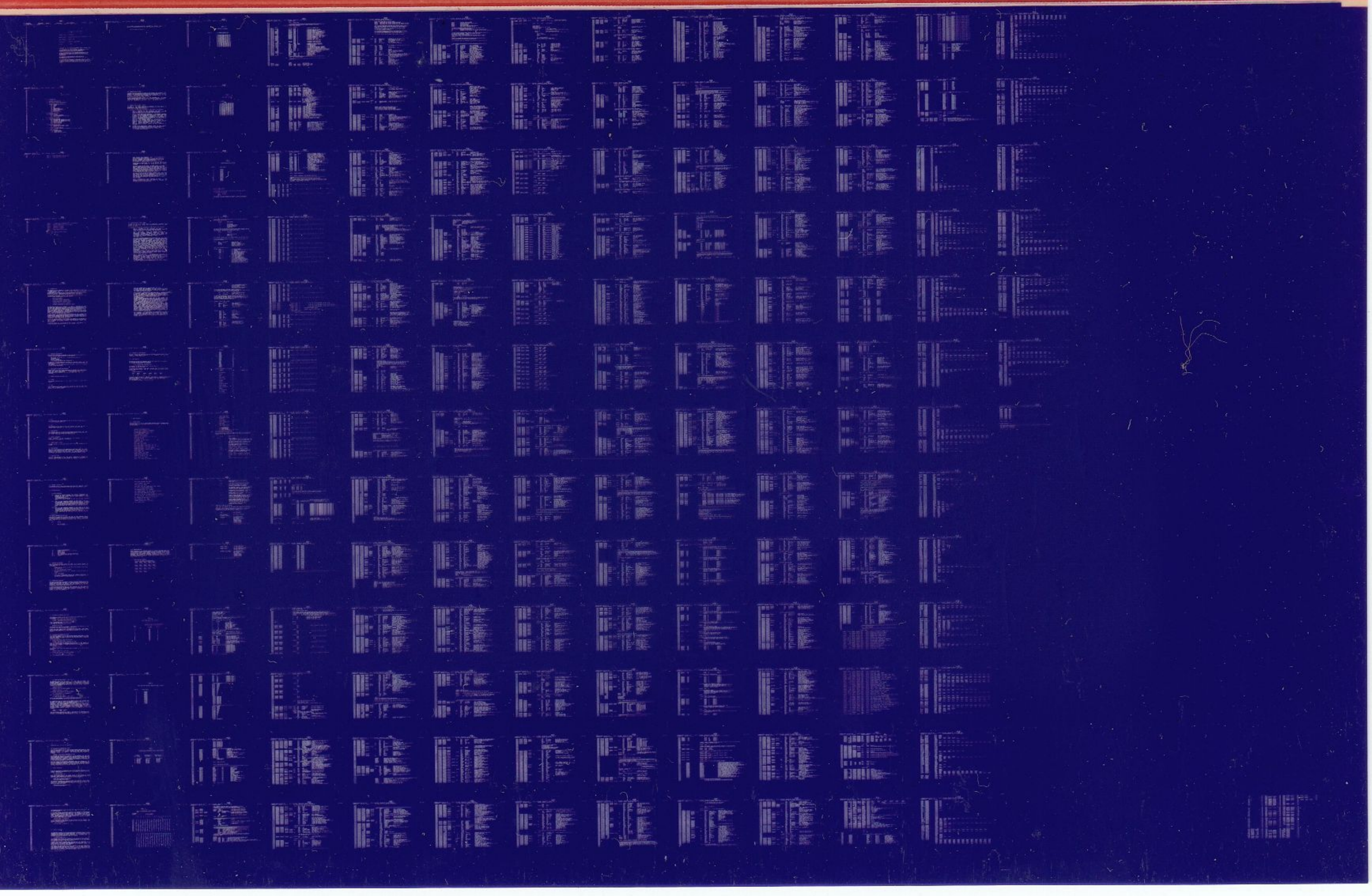
RM03,02

DRIVE COMPATIBILITY TEST
CZRMIB0

AH-B016B-MC

FICHE 1 OF 1

JAN 1978
digital
MADE IN USA



.REM 2

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

PRODUCT CODE: AC-B015B-MC

PRODUCT NAME: CZRMIBO RM03/RM02 DRIVE COMPATIBILITY TEST

DATE CREATED: AUGUST 1977

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR. C. CHEN

TABLE OF CONTENTS

```
                    TABLE A -  BASIC READ/WRITE TEST SECTORS

                    TABLE B -  WORSE CASE DATA PATTERN
```

1.0 INTRODUCTION

THE PURPOSE OF THIS PROGRAM IS TO VERIFY THE COMPATIBILITY OF UP TO 16 RM03/RM02 DRIVES WHICH MAY RESIDE ON ONE / MORE RH11(70)/RM03,RM02 SUBSYSTEMS. COMPATIBILITY IS DEFINED HERE AS THE ABILITY OF A DRIVE TO WRITE DATA WHICH CAN BE READ SUCCESSFULLY BY ALL OTHER DRIVES, AND ADDITIONALLY THE ABILITY OF A DRIVE TO COMPLETELY OVER-WRITE DATA WRITTEN BY ALL OTHER DRIVES.

THE PROGRAM IS DESIGNED TO DETECT THE FOLLOWING CONDITIONS WHICH MOST COMMONLY CAUSE INCOMPATIBILITY BETWEEN DRIVES:

1. HEAD MIS-ALIGNMENT
2. POSITIONER LATERAL MISALIGNMENT
3. SPINDLE-CARTRIDGE INTERFACE RUNOUT
4. IMPROPER LEVELS OF WRITE CURRENT
5. INCORRECT ADDRESSING OF READ/WRITE HEADS

THE TESTING IS DONE IN TWO PASSES. IN PASS 1, COMPATIBILITY DATA PATTERNS ARE WRITTEN BY ALL THE DRIVES UPON THE SAME DISK CARTRIDGE, AND THE BASIC READ/WRITE CAPABILITY OF EACH DRIVE IS DEMONSTRATED. IN PASS 2, THE COMPATIBILITY DATA FROM ALL DRIVES IS READ BY EACH DRIVE, WITH HEAD OFFSET, AND THIS IS COMPARED WITH EACH DRIVE'S ABILITY TO READ ITS OWN DATA. IN ADDITION, EACH DRIVE'S CAPABILITY TO OVERWRITE DATA WRITTEN BY ALL OTHER DRIVES IS TESTED ON THE SECOND PASS. (FOR THE REMAINDER OF THIS SPECIFICATION, THE ABOVE DEFINITIONS OF THE FIRST AND SECOND PASS SHALL APPLY).

IN BOTH PASSES, THE PROGRAM DIRECTS THE OPERATOR IN THE LOADING AND UNLOADING OF DRIVES AND THE MOVEMENT OF THE CARTRIDGE FROM DRIVE TO DRIVE, THROUGH MESSAGES AT THE CONSOLE TERMINAL. AT THE COMPLETION OF TESTING ON EACH DRIVE DURING THE SECOND PASS A SUMMARY IS PRINTED OF COMPATIBILITY TEST RESULTS FOR THAT DRIVE.

WITHIN THE VARIOUS TESTS OF BOTH PASSES, THE CAPABILITY IS PROVIDED TO LOOP ON CURRENT OPERATIONS, AND SWITCH REGISTER OPTIONS ARE PROVIDED, FOR A VARIETY OF LOOPING, RUNNING, AND REPORTING MODES (SEE SECTION 6.2).

UNEXPECTED ERRORS WILL BE REPORTED AS THEY OCCUR. THE REPORT WILL INCLUDE DESCRIPTION AND APPLICABLE DEVICE REGISTER CONTENTS.

2.0 HARDWARE REQUIREMENTS

THE FOLLOWING HARDWARE IS REQUIRED TO RUN THE RM03 DRIVE COMPATIBILITY PROGRAM, FOR EACH SUBSYSTEM WHICH MAY BE PRESENT:

- PDP-11/04, (05,10 MFG. ONLY), 20,30,34,35,40,45,50,70
- 16K MEMORY
- CONSOLE TERMINAL
- RH11(70) CONTROLLER
- 1 TO 8 RM03 DISK DRIVES PER CONTROLLER

IN ADDITION, A SINGLE RM03 DISK CARTRIDGE IS REQUIRED WHICH MUST BE FORMATTED IN 32 SECTOR FORMAT, ON A RELIABLE WELL-ALIGNED(REFERENCE PACK) RM03 DRIVE.
THIS CARTRIDGE WILL BE MOVED FROM DRIVE TO DRIVE, (ON UP TO 16 DRIVES) ON EACH OF THE TWO PASSES.

3.0 PRELIMINARY PROGRAM REQUIREMENTS

BEFORE RUNNING THE R. _3 DRIVE COMPATIBILITY PROGRAM, THE SUBSYSTEM(S) UNDER TEST SHOULD BE CAPABLE OF PASSING THE CONTROLLER DIAGNOSTICS AND THE DRIVE DIAGNOSTICS . IN ADDITION, THE CARTRIDGE MUST BE FORMATTED IN 32 SECTOR FORMAT USING THE PACK FORMATTER.

4.0 GENERAL PROGRAM CONSIDERATIONS

4.1 SYSMAC

THIS PROGRAM USES PORTIONS OF THE SYSMAC DIAGNOSTIC SYSTEM MACRO PACKAGE.

4.2 XXDP

THIS PROGRAM MAY BE LOADED UNDER XXDP, AND MAY BE RUN IN DUMP MODE ONLY. DUE TO MANUAL INTERVENTION AND LACK OF END-OF-PASS HOOKS, THE PROGRAM IS NOT XXDP CHAINABLE.

4.3 ACT

THIS PROGRAM MAY BE LOADED UNDER ACT AND MAY BE RUN IN DUMP MODE ONLY. IT IS NOT CHAINABLE UNDER ACT.

4.4 APT

THIS PROGRAM MAY BE LOADED BY THE APT SYSTEM, BUT MAY BE RUN IN PROGRAM (DUMP) MODE ONLY. IT CANNOT BE RUN IN APT SCRIPT MODE.

4.5 DUAL-ACCESS

THIS PROGRAM DOES NOT UTILIZE THE DUAL-ACCESS OPTION IN ANY WAY, AND ALL DRIVES UNDER TEST SHOULD BE DE-SELECTED THROUGH THE PORT WHICH IS NOT IN USE , OR LOCKED ON THE PORT BEING TESTED.

4.6 MEMORY MANAGEMENT

MEMORY MANAGEMENT IS NOT UTILIZED IN THIS PROGRAM. IF IT IS INSTALLED, IT IS DISABLED BY THE PROGRAM.

4.7 MEMORY PARITY OPTION

IF PARITY MEMORY IS INSTALLED, MEMORY PARITY TRAPS ARE DISABLED BY THE PROGRAM.

4.8 BAD SECTORS

THE LIST OF BAD SECTORS ON THE CARTRIDGE IS OBTAINED FROM THE FIRST DRIVE TO BE TESTED ON THE CURRENT SUBSYSTEM. ACCORDING TO A SWITCH REGISTER OPTION (SEE SECTION 6.2) THIS LIST MAY BE TYPED AT THE CONSOLE AT THE START OF THE FIRST PASS. ALL DATA ERRORS OCCURING IN THE PROGRAM ARE IGNORED IF THEY OCCUR IN SECTORS DESIGNATED AS BAD.

4.9 EXECUTION TIME

THE TOTAL TIME REQUIRED TO RUN THE DRIVE COMPATIBILITY PROGRAM IS DIRECTLY PROPORTIONAL TO THE NUMBER OF DRIVES TO BE TESTED AND REQUIRES ABOUT 2 MINUTES PER DRIVE,EXCLUDING OPERATOR INTERVENTION.

5.0 PROGRAM LOAD MEDIA

THIS PROGRAM CAN BE LOADED FROM PAPER TAPE USING THE ABSOLUTE LOADER OR FROM THE ACT OR APT SYSTEMS OR FROM ANY MEDIA SUPPORTED BY XXDP.

6.0 PROGRAM OPTIONS

6.1 STARTING ADDRESSES

200 - THIS IS THE STARTING ADDRESS FOR DEFAULT PARAMETERS AND RUNNING OF PASS 1 AND PASS 2 ON A SINGLE SUBSYSTEM. THE PROGRAM WILL USE DEFAULT RH11(70) BASE ADDRESS, INTERRUPT VECTOR AND PRIORITY.
THE PROGRAM WILL ASSUME ALL DRIVES TO BE TESTED RESIDE ON ONE RH11(70)/RM03 SUBSYSTEM ONLY.

204 - THIS IS THE STARTING ADDRESS TO RUN PASS 1 ON ALL RH11(70)/RM03 SUBSYSTEMS WHICH RESIDE ON THIS PDP-11 SYSTEM. THE PROGRAM WILL ASK FOR THE RH11(70) BASE ADDRESS, INTERRUPT VECTOR, AND PRIORITY FOR EACH SUBSYSTEM ON THIS SYSTEM, AND IT ASKS FOR THE LETTER NAMES (A THRU H) ASSIGNED TO ALL OTHER SUBSYSTEMS, AND THE DRIVE(S) WHICH WILL BE TESTED ON EACH.

220 - THIS IS THE STARTING ADDRESS TO RUN PASS 2 ON ALL RH11(70)/RM03 SUBSYSTEMS WHICH RESIDE ON THIS PDP-11 SYSTEM. THE PROGRAM WILL ASK FOR THE RH11(70) BASE ADDRESS,INTERRUPT VECTOR FOR EACH SUBSYSTEM ON THIS SYSTEM, AND IT ASKS FOR THE LETTER NAMES (A THRU H) ASSIGNED TO ALL OTHER SUBSYSTEMS, AND THE DRIVE(S) WHICH WILL BE TESTED ON EACH.

6.2 SWITCH REGISTER OPTIONS USED

THIS PROGRAM IS DESIGNED TO ALLOW THE USE OF THE HARDWARE SWITCH REGISTER IF PRESENT, OR THE SYSMAC-SUPPORTED SOFTWARE SWITCH REGISTER (IF HARDWARE SWR IS NOT PRESENT, OR IS SET TO ALL ONES). IN EITHER CASE, THE FOLLOWING OPTIONS ARE IMPLEMENTED WHEN THE APPROPRIATE BITS ARE SET TO 1.

BIT	OPTION
15	HALT ON ERROR
14	LOOP ON CURRENT TEST

13	INHIBIT ERROR REPORTS
12	REPORT DESCRIPTION ONLY, ON ERRORS
11	UNUSED
10	BELL ON ERROR
09	LOOP ON ERROR
08	APPLY RANDOM STALL BETWEEN OPERATIONS
07	TYPE BAD SECTOR FILES (BSF'S) AT START
06-00	UNUSED

7.0 RUNNING THE PROGRAM

ONCE THE PROGRAM HAS BEEN LOADED INTO CORE (IN A GIVEN SYSTEM, IF THERE ARE MULTIPLE SYSTEMS) THE FOLLOWING STEPS MUST BE TAKEN TO RUN THE PROGRAM:

1. INSURE THAT ALL DRIVES TO BE TESTED ARE POWERED UP AND SINGLE PORT SELECTED.
2. LOAD THE DESIRED START ADDRESS.
3. SET ANY DESIRED BITS IN THE HARDWARE SWITCH REGISTER (IF PRESENT).
4. START THE PROGRAM.
5. FOLLOW ALL INSTRUCTIONS TYPED BY THE PROGRAM PERTAINING TO THE MANUAL INTERVENTION REQUIRED, AND THE ALTERNATE USE OF MULTIPLE SYSTEMS (IF THERE ARE ANY).

8.0 OPERATIONAL DIALOGUE

THIS SECTION DESCRIBES THE CONSOLE TERMINAL DIALOGUE THROUGH WHICH THE PROGRAM DIRECTS THE OPERATOR, IN THE SELECTION OF OPTIONS AND THE LOADING AND UNLOADING OF DRIVES, AND THE MOVEMENT OF THE TEST CARTRIDGE. THE EXACT DIALOGUE WHICH IS USED DEPENDS UPON THE STARTING ADDRESS WHICH WAS CHOSEN (SEE SECTION 6.1).

IN THE FOLLOWING DISCUSSION AND IN THE PRINTOUT OF TEST RESULTS, DRIVES TO BE TESTED WILL BE REFERRED TO BY A LETTER AND A NUMBER. THE LETTER IS THE SUBSYSTEM LETTER NAME (OPERATOR ASSIGNED), AND THE NUMBER IS THE DRIVE NUMBER ON THAT SUBSYSTEM. FOR EXAMPLE, DRIVE C6 REFERS TO DRIVE 6 ON SUBSYSTEM C.

8.1 DIALOGUE FOR ADDRESS 200 START

THIS STARTING ADDRESS MAY BE USED FOR DEFAULTING PARAMETERS ON ONE SUB-SYSTEM.
THE PROGRAM FIRST IDENTIFIES ITSELF AS FOLLOWS:

CZRMIB0 - RM03/RM02 DRIVE COMPATIBILITY TEST

THEN, THE PROGRAM ASKS THE DRIVES TO BE TESTED.
THE PROGRAM TYPES THE DRIVE LIST, AS IN THE FOLLOWING EXAMPLE:

DRIVES = 2,5,7<CR>

THE PROGRAM NOW PROCEEDS WITH PASS 1, AND DIRECTS THE OPERATOR IN THE MOUNTING OF THE PACK, AS DESCRIBED IN SECTION 8.4.

PLEASE NOTE THAT THERE IS ONLY ONE SUBSYSTEM ON AN ADR. 200 START, AND IT IS NAMED SUBSYSTEM A. THE DRIVES IN THE ABOVE EXAMPLE WOULD BE REFERRED TO AS A2,A5,A7 IN THE TEST RESULTS PRINTOUT AT THE END OF PASS 2.

8.2 DIALOGUE FOR ADDRESS 204 START

THIS STARTING ADDRESS MUST BE USED ON EACH SYSTEM, WHEN THERE IS MORE THAN ONE SUBSYSTEM, BUT IT MAY ALSO BE USED WHEN THERE IS JUST ONE SUBSYSTEM (TOTAL), TO SPECIFY DRIVES TO TEST AND NON-DEFAULT PARAMETER VALUES, FOR PASS 1.

THE PROGRAM IDENTIFIES ITSELF AS FOLLOWS:

CZRMIB0- RM03/RM02 DRIVE COMPATIBILITY TEST

THEN, THE PROGRAM ASKS THE OPERATOR FOR THE DRIVES TO BE TESTED ON EACH OF THE POSSIBLE SUBSYSTEMS (STARTING WITH A - THE NAMES RANGE FROM SUBSYS A TO SUBSYS H. THERE COULD BE UP TO 8 SUBSYSTEMS, WITH A DRIVE ON EACH) :

SUBSYS A DRIVE(S) =

THE OPERATOR THEN TYPES THE DESIRED DRIVE NUMBERS, AS IN THE FOLLOWING EXAMPLE:

SUBSYS A DRIVE(S) = 2,5,7<CR>

THE PROGRAM THEN VERIFIES THE DRIVE NUMBERS BY TYPING :

WILL TEST DRIVE(S) 2,5,7 ON SUBSYS A.

NEXT, THE PROGRAM ASKS THE FOLLOWING QUESTION:

IS THERE ANOTHER SUBSYS (Y OR N <CR>)?

THE OPERATOR TYPES Y<CR> OR N<CR>. (IF JUST <CR> IS TYPED, THE PROGRAM ASSUMES THAT N<CR> WAS TYPED). IF THE OPERATOR TYPED N, THE PROGRAM PROCEEDS WITH PASS 1, AND DIRECTS THE OPERATOR IN THE MOUNTING OF THE PACK, AS DESCRIBED IN SECTION 8.4. IF Y WAS TYPED, THE PROGRAM ASKS FOR THE NUMBERS OF THE DRIVES TO BE TESTED ON THE NEXT SUBSYSTEM (SUBSYS B) AS FOLLOWS:

SUBSYS B DRIVE(S) =

THE OPERATOR TYPES THE DRIVE NUMBERS, AS IN THE FOLLOWING EXAMPLE:

SUBSYS B DRIVE(S) = 2,3<CR>

THE PROGRAM THEN VERIFIES THE DRIVE NUMBERS. BY TYPING :

WILL TEST DRIVE(S) 2,3 ON SUBSYS B.

NEXT, THE PRCGRAM WILL ASK :

IS THERE ANOTHER SUBSYS (Y OR N <CR>)?

AND IN THE SAME MANNER, THE OPERATOR SPECIFIES THE DRIVES ON EACH OF THE REMAINING SUBSYSTEMS, UNTIL ALL HAVE BEEN SPECIFIED.

ALL SUBSYSTEMS MUST BE TESTED IN THE ORDER IN WHICH THE LETTERS ARE ASSIGNED (A THRU H). NEXT, THE PROGRAM ALLOWS THE OPERATOR TO ALTER THE RH11(70) BUS ADDRESS, VECTOR ADDRESS FOR THIS SUBSYSTEM. FOR EACH PARAMETER THE CURRENT VALUE IS TYPED, AND THE OPERATOR IS GIVEN THE OPPORTUNITY TO TYPE IN A NEW VALUE, PLUS <CR>. IF JUST <CR> IS TYPED, THE PARAMETER IS NOT CHANGED. WHEN THE PROGRAM IS FIRST LOADED, THE FOLLOWING DEFAULT VALUES ARE ASSIGNED: RH11(70) BUS ADDRESS = 177670, VECTOR ADDRESS = 240, (IF 200 START). THE FOLLOWING EXAMPLE SHOWS A PRINTOUT IN WHICH BUS ADDRESS AND VECTOR WERE CHANGED:

```
RMCS1  =  000000   177670
RMVEC  =     000   254
```

THEN THE PROGRAM PROCEEDS WITH PASS 1, AND DIRECTS THE OPERATOR IN THE MOUNTING OF THE PACK, AS DESCRIBED IN SECTION 8.4. AT THE COMPLETION OF PASS 1 ON THE SUBSYSTEM, THE PROGRAM WILL INFORM THE OPERATOR HOW

TO PERFORM PASS 1 ON THE NEXT SUBSYSTEM.

8.3 DIALOGUE FOR ADDRESS 220 START

THIS STARTING ADDRESS MUST BE USED ON EACH SYSTEM, WHEN THERE IS MORE THAN 1 SUBSYSTEM, BUT IT MAY ALSO BE USED WHEN THERE IS JUST ONE SUBSYSTEM (TOTAL) , TO SPECIFY DRIVES TO TEST AND NON-DEFAULT PARAMETER VALUES, FOR PASS 2. THE PROGRAM IDENTIFIES ITSELF, AS FOLLOWS :

```
CZRMIB0 - RM03 DRIVE COMPATIBILITY TEST
```

THE DIALOGUE FOR 220 START IS IDENTICAL TO THE DIALOGUE FOR THE 204 START DESCRIBED ABOVE (SECTION 8.2), FOR THE SELECTION OF SUBSYSTEM PARAMETERS AND THE SPECIFICATION OF ALL DRIVES TO BE TESTED ON THE VARIOUS SUBSYSTEMS. HOWEVER, AFTER THIS DIALOGUE IS COMPLETED, THE PROGRAM PROCEEDS WITH PASS 2, AND DIRECTS THE OPERATOR IN THE MOVEMENT OF THE PACK, AS DESCRIBED IN SECTION 8.5.

NOTE THAT SINCE THE APPROPRIATE PROCESSOR MUST BE STARTED AT THE STARTING ADDRESS FOR EACH SUBSYSTEM TO BE TESTED, THE COMPATIBILITY TEST MAY BE PERFORMED IN STEPS, AT VARIOUS TIMES AND BETWEEN VARIOUS DISTANT LOCATIONS, BY MOVING THE TEST PACK AND SAVING THE PRINTOUT FROM EACH PASS ON EACH PDP-11 SYSTEM INVOLVED.

8.4 PASS 1 DIALOGUE

AFTER THE SELECTION OF PARAMETERS AND DRIVES HAS BEEN COMPLETED ON THE CURRENT SUBSYSTEM (SECTIONS 8.1-8.2), THE PROGRAM INDICATES THE START OF PASS 1 AS FOLLOWS:

```
** STARTING PASS 1 **
```

NEXT, THE PROGRAM SELECTS THE FIRST DRIVE TO BE TESTED ON THIS SUBSYSTEM, AND INSTRUCTS THE OPERATOR TO MOUNT THE TEST CARTRIDGE AND LOAD THE HEADS ON THAT DRIVE, AS IN THE FOLLOWING EXAMPLE:

```
MOUNT PACK ON DRIVE A2 AND LOAD.
TYPE R<CR> WHEN DRIVE READY:
```

THE OPERATOR PERFORMS THIS TASK AND TYPES R<CR> WHEN THE DRIVE READY LIGHT IS ON. THE PROGRAM PERFORMS PASS 1 FUNCTIONS ON THIS DRIVE (SEE SECTION 9.1) AND THEN INSTRUCTS THE OPERATOR TO UNLOAD THE DRIVE AND REMOVE THE PACK AS FOLLOWS:

```
UNLOAD DRIVE A2 AND REMOVE PACK.
TYPE R<CR> WHEN DONE:
```

THE OPERATOR PERFORMS THESE FUNCTIONS AND TYPES R<CR> AFTER THE PACK HAS BEEN REMOVED.

IN THE SAME MANNER, THE PROGRAM INSTRUCTS THE OPERATOR IN THE MOVEMENT OF THE PACK THROUGHOUT THE REST OF THE DRIVES ON THE CURRENT SUBSYSTEM. WHEN THIS HAS BEEN COMPLETED, THE PROGRAM DOES ONE OF THREE THINGS: (1) IF THERE IS ONLY ONE SUBSYSTEM (FROM ADR 200 START) THE PROGRAM BEGINS PASS 2 (SEE SECTION 8.5). (2) IF THERE IS ANOTHER SUBSYSTEM, THE PROGRAM DIRECTS THE OPERATOR TO PERFORM PASS 1 ON THE NEXT SUBSYS AS FOLLOWS :

STARTING PASS 1 ON SUBSYS B

(3) IF THERE ARE NO MORE DRIVES TO TEST IN PASS 1 ON ANY SUBSYS, THE PROGRAM DIRECTS THE OPERATOR TO BEGIN PASS 2 ON THE FIRST SUBSYS (SEE SECT. 8.5) AS FOLLOWS :

STARTING PASS 2 ON SUBSYS A

8.5 PASS 2 DIALOGUE

THE OPERATOR RETURNS TO THE FIRST SUBSYSTEM TO PERFORM PASS 2 EITHER THROUGH THE DIALOGUE OF THE ADR 200 START, OR AFTER THE SELECTION OF PARAMETERS AND DRIVES HAS BEEN COMPLETED IN ACCORDANCE WITH THE DIALOGUE OF THE ADR 220 START (SEE SECTION 8.3). IN EITHER CASE, THE PROGRAM INDICATES THE START OF PASS 2 BY TYPING :

** STARTING PASS 2 **

THE PROGRAM THEN DIRECTS THE OPERATOR IN THE UNLOADING, PACK MOVEMENT, AND LOADING OF ALL DRIVES ON THE FIRST SUBSYSTEM, IN THE SAME MANNER AS DESCRIBED FOR PASS 1 (SEE SECTION 8.4).

HOWEVER, AFTER PASS 2 TESTING (SEE SECTION 9.2) IS COMPLETED ON A GIVEN DRIVE, THE ENTIRE TEST RESULTS FOR THAT DRIVE ARE TYPED. THE DETAILS OF THIS PRINTOUT ARE DESCRIBED IN SECTION 10, AFTER THE DETAILS OF THE TESTING ARE DESCRIBED.

WHEN PASS 2 HAS BEEN COMPLETED FOR ALL DRIVES ON THE FIRST SUBSYSTEM, THE PROGRAM DOES ONE OF TWO THINGS: (1) IF THERE IS ONLY ONE SUBSYSTEM (FROM ADR 200 START) OR IF ALL DRIVES ON ALL SUBSYSTEMS HAVE BEEN TESTED IN PASS 2 (FROM ADR 220 START), THE ENTIRE TESTING AND REPORTING HAVE BEEN COMPLETED, AND THE PROGRAM TYPES:

** END OF TESTING **

```
599
600
601
602                          (2) IF THERE IS ANOTHER SUBSYSTEM, HOWEVER, THE  PROGRAM  DIRECTS  THE
603                          OPERATOR  TO PERFORM PASS 2 ON THE NEXT SUBSYSTEM AS FOLLOWS
604                          :
605
606                               STARTING PASS 2 ON SUBSYS B
```

9.0 DESCRIPTION OF TESTS

THE MAIN FUNCTIONAL BLOCKS OF CODE IN THE PROGRAM ARE ASSIGNED TEST NUMBERS, FOR THE PURPOSE OF IDENTIFICATION IN ERROR PRINTOUTS. TEST 0 REFERS TO THE OPERATOR INPUT DIALOGUE ROUTINES DESCRIBED IN SECTIONS 8.1-8.3. THE OTHER TEST NUMBERS ARE ASSIGNED BELOW, IN THE DESCRIPTION OF PASS 1 AND PASS 2 TESTING.

IN THE FOLLOWING SECTIONS, TABLES A-G ARE REFERRED TO. IN THESE TABLES, DRIVES ARE NAMED FROM 0-7 FOR ILLUSTRATIVE PURPOSES, ALTHOUGH THE DRIVES ARE NAMED THE FOLLOWING WAY IN AN ACTUAL SITUATION : A0,A1, A2,...B0,B1,B2,...C0,C1,C2,... ETC. (SEE SECTION 8.0).

9.1 DESCRIPTION OF PASS 1 TESTS

IN PASS 1, THE BASIC READ/WRITE CAPABILITY OF EACH DRIVE IS DEMONSTRATED, AND COMPATIBILITY DATA PATTERNS ARE WRITTEN BY ALL DRIVES UPON THE SAME TEST CARTRIDGE.

THE SEQUENCE OF OPERATIONS PERFORMED ON EACH DRIVE IS AS FOLLOWS:

1. TEST 1 - MOUNTING OF TEST CARTRIDGE FOR PASS 1 - THE OPERATOR MOUNTS THE PACK ON THIS DRIVE AND MANUALLY LOADS THE HEADS, AS DIRECTED BY THE PROGRAM (SEE SECTION 8.4).
2. TEST 2 - BASIC READ/WRITE DATA TEST - THE PROGRAM PERFORMS A WRITE AND WRITE CHECK OPERATION USING A "WORST CASE" DATA PATTERN, AT THE APPROPRIATE SECTOR FOR THIS DRIVE (SEE TABLE A) ON ALL SURFACES. THE PURPOSE OF THIS OPERATION IS TO VERIFY THE BASIC READ/WRITE CAPABILITY OF THE DRIVE ON PASS 1. THE ENTIRE SECTOR IS WRITTEN WITH THE REPETITION OF THE DATA PATTERN SHOWN IN TABLE B.
3. TEST 3 - THE PROGRAM WRITES ALL SECTORS FOR THIS DRIVE WITHIN THE CYLINDER BLOCKS SHOWN IN TABLE C ON ALL SURFACES USING A SINGLE REPEATED WORD OF THE PATTERN IN TABLE G. DRIVE 0 USES WORD 0, DRIVE 1 USES WORD 1, DRIVE 7 USES WORD 7, ETC. THUS, THE DATA FROM EACH DRIVE IS UNIQUE. TABLE C HAS BEEN DETERMINED AS FOLLOWS:

   IN EACH OF THE SEVEN WRITE CURRENT ZONES ON EACH SURFACE, SECTORS ARE WRITTEN WITHIN TWO DISTINCT CYLINDER BLOCKS. THE FIRST 16 CYLINDERS OF EACH WRITE CURRENT ZONE IS THE FIRST BLOCK USED FOR WRITE TEST IN PASS 2. THE LAST 16 CYLINDERS OF EACH CURRENT

ZONE (EXCEPT THE INNERMOST ZONE) IS THE SECOND BLOCK USED FOR READ TEST IN PASS 2.
WITHIN EACH CURRENT ZONE THESE TWO BLOCKS ARE IDENTICALLY WRITTEN. HOWEVER, THE SECTORS DESIGNATED TO EACH DRIVE ARE ROTATED FROM ZONE TO ZONE SO THAT THE DATA APPEARS AT VARIOUS ANGULAR POSITIONS ON THE PACK.

WITHIN EACH CYLINDER BLOCK, UP TO 32 SECTORS ARE WRITTEN (DEPENDING ON THE NUMBER OF DRIVES BEING TESTED) ON EACH CYLINDER.

THE BASIC LAYOUT OF A TYPICAL CYLINDER BLOCK IS SHOWN IN TABLE D, WHERE THE BLOCK SHOWN IS THE READ TEST BLOCK FOR ZONE 1, WHICH STARTS ON CYLINDER 112, AND HAS THE ROTATING STARTING SECTOR = SECTOR 0. EACH NUMBER INSIDE THE BLOCK IS THE NUMBER OF THE DRIVE WHICH WRITES THAT SECTOR. TABLE D SHOWS THE BLOCKS WRITTEN BY EACH OF 16 DRIVES. IF ANY OF THE DRIVES SHOWN ARE NOT PRESENT, HOWEVER, THE BLOCKS RESERVED FOR THE MISSING DRIVES ARE SIMPLY NOT WRITTEN.

THE ABOVE PATTERN OF SECTOR WRITES INSURES THAT DATA FROM EACH DRIVE IS WRITTEN ON ADJACENT CYLINDERS TO DATA FROM EVERY OTHER DRIVE, IN BOTH DIRECTIONS. IN ADDITION, THE ROTATION OF THE ABOVE SECTORS FROM CURRENT ZONE TO CURRENT ZONE INSURES THAT WRITE TEST AND READ TEST ARE DONE AT SEVERAL DIFFERENT ANGULAR POSITIONS WITH RESPECT TO THE CARTRIDGE.

4. TEST 4 - DISMOUNTING OF TEST CARTRIDGE IN PASS 1 - THE OPERATOR UNLOADS THE DRIVE AND DISMOUNTS THE PACK, AS DIRECTED BY THE PROGRAM (SEE SECTION 8.4), TO PROCEED WITH THE ABOVE STEPS ON THE NEXT DRIVE.

9.2 DESCRIPTION OF PASS 2 TESTS

IN PASS 2,THE ABILITY OF EACH DRIVE TO COMPLETELY OVERWRITE DATA WRITTEN BY ALL OTHER DRIVES AND TO READ DATA WRITTEN BY ALL OTHER DRIVES, IS TESTED.

THE SEQUENCE OF OPERATIONS PERFORMED BY EACH DRIVE IS AS FOLLOWS:

1. TEST 5 - MOUNTING OF TEST CARTRIDGE FOR PASS 2 - THE OPERATOR MOUNTS THE PACK ON THIS DRIVE AND MANUALLY LOADS THE HEADS, AS DIRECTED BY THE PROGRAM (SEE SECTION 8.5).

2. TEST 6 - WRITE TEST - NEXT, THE PROGRAM PROCEEDS TO TEST THIS DRIVE'S OVERWRITE CAPABILITY. FIRST, THE APPROPRIATE CYLINDERS IN TABLE E FOR THIS DRIVE ARE OVERWRITTEN, ON EACH SURFACE. THE DATA USED IS A REPETITION OF A SINGLE WORD OF THE PATTERN IN TABLE G, DRIVE 0 USES WORD 0, DRIVE 1 USES WORD 1, DRIVE 7 USES WORD 7, ETC.

   THEN, EACH CYLINDER OVERWRITTEN IS READ BACK BY THIS DRIVE IN EACH OFFSET DIRECTION (+ AND -). THE PROGRAM SCANS FOR READ ERRORS (DCK,HCRC,ETC.) DURING THIS READ, AND IF ONE OCCURS, THE PROGRAM DETERMINES WHICH DRIVE'S DATA HAS NOT BEEN CORRECTLY OVERWRITTEN, AND A SCORE FOR THAT DRIVE IS INCREMENTED. THEN, THE TRANSFER IS CONTINUED AT THE NEXT SECTOR, WITH THAT OFFSET VALUE. THE READS ARE DONE WITH ALL OF THE ABOVE OFFSETS APPLIED, AND A SEPARATE SCORE FOR EACH DRIVE IS KEPT, WHILE THE CURRENT DRIVE IS PERFORMING THE OVERWRITES. FOR EACH TRACK, SCORES ARE AVERAGED OVER ALL CYLS TESTED, IN EACH OFFSET DIRECTION. AT THE COMPLETION OF THE OVERWRITE TEST ON THIS DRIVE, THE SCORES OF ALL THE DRIVES ARE CONVERTED AND STORED, FOR PRINTING AT THE END OF PASS 2 (AS DESCRIBED IN SECTION 10.2). EACH SCORE PROPORTIONAL TO THE OFFSET IN A GIVEN DIRECTION BY THE CURRENT DRIVE WHILE
   SUCCESSFULLY READING THE DATA IT WROTE OVER ONE OF THE OTHER DRIVE'S DATA. THUS, THE PRINTOUT REVEALS WHICH DRIVES ARE INVOLVED, IN A SITUATION IN WHICH A DRIVE CANNOT OVERWRITE ONE OR SEVERAL OTHER DRIVE'S DATA.

3. TEST 7 - DRIVE SELF-TEST - THE PROGRAM NEXT EVALUATES THE DRIVE'S ABILITY TO WRITE AND READ ITS OWN DATA, AT VARIOUS POSITIONS ON THE PACK. FIRST, ALL SECTORS OF THE APPROPRIATE CYLINDERS SHOWN IN TABLE F FOR THIS DRIVE ARE WRITTEN WITH THE DATA PATTERN SHOWN IN TABLE B, FOR ALL SURFACES. THEN, THE SECTORS ARE READ WITH OFFSET IN EACH DIRECTION.
   THE PROGRAM SCANS FOR READ ERRORS DURING EACH READ, AND IT COMPUTES A SCORE WHICH IS PROPORTIONAL TO THE FAILING OFFSET.

THEN, THE SCORES FOR ALL SECTORS READ IN THIS CYLINDER BLOCK ARE AVERAGED, TO COME UP WITH A DRIVE SELF-TEST SCORE FOR EACH SURFACE FOR EACH OFFSET DIRECTION. THIS SCORE IS SAVED FOR LATER USE, TO BECOME THE STANDARD FOR THE READS WHICH ARE TO FOLLOW.

4. TEST 10(OCTAL) - COMPATIBILITY DATA READ TEST - HAVING ESTABLISHED A SELF-TEST SCORE FOR THIS DRIVE, THE PROGRAM PROCEEDS TO PERFORM THE COMPATIBILITY DATA READS OF THE PATTERNS WRITTEN BY ALL THE DRIVES IN EACH CYLINDER BLOCK (ON EACH SURFACE). EACH COMPATIBILITY CYLINDER BLOCK SHOWN IN TABLE C IS READ, A CYLINDER AT A TIME IN EACH OFFSET DIRECTION.
THE PROGRAM SCANS
FOR READ ERRORS DURING EACH READ AND IF ONE OCCURS, THE PROGRAM DETERMINES WHICH DRIVE'S DATA WAS BEING READ AT THAT INSTANT AND A SCORE FOR THAT DRIVE IS DECREMENTED. THEN, THE TRANSFER IS CONTINUED AT THE NEXT SECTOR, WITH THAT OFFSET VALUE. THE READS ARE DONE WITH OFFSETS IN EACH DIRECTION. AND A SEPARATE SCORE FOR EACH DRIVE IS KEPT, WHILE
THE CURRENT DRIVE IS READING THE COMPATIBILITY DATA. THEN, EACH SCORE IS APPROPRIATELY ADJUSTED TO REFLECT THE SELF-TEST SCORE FOR THE CURRENT DRIVE AT THAT PARTICULAR CYLINDER BLOCK. THE SCORES ARE THEN AVERAGED OVER ALL CYLINDER BLOCKS. EACH SCORE IS PROPORTIONAL TO THE CAPABILITY OF THE CURRENT DRIVE TO SUCCESSFULLY READ THE DATA WRITTEN BY ONE OF THE OTHER DRIVES, AND SCORES ARE COMPUTED SEPARATELY FOR EACH SURFACE (TRACK), FOR EACH OFFSET DIRECTION. THUS, THE PRINTOUT REVEALS WHICH DRIVES ARE INVOLVED IN A SITUATION IN WHICH A PARTICULAR DRIVE HAS DIFFICULTY IN READING THE DATA OF ONE OR SEVERAL OTHER DRIVES.

5. TEST 11(OCTAL) - TYPE TEST SCORES AND DISMOUNT PACK IN PASS 2 - THE OVERWRITE AND COMPATIBILITY DATA READ TEST SCORES FOR THIS DRIVE ARE CONVERTED AND TYPED. THEN, THE OPERATOR UNLOADS THE DRIVE AND DISMOUNTS THE PACK AS DIRECTED BY THE PROGRAM (SEE SECTION 8.5), TO PROCEED WITH THE ABOVE STEPS ON THE NEXT DRIVE.

10.0 PRINTOUT OF TEST RESULTS

THE TEST RESULTS ARE PRINTED AT THE END OF PASS 2 ON EACH DRIVE BEING TESTED. THESE RESULTS PERTAIN TO THE OVERWRITE TEST AND THE COMPATIBILITY DATA READ TEST.

10.1 TEST RESULTS

THE RESULTS OF BOTH THE OVERWRITE AND OF THE COMPATIBILITY DATA READ ARE PRINTED,REGARD OF DEGREE OF SUCCESS.
IF THE TEST IS SUCCESSFUL, THE MESSAGES :

```
        ** ALL DRIVES ARE COMPATIBLE ***
```

IS PRINTED. IF THE TEST IS FAILURE, THE TEST RESULTS ARE TABULAR IN FORM AS SHOWN.

IN THE FOLLOWING EXAMPLE, THERE ARE 2 SYSTEMS, AND THE DRIVES BEING TESTED ARE A0,A1,A2,B0, AND B5. THE TEST RESULTS FOR DRIVE A1 ARE SHOWN BELOW:

```
     SCORES FOR DRIVE A1  .

     TRACK     DRIVE      OVRWRT     OVRWRT     READ       READ
      NO.      READ       OFST-      OFST+      OFST-      OFST+

        0       A2         * 0        * 0
```

THE ABOVE EXAMPLE REVEALS A POSSIBLE COMPATIBILITY PROBLEM EXISTS BETWEEN DRIVES A1 AND A2. NOTICE THAT ON TRACK 0, THAT THE OVERWRITE SCORES WERE UNACCEPTABLY LOW (0), AND THE PROGRAM NOTED THESE BAD SCORES WITH AN ASTERISK (*).
ALL ACCEPTABLE TEST RESULTS ARE NOT PRINTED.

11.0 ERROR REPORTING

11.1 COMMON ERRORS

THE FOLLOWING IS A LIST OF COMMON ERROR MESSAGES WHICH ACCOMPANY ERROR TYPEOUTS FROM THE RM03 DRIVE COMPATIBILITY PROGRAM. THE ERRORS ARE SELF-EXPLANATORY.

