IS THIS DRIVE SELECTED?
003724  001011                  RSTRT3: BNE     DRVOK           ;YES--GO CHECK IF DRIVE IS READY FOR TESTING
003726  005237  001246          RESTART:INC     CHKDRV          ;MOVE  TO NEXT DRIVE NUMBER
```

```
003732  106337  001250                  ASLB    DRVMSK          ;POSITION THE MASK
003736  103367                          BCC     RSTRT2          ;BR IF MORE DRIVES
003740  043737  001252  001232          BIC     DRVBAD,DRVSEL   ;CLEAR SELECTION BITS FOR ANY OFFLINE/UNSAFE
                                                                ;DRIVES
003746  000756                          BR      RSTRT1          ;CONTINUE WITH CYCLE
003750  013702  001246          DRVOK:  MOV     @#CHKDRV,R2     ;PICKUP THE DRIVE NUMBER
003754  110237  001365                  MOVB    R2,DPB+1        ;LOAD DRIVE NUMBER
003760  104401  003766                  TYPE    ,65$            ;;TYPE ASCIZ STRING
003764  000411                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$.  .ASCIZ  <15><12><12>/TESTING DRIVE /
004010                          64$:
004010  010246                          MOV     R2,-(SP)        ;;SAVE R2 FOR TYPEOUT
004012  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
004014     001                          .BYTE   1               ;;TYPE 1 DIGIT(S)
004015     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
004016  104401  001217                  TYPE    $CRLF
004022  013700  001266                  MOV     RPADR,R0        ;RP11 ADDRESS
004026  016037  000012  001254          MOV     RPCA(R0),CYL,CR ;INITIALIZE CURRENT STORAGE
```

```
1899
1900                                    .SBTTL  **** TESTS ****
1901

        ;*;*;*\;*;*;*\;/*\;/*\;/*\;*\;/*\;/*\;/
```

```
004136  113737  001636  001371          MOVB    TK0,DPB+5          ;LOAD USER SPECIFIED TRACK
004144  012706  001100            TESTO:  MOV     #STACK,SP          ;SET UP STACK POINTER
004150  004737  014036                    JSR     PC,SEEKS           ;DO THE SPECIFIED SEEK
004154  000004                    EXITO:  SCOPE                      ;LOOP

                                  ;;*******************************************************
                                  ;*TEST 1          SEEK/SEEK TEST

                                  ;*      THIS TEST WILL CAUSE THE DRIVE TO EXECUTE A FORWARD SEEK
                                  ;*      CYCLE TO "LC", "LT", "LS" FOLLOWED BY A REVERSE SEEK CYCLE TO
                                  ;*      "FC", "FT", "FS". AT THE COMPLETION OF EACH SEEK, THE PROPER
                                  ;*      INDICATORS ARE EXAMINED TO INSURE PROPER OPERATION.

                                  ;;*******************************************************
004156                            TST1:
004156  033737  001326  001234          BIT     @#BITS+(1*2),TSTNMS ;DO THIS TEST?
004164  001002                          BNE     64$                ;YES--BRANCH
004166  000137  004314                  JMP     TST2               ;NO--GO TO THE NEXT TEST
004172  013737  001642  001206    64$:  MOV     @#RPT1,$TIMES      ;GET THE ITERATION COUNT
004200  012737  004262  001106          MOV     #TEST1,@#$LPADR
004206  012737  004156  001110          MOV     #TST1,@#$LPERR     ;SETUP THE ERROR LOOP ADDRESS
004214  012737  000001  001102          MOV     #1,@#$TSTNM        ;SET UP TEST NUMBER AND
                                                                   ;CLEAR THE ERROR FLAG ($ERFLG
004222  013777  001102  174712          MOV     $TSTNM,@DISPLAY    ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
004230  012737  004312  001244          MOV     #EXIT1,BYPASS      ;SETUP BYPASS ADDRESS
004236  013700  001266                  MOV     RPADR,R0           ;RP11 ADDRESS
004242  112737  000011  001364          MOVB    #SEEK,DPB          ;LOAD SEEK OPCODE
004250  005037  001370                  CLR     DPB+4              ;CLEAR SECTOR/TRACK ADDRESS
004254  113737  001652  001371          MOVB    TK1,DPB+5          ;LOAD USER SUPPLIED TRACK ADDRESS
004262  012706  001100            TEST1:  MOV     #STACK,SP          ;SET THE STACK POINTER
004266  013737  001644  001366          MOV     FC1,DPB+2          ;STARTING CYLINDER
004274  004737  014036                  JSR     PC,SEEKS           ;SEEK TO FC
004300  013737  001646  001366          MOV     LC1,DPB+2          ;ENDING CYLINDER
004306  004737  014036                  JSR     PC,SEEKS           ;SEEK TO LC
004312  000004                    EXIT1:  SCOPE                      ;LOOP

                                  ;;*******************************************************
                                  ;*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
                                  ;*      INSURE PROPER OPERATION.

                                  ;;*******************************************************
004314                            TST2:
004314  033737  001330  001234          BIT     @#BITS+(2*2),TSTNMS ;DO THIS TEST?
004322  001002                          BNE     64$                ;YES--BRANCH
004324  000137  004506                  JMP     TST3               ;NO--GO TO THE NEXT TEST
004330  013737  001656  001206    64$:  MOV     @#RPT2,$TIMES      ;GET THE ITERATION COUNT
004336  012737  004420  001106          MOV     #TEST2,@#$LPADR
004344  012737  004314  001110          MOV     #TST2,@#$LPERR     ;SETUP THE ERROR LOOP ADDRESS
```

```
004352  012737  000002  001102          MOV     #2,@#$TSTNM       ;SET UP TEST NUMBER AND
                                                                  ;CLEAR THE ERROR FLAG ($ERFLG)
004360  013777  001102  174554          MOV     $TSTNM,@DISPLAY   ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
004366  012737  004504  001244          MOV     #EXIT2,BYPASS     ;SETUP BYPASS ADDRESS
004374  013700  001266                  MOV     RPADR,R0          ;RP11 ADDRESS
004400  112737  000011  001364          MOVB    #SEEK,DPB         ;LOAD SEEK OPCODE
004406  005037  001370                  CLR     DPB+4             ;CLEAR SECTOR/TRACK ADDRESS
004412  113737  001666  001371          MOVB    TK2,DPB+5         ;LOAD USER SUPPLIED TRACK ADDRESS
004420  012706  001100          TEST2:  MOV     #STACK,SP         ;SET UP THE STACK POINTER
004424  013737  001660  001366          MOV     FC2,DPB+2         ;STARTING CYLINDER
004432  004737  014036          INCSK:  JSR     PC,SEEKS          ;SEEK TO FC
004436  063737  001664  001366          ADD     IC2,DPB+2          ;MOVE TO NEXT CYLINDER
004444  023737  001662  001366          CMP     LC2,DPB+2          ;OUT OF CYLINDERS?
004452  002367                          BGE     INCSK             ;NO--BRANCH
004454  013737  001662  001366          MOV     LC2,DPB+2          ;LOAD LC
004462  004737  014036          DECSK:  JSR     PC,SEEKS          ;SEEK TO LC
004466  163737  001664  001366          SUB     IC2,DPB+2
004474  023737  001660  001366          CMP     FC2,DPB+2
004502  003767                          BLE     DECSK
004504  000004                  EXIT2:  SCOPE                     ;LOOP

                                ;;*******************************************************
                                ;*TEST 3         STEPPING SEEK TEST

                                ;*      THIS TEST WILL COMMAND SEEK CYCLES TO CYLINDER 0,1,2,4,
                                ;*      8, 16, 32, 64, AND 128 (AND 256 IF HW RP03)   AT THE
                                ;*      COMPLETION OF EACH SEEK COMMAND, THE PROPER INDICATORS ARE
                                ;*      EXAMINED TO VERIFY PROPER OPERATION.

                                ;;*******************************************************
004506                          TST3:
004506  033737  001332  001234          BIT     @#BITS+(3*2),TSTNMS ;DO THIS TEST?
004514  001002                          BNE     64$               ;YES--BRANCH
004516  000137  004656                  JMP     TST4              ;NO--GO TO THE NEXT TEST
004522  013737  001672  001206  64$:    MOV     @#RPT3,$TIMES     ;GET THE ITERATION COUNT
004530  012737  004612  001106          MOV     #TEST3,@#$LPADR
004536  012737  004506  001110          MOV     #TST3,@#$LPERR    ;SETUP THE ERROR LOOP ADDRESS
004544  012737  000003  001102          MOV     #3,@#$TSTNM       ;SET UP TEST NUMBER AND
                                                                  ;CLEAR THE ERROR FLAG ($ERFLG)
004552  013777  001102  174362          MOV     $TSTNM,@DISPLAY   ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
004560  012737  004654  001244          MOV     #EXIT3,BYPASS     ;SETUP BYPASS ADDRESS
004566  013700  001266                  MOV     RPADR,R0          ;RP11 ADDRESS
004572  112737  000011  001364          MOVB    #SEEK,DPB         ;LOAD SEEK OPCODE
004600  005037  001370                  CLR     DPB+4             ;CLEAR SECTOR/TRACK ADDRESS
004604  113737  001702  001371          MOVB    TK3,DPB+5         ;LOAD USER SUPPLIED TRACK ADDRESS
004612  012706  001100          TEST3:  MOV     #STACK,SP         ;SET UP THE STACK
004616  013737  001674  001366          MOV     FC3,DPB+2          ;FC
004624  004737  014036                  JSR     PC,SEEKS          ;SEEK TO FC
004630  013701  001700                  MOV     IC3,R1            ;CYLINDER 1
004634  010137  001366          1$:     MOV     R1,DPB+2          ;DESIRED CYLINDER
004640  004737  014036                  JSR     PC,SEEKS          ;SEEK TO NC
004644  006301                          ASL     R1                ;MOVE TO NEXT CYLINDER
004646  020137  001676                  CMP     R1,LC3            ;DONE?
004652  003770                          BLE     1$                ;NO--LOOP
004654  000004                  EXIT3:  SCOPE
```

```
                                        ;;*****************************************************************
                                        ;*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 VERIFY PROPER OPERATION.
                                        ;*
                                        ;;*****************************************************************
004656                                  TST4:
004656  033737  001334  001234                  BIT     @#BITS+<4*2>,TSTNMS ;DO THIS TEST?
004664  001002                                  BNE     64$                 ;YES--BRANCH
004666  000137  005220                          JMP     TST5                ;NO--GO TO THE NEXT TEST
004672  013737  001706  001206          64$:    MOV     @#RPT4,$TIMES       ;GET THE ITERATION COUNT
004700  012737  004776  001106                  MOV     #TEST4,@#$LPADR
004706  012737  004656  001110                  MOV     #TST4,@#$LPERR      ;SETUP THE ERROR LOOP ADDRESS
004714  012737  000004  001102                  MOV     #4,@#$TSTNM         ;SET UP TEST NUMBER AND
                                                                            ;CLEAR THE ERROR FLAG ($ERFLG)
004722  013777  001102  174212                  MOV     $TSTNM,@DISPLAY     ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
004730  012737  005216  001244                  MOV     #EXIT4,BYPASS       ;SETUP BYPASS ADDRESS
004736  013700  001266                          MOV     RPADR,R0            ;RP11 ADDRESS
004742  112737  000011  001364                  MOVB    #SEEK,DPB           ;LOAD SEEK OPCODE
004750  005037  001370                          CLR     DPB+4               ;CLEAR SECTOR/TRACK ADDRESS
004754  113737  001716  001371                  MOVB    TK4,DPB+5           ;LOAD USER SUPPLIED TRACK ADDRESS
004762  005002                                  CLR     R2                  ;CLEAR STALL SWITCH (NO STALL)
004764  032777  000002  174146                  BIT     #SW1,@SWR           ;STALL REQUIRED?
004772  001401                                  BEQ     TEST4               ;NO--BRANCH
004774  005102                                  COM     R2                  ;YES--SET SWITCH
004776  012706  001100                  TEST4:  MOV     #STACK,SP           ;SET UP THE STACK POINTER
005002  013701  001710                          MOV     FC4,R1              ;SET NC TO FC
005006  005037  001274                          CLR     STALLO              ;START AT ZERO IF STALLS REQUIRED
005012  010137  001366                  1$:     MOV     R1,DPB+2            ;NC
005016  004737  014036                          JSR     PC,SEEKS            ;SEEK TO NC
005022  005702                                  TST     R2                  ;STALL?
005024  001403                                  BEQ     2$                  ;NO--BRANCH
005026  004037  015576                          JSR     R0,STALL            ;YES-GO TO STALL ROUTINE
005032  001274                                  .WORD   STALLO              ;TIME POINTER
005034  013737  001710  001366          2$:     MOV     FC4,DPB+2           ;FC
005042  004737  014036                          JSR     PC,SEEKS            ;SEEK TO FC
005046  005702                                  TST     R2                  ;STALL?
005050  001413                                  BEQ     3$                  ;NO--BRANCH
005052  004037  015576                          JSR     R0,STALL            ;YES--GO TO STALL ROUTINE
005056  001274                                  .WORD   STALLO              ;TIME POINTER
005060  005237  001274                          INC     STALLO              ;UPDATE THE TIME
005064  023737  001302  001274                  CMP     MXSTAL,STALLO       ;TIME TO BIG?
005072  003347                                  BGT     1$                  ;NO--BRANCH
005074  005037  001274                          CLR     STALLO              ;YES--START OVER AT ZERO
005100  063701  001714                  3$:     ADD     IC4,R1              ;MOVE TO NEXT CYLINDER
005104  020137  001712                          CMP     R1,LC4              ;LAST CYLINDER COMPLETED?
005110  003740                                  BLE     1$                  ;NO--BRANCH
005112  013701  001712                          MOV     LC4,R1              ;SET NC TO LC
005116  010137  001366                  4$:     MOV     R1,DPB+2            ;NC
005122  004737  014036                          JSR     PC,SEEKS            ;SEEK TO NC
005126  005702                                  TST     R2                  ;STALL?
005130  001403                                  BEQ     5$                  ;NO--BRANCH
```

```
005132  004037  015576                   JSR     R0,STALL          ;YES--GO TO STALL ROUTINE
005136  001274                           .WORD   STALLO            ;TIME POINTER
005140  013737  001712  001366  5$:      MOV     LC4,DPB+2         ;LC
005146  004737  014036                   JSR     PC,SEEKS          ;SEEK TO LC
005152  005702                           TST     R2                ;STALL?
005154  001413                           BEQ     6$                ;NO--BRANCH
005156  004037  015576                   JSR     R0,STALL          ;YES--GO TO STALL ROUTINE
005162  001274                           .WORD   STALLO            ;TIME POINTER
005164  005237  001274                   INC     STALLO            ;UPDATE STALL TIME
005170  023737  001302  001274           CMP     MXSTAL,STALLO     ;TIME TOO BIG?
005176  003347                           BGT     4$                ;NO--BRANCH
005200  005037  001274                   CLR     STALLO            ;YES--SET STALL TIME BACK TO ZERO
005204  163701  001714          6$:      SUB     IC4,R1            ;NEXT CYLINDER
005210  020137  001710                   CMP     R1,FC4            ;DONE?
005214  002373                           BGE     6$                ;NO--BRANCH
005216  000004                  EXIT4:   SCOPE                     ;LOOP

                                ;*****************************************************************
                                ;*TEST 5         CONVERGING/DIVERGING SEEK TEST

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

                                ;*****************************************************************
005220                          TST5:
005220  033737  001336  001234           BIT     @#BITS+(5*2),TSTNMS ;DO THIS TEST?
005226  001002                           BNE     64$               ;YES--BRANCH
005230  000137  005406                   JMP     TST6              ;NO--GO TO THE NEXT TEST
005234  013737  001722  001206  64$:     MOV     @#RPT5,$TIMES     ;GET THE ITERATION COUNT
005242  012737  005324  001106           MOV     #TEST5,@#$LPADR
005250  012737  005220  001110           MOV     #TST5,@#$LPERR    ;SETUP THE ERROR LOOP ADDRESS
005256  012737  000005  001102           MOV     #5,@#$TSTNM       ;SET UP TEST NUMBER AND
                                                                  ;CLEAR THE ERROR FLAG (SERFLG
005264  013777  001102  173650           MOV     $TSTNM,@DISPLAY   ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
005272  012737  005404  001244           MOV     #EXIT5,BYPASS     ;SETUP BYPASS ADDRESS
005300  013700  001266                   MOV     RPADR,R0          ;RP11 ADDRESS
005304  112737  000011  001364           MOVB    #SEEK,DPB         ;LOAD SEEK OPCODE
005312  005037  001370                   CLR     DPB+4             ;CLEAR SECTOR TRACK ADDRESS
005316  113737  001732  001371           MOVB    TK5,DPB+5         ;LOAD USER SUPPLIED TRACK ADDRESS
005324  012706  001100          TEST5:   MOV     #STACK,SP
005330  013701  001724                   MOV     FC5,R1            ;START NC1 AT FC
005334  013702  001726                   MOV     LC5,R2            ;START NC2 AT LC
005340  010137  001366          1$:      MOV     R1,DPB+2          ;NC1
005344  004737  014036                   JSR     PC,SEEKS          ;SEEK TO NC1
005350  010237  001366                   MOV     R2,DPB+2          ;NC2
005354  004737  014036                   JSR     PC,SEEKS          ;SEEK TO NC1
005360  063701  001730                   ADD     IC5,R1            ;NEXT NC1
005364  163702  001730                   SUB     IC5,R2            ;NEXT NC2
005370  020137  001726                   CMP     R1,LC5            ;DONE?
005374  003003                           BGT     EXIT5             ;YES--BRANCH
```

```
005376  020237  001724                    CMP     R2,FC5          ;?
005402  002356                            BGE     1$              ;NO--BRANCH
005404  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 UNITL "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 INSURE PROPER OPERATION.

                                  ;*****************************************************************
005406                            TST6:
005406  033737  001340  001234            BIT     @#BITS+<6*2>,TSTNMS ;DO THIS TEST?
005414  001002                            BNE     64$             ;YES--BRANCH
005416  000137  005640                    JMP     TST7            ;NO--GO TO THE NEXT TEST
005422  013737  001736  001206    64$:    MOV     @#RPT6,$TIMES   ;GET THE ITERATION COUNT
005430  012737  005512  001106            MOV     #TEST6,@#$LPADR
005436  012737  005406  001110            MOV     #TST6,@#$LPERR  ;SETUP THE ERROR LOOP ADDRESS
005444  012737  000006  001102            MOV     #6,@#$TSTNM     ;SET UP TEST NUMBER AND
                                                                  ;CLEAR THE ERROR FLAG ($ERFLG
005452  013777  001102  173462            MOV     $TSTNM,@DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
005460  012737  005636  001244            MOV     #EXIT6,BYPASS   ;SETUP BYPASS ADDRESS
005466  013700  001266                    MOV     RPADR,R0        ;RP11 ADDRESS
005472  112737  000011  001364            MOVB    #SEEK,DPB       ;LOAD SEEK OPCODE
005500  005037  001370                    CLR     DPB+4           ;CLEAR SECTOR/TRACK ADDRESS
005504  113737  001746  001371            MOVB    TK6,DPB+5       ;LOAD USER SUPPLIED TRACK ADDRESS
005512  012706  001100            TEST6:  MOV     #STACK,SP       ;SETUP STACK
005516  013701  001740                    MOV     FC6,R1          ;PICKUP "FC"
005522  013702  001742                    MOV     LC6,R2          ;FORM LAST CYLINDER THAT
005526  162702  000005                    SUB     #5,R2           ;IS AVAILABLE FOR TESTING
005532  020102                    1$:     CMP     R1,R2           ;LAST CYLINDER
005534  003040                            BGT     EXIT6           ;YES--BRANCH
005536  010137  001366                    MOV     R1,DPB+2        ;NC
005542  004737  014036                    JSR     PC,SEEKS        ;SEEK TO NC
005546  062737  000004  001366            ADD     #4,DPB+2        ;NC+4
005554  004737  014036                    JSR     PC,SEEKS        ;SEEK TO NC+4
005560  162737  000003  001366            SUB     #3,DPB+2        ;NC+1
005566  004737  014036                    JSR     PC,SEEKS        ;SEEK TO NC+1
005572  062737  000002  001366            ADD     #2,DPB+2        ;NC+3
005600  004737  014036                    JSR     PC,SEEKS        ;SEEK TO NC+3
005604  162737  000001  001366            SUB     #1,DPB+2        ;NC+2
005612  004737  014036                    JSR     PC,SEEKS        ;SEEK TO NC+2
005616  062737  000003  001366            ADD     #3,DPB+2        ;NC+5
005624  004737  014036                    JSR     PC,SEEKS        ;SEEK TO NC+5
005630  063701  001744                    ADD     IC6,R1
005634  000736                            BR      1$
005636  000004                    EXIT6:  SCOPE                   ;LOOP

                                  ;*****************************************************************
                                  ;*TEST 7         CYLINDER ADDRESSING TEST

                                  ;*      THIS TEST WILL CAUSE THE DRIVE TO SEEK FROM EACH CYLINDER TO EVER
```

```
                                        ;*      OTHER CYLINDER BETWEEN CYLINDERS "FC" AND "LC".  SEEK CYCLES ARE
                                        ;*      COMMANDED FROM CYLINDER "NC" TO CYLINDER "NC1" AND BACK TO CYLINDER
                                        ;*      "NC".  "NC" AND "NC1" START AT "FC".  "NC1" IS INCREMENTED BY "IC"
                                        ;*      UNTIL "NC1" IS EQUAL TO "LC". WHEN "NC1" IS EQUAL TO "LC", IT IS
                                        ;*      RESET TO "FC" AND "NC" IS INCREMENTED BY "IC".  THE CYCLE IS COMPLETE
                                        ;*      WHEN "NC" AND "NC1" BOTH EQUAL "LC".  AT THE COMPLETION OF EACH
                                        ;*      SEEK COMMAND, THE INDICATORS ARE EXAMINED TO VERIFY PROPER OPERATION.

                                        ;;*****************************************************************
005640                                  TST7:
005640  033737  001342  001234                  BIT     @#BITS+<7*2>,TSTNMS ;DO THIS TEST?
005646  001002                                  BNE     64$                 ;YES--BRANCH
005650  000137  006032                          JMP     TST10               ;NO--GO TO THE NEXT TEST
005654  013737  001752  001206          64$:    MOV     @#RPT7,$TIMES       ;GET THE ITERATION COUNT
005662  012737  005744  001106                  MOV     #TEST7,@#$LPADR
005670  012737  005640  001110                  MOV     #TST7,@#$LPERR      ;SETUP THE ERROR LOOP ADDRESS
005676  012737  000007  001102                  MOV     #7,@#$TSTNM         ;SET UP TEST NUMBER AND
                                                                            ;CLEAR THE ERROR FLAG (SERFLG)
005704  013777  001102  173230                  MOV     $TSTNM,@DISPLAY     ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
005712  012737  006030  001244                  MOV     #EXIT7,BYPASS       ;SETUP BYPASS ADDRESS
005720  013700  001266                          MOV     RPADR,R0            ;RP11 ADDRESS
005724  112737  000011  001364                  MOVB    #SEEK,DPB           ;SETUP FOR SEEKING
005732  005037  001370                          CLR     DPB+4               ;CLEAR TRACK/SECTOR STORAGE
005736  113737  001762  001371                  MOVB    TK7,DPB+5           ;LOAD USER SPECIFIED TRACK ADDRESS
005744  012706  001100                  TEST7:  MOV     #STACK,SP           ;LOAD STACK POINTER
005750  013701  001754                          MOV     FC7,R1              ;START NC AT FC
005754  013702  001754                          MOV     FC7,R2              ;START NC1 AT FC
005760  010137  001366                  1$:     MOV     R1,DPB+2            ;NC
005764  004737  014036                          JSR     PC,SEEKS            ;SEEK TO NC
005770  010237  001366                          MOV     R2,DPB+2            ;NC1
005774  004737  014036                          JSR     PC,SEEKS            ;SEEK TO NC1
006000  063702  001760                          ADD     IC7,R2              ;INCREMENT NC1
006004  023702  001756                          CMP     LC7,R2              ;EXCEEDED LC ?
006010  002363                                  BGE     1$                  ;BR IF NOT
006012  013702  001754                          MOV     FC7,R2              ;RESET NC1
006016  063701  001760                          ADD     IC7,R1              ;INCREMENT NC
006022  023701  001756                          CMP     LC7,R1              ;EXCEEDED LC ?
006026  002354                                  BGE     1$                  ;BR IF NOT
006030  000004                          EXIT7:  SCOPE                       ;LOOP ?

                                        ;;*****************************************************************
                                        ;*TEST 10        ONE CYLINDER SEEK TIMER TEST

                                        ;*      THIS TEST PERFORMS ONE CYLINDER SEEKS FROM CYLINDER 0 TO THE
                                        ;*      MAXIMUM CYLINDER FOR THE DRIVE (E.G. 202 FOR RP02'S AND 405 FOR
                                        ;*      RP03'S).  THE SEEK TIMES ARE MEASURED RELATIVE TO THE SECTOR
                                        ;*      PULSES FROM THE DRIVE.  THE TEST DISPLAYS THE FORWARD AND REVERSE
                                        ;*      TIMES.

                                        ;;*****************************************************************
006032                                  TST10:
006032  033737  001344  001234                  BIT     @#BITS+<10*2>,TSTNMS ;DO THIS TEST?
006040  001002                                  BNE     64$                 ;YES--BRANCH
006042  000137  006306                          JMP     TST11               ;NO--GO TO THE NEXT TEST
006046  013737  001766  001206          64$:    MOV     @#RPT10,$TIMES      ;GET THE ITERATION COUNT
006054  012737  006130  001106                  MOV     #TEST10,@#$LPADR
```

```
006062  012737  006032  001110          MOV     #TST10,@#$LPERR  ;SETUP THE ERROR LOOP ADDRESS
006070  012737  000010  001102          MOV     #10.,@#$TSTNM    ;SET UP TEST NUMBER AND
                                                                 ;CLEAR THE ERROR FLAG ($ERFLG
006076  013777  001102  173036          MOV     $TSTNM,@DISPLAY  ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
006104  012737  006304  001244          MOV     #EXIT10,BYPASS   ;SETUP BYPASS ADDRESS
006112  013700  001266                  MOV     RPADR,R0         ;RP11 ADDRESS
006116  112737  000011  001364          MOVB    #SEEK,DPB        ;OPCODE FOR THE TEST
006124  005037  001370                  CLR     DPB+4            ;TRACK ADDRESS
```

```
2299  006130  012706  001100        TEST10: MOV     #STACK,SP       ;RESTORE THE STACK POINTER
```

```
006134  004737  015314                  JSR     PC,RESTOR          ;DO A HOME SEEK
006140  012704  024040                  MOV     #FRWSTR,R4         ;FORWARD TIME STORAGE POINTER
006144  012737  000001  001366          MOV     #1,DPB+2           ;SEEK TO THIS CYLINDER
006152  004737  016526          1$:     JSR     PC,TIMSEK          ;START THE SEEK, AND TIME IT
006156  010324                          MOV     R3,(R4)+           ;STORE THE TIME
006160  005237  001366                  INC     DPB+2              ;INCREMENT THE CYLINDER ADDRESS
006164  023737  001366  001236          CMP     DPB+2,MAXCYL       ;AT THE MAXIMUM ?
006172  003767                          BLE     1$                 ;BR IF NOT
006174  013737  001236  001366          MOV     MAXCYL,DPB+2       ;SETUP TO SEEK BACK TO MINIMUM
006202  005337  001366                  DEC     DPB+2              ;SEEK TO CYLINDER
006206  012704  025514                  MOV     #RVSTR,R4          ;REVERSE TIME STORAGE POINTER
006212  004737  016526          2$:     JSR     PC,TIMSEK          ;START THE SEEK, AND TIME IT
006216  010324                          MOV     R3,(R4)+           ;STORE THE TIME
006220  005337  001366                  DEC     DPB+2              ;DECREMENT THE CYLINDER ADDRESS
006224  100372                          BPL     2$                 ;BR IF NOT AT MINIMUM CYLINDER
006226  104401  006234                  TYPE    ,65$               ;;TYPE ASCIZ STRING
006232  000416                          BR      64$                ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12><12>/ONE CYLINDER SEEK TIMES/
006270                          64$:
006270  013737  001236  006302          MOV     MAXCYL,3$          ;NUMBER OF SEEKS PERFORMED
006276  004037  015660                  JSR     R0,SORT            ;SORT AND TYPE OUT THE TIMES
006302  000000                  3$:     .WORD   0                  ;NUMBER OF TIMES GOES HERE
006304  000004                  EXIT10: SCOPE                      ;LOOP ?

                                ;;*******************************************************************
                                ;*TEST 11        SEEK-SEEK TIMER TEST

                                ;*      THIS TEST IS A GENERAL PURPOSE TIMING TEST WHICH PERFORMS 200
                                ;*      ITERATIONS BETWEEN THE CYLINDER IN 'FC' AND THE CYLINDER IN 'LC'.
                                ;*      BOTH FORWARD AND REVERSE SEEKS ARE TIMED AND DISPLAYED.  THE
                                ;*      SEEK TIMES DISPLAYED ARE RELATIVE TO THE SECTOR COUNTER.
                                ;*      'FC' DEFAULTS TO 0; 'LC' DEFAULTS TO 'MAXCYL' (202 IF RP02'S OR
                                ;*      405 IF RP03'S).

                                ;;*******************************************************************
006306                          TST11:
006306  033737  001346  001234          BIT     @#BITS+<11*2>,TSTNMS ;DO THIS TEST?
006314  001002                          BNE     64$                ;YES--BRANCH
006316  000137  006620                  JMP     TST12              ;NO--GO TO THE NEXT TEST
006322  013737  001772  001206  64$:    MOV     @#RPT11,$TIMES     ;GET THE ITERATION COUNT
006330  012737  006404  001106          MOV     #TEST11,@#$LPADR
006336  012737  006306  001110          MOV     #TST11,@#$LPERR    ;SETUP THE ERROR LOOP ADDRESS
006344  012737  000011  001102          MOV     #11,@#$TSTNM       ;SET UP TEST NUMBER AND
                                                                   ;CLEAR THE ERROR FLAG ($ERFLG)
006352  013777  001102  172562          MOV     $TSTNM,@DISPLAY    ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
006360  012737  006616  001244          MOV     #EXIT11,BYPASS     ;SETUP BYPASS ADDRESS
006366  013700  001266                  MOV     RPADR,R0           ;RP11 ADDRESS
006372  012737  000011  001364          MOV     #SEEK,DPB          ;OPCODE FOR THE TEST
006400  005037  001370                  CLR     DPB+4              ;CLEAR THE TRACK ADDRESS STORAGE
006404  012706  001100          TEST11: MOV     #STACK,SP          ;RESTORE THE STACK POINTER
006410  004737  015314                  JSR     PC,RESTOR          ;DO A HOME SEEK
006414  012701  000311                  MOV     #201.,R1           ;ITERATION COUNT
006420  012704  025514                  MOV     #RVSTR,R4          ;REVERSE TIME STORAGE POINTER
006424  012705  024040                  MOV     #FRWSTR,R5         ;FORWARD TIME STORAGE POINTER
006430  013737  001774  001366          MOV     FC11,DPB+2         ;POSITION THE DISK TO THE FIRST CYLINDER
006436  004737  016526                  JSR     PC,TIMSEK          ;DO THE SEEK
```

```
006442  013737  001776  001366  1$:     MOV     LC11,DPB+2      ;TIME TO THIS CYLINDER
006450  004737  016526                   JSR     PC,TIMSEK       ;START THE SEEK AND TIME IT
006454  010325                           MOV     R3,(R5)+        ;STORE THE TIME
006456  013737  001774  001366           MOV     FC11,DPB+2      ;MINIMUM CYLINDER
006464  004737  016526                   JSR     PC,TIMSEK       ;START THE SEEK AND TIME IT
006470  010324                           MOV     R3,(R4)+        ;STORE THE TIME
006472  005301                           DEC     R1              ;DECREMENT THE ITERATION COUNTER
006474  001362                           BNE     1$              ;BR IF NOT FINISHED
006476  104401  006504                   TYPE    ,65$            ;;TYPE ASCIZ STRING
006502  000421                           BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:   .ASCIZ  <15><12><12>/SEEK TIMES BETWEEN CYLINDERS /
006546                          64$:
006546  013746  001774                   MOV     FC11,-(SP)      ;PUT STARTING CYLINDER ADDRESS ON THE STACK
006552  004737  013250                   JSR     PC,$$B2D        ;CONVERT IT TO DECIMAL
006556  004737  013500                   JSR     PC,$$UPRS       ;TYPE IT
006562  104401  006570                   TYPE    ,67$            ;;TYPE ASCIZ STRING
006566  000402                           BR      66$             ;;GET OVER THE ASCIZ
                                ;;67$:   .ASCIZ  / & /
006574                          66$:
006574  013746  001776                   MOV     LC11,-(SP)      ;PUT ENDING CYLINDER ADDRESS ON THE STACK
006600  004737  013250                   JSR     PC,$$B2D        ;CONVERT IT TO DECIMAL
006604  004737  013500                   JSR     PC,$$UPRS       ;TYPE IT
006610  004037  015660                   JSR     R0,SORT         ;TYPE THE TIMES
006614  000310                           .WORD   200.
006616  000004                  EXIT11:  SCOPE                   ;LOOP ?


                                ;;*****************************************************************
                                ;*TEST 12        SEEK TIME GRAPH TEST

                                ;*      THIS TEST PRODUCES TWO SEEK TIME GRAPHS.  THE FIRST GRAPH IS
                                ;*      TYPED IS SWITCH <04> IS 0.  THIS GRAPH IS A PLOT OF SEEK TIME
                                ;*      AGAINST CYLINDERS SEEKED TO FROM CYLINDER 0.  THE CYLINDERS
                                ;*      PLOTTED ARE: 0,1,2,3,4,6,8,10,12,14,16,18,20,25...100,105...MAXCYL.
                                ;*      THE OTHER GRAPH WHICH IS PRODUCED (SWITCH <04> SET) IS TIME TO
                                ;*      SEEK TO ALL CYLINDERS FROM CYLINDER 0.
                                ;*
                                ;*      IN BOTH SEEK GRAPHS, ONLY THE FORWARD SEEK TIMES ARE MEASURED.
                                ;*
                                ;*      THIS TEST USES THE SECTOR COUNTER ('SOT'  IN THE RP11 AS THE
                                ;*      TIME BASE FOR THE MEASURMENTS. USE OF THE 'SOT' PROVIDES A
                                ;*      RESOLUTION OF 2.5 MS.

                                ;;*****************************************************************
006620                          TST12:
006620  033737  001350  001234           BIT     @#BITS+<12*2>,TSTNMS ;DO THIS TEST?
006626  001002                           BNE     64$             ;YES--BRANCH
006630  000137  007474                   JMP     $EOP            ;NO--GO TO THE END OF THE PROGRAM
006634  013737  002002  001206  64$:     MOV     @#RPT12,$TIMES  ;GET THE ITERATION COUNT
006642  012737  006716  001106           MOV     #TEST12,@#$LPADR
006650  012737  006620  001110           MOV     #TST12,@#$LPERR  ;SETUP THE ERROR LOOP ADDRESS
006656  012737  000012  001102           MOV     #12,@#$TSTNM    ;SET UP TEST NUMBER AND
                                                                 ;CLEAR THE ERROR FLAG ($ERFLG)
006664  013777  001102  172250           MOV     $TSTNM,@DISPLAY  ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
006672  012737  007472  001244           MOV     #EXIT12,BYPASS  ;SETUP BYPASS ADDRESS
006700  013700  001266                   MOV     RPADR,R0        ;RP11 ADDRESS
```

```
006704  112737  000011  001364          MOVB    #SEEK,DPB          ;OPCODE FOR THE TEST
006712  005037  001370                  CLR     DPB+4              ;CLEAR THE TRACK ADDRESS STORAGE
006716  012706  001100          TEST12: MOV     #STACK,SP          ;INITIALIZE THE STACK POINTER
006722  004737  015314                  JSR     PC,RESTOR          ;DO A HOME SEEK
006726  012704  024040                  MOV     #FRWSTR,R4         ;STORE 'SEEK TIME' HERE
006732  005001                          CLR     R1                 ;CLEAR 'FC' STORAGE
006734  005002                          CLR     R2                 ;CLEAR 'NC' STORAGE
006736  010237  001366          1$:     MOV     R2,DPB+2           ;ADDRESS OF NEW CYLINDER 'NC'
006742  004737  016526                  JSR     PC,TIMSEK          ;SEEK TO 'NC' AND TIME IT
006746  010324                          MOV     R3,(R4)+           ;STORE MEASURED SEEK TIME
006750  010137  001366                  MOV     R1,DPB+2           ;'FC' ADDRESS
006754  004737  016526                  JSR     PC,TIMSEK          ;RETURN TO 'FC'
006760  005202                          INC     R2                 ;INCREMENT 'NC'
006762  023702  001236                  CMP     MAXCYL,R2          ;IS 'NC' AT MAXIMUM ?
006766  002363                          BGE     1$                 ;BR IF NOT

                                ;PLOT A GRAPH USING MEASURED SEEK TIMES

006770                          PLOT:
006770  104401  006776                  TYPE    ,65$               ;;TYPE ASCIZ STRING
006774  000412                          BR      64$                ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12><12>/SEEK TIME GRAPH/
007022                          64$:
007022  104401  007030                  TYPE    ,67$               ;;TYPE ASCIZ STRING
007026  000421                          BR      66$                ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  <15><12>/X AXIS - SEEK TIME - MILLI SECS
007072                          66$:
007072  104401  007100                  TYPE    ,69$               ;;TYPE ASCIZ STRING
007076  000423                          BR      68$                ;;GET OVER THE ASCIZ
                                ;;69$:  .ASCIZ  <15><12>/Y AXIS - CYLINDER SEEKED FROM 0/<15><12><12>
007146                          68$:
007146  104401  021473                  TYPE    ,GRPH1             ;TYPE THE GRAPH HEADING
007152  032777  000020  171760          BIT     #SW4,@SWR          ;TYPE COMPLETE GRAPH?
007160  001064                          BNE     CMPGRP             ;YES BRANCH; IF NOT, TYPE SMALL GRAPH

                                ;IN THE SMALL GRAPH, SEEK TIMES ARE PLOTTED ONLY FOR SELECTED CYLINDERS.
                                ;THE CYLINDERS PLOTTED ARE: 0,1,2,3,4,6,8,10,12,14,16,18,20,25,30
                                ;35,40...100,105...MAXCYL

007162  005000                  SMGRP:  CLR     R0
007164  005037  024040                  CLR     FRWSTR             ;FORCE CYL 0 - CYL 0 SEEK TO ZERO TIME
007170  104401  001217          1$:     TYPE    ,$CRLF             ;CR-LF
007174  010046                          MOV     R0,-(SP)           ;TYPE THE MARKERS
007176  104405                          TYPOS
007200  104401  007206                  TYPE    ,65$               ;;TYPE ASCIZ STRING
007204  000401                          BR      64$                ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  /-/
007210                          64$:
007210  010001                          MOV     R0,R1              ;FORM THE ADDRESS OF 'SEEK TIME'
007212  006301                          ASL     R1
007214  016103  024040                  MOV     FRWSTR(R1),R3      ;GET THE SEEK TIME
007220  004737  007432                  JSR     PC,PLTPT           ;GO PLOT IT
007224  022700  000004                  CMP     #4,R0              ;PLOTTED UPTO CYL 4?
007230  003402                          BLE     2$                 ;YES
007232  005200                          INC     R0                 ;INCREMENT BY 1
007234  000755                          BR      1$
```

```
007236  022700  000024          2$:   CMP     #20.,R0          ;PAST CYLINDER 20 ?
007242  003403                        BLE     3$               ;BR IF PAST 20
007244  062700  000002                ADD     #2,R0            ;INCREMENT BY 2
007250  000747                        BR      1$
007264  013746  001236          3$:   MOV     MAXCYL,-(SP)     ;CHECK FOR END
007270  042716  000007                BIC     #7,(SP)          ;CLEAR LOWER CYLINDER BITS
007274  022600                        CMP     (SP)+,R0         ;ALMOST MAX CYL ?
007264  003403                        BLE     4$               ;BR IF ALMOST AT MAX
007266  062700  000005                ADD     #5,R0            ;INCREMENT BY 5
007272  000736                        BR      1$
007274  023700  001236          4$:   CMP     MAXCYL,R0        ;PLOTTED ALL CYLS?
007300  001412                        BEQ     6$               ;BR IF DONE
007302  022737  000312  001236        CMP     #202.,MAXCYL     ;MAXCYL = 202. ?
007310  001003                        BNE     5$               ;BR IF NOT
007312  062700  000002                ADD     #2,R0            ;INCREMENT TO CYLINDER 202
007316  000724                        BR      1$
007320  062700  000005          5$:   ADD     #5,R0            ;INCREMENT TO CYLINDER 405
007324  000721                        BR      1$
007326  000137  007472          6$:   JMP     EXIT12           ;GO TO THE NEXT TEST

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

007332  005000                  CMPGRP: CLR   R0               ;INITIALIZE COUNT
007334  012701  177773                  MOV   #-5,R1           ;INITIALIZE COUNT FOR Y-AXIS MARKER
007340  012702  024040                  MOV   #FRWSTR,R2       ;INITIALIZE POINTER TO SEEK TIMES
007344  104401  001217                  TYPE  $CRLF
007350  000404                          BR    3$
007352  005201                  2$:     INC   R1               ;TYPE OUT Y-AXIS MARKER 'CYL #'
007354  001005                          BNE   4$               ;IF REQUIRED
007356  012701  177773                  MOV   #-5,R1           ;RESET INDEX
007362  010046                  3$:     MOV   R0,-(SP          ;TYPE 'CYL #' ON Y-AXIS
007364  104405                          TYPDS                  ;(IN DECIMAL)
007366  000402                          BR    5$
007370  104401  002037          4$:     TYPE  ,BLNK$6          ;6 BLANKS
007374                          5$:
007374  104401  007402                  TYPE  ,65$             ;;TYPE ASCIZ STRING
007400  000401                          BR    64$              ;;GET OVER THE ASCIZ
007402                          ;65$:   ASCIZ /-/
007404                          64$:
007404  012203                          MOV   (R2)+,R3         ;GET SEEK TIME
007406  004737  007432                  JSR   PC,PLTPT         ;GO PLOT THE POINT
007412  104401  001217                  TYPE  $CRLF
007416  005200                          INC   R0               ;ALL DONE?
007420  022700  001236                  CMP   #MAXCYL,R0       ;LAST CYLINDER ?
007424  001352                          BNE   2$               ;IF NOT, GO BACK
007426  000137  007472          6$:     JMP   EXIT12           ;GO TO THE NEXT TEST

                                ;ROUTINE TO PLOT THE SEEK TIME ON THE GRAPH.  ENTER WITH R3 CONTAINING
                                ;THE HORIZONTAL AXIS COORDINATE (I.E. SEEK TIME).

007432  010546                  PLTPT:  MOV   R5,-(SP          ;STORE R5
007434  012705  000032                  MOV   #26.,R5          ;HORIZONTAL AXIS INDEX
007440  162703  000372          1$:     SUB   #250.,R3         ;FIND OUT HOW MANY BLANKS TO
007444  002404                          BLT   2$               ;INSERT TO PLOT THE POINT
                                                               ;NOTE THE FIRST CELL = 0 MS
```

```
007446  104401  002044                    TYPE    ,BLNK$1         ;1 BLANK
007452  005305                            DEC     R5              ;AT THE END OF THE HORIZONTAL AXIS ?
007454  100371                            BPL     1$              ;BR IF NOT
007456                          2$:
007456  104401  007464                    TYPE    ,65$            ;;TYPE ASCIZ STRING
007462  000401                            BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:    .ASCIZ  <
007466                          64$:
007466  012605                            MOV     (SP)+,R5        ;RESTORE R5
007470  000207                            RTS     PC

007472  000004                  EXIT12:   SCOPE                   ;LOOP ?


                                .SBTTL  END OF PASS ROUTINE

                                ;;*****************************************************************
                                ;;*INCREMENT THE PASS NUMBER ($PASS)
                                ;;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
                                ;;*IF THERES A MONITOR GO TO IT
                                ;;*IF THERE ISN'T JUMP TO RESTART

007474                          $EOP:
007474  104401  007502                    TYPE    ,65$            ;;TYPE ASCIZ STRING
007500  000410                            BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:    .ASCIZ  <15><12><12>/END OF PASS/
007522                          64$:
007522  005737  001232                    TST     @#DRVSEL        ;ANY DRIVES SELECTED?
007526  001434                            BEQ     1$              ;NO--BRANCH
007530  104401  007536                    TYPE    ,67$            ;;TYPE ASCIZ STRING
007534  000405                            BR      66$             ;;GET OVER THE ASCIZ
                                ;;67$:    .ASCIZ  / ON DRIVE/
007550                          66$:
007550  013746  001246                    MOV     @#CHKDRV,-(SP)  ;;SAVE @#CHKDRV FOR TYPEOUT
007554  104403                            TYPOS                   ;;GO TYPE--OCTAL ASCII
007556     002                            .BYTE   2               ;;TYPE 2 DIGIT(S)
007557     000                            .BYTE   0               ;;SUPPRESS LEADING ZEROS
007560  104401  007566                    TYPE    ,69$            ;;TYPE ASCIZ STRING
007564  000412                            BR      68$             ;;GET OVER THE ASCIZ
                                ;;69$:    .ASCIZ  /   ERRORS DETECTED=/
007612                          68$:
007612  013746  001112                    MOV     @#$ERTTL,-(SP)  ;;SAVE @#$ERTTL FOR TYPEOUT
007616  104402                            TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS
007620  005037  001112          1$:       CLR     @#$ERTTL        ;ZERO ERROR TOTAL
007624  005037  001102                    CLR     $TSTNM          ;;ZERO THE TEST NUMBER
007630  005037  001206                    CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
007634  005237  001100                    INC     $PASS           ;;INCREMENT THE PASS NUMBER
007640  042737  100000  001100            BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
007646  005327                            DEC     (PC)+           ;;LOOP?
007650  000001                  $EOPCT:   .WORD   1
007652  003027                            BGT     $DOAGN          ;;YES
007654  012737                            MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
007656  000001                  $ENDCT:   .WORD   1
007660  007650                            $EOPCT
007662  104401  007670                    TYPE    ,65$            ;;TYPE ASCIZ STRING
007666  000407                            BR      64$             ;;GET OVER THE ASCIZ
```

```
                                        ;;65$:  .ASCIZ  <15><12>/END OF TEST/
007706                                  64$:
007706  104401  007736                          TYPE    ,$ENULL         ;TYPE NULL CHARACTER
007712  013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
007716  001405                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
007720  000005                          RESET                   ;;CLEAR THE WORLD
007722  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
007724  000240                          NOP                     ;;SAVE ROOM
007726  000240                          NOP                     ;;FOR
007730  000240                          NOP                     ;;ACT11
007732                          $DOAGN:
007732  000137                          JMP     @(PC)+          ;;RETURN
007734  003726                  $RTNAD: .WORD   RESTART
007736     377     377     000  $ENULL: .BYTE   -1,-1,0          ;;NULL CHARACTER STRING
            000042                      .EVEN

                                .SBTTL  *** SUBROUTINES ***
                                .SBTTL  ERROR HANDLER ROUTINE

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

007742                          $ERROR:
007742  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
007744  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
007750  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
007752  013777  001102  171162          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
007760  032777  002000  171152          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
007766  001402                          BEQ     1$              ;;NO - SKIP
007770  104401  001212                  TYPE    ,$BELL          ;;RING BELL
007774  005237  001112          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
010000  011637  001116                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
010004  162737  000002  001116          SUB     #2,$ERRPC
010012  117737  171100  001114          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
010020  032777  020000  171112          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
010026  001004                          BNE     20$             ;;SKIP TYPEOUTS
010030  004737  010114                  JSR     PC,TYPERR       ;;GO TO USER ERROR ROUTINE
010034  104401  001217                  TYPE    ,$CRLF
010040                          20$:
010040  005777  171074          2$:     TST     @SWR            ;;HALT ON ERROR
010044  100002                          BPL     3$              ;;SKIP IF CONTINUE
010046  000000                          HALT                    ;;HALT ON ERROR!
010050  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
010052  032777  001000  171060  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
010060  001402                          BEQ     4$              ;;BR IF NO
010062  013716  001110                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
010066  005737  001210          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
```

```
010072  001402                          BEQ     5$                      ;;BR IF NONE
010074  013716  001210                  MOV     $ESCAPE,(SP)            ;;FUDGE RETURN ADDRESS FOR ESCAPE
010100                          5$:
010100  023737  000042  000046          CMP     @#42,@#46                       ;ACT11 AUTO MODE?
010106  001001                          BNE     6$                              ;NO, RETURN
010110  000000                          HALT                                    ;HALT ON ERROR
010112  000002                  6$:     RTI                                     ;RETURN

                                ;;*****************************************************************
                                ;.SBTTL  TYPERR - TYPE ERROR ROUTINE
                                ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE
                                ;*WHICH ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR
                                ;*TABLE" ($ERRTB), AND REPORTS THE APPROPIATE INFORMATION
                                ;*CONCERNING THE ERROR.
                                ;CALL
                                ;       JSR     PC,@#TYPERR
                                ;       RETURN

010114  104412                  TYPERR: SAVREG                          ;SAVE R0-R5
010116  005037  001204                  CLR     $TMPO                   ;CLEAR LOCATION FOR TEST NUMBER
010122  113737  001102  001204          MOVB    $TSTNM,$TMPO            ;LOAD TEST NUMBER FOR TYPEOUT
010130  005000                          CLR     R0                      ;CLEAR R0 FOR ERROR NUMBER
010132  113700  001114                  MOVB    $ITEMB,R0               ;ERROR NUMBER
010136  005300                          DEC     R0                      ;FORM INDEX FOR ERROR TABLE
010140  006300                          ASL     R0
010142  006300                          ASL     R0
010144  006300                          ASL     R0
010146  062700  002046          1$:     ADD     #$ERRTB,R0              ;FORM ADDRESS
010152  012037  010166                  MOV     (R0)+,2$                ;GET ERROR MESSAGE (EM) POINTER
010156  001412                          BEQ     3$                      ;BRANCH IF THERE ISN'T ONE
010160  104401  001217                  TYPE    ,$CRLF                  ;"CARRIAGE RETURN - LINE FEED
010164  104401                          TYPE
010166  000000                  2$:     .WORD   0                       ;"EM" POINTER GOES HERE
010170  032777  000100  170742          BIT     #SW06,@SWR              ;TYPE THE REGISTERS INSTEAD OF THE NORMAL
                                                                        ;'DH' & 'DT' LINES ?
010176  001402                          BEQ     3$                      ;BR IF NOT
010200  012700  023532                  MOV     #ETSP,R0                ;CHANGE MESSAGE BLOCK
010204  012037  010220          3$:     MOV     (R0)+,4$                ;PICK UP DATA HEADER (DH) POINTER
010210  001404                          BEQ     5$                      ;BRANCH IF NONE
010212  104401  001217                  TYPE    ,$CRLF                  ;CARRIAGE RETURN-LINE FEED
010216  104401                          TYPE
010220  000000                  4$:     .WORD   0                       ;"DH" POINTER GOES HERE
010222  012001                  5$:     MOV     (R0)+,R1                ;PICKUP DATA TABLE (DT) POINTER
010224  001450                          BEQ     20$                     ;BRANCH IF NONE
010226  005005                          CLR     R5                      ;SET INDENT SWITCH
010230  012000                          MOV     (R0)+,R0                ;DATA FORMAT (DF) POINTER
010232  012002                          MOV     (R0)+,R2                ;NUMBER OF DH'S TO TYPE
010234  001441                          BEQ     17$                     ;BRANCH IF DH NUMBER IS 0
010236  005105                          COM     R5                      ;NO INDENT
010240  104401  001217                  TYPE    ,$CRLF                  ;CARRIAGE RETURN-LINE FEED
010244  112003                  10$:    MOVB    (R0)+,R3                ;NUMBER OF DATA WORDS TO TYPE
010246  112004                          MOVB    (R0)+,R4                ;AND HOW TO TYPE THEM
010250  006004                  11$:    ROR     R4                      ;OCTAL OR DECIMAL?
010252  103403                          BCS     12$                     ;DECIMAL--BRANCH
010254  013146                          MOV     @(R1)+,-(SP)            ;;SAVE @(R1)+ FOR TYPEOUT
010256  104402                          TYPOC                           ;;GO TYPE--OCTAL ASCII ALL DIGITS
```