ADDRESS PLUG CHANGE BIT SET

RH11(70) DIDN'T RESPOND TO ADDRESSING

UNCORRECTABLE MASSBUS PARITY ERROR

FATAL MASSBUS PARITY ERROR

PERSISTENT DEVICE UNSAFE

OPERATION NOT COMPLETED WITHIN TIME LIMIT

DRIVE WENT OFFLINE

NO RESPONSE TO PORT REQUEST

HEADER CRC ERROR

DATA CHECK 'DCK' ERROR

WRITE CHECK ERROR - DATA CHECK 'DCK' SET

WRITE CHCKE ERROR - DATA CHECK 'DCK' NOT SET

HEADER READ ERROR - 'FMT' BIT DROPPED

HEADER READ ERROR - HEADER COMPARE 'HCE' ERROR

FORMAT ERROR 'FER'

HEADER COMPARE 'HCE' ERROR

MISCELLANEOUS DRIVE ERROR

OPERATION INCOMPLETE 'OPI' ERROR

DRIVE TIMING 'DTE' ERROR

PARITY 'PAR' ERROR AFTER OPERATION STARTED

```
884                                     WRITE CLOCK FAILURE 'WCF' ERROR
885
886                                     INVALID ADDRESS 'IAE' ERROR
887
888                                     WRITE LOCK 'WLE' ERROR
889
890                                     DATA CHECK 'DCK' SET DURING WRITE CHECK COMMAND
891
892                                     RH11(70) OR UNIBUS TRANSFER ERROR
893
894                                     BUS ADDRESS OR WORD COUNT INCORRECT
895
896                                     DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED
897
898                                     CAN'T MATCH DATA READ WITH A PATTERN
899
900                                     ERROR BIT(S) SET,BUT NO ERROR SIGNALED BY THE RH11(70)
901
902                                     ECC LOGIC FAILURE - POSITION REGISTER VALUE NOT VALID
903
904                                     BUS ADDRESS AND WORD COUNT NOT CONSISTENT
905
906                                     SEEK INCOMPLETE 'SKI' ERROR
907
908                                     PROGRAM DETECTED POSITIONING ERROR
909
910                                     DRIVE UNSAFE ERROR
911
```

11.2 ERROR HANDLING

ERRORS REPORTED BY THE PROGRAM CONSIST OF COMMON FAILURES RESULTING FROM ATTEMPTED SUBSYSTEM FUNCTIONS, AS WELL AS CERTAIN ERRORS UNIQUE TO PARTICULAR TESTS. EACH ERROR PRINTOUT CONSISTS OF AN ERROR DESCRIPTION AND TEST NUMBER, POSSIBLY FOLLOWED BY HEADER LINES, COLUMN HEADINGS, AND COLUMNS OF REGISTER CONTENTS IN OCTAL. AS MUCH MEANINGFUL REGISTER DATA AS POSSIBLE (FOR EXAMPLE, RH11(70) REGISTERS) ARE REPORTED IN A GIVEN ERROR. OTHER ERROR REPORTS MAY CONSIST OF A SINGLE DESCRIPTIVE LINE.

11.3 ERROR PRINTOUT EXAMPLE

```
        RH11(70) OR UNIBUS TRANSFER ERROR
        DRIVE   RMCS1   RMWC    RMBA    RMDA
        000001  144250  174400  0055030 000431

        RMCS2   RMDS    RMER1   RMAS    RMDB
        000100  010700  000000  000000  000000

        RMMR1   RMDT    RMOF    RMDC    RMMR2
        000050  024024  010000  000716  011777

        RMER2   RMEC1   RMEC2
        000000  004066  000000
```

TABLE A

BASIC READ/WRITE TEST SECTORS

ADDRESS OF SECTOR ON EACH SURFACE

DRIVE NO.	CYLINDER	SECTORS
0	620	0
1	620	1
2	620	2
3	620	3
4	620	4
5	620	5
6	620	6
7	620	7
8	620	8
9	620	9
10	620	10
11	620	11
12	620	12
13	620	13
14	620	14
15	620	15

TABLE B

WORST CASE DATA PATTERN (REPEATS EVERY 16 WORDS)

WORD NO.	DATA (OCTAL)
0	066666
1	155554
2	133331
3	066663
4	155546
5	133315
6	066633
7	155466
8	133155
9	066333
10	154666
11	131555
12	063333
13	146666
14	115555
15	033333

TABLE C

CYLINDER BLOCK ASSIGNMENT FOR A GIVEN SURFACE

CURRENT ZONE - RANGE	OVERWRITE CYL BLK RANGE	COMPATIBILITY CYL BLK RANGE	
1 - CYL 0-127	CYL 0-15	CYL 112-127	
2 - 128-255	128-143	240-255	
3 - 256-383	256-271	368-383	
4 - 384-511	384-399	496-511	
5 - 512-639	512-527	624-639	
6 - 640-767	640-655	752-767	
7 - 768-822	768-783	---	

TABLE D

BASIC CYLINDER BLOCK LAYOUT EXAMPLE

CYLINDER NUMBERS	SECTOR NUMBERS																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
112	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
113	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	
114	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	
115	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	
116	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	
117	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	
118	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	
119	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	
120	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	
121	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	
122	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	
123	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	
124	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	
125	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	
126	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	
127	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0
2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1
3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2
4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3
5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4
6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5
7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6
8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8
10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9
11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10
12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11
13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12
14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13
15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

TABLE E

OVERWRITE CYLINDERS

DRIVE #	CYLINDERS OVERWRITTEN
0	0,128,256,384,512,640,768
1	1,129,257,385,513,641,769
2	2,130,258,386,514,642,770
3	3,131,259,387,515,643,771
4	4,132,260,388,516,644,772
5	5,133,261,389,517,645,773
6	6,134,262,390,518,646,774
7	7,135,263,391,519,647,775
8	8,136,264,392,520,648,776
9	9,137,265,393,521,649,777
10	10,138,266,394,522,650,778
11	11,139,267,395,523,651,779
12	12,140,268,396,524,652,780
13	13,141,269,397,525,653,781
14	14,142,270,398,526,654,782
15	15,143,271,399,527,655,783

1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171

TABLE F

SELF-TEST CYLINDERS

DRIVE #	CYLINDERS
0	17,145,273,401,529,657,785
1	18,146,274,402,530,658,786
2	19,147,275,403,531,659,787
3	20,148,276,404,532,660,788
4	21,149,277,405,533,661,789
5	22,150,278,406,534,662,790
6	23,151,279,407,535,663,791
7	24,152,280,408,536,664,792
8	25,153,281,409,537,665,793
9	26,154,282,410,538,666,794
10	27,155,283,411,539,667,795
11	28,156,284,412,540,668,796
12	29,157,285,413,541,669,797
13	30,158,286,414,542,670,798
14	31,159,287,415,543,671,799
15	32,160,288,416,544,672,800

TABLE G

PSEUDO-RANDOM DATA PATTERN

WORD #	DATA (OCTAL)
0	040135
1	170070
2	070414
3	064531
4	174473
5	062422
6	114352
7	036620
8	010031
9	052336
10	017310
11	011347
12	102367
13	152567
14	001246
15	160073

12.1 RH70(11)/RM03 DRIVER

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

12.2 TO INITIALIZE THE DRIVER:

```
        JSR     PC,RMINIT
        RETURN
```

UPON RETURN YOU MUST EXAMINE THE "DRVSTA" TABLE TO DETERMINE THE DRIVES THAT ARE ONLINE FOR TESTING. THE 'DRVSTA' TABLE IS

EIGHT BYTES; ONE BYTE PER DRIVE. THE STATE OF EACH DRIVE WILL
BE INDICATED AS FOLLOWS:

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

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

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

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

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

CALL:

```
        JSR     R0,RM03         ;MAKE THE CALL
        PNTDPB                  ;ADDRESS OF DPB*
        RETURN1                 ;RETURN IF QUEUE IS FULL
        RETURN2                 ;RETURN IF REQUEST IS IN
                                ;QUEUE OR THERE IS AN
                                ;ERROR CONDITION
```

*DPB (DATA PARAMETER BLOCK)

```
PNTDPB: .BYTE   0               ;(0) DRIVE NUMBER
        .BYTE   0               ;(1) OFFSET VALUE OR FMT22, ECT, AND HCI
        .BYTE   0               ;(2) COMMAND
        .BYTE   0               ;(3) PSEL AND A17 AND A16
        .WORD   0               ;(4) WORD COUNT (MUST BE NEG.)
        .WORD   0               ;(6) BUFFER ADDRESS OR
                                ;REGISTER TABLE POINTER
        .BYTE   0               ;(10) SECTOR ADDRESS OR
                                ;FIRST REG. INDEX
        .BYTE   0               ;(11) TRACK ADDRESS OR
                                ;LAST REG. INDEX
        .WORD   0               ;(12) CYLINDER ADDRESS
        .WORD   0               ;(14) ERROR TABLE POINTER
                                ;POINTS TO THE FIRST OF TWENTY
                                ;LOCATIONS OF WHERE THE DRIVER
                                ;IS TO STORE THE RH70/RM03
                                ;REGISTERS ON AN ERROR. IF LEFT
                                ;ZERO REGISTERS ARE NOT SAVED.
```

```
                        .WORD   0               ;(16) STATUS/ERROR INDICATOR
                                                ;BIT15=1=>ERROR OCCURRED
                                                ;BIT07=1=>DONE
                                                ;BIT14-BIT09 AND BIT06-BIT03
                                                ;INDICATE TYPE OF ERROR

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

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

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


12.4.1  EXAMPLE - WRITE 1000. WORDS

        1$:     JSR     R0,RM03         ;CALL THE DRIVER
                WRTDPB                  ;DPB ADDRESS
                BR      1$              ;WAIT FOR QUEUE IF FULL
        2$:     TST     WRTDPB+16       ;WAIT FOR COMMAND TO COMPLETE
                BEQ     2$
                BMI     ERROR1          ;ERROR OCCURRED
                .
                .
                .

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

                ALTERNATE DPB SETUP

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

12.5    RH70/RM03 REGISTERS
```

MNEMONIC	INDEX
RMCS1	0
RMWC	2
RMBA	4
RMDA	6
RMCS2	10
RMDS	12
RMER1	14
RMAS	16
RMLA	20
RMDB	22
RMMR1	24
RMDT	26
RMSN	30
RMOF	32
RMDC	34
RMHR	36
RMMR2	40
RMER2	42
RMEC1	44
RMEC2	46

12 6 COMMANDS PERFORMED BY THE DRIVER

COMMAND	CODE	COMMAND TYPE
NO OPERATION	101	N
UNLOAD	103	N
SEEK	105	P
RECALIBRATE	107	P
DRIVE CLEAR	111	N
RELEASE	113	N
OFFSET	115	P
RETURN TO CENTER	117	P
READIN PRESET	121	N
PACK ACKNOWLEDGE	123	N
SEARCH	131	P
GET REGISTER(S)	141	S
SET FORMAT	143	S
SELECT DRIVE	145	S

WRITE CHECK DATA	151	D
WRITE CHECK HEADER AND DATA	153	D
WRITE DATA	161	D
WRITE HEADER & DATA	163	D
READ DATA	171	D
READ HEADER & DATA	173	D

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

12.7 DPB STATUS/ERROR INDICATOR WORD

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

BIT NO.	MEANING IF ON A "1"
15	ERROR OCCURRED DONE (BIT07=0); BITS 14-10 SPECIFIES TYPE DONE (BIT07=1); BITS 06-03 SPECIFIES TYPE
14(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON AN OFFLINE OR UNSAFE DRIVE
13(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON A DRIVE THAT HAS AN UNLOAD REQUEST IN QUEUE.
12(2)	PERSISTENT UNSAFE CONDITION EXIST.
11(2)	UNCORRECTABLE PARITY ERROR OCCURRED
10(2)(4)	FATAL PARITY ERROR. A MASSBUS CLEAR WAS PERFORMED, ALL QUEUES WERE EMPTIED, AND ALL DRVACT'S SET TO THE IDLE STATE
3(3)(4)	SOFTWARE TIMEOUT OCCURRED ON THIS DRIVE
8(4)	SOFTWARE TIMEOUT OCCURRED ON ANOTHER DRIVE
7	DONE
6(2)	ERROR OCCURRED DURING AN I O OPERATION

```
                        5(2)            ERROR OCCURRED DURING AN OPERATION
                                        OTHER THAN I/O.

                        4(2)            CORRECTABLE UNSAFE CONDITION OCCURRED

                        3(2)            DRIVE ERROR OCCURRED THAT CAUSED AN
                                        AUTOMATIC "RECALIBRATE" SEQUENCE

                        2               PORT REQUEST TIMEOUT.  THE DRIVER REQUESTED
                                        THE DRIVE BUT THE OPPOSITE PORT DID NOT
                                        RELEASE THE DRIVE WITHIN 20 SECONDS.

                        1               NON-EXISTENT DRIVE REQUESTED.  USER MADE
                                        A REQUEST FOR A NON-EXISTENT DRIVE.

                        (1) =>          REQUEST WASN'T PUT IN QUEUE.  (RH70/RM03
                                        REGISTERS WERE NOT SAVED)

                        (2) =>          REQUEST QUEUE HAS BEEN EMPTIED.  THE DRIVER
                                        ISSUED A "DRIVE CLEAR" TO THE DRIVE.
                                        NOTE: ALL RH70/RM03 REGISTERS ARE SAVED
                                        AS PER DPB+14 BEFORE THE "DRIVE CLEAR".

                        (3) =>          REQUEST QUEUE HAS BEEN EMPTIED.  THE
                                        DRIVER ISSUED A MASSBUS INIT.  ALL
                                        RH70/RM03 REGISTERS FOR THE DRIVE WERE
                                        SAVED AS PER DPB+14 BEFORE THE INIT.

                        (4) =>          A "RECALIBRATE" SHOULD BE ISSUED
                                        BEFORE ANY OTHER COMMAND.

                12.8    ERROR CALLS MADE BY THE DRIVER.

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

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

                        N       TYPE                    DATA AVAILABLE
                        -       ----                    --------------

                        1       RH70 INTERRUPT          *R4= RMCS1'S ADDRESS
                                OCCURRED (RHAS=0)

                        2       UNEXPECTED ATTENTION     R1= DRIVE NUMBER
                                OCCURRED                 R3= ATA BIT
                                                        *R4= RMCS1'S ADDRESS
                                                         R5= (RMAS)
                                                         RMERRS  =RMDS
                                                         RMERRS+2=RMER1
                                                         RMERRS+4=RMER2
                                                         RMERRS+6=RMMR2

                        3       MASSBUS PARITY           RD.ADR= ADDRESS OF REG. READ
```

```
                                        ERROR (MCPE=1)          RD.WRD= WORD READ

                                4       MASSBUS PARITY          WRT.AD= ADDRESS OF REG. WRITTEN
                                        ERROR (PAR=1)           WRT.WD= WORD WRITTEN
                                                                RD.WRD= WORD READ BACK

                                5       ADDRESS PLUG CHANGE     R1= DRIVE NUMBER
                                        31~ SET ('OPE' ERROR)   R3= ATA BIT
                                                               *R4= RMCS1'S ADDRESS
                                                                R5= (RMAS)
                                                                RMERRS  =RMDS
                                                                RMERRS+2=RMER1
                                                                RMERRS+4=RMER2
                                                                RMERRS+6=RMMR2

                                * THIS IS THE ACTUAL UNIBUS ADDRESS (176700)

                                @
```

```
                                .TITLE  CZRMIB0 RM03/2 DR CPT TST
                                ;*COPYRIGHT (C) 1977
                                ;*DIGITAL EQUIPMENT CORP.
                                ;*MAYNARD, MASS. 01754
                                ;*
                                ;*PROGRAM BY C. CHEN
                                ;*
                                ;*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
                                ;*        12            INHIBIT TRACE TRAP
                                ;*        11            INHIBIT ITERATIONS
                                ;*        10            BELL ON ERROR
                                ;*         9            LOOP ON ERROR
                                ;*         8            LOOP ON TEST IN SWR<7:0>
                                ;*         7            TYPE THE BAD SECTOR FILE
                                .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
```

```
1582  000000                PR0=    0               ;;PRIORITY LEVEL 0
1583  000040                PR1=    40              ;;PRIORITY LEVEL 1
1584  000100                PR2=    100             ;;PRIORITY LEVEL 2
1585  000140                PR3=    140             ;;PRIORITY LEVEL 3
1586  000200                PR4=    200             ;;PRIORITY LEVEL 4
1587  000240                PR5=    240             ;;PRIORITY LEVEL 5
1588  000300                PR6=    300             ;;PRIORITY LEVEL 6
1589  000340                PR7=    340             ;;PRIORITY LEVEL 7
1590
1591                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
1592  100000                SW15=   100000
1593  040000                SW14=   40000
1594  020000                SW13=   20000
1595  010000                SW12=   10000
1596  004000                SW11=   4000
1597  002000                SW10=   2000
1598  001000                SW09=   1000
1599  000400                SW08=   400
1600  000200                SW07=   200
1601  000100                SW06=   100
1602  000040                SW05=   40
1603  000020                SW04=   20
1604  000010                SW03=   10
1605  000004                SW02=   4
1606  000002                SW01=   2
1607  000001                SW00=   1
1608                        .EQUIV  SW09,SW9
1609                        .EQUIV  SW08,SW8
1610                        .EQUIV  SW07,SW7
1611                        .EQUIV  SW06,SW6
1612                        .EQUIV  SW05,SW5
1613                        .EQUIV  SW04,SW4
1614                        .EQUIV  SW03,SW3
1615                        .EQUIV  SW02,SW2
1616                        .EQUIV  SW01,SW.
1617                        .EQUIV  SW00,SW0
1618
1619                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
1620  100000                BIT15=  100000
1621  040000                BIT14=  40000
1622  020000                BIT13=  20000
1623  010000                BIT12=  10000
1624  004000                BIT11=  4000
1625  002000                BIT10=  2000
1626  001000                BIT09=  1000
1627  000400                BIT08=  400
1628  000200                BIT07=  200
1629  000100                BIT06=  100
1630  000040                BIT05=  40
1631  000020                BIT04=  20
1632  000010                BIT03=  10
1633  000004                BIT02=  4
1634  000002                BIT01=  2
1635  000001                BIT00=  1
1636                        .EQUIV  BIT09,BIT9
1637                        .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
176700                          ABASE=176700
000254                          AVECT1=254


                        .SBTTL  RM03 REGISTERS


                        .SBTTL  RM03 DRIVER COMMANDS

                        ;;*******************************************************************
000101                  RNOP    =       101     ;NO OPERATION
000105                  SEEK    =       105     ;SEEK
000107                  RECAL   =       107     ;RECALIBRATE
000111                  DRVCLR  =       111     ;DRIVE CLEAR
000113                  RELSE   =       113     ;RELEASE
000115                  OFFSET  =       115     ;OFFSET
000117                  RTC     =       117     ;RETURN TO CENTER LINE
000121                  READIN  =       121     ;READ IN PRESET
000123                  ACK     =       123     ;PACK ACKNOWLEDGE
000131                  SEARCH  =       131     ;SEARCH
000141                  GETREG  =       141     ;GET REGISTERS
000143                  SETFMT  =       143     ;SET FORMAT (& ECI OR HCI)
000145                  SELDRV  =       145     ;SELECT DRIVE
000151                  WCKD    =       151     ;WRITE CHECK DATA
000153                  WCKHD   =       153     ;WRITE CHECK HEADER & DATA
000161                  WRTDAT  =       161     ;WRITE DATA
000163                  WRTHD   =       163     ;WRITE HEADER & DATA
000171                  RDDAT   =       171     ;READ DATA
000173                  RDHD    =       173     ;READ HEADER & DATA


                        .SBTTL  TRAP CATCHER

000000                          .=0
```

```
                                ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                                        .=174
000174  000000                  DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
000176  000000                  SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
                                .SBTTL  STARTING ADDRESS(ES)
000200  000137  003062                  JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204  000137  003072                  JMP     @#START1        ;CHANGE THE RH11 UNIBUS ADDRESS
                                                                ;AFTER INITIAL START
                                        .=220                   ;SECOND PASS STARTING ADDRESS
000220  000137  003102                  JMP     @#START2
                                .SBTTL  ACT11 HOOKS

                                ;;*********************************************************************
                                ;HOOKS REQUIRED BY ACT11
                                        $SVPC=.                 ;SAVE PC
                                        .=46
000046  017742                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
                                        .=52
000052  040000                          .WORD   40000           ;;2)SET LOC.52 TO 40000
                                        .=$SVPC                 ;;  RESTORE PC
                                        .=1100
                                .SBTTL  APT PARAMETER BLOCK

                                ;;*********************************************************************
                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                ;;*********************************************************************
                                        .$X=.   ;;SAVE CURRENT LOCATION
                                        .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024  000200                         200     ;;FOR APT START UP
                                        .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044  001100                          $APTHDR ;;POINT TO APT HEADER BLOCK
                                        .=.$X   ;;RESET LOCATION COUNTER
                                ;;*********************************************************************
                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                ;INTERFACE SPEC.

001100                          $APTHD:
001100  000000                  $HIBTS: .WORD   0        ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001102  001210                  $MBADR: .WORD   $MAIL    ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001104  000764                  $TSTM:  .WORD   500.     ;;RUN TIM OF LONGEST TEST
001106  000764                  $PASTM: .WORD   500.     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001110  000764                  $UNITM: .WORD   500.     ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
001112  000032                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
001114                          TAB.XY=.
```

```
                                .SBTTL  COMMON TAGS

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

                                        .=TAB.XY
001114                  $CMTAG:                         ;;START OF COMMON TAGS
001114  000000                  .WORD   0
001116  000                     $TSTNM: .BYTE   0       ;;CONTAINS THE TEST NUMBER
001117  000                     $ERFLG: .BYTE   0       ;;CONTAINS ERROR FLAG
001120  000000                  $ICNT:  .WORD   0       ;;CONTAINS SUBTEST ITERATION COUNT
001122  000000                  $LPADR: .WORD   0       ;;CONTAINS SCOPE LOOP ADDRESS
001124  000000                  $LPERR: .WORD   0       ;;CONTAINS SCOPE RETURN FOR ERRORS
001126  000000                  $ERTTL: .WORD   0       ;;CONTAINS TOTAL ERRORS DETECTED
001130  000                     $ITEMB: .BYTE   0       ;;CONTAINS ITEM CONTROL BYTE
001131  001                     $ERMAX: .BYTE   1       ;;CONTAINS MAX. ERRORS PER TEST
001132  000000                  $ERRPC: .WORD   0       ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001134  000000                  $GDADR: .WORD   0       ;;CONTAINS ADDRESS OF 'GOOD' DATA
001136  000000                  $BDADR: .WORD   0       ;;CONTAINS ADDRESS OF 'BAD' DATA
001140  000000                  $GDDAT: .WORD   0       ;;CONTAINS 'GOOD' DATA
001142  000000                  $BDDAT: .WORD   0       ;;CONTAINS 'BAD' DATA
001144  000000                          .WORD   0       ;;RESERVED--NOT TO BE USED
001146  000000                          .WORD   0
001150  000                     $AUTOB: .BYTE   0       ;;AUTOMATIC MODE INDICATOR
001151  000                     $INTAG: .BYTE   0       ;;INTERRUPT MODE INDICATOR
001152  000000                          .WORD   0
001154  177570                  $SWR:   .WORD   DSWR    ;;ADDRESS OF SWITCH REGISTER
001156  177570                  DISPLAY: .WORD  DDISP   ;;ADDRESS OF DISPLAY REGISTER
177560                          $TKS:   177560          ;;TTY KBD STATUS
177562                          $TKB:   177562          ;;TTY KBD BUFFER
177564                          $TPS:   177564          ;;TTY PRINTER STATUS REG. ADDRESS
177566                          $TPB:   177566          ;;TTY PRINTER BUFFER REG. ADDRESS
001170  000                     $NULL:  .BYTE   0       ;;CONTAINS NULL CHARACTER FOR FILLS
001171  002                     $FILLS: .BYTE   2       ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001172  012                     $FILLC: .BYTE   12      ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001173  000                     $TPFLG: .BYTE   0       ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001174  000000                  $TIMES: 0               ;;MAX. NUMBER OF ITERATIONS
001176  000000                  $ESCAPE:0               ;;ESCAPE ON ERROR ADDRESS
001200  177607  000377         $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001204  077                     $QUES:  .ASCII  /?/     ;;QUESTION MARK
001205  015                     $CRLF:  .ASCII  <15>    ;;CARRIAGE RETURN
001206  000012                  $LF:    .ASCIZ  <12>    ;;LINE FEED
                                ;;*****************************************************************
                                .SBTTL  APT MAILBOX-ETABLE

                                ;;*****************************************************************
                                .NLIST ME
                                ;
                                ;

                                .EVEN
001210                  $MAIL:                          ;;APT MAILBOX
001210  000000                  $MSGTY: .WORD   AMSGTY   ;;MESSAGE TYPE CODE
001212  000000                  $FATAL: .WORD   AFATAL   ;;FATAL ERROR NUMBER
001214  000000                  $TESTN: .WORD   ATESTN   ;;TEST NUMBER
```

```
1797  001216  000000                $PASS:   .WORD   APASS   ;;PASS COUNT
1798  001220  000000                $DEVCT:  .WORD   ADEVCT  ;;DEVICE COUNT
1799  001222  000000                $UNIT:   .WORD   AUNIT   ;;I/O UNIT NUMBER
1800  001224  000000                $MSGAD:  .WORD   AMSGAD  ;;MESSAGE ADDRESS
1801  001226  000000                $MSGLG:  .WORD   AMSGLG  ;;MESSAGE LENGTH
1802  001230                        $ETABLE:                 ;;APT ENVIRONMENT TABLE
1803  001230     000                $ENV:    .BYTE   AENV    ;;ENVIRONMENT BYTE
1804  001231     000                $ENVM:   .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
1805  001232  000000                $SWREG:  .WORD   ASWREG  ;;APT SWITCH REGISTER
1806  001234  000000                $USWR:   .WORD   AUSWR   ;;USER SWITCHES
1807  001236  000000                $CPUOP:  .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
1808                                                         BITS 15-11=CPU TYPE
1809                                                         ;*
1810                                                         ;*
1811                                                         ;*
1812                                                         ;*
1813                                                         ;*
1814  001240     000                $MAMS1:  .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
1815  001241     000                $MTYP1:  .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
1816                                                         MEM.TYPE BYTE    --  (HIGH BYTE)
1817                                                         ;*
1818                                                         ;*
1819                                                         ;*
1820  001242  000000                $MADR1:  .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
1821                                                         ;*
1822  001244     000                $MAMS2:  .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
1823  001245     000                $MTYP2:  .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
1824  001246  000000                $MADR2:  .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
1825  001250     000                $MAMS3:  .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
1826  001251     000                $MTYP3:  .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
1827  001252  000000                $MADR3:  .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
1828  001254     000                $MAMS4:  .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
1829  001255     000                $MTYP4:  .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
1830  001256  000000                $MADR4:  .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
1831  001260  000254                $VECT1:  .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
1832  001262  000000                $VECT2:  .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
1833  001264  176700                $BASE:   .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
1834  001266  000000                $DEVM:   .WORD   ADEVM   ;;DEVICE MAP
1835  001270  000000                $CDW1:   .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
1836  001272  000000                $CDW2:   .WORD   ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
1837  001274                        $ETEND:
1838                                .MEXIT
1839             000015             CR       =       15
1840             000012             LF       =       12
1841  001274  176700                $RMADR:  .WORD   176700          ;FIRST ADDRESS OF RH11/RM03 REGISTERS
1842  001276  000254                $RMVEC:  .WORD   254             ;RM03 VECTOR ADDRESS
1843  001300  172540                $LKCSR:  .WORD   172540          ;ADDR OF KW11-P STATUS REGISTER
1844  001302  172542                $LKCSB:  .WORD   172542          ;ADDR OF KW11-P COUNTER BUFFER
1845  001304  000104                $LPVEC:  .WORD   104             ;ADDR OF KW11-P VECTOR
1846  001306  177546                $LKS:    .WORD   177546          ;ADDR OF KW11-L STATUS REGISTER
1847  001310  000100                $LLVEC:  .WORD   100             ;ADDR OF KW11-L VECTOR
1848  001312  177777                PCLOCK:  .WORD   -1              ;'0' IF KW11-P IS ON SYSTEM
1849  001314  177777                CLKFLG:  .WORD   -1              ;'0' IF A CLOCK IS AVAILABLE
1850  001316  000074                HZ:      .WORD   74              ;74 (8) IF 60 HZ SYSTEM; 62 (8  IF 50 HZ SYSTEM
1851  001320  000000                STATIN:  .WORD   0               ;'TYPE STATISTICS' INDICATOR
1852  001322  000000                PACK:    .WORD   0               ;ENTRY TO THE TABLE D
```

```
                001222          DRIVE   =       $UNIT           ;DRIVE # STORAGE: ERRORS 1-5 & 10
001324  000000                  ATTN:   .WORD   0               ;ATTN REG STORAGE: ERRORS 1-5 & 10
001326  000000                  UNIT:   .WORD   0               ;DRIVE # STORAGE FOR PRINTOUT
                                                                ;RETRY COUNT IN THE UPPER BYTE
001330  000000                  LSTAD:  .WORD   0               ;STORE LAST MEMORY ADDRESS HERE
001332  000000                  CHGADR: .WORD   0               ;CHANGE RH11 UNIBUS ADDRESS FLAG
001334  000000                  CFLAG:  .WORD   0               ;'CONTROL C' FLAG
001336  000000                  TSTNM:  .WORD   0               ;TEST NUMBER FOR PRINT AND SCORE RT.
001340  000000                  BADSEC: .WORD   0               ;BAD SECTOR/TRACK FLAG
001342  000000                  HOUR:   .WORD   0               ;HOUR COUNT STORED HERE (MAXIMUM - 999.)
001344  000000                  MINUTE: .WORD   0               ;MINUTE'S COUNT STORED HERE
001346  000000                  SECOND: .WORD   0               ;SECOND'S COUNT STORED HERE
001350  000000                  SIXTEE: .WORD   0               ;TIMER ROUTINE COUNTER (FOR ONE SECOND)
001352  000000                  CMCNT:  .WORD   0               ;ZONE COUNT
001354  000000                  CMCYL:  .WORD   0               ;CYLINDER ADDRESS
001356  000000                  STARSC: .WORD   0               ;STARTING SECTOR (FOR TEST 6,8)
001360  000000                  CMSEC:  .WORD   0               ;DALTA CYLINDER COUNT
001362  000000                  CMTRK:  .WORD   0               ;TRACK ADDRESS
001364  000000                  NULINE: .WORD   0       ;NEW LINE FLAG AND COLUMN CTR
001366  000037                  SECLMT: .WORD   31.             ;SECTOR ADDRESS LIMIT
001370  000004                  TRKLMT: .WORD   4.              ;TRACK ADDRESS LIMIT
001372  001466                  CYLIMT: .WORD   822.            ;CYLINDER ADDRESS LIMIT FOR RM03
001374  000000                  FAULT:  .WORD   0               ;=1,IF  ALL DRIVES NOT COMPATIBLE


                                .SBTTL  COMMON PARAMETERS


                                .SBTTL  TABLES, CONSTANTS, AND VARIABLE LOCATIONS

                                ;;*****************************************************************
                                ;TABLE D
                                ;TABLE LISTED BELOW SPECIFIES THE SECTORS TO BE WRITTEN
                                ;BY A LOGICAL DRIVE.  EACH LOGICAL DRIVE WRITES TWO SECTORS IN ONE
                                ;CYLINDER , 16 CYLINDERS IN ONE BLOCK, 2 BLOCKS IN ONE WRITE-CURRENT
                                ;ZONE AND 7 CURRENT ZONES IN A PACK.
                                ;
                                ;
001376  000     017     015     LOG0:   .BYTE   0,15.,13.,10.,6,1,11.,4,12.,13.,9.,14.,2,5,7,8.    ;DRIVE 0
001401  012     006     001
001404  013     004     014
001407  015     011     016
001412  002     005     007
001415  010
001416  001     000     016     LOG1:   .BYTE   1,0,14.,11.,7,2,12.,5,13.,4,10.,15.,3,6,8.,9.     ;DRIVE 1
001421  013     007     002
001424  014     005     015
001427  004     012     017
001432  003     006     010
001435  011
001436  002     001     017     LOG2:   .BYTE   2,1,15.,12.,8.,3,13.,6,14.,5,11.,0,4,7,9.,10.        ;DRIVE 2
001441  014     010     003
001444  015     006     016
001447  005     013     000
001452  004     007     011
```

```
1909  001455      012
1910  001456      003       002       000   LOG3:    .BYTE   3,2,0,13.,9.,4,14.,7,15.,6,12.,1,5,8.,10.,11.
1911  001461      015       011       004
1912  001464      016       007       017
1913  001467      006       014       001
1914  001472      005       010       012
1915  001475      013
1916  001476      004       003       001   LOG4:    .BYTE   4,3,1,14.,10.,5,15.,8.,0,7,13.,2,6,9.,11.,12.
1917  001501      016       012       005
1918  001504      017       010       000
1919  001507      007       015       002
1920  001512      006       011       013
1921  001515      014
1922  001516      005       004       002   LOG5:    .BYTE   5,4,2,15.,11.,6,0,9.,1,8.,14.,3,7,10.,12.,13.
1923  001521      017       013       006
1924  001524      000       011       001
1925  001527      010       016       003
1926  001532      007       012       014
1927  001535      015
1928  001536      006       005       003   LOG6:    .BYTE   6,5,3,0,12.,7,1,10.,2,9.,15.,4,8.,11.,13.,14.
1929  001541      000       014       007
1930  001544      001       012       002
1931  001547      011       017       004
1932  001552      010       013       015
1933  001555      016
1934  001556      007       006       004   LOG7:    .BYTE   7,6,4,1,13.,8.,2,11.,3,10.,0,5,9.,12.,14.,15.
1935  001561      001       015       010
1936  001564      002       013       003
1937  001567      012       000       005
1938  001572      011       014       016
1939  001575      017
1940  001576      010       007       005   LOG8:    .BYTE   8.,7,5,2,14.,9.,3,12.,4,11.,1,6,10.,13.,15.,0
1941  001601      002       016       011
1942  001604      003       014       004
1943  001607      013       001       006
1944  001612      012       015       017
1945  001615      000
1946  001616      011       010       006   LOG9:    .BYTE   9.,8.,6,3,15.,10.,4,13.,5,12.,2,7,11.,14.,0,1
1947  001621      003       017       012
1948  001624      004       015       005
1949  001627      014       002       007
1950  001632      013       016       000
1951  001635      001
1952  001636      012       011       007   LOG10:   .BYTE   10.,9 ,7.,4,0,11.,5,14.,6,13.,3,8.,12.,15.,1,2
1953  001641      004       000       013
1954  001644      005       016       006
1955  001647      015       003       010
1956  001652      014       017       001
1957  001655      002
1958  001656      013       012       010   LOG11:   .BYTE   11.,10.,8.,5,1,12.,6,15.,7,14.,4,9.,13.,0,2,3
1959  001661      005       001       014
1960  001664      006       017       007
1961  001667      016       004       011
1962  001672      015       000       002
1963  001675      003
1964  001676      014       013       011   LOG12:   .BYTE   12.,11.,9.,6,2,13.,7,0,8.,15.,5,10.,14.,1,3,4
```

```
 1965  001701  006  002  015
 1966  001704  007  000  010
 1967  001707  017  005  012
 1968  001712  016  001  003
 1969  001715  004
 1970  001716  015  014  012  LOG13:  .BYTE   13.,12.,10.,7,3,14.,8.,1,9.,0,6,11.,15.,2,4,5
 1971  001721  007  003  016
 1972  001724  010  001  011
 1973  001727  000  006  013
 1974  001732  017  002  004
 1975  001735  005
 1976  001736  016  015  013  LOG14:  .BYTE   14.,13.,11.,8.,4,15.,9.,2,10.,1,7,12.,0,3,5,6
 1977  001741  010  004  017
 1978  001744  011  002  012
 1979  001747  001  007  014
 1980  001752  000  003  005
 1981  001755  006
 1982  001756  017  016  014  LOG15:  .BYTE   15.,14.,12.,9.,5,0,10.,3,11.,2,8.,13.,1,4,6,7
 1983  001761  011  005  000
 1984  001764  012  003  013
 1985  001767  002  010  015
 1986  001772  001  004  006
 1987  001775  007
 1988
 1989  001776  000000  ASNLST: .WORD   0              ;A BIT SET IS AN ASSIGNED LOGICAL DRIVE
 1990
 1991  002000  000000  ASSGN1: .WORD   0              ;A BIT SET IS AN ASSIGNED LOGICAL DRIVE FOR PASS 1
 1992
 1993  002002  000000  ASSGN2: .WORD   0              ;A BIT SET IS AN ASSIGNED LOGICAL DRIVE FOR PASS 2
 1994
 1995  002004  000020  SYSADR: .BLKW   16.     ;SUB SYSTEM ADDRESS TABLE
 1996
 1997  002044  000000  TABLEX: .WORD   0       ;CURRENT SELECTED SCORE BOARD
 1998
 1999                  ;SCORE BOARD TABLES
 2000                  ;
 2001                  ;       TABLE OF OVERWRITE SCORE,NEGATIVE OFFSET SCORE
 2002
 2003
 2004  002046  000  000  000  OVWN0:  .BYTE   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 2005  002051  000  000  000
 2006  002054  000  000  000
 2007  002057  000  000  000
 2008  002062  000  000  000
 2009  002065  000
 2010  002066  000  000  000  OVWN1:  .BYTE   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 2011  002071  000  000  000
 2012  002074  000  000  000
 2013  002077  000  000  000
 2014  002102  000  000  000
 2015  002105  000
 2016  002106  000  000  000  OVWN2:  .BYTE   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 2017  002111  000  000  000
 2018  002114  000  000  000
 2019  002117  000  000  000
 2020  002122  000  000  000
```

```
2021  002125  000
2022  002126  000  000  000  OVWN3:  .BYTE  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2023  002131  000  000  000
2024  002134  000  000  000
2025  002137  000  000  000
2026  002142  000  000  000
2027  002145  000
2028  002146  000  000  000  OVWN4:  .BYTE  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2029  002151  000  000  000
2030  002154  000  000  000
2031  002157  000  000  000
2032  002162  000  000  000
2033  002165  000
2034
2035                          ;TABLE OF OVERWRITE SCORE,POSITIVE OFFSET SCORE
2036
2037
2038  002166  000  000  000  OVWP0:  .BYTE  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2039  002171  000  000  000
2040  002174  000  000  000
2041  002177  000  000  000
2042  002202  000  000  000
2043  002205  000
2044
2045  002206  000  000  000  OVWP1:  .BYTE  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2046  002211  000  000  000
2047  002214  000  000  000
2048  002217  000  000  000
2049  002222  000  000  000
2050  002225  000
2051
2052  002226  000  000  000  OVWP2:  .BYTE  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2053  002231  000  000  000
2054  002234  000  000  000
2055  002237  000  000  000
2056  002242  000  000  000
2057  002245  000
2058
2059  002246  000  000  000  OVWP3:  .BYTE  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2060  002251  000  000  000
2061  002254  000  000  000
2062  002257  000  000  000
2063  002262  000  000  000
2064  002265  000
2065
2066  002266  000  000  000  OVWP4:  .BYTE  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2067  002271  000  000  000
2068  002274  000  000  000
2069  002277  000  000  000
2070  002302  000  000  000
2071  002305  000
2072
2073                          ;TABLE OF READ SCORE,NEGATIVE OFFSET SCORE
2074
2075  002306  000  000  000  RDNO:  .BYTE  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2076  002311  000  000  000
```

```
2077  002314  000  000  000
2078  002317  000  000  000
2079  002322  000  000  000
2080  002325  000
2081  002326  000  000  000  RDN1:   .BYTE   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2082  002331  000  000  000
2083  002334  000  000  000
2084  002337  000  000  000
2085  002342  000  000  000
2086  002345  000
2087  002346  000  000  000  RDN2:   .BYTE   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2088  002351  000  000  000
2089  002354  000  000  000
2090  002357  000  000  000
2091  002362  000  000  000
2092  002365  000
2093  002366  000  000  000  RDN3:   .BYTE   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2094  002371  000  000  000
2095  002374  000  000  000
2096  002377  000  000  000
2097  002402  000  000  000
2098  002405  000
2099  002406  000  000  000  RDN4:   .BYTE   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2100  002411  000  000  000
2101  002414  000  000  000
2102  002417  000  000  000
2103  002422  000  000  000
2104  002425  000
2105
2106                          ;TABLE OF READ SCORE,POSITIVE OFFSET SCORE
2107
2108  002426  000  000  000  RDP0:    BYTE   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2109  002431  000  000  000
2110  002434  000  000  000
2111  002437  000  000  000
2112  002442  000  000  000
2113  002445  000
2114  002446  000  000  000  RDP1:   .BYTE   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2115  002451  000  000  000
2116  002454  000  000  000
2117  002457  000  000  000
2118  002462  000  000  000
2119  002465  000
2120  002466  000  000  000  RDP2:   .BYTE   0,0,0 0,0,0,0,0,0,0,0,0,0,0,0,0
2121  002471  000  000  000
2122  002474  000  000  000
2123  002477  000  000  000
2124  002502  000  000  000
2125  002505  000
2126  002506  000  000  000  RDP3:   .BYTE   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2127  002511  000  000  000
2128  002514  000  000  000
2129  002517  000  000  000
2130  002522  000  000  000
2131  002525  000
2132  002526  000  000  000  RDP4:   .BYTE   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

```
2133  002531  000  000  000
2134  002534  000  000  000
2135  002537  000  000  000
2136  002542  000  000  000
2137  002545  000
2138
2139                          ;TABLE OF SELF TEST SCORE
2140
2141  002546  000  000        SELF0:  .BYTE   0,0
2142  002550  000  000        SELF1:  .BYTE   0,0
2143  002552  000  000        SELF2:  .BYTE   0,0
2144  002554  000  000        SELF3:  .BYTE   0,0
2145  002556  000  000        SELF4:  .BYTE   0,0
2146
2147                          ;THE START LOGICAL DRIVE # TO WRITE ON EACH CYLINDER OF A BLOCK
2148                          ;16 CYLINDERS,2 BLOCKS,TOTAL 32 CYLINDERS IN ONE ZONE
2149
2150  002560  000  001  003  INDST:  .BYTE   0,1,3,6,10.,15.,5,12.,4,13.,7,2,14.,11.,9.,8.
2151  002563  006  012  017
2152  002566  005  014  004
2153  002571  015  007  002
2154  002574  016  013  011
2155  002577  010
2156
2157
2158                          ;BUFTBL:                        ;BUFFER ALLOCATION TABLE ENTRY COUNT
2159                          ;       .IRP    X,<0,1,2,3,4,5,6,7,10,11,12,13,14,15,16,17>
2160                          ;       .WORD   #ENDPGM+<258.*X>        ;BUFFER ADDRESS OF DRIVE X
2161                          ;       .ENDM
2162
2163  002600  037320          BLKADR: .WORD   DRIV0            ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 0
2164  002602  037342                  .WORD   DRIV1            ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 1
2165  002604  037364                  .WORD   DRIV2            ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 2
2166  002606  037406                  .WORD   DRIV3            ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 3
2167  002610  037430                  .WORD   DRIV4            ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 4
2168  002612  037452                  .WORD   DRIV5            ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 5
2169  002614  037474                  .WORD   DRIV6            ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 6
2170  002616  037516                  .WORD   DRIV7            ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 7
2171  002620  037540                  .WORD   DRIV10           ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 10
2172  002622  037562                  .WORD   DRIV11           ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 11
2173  002624  037604                  .WORD   DRIV12           ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 12
2174  002626  037626                  .WORD   DRIV13           ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 13
2175  002630  037650                  .WORD   DRIV14           ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 14
2176  002632  037672                  .WORD   DRIV15           ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 15
2177  002634  037714                  .WORD   DRIV16           ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 16
2178  002636  037736                  .WORD   DRIV17           ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 17
2179
2180                                  .EVEN
2181
2182  002640  000000  OFFCOD: .WORD   0              ;OFFSET CODE TABLE
2183                                                 ;NUMBER FOR NEGATIVE OFFSET (DIR = OUT)
2184                                                 ;NUMBER FOR POSITIVE OFFSET (DIR = IN)
2185
2186                  .EVEN
2187
2188  002642  066666  STNDAT: .WORD   066666
```