```
010260  000402                          BR      13$
010262                          12$:
010262  013146                          MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
010264  104405                          TYPOS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
010266  005303                  13$:    DEC     R3              ;MORE NUMBERS TO TYPE?
010270  001403                          BEQ     14$             ;NO--BRANCH
010272  104401  002043                  TYPE    BLNKS2          ;YES--TYPE SEPARATORS
010276  000764                          BR      11$             ;LOOP
010300  005302                  14$:    DEC     R2              ;MORE DH'S?
010302  003421                          BLE     20$             ;NO--BRANCH
010304  104401  001217                  TYPE    $CRLF           ;YES--START A NEW LINE
010310  005105                          COM     R5              ;INDENT?
010312  001002                          BNE     15$             ;NO--BRANCH
010314  104401  002043                  TYPE    BLNKS2          ;YES--TYPE SPACES
010320  012037  010326          15$:    MOV     @(R0)+,16$      ;GET NEXT DH
010324  104401                          TYPE                    ;AND TYPE IT
010326  000000                  16$:    .WORD   0               ;DH POINTER GOES HERE
010330  104401  001217                  TYPE    $CRLF           ;CARRIAGE RETURN-LINE FEED
010334  005705                          TST     R5              ;INDENT?
010336  001342                          BNE     10$             ;NO--BRANCH
010340  104401  002043          17$:    TYPE    BLNKS2          ;YES--TYPE SPACES
010344  000737                          BR      10$             ;LOOP
010346  104413                  20$:    RESREG                  ;RESTORE R0-R5
010350  000207                          RTS     PC              ;RETURN


                                .SBTTL  TYPE ROUTINE

                                ;;*****************************************************************
                                ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                                ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                                ;*NOTE1:         $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                                ;*NOTE2:         $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                                ;*NOTE3:         $FILLC CONTAINS THE CHARACTER TO FILL AFTER
                                ;*
                                ;*CALL:
                                ;*1) USING A TRAP INSTRUCTION
                                ;*      TYPE    .MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                                ;*OR
                                ;*      TYPE
                                ;*      MESADR
                                ;*


010352  105737  001157          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
010356  100002                          BPL     1$              ;;BR IF YES
010360  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
010362  000407                          BR      3$              ;;LEAVE
010364  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
010366  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
010372  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
010374  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
010376  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
010400  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
010402  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
010406  000002                          RTI                     ;;RETURN
```

```
010410  122716  000011          4$:     CMPB    #HT,(SP          ;;BRANCH IF <HT>
010414  001430                          BEQ     8$
010416  122716  000200                  CMPB    #CRLF, SP)       ;;BRANCH IF NOT <CRLF>
010422  001006                          BNE     5$
010424  005726                          TST     (SP +            ;;POP  <CR><LF> EQUIV
010426  104401                          TYPE                     ;;TYPE A CR AND LF
010430  001217                          $CRLF
010432  105037  010566                  CLRB    $CHARCNT         ;;CLEAR CHARACTER COUNT
010436  000755                          BR      2$               ;;GET NEXT CHARACTER
010440  004737  010522          5$:     JSR     PC,$TYPEC        ;;GO TYPE THIS CHARACTER
010444  123726  001156          6$:     CMPB    $FILLC,(SP)+     ;;IS IT TIME FOR FILLER CHARS.?
010450  001350                          BNE     2$               ;;IF NO GO GET NEXT CHAR.
010452  013746  001154                  MOV     $NULL,-(SP)      ;;GET # OF FILLER CHARS. NEEDED
                                                                 ;;AND THE NULL CHAR.
010456  105366  000001          7$:     DECB    1 SP)            ;;DOES A NULL NEED TO BE TYPED?
010462  002770                          BLT     6$               ;;BR IF NO--GO POP THE NULL OFF OF STACK
010464  004737  010522                  JSR     PC,$TYPEC        ;;GO TYPE A NULL
010470  105337  010566                  DECB    $CHARCNT         ;;DO NOT COUNT AS A COUNT
010474  000770                          BR      7$               ;;LOOP

                                ;HORIZONTAL TAB PROCESSOR

010476  112716  000040          8$:     MOVB    #'  (SP)         ;;REPLACE TAB WITH SPACE
010502  004737  010522          9$:     JSR     PC,$TYPEC        ;;TYPE A SPACE
010506  132737  000007  010566          BITB    #7,$CHARCNT      ;;BRANCH IF NOT AT
010514  001372                          BNE     9$               ;;TAB STOP
010516  005726                          TST     (SP)+            ;;POP SPACE OFF STACK
010520  000724                          BR      2$               ;;GET NEXT CHARACTER
010522  105777  170422          $TYPEC: TSTB    @$TPS            ;;WAIT UNTIL PRINTER IS READY
010526  100375                          BPL     $TYPEC
010530  116677  000002  170414          MOVB    2(SP),@$TPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
010536  122766  000015  000002          CMPB    #CR,2(SP)        ;;IS CHARACTER A CARRIAGE RETURN?
010544  001003                          BNE     1$               ;;BRANCH IF NO
010546  105037  010566                  CLRB    $CHARCNT         ;;YES--CLEAR CHARACTER COUNT
010552  000406                          BR      $TYPEX           ;;EXIT
010554  122766  000012  000002  1$:     CMPB    #LF,2(SP)        ;;IS CHARACTER A LINE FEED?
010562  001402                          BEQ     $TYPEX           ;;BRANCH IF YES
010564  105227                          INCB    (PC)+            ;;COUNT THE CHARACTER
010566  000000                  $CHARCNT:.WORD  0                ;;CHARACTER COUNT STORAGE
010570  000207                  $TYPEX: RTS     PC

                                .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

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

```
                                        ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                        ;*$TYPOS OR $TYPOC
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPON                   ;;CALL FOR TYPEOUT
                                        ;*
                                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOC                   ;;CALL FOR TYPEOUT

010572  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
010576  116637  000001  011015          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
010604  112637  011017                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
010610  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
010614  000406                          BR      $TYPON
010616  112737  000001  011015  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
010624  112737  000006  011017          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
010632  112737  000005  011014  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
010640  010346                          MOV     R3,-(SP)        ;;SAVE R3
010642  010446                          MOV     R4,-(SP)        ;;SAVE R4
010644  010546                          MOV     R5,-(SP)        ;;SAVE R5
010646  113704  011017                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
010652  005404                          NEG     R4
010654  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
010660  110437  011016                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
010664  113704  011015                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
010670  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
010674  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
010676  006105                  1$      ROL     R5              ;;ROTATE MSB INTO "C"
010700  000404                          BR      3$              ;;GO DO MSB
010702  006105                  2$      ROL     R5              ;;FORM THIS DIGIT
010704  006105                          ROL     R5
010706  006105                          ROL     R5
010710  010503                          MOV     R5,R3
010712  006103                  3$      ROL     R3              ;;GET LSB OF THIS DIGIT
010714  105337  011016                  DECB    $OMODE          ;;TYPE THIS DIGIT?
010720  100016                          BPL     7$              ;;BR IF NO
010722  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
010726  001002                          BNE     4$              ;;TEST FOR 0
010730  005704                          TST     R4              ;;SUPPRESS THIS 0?
010732  001403                          BEQ     5$              ;;BR IF YES
010734  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
010736  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
010742  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
010746  110337  011012                  MOVB    R3,8$           ;;SAVE FOR TYPING
010752  104401  011012                  TYPE    8$              ;;GO TYPE THIS DIGIT
010756  105337  011014          7$:     DECB    $OCNT           ;;COUNT BY 1
010762  003347                          BGT     2$              ;;BR IF MORE TO DO
010764  002402                          BLT     6$              ;;BR IF DONE
010766  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
010770  000744                          BR      2$              ;;GO DO THE LAST DIGIT
010772  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
010774  012604                          MOV     (SP)+,R4        ;;RESTORE R4
010776  012603                          MOV     (SP)+,R3        ;;RESTORE R3
011000  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
```

```
011006  012616                          MOV     (SP)+,(SP)
011010  000002                          RTI                     ;;RETURN
011012     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
011013     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
011014     000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
011015     000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
011016  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE

                                .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

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

011020                          $TYPDS:
011020  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
011022  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
011024  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
011026  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
011030  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
011032  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
011036  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
011042  100004                          BPL     1$              ;;BR IF INPUT IS POS.
011044  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
011046  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
011054  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
011056  012703  011234                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
011062  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
011066  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
011070  016001  011224                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
011074  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
011076  002402                          BLT     4$              ;;BR IF DONE
011100  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
011102  000774                          BR      3$
011104  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
011106  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
011110  001002                          BNE     5$              ;;FALL THROUGH IF 0
011112  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
011114  100407                          BMI     7$              ;;BR IF YES
011116  106316                  5$:     ASLB    (SP)            ;;MSD?
011120  103003                          BCC     6$              ;;BR IF NO
011122  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
011130  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
011134  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
011140  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
011142  005720                          TST     (R0)+           ;;JUST INCREMENTING
011144  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
011150  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
011152  003002                          BGT     8$              ;;GO TO EXIT
011154  010502                          MOV     R5,R2           ;;GET THE LSD
```

```
011156  000764                          BR      6$                  ;;GO CHANGE TO ASCII
011160  105726                  8$:     TSTB    (SP)+               ;;WAS THE LSD THE FIRST NON-ZERO
011162  100003                          BPL     9$                  ;;BR IF NO
011164  116663  177777  177776          MOVB    -1(SP),-2(R3)       ;;YES--SET THE SIGN FOR TYPING
011172  105013                  9$:     CLRB    (R3)                ;;SET THE TERMINATOR
011174  012605                          MOV     (SP)+,R5            ;;POP STACK INTO R5
011176  012603                          MOV     (SP)+,R3            ;;POP STACK INTO R3
011200  012602                          MOV     (SP)+,R2            ;;POP STACK INTO R2
011202  012601                          MOV     (SP)+,R1            ;;POP STACK INTO R1
011204  012600                          MOV     (SP)+,R0            ;;POP STACK INTO R0
011206  104401  011234                  TYPE    $DBLK               ;;NOW TYPE THE NUMBER
011212  016666  000002  000004          MOV     2(SP),4(SP)         ;;ADJUST THE STACK
011220  012616                          MOV     (SP)+,(SP)
011222  000002                          RTI                         ;;RETURN TO USER
011224  023420                  $DTBL:  10000.
011226  001750                          1000.
011230  000144                          100.
011232  000012                          10.
011234  000004                  $DBLK:  .BLKW   4

                                .SBTTL  TTY INPUT ROUTINE

                                ;;*******************************************************************
                                .ENABL  LSB
011244  000000                  $TKCNT: .WORD   0                   ;;NUMBER OF ITEMS IN QUEUE
011246  000000                  $TKQIN: .WORD   0                   ;;INPUT POINTER
011250  000000                  $TKQOUT: .WORD  0                   ;;OUTPUT POINTER
011252  000001                  $TKQSRT: .BLKB  1                   ;;TTY KEYBOARD QUEUE
        011253                  $TKQEND=.
        011254                  .EVEN

                                ;*TK INITIALIZE ROUTINE
                                ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
                                ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
                                ;
                                ;*CALL:
                                ;*      JSR     PC,$TKINT
                                ;*      RETURN
                                ;
011254  005037  011244          $TKINT: CLR     $TKCNT              ;;CLEAR COUNT OF ITEMS IN QUEUE
011260  012737  011252  011246          MOV     #$TKQSRT,$TKQIN     ;;MOVE THE STARTING ADDRESS OF THE
011266  013737  011246  011250          MOV     $TKQIN,$TKQOUT      ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
011274  012737  011324  000060          MOV     #$TKSRV,@#TKVEC     ;;INITIALIZE THE KEYBOARD VECTOR
011302  012737  000200  000062          MOV     #200,@#TKVEC+2      ;;"BR" LEVEL 4
011310  005777  167632                  TST     @$TKB               ;;CLEAR DONE FLAG
011314  012777  000100  167622          MOV     #100,@$TKS          ;;ENABLE TTY KEYBOARD INTERRUPT
011322  000207                          RTS     PC                  ;;RETURN TO CALLER

                                ;*TK SERVICE ROUTINE
                                ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
                                ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
                                ;*IT IN THE QUEUE.
                                ;*IF THE CHARACTER IS A "CONTROL-C" (↑C) $TKINT IS CALLED AND
                                ;*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (START5
                                ;
011324  [illegible]             $TKSRV: MOVB    @$TKB,-(SP          ;;PICKUP THE CHARACTER
```

```
011330  042716  177600                  BIC     #^C177,(SP)       ;;STRIP THE JUNK
011334  021627  000003                  CMP     (SP),#3           ;;IS IT A CONTROL C?
011340  001007                          BNE     1$                ;;BRANCH IF NO
011342  104401  012463                  TYPE    ,$CNTLC           ;;TYPE A CONTROL-C (^C)
011346  004737  011254                  JSR     PC,$TKINT         ;;INIT THE KEYBOARD
011352  005726                          TST     (SP)+             ;;CLEAN UP STACK
011354  000137  002720                  JMP     STARTS            ;;CONTROL C RESTART
011360  021627  000007          1$:     CMP     (SP),#7           ;;IS IT A CONTROL G?
011364  001004                          BNE     2$                ;;BRANCH IF NO
011366  022737  000176  001140          CMP     #SWREG,SWR        ;;IS SOFT-SWR SELECTED?
011374  001500                          BEQ     6$                ;;GO TO SWR CHANGE

011376                          2$:
011376  022737  000001  011244          CMP     #1,$TKCNT         ;;IS THE QUEUE FULL?
011404  001004                          BNE     3$                ;;BRANCH IF NO
011406  104401  001212                  TYPE    ,$BELL            ;;RING THE TTY BELL
011412  005726                          TST     (SP)+             ;;CLEAN CHARACTER OFF OF STACK
011414  000451                          BR      5$                ;;EXIT
011416  021627  000023          3$:     CMP     (SP),#23          ;;IS IT A CONTROL-S?
011422  001021                          BNE     32$               ;;BRANCH IF NO
011424  005077  167514                  CLR     @$TKS             ;;DISABLE TTY KEYBOARD INTERRUPTS
011430  005726                          TST     (SP)+             ;;CLEAN CHAR OFF STACK
011432  105777  167506          31$:    TSTB    @$TKS             ;;WAIT FOR A CHAR
011436  100375                          BPL     31$               ;;LOOP UNTIL ITS THERE
011440  117746  167502                  MOVB    @$TKB,-(SP)       ;;GET THE CHARACTER
011444  042716  177600                  BIC     #^C177,(SP)       ;;MAKE IT 7-BIT ASCII
011450  022627  000021                  CMP     (SP)+,#21         ;;IS IT A CONTROL-Q?
011454  001366                          BNE     31$               ;;BRANCH IF NO
011456  012777  000100  167460          MOV     #100,@$TKS        ;;REENABLE TTY KEYBOARD INTERRUPTS
011464  000002                          RTI                       ;;RETURN
011466  005237  011244          32$:    INC     $TKCNT            ;;COUNT THIS CHARACTER
011472  021627  000140                  CMP     (SP),#140         ;;IS IT UPPER CASE?
011476  002405                          BLT     4$                ;;BRANCH IF YES
011500  021627  000175                  CMP     (SP),#175         ;;IS IT A SPECIAL CHAR?
011504  003002                          BGT     4$                ;;BRANCH IF YES
011506  042716  000040                  BIC     #40,(SP)          ;;MAKE IT UPPER CASE
011512  112677  177530          4$:     MOVB    (SP)+,@$TKQIN     ;;AND PUT IT IN QUEUE
011516  005237  011246                  INC     $TKQIN            ;;UPDATE THE POINTER
011522  023727  011246  011253          CMP     $TKQIN,#$TKQEND   ;;GO OFF THE END?
011530  001003                          BNE     5$                ;;BRANCH IF NO
011532  012737  011252  011246          MOV     #$TKQSRT,$TKQIN   ;;RESET THE POINTER
011540  000002                  5$:     RTI                       ;;RETURN

                                ;;*****************************************************************
                                ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                                ;*ROUTINE IS ENTERED  FROM THE TRAP HANDLER, AND WILL
                                ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
                                ;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
011542  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR        ;;IS THE SOFT-SWR SELECTED
011550  001124                          BNE     15$               ;;EXIT IF NOT
011552  105777  167366                  TSTB    @$TKS             ;;IS A CHAR WAITING?
011556  100121                          BPL     15$               ;;IF NOT, EXIT
011560  117746  167362                  MOVB    @$TKB,-(SP)       ;;YES
011564  042716  177600                  BIC     #^C177,(SP)       ;;MAKE IT  7-BIT ASCII
011570  021627  000007                  CMP     (SP),#7           ;;IS IT A CONTROL-G?
011574  001300                          BNE     2$                ;;IF NOT, PUT IT IN THE TTY QUEUE
```

```
                                                                        ;;AND EXIT

                                        ;;*****************************************************************
                                        ;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
                                        ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
                                        ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
011576  123727  001134  000001  6$:     CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
011604  001674                          BEQ     2$              ;;BRANCH IF YES
011606  005726                          TST     (SP)+           ;;CLEAR CONTROL-G OFF STACK
011610  004737  011254                  JSR     PC,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
011614  005077  167324                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
011620  112737  000001  001135          MOVB    #1,$INTAG       ;;SET INTERRUPT MODE INDICATOR

011626  104401  012475                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (↑G)
011632  104401  012502          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
011636  013746  000176                  MOV     $WREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
011642  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS.
011644  104401  012513                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
011650  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
011652  005046                          CLR     -(SP)           ;;THE NEW SWR
011654  105777  167264          7$:     TSTB    @$TKS           ;;CHAR THERE?
011660  100375                          BPL     7$              ;;IF NOT TRY AGAIN

011662  117746  167260                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
011666  042716  177600                  BIC     #↑C177,(SP)     ;;MAKE IT 7-BIT ASCII

011672  021627  000003                  CMP     (SP),#3         ;;IS IT A CONTROL-C?
011676  001015                          BNE     9$              ;;BRANCH IF NOT
011700  104401  012463                  TYPE    ,$CNTLC         ;;YES, ECHO CONTROL-C (↑C)
011704  062706  000006                  ADD     #6,SP           ;;CLEAN UP STACK
011710  123727  001135  000001          CMPB    $INTAG,#1       ;;REENABLE TTY KEYBOARD INTERRUPTS?
011716  001003                          BNE     8$              ;;BRANCH IF NO
011720  012777  000100  167216          MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
011726  000137  002720          8$:     JMP     START$          ;;CONTROL-C RESTART


011732  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
011736  001005                          BNE     10$             ;;BRANCH IF NOT
011740  104401  012470                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (↑U)
011744  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
011750  000737                          BR      19$             ;;LET'S TRY IT AGAIN


011752  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
011756  001022                          BNE     16$             ;;BRANCH IF NO
011760  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
011764  001403                          BEQ     11$             ;;BRANCH IF YES
011766  016677  000002  167144          MOV     2(SP),@SWR      ;;SAVE NEW SWR
011774  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
012000  104401  001217          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
012004  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
012012  001003                          BNE     15$             ;;BRANCH IF NOT
012014  012777  000100  167122          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
012022  000002                  15$:    RTI                     ;;RETURN
012024  004737  010522          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
012030  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
```

```
012034  002420                          BLT     18$             ;;BRANCH IF YES
012036  021627  000067                  CMP     (SP),#67        ;;CHAR > ??
012042  003015                          BGT     18$             ;;BRANCH IF YES
012044  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
012050  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
012054  001403                          BEQ     17$             ;;BRANCH IF YES
012056  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
012060  006316                          ASL     (SP)            ;;    CHAR OVER TO MAKE
012062  006316                          ASL     (SP)            ;;    ROOM FOR NEW ONE.
012064  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
012070  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
012074  000667                          BR      7$              ;;GET THE NEXT ONE
012076  104401  001216          18$:    TYPE    $QUES           ;;TYPE ?<CR><LF>
012102  000720                          BR      20$             ;;SIMULATE CONTROL-U
                                .DSABL  LSB


                                ;;*****************************************************************
                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*CALL:
                                ;*      RDCHR                   ;;GET A CHARACTER FROM THE QUEUE
                                ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
                                ;*                              ;;WITH PARITY BIT STRIPPED OFF
                                ;
012104  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
012106  016666  000004  000002          MOV     4(SP),2(SP)     ;;THE PS
012114  005066  000004                  CLR     4(SP)           ;;GET READY FOR A CHARACTER
012120  005046                          CLR     -(SP)           ;;PUT NEW PS ON STACK
012122  012746  012130                  MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
012126  000002                          RTI                     ;;POP NEW PC AND PS
012130                          64$:
012130  005737  011244          1$:     TST     $TKCNT          ;;WAIT ON A CHARACTER
012134  001775                          BEQ     1$
012136  005337  011244                  DEC     $TKCNT          ;;DECREMENT THE COUNTER
012142  117766  177102  000004          MOVB    @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
012150  005237  011250                  INC     $TKQOUT         ;;UPDATE THE POINTER
012154  023727  011250  011253          CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
012162  001003                          BNE     2$              ;;BRANCH IF NO
012164  012737  011252  011250          MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
012172  000002                  2$:     RTI                     ;;RETURN
                                ;;*****************************************************************
                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                ;*CALL:
                                ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
                                ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                ;*                              ;;TERMINATOR WILL BE A BYTE, OF ALL 0'S

012174  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
012176  005046                          CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
012200  012703  012430          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
012204  022703  012463          2$:     CMP     #$TTYIN+27.,R3  ;;BUFFER FULL?
012210  101456                          BLOS    4$              ;;BR IF YES
012212  104410                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
012214  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
012216  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
```

```
012222  001022                          BNE     5$              ;;BR IF NO
012224  005716                          TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
012226  001007                          BNE     6$              ;;BR IF NO
012230  112737  000134  012426          MOVB    #'\,9$          ;;TYPE A BACK SLASH
012236  104401  012426                  TYPE    ,9$
012242  012716  177777                  MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
012246  005303                  6$:     DEC     R3              ;;BACKUP BY ONE
012250  020327  012430                  CMP     R3,#$TTYIN      ;;STACK EMPTY?
012254  103434                          BLO     4$              ;;BR IF YES
012256  111337  012426                  MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
012262  104401  012426                  TYPE    ,9$             ;;GO TYPE
012266  000746                          BR      2$              ;;GO READ ANOTHER CHAR.
012270  005716                  5$:     TST     (SP)            ;;RUBOUT KEY SET?
012272  001406                          BEQ     7$              ;;BR IF NO
012274  112737  000134  012426          MOVB    #'\,9$          ;;TYPE A BACK SLASH
012302  104401  012426                  TYPE    ,9$
012306  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
012310  122713  000025          7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
012314  001003                          BNE     8$              ;;BR IF NO
012316  104401  012470                  TYPE    ,$CNTLU         ;;TYPE A CONTROL "U"
012322  000726                          BR      1$              ;;GO START OVER
012324  122713  000022          8$:     CMPB    #22,(R3)        ;;IS CHARACTER A "↑R"?
012330  001011                          BNE     3$              ;;BRANCH IF NO
012332  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
012334  104401  001217                  TYPE    ,$CRLF          ;;TYPE A "CR" & "LF"
012340  104401  012430                  TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
012344  000717                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
012346  104401  001216          4$:     TYPE    ,$QUES          ;;TYPE A '?'
012352  000712                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
012354  111337  012426          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
012360  104401  012426                  TYPE    ,9$
012364  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
012370  001305                          BNE     2$              ;;LOOP IF NOT RETURN
012372  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
012376  104401  001220                  TYPE    ,$LF            ;;TYPE A LINE FEED
012402  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
012404  012603                          MOV     (SP)+,R3        ;;RESTORE R3
012406  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
012410  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
012416  012766  012430  000004          MOV     #$TTYIN,4(SP)
012424  000002                          RTI                     ;;RETURN
012426     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
012427     000                          .BYTE   0               ;;TERMINATOR
012430  000033                  $TTYIN: .BLKB   27.             ;;RESERVE 27. BYTES FOR TTY INPUT
012463     136  006503  000012  $CNTLC: .ASCIZ  /↑C/<15><12>     ;;CONTROL "C"
012470  052536  005015     000  $CNTLU: .ASCIZ  /↑U/<15><12>     ;;CONTROL "U"
012475     136  006507  000012  $CNTLG: .ASCIZ  /↑G/<15><12>     ;;CONTROL "G"
012502  005015  053523  020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
012510  020075     000
012513     040  047040  053505  $MNEW:  .ASCIZ  /  NEW = /
012520  036440  000040

                                .SBTTL  SCOPE HANDLER ROUTINE

                                ;;*******************************************************************
                                ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
```

```
                                        ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG. DISPLAY 7:0
                                        ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY(15:08)
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW14=1          LOOP ON TEST
                                        ;*SW11=1          INHIBIT ITERATIONS
                                        ;*SW09=1          LOOP ON ERROR
                                        ;*CALL
                                        ;*      SCOPE           ;;SCOPE=IOT

012524                                  $SCOPE:
012524  104407                                  CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
012526  032777  040000  166404          1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
012534  001101                                  BNE     $OVER           ;;YES IF SW14=1
                                        ;#####START OF CODE FOR THE XOR TESTER#####
012536  000416                          $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                        ;;THIS INSTRUCTION TO A "NOP" (NOP=240.
012540  013746  000004                          MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
012544  012737  012564  000004                  MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
012552  005737  177060                          TST     @#177060        ;;TIME OUT ON XOR?
012556  012637  000004                          MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
012562  000453                                  BR      $SVLAD          ;;GO TO THE NEXT TEST
012564  022626                          5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
012566  012637  000004                          MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
012572  000413                                  BR      7$              ;;LOOP ON THE PRESENT TEST
012574                                  6$::#####END OF CODE FOR THE XOR TESTER#####
012574  105737  001103                  2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
012600  001421                                  BEQ     3$              ;;BR IF NO
012602  123737  001115  001103                  CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
012610  101015                                  BHI     3$              ;;BR IF NO
012612  032777  001000  166320                  BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
012620  001404                                  BEQ     4$              ;;BR IF NO
012622  013737  001110  001106          7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
012630  000443                                  BR      $OVER
012632  105037  001103                  4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
012636  005037  001206                          CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
012642  000415                                  BR      1$              ;;ESCAPE TO THE NEXT TEST
012644  032777  004000  166266          3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
012652  001011                                  BNE     1$              ;;BR IF YES
012654  005737  001100                          TST     $PASS           ;;IF FIRST PASS OF PROGRAM
012660  001406                                  BEQ     1$              ;;      INHIBIT ITERATIONS
012662  005237  001104                          INC     $ICNT           ;;INCREMENT ITERATION COUNT
012666  023737  001206  001104                  CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
012674  002021                                  BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
012676  012737  000001  001104          1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
012704  013737  012754  001206                  MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
012712  105237  001102                  $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
012716  011637  001106                          MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
012722  011637  001110                          MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
012726  005037  001210                          CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
012732  112737  000001  001115                  MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
012740  013777  001102  166174          $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
012746  013716  001106                          MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
012752  000002                                  RTI                     ;;FIXES PS
012754  000001                          $MXCNT: 1                       ;;MAX. NUMBER OF ITERATIONS

                                        .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
```

```
                                        ;;*****************************************************************
                                        ;*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
012756                                  $SAVREG:
012756  010046                                  MOV     R0,-(SP         ;;PUSH R0 ON STACK
012760  010146                                  MOV     R1,-(SP         ;;PUSH R1 ON STACK
012762  010246                                  MOV     R2,-(SP         ;;PUSH R2 ON STACK
012764  010346                                  MOV     R3,-(SP         ;;PUSH R3 ON STACK
012766  010446                                  MOV     R4,-(SP         ;;PUSH R4 ON STACK
012770  010546                                  MOV     R5,-(SP         ;;PUSH R5 ON STACK
012772  016646  000022                          MOV     22(SP),-(SP     ;;SAVE PS OF MAIN FLOW
012776  016646  000022                          MOV     22(SP),-(SP     ;;SAVE PC OF MAIN FLOW
013002  016646  000022                          MOV     22(SP),-(SP     ;;SAVE PS OF CALL
013006  016646  000022                          MOV     22(SP),-(SP     ;;SAVE PC OF CALL
013012  000002                                  RTI

                                        ;*RESTORE R0-R5
                                        ;*CALL:
                                        ;*      RESREG
013014                                  $RESREG:
013014  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
013020  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
013024  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
013030  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
013034  012605                                  MOV     (SP)+,R5        ;;POP STACK INTO R5
013036  012604                                  MOV     (SP)+,R4        ;;POP STACK INTO R4
013040  012603                                  MOV     (SP)+,R3        ;;POP STACK INTO R3
013042  012602                                  MOV     (SP)+,R2        ;;POP STACK INTO R2
013044  012601                                  MOV     (SP)+,R1        ;;POP STACK INTO R1
013046  012600                                  MOV     (SP)+,R0        ;;POP STACK INTO R0
013050  000002                                  RTI

                                        .SBTTL  INTEGER MULTIPLY ROUTINE

                                        ;;*****************************************************************
                                        ;*CALL
                                        ;*      MOV     MULTIPLER,-(SP)
                                        ;*      MOV     MULTIPLICAND,-(SP
                                        ;*      JSR     PC,2$MULT
                                        ;*      RETURN  ;;PRODUCT IS ON THE STACK
                                        ;*
                                        ;*      STACK   PRODUCT
                                        ;*      -----   -------
```

```
                                        ;*      TOP     LSB'S
                                        ;*      +2      MSB'S

013052                                  $MULT:
013052  010046                                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
013054  010146                                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
013056  010246                                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
013060  005046                                  CLR     -(SP)           ;;CLEAR THE SIGN KEY
013062  016601  000012                          MOV     12(SP),R1       ;;GET THE MULTIPLICAND
013066  100002                                  BPL     1$              ;;BR IF PLUS
013070  005216                                  INC     (SP)            ;;SET THE SIGN KEY
013072  005401                                  NEG     R1              ;;MAKE THE MULTIPLICAND POSTIVE
013074  016602  000014                  1$:     MOV     14(SP),R2       ;;GET THE MULTIPLIER
013100  100002                                  BPL     2$              ;;BR IF PLUS
013102  005316                                  DEC     (SP)            ;;UPDATE THE SIGN KEY
013104  005402                                  NEG     R2              ;;MAKE THE MULTIPLIER POSTIVE
013106  012746  000021                  2$:     MOV     #17.,-(SP       ;;SET THE LOOP COUNT
013112  005000                                  CLR     R0              ;;SETUP FOR THE MULTIPLY LOOP
013114  103001                          3$:     BCC     4$              ;;DON'T ADD IF MULTIPLICAND = 0
013116  060200                                  ADD     R2,R0
013120  006000                          4$:     ROR     R0              ;;POSITION THE PARITIAL PRODUCT AND
013122  006001                                  ROR     R1              ;;THE MULTIPLICAND
013124  005316                                  DEC     (SP)            ;;HAS ALL BITS OF THE MULTIPLICAND BEEN DONE?
013126  001372                                  BNE     3$              ;;BR IF NO
013130  022616                                  CMP     (SP)+,(SP       ;;SHOULD PRODUCT BE NEGATIVE?
013132  001403                                  BEQ     5$              ;;GO TO EXIT IF NO
013134  005400                                  NEG     R0              ;;YES--SO MAKE IT SO
013136  005401                                  NEG     R1
013140  005600                                  SBC     R0
013142  005726                          5$:     TST     (SP)+           ;;CLEAR SIGN INFO. OFF OF STACK
013144  010066  000012                          MOV     R0,12(SP        ;;PUT THE PRODUCT ON THE STACK  MSB'S
013150  010166  000010                          MOV     R1,10(SP        ;;LSB'S
013154  012602                                  MOV     (SP)+,R2        ;;POP STACK INTO R2
013156  012601                                  MOV     (SP)+,R1        ;;POP STACK INTO R1
013160  012600                                  MOV     (SP)+,R0        ;;POP STACK INTO R0
013162  000207                                  RTS     PC

                                        .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.

013164  010046                          $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
013166  016600  000002                          MOV     2(SP),R0        ;;GET TRAP ADDRESS
013172  005740                                  TST     -(R0            ;;BACKUP BY 2
013174  111000                                  MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
013176  006300                                  ASL     R0              ;;POSITION FOR INDEXING
013200  016000  01322C                          MOV     $TRPAD(R0),PC   ;;INDEX TO TABLE
013204  000200                                  RTS     R0              ;;GO TO ROUTINE

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

```
013206  011646                  $TRAP2: MOV     SP),-(SP)       ;;MOVE THE PC DOWN
013210  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
013216  000002                          RTI                     ;;RESTORE THE PSW

                                .SBTTL  TRAP TABLE

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

                                ;       ROUTINE
                                ;       -------
013220  013206                  $TRPAD: .WORD   $TRAP2
013222  010352                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
013224  010616                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER WITH LEADING ZEROS
013226  010572                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER  NO LEADING ZEROS
013230  010632                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL
013232  011020                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER  WITH SIGN

013234  011632                          $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING

013236  011542                          $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
013240  012104                          $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
013242  012174                          $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
013244  012756                          $SAVREG ;;CALL=SAVREG   TRAP+12(104412) SAVE R0-R5 ROUTINE
013246  013014                          $RESREG ;;CALL=RESREG   TRAP+13(104413) RESTORE R0-R5 ROUTINE

                                .SBTTL  SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE

                                ;;*****************************************************************
                                ;*THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
                                ;*UNSIGNED DECIMAL ASCIZ NUMBER.
                                ;*CALL
                                ;*      MOV     NUMBER,-(SP)    ;;PUT BINARY NUMBER ON THE STACK
                                ;*      JSR     PC,@#$SB2D      ;;CALL
                                ;*      RETURN                  ;;ADDRESS OF THE 1ST ASCIZ CHAR.IS ON THE STACK

013250  016637  000002  013300  $SB2D:  MOV     2(SP),1$        ;;SAVE BINARY NUMBER
013256  012746  013300                  MOV     #1$,-(SP)       ;;SET POINTER
013262  004737  013304                  JSR     PC,@#$DB2D      ;;CALL DOUBLE LENGTH CONVERT
013266  062716  000005                  ADD     #5,(SP)         ;;ONLY ALLOW FIVE CHARACTERS
013272  012666  000002                  MOV     (SP)+,2(SP)     ;;PICKUP POINTER
013276  000207                          RTS     PC              ;;RETURN
013300  000000  000000          1$:     .WORD   0,0

                                .SBTTL  DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

                                ;;*****************************************************************
                                ;*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
                                ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
                                ;*POSITIVE.
                                ;*CALL
                                ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
                                ;*      JSR     PC,@#$DB2D
                                ;*      RETURN                  ;;THE FIRST ADDRESS OF ASCIZ
                                                                ;;IS ON THE STACK
```

```
013304  104412                          $DB2D:  SAVREG                          ;;SAVE REGISTERS
013306  016602  000002                          MOV     2(SP),R2                ;;PICKUP THE DATA POINTER
013312  012700  013464                          MOV     #$DECVL,R0              ;;GET ADDRESS OF "$DECVL" STRING
013316  010066  000002                          MOV     R0,2(SP)                ;;PUT ADDRESS OF ASCIZ STRING ON STACK
013322  012201                                  MOV     (R2)+,R1                ;;PICKUP THE BINARY NUMBER
013324  012202                                  MOV     (R2)+,R2
013326  012737  000012  013402                  MOV     #10.,4$                 ;;SET UP TO DO 10 CONVERSIONS
013334  012704  013414                          MOV     #$TNPWR,R4              ;;ADDRESS OF TEN POWER
013340  012705  013416                          MOV     #$TNPWR+2,R5
013344  005003                          1$:     CLR     R3                      ;;CLEAR PARTIAL
013346  161401                          2$:     SUB     (R4),R1                 ;;SUBTRACT TEN POWER
013350  005602                                  SBC     R2
013352  161502                                  SUB     (R5),R2
013354  002402                                  BLT     3$                      ;;BR IF TEN POWER TO LARGE
013356  005203                                  INC     R3                      ;;ADD 1 TO PARTIAL
013360  000772                                  BR      2$                      ;;LOOP
013362  062401                          3$:     ADD     (R4)+,R1                ;;RESTORE SUBTRACTED VALUE
013364  005502                                  ADC     R2
013366  062402                                  ADD     (R4)+,R2
013370  022525                                  CMP     (R5)+,(R5)+             ;;MOVE TO NEXT TEN POWER
013372  052703  000060                          BIS     #'0,R3                  ;;CHANGE PARTIAL TO ASCII
013376  110320                                  MOVB    R3,(R0)+                ;;SAVE IT
013400  005327                                  DEC     (PC)+                   ;;DONE?
013402  000000                          4$:     .WORD   0
013404  001357                                  BNE     1$                      ;;BR IF NO
013406  105020                                  CLRB    (R0)+                   ;;TERMINATOR
013410  104413                                  RESREG                          ;;RESTORE REGISTERS
013412  000207                                  RTS     PC                      ;;RETURN
013414  145000                          $TNPWR: 145000                          ;;1.0E09
013416  035632                                  35632
013420  160400                                  160400                          ;;1.0E08
013422  002765                                  2765
013424  113200                                  113200                          ;;1.0E07
013426  000230                                  230
013430  041100                                  041100                          ;;1.0E06
013432  000017                                  17
013434  103240                                  103240                          ;;1.0E05
013436  000001                                  1
013440  023420                                  23420                           ;;1.0E04
013442  000000                                  0
013444  001750                                  1750                            ;;1.0E03
013446  000000                                  0
013450  000144                                  144                             ;;1.0E02
013452  000000                                  0
013454  000012                                  12                              ;;1.0E01
013456  000000                                  0
013460  000001                                  1                               ;;1.0E00
013462  000000                                  0
013464  000014                          $DECVL: .BLKB   12.                     ;;RESERVE STORAGE FOR ASCIZ STRING

                                        .SBTTL  TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS

                                        ;;*****************************************************************
                                        ;;*THIS ROUTINE IS USED TO TYPE AN ASCIZ NUMBER SUPPRESSING THE
```

```
                                        ;*LEADING NUMBERS.
                                        ;*CALL
                                        ;*      MOV     #NUMADR,-(SP)   ;;FIRST ADDRESS OF ASCIZ STRING
                                        ;*      JSR     PC,@#$SUPRS

013500  010046                          $SUPRS: MOV     R0,-(SP)        ;;SAVE R0
013502  016600  000004                          MOV     4(SP),R0        ;;PICKUP THE POINTER
013506  105710                          1$:     TSTB    (R0)            ;;TERMINATEOR?
013510  001403                                  BEQ     2$              ;;BR IF YES
013512  122720  000060                          CMPB    #'0,(R0)+       ;;IS THIS AN ASCII "0" ?
013516  001773                                  BEQ     1$              ;;BR IF YES
013520  005300                          2$:     DEC     R0              ;;BACKUP BY "1"
013522  010037  013530                          MOV     R0,3$           ;;SAVE FOR TYPING
013526  104401                                  TYPE                    ;;GO TYPE
013530  000000                          3$:     .WORD   0               ;;ASCIZ POINTER GOES HERE
013532  012600                                  MOV     (SP)+,R0        ;;RESTORE R0
013534  012616                                  MOV     (SP)+,(SP)      ;;RESTORE THE STACK
013536  000207                                  RTS     PC              ;;RETURN

                                        .SBTTL  RANDOM NUMBER GENERATOR ROUTINE

                                        ;;*****************************************************************
                                        ;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
                                        ;*WITH A RANGE OF 0 TO 2(+33)-1.
                                        ;*CALL:
                                        ;*      JSR     PC,$RAND        ;;CALL THE ROUTINE
                                        ;*      RETURN                  ;;RETURN HERE THE RANDOM
                                        ;*                              ;;NUMBER WILL BE IN
                                        ;*                              ;;$HINUM,$LONUM

013540                                  $RAND:
013540  010046                                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
013542  010146                                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
013544  010246                                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
013546  013700  013640                          MOV     $LONUM,R0       ;;SET R0 WITH LOW
013552  013701  013636                          MOV     $HINUM,R1       ;;SET R1 WITH HIGH
013556  012702  177771                          MOV     #-7,R2          ;;SET SHIFT COUNT
013562  006300                          1$:     ASL     R0              ;;SHIFT R0 LEFT AND
013564  006101                                  ROL     R1              ;;ROTATE CARRY INTO R1 AND
013566  005202                                  INC     R2              ;;CHECK FOR DONE
013570  001374                                  BNE     1$              ;;CONTINUE SHIFT LOOP
013572  063700  013640                          ADD     $LONUM,R0       ;;ADD NUMBER TO MAKE X 129
013576  005501                                  ADC     R1              ;;PROPOGATE CARRY
013600  063701  013636                          ADD     $HINUM,R1       ;;ADD NUMBER TO MAKE X 129
013604  062700  001057                          ADD     #1057,R0        ;;ADD LOW CONSTANT
013610  005501                                  ADC     R1              ;;PROPOGATE CARRY
013612  062701  047401                          ADD     #47401,R1       ;;ADD HIGH CONSTANT
013616  010037  013640                          MOV     R0,$LONUM       ;;SAVE R0
013622  010137  013636                          MOV     R1,$HINUM       ;;SAVE R1
013626  012602                                  MOV     (SP)+,R2        ;;POP STACK INTO R2
013630  012601                                  MOV     (SP)+,R1        ;;POP STACK INTO R1
013632  012600                                  MOV     (SP)+,R0        ;;POP STACK INTO R0
013634  000207                                  RTS     PC              ;;RETURN
013636  176543                          $HINUM: .WORD   176543
013640  123456                          $LONUM: .WORD   123456
```

```
                                        ;;*****************************************************************
                                        ;.SBTTL  RPINIT - INITIALIZE THE RP11 & SEE WHICH DRIVES ARE AVAILABLE
                                        ;*THIS ROUTINE CLEARS THE RP11 AND DETERMINES WHICH DRIVES ARE AVAILABLE.
                                        ;*THE TABLE 'DRVSTA' IS LOADED TO REFLECT THE SYSTEM STATUS.
                                        ;CALL
                                        ;       JSR     PC,@#RPINIT
                                        ;       RETURN

013642  104412                          RPINIT: SAVREG                          ;SAVE R0-R5
013644  013701  001270                          MOV     RPVEC,R1                ;VECTOR ADDRESS
013650  005021                                  CLR     (R1)+                   ;SET INTERRUPT ADDRESS TO ZERO
013652  013711  001272                          MOV     RPPRIO,(R1)             ;RP11 PRIORITY
013656  005037  001240                          CLR     DRVTYP                  ;CLEAR DRIVE TYPE STORAGE
013662  005037  001304                          CLR     DRVSTA                  ;SET DRIVE STATUS TO OFFLINE
013666  005037  001306                          CLR     DRVSTA+2                ;SET DRIVE STATUS TO OFFLINE
013672  005037  001310                          CLR     DRVSTA+4                ;SET DRIVE STATUS TO OFFLINE
013676  005037  001312                          CLR     DRVSTA+6                ;SET DRIVE STATUS TO OFFLINE
013702  013700  001266                          MOV     RPADR,R0                ;PUT RP11 ADDRESS INTO R0
013706  012760  000001  000004                  MOV     #CLEAR,RPCS(R0)         ;CLEAR THE RP11
013714  012760  000001  000004                  MOV     #CLEAR,RPCS(R0)         ;CLEAR THE RP11 AGAIN
013722  012701  000007                          MOV     #7,R1                   ;'DRVSTA' TABLE INDEX
013726  110160  000005          1$:             MOVB    R1,RPCS+1(R0)           ;SELECT DRIVE
013732  032760  001000  000000                  BIT     #SUFU,RPDS(R0)          ;SEE IF DRIVE UNSAFE
013740  001404                                  BEQ     2$                      ;BR IF NOT UNSAFE
013742  112761  177777  001304                  MOVB    #-1,DRVSTA(R1)          ;SET INDICATOR TO 'UNSAFE'
013750  000417                                  BR      4$                      ;EXIT
013752  032760  040000  000000          2$:     BIT     #SUOL,RPDS(R0)          ;IS DRIVE ONLINE ?
013760  001422                                  BEQ     5$                      ;BR IF NOT
013762  032760  004000  000000                  BIT     #SUSI,RPDS(R0)          ;SEEK INCOMPLETE SET ?
013770  001004                                  BNE     3$                      ;BR IF SET
013772  032760  100000  000000                  BIT     #SURDY,RPDS(R0)         ;IS DRIVE READY ?
014000  001412                                  BEQ     5$                      ;BR IF NOT
014002  112761  000001  001304          3$:     MOVB    #1,DRVSTA(R1)           ;SET DRIVE INDICATOR TO 'ONLINE'
014010  032760  020000  000000          4$:     BIT     #SURP03,RPDS(R0)        ;IS THE DRIVE AN RP03 ?
014016  001403                                  BEQ     5$                      ;BR IF NOT
014020  156137  001314  001240                  BISB    ATABIT(R1),DRVTYP       ;SET RP03 INDICATOR
014026  005301                          5$:     DEC     R1                      ;DECREMENT THE TABLE INDEX
014030  100336                                  BPL     1$                      ;BR IF NOT FINISHED
014032  104413                                  RESREG                          ;RESTORE R0-R5
014034  000207                                  RTS     PC                      ;RETURN

                                        ;;*****************************************************************
                                        ;.SBTTL  SEEKS - INITIATE A DIRECT SEEK (OR HOME SEEK) OPERATION
                                        ;*THIS ROUTINE INITIATES A SEEK OPERATION ON THE DRIVE ADDRESSED IN
                                        ;*THE SECOND BYTE OF 'DPB'. 1 SECOND IS ALLOWED FOR THE SEEK
                                        ;*TO COMPLETE: IF THE DRIVE HAS NOT INTERRUPTED ,'AIE' IS SET
                                        ;*WITHIN THAT TIME, THE SUBSYSTEM IS RESET AND THE TESTING FOR THE
                                        ;*DRIVE IS TERMINATED.
                                        ;CALL
                                        ;       JSR     PC,@#SEEKS
                                        ;       RETURN

014036  104412                          SEEKS:  SAVREG                          ;SAVE R0-R5
014040  013737  001272  177776                  MOV     RPPRIO,PSW              ;RP11 PRIORITY
014046  013700  001266                          MOV     RPADR,R0                ;RP11 BUS ADDRESS
```

```
014052  013746  001110                  MOV     $LPERR,-(SP)        ;SAVE OLD LOOP ON ERROR ADDRESS
014056  012737  014064  001110          MOV     #SEEKS0,$LPERR      ;CHANGE LOOP ON ERROR ADDRESS
014064  012760  000001  000004  SEEKS0: MOV     #CLEAR,RPCS(R0)     ;CLEAR THE CONTROLLER
014072  012760  000001  000004          MOV     #CLEAR,RPCS(R0)     ;CLEAR THE CONTROLLER
014100  113760  001365  000005          MOVB    DPB+1,RPCS+1(R0)    ;SELECT THE DRIVE
014106  013737  001366  001262          MOV     DPB+2,CYL,DS        ;DESIRED CYLINDER
014114  013760  001366  000012          MOV     DPB+2,RPCA(R0)      ;LOAD CYLINDER ADDRESS
014122  013760  001370  000014          MOV     DPB+4,RPDA(R0)      ;TRACK ADDRESS
014130  012777  014314  165132          MOV     #SEEKS1,@RPVEC      ;INTERRUPT RETURN
014136  013760  001364  000004          MOV     DPB,RPCS(R0)        ;START THE COMMAND
014144  152760  000040  000005          BISB    #40,RPCS+1(R0)      ;SET ATTENTION INTERRUPT ENABLE
014152  005037  177776                  CLR     PS                  ;ALLOW RP11 INTERRUPTS
014156  010637  001242                  MOV     SP,SPSAV            ;SAVE THE CURRENT STACK POINTER
014162  012737  004704  001300          MOV     #2500.,STALLG       ;SET STALL VALUE TO 2500MS
014170  004037  015572                  JSR     R0,STALLA           ;WAIT THE SPECIFIED VALUE
014174  001300                          .WORD   STALLG              ;STALL VALUE STORAGE
014176  004737  015164                  JSR     PC,SAVRP            ;OPERATION TIMED OUT, SAVE THE REGISTERS
014202  032737  001000  002004          BIT     #SUFU,$RPDS         ;DRIVE UNSAFE ?
014210  001403                          BEQ     1$                  ;BR IF NOT
014212  104002                          ERROR   2                   ;DRIVE UNSAFE
014214  000137  015272                  JMP     DSELCT              ;DESELECT THE DRIVE
014220  032737  040000  002004  1$:     BIT     #SUOL,$RPDS         ;DRIVE STILL ONLINE ?
014226  001003                          BNE     2$                  ;BR IF IT IS
014230  104010                          ERROR   10                  ;DRIVE OFFLINE AFTER POSITIONING
014232  000137  015272                  JMP     DSELCT              ;DESELECT THE DRIVE
014236  032737  100000  002004  2$:     BIT     #SURDY,$RPDS        ;IS THE DRIVE READY ?
014244  001003                          BNE     3$                  ;BR IF THE DRIVE IS READY
014246  104016                          ERROR   16                  ;DRIVE NOT READY AFTER POSITIONING
014250  000137  015272                  JMP     DSELCT              ;DESELCT THE DRIVE
014254  104005                  3$:     ERROR   5                   ;NO INTERRUPT FROM POSITIONING AFTER 1 SECOND
014256  032737  000200  002010          BIT     #RDY,$RPCS          ;IS THE CONTROLLER READY ?
014264  001011                          BNE     4$                  ;BR IF IT IS
014266  012760  000001  000004          MOV     #CLEAR,RPCS(R0)     ;CLEAR THE CONTROLLER
014274  012760  000001  000004          MOV     #CLEAR,RPCS(R0)     ;CLEAR THE CONTROLLER
014302  012777  000100  164634          MOV     #BIT06,@$TKS        ;RESTORE THE KEYBOARD INTERRUPT ENABLE
014310  000177  164730          4$:     JMP     @BYPASS             ;TERMINATE THE PRESENT TEST
014314  005037  177776          SEEKS1: CLR     PSW                 ;GO BACK TO PRIORITY 0
014320  013706  001242                  MOV     SPSAV,SP            ;RESTORE THE STACK POINTER
014324  013700  001266                  MOV     RPADR,R0            ;RP11 ADDRESS
014330  004737  015164                  JSR     PC,SAVRP            ;SAVE THE RP11 REGISTERS
014334  032737  004000  002004          BIT     #SUSI,$RPDS         ;SEEK INCOMPLETE ?
014342  001416                          BEQ     SEEKS2              ;BR IF NOT
014344  012737  014354  001210          MOV     #1$,$ESCAPE         ;ERROR 'ESCAPE' ADDRESS
014352  104015                          ERROR   15                  ;SEEK INCOMPLETE
014354  005037  001210          1$:     CLR     $ESCAPE             ;CLEAR THE ESCAPE ADDRESS
014360  004737  015314                  JSR     PC,RESTOR           ;DO A HOME SEEK TO CLEAR THE DRIVE
014364  032777  001000  164546          BIT     #SW09,@SWR          ;LOOPING ON ERROR ?
014372  001452                          BEQ     SEEKS4              ;BR IF NOT
014374  000177  164510                  JMP     @$LPERR             ;LOOP ON THE OPERATION
014400  032737  100000  002010  SEEKS2: BIT     #ERR,$RPCS          ;'ERROR' SET ?
014406  001410                          BEQ     1$                  ;BR IF NOT
014410  104006                          ERROR   6                   ;DRIVE ERROR AFTER POSITIONING
014412  012760  000001  000004          MOV     #CLEAR,RPCS(R0)     ;CLEAR THE CONTROLLER
014420  012760  000001  000004          MOV     #CLEAR,RPCS(R0)     ;CLEAR THE CONTROLLER
014426  000415                          BR      SEEKS3              ;CHECK THE DISK'S POSITION
014430  005001                  1$:     CLR     R1                  ;CLEAR R1
```