```
2189  002644  155554                        .WORD   155554
2190  002646  133331                        .WORD   133331
2191  002650  066663                        .WORD   066663
2192  002652  155546                        .WORD   155546
2193  002654  133315                        .WORD   133315
2194  002656  066633                        .WORD   066633
2195  002660  155466                        .WORD   155466
2196  002662  133155                        .WORD   133155
2197  002664  066333                        .WORD   066333
2198  002666  154666                        .WORD   154666
2199  002670  131555                        .WORD   131555
2200  002672  063333                        .WORD   063333
2201  002674  146666                        .WORD   146666
2202  002676  115555                        .WORD   115555
2203  002700  033333                        .WORD   033333
2204  002702  040135                PSEUDO: .WORD   040135
2205  002704  177070                        .WORD   177070
2206  002706  070414                        .WORD   070414
2207  002710  064531                        .WORD   064531
2208  002712  174473                        .WORD   174473
2209  002714  062422                        .WORD   062422
2210  002716  114352                        .WORD   114352
2211  002720  036620                        .WORD   036620
2212  002722  010031                        .WORD   010031
2213  002724  052336                        .WORD   052336
2214  002726  017310                        .WORD   017310
2215  002730  011347                        .WORD   011347
2216  002732  102367                        .WORD   102367
2217  002734  152567                        .WORD   152567
2218  002736  001246                        .WORD   001246
2219  002740  160073                        .WORD   160073
2220
2221
2222                          ;;*****************************************************************
2223
```

```
                                .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


002742                          $ERRTB:


                                ;ERROR 1

002742  032274                          EM1                     ;RH70 INTERRUPT OCCURRED (RMAS = 0)
002744  034722                          DH1
002746  035344                          DT1
002750  035470                          DF1

                                ;ERROR 2

002752  032343                          EM2                     ;UNEXPECTED ATTENTION OCCURRED
002754  034731                          DH2
002756  035350                          DT2
002760  035471                          DF2

                                ;ERROR 3

002762  032401                          EM3                     ;MASSBUS PARITY ERROR (MCPE=1)
002764  035013                          DH3
002766  035366                          DT3
002770  035477                          DF3

                                ;ERROR 4

002772  032437                          EM4                     ;MASSBUS PARITY ERROR (PAR=1)
002774  035044                          DH4
002776  035376                          DT4
003000  035502                          DF4

                                ;ERROR 5

003002  032474                          EM5                     ;ADDRESS PLUG BIT CHANGED
003004  034731                          DH2
003006  035350                          DT2
003010  035471                          DF2

                                ;ERROR 6

003012  032530                          EM6                     ;RH11 DIDN'T RESPOND TO ADDRESSING
003014  035106                          DH6
```

```
2280  003016  035410                        DT6
2281  003020  035470                        DF1
2282
2283                                  ;ERROR 7
2284
2285  003022  000000                        0
2286  003024  035117                        DH7
2287  003026  035414                        DT7
2288  003030  000000                        0
2289
2290                                  ;ERROR 10
2291
2292  003032  000000                        0
2293  003034  035170                        DH10
2294  003036  035430                        DT10
2295  003040  000000                        0
2296
2297                                  ;ERROR 11
2298
2299  003042  000000                        0
2300  003044  035241                        DH11
2301  003046  035444                        DT11
2302  003050  000000                        0
2303
2304                                  ;ERROR 12
2305
2306  003052  000000                        0
2307  003054  035312                        DH12
2308  003056  035460                        DT12
2309  003060  000000                        0
2310
2311
2312                                  .SBTTL  SETUP AND INITIALIZATION ROUTINE
2313
2314                                  ;       START ADDRESS = 200
2315                                  ;       ADDRESS TO CHANGE RH11 UNIBUS ADDRESS = 204
2316
2317
2318
2319  003062  012737  000400  001332  START:  MOV     #400,CHGADR     ;200 START ADDRESS FLAG
2320  003070  000406                  BR      START3
2321  003072  012737  177777  001332  START1: MOV     #-1,CHGADR      ;204 START ADDRESS FLAG
2322  003100  000402                  BR      START3
2323  003102  005037  001332          START2: CLR     CHGADR          ;220 START ADDRESS FLAG
2324                                                                  ;COMMON BRANCH POINT
2325  003106  000005                  START3: RESET                   ;CLEAR THE BUS
2326                                  .SBTTL  INITIALIZE THE COMMON TAGS
2327                                  ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
2328  003110  012706  001114                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
2329  003114  005026                  CLR     (R6)+                   ;;CLEAR MEMORY LOCATION
2330  003116  022706  001154                  CMP     #SWR,R6 ;;DONE?
2331  003122  001374                  BNE     .-6                     ;;LOOP BACK IF NO
2332  003124  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
2333                                  ;;INITIALIZE A FEW VECTORS
2334  003130  012737  023012  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
2335  003136  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
```

```
2336  003144  012737  020016  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
2337  003152  012737  000340  000032          MOV     #340,@#EMTVEC+2  ;;LEVEL 7
2338  003160  012737  023272  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
2339  003166  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
2340  003174  012737  021102  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
2341  003202  012737  000340  000026          MOV     #340,@#PWRVEC+2  ;;LEVEL 7
2342  003210  013737  017710  017702          MOV     $ENDCT,$EOPCT    ;;SETUP END-OF-PROGRAM COUNTER
2343  003216  005037  001174                  CLR     $TIMES           ;;INITIALIZE NUMBER OF ITERATIONS
2344  003222  005037  001176                  CLR     $ESCAPE          ;;CLEAR THE ESCAPE ON ERROR ADDRESS
2345  003226  112737  000001  001131          MOVB    #1,$ERMAX        ;;ALLOW ONE ERROR PER TEST
2346                          ;;INITIALIZE THE "T-BIT" TRAP VECTOR, THEN LOAD LOCATION "$RTRN", IN
2347                          ;;THE "END-OF-PASS" ($EOP) ROUTINE, WITH A "RTI" OR "RTT".
2348  003234  012737  020006  000014          MOV     #$RTRN,@#TBITVEC ;;SET "T" BIT VECTOR TO $RTRN
2349  003242  012737  000340  000016          MOV     #340,@#TBITVEC+2 ;;LEVEL 7
2350  003250  012737  000002  020006          MOV     #RTI,$RTRN       ;;SET $RTRN TO A RTI
2351  003256  012737  003304  000010          MOV     #65$,@#RESVEC    ;;TRY TO DO A RTT
2352  003264  005046                          CLR     -(SP)            ;;DUMMY PS
2353  003266  012746  003274                  MOV     #64$,-(SP)       ;;AND PC
2354  003272  000006                          RTT                      ;;TRY THE RTT
2355  003274  012737  000006  020006  64$:    MOV     #RTT,$RTRN       ;;RTT IS LEGAL--SET $RTRN TO A RTT
2356  003302  000402                          BR      66$
2357  003304  062706  000010          65$:    ADD     #10,SP           ;;RTT ILLEGAL--CLEAN OFF THE STACK
2358  003310  012737  000012  000010  66$:    MOV     #RESVEC+2,@#RESVEC ;;RESTORE TRAP CATCHER
2359  003316  005037  020014                  CLR     $TBIT            ;;CLEAR "T" BIT SWITCH
2360  003322  012737  003322  001122          MOV     #.,$LPADR        ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
2361  003330  012737  003330  001124          MOV     #.,$LPERR        ;;SETUP THE ERROR LOOP ADDRESS
2362                          ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
2363                          ;;EQUAL TO A "-1"  SETUP FOR A SOFTWARE SWITCH REGISTER.
2364  003336  013746  000004                  MOV     @#ERRVEC,-(SP)   ;;SAVE ERROR VECTOR
2365  003342  012737  003376  000004          MOV     #67$,@#ERRVEC    ;;SET UP ERROR VECTOR
2366  003350  012737  177570  001154          MOV     #DSWR,SWR        ;;SETUP FOR A HARDWARE SWICH REGISTER
2367  003356  012737  177570  001156          MOV     #DDISP,DISPLAY   ;;AND A HARDWARE DISPLAY REGISTER
2368  003364  022777  177777  175562          CMP     #-1,@SWR         ;;TRY TO REFERENCE HARDWARE SWR
2369  003372  001012                          BNE     69$              ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
2370                                                                   ;;AND  THE HARDWARE SWR IS NOT = -1
2371  003374  000403                          BR      68$              ;;BRANCH IF NO TIMEOUT
2372  003376  012716  003404          67$:    MOV     #68$,(SP)        ;;SET UP FOR TRAP RETURN
2373  003402  000002                          RTI
2374  003404  012737  000176  001154  68$:    MOV     #SWREG,SWR       ;;POINT TO SOFTWARE SWR
2375  003412  012737  000174  001156          MOV     #DISPREG,DISPLAY
2376  003420  012637  000004          69$:    MOV     (SP)+,@#ERRVEC   ;;RESTORE ERROR VECTOR
2377
2378  003424  005037  001216                  CLR     $PASS            ;;CLEAR PASS COUNT
2379  003430  132737  000200  001231          BITB    #APTSIZE,$ENVM   ;;TEST USER SIZE UNDER APT
2380  003436  001403                          BEQ     70$              ;;YES,USE NON-APT SWITCH
2381  003440  012737  001232  001154          MOV     #$SWREG,SWR      ;;NO,USE APT SWITCH REGISTER
2382  003446                          70$:
2383  003446  012737  000240  000032          MOV     #240,@#EMTVEC+2   ;CHANGE EMT PRIORITY TO 5
2384  003454  012737  000240  000036          MOV     #240,@#TRAPVEC+2  ;CHANGE TRAP PRIORITY TO 5
2385  003462  005227  177777                  INC     #-1              ;FIRST START ?
2386  003466  001010                          BNE     1$               ;BR IF NOT
2387  003470  104401  040074                  TYPE    ,TITLE           ;NAME AND MANDEC NUMBER
2388  003474  122737  000077  000041          CMPB    #77,@#41         ;LOAD FROM RM03/RM02
2389  003502  001002                          BNE     1$               ;BRANCH IF NOT
2390  003504  104401  040155                  TYPE    ,LOADRV          ;CHANGE PACK MESSAGE
2391  003510  004737  017076          1$:     JSR     PC,$TKINT        ;TURN ON THE KEYBOARD INTERRUPT
```

```
2392                                    .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
2393  003514  005737  000042                    TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
2394  003520  001012                            BNE     71$             ;;BRANCH IF YES
2395  003522  123727  001230  000001            CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
2396  003530  001406                            BEQ     71$             ;;BRANCH IF YES
2397  003532  023727  001154  000176            CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
2398  003540  001005                            BNE     72$             ;;BRANCH IF NO
2399  003542  104406                            GTSWR                   ;;GET SOFT-SWR SETTINGS
2400  003544  000403                            BR      72$
2401  003546  112737  000001  001150  71$:     MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
2402  003554                            72$:
2403  003554  013737  001274  023526            MOV     $RMADR,RMADR    ;RH11 ADDRESS
2404  003562  013737  001276  023530            MOV     $RMVEC,RMVEC    ;RH11 VECTOR ADDRESS
2405  003570  012705  001776            2$:     MOV     #ASNLST,R5      ;START OF AREA TO CLEAR
2406  003574  005025                    3$:     CLR     (R5)+
2407  003576  022705  002560                    CMP     #INDST,R5       ;LOOK FOR END OF CLEAR AREA
2408  003602  001374                            BNE     3$              ;BR IF NOT FINISHED
2409  003604  012706  001100                    MOV     #STACK,SP       ;SETUP THE STACK POINTER
2410  003610  005037  177776                    CLR     PS              ;CLEAR THE PROCESSOR STATUS WORD
2411  003614  013737  001316  001350            MOV     HZ,SIXTEE       ;1/60 TH OR 1/50 TH SECOND COUNTER VALUE
2412  003622  005037  001342                    CLR     HOUR            ;CLEAR THE HOUR'S COUNTER
2413  003626  005037  001344                    CLR     MINUTE          ;CLEAR THE MINUTE'S COUNTER
2414  003632  005037  001346                    CLR     SECOND          ;CLEAR THE SECOND'S COUNTER
2415  003636  005037  001334                    CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
2416
2417                                    ;ROUTINE TO DETERMINE BUFFER AREA SIZE
2418
2419  003642  005227  177777            SIZMEM: INC     #-1             ;SEE IF TIME TO SIZE MEMORY
2420  003646  001002                            BNE     1$              ;BR IF NOT
2421  003650  004737  031674                    JSR     PC,$SIZE        ;SEE HOW MUCH MEMORY ON SYSTEM
2422  003654  013737  031770  001330    1$:     MOV     $LSTAD,LSTAD    ;SAVE THE LAST ADDRESS
2423  003662  023727  001330  160000            CMP     LSTAD,#160000   ;OVER 28K ?
2424  003670  101403                            BLOS    2$              ;NO, THEN DON'T SET THE NEW LIMIT
2425  003672  012737  160000  001330            MOV     #160000,LSTAD   ;SET NEW LIMIT
2426  003700  162737  005670  001330    2$:     SUB     #1500.*2,LSTAD  ;SAVE XXDP LOADER AND ABSOLUTE LOADER
2427
2428                                    ;       SET UP THE OTHER SYSTEM DEVICES THAT
2429                                    ;       THE PROGRAM WILL USE
2430
2431  003706  004737  015354            SETVEC: JSR     PC,CKCLK        ;START THE CLOCK
2432  003712  012737  177777  023466            MOV     #-1,SAVEFG      ;SET THE SAVE REGISTERS FLAG
2433
2434                                    ;SETUP IF 'XXDP' OR 'ACT11' OPERATION
2435
2436  003720  005001                    MONTR:  CLR     R1              ;DRIVE #
2437  003722  005002                            CLR     R2              ;AVAIL TABLE INDEX
2438  003724  005003                            CLR     R3              ;DRIVE# X 2
2439  003726  016300  002600            1$:     MOV     BLKADR(R3),R0   ;LOAD DPB ADDRESS
2440  003732  004737  016042                    JSR     PC,CLRDPB       ;CLEAR DPB BLOCK
2441  003736  022322                    2$:     CMP     (R3)+,(R2)+     ;INCREMENT INDEX
2442  003740  005201                            INC     R1              ;NEXT DRIVE
2443  003742  022701  000007                    CMP     #7,R1           ;ALL DRIVE ASSIGN ?
2444  003746  002367                            BGE     1$              ;NO
2445
2446
2447                                    ;       ASSIGN LOGICAL DRIVES TO BE TEST IN THE PASS1 AND PASS 2
```

```
2448                                    ;       THREE WORDS ARE USED IN THE BIT MAPS:
2449                                    ;       ASNLST:   SPECIFIES THE LOGICAL DRIVES ASSIGNED
2450                                    ;       ASSGN1:   SPECIFIES THE LOGICAL DRIVES WILL BE TESTED IN PASS1.
2451                                    ;       ASSGN2:   SPECIFIES THE LOGICAL DRIVES WILL BE TESTED IN PASS2.
2452                                    ;
2453                                    ;       EACH LOGICAL DRIVE HAS A HISTORY FILE LABELED DIRV'Z (Z=0 TO 17)
2454                                    ;       THE LOCATIONS LABELED $SYSNM AND $PHYRD IN THE HISTORY FILE
2455                                    ;       STORE THE SYSTEM NAME (A TO H) AND PHYSICAL DRIVE NUMBER
2456                                    ;       (0 TO 7).
2457                                    ;       THE SUB-SYSTEM ADDRESS AND INTERRUPT VECTOR ARE STOREED
2458                                    ;       IN THE TABLE LABELED "SYSADR:".
2459                                    ;
2460                                    ;       THE LOCATIONS SYSADR AND SYSADR+2 FOR SUB-SYSTEM A,SYSADR+4
2461                                    ;       AND SYSADR+6 FOR SUBSYSTEM B , ETC,THE FIRST WORD
2462                                    ;       IS THE SUB SYSTEM ADDRESS WHILE THE SECOND WORD IS THE VECTOR
2463                                    ;
2464                                    ;       THE LOGICAL DRIVES ARE ASSIGNED FROM CONSOLE KEYBOARD.
2465
2466
2467
2468
2469
2470  003750  012700  001776          MOD00:  MOV     #ASNLST,R0      ;ADDRESS OF 1ST BIT MAPS IN R0
2471  003754  012701  002560                  MOV     #INDST,R1       ;LAST ADDRESS TO CLEAR
2472  003760  005020                  1$:     CLR     (R0)+           ;CLEAR CURRENT POINTED ADDRESS
2473  003762  020100                          CMP     R1,R0           ;ALL DONE ?
2474  003764  101375                          BHI     1$              ;NO,THEN BRANCH BACK
2475  003766  012737  000101  001272          MOV     #'A,$CDW2       ;TEMP STORATE OF SYS NAME
2476
2477
2478  003774  005037  001222          MOD21:  CLR     DRIVE           ;TEM STORAGE OF PHYSICAL DRIVE BIT MAP
2479  004000  104401  001205                  TYPE    ,$CRLF
2480  004004  104401  035506                  TYPE    ,MESG1          ;SUB-SYSTEM
2481  004010  104401  001272                  TYPE    ,$CDW2          ;A TO H (ONE OF THEM)
2482  004014  104401  036604                  TYPE    ,LINSP
2483  004020  104401  035517                  TYPE    ,MESG2          ;DRIVES
2484  004024  104401  036664                  TYPE    ,COLON          ;COLON
2485  004030  104411                          RDLIN                   ;READ IN THE DRIVE NUMBERS
2486  004032  012601                          MOV     (SP)+,R1        ;GET THE INPUT LINE ADDRESS
2487  004034  105711                  1$:     TSTB    (R1)            ;END OF STRING ?
2488  004036  001415                          BEQ     2$              ;YES
2489  004040  004537  016532                  JSR     R5,CK.OCT       ;CHECK THE DIGIT MUST 0 TO 7,RETURN VALUE IN R2
2490  004044  000753                          BR      MOD21           ;INCORRECT DRIVE NUMBER,ENTER AGAIN
2491  004046  156237  023514  001222          BISB    ATABIT(R2),DRIVE  ;SET THE ASSIGNED PHYSICAL DRIVE BIT
2492                                                                  ;R2 THE DRIVE NUMBER FROM CK.OCT RT.
2493  004054  005201                          INC     R1              ;NEXT BYTE OF INPUT LINE
2494  004056  105711                          TSTB    (R1)            ;END OF STRING ?
2495  004060  001404                          BEQ     2$              ;YES
2496  004062  122721  000054                  CMPB    #',,(R1)+       ;MUST BE A ,
2497  004066  001342                          BNE     MOD21           ;ENTER AGAIN  IF NOT
2498  004070  000761                          BR      1$              ;LOCATE NEXT DRIVE
2499  004072  005737  001332          2$:     TST     CHGADR          ;START AT 200 ?
2500  004076  003036                          BGT     MOD22           ;BRANCH IF SO
2501  004100  013701  001272                  MOV     $CDW2,R1        ;SYS NAME ASCII FROM A TO H
2502  004104  042701  177760                  BIC     #177760,R1      ;LEFT ONLY 4 BITS
2503  004110  005301                          DEC     R1              ;ADJUST INDEX VALUE
```

```
2504  004112  006301                          ASL     R1              ;2 WORD INDEX VALUE
2505  004114  006301                          ASL     R1              ;
2506  004116  062701  002004                  ADD     #SYSADR,R1      ;SYS ADDRESS TABLE ADDRESS
2507  004122  010137  001270                  MOV     R1,$CDW1        ; SYS ADDRESS TABLE'S ENTRY TO CDW1
2508  004126  000400                  3$:     BR      MOD23           ;
2509
2510  004130  005737  001222          MOD23:  TST     DRIVE           ;CHECK IF ANY PHYSICAL DRIVE(S) ASSIGNED
2511  004134  001416                          BEQ     2$              ;BRANCH IF NONE
2512  004136  013700  001270          1$:     MOV     $CDW1,R0        ;SYS ADDRESS TABLE ENTRY
2513  004142  011037  001274                  MOV     (R0),$RMADR     ; SYS ADDRESS
2514  004146  016037  000002  001276          MOV     2(R0),$RMVEC    ; SYS VECTOR
2515  004154  004737  031772                  JSR     PC,BUSADR       ;CHECK THE ADDRESS WITH THE OPERATOR
2516  004160  013710  001274                  MOV     $RMADR,(R0)     ;NEW RH11/70 ADDRESS  INTO TABLE
2517  004164  013760  001276  000002          MOV     $RMVEC,2(R0)    ;NEW VECTOR OF RH11/70 INTO TABLE
2518  004172  000407                  2$:     BR      MOD11           ;BRANCH TO NEXT MODULE
2519
2520
2521  004174                          MOD22:
2522  004174  013737  001274  002004  3$:     MOV     $RMADR,SYSADR   ;LOAD THE SYSTEM ADDRESS TABLE
2523  004202  013737  001276  002006          MOV     $RMVEC,SYSADR+2 ;LOAD THE VECTOR
2524  004210  000400                  4$:     BR      MOD11
2525
2526
2527
2528
2529                                  ;       $CDW2-- ASCII NAME OF SUB SYSTEM (A TO H)
2530                                  ;       $CDW1--ENTRY TO THE SYS ADDRESS TABLE
2531                                  ;       DRIVE--PHYSICAL DRIVES TO BE ASSIGNED
2532                                  ;       THIS SECTION OF CODING USES THE ABOVE PARAMETERS
2533                                  ;       TO SET UP THE BIT MAP OF ASNLST,ASSGN1,ASSGN2
2534                                  ;
2535                                  ;
2536
2537
2538
2539  004212  005737  001222          MOD11:  TST     DRIVE   ;ANY DRIVE ASSIGN ?
2540  004216  001002                          BNE     MOD30           ;BRANCH IF ANY
2541  004220  000137  004374                  JMP     MOD12           ;BRANCH, IF NONE
2542  004224  022737  177777  001776  MOD30:  CMP     #-1,ASNLST      ;ALL 16 LOGICAL DRIVES HAS BEEN ASSIGNED ?
2543  004232  001457                          BEQ     5$              ;YES, THEN DON'T ASSIGN NEW DRIVES
2544  004234  013700  001776                  MOV     ASNLST,R0       ;FOUND THE AVAILABLE LOGICAL DRIVE LOCATION
2545  004240  012701  000001                  MOV     #1,R1           ;START FROM LOGICAL DRIVE 0
2546  004244  005002                          CLR     R2              ;INDEX VALUE
2547  004246  030100                  1$:     BIT     R1,R0           ;IS THE LOGICAL DRIVE AVAILABLE ?
2548  004250  001406                          BEQ     2$              ;YES
2549  004252  000241                          CLC
2550  004254  006101                          ROL     R1              ;NEXT LOGICAL DRIVE
2551  004256  103445                          BCS     5$              ;BRANCH IF NONE IS AVAILABLE
2552  004260  062702  000002                  ADD     #2,R2           ;INCREMENT INDEX VALUE
2553  004264  000770                          BR      1$              ;LOCATE NEXT LOGICAL DRIVE
2554  004266  016204  002600          2$:     MOV     BLKADR(R2),R4   ;GET THE LOGICAL DRIVE'S HISTORY FILE
2555  004272  113764  001272  000014          MOVB    $CDW2,$SYSNM(R4)  ;LOAD THE ASCII SYS NAME
2556  004300  050137  001776                  BIS     R1,ASNLST               ;SET LOGICAL DRIVE ASSIGN BIT
2557  004304  050137  002000                  BIS     R1,ASSGN1       ;SET PASS 1 BIT
2558  004310  050137  002002                  BIS     R1,ASSGN2       ;SET PASS2 BIT
2559  004314  013700  001222                  MOV     DRIVE,R0        ;LOAD THE ASCII PHYSICAL DRIVE #
```

```
2560  004320  012701  000001          MOV     #1,R1           ;START FROM DRIVE 0
2561  004324  005002                  CLR     R2              ;DRIVE #
2562  004326  030100          3$:     BIT     R1,R0           ;IS THE PHYSICAL DRIVE ASSIGNED
2563  004330  001005                  BNE     4$              ;YES,THEN BRANCH
2564  004332  000241                  CLC                     ;LOCATE THE NEXT DRIVE
2565  004334  006101                  ROL     R1              ;SHIFT ONE BIT LEFT
2566  004336  103415                  BCS     5$              ;BRANCH IF NO MORE DRIVES
2567  004340  005202                  INC     R2              ;PHYSICAL DRIVE NUMBER
2568  004342  000771                  BR      3$              ;LOOPING BACK
2569  004344  110214          4$:     MOVB    R2,(R4)         ;LOAD THE PHYSICAL DRIVE # INTO HISTORY FILE
2570  004346  062702  000060          ADD     #'0,R2          ;ASCII CODE OF DRIVE #
2571  004352  110264  000015          MOVB    R2,$PHYDR(R4)   ;PHYSICAL DRIVE NUMBER
2572  004356  112764  000011  000016  MOVB    #HT,$GAP(R4)    ;MAKE UP FOR SCORE TYPE
2573  004364  040137  001222          BIC     R1,DRIVE        ;RESET THE ASSIGNED DRIVE BIT
2574  004370  001315                  BNE     M0030           ;BRANCH IF NOT ALL DONE
2575  004372  000400          5$:     BR      M0D12           ;CHECK OTHER SUB SYSTEM
2576
2577
2578  004374  005737  001332  M0D12:  TST     CHGADR  ;200 START ?
2579  004400  003066                  BGT     6$              ;YES,THEN EXIT
2580  004402  022737  177777  001776  1$:     CMP     #-1,ASNLST      ;FULL HOUSE
2581  004410  001462                  BEQ     6$              ;YES,THEN EXIT
2582  004412  104401  036415          2$:     TYPE    ,MESG5          ;WILL TEST
2583  004416  104401  035517                  TYPE    ,MESG2          ;DRIVES
2584  004422  005003                  CLR     R3              ;INDEX TO LOAGICAL DRIVE HISTORY FILE
2585  004424  012704  000001          MOV     #1,R4           ;BIT MAP OF ASNLST
2586  004430  030437  001776          3$:     BIT     R4,ASNLST       ;ASSIGNED LOGICAL DRIVE ?
2587  004434  001417                  BEQ     5$              ;NO
2588  004436  016305  002600          MOV     BLKADR(R3),R5   ;LOAD THE HISTORY FILE ADDRESS
2589  004442  126537  000014  001272  CMPB    $SYSNM(R5),$CDW2  ;ON THE SAME SYSTEM ?
2590  004450  001003                  BNE     4$              ;NO
2591  004452  111546                  MOVB    (R5),-(SP)      ;TYPE THE PHYSICAL DRIVE #
2592  004454  104403                  TYPOS
2593  004456     002                  .BYTE   2
2594  004457     000                  .BYTE   0
2595  004460  062703  000002          4$:     ADD     #2,R3           ;INCREMENT TO NEXT LOGICAL DRIVE
2596  004464  000241                  CLC
2597  004466  006104                  ROL     R4              ;BIT MAP OF THE NEXT LOGICAL DRIVE
2598  004470  103401                  BCS     5$              ;BRANCH IF ALL LOGICAL DRIVE'S CHECKED
2599  004472  000756                  BR      3$              ;BRANCH BACK
2600  004474  104401  036604          5$:     TYPE    ,LINSP
2601  004500  104401  036430                  TYPE    ,MESG6          ;ON
2602  004504  104401  035506                  TYPE    ,MESG1          ;SUB SYSTEM
2603  004510  104401  001272                  TYPE    ,$CDW2          ; A TO H
2604
2605  004514  005237  001272                  INC     $CDW2           ;CHECK NEXT SUB SYSTEM
2606  004520  122737  000110  001272          CMPB    #'H,$CDW2               ;ALL EIGHT SUB SYSTEMS CHECKED?
2607  004526  103413                  BLO     6$              ;YES
2608  004530  104401  001205                  TYPE    ,$CRLF
2609  004534  104401  036434                  TYPE    ,MESG7          ;ASK FOR OTHER SUB SYSTEM
2610  004540  104411                  RDLIN                   ;READ IN A LINE
2611  004542  012605                  MOV     (SP)+,R5        ;CHECK THE FIRST CHARACTER
2612  004544  122715  000131          CMPB    #'Y,(R5)        ;ENTER Y ?
2613  004550  001002                  BNE     6$              ;BRANCH IF NOT
2614  004552  000137  003774          JMP     M0D2i           ;SET UP OTHER SUB SYSTEM
2615  004556  005737  001332  6$:     TST     CHGADR          ;START AT 220
```

```
2616  004562  001402                        BEQ     7$              ;YES,EXECUTE PASS2 ONLY
2617  004564  000137  004600                JMP     XPASS1          ;TO PASS1
2618  004570  005037  002000        7$:     CLR     ASSGN1          ;CLEAR THE PASS1 BIT MAP
2619  004574  000137  007106                JMP     XPASS2          ;BRANCH TO PASS2
2620
2621
2622
2623
2624
2625                                ;       PASS ONE , THE VARIAVLES ARE ASSIGNED AS FOLLOWS:
2626                                ;       $CDW1    :  ADDRESS OF THE CURRENT LOGICAL DRIVE HISTORY FILE
2627                                ;       $CDW2    :  SYSTEM NAME A THROUGH H
2628                                ;       $DEVM    :   CURRENT LOGICAL DRIVE #
2629                                ;       ASNLST   :       ASSIGNED LOGICAL DRIVES
2630                                ;       ASSGN1   :       ASSIGNED LOGICAL DRIVES IN THIS PASS
2631                                ;       R0       :       ADDRESS OF DPB BLOCK FEEDED INTO DRIVER-HANDLER
2632                                ;       R1       :       TEMP STORAGE OF ADDRESS OF THE LOGICAL BLOCK
2633
2634
2635
2636
2637  004600  005737  002000        XPASS1: TST     ASSGN1          ;ANY DRIVE ASSIGNED FOR PASS1
2638  004604  001002                        BNE     1$              ;BRANCH IF THEY ARE
2639  004606  000137  005464                JMP     ENDX1           ;END OF PASS 1
2640  004612  005037  001266        1$:     CLR     $DEVM           ;INDEX OF LOGICAL BLOCK
2641  004616  013737  002600  001270        MOV     BLKADR,$CDW1    ;ADDRESS OF LOGICAL BLOCK 0
2642  004624  112737  000101  001272        MOVB    #'A,$CDW2       ;SYS NAME STARTS FROM A
2643  004632  012777  013560  016670        MOV     #IDLEX,@RMVEC   ;RESET ALL INTERRUPT VERTOR
2644  004640  005077  016666                CLR     @RMVEC+2        ;CLEAR THE INTERRUPT LEVEL
2645  004644  013737  002004  023526        MOV     SYSADR,RMADR    ;LOAD SYS-TEM A INTO
2646  004652  013737  002006  023530        MOV     SYSADR+2,RMVEC  ;DIRVER-HANDLER
2647  004660  013701  001270                MOV     $CDW1,R1        ;R1=ADDRESS OF LOGICAL BLOCK 1
2648  004664  012777  013560  016636        MOV     #IDLEX,@RMVEC   ;RESET ALL INTERRUPT VECTOR
2649  004672  005077  016634                CLR     @RMVEC+2        ;CLEAR THE INTERRUPT LEVEL
2650  004676  104401  001205                TYPE    ,$CRLF
2651  004702  104401  035531                TYPE    ,MESG3
2652  004706  104401  001205                TYPE    ,$CRLF
2653  004712  104401  035557                TYPE    ,MESG4
2654  004716  104401  001272                TYPE    ,$CDW2
2655  004722  111137  001222                MOVB    (R1),DRIVE      ;LOAD THE PHYSICAL DRIVE #
2656  004726  111146                        MOVB    (R1),-(SP)      ;TYPE THE DRIVE #
2657  004730  104403                        TYPOS
2658  004732     002                        .BYTE   2
2659  004733     000                        .BYTE   0
2660  004734  104401  036604                TYPE    ,LINSP
2661  004740  104401  035605                TYPE    ,MESG8
2662  004744  104401  001205                TYPE    ,$CRLF
2663  004750  104401  035620                TYPE    ,MESG9
2664  004754  104401  036604                TYPE    ,LINSP
2665
2666
2667
2668
2669  004760  104411                2$:     RDLIN                   ;CHECK IF O.P. READY
2670  004762  012602                        MOV     (SP)+,R2        ;LOCATE THE INPUT LINE
2671  004764  122722  000122                CMPB    #'R,(R2)+       ;FIRST CHARACTER IS A "R" ?
```

```
2672  004770  001373                        BNE     2$              ;BRANCH IF NOT
2673  004772  105712                        TSTB    (R2)            ;FOLLOW BY <CR> ?
2674  004774  001371                        BNE     2$              ;BRANCH IF NOT
2675  004776  004737  023544                JSR     PC,RMINIT       ;INITIALIZE THE SUB SYSTEM
2676  005002  012737  177777  023466        MOV     #-1,SAVEFG      ;SAVE ALL RH/RM REGISTERS
2677  005010  012737  177777  023470        MOV     #-1,SEEKFG      ;DON'T DO IMPLY SEEK
2678  005016  013700  001222                MOV     DRIVE,R0        ;R0=PHYSICAL DRIVE # OF THE SUB SYSTEM
2679  005022  105760  023400                TSTB    DRVSTA(R0)      ;DIRVE EXIST  AND ON LINE ?
2680  005026  003452                        BLE     5$              ;BRANCH IF NOT
2681  005030  105760  023410                TSTB    DRVTYP(R0)      ;DRIVE IS AN RM03 ?
2682  005034  003447                        BLE     5$              ;BRANCH IF NOT
2683                                                                ;SET UP TO RETRIEVE THE BAD SPOT FILE
2684                                                                ;FROM THE LAST TRACK
2685  005036  012700  037760                MOV     #FMTDPB,R0      ;DPB ADDRESS
2686  005042  113710  001222                MOVB    DRIVE,(R0)      ;LOAD THE DRIVE NUMBER
2687  005046  012760  040074  000006        MOV     #ENDPGM,$BUF(R0)  ;LOAD BUFFER ADDRESS
2688  005054  012760  040000  000014        MOV     #RM.REG,14(R0)    ;AREA TO SAVE ALL RH/RM03 REG'S
2689  005062  012760  177400  000004        MOV     #-256.,$WRDM(R0)  ;WORD COUNT (NEG)
2690  005070  013760  001372  000012        MOV     CYLIMT,$CYL(R0) ;CYLINDER 822.
2691  005076  113760  001370  000011        MOVB    TRKLMT,$TRK(R0) ;TRACK ADDRESS 4.
2692  005104  105060  000010                CLRB    $SEC(R0)        ;SEC 0
2693  005110  112760  000171  000002        MOVB    #RDDAT,$COMND(R0) ;READ DATA COMMAND
2694
2695  005116  004037  024272        3$:     JSR     R0,RM03         ;CALL THE DRIVER-HANDLER
2696  005122  037760                        FMTDPB                  ;PARAMETER ADDRESS
2697  005124  000774                        BR      3$              ;LOOPING IF QUEUE IS NOT SUCCESSFUL
2698  005126  005737  037776        4$:     TST     FMTDPB+16       ;COMMAND DONE ?
2699  005132  001775                        BEQ     4$              ;BRANCH IF NOT
2700  005134  100013                        BPL     6$              ;BRANCH IF DONE, WITHOUT ERROR
2701  005136  062737  000002  037770        ADD     #2,FMTDPB+10    ;TRY NEXT SECTOR (0,2,4,6,8)
2702  005144  122737  000010  037770        CMPB    #8.,FMTDPB+10   ;ALL FIVE SECTORS CHECKED ?
2703  005152  101361                        BHI     3$              ;NO,THEN TRY AGAIN
2704
2705  005154  104401  035657        5$:     TYPE    ,MESG10         ;DRIVE IS NOT READY
2706  005160  000137  005464                JMP     ENDX1           ;STOP THE TEST
2707
2708
2709                                                                ;BAD SPOT FILE IS RETRIVED,IS STORED
2710                                                                ;FROM ENDPGM+4 TO ENDPGM+256.
2711                                                                ;FIRST WORD CYLINDER #,SECOND WORD
2712                                                                ;TRK AND SEC NUMBERS,FILE IS TERMINATED
2713                                                                ;BY A -1 IN THE CYLINDER NUMBER
2714
2715
2716  005164  012704  040104        6$:     MOV     #ENDPGM+10,R4   ;R4 ADDRESS OF BAD SPOT FILE
2717  005170  022714  177777        7$:     CMP     #-1,(R4)        ;END OF BAD SPOT FILE ?
2718  005174  001411                        BEQ     8$              ;YES
2719  005176  022704  041074                CMP     #ENDPGM+1000,R4 ;END OF BAD SPOT FILE ?
2720  005202  101406                        BLOS    8$              ;BRANCH IF IT IS
2721  005204  004537  005316                JSR     R5,SPOTX        ;CHECK THE CYLINDER POINTED BY (R4),
2722  005210  000422                        BR      10$             ;BRANCH IF BAD SPOT IN THE TEST ZONES
2723  005212  062704  000004                ADD     #4,R4           ;NEXT BAD SPOT ADDRESS
2724  005216  000764                        BR      7$              ;LOOPING BACK
2725
2726  005220  032777  000200  173726  8$:   BIT     #BIT7,@SWR      ;SWITCH 7 SET ?
2727  005226  001404                        BEQ     9$              ;BRANCH IF NOT SET
```

```
2728  005230  012700  037760                  MOV     #FMTDPB,R0      ;DPB ADDRESS
2729  005234  004737  015022                  JSR     PC,PRTBAD       ;PRINT THE BAD SPOT FILE
2730  005240  105760  000010          9$:     TSTB    $SEC(R0)        ;SECTOR 10 HAS BEEN READ ?
2731  005244  001022                          BNE     12$             ;BRANCH IF SC
2732  005246  112760  000012  000010          MOVB    #10.,$SEC(R0)   ;READ SECTOR 10
2733  005254  000720                          BR      3$              ;LOOPING BACK
2734  005256  104401  035723          10$:    TYPE    ,MESG11         ;PACK NOT ACCEPTABLE
2735  005262  104401  001205                  TYPE    ,$CRLF
2736  005266  032777  000200  173660          BIT     #BIT7,@SWR      ;SWITCH 7 SET ?
2737  005274  001404                          BEQ     11$             ;BRANCH IF NOT SET
2738  005276  012700  037760                  MOV     #FMTDPB,R0      ;DPB ADDRESS
2739  005302  004737  015022                  JSR     PC,PRTBAD       ;TYPE THE BAD SPOT FILE
2740  005306  000137  004600          11$:    JMP     XPASS1          ;RESTART PASS 1
2741  005312  000137  005502          12$:    JMP     TST1            ;PROCEED TO TEST 1
2742
2743
2744                                  ;       SUBROUTINE SPOTX
2745                                  ;       CHECK THE CYLINDER POINTED BY (R4) IS IN THE TESTING ZONES
2746                                  ;       (0-15,128-143,256-271,384-399,512-527,640-655,768-783
2747                                  ;       112-127,240-255,368-383,496-511,624-639,752-767,
2748                                  ;       17-32,145-160,273-288,401-416,529-544,657-672,785-800
2749                                  ;       AND 620 )
2750                                  ;       CALLING SEQ:
2751                                  ;       JSR     R5,SPOTX        ;R4=POINT TO CYLINDER NUMBER
2752                                  ;       RET1                    ;ERROR RET
2753                                  ;       RET2                    ;NORMAL RET1
2754
2755
2756  005316  010146                  SPOTX:  MOV     R1,-(SP)        ;SAVE R1 THROUGH R3
2757  005320  010246                          MOV     R2,-(SP)
2758  005322  010346                          MOV     R3,-(SP)
2759
2760  005324  005003                          CLR     R3              ;ERROR FLAG
2761  005326  005046                          CLR     -(SP)           ;DUMMY PAIR
2762  005330  005046                          CLR     -(SP)           ;DUMMY PAIR
2763  005332  005046                          CLR     -(SP)           ;ZONE STARTING ADDRESS
2764  005334  012746  000007                  MOV     #7,-(SP)        ;SEGMENT NUMBER
2765  005340  012746  000021                  MOV     #17.,-(SP)      ;ZONE STARTING ADDRESS
2766  005344  012746  000007                  MOV     #7,-(SP)        ;SEGMENT NUMBER
2767  005350  012701  000160                  MOV     #112.,R1        ;R1=ZONE STARTING ADDRESS
2768  005354  012702  000006                  MOV     #6,R2           ;R2=SEGMENT NUMBER
2769  005360  021401                  1$:     CMP     (R4),R1         ;CYL IN THE ZONE ?
2770  005362  103410                          BLO     3$              ;BRANCH IF NOT
2771  005364  062701  000017                  ADD     #15.,R1         ;CHECK WITH THE UPPER BOND
2772  005370  021401                          CMP     (R4),R1         ;CYL IN THE ZONE ?
2773  005372  101002                          BHI     2$              ;BRANCH IF NOT
2774  005374  052703  000002                  BIS     #BIT1,R3        ;SET THE ERROR FLAG
2775
2776  005400  162701  000017          2$:     SUB     #15.,R1         ;RESTORE TO THE LOWER BOND
2777  005404  005302                  3$:     DEC     R2              ;DECREMENT THE SEGMENT COUNT
2778  005406  001403                          BEQ     4$              ;ALL SEGMENT CHECKED ?
2779  005410  062701  000200                  ADD     #128.,R1        ;ADJUST ZONE STARTING ADDRESS
2780  005414  000761                          BR      1$              ;LOOPING BACK UNTIL ALL SEGMENTS ARE CHECKED
2781
2782  005416  012602                  4$:     MOV     (SP)+,R2        ;POP THE NEXT SET OF ZONE PARAMETERS
2783  005420  012601                          MOV     (SP)+,R1
```

```
2784  005422  005702                  TST     R2              ;DUMMY PAIR
2785  005424  001355                  BNE     1$              ;BRANCH IF NOT
2786  005426  005701                  TST     R1              ;DUMMY PAIR
2787  005430  001353                  BNE     1$              ;BRANCH IF NOT
2788
2789  005432  021427  001154  5$:     CMP     (R4),#620.      ;ON CYLDER 620 ?
2790  005436  001002                  BNE     6$              ;NO
2791  005440  052703  000002          BIS     #BIT1,R3        ;SET ERROR FLAG
2792  005444  005703          6$:     TST     R3              ;ANY ERROR ?
2793  005446  001002                  BNE     7$              ;YES
2794  005450  062705  000002          ADD     #2,R5           ;ADJUST FOR NORMAL RETURN
2795  005454  012603          7$:     MOV     (SP)+,R3        ;RESTORE R3 THROUGH R1
2796  005456  012602                  MOV     (SP)+,R2
2797  005460  012601                  MOV     (SP)+,R1
2798  005462  000205                  RTS     R5              ;EXIT
2799
2800
2801  005464  104401  001205  ENDX1:  TYPE    ,$CRLF  ;
2802  005470  104401  037107          TYPE    ,MESG21         ;DRIVE NOT ONLINE OR NOT ASSIGNED
2803  005474  000000                  HALT                    ;TEMP HALT FOR DEBUG
2804  005476  000137  000200          JMP     @#200           ;RESTART IF DESIRED
2805                                ;
2806                                ;
2807                                ;
2808                                ;THE FOLLOWING CODING FOR TEST 1 THROUGH TEST 4
2809                                ;PARAMETER IN TST 1
2810                                ;$CDW1  :       ADDRESS OF LOGICAL DRIVE BLOCK
2811                                ;DRIVE  :       PHYSICAL DRIVE #
2812                                ;$DEVM  :       LOGICAL DRIVE # 0-17
2813                                ;ASSGN1 :       ASSIGN LOGICAL DRIVE BIT MAP
2814                                ;ASNLST :       ASSIGNED LOGICAL DRIVE MAP INDICATOR
2815                                ;$CDW2: SYS-NAME
2816                                ;
2817                                ;$CDW2,DRIVE ARE ONLY CHANGED IN TST1 DURING PASS 1
2818                                ;$DEVM,ASSGN1,$CDW1 ARE CHANGED BY THE TEST 4
2819
2820
2821
2822
2823
2824
2825                                ;TST 1:DIRECT OPERATOR TO MOUNT AND LOAD PACKS
2826                                ;
2827                                ;
2828
2829
2830                                ;;*****************************************************************
2831  005502  000004  TST1:   SCOPE
2832  005504  012737  000001  001174          MOV     #1,$TIMES       ;;DO 1 ITERATION
2833  005512  012737  000001  001336          MOV     #1,TSTNM        ;LOAD THE TEST NUMBER
2834  005520  012706  001100                  MOV     #STACK,SP       ;INITIALIZE THE STACK POINT
2835  005524  023737  001270  002600          CMP     $CDW1,BLKADR    ;LOGICAL DRIVE 0
2836  005532  001524                  BEQ     3$              ;THEN EXIT
2837  005534  013701  001270          MOV     $CDW1,R1        ;R1=LOGICAL DRIVE BLOCK ADDRESS
2838  005540  126137  000014  001272  CMPB    $SYSNM(R1),$CDW2 ;STILL ON THE SAME SYSTEM ?
2839  005546  001410                  BEQ     1$              ;THEN .EXIT
```