```
014432  113701  001365                  MOVB    DPB+1,R1            ;DRIVE ADDRESS
014436  136137  001314  002004          BITB    ATABIT(R1),$RPDS    ;ATTENTION BIT SET FOR THE DRIVE
014444  001001                          BNE     2$                  ;BR IF IT IS
014446  104007                          ERROR   7                   ;ATTENTION BIT NOT SET AFTER POSITIONING
014450  023737  001366  002024  2$:     CMP     DPB+2,$SUCA         ;'SUCA' CORRECT ?
014456  001401                          BEQ     SEEKS3              ;BR IF IT IS
014460  104011                          ERROR   11                  ;'SUCA' NOT CORRECT
014462  004737  014554          SEEKS3: JSR     PC,VERIFY           ;VERIFY THE DISK'S POSITION
014466  000414                          BR      SEEKS4              ;AT THE CORRECT CYLINDER
014470  122737  000015  001364          CMPB    #HOMSEK,DPB         ;DOING A HOME SEEK ?
014476  001410                          BEQ     SEEKS4              ;BR IF YES, NO SENSE DOING A HOME SEEK AGAIN
014500  004737  015314                  JSR     PC,RESTOR           ;DO A HOME SEEK
014504  032777  001000  164426          BIT     #SW09,@SWR          ;LOOP ON THE OPERATION ?
014512  001402                          BEQ     SEEKS4              ;BR IF NOT
014514  000177  164370                  JMP     @$LPERR             ;LOOP
014520  032777  000010  164412  SEEKS4: BIT     #SW03,@SWR          ;STALL ?
014526  001403                          BEQ     1$                  ;BR IF NOT
014530  004037  015576                  JSR     R0,STALL            ;STALL
014534  001276                          .WORD   STALL1              ;25MS STALL
014536  016037  000024  001254  1$:     MOV     SUCA(R0),CYL.CR     ;CURRENT CYLINDER ADDRESS
014544  012637  001110                  MOV     (SP)+,$LPERR        ;RESTORE THE LOOP ON ERROR ADDRESS
014550  104413                          RESREG                      ;RESTORE R0-R5
014552  000207                          RTS     PC                  ;RETURN

                                ;*****************************************************************
                                ;SBTTL  VERIFY - ROUTINE TO VERIFY THE POSITION AFTER A SEEK
                                ;*THIS ROUTINE VERIFIES THE POSITION OF THE DRIVE AFTER A SEEK
                                ;*BY READING A HEADER AT THE DESTINATION CYLINDER.  THE HEADER
                                ;*READ IS FROM THE SECTOR WHICH IS 2 SECTORS FROM THE ADDRESS IN
                                ;*THE 'SOT' COUNTER.  THE CYLINDER FIELD FROM 'DPB+2' IS COMPARED
                                ;*WITH THE CYLINDER FIELD FROM THE HEADER.
                                ;CALL
                                ;       JSR     PC,VERIFY
                                ;       RETURN                      ;NORMAL RETURN
                                ;       ERROR RETURN                ;CYLINDER FIELDS DID NOT COMPARE

014554  104412                  VERIFY: SAVREG                      ;SAVE R0-R5
014556  013746  001110                  MOV     $LPERR,-(SP)        ;SAVE THE LOOP ON ERROR ADDRESS
014562  032777  000001  164350          BIT     #SW0,@SWR           ;CHECK POSITION ?
014570  001171                          BNE     VERFY1              ;BR IF NOT
014572  012737  014600  001110          MOV     #VERFYO,$LPERR      ;LOOP ON ERROR ADDRESS
014600  012737  177777  027172  VERFYO: MOV     #-1,BUFFER+2        ;CLEAR CYLINDER STORAGE
014606  013700  001266                  MOV     RPADR,R0            ;LOAD RP11 ADDRESS
014612  016046  000014                  MOV     RPDA(R0),-(SP)      ;GET THE 'SOT'
014616  042716  177417                  BIC     #^C360,(SP)         ;SAVE THE 'SOT' BITS
014622  006216                          ASR     (SP)                ;RIGHT JUSTIFY THE BITS
014624  006216                          ASR     (SP)                ;RIGHT JUSTIFY THE BITS
014626  006216                          ASR     (SP)                ;RIGHT JUSTIFY THE BITS
014630  006216                          ASR     (SP)                ;RIGHT JUSTIFY THE BITS
014632  062716  000002                  ADD     #2,(SP)             ;FORM SECTOR ADDRESS FOR CHECK
014636  022716  000012                  CMP     #10.,(SP)           ;SEE IF THE ADDRESS OVERFLOWED
014642  101005                          BHI     2$                  ;BR IF NOT
014644  103402                          BLO     1$                  ;BR IF ADDR IS 11.
014646  005016                          CLR     (SP)                ;SET SECTOR ADDRESS TO ZERO
014650  000402                          BR      2$                  ;CONTINUE
014652  012716  000001          1$:     MOV     #1,(SP)             ;SET SECTOR ADDRESS TO ONE
```

```
014656  012777  015026  164404  2$:     MOV     #4$,@RPVEC          ;INTERRUPT ADDRESS
014664  112637  001370              MOVB    (SP)+,DPB+4         ;PUT VALUE INTO THE DPB
014670  113737  001370  001260      MOVB    DPB+4,SEC.RD        ;SELECTED SECTOR FOR ERROR MESSAGE
014676  113737  001371  001256      MOVB    DPB+5,TRK.RD        ;TRACK FOR ERROR MESSAGE
014704  014746  001364              MOV     DPB,-(SP)           ;SETUP RPCS LOAD VALUE
014710  112716  000017              MOVB    #READ,(SP)          ;READ COMMAND CODE
014714  052716  014100              BIS     #MODE!HDR!IDE,(SP)  ;SET BITS FOR HEADER READ
014720  012760  177775  000006      MOV     #-3,RPWC(R0)        ;LOAD WORD COUNT
014726  012760  027170  000010      MOV     #BUFFER,RPBA(R0)    ;BUFFER ADDRESS
014734  013760  001370  000014      MOV     DPB+4,RPDA(R0)      ;SECTOR ADDRESS
014742  012660  000004              MOV     (SP)+,RPCS(R0)      ;START THE ORDER
014746  005037  177776              CLR     PSW                 ;SET PRIORITY BACK TO ZERO
014752  010637  001242              MOV     SP,SPSAV            ;SAVE THE CURRENT STACK POINTER
014756  012737  004704  001300      MOV     #2500.,STALLG       ;SET STALL VALUE TO 2500MS
014764  004037  015572              JSR     R0,STALLA           ;WAIT THE SPECIFIED VALUE
014770  001300                      .WORD   STALLG              ;STALL VALUE STORAGE
014772  004737  015164              JSR     PC,SAVRP            ;STORE THE RP11 REGISTERS
014776  104014                      ERROR   14                  ;NO INTERRUPT WITHIN 1 SECOND
015000  012760  000001  000004      MOV     #CLEAR,RPCS(R0)     ;CLEAR THE CONTROLLER
015006  012760  000001  000004      MOV     #CLEAR,RPCS(R0)     ;CLEAR THE CONTROLLER
015014  012777  000100  164122      MOV     #BIT6,@$TKS         ;SET TTY KEYBOARD INTERRUPT ENABLE
015022  000177  164216              JMP     @BYPASS             ;TERMINATE THE PRESENT TEST
015026  005037  177776      4$:     CLR     PSW                 ;SET PRIORITY TO ZERO
015032  013706  001242              MOV     SPSAV,SP            ;RESTORE STACK POINTER
015036  004737  015164              JSR     PC,SAVRP            ;STORE THE RP11 REGISTERS
015042  013700  001266              MOV     RPADR,R0            ;PUT RP11 ADDRESS INTO R0
015046  032737  100000  002010      BIT     #ERR,$RPCS          ;DID THE ORDER TERMINTE WITH AN ERROR
015054  001410                      BEQ     6$                  ;BR IF NOT
015056  104012              5$:     ERROR   12                  ;ERROR DURING POSITIONING VERIFICATION
015060  012760  000001  000004      MOV     #CLEAR,RPCS(R0)     ;CLEAR THE ERROR
015066  012760  000001  000004      MOV     #CLEAR,RPCS(R0)     ;CLEAR THE CONTROLLER
015074  000427                      BR      VERFY1              ;BYPASS POSITIONING CHECK
015076  000006              6$:
015076  006237  027172              ASR     BUFFER+2            ;ALIGN CYLINDER ADDRESS FOR COMPARE
015102  006237  027172              ASR     BUFFER+2            ;ALIGN CYLINDER ADDRESS FOR COMPARE
015106  006237  027172              ASR     BUFFER+2            ;ALIGN CYLINDER ADDRESS FOR COMPARE
015112  006237  027172              ASR     BUFFER+2            ;ALIGN CYLINDER ADDRESS FOR COMPARE
015116  006237  027172              ASR     BUFFER+2            ;ALIGN CYLINDER ADDRESS FOR COMPARE
015122  006237  027172              ASR     BUFFER+2            ;ALIGN CYLINDER ADDRESS FOR COMPARE
015126  023737  001366  027172      CMP     DPB+2,BUFFER+2      ;SEE IF CYLINDER ADDR FROM HEADER IS CORRECT
015134  001407                      BEQ     VERFY1              ;BR IF OK
015136  013737  027172  001126      MOV     BUFFER+2,$BDDAT     ;CYLINDER ADDRESS FROM HEADER
015144  104013                      ERROR   13                  ;CYLINDER ADDRESS DOES NOT COMPARE
015146  062766  000002  000002      ADD     #2,2(SP)            ;RETURN+2
015154  012637  001110      VERFY1: MOV     (SP)+,$LPERR        ;RESTORE LOOP ON ERROR ADDRESS
015160  104413                      RESREG                      ;RESTORE R0-R5
015162  000207                      RTS     PC                  ;RETURN

                            ;*****************************************************************
                            .SBTTL  SAVRP - STORE THE RP11E REGISTERS
                            ;*THIS ROUTINE STORES THE RP11E REGISTERS FOR USE BY THE PROGRAM.
                            ;*THE PROGRAM DOES NOT USE THE 'LIVE' REGISTERS FOR ERROR DETERMINATION.
                            ;CALL
                            ;       JSR     PC,SAVRP
                            ;       RETURN
```

```
015164  010046                  SAVRP:  MOV     R0,-(SP)          ;SAVE R0
015166  013700  001266                  MOV     RPADR,R0          ;RP11 ADDRESS
015172  016037  000000  002004          MOV     RPDS(R0),$RPDS    ;STORE RPDS
015200  016037  000002  002006          MOV     RPER(R0),$RPER    ;STORE RPER
015206  016037  000004  002010          MOV     RPCS(R0),$RPCS    ;STORE RPCS
015214  016037  000006  002012          MOV     RPWC(R0),$RPWC    ;STORE RPWC
015222  016037  000010  002014          MOV     RPBA(R0),$RPBA    ;STORE RPBA
015230  016037  000012  002016          MOV     RPCA(R0),$RPCA    ;STORE RPCA
015236  016037  000014  002020          MOV     RPDA(R0),$RPDA    ;STORE RPDA
015244  016037  000016  002022          MOV     RPM1(R0),$RPM1    ;STORE RPM1
015252  016037  000024  002024          MOV     SUCA(R0),$SUCA    ;STORE SUCA
015260  016037  000026  002026          MOV     SILO(R0),$SILO    ;STORE SILO
015266  012600                          MOV     (SP)+,R0          ;RESTORE R0
015270  000207                          RTS     PC                ;RETURN

                                ;*****************************************************************
                                ;$BTTL  DSELCT - SETUP INDICATORS TO DEASSIGN AN UNSAFE/OFFLINE DRIVE
                                ;*THIS ROUTINE SETS A BIT IN LOCATION 'DSELCT' WHICH IS USED TO DESELCT
                                ;*AN UNSAFE/OFFLINE DRIVE WHICH WAS ASSIGNED BY THE OPERATOR FOR TESTING
                                ;*OR WHICH WENT OFFLINE OR UNSAFE DURING TESTING.
                                ;CALL
                                ;       JMP     @#DSELCT

015272  013701  001246          DSELCT: MOV     CHKDRV,R1         ;GET DRIVE NUMBER
015276  156137  001314  001252          BISB    ATABIT(R1),DRVBAD  ;USE AS INDEX TO LOAD ATA BIT FOR DRIVE
                                                                  ;IN DESELECT LOCATION.
015304  005037  177776                  CLR     PSW               ;SET PRIORITY BACK TO ZERO
015310  000137  007474                  JMP     @#$EOP            ;FORCE END OF PASS FOR THE DRIVE

                                ;*****************************************************************
                                ;$BTTL  RESTOR - HOME SEEK ROUTINE
                                ;*THIS ROUTINE PERFORMS A HOME SEEK ON THE SELECTED DRIVE.  THE ROUTINE
                                ;*IS CALLED AFTER THE PROGRAM DETERMINES THAT THE DRIVE IS POSITIONED
                                ;*INCORRECTLY AFTER A SEEK OR A UTILITY HOME SEEK OPERATION IS REQUIRED.
                                ;CALL
                                ;       JSR     PC,RESTOR
                                ;       RETURN

015314  104412                  RESTOR: SAVREG                    ;SAVE R0-R5
015316  013746  001110                  MOV     $LPERR,-(SP)      ;SAVE LOOP ON ERROR ADDRESS
015322  012737  015334  001110          MOV     #1$,$LPERR        ;LOOP ON ERROR ADDRESS
015330  013700  001266                  MOV     RPADR,R0          ;RP11 ADDRESS
015334  012760  000001  000004  1$:     MOV     #CLEAR,RPCS(R0)   ;CLEAR THE RP11
015342  012760  000001  000004          MOV     #CLEAR,RPCS(R0)   ;CLEAR THE CONTROLLER
015350  012777  015530  163712          MOV     #RESTR1,@RPVEC    ;INTERRUPT RETURN
015356  013746  001364                  MOV     DPB,-(SP)         ;SETUP RPCS CONTENTS
015362  112716  000015                  MOVB    #HOMSEK,(SP)      ;COMMAND CODE
015366  052716  020000                  BIS     #AIE,(SP)         ;ATTENTION INTERRUPT ENABLE
015372  012660  000004                  MOV     (SP)+,RPCS(R0)    ;START THE HOME SEEK
015376  005037  177776                  CLR     PSW               ;SET PRIORITY TO ZERO
015402  010637  001242                  MOV     SP,SPSAV          ;SAVE THE STACK POINTER
015406  012737  004704  001300          MOV     #2500.,STALLG     ;SET STALL VALUE TO 2500MS
015414  004037  015572                  JSR     R0,STALLA         ;WAIT THE SPECIFIED VALUE
015420  001300                          .WORD   STALLG            ;STALL VALUE STORAGE
015422  004737  015164                  JSR     PC,SAVRP          ;GET THE REGISTERS
015426  032737  001000  002004          BIT     #SUFU,$RPDS       ;DRIVE UNSAFE ?
```

```
015434  001403                          BEQ     2$              ;BR IF NOT
015436  104002                          ERROR   2               ;DRIVE UNSAFE
015440  000137  015272                  JMP     DSELCT          ;DESELECT THE DRIVE
015444  032737  040000  002004  2$:     BIT     #SUOL,$RPDS     ;IS THE DRIVE STILL ONLINE ?
015452  001003                          BNE     3$              ;BR IF IT IS
015454  104010                          ERROR   10              ;DRIVE OFFLINE AFTER POSITIONING
015456  000137  015272                  JMP     DSELCT          ;DESELECT THE DRIVE
015462  032737  100000  002004  3$:     BIT     #SURDY,$RPDS    ;IS THE DRIVE READY ?
015470  001003                          BNE     4$              ;BR IF IT IS
015472  104016                          ERROR   16              ;DRIVE NOT READY AFTER POSITIONING
015474  000137  015272                  JMP     DSELCT          ;DESELECT THE DRIVE
015500  104005                  4$:     ERROR   5               ;DRIVE HAS NOT INTERRUPTED WITHIN 1 SEC
015502  012760  000001  000004          MOV     #CLEAR,RPCS(R0) ;CLEAR THE CONTROLLER
015510  012760  000001  000004          MOV     #CLEAR,RPCS(R0) ;CLEAR THE CONTROLLER
015516  012777  000100  163420          MOV     #BIT6,@$TKS     ;SET INTERRUPT ENABLE FOR TTY KEYBOARD
015524  000177  163514                  JMP     @BYPASS         ;TERMINATE THE PRESENT TEST
015530  013706  001242          RESTR1: MOV     SPSAV,SP        ;RESTORE THE STACK POINTER
015534  005037  177776                  CLR     PSW             ;RESET THE PSW
015540  004737  015164                  JSR     PC,SAVRP        ;SAVE THE REGISTERS
015544  032737  004000  002004          BIT     #SUSI,$RPDS     ;SEEK INCOMPLETE ?
015552  001403                          BEQ     2$              ;BR IF NOT
015554  104015                          ERROR   15              ;SEEK INCOMPLETE
015556  000137  015272                  JMP     DSELCT          ;DESELECT THE DRIVE
015562  012637  001110          2$:     MOV     (SP)+,$LPERR    ;RESTORE THE LOOP ON ERROR ADDRESS
015566  104413                          RESREG                  ;RESTORE R0-R5
015570  000207                          RTS     PC              ;RETURN

                                ;*****************************************************************
                                ;SBTTL  STALL - USED TO STALL AFTER A DRIVE FUNCTION IS COMPLETE
                                ;*THIS ROUTINE WILL PROVIDE A STALL IN MILLISECONDS FOR A SPECIFIC
                                ;*AMOUNT OF TIME IF SWITCH 02 IS NOT SET OR A RANDOM AMOUNT OF TIME IF
                                ;*SWITCH 02 IS SET.
                                ;*STALL1 CONTAINS SPECIFIED TIME FOR TESTS 0-6.
                                ;CALL
                                ;       JSR     R0,@#STALL
                                ;       TIME POINTER            ;WHERE TO FIND THE STALL TIME
                                ;
                                ;       JSR     R0,@#STALLA     ;ENTER HERE TO BYPASS RANDOM STALL SELECTION
                                ;       TIME POINTER            ;WHERE TO FIND THE STALL TIME

015572  013046                  STALLA: MOV     @(R0)+,-(SP)    ;PICKUP STALL TIME
015574  000413                          BR      STALLB          ;BYPASS RANDOM STALL CHECK
015576  013046                  STALL:  MOV     @(R0)+,-(SP)    ;PICKUP STALL TIME
015600  032777  000004  163332          BIT     #SW2,@$WR       ;USE A RANDOM TIME?
015606  001406                          BEQ     STALLB          ;NO--BRANCH
015610  004737  013540                  JSR     PC,@#$RAND      ;YES--FORM RANDOM NUMBER
015614  013716  013640                  MOV     @#$LONUM,(SP)   ;AND USE IT FOR THE STALL TIME
015620  042716  177700                  BIC     #^C77,(SP)      ;BUT NEVER > 64 MILLISECONDS
015624  005046                  STALLB: CLR     -(SP)           ;CLEAR TEMP. LOCATION
015626  162766  000001  000002  2$:     SUB     #1,2(SP)        ;MORE STALL REQUIRED?
015634  103407                          BLO     4$              ;NO--BRANCH
015636  012716  000620                  MOV     #400.,(SP       ;STALL  TIME CONSTANT
015642  005700                  3$:     TST     R0              ;NOP TO KILL TIME
015644  005366  000000                  DEC     0(SP)           ;COUNT
015650  001374                          BNE     3$              ;LOOP IF MORE COUNTS NEEDED
015652  000765                          BR      2$
```

```
015654  022626                  4$:     CMP     (SP)+,(SP)+     ;CLEAN OFF THE STACK
015656  000200                          RTS     R0              ;EXIT


                                ;;*****************************************************************
                                ;.SBTTL  SORT - ROUTINE TO SORT & PRINT SEEK TIMES
                                ;*THIS ROUTINE DETERMINES THE NUMBER OF TIMES A PARTICULAR FORWARD OR REVERSE
                                ;*SEEK TIME WAS OBTAINED.  THE SEEK TIMES ARE SORTED IN DECENDING SEQUENCE
                                ;*AND THE NUMBER OF OCCURENCES OF A PARTICULAR SEEK TIME ARE TOTALED.  THE
                                ;*THE ROUTINE WILL TYPEOUT ONLY THE LONGEST 5 SEEK TIMES.  FORWARD SEEK TIMES
                                ;*ARE STORED IN A TABLE STARTING AT 'FRWSTR'; REVERSE SEEK TIMES ARE STORED
                                ;*IN A TABLE STARTING AT 'RVSTR'.
                                ;CALL
                                ;       JSR     R0,SORT         ;SORT AND TYPE THE SEEK TIMES
                                ;       NUMBER OF ITEMS         ;NUMBER OF ITEMS IN THE TABLE (405 MAX)

015660  012037  016126          SORT:   MOV     (R0)+,7$        ;GET THE ENTRY COUNT
015664  104412                          SAVREG                  ;SAVE R0-R5
015666  012705  177776                  MOV     #-2,R5          ;COUNT FOR FORWARD & REVERSE
015672  012737  024040  001162          MOV     #FRWSTR,$REG0   ;INITIALIZE POINTER TO FORWARD SEEK TIMES
015700  012737  024042  001164          MOV     #FRWSTR+2,$REG1  ;SECOND FORWARD SEEK TIME POINTER
015706  012737  023540  001166          MOV     #BUFR4,$REG2
015714  012737  023620  001170          MOV     #BUFR5,$REG3    ;INITIALIZE POINTER TO '# OF TIMES'
015722  013700  001162          1$:     MOV     $REG0,R0        ;TYPE 'SEEK TIME'
015726  013701  001164                  MOV     $REG1,R1
015732  013702  016126                  MOV     7$,R2           ;NUMBER OF ITEMS IN THE TABLE
015736  005003                          CLR     R3
015740  021011                  2$:     CMP     (R0),(R1)       ;SORT THE ITEMS & PUT THEM
015742  003404                          BLE     3$              ;IN DESCENDING ORDER
015744  011004                          MOV     (R0),R4         ;LARGER ITEMS AT TOP OF LIST.
015746  011110                          MOV     (R1),(R0)       ;SMALLER AT THE BOTTOM
015750  010411                          MOV     R4,(R1)
015752  005203                          INC     R3
015754  005720                  3$:     TST     (R0)+
015756  005721                          TST     (R1)+
015760  005302                          DEC     R2
015762  001366                          BNE     2$
015764  005703                          TST     R3              ;SORTED ALL ITEMS?
015766  001355                          BNE     1$              ;IF NOT LOOP BACK
015770  013700  001162                  MOV     $REG0,R0        ;PTR TO 'SEEK TIME'
015774  013701  001166                  MOV     $REG2,R1        ;SAVE 'SEEK TIME' HERE
016000  013702  001170                  MOV     $REG3,R2        ;SAVE '# OF TIMES' HERE
016004  010204                          MOV     R2,R4
016006  005024                          CLR     (R4)+           ;CLR OUT 5 WORDS OF
016010  005024                          CLR     (R4)+           ;'# OF TIMES'BUFR
016012  005024                          CLR     (R4)+
016014  005024                          CLR     (R4)+
016016  005024                          CLR     (R4)+
016020  013703  177773                  MOV     #-5,R3          ;SAVE ONLY 5 DIFFERENT 'SEEK TIMES'
016024  011011                          MOV     (R0),(R1)       ;FIND OUT THE '# OF TIMES'
016026  013703  016126                  MOV     7$,R3           ;EACH 'SEEK TIME' WAS
016032  022011                  4$:     CMP     (R0)+,(R1)
016034  001411                          BEQ     5$              ;OBTAINED
016036  005721                          TST     (R1)+
016040  016011  177776                  MOV     -2(R0),(R1)     ;SAVE 'SEEK TIME'
016044  005722                          TST     (R2)+
```