```
2840  005550  116137  000014  001272          MOVB    $SYSNM(R1),$CDW2 ;LOAD THE NEW SUB SYSTEM NAME
2841  005556  012777  013560  015744          MOV     #IDLEX,@RMVEC   ;RESET THE INTERRUPT VECTOR
2842  005564  005077  015742                  CLR     @RMVEC+2        ;CLEAR THE INTERRUPT LEVEL
2843  005570  104401  001205          1$:     TYPE    ,$CRLF
2844  005574  104401  035531                  TYPE    ,MESG3          ;** STARTING PASS 1 **
2845  005600  104401  001205                  TYPE    ,$CRLF
2846  005604  104401  036430                  TYPE    ,MESG6          ;ON--
2847  005610  104401  035506                  TYPE    ,MESG1          ;SUB-SYSTEM
2848  005614  104401  001272                  TYPE    ,$CDW2          ;A TO H
2849  005620  104401  036604                  TYPE    ,LINSP
2850  005624  111137  001222                  MOVB    (R1),DRIVE      ;LOAD THE PHYSICAL DRIVE #
2851  005630  104401  001205                  TYPE    ,$CRLF
2852  005634  104401  035557                  TYPE    ,MESG4          ;MOUNT PACK ON THE DRIVE
2853  005640  104401  001272                  TYPE    ,$CDW2          ;SYS-NAME
2854  005644  113746  001222                  MOVB    DRIVE,-(SP)     ;THE PHYSICAL DRIVE #
2855  005650  104403                          TYPOS
2856  005652     002                          .BYTE   2
2857  005653     000                          .BYTE   0
2858  005654  104401  036604                  TYPE    ,LINSP
2859  005660  104401  035605                  TYPE    ,MESG8          ;AND LOAD
2860  005664  104401  001205                  TYPE    ,$CRLF
2861  005670  104401  035620                  TYPE    ,MESG9
2862  005674  104411                  2$:     RDLIN
2863  005676  012605                          MOV     (SP)+,R5                ;LOCATE THE READIN LINE
2864  005700  122725  000122                  CMPB    #'R,(R5)+               ;CHECK IF O.P. READY ?
2865  005704  001373                          BNE     2$              ;IF NOT
2866  005706  105715                          TSTB    (R5)            ;NOT CORRECT INPUT LINE FORMAT
2867  005710  001371                          BNE     2$              ;BRANCH IF NOT
2868  005712  113701  001272                  MOVB    $CDW2,R1        ;LOCATE THE SYSTEM ADDRESS TABLE
2869  005716  042701  177760                  BIC     #177760,R1      ;LEFT ON 4 BITS
2870  005722  005301                          DEC     R1              ;ADDJUST THE INDEX VALUE
2871  005724  006301                          ASL     R1              ;FOUR WORD INDEX VALUE
2872  005726  006301                          ASL     R1
2873  005730  016137  002004  023526          MOV     SYSADR(R1),RMADR  ;LOAD THE SYSTEM ADDRESS
2874  005736  016137  002006  023530          MOV     SYSADR+2(R1),RMVEC ;LOAD THE SYSTEM INTERRUPT VECTOR
2875  005744  004737  023544                  JSR     PC,RMINIT       ;INITIALIZE THE SYSTEM
2876  005750  012737  177777  023466          MOV     #-1,SAVEFG      ;SAVE ALL RH/RM REGISTER
2877  005756  012737  177777  023470          MOV     #-1,SEEKFG      ;DON'T DO ANY IMPLY SEEK
2878  005764  013700  001222                  MOV     DRIVE,R0        ;R0=PHYSICAL DRIVE #
2879  005770  105760  023400                  TSTB    DRVSTA(R0)      ;ON-LINE ?
2880  005774  003404                          BLE     4$              ;BRANCH IF NOT
2881  005776  105760  023410                  TSTB    DRVTYP(R0)      ;AN RM03
2882  006002  003401                          BLE     4$              ;BRANCH IF NOT AN RM03
2883  006004  000403                  3$:     BR      TST2            ;ALL DONE PROCEED TO TEST 2
2884  006006  104401  035657          4$:     TYPE    ,MESG10         ;DRIVE NOT READY
2885  006012  000633                          BR      TST1            ;TRY AGAIN
2886
2887
2888                                  ;TEST 2
2889                                  ;BASIC READ AND WRITE TEST
2890                                  ;ALL LOGICAL DRIVE ACCESS CYLINDER 620
2891                                  ;AND SECTOR ADDRESS IS COORESPONDING TO THE LOGICAL DRIVE #
2892                                  ;EACH LOGICAL DRIVE PERFORM WRITE AND WRITE CHECK ON ALL FIVE SURFACES(TRK0 - TRK4)
2893
2894
2895
```

```
2896                                    ;;*****************************************************************
2897  006014  000004                    TST2:   SCOPE
2898  006016  012737  000001  001174            MOV     #1,$TIMES       ;;DO 1 ITERATION
2899  006024  012737  000002  001336            MOV     #2,TSTNM        ;LOAD TEST NUMBER
2900  006032  012706  001100                    MOV     #STACK,SP       ;INITIAL THE STACK POINTER
2901  006036  012700  037760                    MOV     #FMTDPB,R0      ;DPB BLOCK ADDRESS
2902  006042  013701  001270                    MOV     $CDW1,R1        ;ADDRESS OF THE LOGICAL DRIVE BLOCK
2903  006046  113710  001222                    MOVB    DRIVE,(R0)      ;PHYSICAL DRIVE #
2904  006052  013760  001266  000010            MOV     $DEVM,$SEC(R0)  ;LOAD THE SECTOR #,FROM THE LOGICAL DRIVE NUMBER.
2905  006060  012760  001154  000012            MOV     #620.,$CYL(R0)  ;LOAD CYLINDER NUMBER
2906  006066  012760  177400  000004            MOV     #-256.,$WRDM(R0) ;LOAD NEG WORD COUNT
2907  006074  012760  040074  000006            MOV     #ENDPGM,$BUF(R0) ;LOAD BUFFER ADDRESS
2908  006102  012760  040000  000014            MOV     #RM.REG,14(R0)  ;ADDRESS TO SAVE ALL RH/RM03 REG'S
2909  006110  105060  000011                    CLRB    $TRK(R0)        ;START FROM TRACK 0
2910  006114  112760  000161  000002            MOVB    #WRTDAT,$COMND(R0)      ;WRITE DATA COMMAND
2911  006122  004737  015162                    JSR     PC,FILBUF       ;FILL THE BUFFER WITH STANDARD PATTERN
2912  006126  004037  024272            2$:    JSR     R0,RM03         ;CALL THE DRIVER
2913  006132  037760                            FMTDPB
2914  006134  000774                            BR      2$              ;BRANCH IF NOT QUEUE SUCCESSFULLY
2915  006136  005737  037776            3$:    TST     FMTDPB+16
2916  006142  001775                            BEQ     3$              ;BRANCH IF NOT DONE
2917  006144  012700  037760                    MOV     #FMTDPB,R0      ;R0=FMTDPB ADDRESS
2918  006150  004737  013566                    JSR     PC,PROCESS      ;CHECK THE TERMINATION
2919  006154  005760  000016                    TST     16(R0)          ;ERROR FLAG SET ?
2920  006160  100433                            BMI     6$              ;BRANCH IF SO
2921  006162  112760  000151  000002            MOVB    #WCKD,$COMND(R0) ;CHANGE TO THE WRITE CHECK DATA COMMAND
2922  006170  004037  024272            4$:    JSR     R0,RM03         ;CLLL THE DRIVER
2923  006174  037760                            FMTDPB
2924  006176  000774                            BR      4$              ;BRANCH IF NOT QUEU SUCCESSFULLY
2925  006200  005737  037776            5$:    TST     FMTDPB+16       ;DONE ?
2926  006204  001775                            BEQ     5$              ;BRANCH IF NOT DONE
2927  006206  012700  037760                    MOV     #FMTDPB,R0      ;
2928  006212  004737  013566                    JSR     PC,PROCES       ;PROCESS IF ANY ERROR HAPPENS ?
2929  006216  005760  000016                    TST     16(R0)          ;ERROR FLAG SET ?
2930  006222  100412                            BMI     6$              ;BRANCH IF SO
2931  006224  112760  000161  000002            MOVB    #WRTDAT,$COMND(R0) ;RESET TO WRITE DATA COMMAND
2932  006232  105260  000011                    INCB    $TRK(R0)        ;INCREMENT TO THE NEXT TRACK
2933  006236  122760  000005  000011            CMPB    #5.,$TRK(R0)    ;ALL TRACKS DONE ?
2934  006244  101330                            BHI     2$              ;NO
2935  006246  000407                            BR      TST3            ;BRANCH TO NEXT TEST
2936  006250  104401  001205            6$:    TYPE    ,$CRLF
2937  006254  104401  037171                    TYPE    ,HALT1
2938  006260  104401  037152                    TYPE    ,HALTX
2939  006264  000000                            HALT
2940
2941
2942
2943                                    ;TEST 3
2944                                    ;WRITE 7 ZONES FOR WRITE TEST IN PASS 2
2945                                    ;WRITE 6 ZONES FOR READ TEST IN PASS2
2946                                    ;$DEVM  :       LOGICAL DRIVE #
2947                                    ;$CDW1  :       ADDRESS OF LOGICAL DRIVE HISTORY BLOCK FILE
2948                                    ;$CDW2  :       SYS NAME
2949                                    ;DRIVE  :       PHYSICAL DRIVE # OF THIS LOGICAL DRIVE
2950                                    ;PACK   :       ENTRY OF TABLE-D
2951                                    ;CMCNT  :       ZONE COUNT
```

```
                                        ;CMSEC  :       DELTA CYLINDER COUNT
                                        ;R4     :       ENTRY POINTER OF TABLE-D,CANNOT BE DESTORIED.
                                        ;R0     :       ADDRESS OF FMTDPB


                                        ;;*****************************************************************
006266  000004                          TST3:   SCOPE
006270  012737  000001  001174                  MOV     #1,$TIMES       ;;DO 1 ITERATION
006276  012737  000003  001336                  MOV     #3,TSTNM        ;LOAD THE TEST NUMBER
006304  012706  001100                          MOV     #STACK,SP       ;INITIAL THE STACK POINTER
006310  012704  001376                          MOV     #LOGD,R4        ;ADDRESS OF TABLE-D
006314  013700  001266                          MOV     $DEVM,R0        ;LOGICAL DRIVE #
006320  001404                                  BEQ     2$              ;BRANCH IF LOGICAL DRIVE 0
006322  062704  000020                  1$:     ADD     #16.,R4         ;EACH LOGICAL DRIVE TAKES 16 BYTES IN THE TABLE
006326  005300                                  DEC     R0              ;DECREMENT THE DRIVE # COUNT
006330  001374                                  BNE     1$              ;BRANCH ,UNTIL THE ENTRY IS LOCATED
006332  010437  001322                  2$:     MOV     R4,PACK         ;SAVE THE TABLE-D ENTRY IN PACK
                                                                        ;SET UP THE FMTDPB BLOCK
006336  012700  037760                          MOV     #FMTDPB,R0      ;ADDRESS OF FMTDPB
006342  113710  001222                          MOVB    DRIVE,(R0)      ;PHYSICAL DRIVE NUMBER
006346  112760  000161  000002                  MOVB    #WRTDAT,$COMND(R0)      ;WRITE DATA COMMAND
006354  012760  040074  000006                  MOV     #ENDPGM,$BUF(R0) ;BUFFER ADDRESS
006362  012760  040000  000014                  MOV     #RM.REG,14(R0)  ;ADDRESS TO SAVE ALL RH/RM03 REG'S
006370  012760  177400  000004                  MOV     #-256.,$WRDM(R0)        ;NEG WORD COUNT
006376  013701  001266                          MOV     $DEVM,R1        ;LOGICAL DRIVE #
006402  006301                                  ASL     R1              ;WORD INDEX
006404  016104  002702                          MOV     PSEUDO(R1),R4   ;LOAD THE DATA PATTERN
006410  016002  000006                          MOV     $BUF(R0),R2     ;BUFFER ADDRESS IN R2
006414  012703  000400                          MOV     #256.,R3        ;POS WORD COUNT
006420  010422                          3$:     MOV     R4,(R2)+        ;FULL THE BUFFER WITH SIGLE WORD PATTERN
006422  005303                                  DEC     R3              ;DECREMENT THE WORD COUNT
006424  001375                                  BNE     3$              ;BRANCH,UNTIL IT IS FULL
                                        ;
                                        ;
006426  005046                                  CLR     -(SP)           ;DUMY PAIR OF ZONE COUNT
006430  005046                                  CLR     -(SP)           ;DUMY STARTING CYLINDER NUMBER
006432  012746  000006                          MOV     #6,-(SP)        ;ZONE COUNT
006436  012746  000160                          MOV     #112.,-(SP)     ;STARTING CYLINDER NUMBER
006442  012737  000007  001352                  MOV     #7,CMCNT        ;ZONE COUNT
006450  005037  001354                          CLR     CMCYL           ;STARTING CYLINDER NUMBER
006454  012737  000020  001360  4$:     MOV     #16.,CMSEC      ;DELTA CYLINDER NUMBER
006462  012700  037760                          MOV     #FMTDPB,R0      ;R0 DPB ADDRESS
006466  013704  001322                          MOV     PACK,R4         ;R4 ENTRY TO TABLE-D
006472  013760  001354  000012  5$:     MOV     CMCYL,$CYL(R0)  ;STARTING CYLINDER
006500  105060  000011                          CLRB    $TRK(R0)        ;STARTS FROM TRACK 0
006504  111460  000010                  6$:     MOVB    (R4),$SEC(R0)   ;LOAD SECTOR NUMBER FROM TABLE-D
006510  004037  024272                  7$:     JSR     R0,RM03         ;CALL THE DRIVER
006514  037760                                  FMTDPB
006516  000774                                  BR      7$              ;BRANCH IF QUEU IS NOTSUCCESSFUL
006520  005737  037776                  8$:     TST     FMTDPB+16       ;DONE ?
006524  001775                                  BEQ     8$              ;BRANCH IF NOT
006526  012700  037760                          MOV     #FMTDPB,R0
006532  004737  013566                          JSR     PC,PROCES       ;PROCESS TO CHECK IF ANY ERROR
006536  005760  000016                          TST     16(R0)          ;ERROR FLAG SET ?
006542  100453                                  BMI     9$              ;BRANCH IF SO
```

```
3008  006544  062760  000020  000010          ADD     #16.,$SEC(R0)   ;WRITE TWO SECTORS ON ONE CYLINDER
3009  006552  122760  000037  000010          CMPB    #31.,$SEC(R0)   ;ALL DONE-TWO SECTORS
3010  006560  103353                          BHIS    7$              ;BRANCH IF NOT
3011  006562  111460  000010                  MOVB    (R4),$SEC(R0)   ;RESTORE SECTOR #
3012  006566  105260  000011                  INCB    $TRK(R0)        ;INCREMENT TO NEXT TRACK
3013  006572  122760  000004  000011          CMPB    #4.,$TRK(R0)    ;LAST TRACK ?
3014  006600  103343                          BHIS    7$              ;NO, THEN BRANCH
3015  006602  105060  000011                  CLRB    $TRK(R0)        ;RESTORE TO TRACK-0
3016  006606  005260  000012                  INC     $CYL(R0)        ;INCREMENT CYLINDER NUMBER
3017  006612  005204                          INC     R4              ;INCREMENT TABLE-D ENTRY
3018  006614  005337  001360                  DEC     CMSEC           ;DECREMENT THE DELTA CYLINDER COUNT
3019  006620  001331                          BNE     6$              ;BRANCH IF NOT END OF THIS BLOCK
3020  006622  062760  000160  000012          ADD     #112.,$CYL(R0)  ;INCREMENTTHE CYLINDER NUMBER TO NEXT ZONE
3021  006630  016037  000012  001354          MOV     $CYL(R0),CMCYL  ;INITAIL THE STARTING CYLINDER IN THE BLOCK
3022  006636  005337  001352                  DEC     CMCNT   ;DECREMENT THE ZONE COUNT
3023  006642  001304                          BNE     4$              ;LOOPING IF NOT END OF ZONE
3024  006644  012637  001354                  MOV     (SP)+,CMCYL     ;LOAD NEW PAIR OF STARTING CYLINDER
3025  006650  012637  001352                  MOV     (SP)+,CMCNT     ;AND ZONE COUNT
3026  006654  005737  001354                  TST     CMCYL           ;NOT END YET ?
3027  006660  001275                          BNE     4$              ;BRANCH IF NOT
3028  006662  005737  001352                  TST     CMCNT           ;BRANCH IF NOT END
3029  006666  001272                          BNE     4$              ;LOOPING BACK
3030
3031  006670  000407                          BR      TST4            ;BRANCH TO THE NEXT TEST
3032  006672  104401  001205          9$:     TYPE    ,$CRLF
3033  006676  104401  037240                  TYPE    ,HALT2
3034  006702  104401  037152                  TYPE    ,HALTX
3035  006706  000000                          HALT
3036
3037                                  ;TEST 4
3038                                  ;UPDATE THE PARAMETERS,$CDW1,$DEVM,ASSGN1
3039                                  ;DIRECT THE OPERATOR TO DISMOUNT PACK AND LOAD TO OTHER DRIVE
3040                                  ;
3041                                  ;$CDW2,DRIVE ARE CHANGED BY TEST ONE ONLY AFTER THE TEST LOOPING TO TEST1
3042
3043
3044                                  ;;*****************************************************************
3045  006710  000004          TST4:   SCOPE
3046  006712  012737  000001  001174          MOV     #1,$TIMES       ;;DO 1 ITERATION
3047  006720  012737  000004  001336          MOV     #4,TSTNM        ;LOAD THE TEST NUMBER
3048  006726  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
3049  006732  012777  013560  014570          MOV     #IDLEX,@RMVEC   ;RESET THE INTERRUPT VECTOR
3050  006740  005077  014566                  CLR     @RMVEC+2        ;CLEAR THE INTERRUPT LEVEL
3051  006744  104401  001205                  TYPE    ,$CRLF
3052  006750  104401  036001                  TYPE    ,MESG12
3053  006754  104401  001272                  TYPE    ,$CDW2
3054  006760  013746  001222                  MOV     DRIVE,-(SP)
3055  006764  104403                          TYPOS
3056  006766     002                          .BYTE   2
3057  006767     000                          .BYTE   0
3059  006770  104401  036604                  TYPE    ,LINSP
3059  006774  104401  036017                  TYPE    ,MESG13
3060  007000  012701  000001                  MOV     #1,R1
3061  007004  005002                          CLR     R2
3062  007006  020237  001266          1$:     CMP     R2,$DEVM        ;LOCATE THE COORESPONDING BIT MAP
3063  007012  001404                          BEQ     2$              ;BRANCH IF LOCATED
```

```
3064  007014  000241                        CLC                     ;LOCATE NEXT DRIVE
3065  007016  006101                        ROL     R1              ;
3066  007020  005202                        INC     R2              ;NEXT DRIVE #
3067  007022  000771                        BR      1$              ;LOCATE THE BIT MAP
3068  007024  040137  002000          2$:   BIC     R1,ASSGN1       ;DEASSIGN THE LOGICAL DRIVE FOR PASS 1
3069  007030  001410                        BEQ     4$              ;NO MORE DRIVES
3070  007032  005202                  3$:   INC     R2              ;GET NEXT LOGICAL DRIVE #
3071  007034  006302                        ASL     R2              ;WORD INDEX
3072  007036  016237  002600  001270        MOV     BLKADR(R2),$CDW1        ;LOAD THE NEW DPB ADDRESS
3073  007044  006202                        ASR     R2              ;RESTORE R2
3074  007046  010237  001266               MOV     R2,$DEVM        ;LOAD THE NEW LOCICAL DRIVE #
3075  007052  104411                  4$:   RDLIN                   ;WAIT UNTIL IT IS DONE
3076  007054  012605                        MOV     (SP)+,R5        ;LOCATE THE INPUT LINE
3077  007056  122725  000122               CMPB    #'R,(R5)+       ;
3078  007062  001373                        BNE     4$              ;
3079  007064  105715                        TSTB    (R5)            ;TERMINATOR ?
3080  007066  001371                        BNE     4$              ;BRANCH IF NOT
3081  007070  005737  002000               TST     ASSGN1          ;OTHER DRIVES ?
3082  007074  001002                        BNE     5$              ;BRANCH IF MORE DRIVES IN TEST
3083  007076  000137  007106               JMP     XPASS2          ;BRANCH TO PASS 2
3084  007102  000137  005502          5$:   JMP     TST1            ;JUMP TO TEST 1
3085
3086
3087                                  ;XPASS2         INITILIZE FOR PASS 2 TEST
3088                                  ;$CDW1  :       ADDRESS OF THE CURRENT LOGICAL DRIVE HISTORY FILE
3089                                  ;$CDW2  :       SYSTEM NAME A THROUGH H
3090                                  ;$DEVM  :       CURRENT LOGICAL DRIVE # 0 TO 15.
3091                                  ;ASSGN2 :       ASSIGNED LOGICAL DRIVE FOR PASS 2
3092                                  ;ASNLST :       ASSIGNED LOGICAL DRIVE
3093                                  ;DRIVE  :       PHYSICAL DRIVE # OF CURRENT RH/RM SYSTEM
3094
3095
3096
3097  007106  005737  002002          XPASS2: TST     ASSGN2          ;ANYTHING IN TEST FOR PASS 2
3098  007112  001002                        BNE     1$              ;YES,THEN GO ON
3099  007114  000137  013264               JMP     XEND2           ;JUMP TO END OF PASS 2
3100  007120  005037  001266          1$:   CLR     $DEVM           ;START FROM LOGICAL DRIVE 0
3101  007124  013737  002600  001270        MOV     BLKADR,$CDW1    ;ADDRESS OF LOGICAL BLOCK DRIVE 0
3102  007132  112737  000101  001272        MOVB    #'A,$CDW2       ;LOAD SYSTEM NAME "A"
3103  007140  013737  002004  023526        MOV     SYSADR,RMADR    ;LOAD SYSTEM-A ADDRESS TO DRIVER
3104  007146  013737  002006  023530        MOV     SYSADR+2,RMVEC  ;LOAD SYSTEM-A VECTOR TO DRIVER
3105  007154  013701  001270               MOV     $CDW1,R1        ;R1=ADDRESS OF LOGICAL BLOCK
3106  007160  104401  001205               TYPE    ,$CRLF
3107                                  ;               TYPE    ,MESG14 ;STARTING PASS 2
3108  007164  005037  001374               CLR     FAULT           ;RESSET THE NOT COMPATIBLE FLAG
3109  007170  000400                        BR      TST5            ;BRANCH TO TEST 5
3110
3111
3112
3113                                  ;$CDW1  :       ADDRESS OF CURRENT LOGICAL DRIVE BLOCK,ONLY CHANGED BY TST9
3114                                  ;$CDW2  :       SYSTEM-NAME             ,ONLY CHANGED BY TEST 5
3115                                  ;$DEVM  :       CURRENT LOGICAL DRIVE #
3116                                  ;ASSGN2 :       ASSIGNED LOGICAL DRIVE'S BIT MAP FOR PASS 2
3117                                  ;DRIVE  :       PHYSICAL DRIVE #
3118                                  ;ASNLST :       ASSIGNED LOGICAL DRIVE IN THE TEST
3119                                  ;IN TEST 5 DIRECT OPERATOR TO CHANGE,LOAD THE PACK
```

```
                                        ;
                                        ;
                                        ;;*****************************************************************
007172  000004                        TSTS:   SCOPE
007174  012737  000001  001174                MOV     #1,$TIMES       ;;DO 1 ITERATION
007202  012737  000005  001336                MOV     #5,TSTNM        ;LOAD THE TEST NUMBER
007210  012706  001100                        MOV     #STACK,SP       ;LOAD THE STACK POINTER
007214  012777  013560  014306                MOV     #IDLEX,@RMVEC   ;RESET THE INTERRUPT VECTOR
007222  005077  014304                        CLR     @RMVEC+2        ;CLEAR THE INTERRUPT LEVEL
007226  013701  001270                        MOV     $CDW1,R1        ;ADDRESS OF THE HISTORY FILE
007232  126137  000014  001272                CMPB    $SYSNM(R1),$CDW2        ;ON THE SAME SUB-SYSTEM
007240  001420                                BEQ     1$              ;THEN DON'T UPDATE SYSTEM ADDRESS
007242  116137  000014  001272                MOVB    $SYSNM(R1),$CDW2
007250  013700  001272                        MOV     $CDW2,R0        ;LOCATE SYSTEM ADDRESS TABLE
007254  005300                                DEC     R0              ;ADJUST FOR INDEX FORM 0
007256  042700  177760                        BIC     #177760,R0      ;LEFT ON FOUR BITS
007262  006300                                ASL     R0
007264  006300                                ASL     R0              ;INDEX FOR TWO WORD
007266  016037  002004  023526                MOV     SYSADR(R0),RMADR ;SYSTEM ADDRESS
007274  016037  002006  023530                MOV     SYSADR+2(R0),RMVEC ;SYSTEM INTERRUP VECTOR
                                        ;1$:    MOV     #IDLEX,@RMVEC   ;RESET THE INTERRUPT VECTOR
                                                CLR     @RMVEC+2        ;CLEAR THE INTERRUPT LEVEL
007302  104401  001205                1$:     TYPE    ,$CRLF
007306  104401  036072                        TYPE    ,MESG14         ;START THE PASS 2
007312  104401  001205                        TYPE    ,$CRLF          ;
007316  104401  035557                        TYPE    ,MESG4  ;MOUNT PACK ON DRIVE
007322  104401  001272                        TYPE    ,$CDW2  ;SYSTEM NAME
007326  111137  001222                        MOVB    (R1),DRIVE      ;LOCATE THE PHYSICAL DRIVE #
007332  111146                                MOVB    (R1),-(SP)
007334  104403                                TYPOS
007336     002                                .BYTE   2
007337     000                                .BYTE   0
007340  104401  036604                        TYPE    ,LINSP
007344  104401  035605                        TYPE    ,MESG8  ;AND LOAD
007350  104401  001205                        TYPE    ,$CRLF
007354  104401  035620                        TYPE    ,MESG9  ;TYPE R<CR> WHEN DRIVE IS READY
007360  104411                        2$:     RDLIN
007362  012602                                MOV     (SP)+,R2        ;LOCATE THE READ IN LINE
007364  122722  000122                        CMPB    #'R,(R2)+       ;FIRST CHAR IS A "R"
007370  001373                                BNE     2$              ;BRANCH IF NOT
007372  105712                                TSTB    (R2)            ;FOLLOWED BY A TERMINATOR
007374  001371                                BNE     2$              ;BRANCH IF NOT
007376  004737  023544                        JSR     PC,RMINIT
007402  012737  177777  023466                MOV     #-1,SAVEFG      ;SAVE ALL RH/RM REGISTERS
007410  012737  177777  023470                MOV     #-1,SEEKFG      ;CON'T DO IMPLY SEEK
007416  013700  001222                        MOV     DRIVE,R0        ;LOAD THE PHYSICAL DRIVE NUMBER
007422  105760  023400                        TSTB    DRVSTA(R0)      ;DRIVE EXISTS AND ON-LINE ?
007426  003405                                BLE     3$              ;BRANCH IF NOT
007430  105760  023410                        TSTB    DRVTYP(R0)      ;AN RM03 ?
007434  003402                                BLE     3$              ;NOT AN RM03
007436  000137  007454                        JMP     TST6            ;BRANCH IF IT IS.(NEXT TEST)
007442  104401  001205                3$:     TYPE    ,$CRLF
007446  104401  035657                        TYPE    ,MESG10 ;DRIVE IS NOT READY
007452  000647                                BR      TSTS            ;TRY AGAIN
```

```
3176
3177
3178                                    ;DRIVE  :       PHYSICAL DRIVE NUMBER
3179                                    ;$CDW1  :       DPB BLOCK OF THIS LOGICAL DRIVE
3180                                    ;$DEVM  :       LOGICAL DRIVE #
3181                                    ;$CDW2  :       SUB-SYSTEM NAME
3182                                    ;RMADR  :       SUB-SYSTEM BASE REGISTER ADDRESS
3183                                    ;RMVEC  :       SUB-SYSTEM INTERRUPT VECTOR
3184
3185                                    ;THE ABOVE PARAMETERS CANNOT BE MODIFIED BY THIS TEST (TST6)
3186                                    ;THE FOLLOWING REG'S ARE ASSIGNED AS:
3187                                    ;R0     :       ADDRESS OF DPB FEEDED INTO DRIVER HANDLER=FMTDPB
3188                                    ;R1     :       ADDRESS OF THE HISTORY FILE BLOCK OF THE LOGICAL DRIVE=$CDW1
3189
3190
3191                                    ;IN TEST 6,EACH LOGICAL DRIVE WRITES 7 CYLINDERS ON EACH SURFACE WITH
3192                                    ;AN UNIQUE DATA PATTERN.  THESE 7 CYLINDERS HAS BEEN
3193                                    ;WRITTEN BY OTHER DRIVES IN PASS 1.
3194                                    ;THEN,THIS LOGICAL DRIVE EXECUTES "WRITE CHECK DATA" TO SEE IF
3195                                    ;THIS LOGICAL DRIVE CAN OVER-WRITE ALL DATA WRITTEN BY OTHER
3196                                    ;DRIVES.
3197
3198                                    ;THESE 7 CYLINDERS ARE SPECIFIED AS $DEVM,$DEVM+128.,$DEVM+256
3199                                    ;$DEVM+384,$DEVM+512,$DEVM+640 AND $DEVM+768.
3200
3201                                    ;THE OVER-WRITE TEST IS PERFORMED WITH OFFSETS IN BOTH DIRECTIONS.
3202                                    ;
3203
3204
3205                                    ;;*****************************************************************
3206  007454  000004                   TST6:   SCOPE
3207  007456  012737  000001  001174           MOV     #1,$TIMES       ;;DO 1 ITERATION
3208  007464  012737  000006  001336           MOV     #6,TSTNM        ;LOAD TEST NUMBER
3209  007472  012706  001100                   MOV     #STACK,SP       ;INITIAL THE STACK POINT
3210  007476  012703  002046                   MOV     #OVWNO,R3       ;1 ST ADDRESS TO CLEAR
3211  007502  012704  002306                   MOV     #RDNO,R4        ;LAST ADDRESS+2
3212  007506  005023                   1$:     CLR     (R3)+           ;RESET ALL SCORE BOARD
3213  007510  020403                           CMP     R4,R3           ;ALL LOCATIONS ARE CLEARED ?
3214  007512  101375                           BHI     1$              ;BRANCH IF NOT
3215  007514  012700  037760                   MOV     #FMTDPB,R0      ;SET UP PARAMETER BLOCK
3216  007520  013701  001270                   MOV     $CDW1,R1        ;HISTORY FILE BLOCK
3217  007524  113710  001222                   MOVB    DRIVE,(R0)      ;LOAD THE PHYSICAL DRIVE #
3218  007530  012760  160000  000004           MOV     #-8192.,$WRDM(R0) ;LOAD THE WORD COUNT
3219  007536  005060  000010                   CLR     $SEC(R0)        ;START FROM  TRK0,SEC0
3220  007542  112760  000161  000002           MOVB    #WRTDAT,$COMND(R0)  ;WRITE DATA COMMAND
3221  007550  013760  001266  000012           MOV     $DEVM,$CYL(R0)  ;LOAD THE STARTING CYLINDER
3222  007556  012760  040074  000006           MOV     #ENDPGM,$BUF(R0) ;LOAD THE BUFFER ADDRESS
3223  007564  012760  040000  000014           MOV     #RM.REG,14(R0)  ;REG'S SAVE ADDRESS
3224  007572  013702  001266                   MOV     $DEVM,R2        ;LOCATE THE DATA PATTERN
3225  007576  006302                           ASL     R2              ;WORD INDEX
3226  007600  016202  002702                   MOV     PSEUDO(R2),R2   ;LOAD R2 THE DATA PATTERN POINTED
3227                                                                   ;BY ITSELF
3228  007604  016003  000006                   MOV     $BUF(R0),R3     ;BUFFER ADDRESS
3229  007610  012704  020000                   MOV     #8192.,R4       ;WORD COUNT
3230  007614  010223                   2$:     MOV     R2,(R3)+        ;FILL THE BUFFER
3231  007616  005304                           DEC     R4              ;DECREMENT WORD CTR
```

```
3232  007620  001375                        BNE     2$              ;BRANCH IF NOT DONE
3233                                                                  START TO WRITE ONE CYLINDER
3234                                                                  ON EACH SURFACE OF EACH WRITE
3235                                                                  CURRENT ZONE ( 7 ZONES ON ONE PACK)
3236  007622  004037  024272          3$:     JSR     R0,RM03         ;CALL THE DRIVER
3237  007626  037760                        FMTDPB                  ;PARAMETER BLOCK
3238  007630  000774                        BR      3$              ;BRANCH IF QUEUE FAIL
3239  007632  005737  037776          4$:     TST     FMTDPB+16       ;WRITE COMMAND DONE ?
3240  007636  001775                        BEQ     4$              ;NO,THEN WAIT
3241  007640  012700  037760                MOV     #FMTDPB,R0      ;PROCESS IF ANY ERROR
3242  007644  004737  013566                JSR     PC,PROCES
3243  007650  005760  000016                TST     16(R0)          ;ERROR FLAG SET ?
3244  007654  100010                        BPL     22$             ;BRANCH IF NOT
3245  007656  004737  023544                JSR     PC,RMINIT       ;INITIAL THE DRIVE
3246  007662  012737  177777  023466        MOV     #-1,SAVEFG
3247  007670  012737  177777  023470        MOV     #-1,SEEKFG
3248  007676  105260  000011          22$:    INCB    $TRK(R0)        ;NEXT TRACK
3249  007702  122760  000005  000011        CMPB    #5,$TRK(R0)     ;LAST TRACK IS DONE ?
3250  007710  101344                        BHI     3$              ;NO,THEN BRANCH
3251  007712  105060  000011                CLRB    $TRK(R0)        ;RESET TRACK NUMBER
3252  007716  062760  000200  000012        ADD     #128.,$CYL(R0)  ;MOVE TO NEXT ZONE
3253  007724  022760  001417  000012        CMP     #783.,$CYL(R0)  ;LAST ZONE IS DONE ?
3254  007732  103333                        BHIS    3$              ;BRANCH IF NOT
3255
3256
3257                                                                  ;RESET THE FMTDPB BLOCK AND
3258                                                                  ;EXECUTE WRITE-CHECK COMMAND
3259                                                                  ;TO DETECT ANY COMPATIBLE PROBLEM
3260                                                                  ;THE FOLLOWING SUBROUTINE ARE CALLED
3261                                                                  ;SCORE,OFFST,MAKEUP.
3262
3263  007734  005037  002640                CLR     OFFCOD          ;SET NEGATIVE OFFSET DIRECTION FLAG
3264  007740  012700  037760          LOOP1:  MOV     #FMTDPB,R0      ;R0=FMTDPB ADDRESS
3265  007744  005060  000010                CLR     $SEC(R0)        ;START FROM SECTOR 0,SURFACE 0
3266  007750  013760  001266  000012        MOV     $DEVM,$CYL(R0)  ;STARTING CYLINDER NUMBER
3267                                                                  ;TOTAL 7 CYLINDERS ON ONE SURFACE
3268  007756  112760  000151  000002  LOOP2: MOVB    #WCKD,$COMND(R0)        ;WRITE-CHECK-DATA COMMAND
3269  007764  012760  040074  000006        MOV     #ENDPGM,$BUF(R0)        ;RESET BUFFER ADDRESS
3270  007772  012760  040000  000014        MOV     #RM.REG,14(R0)  ;ADDRESS TO SAVE RH/RM03 REG'S
3271  010000  004037  024272          1$:     JSR     R0,RM03         ;CALL THE DRIVER
3272  010004  037760                        FMTDPB                  ;PARAMETER BLOCK ADDRESS
3273  010006  000774                        BR      1$              ;BRANCH IF QUEUE FAILURE
3274  010010  005737  037776          2$:     TST     FMTDPB+16       ;TEST IF COMMAND IS DONE ?
3275  010014  001775                        BEQ     2$              ;BRANCH , IF NOT
3276  010016  012700  037760                MOV     #FMTDPB,R0      ;LOAD THE PARAMETER BLOCK ADDRESS
3277  010022  004737  013566                JSR     PC,PROCES       ;REPORT IF ANY ERROR
3278  010026  004737  011144                JSR     PC,LABAD        ;LOCATE STARTING AND ENDING SECTORS
3279  010032  005760  000016                TST     16(R0)          ;ANY ERROR ?
3280  010036  100011                        BPL     3$              ;BRANCH IF NONE
3281  010040  004737  023544                JSR     PC,RMINIT       ;INITIAL THE SYSTEM
3282  010044  012737  177777  023466        MOV     #-1,SAVEFG
3283  010052  012737  177777  023470        MOV     #-1,SEEKFG
3284  010060  000414                        BR      5$              ;NOT UPDATE THE SCORE
3285                                    ;   CMP     CMSEC,STARSC    ;ON THE SAME SECTOR FOR STARTING AND ENDING
3286                                    ;   BLOS    4$              ;DON'T INCREMENT SCORES
3287                                    ;   DEC     CMSEC           ;DECREMENT THE SECTOR ADDRESS
```

```
3288  010062  004737  010724          3$:     JSR     PC,SCORE        ;INCREMENT SCORE
3289  010066  005760  000022          4$:     TST     $RMWC(R0)       ;WORD COUNT = 0 /
3290  010072  001407                          BEQ     5$              ;BRANCH, IF WORD COUNT IS 0
3291  010074  116060  000026  000010          MOVB    $RMDA(R0),$SEC(R0)  ;UPDATE STARTING SECTOR
3292  010102  016060  000022  000004          MOV     $RMWC(R0),$WRDM(R0) ;UPDATE WORD COUNT
3293  010110  000733                          BR      1$              ;TO READ THE REST SECTORS
3294
3295
3296                                          ;THE FOLLOWING CODING TEST THE
3297                                          ;COMPATIBLE PROBLEM IN OFFSET MODE
3298                                          ;OFFCOD=0,NEGATIVE ;OFFCOD=1 ,POSITIVE
3299
3300  010112  012700  037760          5$:     MOV     #FMTDPB,R0      ;RESET THE DPB BLOCK
3301  010116  012760  160000  000004          MOV     #-8192.,$WRDM(R0)   ;FULL TRACK WORD COUNT
3302  010124  105060  000010                  CLRB    $SEC(R0)        ;RESET TO SECTOR 0,SURFACE NOT CHANGED
3303  010130  012760  040074  000006          MOV     #ENDPGM,$BUF(R0)        ;BUFFER ADDRESS
3304  010136  012760  040000  000014          MOV     #RM.REG,14(R0)  ;ADDRESS TO SAVE ALL RH/RM03 REG'S
3305  010144  004537  010354          6$:     JSR     R5,OFFST        ;CALL OFFSET
3306  010150  000443                          BR      10$             ;BRANCH TO NEXT CYLINDER,IF OFFSET FAILS
3307  010152  004737  010606                  JSR     PC,MAKEUP       ;CALL WRITE CHECK IN OFFSET MODE
3308  010156  005737  037776          7$:     TST     FMTDPB+16       ; OFFSET WRITE CHECK IS DONE ?
3309  010162  001775                          BEQ     7$              ;BRANCH IF NOT
3310
3311  010164  012700  037760                  MOV     #FMTDPB,R0      ;LOAD THE DPB ADDRESS
3312  010170  004737  013566                  JSR     PC,PROCES       ;REPORT  IF ANY ERROR
3313  010174  004737  011144                  JSR     PC,LABAD        ;LOCATE STARTINF AND ENDING SECTORS
3314  010200  005760  000016                  TST     16(R0)          ;ANY ERROR ?
3315  010204  100011                          BPL     8$              ;BRANCH, IF NONE
3316  010206  004737  023544                  JSR     PC,RMINIT       ;INITIAL THE SYSTEM
3317  010212  012737  177777  023466          MOV     #-1,SAVEFG
3318  010220  012737  177777  023470          MOV     #-1,SEEKFG
3319  010226  000414                          BR      10$             ;NOT UPDATE THE SCORE
3320                                  ;       CMP     CMSEC,STARSC    ;STARTING AND ENDING ADDRESSES O.K ?
3321                                  ;       BLOS    9$              ;BRANCH IF NOT
3322                                  ;       DEC     CMSEC           ;EXCLUDING THE BAD SECTOR
3323  010230  004737  010724          8$:     JSR     PC,SCORE        ;UPDATE THE TEST SCORE
3324  010234  005760  000022          9$:     TST     $RMWC(R0)       ;WORD CTR = 0 ?
3325  010240  001407                          BEQ     10$             ;IF WORD CTR=0 ,BRANCH TO NEXT OP
3326  010242  116060  000026  000010          MOVB    $RMDA(R0),$SEC(R0) ;UPDATE THE NEW STARTING ADDRESS
3327  010250  016060  000022  000004          MOV     $RMWC(R0),$WRDM(R0) ;UPDATE THE NEW WORD COUNT
3328  010256  000732                          BR      6$
3329  010260  062760  000200  000012  10$:     ADD     #128.,$CYL(R0)  ;ADJUST CYLINDER ADDRESS TO NEXT ZONE
3330  010266  022760  001417  000012          CMP     #763.,$CYL(R0)  ;ALL 7 ZONES HAVE BEEN TESTED ?
3331  010274  103402                          BLO     11$             ;BRANCH, IF ALL DONE
3332  010276  000137  007756                  JMP     LOOP2           ;TO NEXT WRITE CURRENT ZONE
3333  010302  013760  001266  000012  11$:     MOV     $DEVM,$CYL(R0)  ;RESET CYLINDER ADDRESS
3334  010310  105260  000011                  INCB    $TRK(R0)        ;INCREMENT TO NEXT SURFACE
3335  010314  122760  000005  000011          CMPB    #5,$TRK(R0)     ;ALL FIVE SURFACES ARE TESTED ?
3336  010322  101402                          BLOS    12$             ;BRANCH IF ALL DONE
3337  010324  000137  007756                  JMP     LOOP2           ;TO NEXT SURFACE
3338  010330  005737  002640          12$:    TST     OFFCOD          ;FINISHING TEST THE POSITIVE OFFSET ?
3339  010334  001005                          BNE     13$             ;BRANCH IF ALL DONE
3340  010336  012737  000001  002640          MOV     #1,OFFCOD       ;SET POSITIVE OFFSET DIRECTION FLAG
3341  010344  000137  007740                  JMP     LOOP1           ;RESTART LOCATION
3342  010350  000137  011250          13$:    JMP     TST7            ;BRANCH TO NEXT TEST
3343
```

```
                                        ;OFFST ROUTINE
                                        ;OFFSET THE HEAD IN THE DIRECTION TOWARD SPINDLE OR AWAY FROM THE SPINDLE
                                        ;OFFCOD = 1      TOWARD SPINDLE (POSITIVE)
                                        ;OFFCOD = 0      AWAY FROM SPINDLE (NEGATIVE)
                                        ;DRIVE           PHICAL DRIVE NUMBER
                                        ;R0              DPB ADDRESS
                                        ;RETRY  3 TIMES
                                        ;CALLING SEQUENCE
                                        ;       JSR     R5,OFFST
                                        ;       RET1                    OFFSET FAIL RETURN ADDRESS
                                        ;       RET2                    OFFSET SUCCESSFUL RETURN

010354  010146          OFFST:  MOV     R1,-(SP)                        ;SAVE ALL REGISTERS
010356  010246                  MOV     R2,-(SP)
010360  010346                  MOV     R3,-(SP)
010362  010446                  MOV     R4,-(SP)
010364  012700  037760          MOV     #FMTDPB,R0      ;R0 DPB ADDRESS
010370  116046  000002          MOVB    $COMND(R0),-(SP)        ;SAVE THE I/O COMMAND
010374  113710  001222          MOVB    DRIVE,(R0)      ;LOAD THE DRIVE NUMBER
010400  112760  000117  000002  MOVB    #117,$COMND(R0) ;RETURN TO CENTER COMMAND
010406  004037  024272          JSR     R0,RM03         ;CALL THE DRIVE HANDLE
010412  037760                  FMTDPB
010414  000465                  BR      6$              ;BRANCH IF QUEUE FAILS
010416  005737  037776  1$:     TST     FMTDPB+16       ;COMMAND DONE ?
010422  001775                  BEQ     1$              ;BRANCH IF NOT
010424  100461                  BMI     6$              ;BRANCH IF ERROR EXIST
010426  013746  177776          MOV     @#PS,-(SP)      ;SAVE THE PSW
010432  012737  000240  177776  MOV     #(5*32.),@#PS   ;LOAD PS 5
010440  012700  037760          MOV     #FMTDPB,R0      ;DPB ADDRESS
010444  013704  023526          MOV     RMADR,R4        ;MUSS BUS ADDRESS
010450  013701  001222          MOV     DRIVE,R1        ;DRIVE NUMBER
010454  010164  000010          MOV     R1,RMCS2(R4)    ;LOAD THE DRIVE NUMBER INTO CONTROLLER
010460  016064  000012  000034  MOV     $CYL(R0),RMDC(R4) ;CYLINDER NUMBER
010466  016064  000010  000006  MOV     $SEC(R0),RMDA(R4) ;SECTOR AND TRACK NUMBER
010474  016064  000004  000002  MOV     $WRDM(R0),RMWC(R4) ;WORD COUNT
010502  016064  000006  000004  MOV     $BUF(R0),RMBA(R4) ;BUFFER ADDRESS
010510  012637  177776          MOV     (SP)+,@#PS      ;LOAD THE PSW BACK
010514  112760  000200  000001  MOVB    #BIT7,$FMT(R0)  ;LOAD THE OFFSET DIRECTION
010522  005737  002640          TST     OFFCOD          ;NEG ?
010526  001003                  BNE     2$              ;BRANCH IF NOT
010530  112760  000000  000001  MOVB    #0,$FMT(R0)     ;CHANGE TO OTHER DIRECTION
010536  112760  000115  000002  2$:     MOVB    #115,$COMND(R0) ;LOAD THE OFFSET COMMAND
010544  004037  024272          JSR     R0,RM03         ;CALL THE DRIVE HANDLE
010550  037760                  FMTDPB
010552  000406                  BR      6$              ;BRANCH IF QUEUE FAILS
010554  005737  037776  3$:     TST     FMTDPB+16       ;OFFSET DONE ?
010560  001775                  BEQ     3$              ;BRANCH IF SO
010562  100402                  BMI     6$              ;BRANCH IF ERROR
010564  062705  000002          ADD     #2,R5           ;ADJUST RETURN ADDRESS
010570                  6$:
010570  112637  037762          MOVB    (SP)+,FMTDPB+$COMND     ;RESTORE THE I/O COMMAND
010574  012604                  MOV     (SP)+,R4        ;RESTORE REG
010576  012603                  MOV     (SP)+,R3
```

```
3400  010600  012602                        MOV     (SP)+,R2
3401  010602  012601                        MOV     (SP)+,R1
3402  010604  000205                        RTS     R5              ;EXIT
3403
3404
3405                                  ;MAKEUP ROUTINE
3406                                  ;THIS ROUTINE ISSUES A WRITE CHECK OR READ  COMMAND TO THE SELECTED DRIVE
3407                                  ;IN OFFSET MODE.
3408                                  ;AND SET UP THE FOLLOWING PARAMETERS
3409                                  ;
3410                                  ; DTUW = PHYSICAL DRIVE NUMBER
3411                                  ; TRNSWT  =  FMTDPB
3412                                  ; DRVACT (DRIVE)  = 1
3413                                  ;TIMER (DRIVE)  = 1 SECOND
3414                                  ;
3415                                  ;CALLING SEQ
3416                                  ;
3417                                  ;       JSR     PC,MAKEUP
3418                                  ;       RET
3419                                  ;MAIN PURPOSE OF THIS ROUTINE,TO EXECUTE A COMMAND WHILE ASSURE THAT
3420                                  ;THIS READ OR WRITE-CHECK COMMAND BEING EXECUTED IN OFFSET MODE.
3421                                  ;ROUTINES USED TD,SC,STO,( IN DRIVE HANDLER )
3422                                  ;
3423
3424
3425  010606  010146                  MAKEUP: MOV     R1,-(SP)
3426  010610  010246                          MOV     R2,-(SP)
3427  010612  010446                          MOV     R4,-(SP)
3428  010614  012702  000151                  MOV     #WCKD,R2        ;WRITE CHECK DATA IN TEST 6
3429  010620  022737  000010  001336          CMP     #10,TSTNM       ;ON TEST 8 ?
3430  010626  001002                          BNE     1$              ;BRANCH IF NOT
3431  010630  012702  000171                  MOV     #RDDAT,R2       ;READ DATA COMMAND IN TEST 8
3432  010634  005037  037776          1$:     CLR     FMTDPB+16       ;CLEAR THE STATUS WORD
3433  010640  110237  037762                  MOVB    R2,FMTDPB+$COMND ;LOAD THE COMMAND INTO PDB
3434  010644  013737  001222  023512          MOV     DRIVE,DTUW      ;ACTIEV DRIVE NUMBER
3435  010652  012737  037760  023440          MOV     #FMTDPB,TRNSWT  ;TRANSFER UNDERWAY FLAG
3436  010660  013701  001222                  MOV     DRIVE,R1        ;DRIVE NUMBER
3437  010664  013704  023526                  MOV     RMADR,R4        ;RH/RM BASE ADDRESS
3438  010670  112761  000001  023370          MOVB    #1,DRVACT(R1)   ;ACTIVE DRIVE FLAG
3439  010676  006301                          ASL     R1              ;WORD INDEX
3440  010700  012761  060000  023472          MOV     #60000,TIMER(R1) ; ONE SECOND TIMER
3441  010706  006201                          ASR     R1
3442  010710  010264  000000                  MOV     R2,RMCS1(R4)    ;ISSURE WRITE CHECK OR READ COMMAND
3443  010714  012604                          MOV     (SP)+,R4        ;RESTORE REG 4, 1
3444  010716  012602                          MOV     (SP)+,R2
3445  010720  012601                          MOV     (SP)+,R1
3446  010722  000207                          RTS     PC              ;EXIT
3447
3448
3449                                  ;SCORE ROUTINE
3450                                  ;ROUTINE TO UPDATE THE TEST SCORE
3451                                  ;(TABLEX):ADDRESS OF CRRENT SCORE BOARD
3452                                  ;$DEVM: LOGICAL DRIVE #
3453                                  ;DRIVE: PHYSICAL DRIVE NUMBER
3454                                  ;CMSEC: END SECTOR ADDRESS
3455                                  ;STARSC: START SECTOR ADDRESS
```

```
3456                                    ;CALL SEQ
3457                                    ;
3458                                    ;       JSR     PC,SCORE
3459                                    ;       RET
3460
3461
3462  010724                            SCORE:
3463  010724  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3464  010726  010246                            MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3465  010730  010346                            MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3466  010732  010446                            MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3467  010734  023737  001360  001356            CMP     CMSEC,STARSC    ;CORRECT START AND STOP ADDRESSES
3468  010742  003473                            BLE     9$              ;BRANCH IF NOT
3469  010744  022737  000010  001336            CMP     #10,TSTNM       ;ON TEST 8
3470  010752  001011                            BNE     2$              ;BRANCH IF NOT (MUST BE TEST 8,
3471  010754  005737  002640                    TST     OFFCOD          ;NEGATIVE OFFSET ?
3472  010760  001403                            BEQ     1$              ;BRANCH IF NEGATIVE OFFSET
3473  010762  012703  002426                    MOV     #RDPO,R3        ;SCORE BOARD ADDRESS
3474  010766  000413                            BR      4$
3475  010770  012703  002306            1$:     MOV     #RDNO,R3        ;SCORE BOARD ADDRESS
3476  010774  000410                            BR      4$
3477  010776  005737  002640            2$:     TST     OFFCOD          ;NEGATIVE OFFSET
3478  011002  001403                            BEQ     3$              ;BRANCH IF NEGATIVE OFFSET
3479  011004  012703  002166                    MOV     #OVWPO,R3       ;SCORE BOARD ADDRESS
3480  011010  000402                            BR      4$
3481  011012  012703  002046            3$:     MOV     #OVWNO,R3       ;SCORE BOARD ADDRESS
3482  011016  113702  037771            4$:     MOVB    @#FMTDPB+$TRK,R2 ;LOAD THE SURFACE NUMBER
3483  011022  005702                            TST     R2              ;ON SURFACE 0
3484  011024  001404                            BEQ     6$              ;BRANCH IF IT IS
3485  011026  062703  000020            5$:     ADD     #16.,R3         ;EACH SCORE BOARD TAKES 16 BYTES
3486  011032  005302                            DEC     R2              ;LOCATED ?
3487  011034  001374                            BNE     5$              ;BRANCH IF NOT
3488  011036  010337  002044            6$:     MOV     R3,TABLEX       ;STORE THE TABLE STARTING ADDRESS
3489  011042  010301                            MOV     R3,R1           ;RE ASSIGN REGISTERS
3490  011044  013702  001266                    MOV     $DEVM,R2        ;LOGICAL DRIVE #
3491  011050  013703  001356                    MOV     STARSC,R3       ;START SECTOR
3492  011054  116204  002560                    MOVB    INDST(R2),R4    ;LOCATE THE STARTING PRONT FOR SCORE BOARD
3493  011060  060304                            ADD     R3,R4           ;UPDATE POINTER
3494  011062  022704  000017            11$:    CMP     #15.,R4         ;SHOULD ADJUST POINTER ?
3495  011066  003403                            BLE     7$              ;NO
3496  011070  162704  000020                    SUB     #16.,R4         ;ENTRY POINT AT THE SCORE BOARD
3497  011074  000772                            BR      11$             ;
3498  011076  060401            7$:     ADD     R4,R1           ;
3499  011100  023703  001360            8$:     CMP     CMSEC R3        ;ENDING SECTOR REACHED ?
3500  011104  002412                            BLT     9$              ;BRANCH IF IT IS
3501  011106  105221                            INCB    (R1)+           ;INCREMENT SCORE AND POINT TO NEXT
3502                                                                    ;LOGICAL DRIVE
3503  011110  005204                            INC     R4              ;INCREMENT LOGICAL DRIVE #
3504  011112  005203                            INC     R3              ;INCREMENT SECTOR COUNT
3505  011114  022704  000017                    CMP     #15.,R4         ;TIME TO RESET TABLE ?
3506  011120  002367                            BGE     8$              ;BRANCH IF NOT
3507  011122  013701  002044                    MOV     TABLEX,R1       ;RESET TABLE ADDRESS
3508  011126  005004                            CLR     R4              ;LOGICAL DRIVE 0
3509  011130  000763                            BR      8$              ;LOOPING BACK
3510  011132                            9$:
3511  011132  012604                            MOV     (SP)+,R4        ;;POP STACK INTO R4
```

```
3512  011134  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
3513  011136  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
3514  011140  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
3515  011142  000207                          RTS     PC
3516
3517
3518                                    ;LABAD ROUTINE
3519                                    ;LOCATE THE START SECTOR AND THE TERMINATE SECTOR OF THE PREVIOUS
3520                                    ;OPERATION
3521                                    ;
3522                                    ;STARSC :       STARTING SECTOR
3523                                    ;CMSEC  :       ENDING SECTOR
3524                                    ;INFORMATION FROM $RMDA(FMTDPB),$RMWC(FMTDPB),$SEC(FMTDPB)
3525                                    ;               $SEC(FMTDPB),$WRDM(FMTDPB)
3526                                    ;
3527                                    ;CALLING SEQ
3528                                    ;
3529                                    ;       JSR     PC,LABAD
3530                                    ;       RET
3531
3532  011144                            LABAD:
3533  011144  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3534  011146  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3535  011150  012700  037760                  MOV     #FMTDPB,R0      ;DPB ADDRESS
3536  011154  116002  000026                  MOVB    $RMDA(R0),R2    ;HARDWARE TERMINATING SECTOR
3537  011160  042702  177740                  BIC     #177740,R2      ;CHOP OFF HIGH ORDER BITS IF ANY
3538  011164  116037  000010  001356          MOVB    $SEC(R0),STARSC ;STARTING SECTOR ADDRESS
3539  011172  005702                          TST     R2              ;TERMINATOR AT 0 SECTOR
3540  011174  001404                          BEQ     1$              ;BRANCH IF IT IS
3541  011176  005302                          DEC     R2              ;DECREMENT ONE SECTOR COUNT
3542  011200  010237  001360                  MOV     R2,CMSEC        ;ENDING SECTOR ADDRESS
3543  011204  000416                          BR      3$              ;EXIT
3544  011206  005760  000022          1$:     TST     $RMWC(R0)       ;WORD COUNT = 0
3545  011212  001410                          BEQ     2$              ;BRANCH IF IT IS
3546  011214  026060  000022  000004          CMP     $RMWC(R0),$WRDM(R0) ; WORD COUNT CHANGED AT ALL ?
3547  011222  001004                          BNE     2$              ;BRANCH IF CHANGED
3548  011224  116037  000010  001360          MOVB    $SEC(R0),CMSEC ;END SECTOR = START SECTOR
3549  011232  000403                          BR      3$              ;EXIT
3550  011234  112737  000037  001360  2$:     MOVB    #31.,CMSEC      ;END AT SECTOR 31
3551  011242                            3$:
3552  011242  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
3553  011244  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
3554  011246  000207                          RTS     PC
3555
3556                                    ;TEST 7
3557                                    ;SELF TEST:WRITE CYLINDERS $DEVM+17,$DEVM+17+128.$DEVM+17+128X2.ETC.
3558                                    ;THEN EXECUTE WRITE CHECK TO DECTECT ANY DATA OR HEADER PROBLEM
3559                                    ;FOR THE SELECTED LOGICAL DRIVE
3560                                    ;$DEVM: LOGICAL DRIVE #
3561                                    ;DRIVE:         PHYSICAL DRIVE #
3562                                    ;
3563                                    ;
3564
3565
3566                                    ;;*****************************************************************
3567  011250  000004                    ST7:    SCOPE
```

```
3568  011252  012737  000001  001174          MOV     #1,$TIMES       ;;DO 1 ITERATION
3569  011260  004737  023544                  JSR     PC,RMINIT       ;INITIAL THE DRIVE
3570  011264  012737  177777  023466          MOV     #-1,SAVEFG      ;SAVE THE RH REG'S
3571  011272  012737  177777  023470          MOV     #-1,SEEKFG
3572  011300  012737  000007  001336          MOV     #7,TSTNM        ;LOAD TEST NUMBER
3573  011306  012706  001100                  MOV     #STACK,SP       ;INITIAL THE STACK
3574  011312  012702  002546                  MOV     #SELFO,R2       ;CLEAR THE SELF TEST SCORE BOARDS
3575  011316  005022                  1$:     CLR     (R2)+           ;
3576  011320  022702  002560                  CMP     #SELFO+12,R2    ;ALL DONE ?
3577  011324  101374                          BHI     1$              ;BRANCH IF NOT
3578  011326  012700  037760                  MOV     #FMTDPB,R0                      ;SET UP DPB
3579  011332  012760  160000  000004          MOV     #-8192.,$WRDM(R0) ;LOAD FULL TRACK WORD COUNT
3580  011340  112760  000161  000002          MOVB    #WRTDAT,$COMND(R0)              ;WRITE DATA COMMAND
3581  011346  005060  000010                  CLR     $SEC(R0)                        ;SECTOR 0,SURFACE 0
3582  011352  113710  001222                  MOVB    DRIVE,(R0)                      ;LOAD PHY. DRIVE
3583  011356  013702  001266                  MOV     $DEVM,R2                        ;LOCATE THE STARTING CYLINDER
3584  011362  062702  000021                  ADD     #17.,R2
3585  011366  010260  000012                  MOV     R2,$CYL(R0)                     ;CYLINDER ADDRESS
3586  011372  012760  040074  000006          MOV     #ENDPGM,$BUF(R0)                ;RESET BUFFER ADDRESS
3587  011400  012760  040000  000014          MOV     #RM.REG,14(R0)                  ;ADDRESS TO SAVE ALL RH/RM03 REG'S
3588  011406  004737  015162                  JSR     PC,FILBUF                       ;FILL THE BUFFER WITH WORSE CASE PATTERN
3589  011412  004037  024272          2$:     JSR     R0,RM03                         ;CALL DRIVER HANDLER
3590  011416  037760                          FMTDPB
3591  011420  000774                          BR      2$                              ;BRANCH IF QUEUE FAILS
3592  011422  005737  037776          3$:     TST     FMTDPB+16                       ;ALL DONE ?
3593  011426  001775                          BEQ     3$                              ;BRANCH IF NOT
3594  011430  012700  037760                  MOV     #FMTDPB,R0                      ;REPORT IF ANY ERROR
3595  011434  004737  013566                  JSR     PC,PROCES
3596  011440  005760  000016                  TST     16(R0)          ;ERROR FLAG SET ?
3597  011444  100010                          BPL     22$             ;BRANCH IF NOT
3598  011446  004737  023544                  JSR     PC,RMINIT       ;INITIAL THE DRIVE
3599  011452  012737  177777  023466          MOV     #-1,SAVEFG
3600  011460  012737  177777  023470          MOV     #-1,SEEKFG
3601  011466  105260  000011          22$:    INCB    $TRK(R0)                        ;NEXT SURFACE
3602  011472  122760  000005  000011          CMPB    #5,$TRK(R0)                     ;ALL FIVE SURFACE ARE DONE ?
3603  011500  101344                          BHI     2$                              ;BRANCH IF NOT
3604  011502  005060  000010                  CLR     $SEC(R0)                        ;RESET SECTOR AND TRACK
3605  011506  062760  000200  000012          ADD     #128.,$CYL(R0)                  ;ADVANCE TO NEXT ZONE
3606  011514  022760  001440  000012          CMP     #800.,$CYL(R0)                  ; ALL 7 ZONES ARE DONE ?
3607  011522  103333                          BHIS    2$                              ;BRANCH IF NOT
3608                                                                                  ;EXECUTE WRITE CHECK
3609                                                                                  ;TO DETECT ANY DATA OR HEADER
3610                                                                                  ;PROBLEM
3611  011524  112760  000151  000002          MOVB    #WCKD,$COMND(R0)                ;CHANGE TO WRITE CHECK COMMAND
3612  011532  012702  002546                  MOV     #SELFO,R2                       ;R2 POINTS TO SCORE BOARD
3613                                                                                  ;CAN NOT BE DESTORIED
3614  011536  013703  001266                  MOV     $DEVM,R3                        ;LOACATE STARTING ADDRESS
3615  011542  062703  000021                  ADD     #17.,R3
3616  011546  010360  000012                  MOV     R3,$CYL(R0)                     ;STARTING CYLINDER
3617  011552  005037  002640                  CLR     OFFCOD                          ;SET NEGATIVE OFFSET FLAG
3618  011556  004037  024272          4$:     JSR     R0,RM03                         ;CALL DRIVE HANDLER
3619  011562  037760                          FMTDPB
3620  011564  000774                          BR      4$                              ;BRANCH IF QUEUE FAILS
3621  011566  005737  037776          5$:     TST     FMTDPB+16                       ;ALL DONE ?
3622  011572  001775                          BEQ     5$                              ;BRANCH IF NOT
3623  011574  012700  037760                  MOV     #FMTDPB,R0
```

```
3624  011600  004737  013566          JSR     PC,PROCES
3625  011604  005760  000016          TST     16(R0)                  ;ANY ERROR
3626  011610  100401                  BMI     6$                      ;YES,THEN DON'T INCREMENT SCORE
3627  011612  105212                  INCB    (R2)                    ;INCREMENT SCORE
3628  011614  004537  010354  6$:     JSR     R5,OFFST                ;OFFSET
3629  011620  000415                  BR      8$                      ;BRANCH IF OFFSET FAILS
3630  011622  004737  010606          JSR     PC,MAKEUP               ;EXECUTE WRITE CHECK IN OFFSET MODE
3631  011626  005737  037776  7$:     TST     FMTDPB+16               ;ALL DONE /
3632  011632  001775                  BEQ     7$                      ;BRANCH IF NOT
3633  011634  012700  037760          MOV     #FMTDPB,R0
3634  011640  004737  013566          JSR     PC,PROCES               ;REPORT IF ANY ERROR
3635  011644  005760  000016          TST     16(R0)                  ;ERROR BIT SET ?
3636  011650  100401                  BMI     8$                      ;DON'T INCREMENT SCORE
3637  011652  105212                  INCB    (R2)                    ;INCREMENT SCORE
3638  011654  005737  002640  8$:     TST     OFFCOD                  ;POSITIVE OFFSET IS DONE ?
3639  011660  001006                  BNE     9$                      ;BRANCH ,IF DONE
3640  011662  005202                  INC     R2                      ;INCREMENT SCORE BOARD POINTER
3641  011664  012737  000001  002640  MOV     #1,OFFCOD               ;SET POSITIVE OFFSET FLAG
3642  011672  000137  011556          JMP     4$                      ;POSITIVE OFFSET TEST
3643  011676  105260  000011  9$:     INCB    $TRK(R0)                ;INCREMENT TO NEXT SURFACE
3644  011702  005202                  INC     R2                      ;ADVANCE SCORE BOARD POINTER
3645  011704  122760  000005  000011  CMPB    #5,$TRK(R0)             ;ALL FIVE SURFACES ARE DONE ?
3646  011712  101404                  BLOS    10$                     ;BRANCH , IF ALL DNOE
3647  011714  005037  002640          CLR     OFFCOD                  ;RESET OFFSET DIRECTION
3648  011720  000137  011556          JMP     4$                      ;TRY THE NEXT SURFACE
3649  011724  005060  000010  10$:    CLR     $SEC(R0)                ;SURFACE 0, SECTOR 0
3650  011730  012702  002546          MOV     #SELF0,R2               ;RESET SCORE BOARD
3651  011734  062760  000200  000012  ADD     #128.,$CYL(R0)  ;ADVANCE TO NEXT ZONE
3652  011742  022760  001440  000012  CMP     #800.,$CYL(R0)  ;ALL 7 ZONES ARE DONE ?
3653  011750  103404                  BLO     11$                     ;BRANCH , IF ALL DONE
3654  011752  005037  002640          CLR     OFFCOD                  ;RESET OFFSET FLAG
3655  011756  000137  011556          JMP     4$                      ;TO NEXT ZONE
3656  011762  000240  11$:    NOP                                     ;TO NEXT TEST
3657
3658
3659                                  ;TEST10  TEST 8     READ COMPATIBLE TEST
3660                                  ;CYLINDERS TESTED : $DEVM+112,$DEVM+112+128XN   N=1 TO 5
3661                                  ;THIS TEST SELECT ONE CYLINDER FOR EACH LOGICAL DRIVE FROM
3662                                  ;ZONE 1 TO ZONE 6
3663                                  ;$DEVM :   LOGICAL DRIVE #
3664                                  ;DRIVE :        PHYSICAL DRIVE #
3665
3666
3667                                  ;;*****************************************************************
3668  011764  000004          TST10:  SCOPE
3669  011766  012737  000001  001174          MOV     #1,$TIMES       ;;DO 1 ITERATION
3670  011774  012737  000010  001336          MOV     #10,TSTNM       ;LOAD TEST NUMBER
3671  012002  012706  001100                  MOV     #STACK,SP       ;INITAIL THE STACK POINTER
3672  012006  012702  002306                  MOV     #RDNO,R2        ;CLEAR THE SCORE BOARD OF READ TEST
3673  012012  005022          1$:     CLR     (R2)+
3674  012014  022702  002546                  CMP     #RDP4+16.,R2    ;ALL DONE
3675  012020  101374                  BHI     1$                      ;BRANCH IF NOT
3676  012022  005037  002640                  CLR     OFFCOD          ;SET NEGATIVE OFFSET FLAG
3677  012026  012700  037760  LOOP3: MOV     #FMTDPB,R0              ;SET UP DPB BLOCK
3678  012032  012760  160000  000004          MOV     #-8192.,$WRDM(R0) ;LOAD THE WORD CTR,FULL TRACK
3679  012040  005060  000010                  CLR     $SEC(R0)        ;SURFACE 0 AND SECTOR 0
```

```
3680  012044  112760  000171  000002          MOVB    #RDDAT,$COMND(R0) ;LOAD THE READ DATA COMMAND
3681  012052  012760  040074  000006          MOV     #ENDPGM,$BUF(R0) ;RESET BUFFER ADDRESS
3682  012060  012760  040000  000014          MOV     #RM.REG,14(R0)  ;ADDRESS TO SAVE ALL RH/RM03 ADDRESS
3683  012066  113710  001222                  MOVB    DRIVE,(R0)      ;LOAD PHY. DRIVE ADDRESS
3684  012072  013703  001266                  MOV     $DEVM,R3        ;LOCATE STARTING CYL
3685  012076  062703  000160                  ADD     #112.,R3
3686  012102  010360  000012                  MOV     R3,$CYL(R0)     ;STARTING CYL ADDRESS
3687  012106  004037  024272          1$:     JSR     R0,RM03         ;CALL THE DRIVE HANDLER
3688  012112  037760                          FMTDPB
3689  012114  000774                          BR      1$
3690  012116  005737  037776          2$:     TST     FMTDPB+16       ;COMMAND DONE ?
3691  012122  001775                          BEQ     2$              ;BRANCH IF NOT
3692  012124  012700  037760                  MOV     #FMTDPB,R0
3693  012130  004737  013566                  JSR     PC,PROCES       ;REPORT IF ANY ERROR
3694  012134  004737  011144                  JSR     PC,LABAD        ;LOACATE START AND STOP ADDRESS
3695  012140  005760  000016                  TST     16(R0)          ;ANY ERROR ?
3696  012144  100011                          BPL     3$              ;BRANCH, IF NONE
3697  012146  004737  023544                  JSR     PC,RMINIT       ;INITIAL THE SYSTEM
3698  012152  012737  177777  023466          MOV     #-1,SAVEFG
3699  012160  012737  177777  023470          MOV     #-1,SEEKFG
3700  012166  000414                          BR      5$              ;NOT INCREMENT THE SCORE
3701                                  ;       CMP     CMSEC,STARSC    ;ON THE SAME SECTOR ADDRESS
3702                                  ;       BLOS    4$              ;
3703                                  ;       DEC     CMSEC           ;DECREMENT ONE SECTOR COUNT
3704  012170  004737  010724          3$:     JSR     PC,SCORE        ;UPDATE THE SCORE
3705  012174  005760  000022          4$:     TST     $RMWC(R0)       ;WORD COUNT= 0
3706  012200  001407                          BEQ     5$              ;BRANCH IIF IT IS
3707  012202  116060  000026  000010          MOVB    $RMDA(R0),$SEC(R0) ;UPDATE THE NEW STARTING SECTOR
3708  012210  016060  000022  000004          MOV     $RMWC(R0),$WRDM(R0) ;UPDATE WORD COUNT
3709  012216  000733                          BR      1$              ;CONTINUE
3710  012220  012700  037760          5$:     MOV     #FMTDPB,R0      ;RESET THE DPB BLOCK
3711  012224  012760  160000  000004          MOV     #-8192.,$WRDM(R0)  ;FULL TRACK WORD COUNT
3712  012232  105060  000010                  CLRB    $SEC(R0)        ;RESET TO SECTOR 0,TRACK NOT CHANGED
3713  012236  004537  010354          6$:     JSR     R5,OFFST        ;CALL OFFSET
3714  012242  000443                          BR      10$             ;BRANCH IF OFFSET FAILS
3715  012244  004737  010606                  JSR     PC,MAKEUP       ;EXECUTE READ DATA IN OFFSET MODE
3716  012250  005737  037776          7$:     TST     FMTDPB+16       ;OFFSET READ IS DONE ?
3717  012254  001775                          BEQ     7$              ;BRANCH IF NOT
3718  012256  012700  037760                  MOV     #FMTDPB,R0      ;REPORT IF ANY ERROR
3719  012262  004737  013566                  JSR     PC,PROCES
3720  012266  004737  011144                  JSR     PC,LABAD        ;LOCATE THE START AND STOP ADDRESSES
3721  012272  005760  000016                  TST     16(R0)          ;ANY ERROR ?
3722  012276  100011                          BPL     8$              ;BRANCH, IF NONE
3723  012300  004737  023544                  JSR     PC,RMINIT       ;INITIAL THE SYSTEM
3724  012304  012737  177777  023466          MOV     #-1,SAVEFG
3725  012312  012737  177777  023470          MOV     #-1,SEEKFG
3726  012320  000414                          BR      10$             ;NOT UPDATE THE SCORE
3727                                  ;       CMP     CMSEC,STARSC    ;START AND STOP SECTOR ADDRESSES IS O.K  '
3728                                  ;       BLOS    9$              ;BRANCH IF NOT
3729                                  ;       DEC     CMSEC           ;END SECTOR IS ERROR SECTOR
3730  012322  004737  010724          8$:     JSR     PC,SCORE        ;UPDATE THE SCORE
3731  012326  005760  000022          9$:     TST     $RMWC(R0)       ;WORD COUNT IS 0
3732  012332  001407                          BEQ     10$             ;BRANCH IF IT IS
3733  012334  116060  000026  000010          MOVB    $RMDA(R0),$SEC(R0)  ;UPDATE SECTOR ADDRESS
3734  012342  016060  000022  000004          MOV     $RMWC(R0),$WRDM(R0) ;UPDATE WORD COUNT
3735  012350  000732                          BR      6$              ;LOOPING UNTIL CURRENT TRACK IS DONE
```

```
3736  012352  062760  000200  000012  10$:    ADD     #128.,$CYL(R0)  ;ADJUST CYLINDER TO NEXT ZONE
3737  012360  022760  001377  000012          CMP     #767.,$CYL(R0)          ;ALL 6 ZONES ARE DONE ?
3738  012366  103402                          BLO     11$             ;BRANCH IF ALL DONE
3739  012370  000137  012106                  JMP     1$
3740  012374  013703  001266          11$:    MOV     $DEVM,R3        ;LOCATE STARTING CYLINDER
3741  012400  062703  000160                  ADD     #112.,R3        ;
3742  012404  010360  000012                  MOV     R3,$CYL(R0)     ;STARTING CYLINDER
3743  012410  105260  000011                  INCB    $TRK(R0)        ;TO NEXT SURFACE
3744  012414  122760  000005  000011          CMPB    #5,$TRK(R0)     ;ALL 5 SRUFACE ARE DONE
3745  012422  101402                          BLOS    12$             ;BRANCH IF ALL DONE
3746  012424  000137  012106                  JMP     1$              ;LOOPING UNTIL CURRENT TRACK IS DONE
3747  012430  005737  002640          12$:    TST     OFFCOD          ;OFFSET CODE = POSITIVE ?
3748  012434  001005                          BNE     13$             ;IF EQUAL,THEN ALL DONE
3749  012436  012737  000001  002640          MOV     #1,OFFCOD       ;SET POSITIVE OFFSET FLAG
3750  012444  000137  012026                  JMP     LOOP3
3751  012450  000240          13$:    NOP                             ;TO NEXT TEST
3752
3753
3754
3755
3756                                  ;TEST11  TEST 9
3757                                  ;REPORT THE TEST SCORES
3758                                  ;DIRECT THE OPERATOR TO CHANGE PACK AND MOUNT TO OTHER DIRVE
3759                                  ;
3760                                  ;$DEVM    :  LOGICAL DRIVE #, SHOULD NOT BE UPDATED BEFORE THE REPORT IS COMPLETED.
3761                                  ;
3762                                  ;BADSEC.        ACCEPTANCE FLAG, BADSEC = 0, ALL DRIVES ARE COMPATIBLE
3763                                  ;                                BADSEC = -1, DRIVES ARE NOT COMPATIBLE
3764                                  ;CMTRK :    TRACK (HEAD) NUMBER FOR CONTROLLING THE SCORE TYPING.
3765                                  ;
3766
3767
3768
3769                                  ;;*****************************************************************
3770  012452  000004          TST11:  SCOPE
3771  012454  012737  000001  001174          MOV     #1,$TIMES       ;;DO 1 ITERATION
3772  012462  012737  000011  001336          MOV     #11,TSTNM       ;LOAD THE TEST NUMBER
3773  012470  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
3774                                                                  ;ADJUST THE READ COMPATIBLE TEST SCORE
3775                                                                  ;IF THE SELF READ TEST SCORE IS TOO LOW
3776  012474  005046                          CLR     -(SP)           ;DUMMY SCORE BOARD ADDRESSES
3777  012476  005046                          CLR     -(SP)           ;
3778  012500  012746  002547                  MOV     #SELF0+1,-(SP)  ;POSITIVE OFFSET READ TEST SCORE ADDRESS
3779  012504  012746  002426                  MOV     #RDP0,-(SP)     ;POSITIVE OFFSET TEST SCORE
3780  012510  012702  002546                  MOV     #SELF0,R2       ;NEGATIVE OFFSET READ TEST SCORE
3781  012514  012703  002306                  MOV     #RDN0,R3        ;NEGATIVE OFFSET READ COMPATIBLE TEST SCORE
3782  012520  012701  000005          1$:     MOV     #5,R1           ;R1= SURFACE (TRACK) NUMBER
3783  012524  122712  000007          2$:     CMPB    #7,(R2)         ;SELF READ TEST SCORE IS TOO LOW
3784  012530  101411                          BLOS    4$              ;BRANCH IF NOT
3785  012532  012705  000020                  MOV     #16.,R5         ;INCREMENT READ COMPATIBLE SCORE FOR ALL 16 DIRVES
3786  012536  111304          3$:     MOVB    (R3),R4         ;ADJUST SCORE BY ADDING 6
3787  012540  062704  000006                  ADD     #6,R4
3788  012544  110423                          MOVB    R4,(R3)+        ;UPDATE SCORE AND POINTS TO NEXT DRIVE
3789  012546  005305                          DEC     R5              ;ALL DRIVES ARE UPDATED ?
3790  012550  001372                          BNE     3$              ;BRANCH IF NOT
3791  012552  000402                          BR      5$
```

```
3792  012554  062703  000020          4$:     ADD     #16.,R3         ;ADJUST POINTER OF SCORE BOARD ADDRESS
3793  012560  062702  000002          5$:     ADD     #2,R2           ;UPDATE THE SELF TEST SCORE ADDRESS
3794  012564  005301                  DEC     R1              ;ALL FIVE SURFACES ARE DONE ?
3795  012566  001356                  BNE     2$              ;BRANCH IF NOT
3796  012570  012603                  MOV     (SP)+,R3        ;GET NEXT PAIR
3797  012572  012602                  MOV     (SP)+,R2        ;EXIT THE THEY ARE 0
3798  012574  001351                  BNE     1$
3799                                                          ;
3800                                                          ;  SET ACCEPTANCE FLAG,INITIALIZE THE
3801                                                          ;  TABLE POINTERS AND TRACK NUMBER
3802  012576  005037  001340          CLR     BADSEC          ;SET ACCEPTANCE FLAG
3803  012602  005037  001362          CLR     CMTRK           ;START FROM TRACK 0
3804
3805  012606  005001                  CLR     R1              ;START FROM LOG DRV 0
3806  012610  012702  002046          MOV     #OVWNO,R2       ;SCORE BOARD BASE ADDRESS
3807  012614  012703  000001          MOV     #BIT0,R3        ;BIT POSITION FOR LOG DRV 0
3808  012620  030337  001776  LOOP4:  BIT     R3,ASNLST       ;IS THE LOG DRV UNDER TEST ?
3809  012624  001502                  BEQ     LOOP5           ;BRANCH IF NOT
3810  012626  005037  001364          CLR     NULINE          ;NEW LINE INDICATOR
3811  012632  123701  001266          CMPB    $DEVM,R1        ;SELF SCORE ?
3812  012636  001434                  BEQ     SPATH           ;BRANCH IF IT IS
3813  012640  122712  000016  KPATH:  CMPB    #14.,(R2)               ;OVERWRITE NEG OFFSET < 14 ?
3814  012644  101403                  BLOS    1$              ;BRANCH IF NOT
3815  012646  004537  013270          JSR     R5,PRINT        ;REPORT EXCEPTIONS
3816  012652  000001                  000001                  ;COLUMN POSITION ON REPORT
3817  012654  122762  000016  000120  1$:     CMPB    #14.,80.(R2)    ;OVERWRITE POS OFFSET < 14 ?
3818  012662  101403                  BLOS    2$              ;BRANCH IF NO:
3819  012664  004537  013270          JSR     R5,PRINT        ;REPORT EXECPTIONS
3820  012670  000002                  000002                  ;COLUMN POSITION ON REPORT
3821  012672  122762  000014  000240  2$:     CMPB    #12.,160.(R2)   ;READ NEG OFFSET < 6
3822  012700  101403                  BLOS    3$              ;BRANCH IF NOT
3823  012702  004537  013270          JSR     R5,PRINT        ;REPORT EXCEPTIONS
3824  012706  000003                  000003                  ;COLUMN POSITION ON REPORT
3825  012710  122762  000014  000360  3$:     CMPB    #12.,240.(R2)   ;READ POS OFFSET < 12 ?
3826  012716  101445                  BLOS    LOOP5           ;BRANCH IF NOT
3827  012720  004537  013270          JSR     R5,PRINT        ;REPORT EXECPTIONS
3828  012724  000004                  000004                  ;COLUMN POSITION ON REPORT
3829  012726  000441                  BR      LOOP5           ;EXIT
3830  012730  122712  000016  SPATH:  CMPB    #14.,(R2)               ;OVERWRITE NEG OFF < 14
3831  012734  101403                  BLOS    1$              ;BRANCH IF NOT
3832  012736  004537  013270          JSR     R5,PRINT        ;REPORT EXECPTION
3833  012742  000001                  000001                  ;COLUMN POSITION ON REPORT
3834  012744  122762  000016  000120  1$:     CMPB    #14.,80.(R2)    ;OVERWRITE POS OFFSET < 14
3835  012752  101403                  BLOS    2$              ;BRANCH IF NOT
3836  012754  004537  013270          JSR     R5,PRINT        ;REPORT EXCEPTION
3837  012760  000002                  000002                  ;COLUMN POSITION ON REPORT
3838  012762  013704  001362  2$:     MOV     CMTRK,R4        ;LOCATE THE SELF TEST SCORE
3839  012766  006304                  ASL     R4              ;WORD INDEX
3840  012770  122764  000006  002546  CMPB    #6,SELF0(R4)    ;SELF READ NEG OFFSET < 6 ?
3841  012776  101403                  BLOS    3$              ;BRANCH IF NOT
3842  013000  004537  013270          JSR     R5,PRINT
3843  013004  000003                  000003                  ;REPORT EXCEPTIONS
3844  013006  013704  001362  3$:     MOV     CMTRK,R4        ;LOCATE THE SELF TEST SCORE
3845  013012  006304                  ASL     R4              ;WORD INDEX
3846  013014  122764  000006  002547  CMPB    #6,SELF0+1(R4)  ;SELF READ POS OFFSET < 6 ?
3847  013022  101403                  BLOS    LOOP5           ;BRANCH IF NOT
```