```
016046  012712  000001                  MOV     #1,(R2)         ;KEEP '# OF TIMES'
016052  005303                          DEC     R3
016054  001404                          BEQ     6$
016056  000765                          BR      4$
016060  005212          5$:             INC     (R2)            ;INCREMENT '# OF TIMES'
016062  005303                          DEC     R3              ;ALL DONE?
016064  001362                          BNE     4$              ;IF NOT, GO BACK
016066  005205          6$:             INC     R5              ;SORTED BOTH FORWARD & REVERSE ?
016070  001417                          BEQ     GOTYPE          ;IF YES, TYPE THE VALUES
016072  012737  025514  001162          MOV     #RVSTR,$REG0    ;IF NOT, INITIALIZE POINTER TO REVERSE SEEK TIMES
016100  012737  025516  001164          MOV     #RVSTR+2,$REG1  ;SECOND REVERSE SEEK TIME POINTER
016106  012737  023700  001166          MOV     #BUFR6,$REG2    ;SAVE 'SEEK TIME'
016114  012737  023760  001170          MOV     #BUFR7,$REG3    ;SAVE '# OF TIMES' HERE
016122  000137  015722                  JMP     1$              ;GO BACK & DO SORTING FOR REVERSE SEEK TIMES

016126  000000          7$:             .WORD   0               ;ITEMS IN TABLE COUNT HERE

                        ;TYPEOUT THE HEADER FOR THE SEEK TIME RESULTS

016130                  GOTYPE:
016130  104401  016136                  TYPE    ,65$            ;;TYPE ASCIZ STRING
016134  000421                          BR      64$             ;;GET OVER THE ASCIZ
                        ;;65$:          .ASCIZ  <15><12>/(SEEK TIME IS IN MICRO SECONDS
016200                  64$:
016200  104401  016206                  TYPE    ,67$            ;;TYPE ASCIZ STRING
016204  000420                          BR      66$             ;;GET OVER THE ASCIZ
                        ;;67$:          .ASCIZ  <15><12><12>      FORWARD         REVERSE
016246                  66$:
016246  104401  016254                  TYPE    ,69$            ;;TYPE ASCIZ STRING
016252  000420                          BR      68$             ;;GET OVER THE ASCIZ
                        ;;69$:          .ASCIZ  <15><12>    #  OF    SEEK   #  OF    SEEK
016314                  68$:
016314  104401  016322                  TYPE    ,71$            ;;TYPE ASCIZ STRING
016320  000421                          BR      70$             ;;GET OVER THE ASCIZ
                        ;;71$:          .ASCIZ  <15><12>/  SEEKS    TIME  SEEKS    TIME <15><12>
016364                  70$:

                        ;TYPEOUT THE '# OF SEEKS' & 'SEEK TIME' OBTAINED FOR EACH OF THOSE SEEKS

016364  005000          TYPTIM: CLR     R0
016366  005005                  CLR     R5
016370  104401  001217  1$:     TYPE    ,$CRLF
016374  016046  023620          MOV     BUFR5(R0),-(SP) ;GET '# OF SEEKS', IF NONE <0
016400  001436                  BEQ     3$              ;SKIP TYPING (FRWRD SEEK)
016402  104405                  TYPOS                   ;GO TYPE OUT DECIMAL '# OF SEEKS'
016404  104401                  TYPE
016406  002043                  BLNKS2
016410  016046  023540          MOV     BUFR4(R0),-(SP) ;GET 'SEEK TIME' FOR EACH OF
016414  104405                  TYPOS                   ;OF THAT '# OF SEEKS'. 'GO
                                                        ;TYPE OUT IN DECIMAL
016416  104401  016424          TYPE    ,65$            ;;TYPE ASCIZ STRING
016422  000401                  BR      64$             ;;GET OVER THE ASCIZ
                        ;;65$:          .ASCIZ  <0
016426                  64$:
016426  016046  023760  2$:     MOV     BUFR7(R0),-(SP) ;GET '# OF SEEKS', IF NONE - 0
016432  001424                  BEQ     4$              ;SKIP TYPING  REVRSE SEEK
```

```
016434  005705                          TST     R5              ;'FORWARD' VALUE FOR THIS LINE ?
016436  001404                          BEQ     6$              ;BR IF YES
016440  104401  002035                  TYPE    ,BLNKS8         ;TYPE FILLER BLANKS
016444  104401  002036                  TYPE    ,BLNKS7
016450  104405                  6$:     TYPDS                   ;TYPE OUT IN DECIMAL
016452  104401  002043                  TYPE    ,BLNKS2         ;2 BLANKS
016456  016046  023700                  MOV     BUFR6(R0),-(SP) ;GET 'SEEK TIME' & TYPE IT
016462  104405                          TYPDS                   ;OUT IN DECIMAL
016464  104401  016472                  TYPE    ,67$            ;;TYPE ASCIZ STRING
016470  000401                          BR      66$             ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  /0/
016474                          66$:
016474  000406                          BR      5$
016476  005726                  3$:     TST     (SP)+           ;POP STACK
016500  005205                          INC     R5
016502  000751                          BR      2$
016504  005726                  4$:     TST     (SP)+           ;POP STACK
016506  005705                          TST     R5
016510  001004                          BNE     7$
016512  005720                  5$:     TST     (R0)+           ;INCREMENT PTR TO TABLES
016514  020027  000012                  CMP     R0,#10.         ;ALL DONE?
016520  001323                          BNE     1$              ;IF NOT GO BACK
016522  104413                  7$:     RESREG                  ;RESTORE R0-R5
016524  000200                          RTS     R0              ;RETURN

                                ;*****************************************************************
                                ;.SBTTL  SEEK TIME MEASUREMENT ROUTINE
                                ;*THIS ROUTINE MEASURES THE TIME REQUIRED TO SEEK TO THE SPECIFIED CYLINDER
                                ;*THE ROUTINE USES THE 'SOT' COUNTER IN THE CONTROLLER TO MEASURE THE SEEK
                                ;*TIME.  THE SEEK TIME MEASUREMENT IS BASED ON THE NUMBER OF SECTORS
                                ;*OCCURING BETWEEN SEEK START AND SEEK COMPLETE.  THE RESOLUTION OF THIS
                                ;*MEASUREMENT METHOD IS + OR - 2.5 MS (ONE SECTOR TIME).
                                ;CALL
                                ;       JSR     PC,TIMSEK       ;TIME THE SEEK TO THE CYLINDER IN DPB+2
                                ;       RETURN                  ;SEEK TIME IS RETURNED IN R3

016526  010046                  TIMSEK: MOV     R0,-(SP)        ;SAVE R0
016530  010246                          MOV     R2,-(SP)        ;SAVE R2
016532  013746  001110                  MOV     $LPERR,-(SP)    ;SAVE CURRENT LOOP ON ERROR ADDRESS
016536  012737  016550  001110          MOV     #TIMSK1,$LPERR  ;NEW LOOP ON ERROR ADDRESS
016544  013700  001266                  MOV     RPADR,R0        ;RP11 BASE ADDRESS
016550  005003                  TIMSK1: CLR     R3              ;R3 WILL BE USED TO COUNT THE DISK REVOLUTIONS
016552  012737  000062  001300          MOV     #50.,STALLG     ;SET STALL VALUE TO 50MS
016560  004037  015572                  JSR     R0,STALLA       ;WAIT THE SPECIFIED VALUE
016564  001300                          .WORD   STALLG          ;STALL VALUE STORAGE
016566  016002  000014          1$:     MOV     RPDA(R0),R2     ;READ THE SECTOR/TRACK REGISTER
016572  032702  000200                  BIT     #BIT7,R2        ;WAIT FOR SECTOR 8, THEN START LOOKING
016576  001773                          BEQ     1$              ;FOR SECTOR 0.
016600  016002  000014          2$:     MOV     RPDA(R0),R2     ;READ THE SECTOR REGISTER
016604  026002  000014                  CMP     RPDA(R0),R2     ;WAS IT CHANGING ?
016610  001373                          BNE     2$              ;BR IF IT WAS
016612  032702  000360                  BIT     #360,R2         ;WAIT FOR SEC 0, INDEX MARK
016616  001370                          BNE     2$              ;AS SOON AS IT IS AT SECTOR 0, START
                                                                ;A SEEK & BEGIN TIMING
016620  012777  017066  162442          MOV     #TIMSK2,@RP.EC  ;INTERRUPT ADDRESS
```

```
016626  012760  000377  000000          MOV     #377,RPDS(R0)     ;CLEAR ANY ATTENTION BITS
016634  013760  001366  000012          MOV     DPB+2,RPCA(R0)    ;LOAD CYLINDER ADDRESS
016642  013746  001364                  MOV     DPB,-(SP)         ;SETUP COMMAND
016646  062716  020000                  BIS     #AIE,(SP)         ;SET ATTENTION INTERRUPT ENABLE
016652  012660  000004                  MOV     (SP)+,RPCS(R0)    ;ISSUE A SEEK, START TIMING
016656  005037  177776                  CLR     PSW               ;SET PRIORITY TO ZERO
016662  016002  000014          3$:     MOV     RPDA(R0),R2       ;READ THE SECTOR REGISTER
016666  026002  000014                  CMP     RPDA(R0),R2       ;WAS IT CHANGING ?
016672  001373                          BNE     3$                ;BR IF IT WAS
016674  032702  000360                  BIT     #360,R2           ;WAIT FOR SEC CNTR TO MOVE
016700  001770                          BEQ     3$                ;FROM 0 TO 1
016702  016002  000014          4$:     MOV     RPDA(R0),R2       ;READ THE SECTOR REGISTER
016706  026002  000014                  CMP     RPDA(R0),R2       ;WAS IT CHANGING ?
016712  001373                          BNE     4$                ;BR IF IT WAS
016714  032702  000360                  BIT     #360,R2           ;WAIT FOR SECTOR 0
016720  001370                          BNE     4$                ;BR IF NOT AT 0
016722  005203                          INC     R3                ;COUNT A DISK REVOLUTION (25MS
016724  022703  000050                  CMP     #40.,R3           ;1 SECOND ELAPSED ?
016730  003354                          BGT     3$                ;BR IF NOT
016732  004737  015164                  JSR     PC,SAVRP          ;SAVE THE REGISTERS
016736  032737  001000  002004          BIT     #SUFU,$RPDS       ;FILE UNSAFE ?
016744  001403                          BEQ     5$                ;BR IF NOT
016746  104002                          ERROR   2                 ;REPORT DRIVE UNSAFE
016750  000137  015272                  JMP     DSELCT            ;SETUP TO DESELECT THE DRIVE
016754                          5$:
016754  013737  002004  001126          MOV     $RPDS,$BDDAT      ;GET CONTENTS OF RPDS
016762  012737  000000  001122          MOV     #RPDS,$BDADR    ;FORM REGISTER ADDRESS FOR ERROR MESSAGE
016770  063737  001266  001122          ADD     RPADR,$BDADR      ;ADD RP11 BASE ADDRESS
016776  012737  140000  001124          MOV     #SURDY!SUOL,$GDDAT  ;WHAT REGISTER SHOULD BE
017004  013737  001126  001204          MOV     $BDDAT,$TMPO      ;MOVE REGISTER CONTENTS TO '$TMPO'
017012  042737  037777  001204          BIC     #^C140000,$TMPO   ;SAVE THE SPECIFIED BITS
017020  023737  001124  001204          CMP     $GDDAT,$TMPO      ;COMPARE THE BITS
017026  001403                          BEQ     6$                ;BR IF OK
017030  104017                          ERROR   17                ;NOT READY/OFFLINE DURING TIMING TEST
017032  000137  015272                  JMP     DSELCT            ;DESELECT THE DRIVE
017036  104005                  6$:     ERROR   5                 ;NO INTERRUPT AFTER 1 SECOND
017040  012760  000001  000004          MOV     #CLEAR,RPCS(R0)   ;CLEAR THE CONTROLLER
017046  012760  000001  000004          MOV     #CLEAR,RPCS(R0)   ;CLEAR THE CONTROLLER
017054  012777  000100  162062          MOV     #BIT06,@$TKS      ;SET KEYBOARD INTERRUPT ENABLE
017062  000177  162156                  JMP     @BYPASS           ;TERMINATE THE TEST
017066  012626                  TIMSK2: MOV     (SP)+,(SP)+       ;RESTORE THE STACK POINTER
017070  016002  000014                  MOV     RPDA(R0),R2       ;SAVE RPDA
017074  005037  177776                  CLR     PSW               ;SET PRIORITY TO ZERO
017100  032702  000360                  BIT     #360,R2           ;AT SECTOR 0 ?
017104  001001                          BNE     1$                ;BR IF NOT
017106  005203                          INC     R3                ;INCREMENT THE REVOLUTION COUNTER
017110  004737  015164          1$:     JSR     PC,SAVRP          ;STORE THE REGISTERS
017114  032737  004000  002004          BIT     #SUSI,$RPDS       ;SEEK INCOMPLETE ?
017122  001405                          BEQ     2$                ;BR IF NOT
017124  104015                          ERROR   15                ;SEEK INCOMPLETE
017126  004737  015314                  JSR     PC,RESTOR         ;DO A HOME SEEK
017132  000177  162106                  JMP     @BYPASS           ;TERMINATE THE PRESENT TEST
017136  032737  100000  000004  2$:     BIT     #ERR,RPCS         ;OTHER ERRORS ?
017144  001412                          BEQ     SKDON             ;BR IF NOT
017146  104004                          ERROR   4                 ;REPORT ERROR DURING TIMING TEST
017150  012760  000001  000004          MOV     #CLEAR,RPCS(R0)   ;CLEAR THE CONTROLLER
```

```
017156  012760  000001  000004          MOV     #CLEAR,RPCS(R0) ;CLEAR THE CONTROLLER
017164  113760  001365  000005          MOVB    DPB+1,RPCS+1(R0)  ;RESELECT THE DRIVE
017172  042702  177417          SKDON:  BIC     #^C360,R2       ;CLEAR SECTOR/TRACK BITS - LEAVE THE 'SOT'
017176  006202                          ASR     R2              ;RIGHT JUSTIFY 'SOT'
017200  006202                          ASR     R2              ;RIGHT JUSTIFY 'SOT'
017202  006202                          ASR     R2              ;RIGHT JUSTIFY 'SOT'
017204  006202                          ASR     R2              ;RIGHT JUSTIFY 'SOT'
017206  012746  000010                  MOV     #10.,-(SP)              ;;PUT THE MULTIPLER ON THE STACK
017212  010346                          MOV     R3,-(SP)                ;;PUT THE MULTIPLICAND ON THE STACK
017214  004737  013052                  JSR     PC,@#$MULT      ;;CALL THE MULTPLY ROUTINE
017220  012616                          MOV     (SP)+,(SP)      ;;DISREGARD THE MSB'S
017222  012603                          MOV     (SP)+,R3                ;;GET THE LSB'S OF THE PRODUCT
017224  060203                          ADD     R2,R3           ;ADD PARTIAL SECTOR COUNT
                                                                ;NOTE THERE IS A SCALE FACTOR
017226  012746  000372                  MOV     #250.,-(SP)             ;;PUT THE MULTIPLER ON THE STACK
017232  010346                          MOV     R3,-(SP)                ;;PUT THE MULTIPLICAND ON THE STACK
017234  004737  013052                  JSR     PC,@#$MULT      ;;CALL THE MULTPLY ROUTINE
017240  012616                          MOV     (SP)+,(SP)      ;;DISREGARD THE MSB'S
017242  012603                          MOV     (SP)+,R3                ;;GET THE LSB'S OF THE PRODUCT

                                ;;*****************************************************************
                                ;SEEK TIME (IN US) = [(R3)X(10)+(R2)]X(250)X(10)
                                ;;*****************************************************************

017244  012637  001110                  MOV     (SP)+,$LPERR    ;RESTORE THE LOOP ON ERROR ADDRESS
017250  012602                          MOV     (SP)+,R2        ;RESTORE R2
017252  012600                          MOV     (SP)+,R0        ;RESTORE R0
017254  000207                          RTS     PC              ;RETURN

                                ;;*****************************************************************
                                ;SBTTL  GETADR - GET BUS ADDRESS AND VECTOR ADDRESS
                                ;*THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS
                                ;*OF THE RP11 IS SETUP TO READ THE PROPER VALUE.
                                ;*IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
                                ;*REQUIRED.
                                ;NOTE: THIS ROUTINE DESTROYS R0-R4
                                ;CALL
                                ;
                                ;       JSR     PC,@#GETADR
                                ;       RETURN

017256  005737  001224          GETADR: TST     @#BUSADR        ;INPUT FROM TTY REQUESTED?
017262  001447                          BEQ     7$              ;NO--BRANCH
017264  005037  001224                  CLR     @#BUSADR        ;YES--CLEAR THE REQUEST FLAG
017270  012700  001266          1$:     MOV     #RPADR,R0       ;FIRST ADDRESS
017274  012703  021440                  MOV     #MRPCS,R3       ;"RPADR="
017300  011004                          MOV     (R0),R4         ;PRESENT RP11 ADDRESS
017302  004037  017460                  JSR     R0,@#GETNUM     ;GET NEW RP11 ADDRESS
017306  000402                          BR      2$              ;COMMA
017310  000767                          BR      1$              ;PERIOD
017312  000430                          BR      5$              ;DOUBLE PERIOD
017314  010420                  2$:     MOV     R4,(R0)+        ;SAVE NEW RHCS1
017316  012703  021450                  MOV     #MRPVEC,R3      ;"RPVEC="
017322  011004                          MOV     (R0),R4         ;PRESENT RP11 VECTOR ADDRESS
017324  004037  017460                  JSR     R0,@#GETNUM     ;GET NEW RPVEC
017330  000402                          BR      3$              ;COMMA
```

```
017332  000756                          BR      1$              ;PERIOD
017334  000417                          BR      5$              ;DOUBLE PERIOD
017336  010420                  3$:     MOV     R4,(R0)+        ;SAVE NEW RPVEC
017340  012703  021461                  MOV     #MRPPRI,R3      ;"RPPRIO="
017344  011004                          MOV     (R0),R4         ;PRESENT RP11 PRIORITY LEVEL
017346  006304                          ASL     R4              ;POSITION FOR TYPEOUT
017350  006304                          ASL     R4
017352  006304                          ASL     R4
017354  000304                          SWAB    R4
017356  004037  017460                  JSR     R0,@#GETNUM     ;GET NEW RPPRIO
017362  000402                          BR      4$              ;COMMA
017364  000401                          BR      4$              ;PERIOD
017366  000404                          BR      6$              ;DOUBLE PERIOD
017370  010210                  4$:     MOV     R2,(R0)         ;SAVE NEW RPPRIO
017372  000736                          BR      1$              ;LOOP
017374  010410                  5$:     MOV     R4,(R0)         ;SAVE INPUT
017376  000401                          BR      7$
017400  010210                  6$:     MOV     R2,(R0)         ;SAVE PRIORITY
017402  013701  000004          7$:     MOV     @#ERRVEC,R1     ;SAVE THE ERROR VECTOR
017406  012737  017426  000004          MOV     #8$,@#ERRVEC    ;SETUP FOR TRAP
017414  005777  161646                  TST     @RPADR          ;CHECK FOR RP11
017420  010137  000004                  MOV     R1,@#ERRVEC     ;RESTORE ERROR VECTOR
017424  000207                          RTS     PC              ;RETURN
017426  010137  000004          8$:     MOV     R1,@#ERRVEC     ;RESTORE ERROR VECTOR
017432  022626                          CMP     (SP)+,(SP)+     ;CLEAN OFF THE STACK
017434  104001                          ERROR   1               ;REPORT THE ERROR
017436  005737  000042                  TST     @#42            ;IS THERE A MONITOR?
017442  001712                          BEQ     1$              ;NO--GO ASK FOR ADDRESS
017444  005037  001232                  CLR     @#DRVSEL        ;YES--NO DRIVES SELECTED
017450  005037  007650                  CLR     @#$EOPCT        ;NO PASSES
017454  000137  007474                  JMP     @#$EOP          ;GO TO END OF PROGRAM

                                ;;*******************************************************
                                ;SBTTL  GETNUM - ROUTINE TO GET A NUMBER
                                ;*THIS ROUTINE WILL TYPE AN ASCIZ MESSAGE AND THEN
                                ;*INPUT AN ASCIZ STRING AND CHANGE THE STRING TO OCTAL
                                ;*IF REQUIRED.
                                ;*NOTE: THIS ROUTINE DESTROYS R1
                                ;CALL
                                ;       MOV     #ADR,R3         ;ADDRESS OF ASCIZ MESSAGE
                                ;       MOV     #NUM,R4         ;OCTAL NUMBER
                                ;       JSR     R0,@#GETNUM
                                ;       RETURN1                 ;INPUT TERMINATED WITH A COMMA
                                ;       RETURN2                 ;WITH A PERIOD
                                ;       RETURN3                 ;WITH A DOUBLE PERIOD
                                ;                               ;R4=INPUT NUMBER AND
                                ;                               ;R2=R4*32 FOR ALL
                                ;                               ;THREE RETURNS

017460  010337  017466          GETNUM: MOV     R3,2$           ;SAVE MESSAGE POINTER
017464  104401                  1$:     TYPE                    ;TYPE THE MESSAGE
017466  000000                  2$:     .WORD   0               ;MESSAGE POINTER GOES HERE
017470  010446                          MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
017472  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII,ALL DIGITS
017474  104411                          RDLIN                   ;READ AN ASCIZ STRING
017476  012601                          MOV     (SP)+,R1        ;ADDRESS OF FIRST CHARACTER
```

```
017500  004037  021044          JSR     R0,@#CK.CHR     ;CHECK ONE CHARACTER
017504  017464                  1$                      ;ILLEGAL CHARACTER
017506  017464                  1$                      ;CARRIAGE RETURN
017510  017520                  3$                      ;" "
017512  017544                  7$                      ;","
017514  017552                  8$                      ;"."
017516  017464                  1$                      ;DIGIT 0-9
017520                  3$:
017520  004037  021300          JSR     R0,@#CK.NUM     ;CHECK THE NUMBER
017524  017464                  1$                      ;ILLEGAL INPUT
017526  017540                  6$                      ;TERMINATED WITH A "."
017530  017536                  5$                      ;TERMINATED WITH A ":"
017532  017534                  4$                      ;TERMINATED WITH A ".."
017534  005720                  4$:     TST     (R0)+   ;DOUBLE PERIOD
017536  005720                  5$:     TST     (R0)+   ;SINGLE PERIOD
017540  010204                  6$:     MOV     R2,R4   ;COMMA--SAVE INPUT NUMBER
017542  000414                          BR      10$     ;GO TO EXIT
017544  105711                  7$:     TSTB    (R1)    ;TERMINATOR AFTER A COMMA?
017546  001346                          BNE     1$      ;NO--LOOP
017550  000411                          BR      10$     ;YES--EXIT
017552  105711                  8$:     TSTB    (R1)    ;TERMINATOR AFTER A PERIOD?
017554  001406                          BEQ     9$      ;YES--EXIT
017556  122721  000056                  CMPB    #".,(R1)+ ;NO--DOUBLE PERIOD?
017562  001340                          BNE     1$      ;NO--LOOP
017564  105711                          TSTB    (R1)    ;YES--TERMINATOR?
017566  001336                          BNE     1$      ;NO--LOOP
017570  005720                          TST     (R0)+   ;DOUBLE PERIOD
017572  005720                  9$:     TST     (R0)+   ;PERIOD
017574  010402                  10$:    MOV     R4,R2   ;COMMA--POSITION THE
017576  000302                          SWAB    R2      ;NUMBER IN CASE IT
017600  006202                          ASR     R2      ;IS THE PRIORITY LEVEL
017602  006202                          ASR     R2
017604  006202                          ASR     R2
017606  000200                          RTS     R0      ;EXIT

                                ;;*******************************************************
                                ;SBTTL  GT.PRM - GET PARAMETERS
                                ;*THIS ROUTINE IS USED TO CHANGE OR MODIFY
                                ;*THE TEST PARAMETERS.  IT GIVES THE OPERATOR
                                ;*THE CAPABILITY OF SPECIFYING WHICH DRIVES TO TEST, WHICH
                                ;*TESTS TO RUN AND HOW MANY TIMES TO
                                ;*REPEAT EACH TEST

017610  104412                  GT.PRM: SAVREG          ;SAVE R0-R5
017612  005037  001232          GT.PR1: CLR     DRVSEL  ;NO DRIVE SELECTED
017616  104401  017624                  TYPE    ,65$    ;;TYPE ASCIZ STRING
017622  000406                          BR      64$     ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12> DRIVE S)=
017640                          64$:
017640  104411                          RDLIN           ;READ TTY
017642  012601                          MOV     SP,+,R1 ;ADDRESS OF ASCIZ STRING
017644  004037  021044                  JSR     R0,@#CK.CHR ;CHECK ONE CHARACTER
017650  017612                          GT.PR1          ;ILLEGAL CHARACTER
017652  017612                          GT.PR1          ;CARRIAGE RETURN
017654  017612                          GT.PR1          ;" "
017656  017612                          GT.PR1          ;","
```

```
017660  017612                          GT.PR1                  ;".."
017662  017664                          1$                      ;DIGIT 0-9
017664  005301                  1$:     DEC     R1
017666                          2$:
017666  012702  000007                  MOV     #7,R2   ;UPPER LIMIT OF INPUT
017672  004037  021120                  JSR     R0,@#CK.DIG     ;CHECK THE DIGIT(S)
017676  017612                          GT.PR1                  ;ILLEGAL INPUT
017700  017612                          GT.PR1                  ;INPUT TO LARGE
017702  017710                          3$                      ;TERMINATED WITH A ","
017704  017730                          4$                      ;TERMINATED WITH A ":"
017706  017730                          4$                      ;TERMINATED WITH A "."
017710  156237  001314  001232  3$:     BISB    ATABIT(R2),DRVSEL ;SET THE DRIVE SELECTED BIT
017716  105741                          TSTB    -(R1)           ;WAS THE LINE TERMINATED?
017720  001362                          BNE     2$              ;NO-GET THE NEXT DRIVE
017722  005037  001234                  CLR     @#TSTNMS        ;DESELECT ALL TESTS
017726  000405                          BR      GTTST1          ;YES--SELECT TEST
017730  156237  001314  001232  4$:     BISB    ATABIT(R2),DRVSEL ;SET THE SELECTED DRIVE BITS
017736  104413                  GT.PR2: RESREG                  ;RESTORE R0-R5
017740  000207                          RTS     PC              ;EXIT

017742                          GTTST1:
017742  104401  017750                  TYPE    ,65$            ;;TYPE ASCIZ STRING
017746  000403                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  /TEST=/
017756                          64$:
017756  104411                          RDLIN                   ;READ AN ASCIZ STRING
017760  012601                          MOV     (SP)+,R1        ;POINTER TO R1
017762  122711  000056                  CMPB    #'.,(R1)        ;DOUBLE PERIOD?
017766  001007                          BNE     1$              ;NO--BRANCH
017770  122761  000056  000001          CMPB    #'.,1(R1)
017776  001003                          BNE     1$
020000  105761  000002                  TSTB    2(R1)           ;"CR"?
020004  001754                          BEQ     GT.PR2          ;YES--EXIT
020006  005037  001234          1$:     CLR     TSTNMS          ;NO TEST SELECTED
020012  005003                          CLR     R3              ;NO TEST TO BE OPENED
020014  005004                  GTTST2: CLR     R4              ;NO TEST BITS SET
020016  004037  020770                  JSR     R0,@#CK.OCT     ;OCTAL DIGIT?
020022  000460                          BR      GTTST4          ;NO--BRANCH
020024  010205                          MOV     R2,R5           ;YES--SAVE IT
020026  005201                          INC     R1              ;MOVE TO NEXT CHARACTER
020030  004037  020770                  JSR     R0,@#CK.OCT     ;OCTAL DIGIT
020034  000405                          BR      1$              ;NO--BRANCH
020036  005201                          INC     R1              ;MOVE TO NEXT CHARACTER
020040  006305                          ASL     R5              ;SCALE HIGH DIGIT
020042  006305                          ASL     R5
020044  006305                          ASL     R5
020046  060502                          ADD     R5,R2           ;COMBINE HIGH & LOW DIGITS
020050  020227  000012          1$:     CMP     R2,#$TN-1       ;LEGAL TEST NUMBER?
020054  003332                          BGT     GTTST1          ;NO--BRANCH
020056  006302                          ASL     R2
020060  016204  001324                  MOV     BITS(R2),R4     ;SELECT TEST
020064  121127  000055                  CMPB    (R1),#'-        ;TEST STRING?
020070  001035                          BNE     GTTST4          ;NO--BRANCH
020072  005201                          INC     R1              ;YES--MOVE TO NEXT CHARACTER
020074  004037  020770                  JSR     R0,@#CK.OCT     ;OCTAL DIGIT?
020100  000720                          BR      GTTST1          ;NO--BRANCH
```

```
020102  010205                          MOV     R2,R5           ;YES--SAVE IT
020104  005201                          INC     R1              ;MOVE TO NEXT CHARACTER
020106  004037  020770                  JSR     R0,@#CK.OCT     ;OCTAL DIGIT?
020112  000405                          BR      2$              ;NO--BRANCH
020114  005201                          INC     R1              ;YES--MOVE TO NEXT CHARACTER
020116  006305                          ASL     R5              ;SCALE HIGH DIGIT
020120  006305                          ASL     R5
020122  006305                          ASL     R5
020124  060502                          ADD     R5,R2           ;COMBINE HIGH & LOW DIGIT
020126  020227  000012          2$:     CMP     R2,#$TN-1       ;LEGAL TEST NUMBER?
020132  003303                          BGT     GTTST1          ;NO--BRANCH
020134  006302                          ASL     R2
020136  020462  001324                  CMP     R4,BITS(R2      ;IS THE FIRST NUMBER OF THE
```

```
                                                                        ;STRING SMALLER THAN THE LAST?
020142  002277                          BGE     GTTST1                  ;NO--BRANCH
020144  056204  001324          3$:     BIS     BITS(R2),R4             ;YES--SET TEST SELECT BIT FOR SELECTED TESTS
020150  005742                          TST     -(R2)                   ;ADJUST POINTER
020152  036204  001324                  BIT     BITS(R2),R4             ;DONE?
020156  001772                          BEQ     3$                      ;NO--BRANCH
020160  000401                          BR      GTTST4                  ;SKIP THE INCREMENT
020162  005201                  GTTST3: INC     R1                      ;MOVE TO NEXT CHARACTER
020164  050437  001234          GTTST4: BIS     R4,TSTNMS               ;SET SELECTED TEST BITS INTO
                                                                        ;TEST SELECT WORD
020170  121127  000056                  CMPB    (R1),#'.                ;"PERIOD"?
020174  001420                          BEQ     GTTST5                  ;YES--BRANCH
020176  005704                          TST     R4                      ;ANY TEST SELECTED THIS CYCLE?
020200  001660                          BEQ     GTTST1                  ;NO--BRANCH
020202  121127  000057                  CMPB    (R1),#'/                ;"OPEN"?
020206  001002                          BNE     1$                      ;NO--BRANCH
020210  050403                          BIS     R4,R3                   ;YES--SET BITS FOR TEST TO OPEN
```

```
020212  000403                          BR      2$
020214  121127  000054          1$:     CMPB    (R1),#',        ;"COMMA"?
020220  001250                          BNE     GTTST1          ;NO--BRANCH
020222  005201                  2$:     INC     R1              ;MOVE TO NEXT CHARACTER
020224  105711                          TSTB    (R1)            ;"CR"?
020226  001272                          BNE     GTTST2          ;NO--GO GET NEXT CHARACTER
020230  005703                          TST     R3              ;ANY TESTS TO OPEN?
020232  001026                          BNE     OPNTST          ;YES--BRANCH
020234  000642                          BR      GTTST1          ;NO--START AGAIN
020236  005201                  GTTST5: INC     R1              ;MOVE TO NEXT CHARACTER
020240  121127  000056                  CMPB    (R1),#'.        ;"PERIOD"?
020244  001410                          BEQ     GTTST6          ;YES--BRANCH
020246  105711                          TSTB    (R1)            ;"CR"?
020250  001402                          BEQ     1$              ;YES--BRANCH
020252  000137  017742                  JMP     GTTST1
020256  005703                  1$:     TST     R3              ;ANY TEST TO BE OPENED?
020260  001013                          BNE     OPNTST          ;YES--BRANCH
020262  000137  017736                  JMP     GT.PR2          ;NO--GO START TESTING
020266  005201                  GTTST6: INC     R1              ;MOVE TO NEXT CHARACTER
020270  105711                          TSTB    (R1)            ;"CR"?
020272  001402                          BEQ     1$              ;YES--BRANCH
020274  000137  017742                  JMP     GTTST1          ;NO--GO ASK FOR TEST
020300  005703                  1$:     TST     R3              ;ANY TEST TO BE OPENED?
020302  001002                          BNE     OPNTST          ;YES--BRANCH
020304  000137  017736                  JMP     GT.PR2          ;NO--GO START TESTING
                                ;OPEN THE TEST FOR CHANGES
020310  104412                  OPNTST: SAVREG                  ;SAVE R0-R5
020312  005027                          CLR     (PC)+           ;START WITH TEST 0
020314  000000                  OPN.CT: .WORD   0               ;COUNT STORED HERE
020316  000411                          BR      OPN.2           ;SKIP THE INCREMENT
020320  005237  020314          OPN.1:  INC     OPN.CT          ;MOVE TO THE NEXT TEST
020324  022737  000012  020314          CMP     #$TN-1,OPN.CT   ;TEST NUMBER TO BIG?
020332  002003                          BGE     OPN.2           ;NO--OPEN THE NEXT TEST
020334  104413                          RESREG                  ;RESTORE R0-R5
020336  000137  017742                  JMP     GTTST1          ;YES--GO ASK FOR MORE TESTS
020342  013705  020314          OPN.2:  MOV     OPN.CT,R5       ;SETUP TO USE THE
020346  006305                          ASL     R5              ;TEST NUMBER AS AN INDEX
020350  036503  001324                  BIT     BITS(R5),R3     ;OPEN THIS TEST?
020354  001761                          BEQ     OPN.1           ;NO--MOVE TO NEXT TEST
020356  104401  020364                  TYPE    ,65$            ;;TYPE ASCIZ STRING
020362  000404                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  / TEST
020374                          64$:
020374  013746  020314                  MOV     OPN.CT,-(SP)    ;;SAVE OPN.CT FOR TYPEOUT
                                                                ;;TEST NUMBER
020400  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
020402     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
020403     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
020404  104401  001217                  TYPE    ,$CRLF          ;TYPE "CR" & "LF"
020410  016500  001406                  MOV     PRMPT(R5),R0    ;PICKUP PARAMETER POINTER
020414  011046                          MOV     (R0),-(SP)      ;SAVE THE VARIABLE INDICATOR
020416  012702  001372                  MOV     #PRM,R2         ;FIRST ADDRESS OF TABLE
020422  000405                          BR      2$
020424  006216                  1$:     ASR     (SP)            ;CHECK FOR A VARIABLE
020426  103403                          BCS     2$              ;GO MOVE THIS ONE
020430  001404                          BEQ     OPNPRM          ;DONE
```

```
020432  005722                          TST     (R2)+           ;BUMP THE POINTER
020434  000773                          BR      1$
020436  012022                  2$:     MOV     (R0)+,(R2)+     ;MOVE THIS VARIABLE INTO THE
020440  000771                          BR      1$              ;COMMON AREA
020442  013716  001372          OPNPRM: MOV     @#PRM,(SP)      ;GET THE VARIABLE INDICATOR
020446  005004                          CLR     R4              ;ZERO THE INDEX
020450  006216                  1$:     ASR     (SP)            ;CHECK FOR A VARIABLE
020452  103403                          BCS     3$              ;GO GET IT
020454  001772                          BEQ     OPNPRM          ;OUT OF VARIABLES
020456  005724                  2$:     TST     (R4)+           ;UPDATE THE INDEX
020460  000773                          BR      1$
020462  104401  002043          3$:     TYPE    ,BLNKS2         ;TYPE SPACES
020466  016437  001446  020476          MOV     PRMMSG(R4),4$   ;TYPE THE NAME OF THIS VARIABLE
020474  104401                          TYPE
020476  000000                  4$:     .WORD   0
020500  104401  021436                  TYPE    ,MSG.EQ         ;TYPE "="
020504  016446  001374                  MOV     RPT(R4),-(SP)   ;PUT RPT(R4) ON THE STACK
020510  004737  013250                  JSR     PC,$SB2D        ;CONVERT RPT(R4)
020514  004737  013500                  JSR     PC,$SUPRS       ;TYPE RPT(R4)
020520  104411                          RDLIN
020522  012601                          MOV     (SP)+,R1        ;READ AN ASCIZ STRING
020524  004037  021044                  JSR     R0,@#CK.CHR     ;CHECK ONE CHARACTER
020530  020462                          3$                      ;ILLEGAL CHARACTER
020532  020462                          3$                      ;CARRIAGE RETURN
020534  020600                          7$                      ;"/"
020536  020544                          5$                      ;":"
020540  020552                          6$                      ;"."
020542  020462                          3$                      ;DIGIT 0-9
020544  105711                  5$:     TSTB    (R1)            ;"CR"?
020546  001345                          BNE     3$              ;NO--STAY ON THIS VARIABLE
020550  000742                          BR      2$              ;YES--MOVE TO NEXT VARIABLE
020552  105711                  6$:     TSTB    (R1)            ;IS THERE A "CR" AFTER THE PERIOD?
020554  001002                          BNE     64$             ;NO
020556  000137  020634                  JMP     OPN.N2          ;YES--GO CLOSE THIS TEST
020562  122721  000056          64$:    CMPB    #'.,(R1)+       ;DOUBLE PERIOD?
020566  001335                          BNE     3$              ;NO--GO ASK FOR THIS VARIABLE
020570  105711                          TSTB    (R1)            ;YES--IS A "CR" AFTER THE DOUBLE PERIOD?
020572  001333                          BNE     3$              ;NO--ASK FOR THIS VARIABLE AGAIN
020574  000137  020652                  JMP     OPN.X2          ;YES--CLOSE ALL TEST
020600                          7$:
020600  016402  001434                  MOV     PRMLMT(R4),R2   ;UPPER LIMIT OF INPUT
020604  004037  021120                  JSR     R0,@#CK.DIG     ;CHECK THE DIGIT(S)
020610  020462                          3$                      ;ILLEGAL INPUT
020612  020462                          3$                      ;INPUT TO LARGE
020614  020622                          8$                      ;TERMINATED WITH A ":"
020616  020630                          OPN.N1                  ;TERMINATED WITH A "."
020620  020646                          OPN.X1                  ;TERMINATED WITH A ".."
020622  010264  001374          8$:     MOV     R2,RPT(R4)      ;SAVE THIS VARIABLE
020626  000713                          BR      2$              ;MOVE TO NEXT VARIABLE
020630  010264  001374          OPN.N1: MOV     R2,RPT(R4)      ;SAVE THIS VARIABLE
020634  005726                  OPN.N2: TST     (SP)+           ;CLEAN OFF THE STACK
020636  004737  020710                  JSR     PC,CLOSE        ;CLOSE THIS TEST
020642  000137  020320                  JMP     OPN.1           ;GO OPEN THE NEXT TEST
020646  010264  001374          OPN.X1: MOV     R2,RPT(R4)      ;SAVE THIS VARIABLE
020652  005726                  OPN.X2: TST     (SP)+           ;CLEAN OFF THE STACK
020654  004737  020710          1$:     JSR     PC,CLOSE        ;CLOSE THIS TEST
```

```
020660  005725                  2$:     TST     (R5)+           ;UPDATE THE INDEX
020662  020527  000034                  CMP     R5,#16*2        ;INDEX TO BIG?
020666  002404                          BLT     3$              ;NO--BRANCH
020670  104413                          RESREG                  ;YES, RESTORE R0-R5
020672  104413                          RESREG                  ;YES, RESTORE R0-R5
020674  000137  017736                  JMP     GT.PR2          ;GO TO EXIT
020700  036503  001324          3$:     BIT     BITS(R5),R3     ;IS THIS TEST OPEN FOR CHANGE?
020704  001363                          BNE     1$              ;YES--GO CLOSE IT
020706  000764                          BR      2$              ;NO--MOVE TO NEXT TEST
020710  104412                  CLOSE:  SAVREG                  ;SAVE R0-R5
020712  012700  001372                  MOV     #PRM,R0         ;"FROM" ADDRESS
020716  016501  001406                  MOV     PRMPT(R5),R1    ;"TO" ADDRESS
020722  012002                          MOV     (R0)+,R2        ;"FROM" INDICATOR
020724  012103                          MOV     (R1)+,R3        ;"TO" INDICATOR
020726  012704  000001                  MOV     #1,R4           ;TEST BIT START A "RPT"
020732  030402                  1$:     BIT     R4,R2           ;PARAMETER TO BE MOVED?
020734  001403                          BEQ     2$              ;NO--BRANCH
020736  030403                          BIT     R4,R3           ;A PLACE TO PUT IT?
020740  001404                          BEQ     3$              ;NO--BRANCH
020742  011011                          MOV     (R0),(R1)       ;YES--MOVE "FROM" TO "TO"
020744  030403                  2$:     BIT     R4,R3           ;"TO" PARAMETER?
020746  001401                          BEQ     3$              ;NO--BRANCH
020750  005721                          TST     (R1)+           ;YES--UPDATE THE POIN ER
020752  005720                  3$:     TST     (R0)+           ;UPDATE FROM POINTER
020754  006304                          ASL     R4              ;POSITION THE TEST BIT
020756  032704  002000                  BIT     #BIT10,R4       ;DONE?
020762  001763                          BEQ     1$              ;NO--BRANCH
020764  104413                          RESREG                  ;YES, RESTORE R0-R5
020766  000207                          RTS     PC              ;RETURN
                                ;*******************************************************************
                                .SBTTL  CK.OCT -- CHECK FOR OCTAL CHARACTER
                                ;*THIS ROUTINE IS USED TO CHECK IF AN
                                ;*ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
                                ;CALL
                                ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
                                ;       JSR     R0,@#CK.OCT     ;CHECK THE CHARACTER
                                ;       RETURN1                 ;CHARACTER IS NOT BETWEEN 0-7
                                ;       RETURN2                 ;CHARACTER IS IN R2 AS A
                                ;                               ;OCTAL DIGIT

020770  121127  000060          CK.OCT: CMPB    (R1),#'0        ;LESS THAN ZERO?
020774  103407                          BLO     1$              ;YES -- BRANCH
020776  121127  000067                  CMPB    (R1),#'7        ;GREATER THAN SEVEN?
021002  101004                          BHI     1$              ;YES -- BRANCH
021004  111102                          MOVB    (R1),R2         ;GET THE CHARACTER
021006  042702  177770                  BIC     #^C7,R2         ;STRIP AWAY THE ASCII
021012  005720                          TST     (R0)+           ;ADJUST FOR RETURN
021014  000200                  1$:     RTS     R0              ;RETURN

                                ;*******************************************************************
                                .SBTTL  CK.DEC -- CHECK FOR DECIMAL CHARACTER
                                ;*THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
                                ;*AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
                                ;CALL
                                ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
                                ;       JSR     R0,@#CK.DEC     ;CHECK THE CHARACTER
```

```
                                        ;       RETURN1                 ;NOT BETWEEN 0 AND 9
                                        ;       RETURN2                 ;BETWEEN 0 AND 9
                                        ;                               ;R2 = DIGIT
021016  121127  000060                  CK.DEC: CMPB    (R1),#'0        ;LESS THAN ZERO?
021022  103407                                  BLO     1$              ;YES -- BRANCH
021024  121127  000071                          CMPB    (R1),#'9                ;GREATER THAN NINE?
021030  101004                                  BHI     1$              ;YES -- BRANCH
021032  111102                                  MOVB    (R1),R2         ;GET THE CHARACTER
021034  042702  000060                          BIC     #'0,R2          ;STRIP AWAY THE ASCII
021040  005720                                  TST     (R0)+           ;ADJUST FOR RETURN
021042  000200                          1$:     RTS     R0              ;RETURN

                                        ;*******************************************************************
                                        ;SBTTL  CK.CHR -- CHECK CHARACTER
                                        ;*THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
                                        ;*DETERMINE WHAT IT IS.
                                        ;CALL
                                        ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
                                        ;       JSR     R0,@#CK.CHR     ;CHECK CHARACTER
                                        ;       RETURN  ADR1            ;UNKNOWN CHARACTER
                                        ;       RETURN  ADR2            ;CARRIAGE RETURN * (R1)=ADR+1
                                        ;       RETURN  ADR3            ;SLASH * (R1)=ADR+1
                                        ;       RETURN  ADR4            ;COMMA * (R1)=ADR+1
                                        ;       RETURN  ADR5            ;PERIOD * (R1)=ADR+1
                                        ;       RETURN  ADR6            ;DIGIT BETWEEN 0 AND 9.
                                        ;                               ;R2 = DIGIT * (R1)=ADR+1

021044  105711                          CK.CHR: TSTB    (R1)            ;"CARRIAGE RETURN"?
021046  001420                                  BEQ     4$              ;YES -- BRANCH
021050  121127  000057                          CMPB    (R1),#'         ;"SLASH"?
021054  001414                                  BEQ     3$              ;YES -- BRANCH
021056  121127  000054                          CMPB    (R1),#',        ;"COMMA"?
021062  001410                                  BEQ     2$              ;YES -- BRANCH
021064  121127  000056                          CMPB    (R1),#'.        ;"PERIOD"?
021070  001404                                  BEQ     1$              ;YES -- BRANCH
021072  004037  021016                          JSR     R0,@#CK.DEC     ;"DIGIT"?
021076  000406                                  BR      5$              ;NO -- BRANCH
021100  005720                                  TST     (R0)+           ;DIGIT BETWEEN 0-9
021102  005720                          1$:     TST     (R0)+           ;PERIOD
021104  005720                          2$:     TST     (R0)+           ;COMMA
021106  005720                          3$:     TST     (R0)+           ;SLASH
021110  005720                          4$:     TST     (R0)+           ;CARRIAGE RETURN
021112  005201                                  INC     R1              ;MOVE POINTER TO NEXT CHARACTER
021114  011000                          5$:     MOV     (R0),R0         ;UNKNOWN CHARACTER
021116  000200                                  RTS     R0              ;RETURN

                                        ;*******************************************************************
                                        ;SBTTL  CK.DIG - CHECK DIGIT
                                        ;*THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL
                                        ;*CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
                                        ;CALL
                                        ;       MOV     #ADR,R1         ;ADDRESS OF ASCIZ STRING
                                        ;       MOV     #NUM,R2         ;MAX. MAGNITUDE OF INPUT NUMBER
                                        ;       JSR     R0,@#CK.DIG     ;CHECK DIGITS
                                        ;       RETURN  ADR1            ;ILLEGAL CHARACTER -- R2=?
```

```
                                        ;       RETURN  ADR2            ;INPUT NUMBER TO LARGE -- R2=
                                        ;       RETURN  ADR3            ;"COMMA" -- R2 = NUMBER
                                        ;       RETURN  ADR4            ;"PERIOD" -- R2 = NUMBER
                                        ;       RETURN  ADR5            ;"PERIOD-PERIOD" -- R2 = NUMBER