```
3848  013024  004537  013270            JSR     R5,PRINT        ;REPORT EXCEPTIONS
3849  013030  000004                    000004                  ;COLUMN POSITION FOR REPORT PRINTING
3850  013032  005201            LOOP5:  INC     R1              ;INCREASE THE LOGICAL DRIVE #
3851  013034  000241                    CLC                     ;ADJUST THE BIT POSITION
3852  013036  006103                    ROL     R3              ;POINTS TO NEXT DRIVE
3853  013040  005202                    INC     R2              ;UPDATE SCORE BOARD BASE ADDRESS
3854  013042  022701  000017            CMP     #15.,R1         ;ALL DRIVES ARE CHECK ?
3855  013046  103264                    BHIS    LOOP4           ;BRANCH IF NOT ALL DONE
3856  013050  005237  001362    LOOP6:  INC     CMTRK           ;NEXT SURFACE
3857  013054  012702  002046            MOV     #OVWND,R2       ;RESET SCORE BOARD BASE ADDRESS
3858  013060  005001                    CLR     R1              ;LOGICAL DRIVE START FROM 0
3859  013062  013703  001362            MOV     CMTRK,R3        ;LOCATE THE SCORE BOARD BASE ADDRESS
3860  013066  001404                    BEQ     2$              ;BRANCH IF ON SURFACE 0
3861  013070  062702  000020    1$:     ADD     #16.,R2         ;FOR EACH SURFACE,16 BYTES
3862  013074  005303                    DEC     R3              ;ALL SURFACE DONE ?
3863  013076  001374                    BNE     1$              ;BRANCH IF NOT
3864  013100  012703  000001    2$:     MOV     #BIT0,R3                ;BIT MAP POSITION FOR DRIVE 0
3865                                                            ;AT THIS POINT:
3866                                                            ;R1=0,LOGICAL DRIVE # START FROM 0
3867                                                            ;R2 ADDRESS OF SCORE BOARD BA'E ADDRESS
3868                                                            ;R3=1,BIT MAP ,INDICATE LOGICAL DRIVE 0
3869  013104  122737  000005  001362    CMPB    #5,CMTRK        ;ALL SURFACES ARE DONE ?
3870  013112  101242                    BHI     LOOP4           ;CONTINUE
3871  013114  005737  001340            TST     BADSEC          ;ACCEPTANCE FLAG SET ?
3872  013120  100402                    BMI     DISMNT          ;NO,THEN BRANCH
3873  013122  104401  001205            TYPE    ,$CRLF
3874                                    TYPE    ,MESG15 ;** ALL DRIVES ARE COMPATIBLE **
3875  013126  104401  001205    DISMNT: TYPE    ,$CRLF          ;CR-LF
3876  013132  012777  013560  010370    MOV     #IDLEX,@RMVEC   ;RESET THE INTERRUPT VECTOR
3877  013140  005077  010366            CLR     @RMVEC+2        ;CLEAR THE INTERRUPT VECTOR
3878  013144  104401  036001            TYPE    ,MESG12         ;UNLOAD AND DISMOUNT MESSAGE
3879  013150  104401  001272            TYPE    ,$CDW2          ;SYSTEM NAME
3880  013154  013746  001222            MOV     DRIVE,-(SP)     ;PHYSICAL DRIVE NUMBER
3881  013160  104403                    TYPOS
3882  013162     002                    .BYTE   2
3883  013163     000                    .BYTE   0
3884  013164  104401  036017            TYPE    ,MESG13         ;MESSAGE TYPE R<CR> WHEN READY
3885  013170  104411            1$:     RDLIN                   ;READ IN ONE LINE
3886  013172  012605                    MOV     (SP)+,R5        ;LOCATE THE READ IN LINE
3887  013174  122527  000122            CMPB    (R5)+,#'R       ;READY ?
3888  013200  001373                    BNE     1$              ;TRY AGAIN IF NOT
3889  013202  105715                    TSTB    (R5)            ;TERMINATOR ?
3890  013204  001371                    BNE     1$              ;BRANCH IF NOT
3891  013206  005002            DEASG:  CLR     R2              ;LOGICAL DRIVE NUMBER
3892  013210  012703  000001            MOV     #BIT0,R3        ;BIT MAP OF ASSIGNED DRIVE
3893  013214  020237  001266    1$:     CMP     R2,$DEVM        ;IS THE LOGICAL DRIVE UNDER TEST ?
3894  013220  001404                    BEQ     2$              ;BRANCH IF IT IS
3895  013222  000241                    CLC                     ;SHIFT THE BIT MAP FOR
3896  013224  006103                    ROL     R3              ;NEXT DRIVE
3897  013226  005202                    INC     R2              ;INCREMENT THE LOGICAL DRIVE #
3898  013230  000771                    BR      1$              ;LOOPING ,UNTIL THE LOGICAL DRIVE LOCATED
3899  013232  040337  002002    2$:     BIC     R3,ASSGN2       ;CLEAR THE ASSIGNED BIT
3900  013236  001412                    BEQ     XEND2           ;BRANCH IF NO MORE DRIVES
3901  013240  005237  001266            INC     $DEVM           ;INCREMENT THE LOGICAL DRIVE #
3902  013244  013702  001266            MOV     $DEVM,R2        ;UPDATE SYSTEM BLOCK ADDRESS
3903  013250  006302                    ASL     R2
```

```
3904  013252  016237  002600  001270          MOV     BLKADR(R2),$CDW1        ;SYSTEM BLOCK ADDRESS
3905  013260  000137  007172                  JMP     TST5                    ;JUMP TO TEST 5 FOR OTHER DRIVE
3906
3907
3908  013264  000137  017654          XEND2:  JMP     $EOP    ;END OF PASS
3909
3910
3911                                    ;PRINT ROUTINE
3912                                    ;NAME PRINT
3913                                    ;PRINT EXCEPTIONS FOR TEST SCORE
3914                                    ;PARAMETER USED
3915                                    ;   NULINE = 0          A NEW LINE TO PRINT
3916                                    ;          = i - 4      COLUMN NUMBER TO PRINT THE EXCEPTION MARK
3917                                    ;
3918                                    ;   BADSEC = 0          FIRST EXCEPTION DETECTED
3919                                    ;    BADSEC = -1        NOT FIRST EXCEPTION (DON'T PRINT THE TITLE)
3920                                    ;
3921                                    ;       CALLING SEQ:
3922                                    ;       JSR     R5,PRINT
3923                                    ;       NUMBER                  COLUMN NUMBER
3924                                    ;       RET
3925                                    ;
3926                                    ;       R1: LOGICAL DRIVE #
3927                                    ;       $DEVM: LOGICAL DRIVE UNDER TEST
3928                                    ;       R3: BIT POSITION OF LOGICAL DRIVE IN R1
3929                                    ;
3930
3931
3932
3933  013270                    PRINT:
3934  013270  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3935  013272  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3936  013274  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3937  013276  010446                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3938  013300  005737  001340          TST     BADSEC          ;FIRST EXCEPTION
3939  013304  100432                  BMI     1$              ;BRANCH IF NOT,DON'T HAVE TO PRINT
3940                                                          ;TITLE
3941  013306  104401  001205          TYPE    ,$CRLF          ;CRLF
3942  013312  104401  036161          TYPE    ,MESG16         ;SCORES FOR DRIVE --
3943  013316  104401  001272          TYPE    ,$CDW2          ;SYSTEM NAME
3944  013322  013746  001222          MOV     DRIVE,-(SP)     ;TYPE THE PHYSICAL DRIVE #
3945  013326  104403                  TYPOS
3946  013330     002                  .BYTE   2
3947  013331     000                  .BYTE   0
3948  013332  104401  001205          TYPE    ,$CRLF          ;CR-LF
3949  013336  104401  036204          TYPE    ,MESG17         ;SUB TITLE 1
3950  013342  104401  001205          TYPE    ,$CRLF
3951  013346  104401  036300          TYPE    ,MESG18         ;SUB TITLE 2
3952  013352  104401  001205          TYPE    ,$CRLF
3953  013356  012737  177777  001340  MOV     #-1,BADSEC      ;RESET THE ACCEPTANCE FLAG
3954  013364  012737  000001  001374  MOV     #1,FAULT        ;NOT COMPATIBLE FLAG
3955  013372  005737  001364      1$: TST     NULINE          ;NEW LINE ?
3956  013376  001045                  BNE     5$              ;BRANCH IF NOT
3957  013400  104401  001205          TYPE    ,$CRLF
3958  013404  104401  001205          TYPE    ,$CRLF
3959  013410  013746  001362          MOV     CMTRK,-(SP)     ;TYPE THE SURFACE NUMBER
```

```
3960  013414  104403                          TYPOS
3961  013416     002                          .BYTE   2
3962  013417     000                          .BYTE   0
3963  013420  104401  036403                 TYPE    ,TAB            ;TYPE "TABLE" CHARACTER
3964  013424  123701  001266                 CMPB    $DEVM,R1        ;SCORE FOR $DEVM ITSELF ?
3965  013430  001005                         BNE     2$              ;BRANCH IF NOT
3966  013432  104401  036375                 TYPE    ,SELFX          ;MESSAGE: SELF  .
3967  013436  104401  036403                 TYPE    ,TAB
3968  013442  000420                         BR      4$              ;NEXT STEP
3969  013444  006301              2$:        ASL     R1              ;LOCATE THE SYS HISTORY FILE
3970  013446  016137  002600  013476         MOV     BLKADR(R1),@#3$ ;LOCATE THE SYSTEM NAME AND DRIVE #
3971  013454  006201                         ASR     R1              ;RESTORE DRIVE #
3972  013456  062737  000014  013476         ADD     #$SYSNM,@#3$    ;LOCATE THE SYSTEM NAME AND DRIVE #
3973  013464  104401  036604                 TYPE    ,LINSP          ;
3974  013470  104401  036604                 TYPE    ,LINSP          ;
3975  013474  104401                         TYPE                    ;TYPE THE SYSTEM AND DRIVE
3976  013476  000000              3$:        .WORD   0               ;ADDRESS FOR TYPING MESSAGE
3977  013500  104401  036403                 TYPE    ,TAB            ;
3978  013504  012737  000001  001364  4$:    MOV     #1,NULINE       ;INDICATE PRINTER STOPS AT COULMN 1
3979  013512  012504              5$:        MOV     (R5)+,R4        ;RETRIEVE THE DESIRED COLUMN # FROM
3980                                                                  ;CALLING ROUTINE
3981  013514  020437  001364              6$: CMP     R4,NULINE       ;ON THE RIGHT COLUMN ?
3982  013520  103412                         BLO     8$              ;IF LOW NOT PRINT
3983  013522  001405                         BEQ     7$              ;BRANCH IF LOCATED
3984  013524  104401  036403                 TYPE    ,TAB            ;ADVANCE TO NEXT COLUMN
3985  013530  005237  001364                 INC     NULINE          ;INCREMENT COLUMN CTR
3986  013534  000767                         BR      6$              ;CHECK AGAIN
3987  013536  104401  036405              7$: TYPE    ,MARKX          ;THE EXCEPTION MARK * 0
3988  013542  005237  001364                 INC     NULINE          ;NEXT COLUMN
3989  013546                              8$:
3990  013546  012604                         MOV     (SP)+,R4        ;;POP STACK INTO R4
3991  013550  012603                         MOV     (SP)+,R3        ;;POP STACK INTO R3
3992  013552  012602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
3993  013554  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
3994  013556  000205                         RTS     R5              ;EXIT
3995
3996
3997
3998  013560  000240              IDLEX:     NOP                     ;RESET ALL RH VECTOR FOR MOUNT AND DISMOUNT
3999  013562  000240                         NOP
4000  013564  000002                         RTI                     ;EXIT
4001                              ;PROCESS THE ORDER TERMINATION
4002
4003  013566  111037  001326      PROCES:    MOVB    (R0),UNIT       ;DRIVE NUMBER FOR ANY ERROR MESSAGES
4004  013572  005760  000016                 TST     $TATUS(R0)      ;SEE IF DRIVER SIGNALED AN ERROR
4005  013576  100431                         BMI     ERPROC          ;BR IF ERROR
4006  013600  032760  100000  000020         BIT     #BIT15,$RMCS1(R0)  ;SEE IF 'SC' SET
4007  013606  001410                         BEQ     1$              ;BR IF NOT SET
4008  013610  032760  040000  000020         BIT     #BIT14,$RMCS1(R0)  ;SEE IF 'TRE' SET
4009  013616  001021                         BNE     ERPROC          ;BR IF SET
4010  013620  032760  040000  000032         BIT     #BIT14,$RMDS(R0)  ;SEE IF 'ERR' SET
4011  013626  001015                         BNE     ERPROC          ;BR IF SET
4012  013630  004737  014712              1$: JSR     PC,CKERR        ;NO ERROR, CHECK ERROR BITS ANYWAY
4013  013634  004737  014760                 JSR     PC,CKBUS        ;NO ERROR, CHECK BUS ADDR & WC
4014  013640  004737  023544                 JSR     PC,RMINIT       ;INITIALIZE THE SUB SYSTEM
4015  013644  012737  177777  023466         MOV     #-1,SAVEFG      ;CLEAR THE SAVE FLAG
```

```
4016  013652  012737  177777  023470          MOV     #-1,SEEKFG      ;SET THE NOT IMP FLAG
4017  013660  000207          2$:     RTS     PC              ;RETURN
4018
4019                          ;ORDER TERMINATED WITH AN ERROR - PROCESS THE ERROR
4020
4021  013662  032760  000200  000016  ERPROC: BIT     #BIT07,$TATUS(R0) ;DONE BIT SET ?
4022  013670  001402                          BEQ     ERPRC1          ;BR IF ORDER DIDN'T COMPLETE NORMALLY
4023  013672  000137  014062                  JMP     DONE            ;PROCESS ERROR WITH 'DONE' BIT SET
4024
4025                          ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE NOT' BITS
4026
4027  013676  032760  010000  000016  ERPRC1: BIT     #BIT12,$TATUS(R0) ;SEE IF DRIVE WAS UNSAFE
4028  013704  001025                          BNE     PUNSAF          ;BR IF YES
4029  013706  032760  004000  000016          BIT     #BIT11,$TATUS(R0) ;PARITY ERROR OCCURRED
4030  013714  001025                          BNE     UCPAR           ;BR IF IT DID
4031  013716  032760  002000  000016          BIT     #BIT10,$TATUS(R0) ;FATAL PARITY ERROR?
4032  013724  001025                          BNE     FALPAR          ;BR IF THERE IS ONE
4033  013726  032760  001000  000016          BIT     #BIT09,$TATUS(R0) ;TIMEOUT?
4034  013734  001025                          BNE     SWTIM           ;BR IF YES
4035  013736  032760  040002  000016          BIT     #BIT14'BIT01,$TATUS(R0)  ;DRIVE WENT OFFLINE ?
4036  013744  001025                          BNE     OFLIN           ;BR IF IT DID
4037  013746  032760  000004  000016          BIT     #BIT2,$TATUS(R0)  ;PORT REQUEST TIME OUT ?
4038  013754  001025                          BNE     PRTIM           ;BR IF IT DID
4039  013756  000207                          RTS     PC              ;ERROR. RETURN
4040
4041                          ;DRIVE IS PERSISTENTLY UNSAFE
4042
4043  013760                  PUNSAF:
4044  013760  104414  032674                  DISPLY  ,EM12
4045  013764  000137  014034                  JMP     DUMP2
4046
4047                          ;UNCORRECTABLE MASSBUS PARITY ERROR OCCURRED
4048
4049  013770                  UCPAR:
4050  013770  104414  032576                  DISPLY  ,EM10
4051  013774  000137  014034                  JMP     DUMP2
4052
4053                          ;'FATAL' MASSBUS PARITY ERROR OCCURRED
4054
4055  014000                  FALPAR:
4056  014000  104414  032641                  DISPLY  ,EM11
4057  014004  000137  014034                  JMP     DUMP2
4058
4059                          ;SOFTWARE TIMEOUT OCCURRED
4060
4061  014010                  SWTIM:
4062  014010  104414  032725                  DISPLY  ,EM13
4063  014014  000137  014034                  JMP     DUMP2
4064
4065                          ;DRIVE WENT OFFLINE
4066
4067  014020                  OFLIN:
4068  014020  104414  032777                  DISPLY  ,EM14
4069  014024  000137  014034                  JMP     DUMP2
4070                          ;PORT REQUEST TIMEOUT ERROR
4071
```

```
 4072  014030                          PRTIM:
 4073  014030  104414  033022                DISPLY  ,EM15
 4074  014034  104401  001205          DUMP2:TYPE    ,$CRLF
 4075  014040  104401  037303                TYPE    ,XFATL
 4076  014044  104401  001205                TYPE    ,$CRLF          ;CR-LF
 4077  014050  104401  037152                TYPE    ,HALTX
 4078  014054  000000                       HALT
 4079  014056  000137  000200                JMP     @#200
 4080
 4081                                  ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE' BITS SET
 4082
 4083  014062  032760  000030  000016 DONE:  BIT     #BIT04!BIT03,$STATUS(R0) ;UNSAFE OCCURRED
 4084  014070  001402                       BEQ     .+6             ;BR IF NOT
 4085  014072  000137  014650                JMP     UNSAF           ;REPORT UNSAFE
 4086  014076  032760  040000  000030       BIT     #BIT14,$RMCS2(R0)  ;IS 'WCE' SET ?
 4087  014104  001402                       BEQ     .+6             ;BRANCH IF NOT SET
 4088  014106  000137  014430                JMP     WCKER           ;WRITE CHECK ERROR
 4089  014112  032760  040000  000032       BIT     #BIT14,$RMDS(R0) ;CHECK 'ERR'
 4090  014120  001002                       BNE     1$              ;BR IF SET
 4091  014122  000137  014620                JMP     TRFER           ;PROCESS 'TRE'
 4092  014126  032760  000400  000034 1$:    BIT     #BIT08,$RMER1(R0) ;'HCRC' SET?
 4093  014134  001402                       BEQ     .+6             ;BR IF NOT
 4094  014136  000137  014460                JMP     HCRCER          ;PROCESS 'HCRC'
 4095  014142  032760  000020  000034       BIT     #BIT04,$RMER1(R0) ;'FMT' SET?
 4096  014150  001402                       BEQ     .+6             ;BR IF NOT SET
 4097  014152  000137  014500                JMP     CKFMT           ;CHECK FORMAT ERROR
 4098  014156  032760  000200  000034       BIT     #BIT07,$RMER1(R0) ;'HCE' SET?
 4099  014164  001402                       BEQ     .+6             ;BR IF NOT SET
 4100  014166  000137  014510                JMP     CKHCE           ;CHECK 'HCE' ERROR
 4101  014172  032760  020000  000034       BIT     #BIT13,$RMER1(R0) ;'OPI' SET?
 4102  014200  001402                       BEQ     .+6             ;BR IF NOT SET
 4103  014202  000137  014530                JMP     OPIER           ;REPORT 'OPI'
 4104  014206  032760  000010  000034       BIT     #BIT3,$RMER1(R0) ;'PAR' SET?
 4105  014214  001402                       BEQ     .+6             ;BR IF NOT SET
 4106  014216  000137  014550                JMP     PARER           ;REPORT 'PAR'
 4107  014222  032760  000040  000034       BIT     #BIT5,$RMER1(R0) ;'WCF' SET?
 4108  014230  001402                       BEQ     .+6             ;BR IF NOT SET
 4109  014232  000137  014640                JMP     WCFER           ;REPORT 'WCF'
 4110  014236  032760  002000  000034       BIT     #BIT10,$RMER1(R0) ;'IAE' SET?
 4111  014244  001402                       BEQ     .+6             ;BR IF NOT SET
 4112  014246  000137  014560                JMP     IAEER           ;REPORT 'IAE'
 4113  014252  032760  004000  000034       BIT     #BIT11,$RMER1(R0) ;'WLE' SET?
 4114  014260  001402                       BEQ     .+6             ;BR IF NOT SET
 4115  014262  000137  014570                JMP     WLEER           ;REPORT 'WLE'
 4116  014266  032760  001000  000034       BIT     #BIT9,$RMER1(R0) ;'AOE' SET?
 4117  014274  001405                       BEQ     2$              ;BR IF NOT SET
 4118  014276  032760  002000  000032       BIT     #BIT10,$RMDS(R0) ;'LST' SET?
 4119  014304  001401                       BEQ     2$              ;BR IF NOT SET
 4120  014306  000207                       RTS     PC              ;'AOE' & 'LST' SET, EXIT
 4121  014310  032760  010000  000034 2$:    BIT     #BIT12,$RMER1(R0)  ;SEE IF 'DTE' SET
 4122  014316  001402                       BEQ     .+6             ;BR IF NOT
 4123  014320  000137  014540                JMP     DTEER           ;REPORT 'DTE' ERROR
 4124  014324  005760  000034                TST     $RMER1(R0)      ;SEE IF 'DCK' SET
 4125  014330  100002                       BPL     .+6             ;BR IF NOT
 4126  014332  000137  014420                JMP     DCKER           ;PROCESS 'DCK'
 4127  014336  032760  060000  000062       BIT     #BIT14!BIT13,$RMER2(R0) ;'SKI' OR 'DCYL' SET
```

```
4128  014344  001006                        BNE     3$                      ;BR IF IT IS
4129  014346  032760  100000  000062        BIT     #BIT15,$RMER2(R0)       ;BAD SPOT ?
4130  014354  001004                        BNE     4$                      ;BRANCH IF SO
4131  014356  000137  014470                JMP     DRVER                   ;REPORT ERROR
4132  014362  000137  014630        3$:     JMP     SKIER                   ;REPORT DRIVE ERROR
4133  014366  104401  001205        4$:     TYPE    ,$CRLF
4134  014372  104401  035723                TYPE    ,MESG11
4135  014376  104007                        ERROR   7
4136  014400  104010                        ERROR   10
4137  014402  104011                        ERROR   11
4138  014404  104012                        ERROR   12
4139  014406  104401  037152                TYPE    ,HALTX
4140  014412  000000                        HALT
4141  014414  000137  000200                JMP     @#200
4142
4143                                ;PROCESS DATA ('DCK') CHECK ERROR
4144
4145  014420                        DCKER:
4146  014420  104414  033077                DISPLY  ,EM21
4147  014424  000137  014660                JMP     DUMP
4148
4149                                ;WRITE CHECK ERROR PROCESSING
4150
4151  014430                        WCKER:
4152  014430  032760  100000  000034        BIT     #BIT15,$RMER1(R0)       ;DCK BIT SET ?
4153  014436  001004                        BNE     1$                      ;BRANCH IF SET
4154  014440  104414  033203                DISPLY  ,EM23
4155  014444  000137  014660                JMP     DUMP
4156  014450  104414  033130        1$:     DISPLY  ,EM22
4157  014454  000137  014660                JMP     DUMP
4158
4159                                ;REPORT 'HCRC' ERROR
4160
4161  014460                        HCRCER:
4162  014460  104414  033056                DISPLY  ,EM20
4163  014464  000137  014660                JMP     DUMP
4164
4165                                ;REPORT DRIVE ERROR
4166
4167  014470                        DRVER:
4168  014470  104414  033473                DISPLY  ,EM30
4169  014474  000137  014660                JMP     DUMP
4170
4171                                ;PROCESS FORMAT ('FER') ERROR
4172
4173  014500                        CKFMT:
4174  014500  104414  033262                DISPLY  ,EM24
4175  014504  000137  014660                JMP     DUMP
4176
4177                                ;PROCESS HEADER COMPARE ('HCE') ERROR
4178
4179  014510                        CKHCE:
4180  014510  104414  033330                DISPLY  ,EM25
4181  014514  000137  014660                JMP     DUMP
4182                                ;POSSIBLE POSITIONING ERROR
4183
```

```
4184  014520  104414  034631          POSER:  DISPLY  ,EM51
4185  014524  000137  014660                  JMP     DUMP
4186                                  ;REPORT 'OPI' ERROR
4187  014530                          OPIER:
4188  014530  104414  033525                  DISPLY  ,EM31
4189  014534  000137  014660                  JMP     DUMP
4190                                  ;REPORT 'DTE' ERROR
4191
4192  014540                          DTEER:
4193  014540  104414  033570                  DISPLY  ,EM32
4194  014544  000137  014660                  JMP     DUMP
4195                                  ;REPORT 'PAR' ERROR
4196
4197  014550                          PARER:
4198  014550  104414  033623                  DISPLY  ,EM33
4199  014554  000137  014660                  JMP     DUMP
4200                                  ;REPORT 'IAE' ERROR
4201
4202  014560                          IAEER:
4203  014560  104414  033742                  DISPLY  ,EM35
4204  014564  000137  014660                  JMP     DUMP
4205
4206                                  ;REPORT WLE ERROR
4207
4208  014570                          WLEER:
4209  014570  104414  034000                  DISPLY  ,EM36
4210  014574  000137  014660                  JMP     DUMP
4211                                  ;REPORT FORMAT ERROR
4212
4213  014600                          FMTER:
4214  014600  104414  033411                  DISPLY  ,EM26
4215  014604  000137  014660                  JMP     DUMP
4216                                  ;REPORT HEADER COMPARE ERROR
4217
4218  014610  104414  033436          HCEER:  DISPLY  ,EM27
4219  014614  000137  014660                  JMP     DUMP
4220
4221                                  ;PROCESS CONTROL/INTERFACE TRANSFER ERROR
4222
4223  014620                          TRFER:
4224  014620  104414  034113                  DISPLY  ,EM40
4225  014624  000137  014660                  JMP     DUMP
4226
4227                                  ;PROCESS 'SKI' OR 'OCYL'
4228
4229  014630                          SKIER:
4230  014630  104414  034573                  DISPLY  ,EM50
4231  014634  000137  014660                  JMP     DUMP
4232                                  ;REPORT WRITE CLOCK FAILURE
4233
4234  014640                          WCFER:
4235  014640  104414  033700                  DISPLY  ,EM34
4236  014644  000137  014660                  JMP     DUMP
4237
4238                                  ;REPORT DRIVE UNSAFE ERROR
4239
```

```
4240  014650                          UNSAF:
4241  014650  104414  034674                  DISPLY  ,EM60
4242  014654  000137  014660                  JMP     DUMP
4243
4244
4245  014660                          DUMP:
4246  014660  104007                          ERROR   7
4247  014662  104010                          ERROR   10
4248  014664  104011                          ERROR   11
4249  014666  104012                          ERROR   12
4250  014670  004737  023544                  JSR     PC,RMINIT       ;CLEAR THE SUB-SYSTEM
4251  014674  012737  177777  023466          MOV     #-1,SAVEFG
4252  014702  012737  177777  023470          MOV     #-1,SEEKFG
4253  014710  000207                          RTS     PC
4254
4255
4256                                  ;CKECK ERROR BITS IN THE RH11 AND RM03 REGSTERS
4257
4258  014712  032760  060000  000020  CKERR:  BIT     #60000,$RMCS1(R0)       ;SEE IF 'TRE' OR 'MCPE'
4259  014720  001012                          BNE     1$                      ;YES
4260  014722  032760  177400  000030          BIT     #177400,$RMCS2(R0)      ;ERROR BITS IN CS2 /
4261  014730  001006                          BNE     1$                      ;YES
4262  014732  005760  000034                  TST     $RMER1(R0)              ;ANY ERROR IN ER1
4263  014736  001003                          BNE     1$                      ;YES
4264  014740  005760  000062                  TST     $RMER2(R0)              ;ANY ERROR IN ER2
4265  014744  001404                          BEQ     2$              ;BRANCH IF NO ERROR
4266  014746  104414  034347          1$:     DISPLY  ,EM44
4267  014752  000137  014660                  JMP     DUMP            ;TYPE ALL REGISTERS
4268  014756  000207                  2$:     RTS     PC
4269
4270                                  ;CHECK BUS ADDRESS REGISTER AND WORD COUNT REGISTER
4271
4272  014760  005760  000022  CKBUS:  TST     $RMWC(R0)               ;WORD COUNT = 0
4273  014764  001011                          BNE     1$                      ;NO
4274  014766  016046  000004                  MOV     $WRDM(R0),-(SP) ;WORD LENGTH
4275  014772  005416                          NEG     (SP)                    ;GET THE POSITIVE NUMBER OF WORD COUNT
4276  014774  006316                          ASL     (SP)                    ;BYTE COUNT
4277  014776  066016  000006                  ADD     $BUF(R0),(SP)
4278  015002  022660  000024                  CMP     (SP)+,$RMBA(R0)
4279  015006  001404                          BEQ     2$
4280  015010  104414  034155          1$:     DISPLY  ,EM41
4281  015014  000137  014660                  JMP     DUMP            ;TYPE ALL REGISTERS
4282  015020  000207                  2$:     RTS     PC
4283                                  ;ROUTINE TO DISPLAY THE SECTOR WHICH GAVE THE HARD ERROR
4284
4285  015022  104401  001205  PRTBAD: TYPE    ,$CRLF
4286  015026  104401  037034          TYPE    ,MESG19
4287  015032  104401  001205          TYPE    ,$CRLF
4288  015036  016001  000024          MOV     $RMBA(R0),R1    ;PUT THE END ADDRESS INTO R1
4289  015042  016046  000004          MOV     $WRDM(R0),-(SP) ;FIND THE BEGINNING OF THE SECTOR
4290  015046  005416                  NEG     (SP)                    ;GET THE POSITIVE NUMBER OF WORD COUNT
4291  015050  066016  000022          ADD     $RMWC(R0),(SP)  ;SUBTRACT THE WORDS NOT TRANSFERED
4292  015054  005046                  CLR     -(SP)                   ;MAKE THE UPPER DIVIDEND 0
4293  015056  012746  000400          MOV     #256.,-(SP)     ;DIVDE THE WORDS TRANSFERED BY THE SECTOR SIZE
4294  015062  004737  016106          JSR     PC,LINKDV               ,DIVIDE
4295  015066  005716                  TST     (SP)                    ;REMANDER = 0 ?
```

```
4296  015070  001403                  BEQ     1$              ;BR IF IT IS - COMPLETE SECTOR TRANSFERED
4297  015072  006316                  ASL     (SP)            ;CONVERT THE RESIDUAL SECTOR SIZE INTO BYTE COUNT
4298  015074  161601                  SUB     (SP),R1         ;SUBTRACT IT FROM THE END ADDRESS
4299  015076  000402                  BR      2$              ;FINISH THE SIZING
4300  015100  162701  001000  1$:     SUB     #1000,R1        ;SUBTRACT FULL SECTOR SIZE FROM END ADDR
4301  015104  062706  000004  2$:     ADD     #4,SP           ;RESTORE THE STACK POINTER
4302  015110  012702  000007  3$:     MOV     #7,R2           ;R2 CONTAINS THE WORDS/LINE COUNT
4303  015114  020160  000024  4$:     CMP     R1,$RMBA(R0)    ;PRINTED ALL THE SECTOR ?
4304  015120  001415                  BEQ     5$              ;BR IF ALL PRINTED
4305  015122  104414  036604          DISPLY  ,LINSP          ;SPACES
4306  015126  012146                  MOV     (R1)+,-(SP)     ;PUT THE DATA ON THE STACK
4307  015130  004737  015302          JSR     PC,LINOCT       ;TYPE THE DATA
4308  015134  022711  177777          CMP     #-1,(R1)        ;END OF FILE ?
4309  015140  001405                  BEQ     5$              ;BRANCH IF SO
4310  015142  005302                  DEC     R2              ;DECREMENT THE HORIZONTAL COUNT
4311  015144  001363                  BNE     4$              ;BR IF NOT AT THE END OF THE LINE
4312  015146  104414  001205          DISPLY  ,$CRLF          ;CR-LF
4313  015152  000756                  BR      3$              ;RESTORE THE WORDS/LINE COUNT
4314  015154  104414  001205  5$:     DISPLY  ,$CRLF          ;PRINT WHAT REMAINS IN THE BUFFER
4315  015160  000207          6$:     RTS     PC              ;RETURN
4316
4317
4318
4319  015162  104412          FILBUF: SAVREG                  ;SAVE THE REGISTERS
4320  015164  016001  000006          MOV     $BUF(R0),R1     ;BUFFER ADDRESS
4321  015170  016002  000004          MOV     $WRDM(R0),R2    ;POSITIVE WORD COUNT
4322  015174  005402                  NEG     R2              ;
4323  015176  012705  002642          MOV     #STNDAT,R5      ;PATTERN ADDRESS
4324  015202  012703  000020          MOV     #20,R3          ;PATTERN COUNT
4325  015206  012521          3$:     MOV     (R5)+,(R1)+     ;MOVE THE PATTERN INTO THE BUFFER
4326  015210  005302                  DEC     R2              ;DECREMENT THE WORD COUNT
4327  015212  003407                  BLE     4$              ;BR IF DONE (WORD COUNT = 0)
4328  015214  005303                  DEC     R3              ;DECREMENT THE PATTERN COUNT
4329  015216  001373                  BNE     3$              ;BR IF MORE PATTERN
4330  015220  012703  000020          MOV     #20,R3          ;RESTORE PATTERN COUNT
4331  015224  012705  002642          MOV     #STNDAT,R5      ;RESTORE THE ADDRESS
4332  015230  000766                  BR      3$              ;CONTINUE DISTRIBUTING THE PATTERN
4333  015232  104413          4$:     RESREG                  ;RESTORE THE REGISTERS
4334  015234  000207                  RTS     PC              ;RETURN
4335
4336                          ;;*****************************************************************
4337
4338                          .SBTTL  ERROR MESSAGE GENERATION ROUTINES
4339
4340                          ;;*****************************************************************
4341
4342                          ;PRINT LINE 1 OF ERROR MESSAGE:
4343                          ;'HH:MM:SS'
4344
4345  015236  032777  002000  163710  LINE1:  BIT     #SW10,@SWR      ;SWITCH 10 SET ?
4346  015244  001402                  BEQ     1$              ;BR IF NOT
4347  015246  104401  001200          TYPE    ,$BELL          ;RING THE BELL
4348  015252  032777  020000  163674  1$:     BIT     #SW13,@SWR      ;INHIBIT TYPEOUT ?
4349  015260  001403                  BEQ     2$              ;BR IF NOT
4350  015262  104414  001205          DISPLY  ,$CRLF          ;CR-LF
4351  015266  000404                  BR      3$              ;EXIT
```

```
4352  015270  004737  015552          2$:     JSR     PC,$TIME        ;TYPE THE TIME
4353  015274  104414  037015                  DISPLY  ,LINSPO         ;SPACES
4354  015300  000207                  3$:     RTS     PC              ;RETURN & TYPE DESCRIPTION
4355
4356                                  ;OCTAL TYPEOUT ROUTINE
4357                                  ;CALL:
4358                                  ;       MOV     NUM,-(SP)       ;PUT THE NUMBER ON THE STACK
4359                                  ;       JSR     PC,LINOCT
4360                                  ;       RETURN
4361
4362  015302  016646  000002          LINOCT: MOV     2(SP),-(SP)     ;PUT NUMBER IN PROPER LOCATION ON STACK
4363  015306  004737  017046                  JSR     PC,$S8Z0        ;CONVERT THE NUMBER TO OCTAL
4364  015312  012637  015326                  MOV     (SP)+,1$        ;GET THE ADDRESS OF THE ASCII STRING
4365  015316  062737  000005  015326          ADD     #5.,1$          ;ADDRESS THE LAST 6 ASCII DIGITS
4366  015324  104414                          DISPLY                  ;TYPE IT
4367  015326  000000                  1$:     .WORD   0               ;ADDRESS
4368  015330  012616                          MOV     (SP)+,(SP)      ;CORRECT THE STACK
4369  015332  000207                          RTS     PC              ;RETURN
4370
4371                                  ;ROUTINE TO CONVERT THE INPUT NUMBER TO DECIMAL AND TYPE IT WITH
4372                                  ;LEADING ZERO SUPRESSION
4373                                  ;CALL:
4374                                  ;       MOV     NUM,-(SP)       ;PUT THE NUMBER ON THE STACK
4375                                  ;       JSR     PC,LINDEC
4376                                  ;       RETURN
4377
4378  015334  016646  000002          LINDEC: MOV     2(SP),-(SP)     ;SET UP STACK FOR CONVERT
4379  015340  004737  017016                  JSR     PC,$S82D        ;CONVERT IT TO DECIMAL
4380  015344  004737  016422                  JSR     PC,$SUPRS       ;TYPE IT (WITH LEADING ZEROS SUPRESSED)
4381  015350  012616                          MOV     (SP)+,(SP)      ;RESTORE STACK POINTER
4382  015352  000207                          RTS     PC
4383
4384                                  ;;*****************************************************************
4385
4386                                  .SBTTL  GENERAL SUPPORT SUBROUTINES
4387
4388                                  ;;*****************************************************************
4389
4390
4391                                  ;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS
4392
4393  015354  012737  177777  001314  CKCLK:   MOV     #-1,CLKFLG      ;CLEAR CLOCK AVAILABILITY FLAG
4394  015362  012737  177777  001312          MOV     #-1,PCLOCK      ;CLEAR KW11-P CLOCK AVAILABILITY FLAG
4395  015370  012737  015450  000004          MOV     #CKCLK1,ERRVEC  ;SET UP VECTOR FOR CLOCK CHECK
4396  015376  005037  000006                  CLR     @#ERRVEC+2      ;NEW PSW
4397  015402  005777  163672                  TST     @$LKCSR         ;CHECK FOR KW11-P
4398  015406  005037  001314                  CLR     CLKFLG          ;SET CLOCK AVAILABILITY FLAG
4399  015412  005037  001312                  CLR     PCLOCK          ;SET KW11-P CLOCK FLAG
4400  015416  013701  001304                  MOV     $LPVEC,R1       ;KW11-P VECTOR ADDRESS
4401  015422  012721  015650                  MOV     #CLOCK,(R1)+    ;SET UP KW11-P VECTOR
4402  015426  012711  000300                  MOV     #300,(R1)       ;PSW - PRI 6
4403  015432  012777  174575  163642          MOV     #-1667.,@$LKCSB ;LOAD COUNTER BUFFER WITH 16.67
4404  015440  012777  000131  163632          MOV     #131,@$LKCSR    ;SET CLOCK - CNT UP, 10US, CONT INT
4405  015446  000435                          BR      CKCLK3
4406  015450  062706  000004          CKCLK1: ADD     #4,SP           ;RESTORE THE STACK POINTER
4407  015454  012737  015516  000004          MOV     #CKCLK2,@#ERRVEC  ;CHANGE ERROR VECTOR TO CHECK FOR KW11-L
```

```
4408  015462  005777  163620          TST     @$LKS           ;LOOK FOR KW11-L
4409  015466  005037  001314          CLR     CLKFLG          ;SET CLOCK FLAG
4410  015472  013701  001310          MOV     $LLVEC,R1       ;KW11-L VECTOR ADDRESS
4411  015476  012721  015650          MOV     #CLOCK,(R1)+    ;SET UP KW11-L VECTOR
4412  015502  012711  000300          MOV     #300,(R1)       ;PSW - PRI 6
4413  015506  012777  000100  163572  MOV     #100,@$LKS      ;SET KW11-L INTERRUPT
4414  015514  000412                  BR      CKCLK3
4415  015516  062706  000004  CKCLK2: ADD     #4,SP           ;RESTORE THE STACK POINTER
4416  015522  104401  036613          TYPE    ,NEDCLK         ;'P OR L CLOCK MUST BE ON SYSTEM'
4417  015526  005737  000042          TST     42              ;UNDER MONITOR CONTROL ?
4418  015532  001400                  BEQ     1$              ;BR IF NOT
4419  015534  000000  1$:     HALT                            ;HALT
4420  015536  000137  003062          JMP     START           ;TRY AGAIN
4421  015542  012737  000006  000004  CKCLK3: MOV     #6,@#ERRVEC     ;RESTORE THE ERROR VECTOR
4422  015550  000207                  RTS     PC
4423
4424
4425                                  ;ROUTINE TO TYPE THE TIME
4426
4427  015552  005737  001314  $TIME:  TST     CLKFLG          ;CLOCK ON THE SYSTEM ?
4428  015556  001033                  BNE     1$              ;BR IF NOT
4429  015560  104401  001205          TYPE    ,$CRLF          ;CR-LF
4430  015564  013746  001342          MOV     HOUR,-(SP)      ;PUT 'HOURS' ON THE STACK
4431  015570  004737  017016          JSR     PC,$SB2D        ;CONVERT TO DECIMAL
4432  015574  004537  016332          JSR     R5,REPLZ        ;TYPE IT
4433  015600  000002                  .WORD   2               ;TYPE 2 DIGITS
4434  015602  104401  036664          TYPE    ,COLON          ;':'
4435  015606  013746  001344          MOV     MINUTE,-(SP)    ;PUT 'MINUTES' ON THE STACK
4436  015612  004737  017016          JSR     PC,$SB2D        ;CONVERT TO DECIMAL
4437  015616  004537  016332          JSR     R5,REPLZ        ;TYPE IT
4438  015622  000002                  .WORD   2               ;TYPE 2 DIGITS
4439  015624  104401  036664          TYPE    ,COLON          ;':'
4440  015630  013746  001346          MOV     SECOND,-(SP)    ;PUT SECONDS ON THE STACK
4441  015634  004737  017016          JSR     PC,$SB2D        ;CONVERT TO DECIMAL
4442  015640  004537  016332          JSR     R5,REPLZ        ;TYPE IT
4443  015644  000002                  .WORD   2               ;TYPE 2 DIGITS
4444  015646  000207          1$:     RTS     PC
4445
4446                                  ;CLOCK HANDLER ROUTINE
4447
4448  015650  005337  001350  CLOCK:  DEC     SIXTEE          ;INCREMENT THE 1/60 SECOND COUNTER
4449  015654  001033                  BNE     1$              ;BR IF A SECOND NOT COUNTED
4450  015656  013737  001316  001350  MOV     HZ,SIXTEE       ;RESTORE THE VALUE
4451  015664  005237  001346          INC     SECOND          ;COUNT THE SECOND
4452  015670  022737  000074  001346  CMP     #60.,SECOND     ;AT MAXIMUM ?
4453  015676  001022                  BNE     1$              ;BR IF NOT
4454  015700  005037  001346          CLR     SECOND          ;CLEAR THE SECOND'S COUNTER
4455  015704  005237  001344          INC     MINUTE          ;COUNT THE MINUTE
4456  015710  022737  000074  001344  CMP     #60.,MINUTE     ;AT MAXIMUM ?
4457  015716  001012                  BNE     1$              ;BR IF NOT
4458  015720  005037  001344          CLR     MINUTE          ;CLEAR THE MINUTE'S COUNTER
4459  015724  005237  001342          INC     HOUR            ;COUNT THE HOURS
4460  015730  022737  001747  001342  CMP     #999.,HOUR      ;AT MAXIMUM
4461  015736  103002                  BHIS    1$              ;BR IF NOT
4462  015740  005037  001342          CLR     HOUR            ;CLEAR THE HOURS
4463  015744  012746  000021  1$:     MOV     #17.,-(SP)      ;17 MS ON THE STACK
```

```
4464  015750  004737  027740            JSR     PC,RPTMR        ;DRIVER TIMER ROUTINE
4465  015754  000002            2$:     RTI
4466
4467                                    ;COMMAND DECODE ROUTINE
4468                                    ;CALL:
4469                                    ;       MOV     #-1,CFLAG       ;'CFLAG' IS NORMALLY SET BY THE TTY SERVICE
4470                                    ;                               ;ROUTINE IN INTERRUPT MODE
4471                                    ;       JSR     PC,KSR
4472                                    ;       RETURN1                 ;SYSTEM BUSY RETURN
4473                                    ;       RETURN2                 ;RETURN AFTER KEYBOARD SERVICED
4474
4475  015756  104412            KSR:    SAVREG                  ;SAVE THE REGISTERS
4476  015760  012737  000200  177776      MOV     #PR4,PS         ;SET PRIORITY TO 4
4477  015766  005037  001334              CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
4478  015772  004737  015552              JSR     PC,$TIME        ;TYPE THE TIME
4479  015776  005777  163160              TST     @$TKB           ;CLEAR ANY GARBAGE IN THE TTY BUFFER
4480  016002  005737  001334              TST     CFLAG           ;CHECK THE CONTROL C FLAG
4481  016006  001002                      BNE     7$              ;EXIT IF 'CONTROL C' ENTERED
4482  016010  000240                      NOP                     ;DUMP CODING FOR LATER USE 3/8/77
4483  016012  000240                      NOP
4484  016014  104413            7$:     RESREG                  ;RESTORE R0 - R5
4485  016016  062716  000002              ADD     #2,(SP)         ;INCREMENT THE RETURN ADDRESS
4486  016022  005777  163134              TST     @$TKB           ;CLEAR THE TTY BUFFER
4487  016026  052777  000100  163124      BIS     #BIT06,@$TKS    ;SET TTY INTERRUPT ENABLE
4488  016034  005037  177776              CLR     PS              ;SET PRIORITY BACK TO ZERO
4489  016040  000207                      RTS     PC              ;RETURN
4490
4491                                    ;ROUTINE TO CLEAR THE DPB FOR THE ASSIGNED DRIVE
4492                                    ;CALL:
4493                                    ;       MOV     #DPB,R0         ;DPB ADDRESS
4494                                    ;       JSR     PC,CLRDPB
4495                                    ;       RETURN
4496
4497  016042            CLRDPB:
4498  016042  010146                      MOV     R1,-(SP)        ;;PUSH R1 ON STACK
4499  016044  010346                      MOV     R3,-(SP)        ;;PUSH R3 ON STACK
4500  016046  010446                      MOV     R4,-(SP)        ;;PUSH R4 ON STACK
4501  016050  010546                      MOV     R5,-(SP)        ;;PUSH R5 ON STACK
4502  016052  010004                      MOV     R0,R4           ;GET THE DPB ADDRESS
4503  016054  062704  000002              ADD     #2,R4           ;ADDRESS OF FIRST LOCN TO BE CLEARED
4504  016060  012703  000020              MOV     #$EMTAB-2,R3    ;NUMBER OF LOCATIONS TO CLEAR
4505  016064  005024            1$:     CLR     (R4)+           ;CLEAR THE STORAGE LOCATION
4506  016066  162703  000002              SUB     #2,R3           ;DECREMENT THE BYTE COUNT
4507  016072  001374                      BNE     1$              ;LOOPING BACK
4508  016074  012605                      MOV     (SP)+,R5        ;;POP STACK INTO R5
4509  016076  012604                      MOV     (SP)+,R4        ;;POP STACK INTO R4
4510  016100  012603                      MOV     (SP)+,R3        ;;POP STACK INTO R3
4511  016102  012601                      MOV     (SP)+,R1        ;;POP STACK INTO R1
4512  016104  000207                      RTS     PC              ;RETURN
4513
4514  016106  104412            LINKDV: SAVREG                  ;STORE R0 - R5
4515  016110  016605  000026              MOV     26(SP),R5       ;DIVISOR
4516  016114  005004                      CLR     R4              ;OTHER DIVISOR WORD
4517  016116  016602  000030              MOV     30(SP),R2       ;UPPER DIVIDEND WORD
4518  016122  016603  000032              MOV     32(SP),R3       ;LOWER DIVIDEND WORD
4519  016126  005000                      CLR     R0              ;CLEAR OTHER DIVIDEND REGISTERS
```