021120  010446                          CK.DIG: MOV     R4,-(SP)        ;SAVE R4
021122  010346                                  MOV     R3,-(SP)        ;SAVE R3
021124  010246                                  MOV     R2,-(SP)        ;SAVE THE MAX. SIZE ON THE STACK
021126  005002                                  CLR     R2              ;START WITH 0
021130  005003                                  CLR     R3
021132  005004                                  CLR     R4
021134  004037  021044                          JSR     R0,@#CK.CHR     ;CHECK ONE CHARACTER
021140  021264                                  8$                      ;ILLEGAL CHARACTER
021142  021264                                  8$                      ;CARRIAGE RETURN
021144  021264                                  8$                      ;"/"
021146  021264                                  8$                      ;"."
021150  021264                                  8$                      ;"-"
021152  021154                                  1$                      ;DIGIT 0-9
021154  006303                          1$:     ASL     R3              ;*2
021156  010346                                  MOV     R3,-(SP)        ;SAVE *2
021160  006303                                  ASL     R3              ;*4
021162  006303                                  ASL     R3              ;*8
021164  062603                                  ADD     (SP)+,R3        ;(*8)+(*2)=*10.
021166  060203                                  ADD     R2,R3           ;UPDATE THE INPUT NUMBER
021170  004037  021044                          JSR     R0,@#CK.CHR     ;CHECK ONE CHARACTER
021174  021264                                  8$                      ;ILLEGAL CHARACTER
021176  021264                                  8$                      ;CARRIAGE RETURN
021200  021264                                  8$                      ;" "
021202  021212                                  3$                      ;"."
021204  021210                                  2$                      ;"-"
021206  021154                                  1$                      ;DIGIT 0-9
021210  005724                          2$:     TST     (R4)+           ;"PERIOD"
021212  005724                          3$:     TST     (R4)+           ;"COMMA"
021214  004037  021044                          JSR     R0,@#CK.CHR     ;CHECK ONE CHARACTER
021220  021264                                  8$                      ;ILLEGAL CHARACTER
021222  021254                                  6$                      ;CARRIAGE RETURN
021224  021264                                  8$                      ;"/"
021226  021264                                  8$                      ;"."
021230  021234                                  4$                      ;"-"
021232  021244                                  5$                      ;DIGIT 0-9
021234  005724                          4$:     TST     (R4)+           ;"PERIOD-PERIOD"
021236  105711                                  TSTB    (R1)            ;"CR"?
021240  001405                                  BEQ     6$              ;YES--BRANCH
021242  000410                                  BR      8$
021244  126127  177776  000054          5$:     CMPB    -2(R1),#',      ;WAS CHARACTER BEFORE THE DIGIT A COMMA?
021252  001004                                  BNE     8$              ;NO--EXIT
021254  020316                          6$:     CMP     R3,(SP)         ;INPUT TO LARGE?
021256  101001                                  BHI     7$              ;YES -- BRANCH
021260  060400                                  ADD     R4,R0           ;ADJUST RETURN ADDRESS
021262  005720                          7$:     TST     (R0)+
021264  010302                          8$:     MOV     R3,R2           ;NUMBER TO R2
021266  005726                                  TST     (SP)+           ;CLEAN MAX. SIZE OFF OF STACK
021270  012603                                  MOV     (SP)+,R3        ;RESTORE R3
021272  012604                                  MOV     (SP)+,R4        ;RESTORE R4
021274  011000                                  MOV     (R0),R0         ;GET RETURN ADDRESS
021276  000200                                  RTS     R0              ;RETURN
```

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

021300  010346                          CK.NUM: MOV     R3,-(SP         ;SAVE R3
021302  005003                                  CLR     R3              ;START NUMBER AT ZERO
021304  004037  020770                          JSR     R0,@#CK.OCT     ;OCTAL DIGIT?
021310  000440                                  BR      6$              ;NO--BRANCH
021312  005201                          1$:     INC     R1              ;MOVE TO NEXT CHARACTER
021314  006303                                  ASL     R3              ;FOR THE OCTAL NUMBER IN R3
021316  103435                                  BCS     6$              ;DON'T LET IT GET TO BIG
021320  006303                                  ASL     R3
021322  103433                                  BCS     6$
021324  006303                                  ASL     R3
021326  103431                                  BCS     6$
021330  060203                                  ADD     R2,R3
021332  004037  020770                          JSR     R0,@#CK.OCT     ;IS THIS AN OCTAL DIGIT?
021336  000401                                  BR      2$              ;NO--FIND OUT WHAT IT IS
021340  000764                                  BR      1$              ;YES--MAKE IT PART OF THE NUMBER
021342  010302                          2$:     MOV     R3,R2           ;SAVE THE OCTAL NUMBER
021344  005003                                  CLR     R3              ;START WITH ZERO INDEX
021346  004037  021044                          JSR     R0,@#CK.CHR     ;CHECK ONE CHARACTER
021352  021412                                  6$                      ;ILLEGAL CHARACTER
021354  021412                                  6$                      ;CARRIAGE RETURN
021356  021412                                  6$                      ;"/"
021360  021402                                  5$                      ;":"
021362  021366                                  3$                      ;"."
021364  021412                                  6$                      ;DIGIT 0-9
021366  005723                          3$:     TST     (R3)+           ;"PERIOD"
021370  121127  000056                          CMPB    (R1),#'.        ;"PERIOD-PERIOD"?
021374  001002                                  BNE     5$              ;NO--BRANCH
021376  005201                                  INC     R1              ;YES--ADVANCE THE POINTER
021400  005723                          4$:     TST     (R3)+           ;"PERIOD-PERIOD"
021402  005723                          5$:     TST     (R3)+           ;"COMMA"
021404  105711                                  TSTB    (R1)            ;"CR"?
021406  001001                                  BNE     6$              ;NO--BRANCH
021410  060300                                  ADD     R3,R0           ;YES--SAVE THE OCTAL NUMBER
021412  012603                          6$:     MOV     (SP)+,R3        ;RESTORE R3
021414  011000                                  MOV     (R0),R0         ;PICKUP EXIT ADDRESS
021416  000200                                  RTS     R0              ;RETURN


                                        .SBTTL  ASCIZ MESSAGES

021420  000122                          MSG.R:  .ASCIZ  R
021422  041506  000                     MSG.FC: .ASCIZ  FC
021425  [illegible]  000103             MSG.LC: .ASCIZ  L[illegible]
```

```
021430  041511     000          MSG.IC: .ASCIZ  IC
021433     124  000113          MSG.TK: .ASCIZ  TK
021436  000075                  MSG.EQ: .ASCIZ  =
021440  005015  050122  030461  MRPCS:  .ASCIZ  <15><12> RP11=/
021446  000075
021450  005015  050122  042526  MRPVEC: .ASCIZ  <15><12> RPVEC=
021456  036503     000
021461     015  051012  050120  MRPPRI: .ASCIZ  <15><12>/RPPRIO=
021466  044522  036517     000

021473     040  020040  020040  GRPH1:  .ASCII          0   10  20  30  40  50  60  70  80  90  100/ 15/<12/
021500  020040  020060  020040
021506  030061  020040  030062
021514  020040  030063  020040
021522  030064  020040  030065
021530  020040  030066  020040
021536  030067  020040  030070
021544  020040  030071  020040
021552  030061  006460     012
021557     040  020040  020040          .ASCIZ          I---I---I---I---I---I---I---I---I---I---I---I/<15><12>
021564  020040  026511  026455
021572  026511  026455  026511
021600  026455  026511  026455
021606  026511  026455  026511
021614  026455  026511  026455
021622  026511  026455  026511
021630  026455  026511  026455
021636  006511  000012
                                        .EVEN


                                .SBTTL  ERROR HEADER (EM) MESSAGES

021642  050122  030461  043040  EM1:    .ASCIZ  @RP11 FAILED TO RESPOND TO ADDRESSING@
021650  044501  042514  020104
021656  047524  051040  051505
021664  047520  042116  052040
021672  020117  042101  051104
021700  051505  044523  043516
021706     000
021707     104  044522  042526  EM2:    .ASCIZ  @DRIVE UNSAFE@
021714  052440  051516  043101
021722  000105
021724  051104  053111  020105  EM3:    .ASCIZ  @DRIVE OFFLINE@
021732  043117  046106  047111
021740  000105
021742  051104  053111  020105  EM4:    .ASCIZ  @DRIVE ERROR DURING TIMING TEST@
021750  051105  047522  020122
021756  052504  044522  043516
021764  052040  046511  047111
021772  020107  042524  052123
022000     000
022001     116  020117  047111  EM5:    .ASCIZ  @NO INTERRUPT FROM POSITIONING AFTER 1 SECOND@
022006  042524  051122  050125
022014  020124  051106  046517
022022  050040  051517  052111
```

```
022030  047511  044516  043516
022036  040440  052106  051105
022044  030440  051440  041505
022052  047117  000104
022056  044504  045523  042440  EM6:    .ASCIZ  @DISK ERROR AFTER POSITIONING@
022064  051122  051117  040440
022072  052106  051105  050040
022100  051517  052111  047511
022106  044516  043516     000
022113     103  051117  042522  EM7:    .ASCIZ  @CORRECT ATTN BIT NOT SET AFTER POSITIONING@
022120  052103  040440  052124
022126  020116  044502  020124
022134  047516  020124  042523
022142  020124  043101  042524
022150  020122  047520  044523
022156  044524  047117  047111
022164  000107
022166  051104  053111  020105  EM10:   .ASCIZ  @DRIVE OFFLINE AFTER POSITIONING@
022174  043117  046106  047111
022202  020105  043101  042524
022210  020122  047520  044523
022216  044524  047117  047111
022224  000107
022226  051447  041525  023501  EM11:   .ASCIZ  @'SUCA' NOT CORRECT AFTER POSITIONING@
022234  047040  052117  041440
022242  051117  042522  052103
022250  040440  052106  051105
022256  050040  051517  052111
022264  047511  044516  043516
022272     000
022273     104  051511  020113  EM12:   .ASCIZ  @DISK ERROR WHILE VERIFYING POSITION@
022300  051105  047522  020122
022306  044127  046111  020105
022314  042526  044522  054506
022322  047111  020107  047520
022330  044523  044524  047117
022336     000
022337     104  051511  020113  EM13:   .ASCIZ  @DISK POSITIONED TO WRONG CYLINDER@
022344  047520  044523  044524
022352  047117  042105  052040
022360  020117  051127  047117
022366  020107  054503  044514
022374  042116  051105     000
022401     116  020117  047111  EM14:   .ASCIZ  @NO INTERRUPT FROM I/O AFTER 1 SECOND@
022406  042524  051122  050125
022414  020124  051106  046517
022422  044440  047457  040440
022430  052106  051105  030440
022436  051440  041505  047117
022444  000104
022446  042523  045505  044440  EM15:   .ASCIZ  @SEEK INCOMPLETE@
022454  041516  046517  046120
022462  052105  000105
022466  051104  053111  020105  EM16:   .ASCIZ  @DRIVE NOT READY AFTER POSITIONING@
022474  047516  020124  042522
022502  042101  020131  043101
```

```
022510  042524  020122  047520
022516  044523  044524  047117
022524  047111  000107

022530  051104  053111  020105  EM17:   .ASCIZ  @DRIVE NOT READY/OFFLINE DURING TIMING TEST@
022536  047516  020124  042522
022544  042101  027531  043117
022552  046106  047111  020105
022560  052504  044522  043516
022566  052040  046511  047111
022574  020107  042524  052123
022602     000

                                .SBTTL  DATA HEADER (DH) MESSAGES

022603     105  051122  050040  DH1:    .ASCIZ  @ERR PC  RPCS@
022610  020103  051040  041520
022616  000123
022620  042524  052123  021440  DH2:    .ASCIZ  @TEST #  ERR PC  DRIVE   RPDS    RPER    RPCS@
022626  020040  051105  020122
022634  041520  020040  051104
022642  053111  020105  020040
022650  050122  051504  020040
022656  020040  050122  051105
022664  020040  020040  050122
022672  051503     000
022675     124  051505  020124  DH11:   .ASCIZ  @TEST #  ERR PC  DRIVE FROM CYL DEST CYL SUCA    RPCA@
022702  020043  042440  051122
022710  050040  020103  042040
022716  044522  042526  043040
022724  047522  020115  054503
022732  020114  042504  052123
022740  041440  046131  051440
022746  041525  020101  020040
022754  051040  041520  000101
022762  042524  052123  021440  DH12:   .ASCIZ  @TEST #  ERR PC  DRIVE   RPDS    RPER    RPCS    RPCA    RPDA@
022770  020040  051105  020122
022776  041520  020040  051104
023004  053111  020105  020040
023012  050122  051504  020040
023020  020040  050122  051105
023026  020040  020040  050122
023034  051503  020040  020040
023042  050122  040503  020040
023050  020040  050122  040504
023056     000
023057     124  051505  020124  DH13:   .ASCIZ  @TEST #  ERR PC  DRIVE FROM CYL DEST CYL HDR CYL@
023064  020043  042440  051122
023072  050040  020103  042040
023100  044522  042526  043040
023106  047522  020115  054503
023114  020114  042504  052123
023122  041440  046131  044040
023130  051104  041440  046131
023136     000
023137     122  041520  020124  DH13A:  .ASCIZ  @PPCS    PPCA    RPDA    SUCA@
```

```
023144  020040  051040  041520
023152  020101  020040  051040
023160  042120  020101  020040
023166  051440  041525  000101
023174  042524  052123  021440  DHSP:    .ASCIZ  @TEST #  ERR PC  DRIVE   RPDS    RPER    RPCS    RPWC@
023202  020040  051105  020122
023210  041520  020040  051104
023216  053111  020105  020040
023224  050122  051504  020040
023232  020040  050122  051105
023240  020040  020040  050122
023246  051503  020040  020040
023254  050122  041527     000
023261     122  041120  020101  DHSPA:   .ASCIZ  @RPBA    RPCA    RPDA    RPMI    SUCA    SILO@
023266  020040  051040  041520
023274  020101  020040  051040
023302  042120  020101  020040
023310  051040  046520  020061
023316  020040  051440  041525
023324  020101  020040  051440
023332  046111  000117

                                .EVEN

                                .SBTTL  DATA TABLE (DT)

023336  001116  001266          DT1:     .WORD   $ERRPC,RPADR
023342  001204  001116  001246  DT2:     .WORD   $TMPO,$ERRPC,CHKDRV,$RPDS,$RPER,$RPCS
023350  002004  002006  002010
023356  001204  001116  001246  DT11:    .WORD   $TMPO,$ERRPC,CHKDRV,CYL.CR,CYL.DS,$SUCA,$RPCA
023364  001254  001262  002024
023372  002016
023374  001204  001116  001246  DT12:    .WORD   $TMPO,$ERRPC,CHKDRV,$RPDS,$RPER,$RPCS,$RPCA,$RPDA
023402  002004  002006  002010
023410  002016  002020
023414  001204  001116  001246  DT13:    .WORD   $TMPO,$ERRPC,CHKDRV,CYL.CR,CYL.DS,$BDDAT
023422  001254  001262  001126
023430  002010  002016  002020  DT13A:   .WORD   $RPCS,$RPCA,$RPDA,$SUCA
023436  002024
023440  001204  001116  001246  DTSP:    .WORD   $TMPO,$ERRPC,CHKDRV,$RPDS,$RPER,$RPCS,$RPWC
023446  002004  002006  002010
023454  002012
023456  002014  002016  002020  DTSPA:   .WORD   $RPBA,$RPCA,$RPDA,$RPMI,$SUCA,$SILO
023464  002022  002024  002026

                                .SBTTL  DATA FORMAT (DF) TABLE

023472  000001                  DF1:     .WORD   1                  ;NUMBER OF DATA HEADERS
023474     002                           .BYTE   2                  ;NUMBER OF WORDS IN DATA TABLE
023475     000                           .BYTE   0                  ;BOTH NUMBERS ARE OCTAL

023476  000001                  DF2:     .WORD   1
023500     006                           .BYTE   6
023501     000                           .BYTE   0
```

```
023502  000001          DF11:   .WORD   1
023504     007                  .BYTE   7
023505     000                  .BYTE   0

023506  000001          DF12:   .WORD   1
023510     010                  .BYTE   8.
023511     000                  .BYTE   0

023512  000002          DF13:   .WORD   2               ;2 DH'S TO BE TYPED
023514     006                  .BYTE   6
023515     000                  .BYTE   0
023516  023137                  .WORD   DH13A
023520     004                  .BYTE   4
023521     000                  .BYTE   0

023522  000002          DFSP:   .WORD   2
023524     007                  .BYTE   7
023525     000                  .BYTE   0
023526  023261                  .WORD   DHSPA
023530     006                  .BYTE   6
023531     000                  .BYTE   0

                        .EVEN

023532  023174          ETSP:   .WORD   DHSP            ;MESSAGE BLOCK FOR SWITCH & TYPEOUTS
023534  023440                  .WORD   DTSP
023536  023522                  .WORD   DFSP

023540  000030          BUFR4:  .BLKW   24.             ;WORKING LOCATIONS FOR SEEK TIMER TEST
023620  000030          BUFR5:  .BLKW   24.
023700  000030          BUFR6:  .BLKW   24.
023760  000030          BUFR7:  .BLKW   24.
024040  000626          FRWSTR: .BLKW   406.
025514  000626          RVSTR:  .BLKW   406.
        027170          BUFFER=.

        00000.          .END
```

```
AIE     = 020000
ATABIT    001314
BITS      001324
BIT0    = 000001
BIT00   = 000001
BIT01   = 000002
BIT02   = 000004
BIT03   = 000010
BIT04   = 000020
BIT05   = 000040
BIT06   = 000100
BIT07   = 000200
BIT08   = 000400
BIT09   = 001000
BIT1    = 000002
BIT10   = 002000
BIT11   = 004000
BIT12   = 010000
BIT13   = 020000
BIT14   = 040000
BIT15   = 100000
BIT2    = 000004
BIT3    = 000010
BIT4    = 000020
BIT5    = 000040
BIT6    = 000100
BIT7    = 000200
BIT8    = 000400
BIT9    = 001000
BLNKS1    002044
BLNKS2    002043
BLNKS3    002042
BLNKS4    002041
BLNKS5    002040
BLNKS6    002037
BLNKS7    002036
BLNKS8    002035
BLNKS9    002034
BLNK10    002033
BLNK13    002030
BPTVEC=   000014
BUFFER=   027170
BUFR4     023540
BUFR5     023620
BUFR6     023700
BUFR7     023760
BUSADR    001224
BYPASS    001244
CHKDRV    001246
CKSWR   = 104407
CK.CHR    021044
CK.DEC    021016
CK.DIG    021120
CK.NUM    021300
CK.OCT    020770
CLEAR   = 000001
CLOSE     020710
CMPGRP    007332
CNTRLC    001222
CR      = 000015
CRLF    = 000200
CSME    = 000040
CYL.CR    001254
CYL.DS    001262
DDISP   = 177570
DECSK     004462
DFLT      001460
DFSP      023522
DF1       023472
DF11      023502
DF12      023506
DF13      023512
DF2       023476
DHSP      023174
DHSPA     023261
DH1       022603
DH11      022675
DH12      022762
DH13      023057
DH13A     023137
DH2       022620
DISPLA    001142
DISPRE    000174
DPB       001364
DRVBAD    001252
DRVMSK    001250
DRVOK     003750
DRVSEL    001232
DRVSTA    001304
DRVTYP    001240
DSELCT    015272
DSKERR=   000001
DSWR    = 177570
DTSP      023440
DTSPA     023456
DT1       023336
DT11      023356
DT12      023374
DT13      023414
DT13A     023430
DT2       023342
EMTVEC=   000030
EM1       021642
EM10      022166
EM11      022226
EM12      022273
EM13      022337
EM14      022401
EM15      022446
EM16      022466
EM17      022530
EM2       021707
EM3       021724
EM4       021742
EM5       022001
EM6       022056
EM7       022113
EOP     = 000002
ERR     = 100000
ERRVEC=   000004
ETSP      023532
EXIT0     004154
EXIT1     004312
EXIT10    006304
EXIT11    006616
EXIT12    007472
EXIT2     004504
EXIT3     004654
EXIT4     005216
EXIT5     005404
EXIT6     005636
EXIT7     006030
FC        001376
FC1       001644
FC11      001774
FC2       001660
FC3       001674
FC4       001710
FC5       001724
FC6       001740
FC7       001754
FMTE    = 001000
FRWSTR    024040
FUV     = 040000
GETADR    017256
GETNUM    017460
GOTYPE    016130
GRPH1     021473
GTSWR   = 104406
GTTST1    017742
GTTST2    020014
GTTST3    020162
GTTST4    020164
GTTST5    020236
GTTST6    020266
GT.PRM    017610
GT.PR1    017612
GT.PR2    017736
HDR     = 004000
HE      = 040000
HNF     = 010000
HOMSEK=   000015
HT      = 000011
IC        001402
IC1       001650
IC2       001664
IC3       001700
IC4       001714
IC5       001730
IC6       001744
IC7       001760
IDE     = 000100
INADR     001226
INCSK     004432
IOTVEC=   000020
LC        001400
LC1       001646
LC11      001776
LC2       001662
LC3       001676
LC4       001712
LC5       001726
LC6       001742
LC7       001756
LF      = 000012
LPE     = 000200
MAXCYL    001236
MODE    = 010000
MODERR=   000400
MRPCS     021440
MRPPRI    021461
MRPVEC    021450
MSG.EQ    021436
MSG.FC    021422
MSG.IC    021430
MSG.LC    021425
MSG.R     021420
MSG.TK    021433
MXSTAL    001302
NXC     = 020000
NXME    = 000004
NXS     = 004000
NXT     = 010000
OPNPRM    020442
OPNTST    020310
OPN.CT    020314
OPN.N1    020630
OPN.N2    020634
OPN.X1    020646
OPN.X2    020652
OPN 1     020320
OPN.2     020342
OUTADR    001230
PIRQ    = 177772
PIRQVE=   000240
PLCT      006770
PLTPT     007432
PRM       001372
PRMLMT    001434
PRMMSG    001446
PRMPT     001406
PRM0      001632
PRM1      001640
PRM10     001764
PRM11     001770
PRM12     002000
PRM2      001654
PRM3      001670
PRM4      001704
PRM5      001720
PRM6      001734
PRM7      001750
PROG    = 002000
PR0     = 000000
PR1     = 000040
PR2     = 000100
PR3     = 000140
PR4     = 000200
PR5     = 000240
PR6     = 000300
PR7     = 000340
PS      = 177776
PSW     = 177776
PWRVEC=   000024
RDCHR   = 104410
RDLIN   = 104411
RDSEK   = 000005
RDY     = 000200
READ    = 000017
RESREG=   104413
RESTAR    003726
RESTOR    015314
RESTR1    015530
RESVEC=   000010
RPADR     001266
RPBA    = 000010
RPCA    = 000012
RPCS    = 000004
RPDA    = 000014
RPDS    = 000000
RPER    = 000002
RPINIT    013642
RPM1    = 000016
RPM2    = 000020
RPM3    = 000022
```

```
RPPRIO  001272
RPT     001374
RPT0    001634
RPT1    001642
RPT10   001766
RPT11   001772
RPT12   002002
RPT2    001656
RPT3    001672
RPT4    001706
RPT5    001722
RPT6    001736
RPT7    001752
RPVEC   001270
RPWC  = 000006
RSTRT1  003704
RSTRT2  003716
RSTRT3  003724
RVSTR   025514
R6    =%000006
R7    =%000007
SAVREG= 104412
SAVRP   015164
SEC.RD  001260
SEEK  = 000011
SEEKS   014036
SEEKS0  014064
SEEKS1  014314
SEEKS2  014400
SEEKS3  014462
SEEKS4  014520
SILO  = 000026
SKDON   017172
SMGRP   007162
SORT    015660
SPSAV   001242
SRTINT  002734
STACK = 001100
STALL   015576
STALLA  015572
STALLB  015624
STALLG  001300
STALLO  001274
STALL1  001276
START   002302
START1  002246
START2  002270
START3  002236
START4  002260
START5  002720
START6  003442
STKLMT= 177774
STRT1A  002252
STRT2A  002274
STR2OX  001264
SUCA  = 000024
SUFU  = 001000
SUOL  = 040000
SURDY = 100000
SURPO3= 020000
SUSI  = 004000
SUSU  = 002000
SUWP  = 000400
SWR     001140
SWREG   000176
SW0   = 000001
SW00  = 000001
SW01  = 000002
SW02  = 000004
SW03  = 000010
SW04  = 000020
SW05  = 000040
SW06  = 000100
SW07  = 000200
SW08  = 000400
SW09  = 001000
SW1   = 000002
SW10  = 002000
SW11  = 004000
SW12  = 010000
SW13  = 020000
SW14  = 040000
SW15  = 100000
SW2   = 000004
SW3   = 000010
SW4   = 000020
SW5   = 000040
SW6   = 000100
SW7   = 000200
SW8   = 000400
SW9   = 001000
TBITVE= 000014
TEST0   004144
TEST1   004262
TEST10  006130
TEST11  006404
TEST12  006716
TEST2   004420
TEST3   004612
TEST4   004776
TEST5   005324
TEST6   005512
TEST7   005744
TIMEE = 000020
TIMSEK  016526
TIMSK1  016550
TIMSK2  017066
TK      001404
TKVEC = 000060
TK0     001636
TK1     001652
TK2     001666
TK3     001702
TK4     001716
TK5     001732
TK6     001746
TK7     001762
TPVEC = 000064
TRAPVE= 000034
TRK.RD  001256
TRTVEC= 000014
TSTNMS  001234
TST0    004034
TST1    004156
TST10   006032
TST11   006306
TST12   006620
TST2    004314
TST3    004506
TST4    004656
TST5    005220
TST6    005406
TST7    005640
TYPDS = 104405
TYPE  = 104401
TYPERR  010114
TYPOC = 104402
TYPON = 104404
TYPOS = 104403
TYPTIM  016364
VERFYO  014600
VERFYI  015154
VERIFY  014554
WCE   = 000010
WCKSEK= 000007
WPE   = 000100
WPV   = 100000
WRITE = 000013
WRTSEK= 000003
$AUTOB  001134
$BDADR  001122
$BDDAT  001126
$BELL   001212
$CHARC  010566
$CKSWR  011542
$CMTAG  001100
$CM1  = 000011
$CM2  = 000022
$CM3  = 000011
$CM4  = 000001
$CNTLC  012463
$CNTLG  012475
$CNTLU  012470
$CRLF   001217
$DBLK   011234
$DB2D   013304
$DECVL  013464
$DOAGN  007732
$DTBL   011224
$ENDAD  007722
$ENDCT  007656
$ENULL  007736
$EOP    007474
$EOPCT  007650
$ERFLG  001103
$ERMAX  001115
$ERROR  007742
$ERRPC  001116
$ERRTB  002046
$ERTTL  001112
$ESCOP  001210
$FILLC  001156
$FILLS  001155
$GDADR  001120
$GDDAT  001124
$GET42  007712
$GTSWR  011632
$HD   = 000000
$HINUM  013636
$ICNT   001104
$INTAG  001135
$ITEMB  001114
$LF     001220
$LONUM  013640
$LPADR  001106
$LPERR  001110
$MNEW   012513
$MSWR   012502
$MUL  = 000003
$MULT   013052
$MXCNT  012754
$NULL   001154
$NWTST= 000001
$OCNT   011014
$OMODE  011016
$OVER   012740
$PASS   001100
$QUES   001216
$RAND   013540
$RDCHR  012104
$RDLIN  012174
$RDSZ = 000033
$REGAD  001160
$REG0   001162
$REG1   001164
$REG10  001202
$REG2   001166
$REG3   001170
$REG4   001172
$REG5   001174
$REG6   001176
$REG7   001200
$RESRE  013014
$RPBA   002014
$RPCA   002016
$RPCS   002010
$RPDA   002020
$RPDS   002004
$RPER   002006
$RPM1   002022
$RPWC   002012
$RTNAD  007734
$SAVRE  012756
$SB2D   013250
$SCOPE  012524
$SETUP= 000107
$SILO   002026
$STUP = 177777
$SUCA   002024
$SUPRS  013500
$SVLAD  012712
$SVPC = 000200
$SWR  = 167000
$SLRMK= 000000
$TIMES  001206
$TKB    001146
$TKCNT  011244
$TKINT  011254
$TKQEN= 011253
$TKQIN  011246
$TKQOU  011250
$TKQSR  011252
$TKS    001144
$TKSRV  011324
$TMPO   001204
$TN   = 000013
$TNPWR  013414
$TPB    001152
$TPFLG  001157
$TPS    001150
$TRAP   013164
$TRAP2  013206
$TRP  = 000014
$TRPAD  013220
$TSTNM  001102
```

```
$TTYIN  012430      $TYPEC  010522      $TYPON  010632      $$GET4= 000000
$TYPDS  011020      $TYPEX  010570      $TYPOS  010572      $OFILL  011015
$TYPE   010352      $TYPOC  010616      $XTSTR  012536      .       = 027170

. ABS.  027170  000

 ERRORS DETECTED:  0

 DSKW:DZRPZC,DSKW:DZRPZC/SOL=DSKW:SYSMAC.SML,DSKM:DZRPZC.P11
 RUN-TIME: 14 19 .6 SECONDS
 RUN-TIME RATIO: 144/34=4.1
 CORE USED:  35K  (69 PAGES)
```

L08