```
4520  016130  005001                          CLR     R1
4521  016132  004737  016154                  JSR     PC,M.DPID       ;GO TO THE DIVIDE ROUTINE
4522  016136  010166  000030                  MOV     R1,30(SP)       ;REMAINDER ON THE STACK
4523  016142  010366  000032                  MOV     R3,32(SP)       ;QUOTIENT ON THE STACK
4524  016146  104413                          RESREG                  ;RESTORE R0 - R5
4525  016150  012616                          MOV     (SP)+,(SP)      ;MOVE RETURN UP THE STACK
4526  016152  000207                          RTS     PC
4527
4528                                  ;       DIVISION UTILITY SUBROUTINE
4529                                  ;       R0-R1-R2-R3=DIVIDEND
4530                                  ;       R4-R5=DIVISOR
4531                                  ;       R0-R1=REMAINDER AFTER DIVISION
4532                                  ;       R2-R3=QUOTIENT AFTER DIVISION
4533                                  ;       ENTER WITH JSR  PC,M.DPID
4534                                  ;
4535  016154  012746  000040          M.DPID: MOV     #40,-(SP)       ;COUNTER FOR DIVISION CYCLES
4536  016160  010446                          MOV     R4,-(SP)        ;HIGH ORDER
4537  016162  010546                          MOV     R5,-(SP)        ;LOW ORDER DIVISOR TO THE STACK
4538  016164  005466  000002                  NEG     2(SP)           ;FORM NEGATIVE
4539  016170  005416                          NEG     @SP             ;VERSION OF THE DIVISOR
4540  016172  005666  000002                  SBC     2(SP)
4541  016176  061601                          ADD     @SP,R1
4542  016200  005500                          ADC     R0              ;PERFORM THE INITIAL SUBTRACTION
4543  016202  066600  000002                  ADD     2(SP),R0
4544  016206  103445                          BCS     M.DP50          ;IF CARRY THEN OVERFLOW HAS OCCURRED
4545  016210  005046                          CLR     -(SP)           ;THIS IS A LONGER LASTING CARRY BIT
4546  016212  006103                  M.DP40: ROL     R3
4547  016214  006102                          ROL     R2
4548  016216  006101                          ROL     R1
4549  016220  006100                          ROL     R0
4550  016222  005716                          TST     @SP             ;TEST "CARRY" INDICATOR
4551  016224  001410                          BEQ     M.DP41          ;IF NO "CARRY" THEN ADD ELSE SUBTRACT
4552  016226  005016                          CLR     @SP             ;CLEAR UP FOR NEXT TIME
4553  016230  066601  000002                  ADD     2(SP),R1
4554  016234  005500                          ADC     R0              ;ADD -(DIVISOR)
4555  016236  005516                          ADC     @SP     ;  I    ;SET "CARRY"
4556  016240  066600  000004                  ADD     4(SP),R0;<-
4557  016244  000404                          BR      M.DP42
4558  016246  060501                  M.DP41: ADD     R5,R1
4559  016250  005500                          ADC     R0              ;ADD +(DIVISOR)
4560  016252  005516                          ADC     @SP     ;  I    ;SET "CARRY"
4561  016254  060400                          ADD     R4,R0   ;<-
4562  016256  005516                  M.DP42: ADC     @SP             ;SET "CARRY"
4563  016260  005716                          TST     @SP             ;TEST THE UPDATE INDICATOR
4564  016262  001401                          BEQ     .+4     ;->     ;IF ZERO FORGET IT
4565  016264  005203                          INC     R3      ;  I    ;NO CARRY POSSIBLE HERE
4566  016266  005366  000006                  DEC     6(SP)   ;<-     ;DECREMENT COUNTER
4567  016272  003347                          BGT     M.DP40          ;BRANCH IF MORE TO DO
4568  016274  006003                          ROR     R3
4569  016276  103404                          BCS     M.DP44
4570  016300  060501                          ADD     R5,R1
4571  016302  005500                          ADC     R0
4572  016304  060400                          ADD     R4,R0
4573  016306  000241                          CLC
4574  016310  006103                  M.DP44: ROL     R3
4575  016312  062706  000010                  ADD     #10,SP          ;ADJUST STACK BY 4 WORDS
```

```
4576  016316  000242                        CLV
4577  016320  000207                        RTS     PC
4578  016322  062706  000006        M.DPS0: ADD     #6,SP
4579  016326  000262                        SEV
4580  016330  000207                        RTS     PC
4581
4582
4583                                ;ROUTINE TO REPLACE LEADING ZEROS IN A NUMERIC STRING WITH SPACES
4584                                ;CALL
4585                                ;       MOV     #ADR,-(SP)      ;ADDRESS OF NUMBER (IN ASCII)
4586                                ;       JSR     R5,REPLZ
4587                                ;       .WORD   N               ;'N' IS NUMBER OF DIGITS TO BE TYPED
4588
4589  016332  010046                REPLZ:  MOV     R0,-(SP)        ;SAVE R0
4590  016334  012746  000012                MOV     #10.,-(SP)      ;MAXIMUM NUMBER OF DIGITS TO BE TYPED
4591  016340  162516                        SUB     (R5)+,(SP)      ;SUBTACT DIGITS TO FORM INDEX
4592  016342  016600  000006                MOV     6(SP),R0        ;ADDRESS OF NUMBER TO R0
4593  016346  122710  000060        1$:     CMPB    #'0,(R0)        ;BYTE EQUAL TO ASCII '0' ?
4594  016352  001004                        BNE     2$              ;BR IF NOT
4595  016354  112710  000040                MOVB    #40,(R0)        ;REPLACE THE ZERO WITH A SPACE
4596  016360  005200                        INC     R0              ;INCREMENT THE BYTE ADDRESS
4597  016362  000771                        BR      1$              ;GO BACK AND LOOK FOR MORE LEADING ZEROS
4598  016364  105710                2$:     TSTB    (R0)            ;SEE IF ZERO BYTE TERMINATOR
4599  016366  001003                        BNE     3$              ;BR IF NOT
4600  016370  005300                        DEC     R0              ;BACKUP STRING POINTER
4601  016372  112710  000060                MOVB    #'0,(R0)        ;PUT A ZERO BACK IN
4602  016376  016637  000006  016412  3$:   MOV     6(SP),4$        ;PUT ADDRESS IN LOCATION FOR TYPEOUT
4603  016404  062637  016412                ADD     (SP)+,4$        ;BEGINNING OF SIGNIFICANT DIGITS
4604  016410  104401                        TYPE                    ;TYPE THE NUMBER
4605  016412  000000                4$:     .WORD   0               ;ADDRESS OF NUMBER
4606  016414  012600                        MOV     (SP)+,R0        ;RESTORE R0
4607  016416  012616                        MOV     (SP)+,(SP)      ;MOVE RETURN ADDRESS
4608  016420  000205                        RTS     R5              ;RETURN
4609
4610                                ;TYPE NUMERICAL ASCIZ STRING SUPRESS LEADING ZEROS
4611
4612                                ;CALL
4613                                ;       MOV     #NUMADR,-(SP)   ;FIRST ADDRESS OF ASCIZ STRING
4614                                ;       JSR     PC,$SUPRS
4615
4616  016422  010046                $SUPRS: MOV     R0,-(SP)        ;SAVE R0
4617  016424  016600  000004                MOV     4(SP),R0        ;PICKUP THE POINTER
4618  016430  105710                1$:     TSTB    (R0)            ;TERMINATOR ?
4619  016432  001403                        BEQ     2$              ;BR IF YES
4620  016434  122720  000060                CMPB    #'0,(R0)+       ;IS THIS AN ASCII '0' ?
4621  016440  001773                        BEQ     1$              ;BR IF YES
4622  016442  005300                2$:     DEC     R0              ;BACKUP BY '1'
4623  016444  010037  016452                MOV     R0,3$           ;SAVE FOR TYPING
4624  016450  104414                        DISPLY                  ;GO PRINT
4625  016452  000000                3$:     .WORD   0               ;ASCIZ POINTER GOES HERE
4626  016454  012600                        MOV     (SP)+,R0        ;RESTORE R0
4627  016456  012616                        MOV     (SP)+,(SP)      ;RESTORE THE STACK
4628  016460  000207                        RTS     PC              ;RETURN
4629
4630                                ;ROUTINE TO TYPE AT PRIORITY 4
4631
```

```
4632  016462  013746  177776          TYPRI4: MOV     @#PS,-(SP)      ;SAVE THE PRESENT STATUS
4633  016466  012737  000200  177776          MOV     #200,@#PS       ;CHANGE THE PRIORITY TO 4
4634  016474  012537  016504                  MOV     (R5)+,1$        ;MESSAGE ADDRESS
4635  016500  004737  020352                  JSR     PC,$TYPE        ;TYPE THE MESSAGE
4636  016504  000000                  1$:     .WORD   0               ;MESSAGE ADDRESS GOES HERE
4637  016506  000205                          RTS     R5              ;RETURN
4638
4639                                          ;ROUTINE TO TYPE ERRORS
4640                                          ;CALL
4641                                          ;       DISPLY                  ;MUST DEFINED IN 'TRAP' TABLE
4642                                          ;       MESADR                  ;ADDRESS OF MESSAGE
4643                                          ;       RETURN
4644
4645  016510  032777  020000  162436  $DSPLY: BIT     #BIT13,@SWR     ;INHIBIT ERROR TYPEOUT ?
4646  016516  001002                          BNE     1$              ;BR IF YES
4647  016520  000137  020352                  JMP     $TYPE           ;TYPE THE MESSAGE
4648  016524  062716  000002          1$:     ADD     #2,(SP)         ;INCREMENT THE RETURN
4649  016530  000002                          RTI                     ;RETURN
4650
4651                                          ;THIS ROUTINE IS USED TO CHECK IF AN
4652                                          ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
4653                                          ;CALL
4654                                          ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
4655                                          ;       JSR     R5,CK.OCT       ;CHECK THE CHARACTER
4656                                          ;       RETURN1                 ;CHARACTER IS NOT BETWEEN 0-7
4657                                          ;       RETURN2                 ;CHARACTER IS IN R2 AS A
4658                                          ;                               ;OCTAL DIGIT
4659
4660  016532  121127  000060          CK.OCT: CMPB    (R1),#'0        ;LESS THAN ZERO?
4661  016536  103407                          BLO     1$              ;YES -- BRANCH
4662  016540  121127  000067                  CMPB    (R1),#'7        ;GREATER THAN SEVEN?
4663  016544  101004                          BHI     1$              ;YES -- BRANCH
4664  016546  111102                          MOVB    (R1),R2         ;GET THE CHARACTER
4665  016550  042702  177770                  BIC     #^C7,R2         ;STRIP AWAY THE ASCII
4666  016554  005725                          TST     (R5)+           ;ADJUST FOR RETURN
4667  016556  000205                  1$:     RTS     R5              ;RETURN
4668
4669                                          ;THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
4670                                          ;AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
4671                                          ;CALL
4672                                          ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
4673                                          ;       JSR     R5,CK.DEC       ;CHECK THE CHARACTER
4674                                          ;       RETURN1                 ;NOT BETWEEN 0 AND 9
4675                                          ;       RETURN2                 ;BETWEEN 0 AND 9
4676                                          ;                               ;R2 = DIGIT
4677
4678  016560  121127  000060          CK.DEC: CMPB    (R1),#'0        ;LESS THAN ZERO?
4679  016564  103407                          BLO     1$              ;YES -- BRANCH
4680  016566  121127  000071                  CMPB    (R1),#'9        ;GREATER THAN NINE?
4681  016572  101004                          BHI     1$              ;YES -- BRANCH
4682  016574  111102                          MOVB    (R1),R2         ;GET THE CHARACTER
4683  016576  042702  000060                  BIC     #'0,R2          ;STRIP AWAY THE ASCII
4684  016602  005725                          TST     (R5)+           ;ADJUST FOR RETURN
4685  016604  000205                  1$:     RTS     R5              ;RETURN
4686
4687                                          ;THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
```

```
                                        ;DETERMINE WHAT IT IS.
                                        ;CALL
                                        ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
                                        ;       JSR     R5,CK.CHR       ;CHECK CHARACTER
                                        ;       RETURN  ADR1            ;UNKNOWN CHARACTER
                                        ;       RETURN  ADR2            ;CARRIAGE RETURN * (R1)=ADR+1
                                        ;       RETURN  ADR3            ;COMMA * (R1)=ADR+1
                                        ;       RETURN  ADR4            ;PERIOD * (R1)=ADR+1
                                        ;       RETURN  ADR5            ;DIGIT BETWEEN 0 AND 7.
                                        ;       RETURN  ADR6            ;DIGIT BETWEEN 8 AND 9.
                                        ;                               ;R2 = DIGIT * (R1)=ADR+1

016606  105711          CK.CHR: TSTB    (R1)            ;"CARRIAGE RETURN"?
016610  001417                  BEQ     3$              ;YES -- BRANCH
016612  121127  000054          CMPB    (R1),#',        ;"COMMA"?
016616  001413                  BEQ     2$              ;YES -- BRANCH
016620  121127  000056          CMPB    (R1),#'.        ;"PERIOD"?
016624  001407                  BEQ     1$              ;YES -- BRANCH
016626  004537  016560          JSR     R5,CK.DEC       ;"DIGIT"?
016632  000410                  BR      4$              ;NO -- BRANCH
016634  004537  016532          JSR     R5,CK.OCT       ;OCTAL ?
016640  005725                  TST     (R5)+           ;DIGIT BETWEEN 8-9
016642  005725                  TST     (R5)+           ;DIGIT BETWEEN 0-7
016644  005725          1$:     TST     (R5)+           ;PERIOD
016646  005725          2$:     TST     (R5)+           ;COMMA
016650  005725          3$:     TST     (R5)+           ;CARRIAGE RETURN
016652  005201                  INC     R1              ;MOVE POINTER TO NEXT CHARACTER
016654  011505          4$:     MOV     (R5),R5         ;UNKNOWN CHARACTER
016656  000205                  RTS     R5              ;RETURN

                                        ;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
                                        ;CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
                                        ;CALL
                                        ;       MOV     #ADR,R1         ;ADDRESS OF ASCIZ STRING
                                        ;       MOV     #NUM,R2         ;MAX. MAGNITUDE OF INPUT NUMBER
                                        ;       JSR     R5,CK.DIG       ;CHECK DIGITS
                                        ;       RETURN  ADR1            ;"CR" ONLY ENTERED -- R2=0
                                        ;       RETURN  ADR2            ;"PERIOD" ONLY ENTERED -- R2=0
                                        ;       RETURN  ADR3            ;ILLEGAL CHARACTER OR INPUT TOO LARGE -- R2=^
                                        ;       RETURN  ADR4            ;"CR" -- R2 = NUMBER
                                        ;       RETURN  ADR5            ;"COMMA" -- R2 = NUMBER
                                        ;       RETURN  ADR6            ;"PERIOD" -- R2 = NUMBER

016660  010446          CK.DIG: MOV     R4,-(SP)        ;SAVE R4
016662  010346                  MOV     R3,-(SP)        ;SAVE R3
016664  010246                  MOV     R2,-(SP)        ;SAVE THE MAX. SIZE ON THE STACK
016666  005002                  CLR     R2              ;START WITH 0
016670  005003                  CLR     R3
016672  005004                  CLR     R4
016674  004537  016606          JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
016700  016774                  6$                      ;ILLEGAL CHARACTER
016702  017002                  9$                      ;CARRIAGE RETURN
016704  016774                  6$                      ;","
016706  016776                  7$                      ;"."
016710  016714                  1$                      ;DIGIT 0-7
016712  016714                  1$                      ;DIGIT 8-9
```

```
4744  016714  062705  000004          1$:     ADD     #4,R5           ;STEP RETURN POINTER PAST "CR" & "PERIOD" RETURNS
4745  016720  006303                  2$:     ASL     R3              ;INPUT NUMBER *2
4746  016722  010346                          MOV     R3,-(SP)        ;SAVE *2
4747  016724  006303                          ASL     R3              ;*4
4748  016726  006303                          ASL     R3              ;*8
4749  016730  062603                          ADD     (SP)+,R3        ;(*2)+(*8) = *10
4750  016732  060203                          ADD     R2,R3           ;UPDATE THE INPUT NUMBER
4751  016734  004537  016606                  JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
4752  016740  017000                          8$                      ;ILLEGAL CHARACTER
4753  016742  016764                          5$                      ;CARRIAGE RETURN
4754  016744  016762                          4$                      ;"."
4755  016746  016754                          3$                      ;"'"
4756  016750  016720                          2$                      ;DIGIT 0-7
4757  016752  016720                          2$                      ;DIGIT 8-9
4758  016754  105711                  3$:     TSTB    (R1)            ;DOES A "CR" FOLLOW THE "PERIOD"
4759  016756  001010                          BNE     8$              ;BR IF NOT
4760  016760  005724                          TST     (R4)+           ;INCREMENT THE RETURN
4761  016762  005724                  4$:     TST     (R4)+           ;INCREMENT THE RETURN
4762  016764  005724                  5$:     TST     (R4)+           ;INCREMENT THE RETURN
4763  016766  020316                          CMP     R3,(SP)         ;CHECK THE MAGNITUDE OF THE NUMBER
4764  016770  101004                          BHI     9$              ;BR IF ENTERED NUMBER TOO LARGE
4765  016772  000402                          BR      8$              ;BYPASS INCREMENT
4766  016774  005725                  6$:     TST     (R5)+           ;INCREMENT RETURN PAST INVALID RETURN
4767  016776  005725                  7$:     TST     (R5)+           ;INCREMENT RETURN
4768  017000  060405                  8$:     ADD     R4,R5           ;SETUP RETURN POINTER
4769  017002  010302                  9$:     MOV     R3,R2           ;ENTERED VALUE
4770  017004  005726                          TST     (SP)+           ;CLEAN MAX. SIZE OFF OF STACK
4771  017006  012603                          MOV     (SP)+ R3        ;RESTORE R3
4772  017010  012604                          MOV     (SP)+,R4        ;RESTORE R4
4773  017012  011505                          MOV     (R5),R5         ;GET RETURN ADDRESS
4774  017014  000205                          RTS     R5              ;RETURN
4775
4776                                  ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
4777                                  ;UNSIGNED DECIMAL ASCIZ NUMBER.
4778                                  ;CALL
4779                                  ;       MOV     NUMBER,-(SP)    ;PUT THE NUMBER ON THE STACK
4780                                  ;       JSR     PC,$SB2D        ;CALL
4781                                  ;       RETURN                  ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
4782                                  ;
4783                                  ;NOTE:  THE PROGRAM REQUIRES THIS FORM OF '$SB2D', NOT THE VERSION ON
4784                                  ;       THE SYSMAC LIBRARY, REV C AND LATER
4785
4786  017016  016637  000002  017042  $SB2D:  MOV     2(SP),1$        ;SAVE THE BINARY NUMBER
4787  017024  012746  017042                  MOV     #1$,-(SP)       ;SET THE POINTER
4788  017030  004737  022476                  JSR     PC,$DB2D        ;CALL THE DOUBLE LENGTH CONVERT
4789  017034  012666  000002                  MOV     (SP)+,2(SP)     ;PICKUP THE POINTER
4790  017040  000207                          RTS     PC              ;RETURN
4791  017042  000000  000000          1$:     .WORD   0,0
4792
4793                                  ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
4794                                  ;UNSIGNED OCTAL ASCIZ NUMBER.
4795                                  ;CALL
4796                                  ;       MOV     NUMBER,-(SP)    ;PUT THE NUMBER ON THE STACK
4797                                  ;       JSR     PC,$SB2O        ;CALL
4798                                  ;       RETURN                  ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
4799                                  ;
```

```
                                        ;NOTE:   THE PROGRAM REQUIRES THIS FORM OF '$SB20', NOT THE VERSION ON
                                        ;        THE SYSMAC LIBRARY, REV C AND LATER
                                        ;
017046  016637  000002  017072  $SB20:   MOV     2(SP),1$          ;SAVE THE BINARY NUMBER
017054  012746  017072                   MOV     #1$,-(SP)         ;SET THE POINTER
017060  004737  022672                   JSR     PC,$DB20          ;CALL THE DOUBLE LENGTH CONVERT
017064  012666  000002                   MOV     (SP)+,2(SP)       ;PICKUP THE POINTER
017070  000207                           RTS     PC                ;RETURN
017072  000000  000000          1$:      .WORD   0,0
                                        ;KEYBOARD INTERRUPT INITIALIZATION ROUTINE
                                        ;CALL
                                        ;        JSR     PC,$TKINT
                                        ;        RETURN
017076  012737  017126  000060  $TKINT:  MOV     #$TKSRV,TKVEC     ;SETUP VECTOR
017104  012737  000240  000062           MOV     #PR5,TKVEC+2      ;PRIORITY TO 5
017112  005777  162044                   TST     @$TKB             ;CLEAR THE BUFFER
017116  012777  000100  162034           MOV     #BIT06,@$TKS      ;SET INTERRUPT ENABLE
017124  000207                           RTS     PC                ;RETURN
                                        ;KEYBOARD INTERRUPT SERVICE ROUTINE
                                        ;CALL
                                        ;        ENTER VIA INTERRUPT
017126  104410                  $TKSRV:  RDCHR                     ;READ THE KEYBOARD
017130  112637  017272                   MOVB    (SP)+,5$          ;GET THE CHARACTER
017134  023727  017272  000003           CMP     5$,#3             ;'CONTROL C' ?
017142  001020                           BNE     1$                ;BR IF NOT
017144  104401  001205                   TYPE    ,$CRLF            ;CR-LF
017150  104401  017576                   TYPE    ,$CNTLC           ;'↑C'
017154  012737  177777  001334           MOV     #-1,CFLAG         ;SET THE 'CONTROL C' FLAG
017162  005077  161772                   CLR     @$TKS             ;CLEAR THE TTY INTERRUPT
017166  104401  001205                   TYPE    ,$CRLF
017172  104401  037152                   TYPE    ,HALTX            ;HALT THE PROGRAM
017176  000000                           HALT
017200  000137  000200                   JMP     @#200             ;
017204  023727  001154  000176  1$:      CMP     SWR,#SWREG        ;SOFTWARE SWITCH REGISTER IN USE ?
017212  001024                           BNE     3$                ;BR IF NOT
017214  023727  017272  000007           CMP     5$,#7             ;'CONTROL G' ?
017222  001020                           BNE     3$                ;BR IF NOT
017224  104401  001205                   TYPE    ,$CRLF            ;CR-LF
017230  104401  022353                   TYPE    ,$CNTLG           ;'↑G'
017234  013746  177776                   MOV     PS,-(SP)          ;PUT THE STATUS WORD ON THE STACK
017240  012746  017254                   MOV     #2$,-(SP)         ;RETURN ADDRESS
017244  005077  161710                   CLR     @$TKS             ;CLEAR THE TTY INTERRUPT ENABLE
017250  000137  022014                   JMP     $GTSWR            ;GET THE SWITCH REGISTER ENTRY
017254  012777  000100  161676  2$:      MOV     #100,@$TKS        ;ENABLE TTY KEYBOARD INTERRUPT
017262  000402                           BR      4$                ;EXIT
017264  104401  017272          3$:      TYPE    ,5$               ;ECHO THE CHARACTER
017270  000002                  4$:      RTI                       ;RETURN

017272  000000                  5$:      .WORD   0                 ;ENTERED CHARACTER

                                        ;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
                                        ;
017274  010346                  $RDLIN: MOV     R3,-(SP)                ;SAVE R3
017276  005046                          CLR     -(SP)                   ;CLEAR THE RUBOUT KEY
017300  012703  017552          1$:     MOV     #$TTYIN,R3              ;GET ADDRESS
017304  022703  017564          2$:     CMP     #$TTYIN+10.,R3          ;BUFFER FULL?
017310  101467                          BLOS    4$                      ;BR IF YES
017312  104410                          RDCHR                           ;GO READ ONE CHARACTER FROM THE TTY
017314  112613                          MOVB    (SP)+,(R3)              ;GET CHARACTER
017316  122713  000177                  CMPB    #177,(R3)               ;IS IT A RUBOUT
017322  001022                          BNE     5$                      ;BR IF NO
017324  005716                          TST     (SP)                    ;IS THIS THE FIRST RUBOUT?
017326  001007                          BNE     6$                      ;BR IF NO
017330  112737  000134  017550          MOVB    #'\,9$                  ;TYPE A BACK SLASH
017336  104401  017550                  TYPE    ,9$
017342  012716  177777                  MOV     #-1,(SP)                ;SET THE RUBOUT KEY
017346  005303                  6$:     DEC     R3                      ;BACKUP BY ONE
017350  020327  017552                  CMP     R3,#$TTYIN              ;STACK EMPTY?
017354  103445                          BLO     4$                      ;BR IF YES
017356  111337  017550                  MOVB    (R3),9$                 ;SETUP TO TYPEOUT THE DELETED CHAR.
017362  104401  017550                  TYPE    ,9$                     ;GO TYPE
017366  000746                          BR      2$                      ;GO READ ANOTHER CHAR.
017370  005716                  5$:     TST     (SP)                    ;RUBOUT KEY SET?
017372  001406                          BEQ     7$                      ;BR IF NO
017374  112737  000134  017550          MOVB    #'\,9$                  ;TYPE A BACK SLASH
017402  104401  017550                  TYPE    ,9$
017406  005016                          CLR     (SP)                    ;CLEAR THE RUBOUT KEY
017410  122713  000025          7$:     CMPB    #25,(R3)                ;IS CHARACTER A CTRL U?
017414  001003                          BNE     10$                     ;BR IF NO
017416  104401  022346                  TYPE    ,$CNTLU                 ;TYPE A CONTROL "U"
017422  000726                          BR      1$                      ;GO START OVER
017424  122713  000003          10$:    CMPB    #3,(R3)                 ;IS CHARACTER A CTRL C ?
017430  001006                          BNE     8$                      ;BR IF NOT
017432  012737  177777  001334          MOV     #-1,CFLAG               ;SET CNTRL C FLAG
017440  104401  017576                  TYPE    ,$CNTLC                 ;ECHO IT
017444  000427                          BR      11$                     ;EXIT
017446  122713  000012          8$:     CMPB    #12,(R3)                ;IS CHARACTER A "LF"?
017452  001011                          BNE     3$                      ;BRANCH IF NO
017454  105013                          CLRB    (R3)                    ;CLEAR THE CHARACTER
017456  104401  001205                  TYPE    ,$CRLF                  ;TYPE A "CR" & "LF"
017462  104401  017552                  TYPE    ,$TTYIN                 ;TYPE THE INPUT STRING
017466  000706                          BR      2$                      ;GO PICKUP ANOTHER CHACTER
017470  104401  001204          4$:     TYPE    ,$QUES                  ;TYPE A '?'
017474  000701                          BR      1$                      ;CLEAR THE BUFFER AND LOOP
017476  111337  017550          3$:     MOVB    (R3),9$                 ;ECHO THE CHARACTER
017502  104401  017550                  TYPE    ,9$
017506  122723  000015                  CMPB    #15,(R3)+               ;CHECK FOR RETURN
017512  001274                          BNE     2$                      ;LOOP IF NOT RETURN
017514  105063  177777                  CLRB    -1(R3)                  ;CLEAR RETURN (THE 15)
017520  104401  001206                  TYPE    ,$LF                    ;TYPE A LINE FEED
017524  005726                  11$:    TST     (SP)+                   ;CLEAN RUBOUT KEY FROM THE STACK
017526  012603                          MOV     (SP)+,R3                ;RESTORE R3
017530  011646                          MOV     (SP),-(SP)              ;ADJUST THE STACK AND PUT ADDRESS OF THE
017532  016666  000004  000002          MOV     4(SP),2(SP)             ;       FIRST ASCII CHARACTER ON IT
```

```
017540  012766  017552  000004          MOV     #$TTYIN,4(SP)
017546  000002                          RTI                     ;RETURN
017550     000          9$:     .BYTE   0                       ;STORAGE FOR ASCII CHAR. TO TYPE
017551     000                  .BYTE   0                       ;TERMINATOR
017552  000024          $TTYIN: .BLKB   20.                     ;RESERVE 20 BYTES FOR TTY INPUT
017576  041536  005015     000  $CNTLC: .ASCIZ   /^C/<CR><LF>    ;CONTROL "C"

        017604                  .EVEN


017604  005737  001374          FINISH: TST     FAULT           ;COMPATIBLE ?
017610  001006                  BNE     1$                      ;BRANCH IF NOT
017612  104401  001205                  TYPE    ,$CRLF
017616  104401  001205                  TYPE    ,$CRLF
017622  104401  036120                  TYPE    ,MESG15         ;MESSAGE: ALL DRIVE COMPATIBLE
017626  104401  001205          1$:     TYPE    ,$CRLF
017632  104401  001205                  TYPE    ,$CRLF
017636  104401  037060                  TYPE    ,MESG20
017642  104401  001205                  TYPE    ,$CRLF
017646  000000                          HALT
017650  000137  000200                  JMP     @#200           ;BRANCH TO 200 START IF DESIRED


                                .SBTTL  END OF PASS ROUTINE

                                ;;******************************************************************
                                ;*INCREMENT THE PASS NUMBER ($PASS)
                                ;*IF SW12=1 INHIBIT TRACE TRAP
                                ;*IF THERES A MONITOR GO TO IT
                                ;*IF THERE ISN'T JUMP TO FINISH


017654                          $EOP:
017654  000240                          NOP
017656  005037  001116                  CLR     $TSTNM          ;;ZERO THE TEST NUMBER
017662  005037  001174                  CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
017666  005237  001216                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
017672  042737  100000  001216          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
017700  005327                          DEC     (PC)+           ;;LOOP?
017702  000001          $EOPCT: .WORD   1
017704  003022                  BGT     $DOAGN                  ;;YES
017706  012737                  MOV     (PC)+,@(PC)+            ;;RESTORE COUNTER
017710  000001          $ENDCT: .WORD   1
017712  017702                  $EOPCT
017714  013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
017720  001414                  BEQ     $DOAGN                  ;;BRANCH IF NO MONITOR
017722  005046                  CLR     -(SP)                   ;;INSURE THE "T" BIT IS CLEAR
017724  012746  017732          MOV     #$CLR.T,-(SP)           ;;SETUP FOR AN RTI OR RTT
017730  000426                  BR      $RTRN                   ;;GO DO AN RTI OR RTT TO LOAD THE PSW
                                                                ;;WITH A CLEARED "T" BIT

017732                          $CLR.T:
017732  013700  000042          MOV     @#42,R0                 ;;INSURE R0 CONTAINS THE MONITORS
017736  001405                  BEQ     $DOAGN                  ;;RETURN ADDRESS
017740  000005                  RESET                           ;;CLEAR THE WORLD
017742  004710          $ENDAD: JSR     PC,(R0)                 ;;GO TO MONITOR
017744  000240                  NOP                             ;;SAVE ROOM
017746  000240                  NOP                             ;;FOR
```

```
 4968  017750  000240                          NOP                     ;;ACT11
 4969  017752                              $DOAGN:
 4970  017752  104400                          TRAP                    ;;PUSH OLD PSW AND PC ON STACK
 4971  017754  042716  000020                  BIC     #20,(SP)        ;;CLEAR THE "T" BIT
 4972  017760  032777  010000  161166          BIT     #BIT12,@SWR     ;;RUN WITH TRACE TRAP?
 4973  017766  001005                          BNE     1$              ;;BR IF NO
 4974  017770  005137  020014                  COM     $TBIT           ;;IS IT TIME FOR TRACE TRAP
 4975  017774  100402                          BMI     1$              ;;BR IF NO
 4976  017776  052716  000020                  BIS     #20,(SP)        ;;SET TRACE TRAP
 4977  020002  012746  020010          1$:     MOV     #$LOOP,-(SP)    ;;JUMP TO START OF TEST
 4978  020006  000002                  $RTRN:  RTI                     ;;RETURN--THIS IS CHANGED TO
 4979                                                                  ;;AN "RTT" IF "RTT" IS A LEGAL
 4980                                                                  ;;INSTRUCTION
 4981  020010                          $LOOP:
 4982  020010  000137                          JMP     @(PC)+          ;;RETURN
 4983  020012  017604                  $RTNAD: .WORD   FINISH
 4984  020014  000000                  $TBIT:  .WORD   0               ;;"T" BIT STATE INDICATOR
 4985
 4986                                  .SBTTL  ERROR HANDLER ROUTINE
 4987
 4988                                  ;;**************************************************************
 4989                                  ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
 4990                                  ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
 4991                                  ;*AND GO TO $ERRTYP ON ERROR
 4992                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
 4993                                  ;*SW15=1         HALT ON ERROR
 4994                                  ;*SW13=1         INHIBIT ERROR TYPEOUTS
 4995                                  ;*SW10=1         BELL ON ERROR
 4996                                  ;*SW09=1         LOOP ON ERROR
 4997                                  ;*CALL
 4998                                  ;*       ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
 4999
 5000  020016                          $ERROR:
 5001  020016  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
 5002  020020  105237  001117          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
 5003  020024  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
 5004  020026  013777  001116  161122          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
 5005  020034  032777  002000  161112          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
 5006  020042  001402                          BEQ     1$              ;;NO - SKIP
 5007  020044  104401  001200                  TYPE    ,$BELL          ;;RING BELL
 5008  020050  005237  001126          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
 5009  020054  011637  001132                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
 5010  020060  162737  000002  001132          SUB     #2,$ERRPC
 5011  020066  117737  161040  001130          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
 5012  020074  032777  020000  161052          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
 5013  020102  001004                          BNE     20$             ;;SKIP TYPEOUTS
 5014  020104  004737  020216                  JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
 5015  020110  104401  001205                  TYPE    ,$CRLF
 5016  020114                          20$:
 5017  020114  122737  000001  001230          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
 5018  020122  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
 5019  020124  113737  001130  020136          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
 5020  020132  004737  020652                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
 5021  020136     000          21$:    .BYTE   0
 5022  020137     000                  .BYTE   0
 5023  020140  000777          22$:    BR      22$             ;;APT ERROR LOOP
```

```
5024  020142  005777  161006          2$:     TST     @SWR            ;;HALT ON ERROR
5025  020146  100002                          BPL     3$              ;;SKIP IF CONTINUE
5026  020150  000000                          HALT                    ;;HALT ON ERROR!
5027  020152  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
5028  020154  032777  001000  160772    3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
5029  020162  001402                          BEQ     4$              ;;BR IF NO
5030  020164  013716  001124                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
5031  020170  005737  001176          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
5032  020174  001402                          BEQ     5$              ;;BR IF NONE
5033  020176  013716  001176                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
5034  020202                          5$:
5035  020202  022737  017742  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
5036  020210  001001                          BNE     6$              ;;BRANCH IF NO
5037  020212  000000                          HALT                    ;;YES
5038  020214                          6$:
5039  020214  000002                          RTI                     ;;RETURN
5040
5041                                  .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
5042
5043                                  ;;**************************************************************
5044                                  ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
5045                                  ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
5046                                  ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
5047
5048  020216                          $ERRTYP:
5049  020216  104401  001205                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
5050  020222  010046                          MOV     R0,-(SP)        ;;SAVE R0
5051  020224  005000                          CLR     R0              ;;PICKUP THE ITEM INDEX
5052  020226  153700  001130                  BISB    @#$ITEMB,R0
5053  020232  001004                          BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
5054                                                                  ;;TYPE THE PC OF THE ERROR
5055  020234  013746  001132                  MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
5056                                                                  ;;ERROR ADDRESS
5057  020240  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
5058  020242  000426                          BR      6$              ;;GET OUT
5059  020244  005300                  1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
5060  020246  006300                          ASL     R0              ;;       WORK FOR THE ERROR TABLE
5061  020250  006300                          ASL     R0
5062  020252  006300                          ASL     R0
5063  020254  062700  002742                  ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
5064  020260  012037  020270                  MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
5065  020264  001404                          BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
5066  020266  104401                          TYPE                    ;;TYPE THE "ERROR MESSAGE"
5067  020270  000000                  2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
5068  020272  104401  001205                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
5069  020276  012037  020306          3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
5070  020302  001404                          BEQ     5$              ;;SKIP TYPEOUT IF 0
5071  020304  104401                          TYPE                    ;;TYPE THE "DATA HEADER"
5072  020306  000000                  4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
5073  020310  104401  001205                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
5074  020314  011000                  5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
5075  020316  001004                          BNE     7$              ;;GO TYPE THE DATA
5076  020320  012600                  6$:     MOV     (SP)+,R0        ;;RESTORE R0
5077  020322  104401  001205                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
5078  020326  000207                          RTS     PC              ;;RETURN
5079  020330                          7$:
```

```
5080  020330  013046                  MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
5081  020332  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
5082  020334  005710                  TST     (R0)            ;;IS THERE ANOTHER NUMBER?
5083  020336  001770                  BEQ     6$              ;;BR IF NO
5084  020340  104401  020346          TYPE    ,8$             ;;TYPE TWO(2) SPACES
5085  020344  000771                  BR      7$              ;;LOOP
5086  020346  020040      000 8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
5087          020352                  .EVEN
5088
5089                                  .SBTTL  TYPE ROUTINE
5090
5091                                  ;;**************************************************************
5092                                  ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
5093                                  ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
5094                                  ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
5095                                  ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
5096                                  ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
5097                                  ;*
5098                                  ;*CALL:
5099                                  ;*1) USING A TRAP INSTRUCTION
5100                                  ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
5101                                  ;*OR
5102                                  ;*      TYPE
5103                                  ;*      MESADR
5104                                  ;*
5105
5106  020352  105737  001173  $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
5107  020356  100002                  BPL     1$              ;;BR IF YES
5108  020360  000000                  HALT                    ;;HALT HERE IF NO TERMINAL
5109  020362  000430                  BR      3$              ;;LEAVE
5110  020364  010046          1$:     MOV     R0,-(SP)        ;;SAVE R0
5111  020366  017600  000002          MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
5112  020372  122737  000001  001230  CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
5113  020400  001011                  BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
5114  020402  132737  000100  001231  BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
5115  020410  001405                  BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
5116  020412  010037  020422          MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
5117  020416  004737  020642          JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
5118  020422  000000          61$:    .WORD   0               ;;MESSAGE ADDRESS
5119  020424  132737  000040  001231  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
5120  020432  001003                  BNE     60$             ;;YES,SKIP TYPE OUT
5121  020434  112046          2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
5122  020436  001005                  BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
5123  020440  005726                  TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
5124  020442  012600          60$:    MOV     (SP)+,R0        ;;RESTORE R0
5125  020444  062716  000002  3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
5126  020450  000002                  RTI                     ;;RETURN
5127  020452  122716  000011  4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
5128  020456  001430                  BEQ     8$
5129  020460  122716  000200          CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
5130  020464  001006                  BNE     5$
5131  020466  005726                  TST     (SP)+           ;;POP  <CR><LF> EQUIV
5132  020470  104401                  TYPE                    ;;TYPE A CR AND LF
5133  020472  001205                  $CRLF
5134  020474  105037  020630          CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
5135  020500  000755                  BR      2$              ;;GET NEXT CHARACTER
```

```
 5136  020502  004737  020564          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
 5137  020506  123726  001172          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
 5138  020512  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
 5139  020514  013746  001170                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
 6140                                                                  ;;AND THE NULL CHAR.
 5141  020520  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
 5142  020524  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
 5143  020526  004737  020564                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
 5144  020532  105337  020630                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
 5145  020536  000770                          BR      7$              ;;LOOP
 5146
 5147                                  ;HORIZONTAL TAB PROCESSOR
 5148
 5149  020540  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
 5150  020544  004737  020564          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
 5151  020550  132737  000007  020630          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
 5152  020556  001372                          BNE     9$              ;;TAB STOP
 5153  020560  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
 5154  020562  000724                          BR      2$              ;;GET NEXT CHARACTER
 5155  020564  105777  160374          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
 5156  020570  100375                          BPL     $TYPEC
 5157  020572  116677  000002  160366          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
 5158  020600  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
 5159  020606  001003                          BNE     1$              ;;BRANCH IF NO
 5160  020610  105037  020630                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
 5161  020614  000406                          BR      $TYPEX          ;;EXIT
 5162  020616  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
 5163  020624  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
 5164  020626  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
 5165  020630  000000          $CHARCNT:.WORD  0                       ;;CHARACTER COUNT STORAGE
 5166  020632  000207          $TYPEX: RTS     PC
 5167
 5168
 5169                                  .SBTTL  APT COMMUNICATIONS ROUTINE
 5170
 5171                                  ;;**************************************************************
 5172  020634  112737  000001  021100  $ATY1:   MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
 5173  020642  112737  000001  021076  $ATY3:   MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
 5174  020650  000403                          BR      $ATYC
 5175  020652  112737  000001  021100  $ATY4:   MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
 5176  020660                          $ATYC:
 5177  020660  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 5178  020662  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 5179  020664  105737  021076                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
 5180  020670  001450                          BEQ     5$              ;;IF NOT:  BR
 5181  020672  122737  000001  001230          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
 5182  020700  001031                          BNE     3$              ;;IF NOT:  BR
 5183  020702  132737  000100  001231          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
 5184  020710  001425                          BEQ     3$              ;;IF NOT:  BR
 5185  020712  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
 5186  020716  062766  000002  000004          ADD     #2,4(SP)        ;;BUMP RETURN ADDR.
 5187  020724  005737  001210          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
 5188  020730  001375                          BNE     1$              ;;IF NOT:  WAIT
 5189  020732  010037  001224                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
 5190  020736  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
 5191  020740  001376                          BNE     2$
```

```
5192  020742  163700  001224            SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
5193  020746  006200                    ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
5194  020750  010037  001226            MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
5195  020754  012737  000004  001210    MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
5196  020762  000413                    BR      5$
5197  020764  017637  000004  021010  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
5198  020772  062766  000002  000004          ADD     #2,4(SP)        ;;BUMP RETURN ADDRESS
5199  021000  013746  177776            MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
5200  021004  004737  020352            JSR     PC,$TYPE        ;;CALL TYPE MACRO
5201  021010  000000                  4$:     .WORD   0
5202  021012                          5$:
5203  021012  105737  021100          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
5204  021016  001416                    BEQ     12$             ;;IF NOT:  BR
5205  021020  005737  001230            TST     $ENV            ;;RUNNING UNDER APT?
5206  021024  001413                    BEQ     12$             ;;IF NOT:  BR
5207  021026  005737  001210          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
5208  021032  001375                    BNE     11$             ;;IF NOT:  WAIT
5209  021034  017637  000004  001212        MOV     @4(SP),$FATAL   ;;GET ERROR #
5210  021042  062766  000002  000004        ADD     #2,4(SP)        ;;BUMP RETURN ADDR.
5211  021050  005237  001210            INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
5212  021054  105037  021100          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
5213  021060  105037  021077            CLRB    $LFLG           ;;CLEAR LOG FLAG
5214  021064  105037  021076            CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
5215  021070  012601                    MOV     (SP)+,R1        ;;POP STACK INTO R1
5216  021072  012600                    MOV     (SP)+,R0        ;;POP STACK INTO R0
5217  021074  000207                    RTS     PC              ;;RETURN
5218  021076     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
5219  021077     000          $LFLG:  .BYTE   0               ;;LOG FLAG
5220  021100     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
5221          021102                  .EVEN
5222          000200          APTSIZE=200
5223          000001          APTENV=001
5224          000100          APTSPOOL=100
5225          000040          APTCSUP=040
5226
5227                          .SBTTL  POWER DOWN AND UP ROUTINES
5228
5229                          ;;*****************************************************************
5230                          ;POWER DOWN ROUTINE
5231  021102  012737  021254  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
5232  021110  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
5233  021116  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
5234  021120  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
5235  021122  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
5236  021124  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
5237  021126  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
5238  021130  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
5239  021132  017746  160016                MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
5240  021136  010637  021260                MOV     SP,$SAVR6       ;;SAVE SP
5241  021142  012737  021154  000024        MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
5242  021150  000000                        HALT
5243  021152  000776                        BR      .-2             ;;HANG UP
5244
5245                          ;;*****************************************************************
5246                          ;POWER UP ROUTINE
5247  021154  012737  021254  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
```

```
5248  021162  013706  021260            MOV     $SAVR6,SP       ;;GET SP
5249  021166  005037  021260            CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
5250  021172  005237  021260    1$:     INC     $SAVR6          ;;WAIT FOR THE INC
5251  021176  001375                    BNE     1$              ;;OF  WORD
5252  021200  012677  157750            MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
5253  021204  012605                    MOV     (SP)+,R5        ;;POP STACK INTO R5
5254  021206  012604                    MOV     (SP)+,R4        ;;POP STACK INTO R4
5255  021210  012603                    MOV     (SP)+,R3        ;;POP STACK INTO R3
5256  021212  012602                    MOV     (SP)+,R2        ;;POP STACK INTO R2
5257  021214  012601                    MOV     (SP)+,R1        ;;POP STACK INTO R1
5258  021216  012600                    MOV     (SP)+,R0        ;;POP STACK INTO R0
5259  021220  012737  021102  000024    MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
5260  021226  012737  000340  000026    MOV     #340,@#PWRVEC+2 ;;PRIO:7
5261  021234  104401                    TYPE                    ;;REPORT THE POWER FAILURE
5262  021236  021262    $PWRMG: .WORD   $POWER                  ;;POWER FAIL MESSAGE POINTER
5263  021240  042766  000020  000002    BIC     #20,2(SP)       ;;CLEAR "T" BIT
5264  021246  005037  020014            CLR     $TBIT           ;;CLEAR THE "T" BIT FLAG
5265  021252  000002                    RTI
5266  021254  000000    $ILLUP: HALT                            ;;THE POWER UP SEQUENCE WAS STARTED
5267  021256  000776                    BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
5268  021260  000000    $SAVR6: 0                               ;;PUT THE SP HERE
5269  021262  005015  047520  042527  $POWER: .ASCIZ  <15><12>"POWER"
5270  021270  000122
5271
5272
5273                    .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
5274
5275                    ;;**************************************************************
5276                    ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
5277                    ;*OCTAL (ASCII) NUMBER AND TYPE IT.
5278                    ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
5279                    ;*CALL:
5280                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
5281                    ;*      TYPOS                   ;;CALL FOR TYPEOUT
5282                    ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
5283                    ;*      .BYTE   M               ;;M=1 OR 0
5284                    ;*                                      ;;1=TYPE LEADING ZEROS
5285                    ;*                                      ;;0=SUPPRESS LEADING ZEROS
5286                    ;*
5287                    ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
5288                    ;*$TYPOS OR $TYPOC
5289                    ;*CALL:
5290                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
5291                    ;*      TYPON                   ;;CALL FOR TYPEOUT
5292                    ;*
5293                    ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
5294                    ;*CALL:
5295                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
5296                    ;*      TYPOC                   ;;CALL FOR TYPEOUT
5297
5298  021272  017646  000000    $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
5299  021276  116637  000001  021515    MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
5300  021304  112637  021517            MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
5301  021310  062716  000002            ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
5302  021314  000406                    BR      $TYPON
5303  021316  112737  000001  021515  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
```

```
5304   021324   112737   000006   021517          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
5305   021332   112737   000005   021514  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
5306   021340   010346                           MOV     R3,-(SP)        ;;SAVE R3
5307   021342   010446                           MOV     R4,-(SP)        ;;SAVE R4
5308   021344   010546                           MOV     R5,-(SP)        ;;SAVE R5
5309   021346   113704   021517                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
5310   021352   005404                           NEG     R4
5311   021354   062704   000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
5312   021360   110437   021516                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
5313   021364   113704   021515                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
5314   021370   016605   000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
5315   021374   005003                           CLR     R3              ;;CLEAR THE OUTPUT WORD
5316   021376   006105                   1$:     ROL     R5              ;;ROTATE MSB INTO "C"
5317   021400   000404                           BR      3$              ;;GO DO MSB
5318   021402   006105                   2$:     ROL     R5              ;;FORM THIS DIGIT
5319   021404   006105                           ROL     R5
5320   021406   006105                           ROL     R5
5321   021410   010503                           MOV     R5,R3
5322   021412   006103                   3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
5323   021414   105337   021516                  DECB    $OMODE          ;;TYPE THIS DIGIT?
5324   021420   100016                           BPL     7$              ;;BR IF NO
5325   021422   042703   177770                  BIC     #177770,R3      ;;GET RID OF JUNK
5326   021426   001002                           BNE     4$              ;;TEST FOR 0
5327   021430   005704                           TST     R4              ;;SUPPRESS THIS 0?
5328   021432   001403                           BEQ     5$              ;;BR IF YES
5329   021434   005204                   4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
5330   021436   052703   000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
5331   021442   052703   000040           5$:    BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
5332   021446   110337   021512                  MOVB    R3,8$           ;;SAVE FOR TYPING
5333   021452   104401   021512                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
5334   021456   105337   021514           7$:    DECB    $OCNT           ;;COUNT BY 1
5335   021462   003347                           BGT     2$              ;;BR IF MORE TO DO
5336   021464   002402                           BLT     6$              ;;BR IF DONE
5337   021466   005204                           INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
5338   021470   000744                           BR      2$              ;;GO DO THE LAST DIGIT
5339   021472   012605                   6$:     MOV     (SP)+,R5        ;;RESTORE R5
5340   021474   012604                           MOV     (SP)+,R4        ;;RESTORE R4
5341   021476   012603                           MOV     (SP)+,R3        ;;RESTORE R3
5342   021500   016666   000002   000004         MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
5343   021506   012616                           MOV     (SP)+,(SP)
5344   021510   000002                           RTI                     ;;RETURN
5345   021512      000           8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
5346   021513      000                   .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
5347   021514      000           $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
5348   021515      000           $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
5349   021516   000000           $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
5350
5351                           .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
5352
5353                           ;;**************************************************************
5354                           ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
5355                           ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
5356                           ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
5357                           ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
5358                           ;*REPLACED WITH SPACES.
5359                           ;*CALL:
```

```
                                        ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
                                        ;*      TYPDS                   ;;GO TO THE ROUTINE
                                        $TYPDS:
021520                                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
021520  010046                                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
021522  010146                                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
021524  010246                                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
021526  010346                                  MOV     R5,-(SP)        ;;PUSH R5 ON STACK
021530  010546                                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
021532  012746  020200                         MOV     20(SP),R5       ;;GET THE INPUT NUMBER
021536  016605  000020                         BPL     1$              ;;BR IF INPUT IS POS.
021542  100004                                 NEG     R5              ;;MAKE THE BINARY NUMBER POS.
021544  005405                                 MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
021546  112766  000055  000001                 CLR     R0              ;;ZERO THE CONSTANTS INDEX
021554  005000                         1$:     MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
021556  012703  021734                         MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
021562  112723  000040                 2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
021566  005002                         3$:     MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
021570  016001  021724                         SUB     R1,R5           ;;FORM THIS BCD DIGIT
021574  160105                                 BLT     4$              ;;BR IF DONE
021576  002402                                 INC     R2              ;;INCREASE THE BCD DIGIT BY 1
021600  005202                                 BR      3$
021602  000774                         4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
021604  060105                                 TST     R2              ;;CHECK IF BCD DIGIT=0
021606  005702                                 BNE     5$              ;;FALL THROUGH IF 0
021610  001002                                 TSTB    (SP)            ;;STILL DOING LEADING 0'S?
021612  105716                                 BMI     7$              ;;BR IF YES
021614  100407                         5$:     ASLB    (SP)            ;;MSD?
021616  106316                                 BCC     6$              ;;BR IF NO
021620  103003                                 MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
021622  116663  000001  177777         6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
021630  052702  000060                 7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
021634  052702  000040                         MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
021640  110223                                 TST     (R0)+           ;;JUST INCREMENTING
021642  005720                                 CMP     R0,#10          ;;CHECK THE TABLE INDEX
021644  020027  000010                         BLT     2$              ;;GO DO THE NEXT DIGIT
021650  002746                                 BGT     8$              ;;GO TO EXIT
021652  003002                                 MOV     R5,R2           ;;GET THE LSD
021654  010502                                 BR      6$              ;;GO CHANGE TO ASCII
021656  000764                         8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
021660  105726                                 BPL     9$              ;;BR IF NO
021662  100003                                 MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
021664  116663  177777  177776         9$:     CLRB    (R3)            ;;SET THE TERMINATOR
021672  105013                                 MOV     (SP)+,R5        ;;POP STACK INTO R5
021674  012605                                 MOV     (SP)+,R3        ;;POP STACK INTO R3
021676  012603                                 MOV     (SP)+,R2        ;;POP STACK INTO R2
021700  012602                                 MOV     (SP)+,R1        ;;POP STACK INTO R1
021702  012601                                 MOV     (SP)+,R0        ;;POP STACK INTO R0
021704  012600                                 TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
021706  104401  021734                         MOV     2(SP),4(SP)     ;;ADJUST THE STACK
021712  016666  000002  000004                 MOV     (SP)+,(SP)
021720  012616                                 RTI                     ;;RETURN TO USER
021722  000002                         $DTBL:  10000.
021724  023420                                 1000.
021726  001750                                 100.
021730  000144
```

```
                                        10.
021732  000012                  $DBLK:  .BLKW   4
021734  000004
                                .SBTTL  TTY INPUT ROUTINE
                                ;;*****************************************************************
                                .ENABL  LSB

                                ;;*****************************************************************
                                ;;*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 FLAG MODE.
021744  022737  000176  001154  $CKSWR: CMP     #$SWREG,SWR     ;;IS THE SOFT-SWR SELECTED?
021752  001074                          BNE     15$             ;;BRANCH IF NO
021754  105777  157200                  TSTB    @$TKS           ;;CHAR THERE?
021760  100071                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
021762  117746  157174                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
021766  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
021772  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
021776  001062                          BNE     15$             ;;NO, RETURN TO USER
022000  123727  001150  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
022006  001456                          BEQ     15$             ;;BRANCH IF YES

022010  104401  022353                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
022014  104401  022360          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
022020  013746  000176                  MOV     $WREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
022024  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
022026  104401  022371                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
022032  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
022034  005046                          CLR     -(SP)           ;;THE NEW SWR
022036  105777  157116          7$:     TSTB    @$TKS           ;;CHAR THERE?
022042  100375                          BPL     7$              ;;IF NOT TRY AGAIN

022044  117746  157112                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
022050  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII

022054  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
022060  001005                          BNE     10$             ;;BRANCH IF NOT
022062  104401  022346                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
022066  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
022072  000757                          BR      19$             ;;LET'S TRY IT AGAIN

022074  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
022100  001022                          BNE     16$             ;;BRANCH IF NO
022102  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
022106  001403                          BEQ     11$             ;;BRANCH IF YES
022110  016677  000002  157036          MOV     2(SP),@SWR      ;;SAVE NEW SWR
022116  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
022122  104401  001205          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
022126  123727  001151  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS
022134  001003                          BNE     15$             ;;BRANCH IF NOT
022136  012777  000100  157014          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
```

```
5472  022144  000002                  15$:    RTI                     ;;RETURN
5473  022146  004737  020564          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
5474  022152  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
5475  022156  002420                          BLT     18$             ;;BRANCH IF YES
5476  022160  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
5477  022164  003015                          BGT     18$             ;;BRANCH IF YES
5478  022166  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
5479  022172  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
5480  022176  001403                          BEQ     17$             ;;BRANCH IF YES
5481  022200  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
5482  022202  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
5483  022204  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
5484  022206  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
5485  022212  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
5486  022216  000707                          BR      7$              ;;GET THE NEXT ONE
5487  022220  104401  001204          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
5488  022224  000720                          BR      20$             ;;SIMULATE CONTROL-U
5489                                  .DSABL  LSB
5490
5491
5492                                  ;;**************************************************************
5493                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
5494                                  ;*CALL:
5495                                  ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
5496                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
5497                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
5498                                  ;
5499
5500  022226  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
5501  022230  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
5502  022236  105777  156716          1$:     TSTB    @$TKS           ;;WAIT FOR
5503  022242  100375                          BPL     1$              ;;A CHARACTER
5504  022244  117766  156712  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
5505  022252  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
5506  022260  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
5507  022266  001013                          BNE     3$              ;;BRANCH IF NO
5508  022270  105777  156664          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
5509  022274  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
5510  022276  117746  156660                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
5511  022302  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
5512  022306  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
5513  022312  001366                          BNE     2$              ;;IF NOT DISCARD IT
5514  022314  000750                          BR      1$              ;;YES, RESUME
5515  022316  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
5516  022324  002407                          BLT     4$              ;;BRANCH IF YES
5517  022326  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
5518  022334  003003                          BGT     4$              ;;BRANCH IF YES
5519  022336  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
5520  022344  000002                  4$:     RTI                     ;;GO BACK TO USER
5521  022346  052536  005015     000  $CNTLU: .ASCIZ   /^U/<15><12>    ;;CONTROL "U"
5522  022353     136  006507  000012  $CNTLG: .ASCIZ   /^G/<15><12>    ;;CONTROL "G"
5523  022360  005015  053523  020122  $MSWR:  .ASCIZ   <15><12>/SWR = /
5524  022366  020075     000
5525  022371     040  047040  053505  $MNEW:  .ASCIZ   /  NEW =
5526  022376  036440  000040
5527
```

```
                                        .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

022402                                  $SAVREG:
022402  010046                                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
022404  010146                                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
022406  010246                                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
022410  010346                                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
022412  010446                                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
022414  010546                                  MOV     R5,-(SP)        ;;PUSH R5 ON STACK
022416  016646  000022                         MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
022422  016646  000022                         MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
022426  016646  000022                         MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
022432  016646  000022                         MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
022436  000002                                  RTI

                                        ;*RESTORE R0-R5
                                        ;*CALL:
                                        ;*      RESREG
022440                                  $RESREG:
022440  012666  000022                         MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
022444  012666  000022                         MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
022450  012666  000022                         MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
022454  012666  000022                         MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
022460  012605                                  MOV     (SP)+,R5        ;;POP STACK INTO R5
022462  012604                                  MOV     (SP)+,R4        ;;POP STACK INTO R4
022464  012603                                  MOV     (SP)+,R3        ;;POP STACK INTO R3
022466  012602                                  MOV     (SP)+,R2        ;;POP STACK INTO R2
022470  012601                                  MOV     (SP)+,R1        ;;POP STACK INTO R1
022472  012600                                  MOV     (SP)+,R0        ;;POP STACK INTO R0
022474  000002                                  RTI

                                        .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
                                        $DB2D:  SAVREG                  ;;SAVE REGISTERS
022476  104412                                  MOV     2(SP),R2        ;;PICKUP THE DATA POINTER
022500  016602  000002                          MOV     #$DECVL,R0      ;;GET ADDRESS OF "$DECVL" STRING
022504  012700  022656                          MOV     R0,2(SP)        ;;PUT ADDRESS OF ASCIZ STRING ON STACK
022510  010066  000002                          MOV     (R2)+,R1        ;;PICKUP THE BINARY NUMBER
022514  012201                                  MOV     (R2)+,R2
022516  012202
022520  012737  000012  022574                  MOV     #10.,4$         ;;SET UP TO DO 10 CONVERSIONS
022526  012704  022606                          MOV     #$TNPWR,R4      ;;ADDRESS OF TEN POWER
022532  012705  022610                          MOV     #$TNPWR+2,R5
022536  005003                          1$:     CLR     R3              ;;CLEAR PARTIAL
022540  161401                          2$:     SUB     (R4),R1         ;;SUBTRACT TEN POWER
022542  005602                                  SBC     R2
022544  161502                                  SUB     (R5),R2
022546  002402                                  BLT     3$              ;;BR IF TEN POWER TO LARGE
022550  005203                                  INC     R3              ;;ADD 1 TO PARTIAL
022552  000772                                  BR      2$              ;;LOOP
022554  062401                          3$:     ADD     (R4)+,R1        ;;RESTORE SUBTRACTED VALUE
022556  005502                                  ADC     R2
022560  062402                                  ADD     (R4)+,R2
022562  022525                                  CMP     (R5)+,(R5)+     ;;MOVE TO NEXT TEN POWER
022564  052703  000060                          BIS     #'0,R3          ;;CHANGE PARTIAL TO ASCII
022570  110320                                  MOVB    R3,(R0)+        ;;SAVE IT
022572  005327                                  DEC     (PC)+           ;;DONE?
022574  000000                          4$:     .WORD   0
022576  001357                                  BNE     1$              ;;BR IF NO
022600  105020                                  CLRB    (R0)+           ;;TERMINATOR
022602  104413                                  RESREG                  ;;RESTORE REGISTERS
022604  000207                                  RTS     PC              ;;RETURN
022606  145000                          $TNPWR: 145000                  ;;1.0E09
022610  035632                                  35632
022612  160400                                  160400                  ;;1.0E08
022614  002765                                  2765
022616  113200                                  113200                  ;;1.0E07
022620  000230                                  230
022622  041100                                  041100                  ;;1.0E06
022624  000017                                  17
022626  103240                                  103240                  ;;1.0E05
022630  000001                                  1
022632  023420                                  23420                   ;;1.0E04
022634  000000                                  0
022636  001750                                  1750                    ;;1.0E03
022640  000000                                  0
022642  000144                                  144                     ;;1.0E02
022644  000000                                  0
022646  000012                                  12                      ;;1.0E01
022650  000000                                  0
022652  000001                                  1                       ;;1.0E00
022654  000000                                  0
022656  000014                          $DECVL: .BLKB   12.             ;;RESERVE STORAGE FOR ASCIZ STRING

                                        .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
```

```
                                ;;**********************************************************************
                                ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
                                ;*UNSIGNED OCTAL ASCIZ NUMBER.
                                ;*CALL
                                ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
                                ;*      JSR     PC,@#$DB20      ;;CALL THE ROUTINE
                                ;*      RETURN                  ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK

022672  104412          $DB20:  SAVREG                  ;;SAVE ALL REGISTERS
022674  016601  000002          MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
022700  012705  023011          MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
022704  012704  000014          MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
022710  012703  177770          MOV     #^C7,R3         ;;MASK
022714  012100                  MOV     (R1)+,R0        ;;LOWER WORD
022716  012101                  MOV     (R1)+,R1        ;;HIGH WORD
022720  005002                  CLR     R2              ;;TERMINATOR
022722  110245          1$:     MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
022724  010002                  MOV     R0,R2           ;;GET THIS DIGIT
022726  005304                  DEC     R4              ;;COUNT THIS CHARACTER
022730  003007                  BGT     3$              ;;BR IF NOT THE LAST DIGIT
022732  001405                  BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
022734  005205                  INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
022736  010566  000002          MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
022742  104413                  RESREG                  ;;RESTORE ALL REGISTERS
022744  000207                  RTS     PC              ;;RETURN TO USER
022746  006203          2$:     ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
022750  006001          3$:     ROR     R1              ;;POSITION THE BINARY NUMBER FOR
022752  006000                  ROR     R0              ;;      THE NEXT OCTAL DIGIT
022754  006001                  ROR     R1
022756  006000                  ROR     R0
022760  006001                  ROR     R1
022762  006000                  ROR     R0
022764  040302                  BIC     R3,R2           ;;MASK OUT ALL JUNK
022766  062702  000060          ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
022772  000753                  BR      1$              ;;GO PUT IT IN THE DATA TABLE
022774  000016          $OCTVL: .BLKB   14.             ;;RESERVE DATA TABLE

                                .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
                                ;*SW08=1         LOOP ON TEST IN SWR<7:0>
                                ;*CALL
                                ;*      SCOPE           ;;SCOPE=IOT

023012          $SCOPE:
023012  104407                  CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
023014  032777  040000  156132  1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
023022  001114                  BNE     $OVER           ;;YES IF SW14=1
```

```
5696
5697  023024  000416                    ;*****START OF CODE FOR THE XOR TESTER*****
5698  $XTSTR: BR      6$                ;;IF RUNNING ON THE "XOR" TESTER CHANGE
5699                                    ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
5700  023026  013746  000004           ;;SAVE THE CONTENTS OF THE ERROR VECTOR
5701  023032  012737  023052  000004   ;;SET FOR TIMEOUT
5702  023040  005737  177060           ;;TIME OUT ON XOR?
5703  023044  012637  000004           ;;RESTORE THE ERROR VECTOR
5704  023050  000463                    ;;GO TO THE NEXT TEST
5705  023052  022626          5$:       ;;CLEAR THE STACK AFTER A TIME OUT
5706  023054  012637  000004           ;;RESTORE THE ERROR VECTOR
5707  023060  000423                    ;;LOOP ON THE PRESENT TEST
5708  023062                    6$:;*****END OF CODE FOR THE XOR TESTER*****
5709  023062  032777  000400  156064   ;;LOOP ON SPEC. TEST?
5710  023070  001404                    ;;BR IF NO
5711  023072  127737  156056  001116   ;;ON THE RIGHT TEST?    SWR<7:0>
5712  023100  001465                    ;;BR IF YES
5713  023102  105737  001117  2$:       ;;HAS AN ERROR OCCURRED?
5714  023106  001421                    ;;BR IF NO
5715  023110  123737  001131  001117   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
5716  023116  101015                    ;;BR IF NO
5717  023120  032777  001000  156026   ;;LOOP ON ERROR?
5718  023126  001404                    ;;BR IF NO
5719  023130  013737  001124  001122  7$:       ;;SET LOOP ADDRESS TO LAST SCOPE
5720  023136  000446                    ;;
5721  023140  105037  001117  4$:       ;;ZERO THE ERROR FLAG
5722  023144  005037  001174           ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
5723  023150  000415                    ;;ESCAPE TO THE NEXT TEST
5724  023152  032777  004000  155774   ;;INHIBIT ITERATIONS?
5725  023160  001011                    ;;BR IF YES
5726  023162  005737  001216           ;;IF FIRST PASS OF PROGRAM
5727  023166  001406                    ;;      INHIBIT ITERATIONS
5728  023170  005237  001120           ;;INCREMENT ITERATION COUNT
5729  023174  023737  001174  001120   ;;CHECK THE NUMBER OF ITERATIONS MADE
5730  023202  002024                    ;;BR IF MORE ITERATION REQUIRED
5731  023204  012737  000001  001120  1$:       ;;REINITIALIZE THE ITERATION COUNTER
5732  023212  013737  023270  001174   ;;SET NUMBER OF ITERATIONS TO DO
5733  023220  105237  001116  $SVLAD:  ;;COUNT TEST NUMBERS
5734  023224  113737  001116  001214   ;;SET TEST NUMBER IN APT MAILBOX
5735  023232  011637  001122           ;;SAVE SCOPE LOOP ADDRESS
5736  023236  011637  001124           ;;SAVE ERROR LOOP ADDRESS
5737  023242  005037  001176           ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
5738  023246  112737  000001  001131   ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
5739  023254  013777  001116  155674  $OVER:   ;;DISPLAY TEST NUMBER
5740  023262  013716  001122           ;;FUDGE RETURN ADDRESS
5741  023266  000002                    ;;FIXES PS
5742  023270  003720          $MXCNT:  2000.            ;;MAX. NUMBER OF ITERATIONS
5743
5744                          .SBTTL  TRAP DECODER
5745
5746                          ;;**************************************************************
5747                          ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
5748                          ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
5749                          ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
5750                          ;*GO TO THAT ROUTINE.
5751  023272  010046          $TRAP:   MOV     R0,-(SP)         ;;SAVE R0
```

```
5752  023274  016600  000002          MOV     2(SP),R0        ;;GET TRAP ADDRESS
5753  023300  005740                  TST     -(R0)           ;;BACKUP BY 2
5754  023302  111000                  MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
5755  023304  006300                  ASL     R0              ;;POSITION FOR INDEXING
5756  023306  016000  023326          MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
5757  023312  000200                  RTS     R0              ;;GO TO ROUTINE
5758
5759
5760                                  .;THIS IS USE TO HANDLE THE "GETPRI" MACRO
5761
5762  023314  011646          $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
5763  023316  016666  000004  000002  MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
5764  023324  000002                  RTI                     ;;RESTORE THE PSW
5765
5766                                  .SBTTL  TRAP TABLE
5767
5768                                  ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
5769                                  ;*BY THE "TRAP" INSTRUCTION.
5770
5771                                  ;       ROUTINE
5772                                  ;       -------
5773  023326  023314          $TRPAD: .WORD   $TRAP2
5774  023330  020352                  $TYPE   ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
5775  023332  021316                  $TYPOC  ;;CALL=TYPOC     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
5776  023334  021272                  $TYPOS  ;;CALL=TYPOS     TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
5777  023336  021332                  $TYPON  ;;CALL=TYPON     TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
5778  02334C  021520                  $TYPDS  ;;CALL=TYPDS     TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
5779
5780  023342  022014                  $GTSWR  ;;CALL=GTSWR     TRAP+6(104406)  GET SOFT-SWR SETTING
5781
5782  023344  021744                  $CKSWR  ;;CALL=CKSWR     TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
5783  023346  022226                  $RDCHR  ;;CALL=RDCHR     TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
5784  023350  017274                  $RDLIN  ;;CALL=RDLIN     TRAP+11(104411) TTY TYPEIN STRING ROUTINE
5785  023352  022402                  $SAVREG ;;CALL=SAVREG    TRAP+12(104412) SAVE R0-R5 ROUTINE
5786  023354  022440                  $RESREG ;;CALL=RESREG    TRAP+13(104413) RESTORE R0-R5 ROUTINE
5787  023356  016510                  $DSPLY  ;;CALL=DISPLY    TRAP+14(104414) ROUTINE TO TYPE ERROR MESSAGES
5788          000032          $TERM=.-$TRPAD
5789
5790                                  ;;*****************************************************************
5791
5792
5793
5794                                  .SBTTL  SINGLE/DUAL PORT RH70/RM03 DRIVER (REV 5.1)-24-AUG-77
5795                                  ;NEW DRIVE TYPE ID FOR RM02 **********
5796                                  ; 10-AUG-77 ***********
5797
5798                                  ;COPYRIGHT (C) 1977
5799                                  ;DIGITAL EQUIPMENT CORP.
5800                                  ;MAYNARD, MA 01754
5801                                  ;AUTHOR(S): JIM LACEY/CHUCK HESS/C. CHEN
5802
5803                                  ;;*****************************************************************
5804
5805                                  ;STORAGE FOR RMDS, RMER1, RMER2, AND RMMR2 ON AN ERROR "2"
5806                                          ;RMERRS   = RMDS
5807                                          ;RMERRS+2 = RMER1
```

```
                                        ;RMERRS+4 = RMER2
                                        ;RMERRS+6 = RMMR2

023360  000000  000000  000000  RMERRS: .WORD   0,0,0,0
023366  000000

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

023370      000                 DRVACT: .BYTE   0               ;DRIVE 0
023371      000                         .BYTE   0               ;DRIVE 1
023372      000                         .BYTE   0               ;DRIVE 2
023373      000                         .BYTE   0               ;DRIVE 3
023374      000                         .BYTE   0               ;DRIVE 4
023375      000                         .BYTE   0               ;DRIVE 5
023376      000                         .BYTE   0               ;DRIVE 6
023377      000                         .BYTE   0               ;DRIVE 7

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

023400      000                 DRVSTA: .BYTE   0               ;DRIVE 0
023401      000                         .BYTE   0               ;DRIVE 1
023402      000                         .BYTE   0               ;DRIVE 2
023403      000                         .BYTE   0               ;DRIVE 3
023404      000                         .BYTE   0               ;DRIVE 4
023405      000                         .BYTE   0               ;DRIVE 5
023406      000                         .BYTE   0               ;DRIVE 6
023407      000                         .BYTE   0               ;DRIVE 7

                                ;TABLE OF DRIVE TYPES (DRVTYP=8 BYTES)
                                        ;DRVTYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
                                        ;DRVTYP=5 IF DRIVE IS RM02 ******
                                        ;DRVTYP=4 IF DRIVE IS RM03
                                        ;DRVTYP=-1 IF NOT RM03

023410      000                 DRVTYP: .BYTE   0               ;DRIVE 0
023411      000                         .BYTE   0               ;DRIVE 1
023412      000                         .BYTE   0               ;DRIVE 2
023413      000                         .BYTE   0               ;DRIVE 3
023414      000                         .BYTE   0               ;DRIVE 4
023415      000                         .BYTE   0               ;DRIVE 5
023416      000                         .BYTE   0               ;DRIVE 6
023417      000                         .BYTE   0               ;DRIVE 7

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

023420      000                 DPINT:  .BYTE   0               ;DRIVE 0
023421      000                         .BYTE   0               ;DRIVE 1
023422      000                         .BYTE   0               ;DRIVE 2
```

```
5864  023423  000                       .BYTE   0               ;DRIVE 3
5865  023424  000                       .BYTE   0               ;DRIVE 4
5866  023425  000                       .BYTE   0               ;DRIVE 5
5867  023426  000                       .BYTE   0               ;DRIVE 6
5868  023427  000                       .BYTE   0               ;DRIVE 7
5869
5870                              ;TABLE OF PENDING DUAL PORT REQUESTS
5871                                      ;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
5872                                      ;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE
5873
5874  023430  000                 DPRQS:  .BYTE   0               ;DRIVE 0
5875  023431  000                         .BYTE   0               ;DRIVE 1
5876  023432  000                         .BYTE   0               ;DRIVE 2
5877  023433  000                         .BYTE   0               ;DRIVE 3
5878  023434  000                         .BYTE   0               ;DRIVE 4
5879  023435  000                         .BYTE   0               ;DRIVE 5
5880  023436  000                         .BYTE   0               ;DRIVE 6
5881  023437  000                         .BYTE   0               ;DRIVE 7
5882
5883                              ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
5884                                      ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
5885                                      ;"DPB" OF THE I/O OPERATION.
5886
5887  023440  000000              TRNSWT: .WORD   0
5888
5889                              ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
5890                                      ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
5891                                      ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
5892                                      ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
5893                                      ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
5894
5895  023442  000000              SRCHWT: .WORD   0
5896
5897                              ;RM03 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
5898                                      ;ACTDRV=0 IF DRIVER IS INACTIVE
5899                                      ;ACTDRV>0 IF DRIVER IS ACTIVE
5900
5901  023444  000                 ACTDRV: .BYTE   0
5902
5903                              ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
5904                                      ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
5905                                      ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
5906
5907  023445  000                 ACTSTR: .BYTE   0
5908
5909                              ;UNLOAD FLAG (ULDFLG=8 BYTES)
5910                                      ;ULDFLG=0 IF NO UNLOAD COMMAND
5911                                      ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
5912                                      ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
5913
5914  023446  000                 ULDFLG: .BYTE   0               ;DRIVE 0
5915  023447  000                         .BYTE   0               ;DRIVE 1
5916  023450  000                         .BYTE   0               ;DRIVE 2
5917  023451  000                         .BYTE   0               ;DRIVE 3
5918  023452  000                         .BYTE   0               ;DRIVE 4
5919  023453  000                         .BYTE   0               ;DRIVE 5
```

```
5920  023454  000                     .BYTE   0               ;DRIVE 6
5921  023455  000                     .BYTE   0               ;DRIVE 7
5922
5923                                ;LOOK AHEAD COUNT (LACNT=8 BYTES)
5924                                        ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
5925
5926  023456  000                   LACNT:  .BYTE   0               ;DRIVE 0
5927  023457  000                           .BYTE   0               ;DRIVE 1
5928  023460  000                           .BYTE   0               ;DRIVE 2
5929  023461  000                           .BYTE   0               ;DRIVE 3
5930  023462  000                           .BYTE   0               ;DRIVE 4
5931  023463  000                           .BYTE   0               ;DRIVE 5
5932  023464  000                           .BYTE   0               ;DRIVE 6
5933  023465  000                           .BYTE   0               ;DRIVE 7
5934
5935                                ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
5936                                        ;SAVEFG <0 IF SAVE THE RH70/RM03 REGISTERS WHEN THE
5937                                        ;OPERATION IS COMPLETED AS PER (DPB+14).
5938                                        ;SAVEFG=0 IF SAVE THE RH70/RM03 REGISTERS, AS PER
5939                                        ;(DPB+14), AFTER AN ERROR.
5940
5941  023466  000000                SAVEFG: .WORD   0
5942
5943                                ;SEEK FLAG (SEEKFG=1 WORD)
5944                                        ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
5945                                        ;FOR A DATA TRANSFER START A SEARCH COMMAND
5946                                        ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
5947                                        ;DISREGARD THE WINDOW
5948
5949  023470  177777                SEEKFG: .WORD   -1
5950
5951                                ;TIMEOUT TABLE (TIMER=8 WORDS)
5952                                        ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
5953
5954  023472  177777                TIMER:  .WORD   -1      ;DRIVE 0
5955  023474  177777                        .WORD   -1      ;DRIVE 1
5956  023476  177777                        .WORD   -1      ;DRIVE 2
5957  023500  177777                        .WORD   -1      ;DRIVE 3
5958  023502  177777                        .WORD   -1      ;DRIVE 4
5959  023504  177777                        .WORD   -1      ;DRIVE 5
5960  023506  177777                        .WORD   -1      ;DRIVE 6
5961  023510  177777                        .WORD   -1      ;DRIVE 7
5962
5963                                ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
5964                                        ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
5965                                        ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
5966
5967  023512  177777                DTUW:   .WORD   -1
5968
5969                                ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
5970                                        ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
5971                                        ;ATTENTION BIT
5972
5973  023514  001                   ATABIT: .BYTE   1       ;DRIVE 0
5974  023515  002                           .BYTE   2       ;DRIVE 1
5975  023516  004                           .BYTE   4       ;DRIVE 2
```

```
5976  023517  010                     .BYTE   10      ;DRIVE 3
5977  023520  020                     .BYTE   20      ;DRIVE 4
5978  023521  040                     .BYTE   40      ;DRIVE 5
5979  023522  100                     .BYTE   100     ;DRIVE 6
5980  023523  200                     .BYTE   200     ;DRIVE 7
5981
5982                                  ;FSRM03 TO RH70 "MASSBUS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE
5983                                  ;CALLING IT FATAL (MCPEMX=1 WORD)
5984
5985  023524  000003                  MCPEMX: .WORD   3
5986
5987                                  ;STORAGE FOR RMADR (THE FIRST ADDRESS (776700) OF THE RH70/RM03),
5988                                  ;RMVEC (THE VECTOR ADDRESS (254)), AND RMVEC+2 (THE BR LEVEL (5)).
5989
5990  023526  176700                  RMADR:  .WORD   176700
5991  023530  000254  000240          RMVEC:  .WORD   254,5*32.
5992
5993                                  ;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)
5994
5995  023534  000004                  MXLACT: .WORD   4
5996                                  ;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)
5997
5998  023536  001000                  MXDLTA: .WORD   8.*64.
5999                                  ;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)
6000
6001  023540  000200                  MNDLTA: .WORD   2*64.
6002                                  ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)
6003
6004  023542  000005                  MXWNDW: .WORD   5
6005
6006                                  ;DEFINITIONS OF THE RH70/RM03 ADDRESS INDEXES
6007
6008  000000                  RMCS1=0         ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
6009  000002                  RMWC=2          ;WORD COUNT REGISTER (NOT A DRIVE REG)
6010  000004                  RMBA=4          ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
6011  000006                  RMDA=6          ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
6012  000010                  RMCS2=10        ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
6013  000012                  RMDS=12         ;DRIVE STATUS REGISTER (DRIVE REG 01)
6014  000014                  RMER1=14        ;ERROR REGISTER #1 (DRIVE REG. 02)
6015  000016                  RMAS=16         ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
6016  000020                  RMLA=20         ;LOOK AHEAD REGISTER (DRIVE REG. 07)
6017  000022                  RMDB=22         ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
6018  000024                  RMMR1=24        ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
6019  000026                  RMDT=26         ;DRIVE TYPE REGISTER (DRIVE REG. 06)
6020  000030                  RMSN=30         ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
6021  000032                  RMOF=32         ;OFFSET REGISTER (DRIVE REG. 11)
6022  000034                  RMDC=34         ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
6023  000036                  RMHR=36         ;DUMMY  ADDRESS REGISTER (DRIVE REG. 13)
6024  000040                  RMMR2=40        ;MAINTENANCE REGISTER #2
6025  000042                  RMER2=42        ;ERROR REGISTER #2 (DRIVE REG. 15)
6026  000044                  RMEC1=44        ;ECC POSITION REGISTER (DRIVE REG. 16)
6027  000046                  RMEC2=46        ;ECC PATTERN REGISTER (DRIVE REG.  17)
6028
6029                          ;RH70/RM03 DRIVER INITIALIZATION CODE
6030                                  ;THIS ROUTINE WILL DETERMINE WHICH RM03 DRIVES ARE
6031                                  ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
```

```
                        ;TO THE PROPER STATE FOR EACH DRIVE.
                        ;NOTE: THIS ROUTINE CALLS DRVINT
                        ;
                        ;CALL
                        ;
                        ;       JSR     PC,RMINIT
                        ;       RETURN
                        ;
                        ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
                        ;
023544  104412          RMINIT: SAVREG                  ;SAVE R0 - R5
023546  013746  177776          MOV     @#PS,-(SP)      ;SAVE THE PRESENT PROCESSOR STATUS
023552  012737  000240  177776  MOV     #<5*32.>,@#PS   ;CHANGE THE PRIORITY TO 5
023560  004737  031406          JSR     PC,CLRQUE       ;CLEAR ALL REQUEST QUEUES
023564  012701  023360          MOV     #RMERRS,R1      ;FIRST ADDRESS TO BE CLEARED
023570  012702  023470          MOV     #SEEKFG,R2      ;LAST ADDRESS TO BE CLEARED
023574  005021          1$:     CLR     (R1)+           ;CLEAR
023576  020102                  CMP     R1,R2           ;ARE WE DONE?
023600  103775                  BLO     1$              ;BRANCH IF NO
023602  012702  023512          MOV     #DTUW,R2        ;LAST ADDRESS
023606  012721  177777  2$:     MOV     #-1,(R1)+       ;INITIALIZE
023612  020102                  CMP     R1,R2           ;DONE?
023614  101774                  BLOS    2$              ;LOOP IF NO
023616  005037  023400          CLR     DRVSTA          ;SET ALL DRIVES TO OFFLINE
023622  005037  023402          CLR     DRVSTA+2
023626  005037  023404          CLR     DRVSTA+4
023632  005037  023406          CLR     DRVSTA+6
023636  013703  023530          MOV     RMVEC,R3        ;SETUP THE RH70/RM03 VECTOR
023642  012723  026356          MOV     #ISR,(R3)+
023646  013713  023532          MOV     RMVEC+2,(R3)
023652  013704  023526          MOV     RMADR,R4        ;FIRST ADDRESS OF RH70/RM03
023656  012764  000040  000010  MOV     #BIT05,RMCS2(R4)  ;MASSBUS INIT
023664  005001                  CLR     R1              ;START WITH DRIVE 0
023666  004037  023756  3$:     JSR     R0,DRVINT       ;INIT THE DRIVE
023672  000401                  BR      4$              ;'DVA' NOT SET OR PARITY ERROR
023674  000402                  BR      5$              ;NORMAL RETURN
023676  105061  023400  4$:     CLRB    DRVSTA(R1)      ;SET DRIVE STATUS TO OFFLINE
023702  005201          5$:     INC     R1              ;GO TO NEXT DRIVE
023704  042701  177770          BIC     #^C7,R1         ;MASK OUT UNUSED BITS
023710  001366                  BNE     3$              ;BR IF MORE DRIVES TO GO
023712  012701  000007          MOV     #7,R1           ;START WITH DRIVE 7
023716  005037  177776          CLR     @#PS            ;CLEAR THE PROCESSOR STATUS
023722  105761  023420  6$:     TSTB    DPINT(R1)       ;WAITING FOR DRIVE TO SWITCH PORTS ?
023726  001405                  BEQ     8$              ;BR NOT WAITING
023730  004737  031042          JSR     PC,SET.IE       ;SET INTERRUPT
023734  105761  023420  7$:     TSTB    DPINT(R1)       ;DRIVE SWITCHED PORTS ?
023740  001375                  BNE     7$              ;BR IF NOT
023742  005301          8$:     DEC     R1              ;GO TO THE NEXT DRIVE
023744  100366                  BPL     6$              ;CHECK NEXT DRIVE
023746  012637  177776          MOV     (SP)+,@#PS      ;RESTORE THE PROCESSOR STATUS
023752  104413                  RESREG                  ;RESTORE R0 - R5
023754  000207                  RTS     PC              ;BYE-BYE
                        ;DRIVE INITILIZATION ROUTINE
                        ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
                        ;AN RM03. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT22
```

```
                                                ;IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
                                                ;INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
                                                ;DRVSTA IS SET TO THE PROPER CONDITION.
                                        ;CALL
                                        ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
                                        ;       MOV     RMADR,R4        ;UNIBUS ADDRESS OF RH70/RM03 (RMCS1)
                                        ;       JSR     R0,DRVINT       ;CALLED BY A JSR
                                        ;       RETURN1                 ;ERROR OCCURRED (PARITY)
                                        ;       RETURN2                 ;NORMAL RETURN
                                        ;
023756  010546                 DRVINT: MOV     R5,-(SP)        ;SAVE R5
023760  105061  023400          DULP:   CLRB    DRVSTA(R1)      ;START DRIVE STATUS AS OFFLINE
023764  105061  023410                  CLRB    DRVTYP(R1)      ;CLEAR THE DRIVE TYPE INDICATOR
023770  105061  023446                  CLRB    ULDFLG(R1)      ;CLEAR THE UNLOAD FLAG
023774  010164  000010                  MOV     R1,RMCS2(R4)    ;SELECT A DRIVE
024000  112764  000111  000000          MOVB    #111,RMCS1(R4)  ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
024006  032764  010000  000010          BIT     #BIT12,RMCS2(R4) ;NONEXISTENT DRIVE?
024014  001403                          BEQ     1$              ;NO---BRANCH
024016  004737  031042                  JSR     PC,SET.IE       ;GO SET "IE" WITHOUT A "?, E"
024022  000507                          BR      6$              ;LEAVE THIS ROUTINE
024024  105061  023400          1$:     CLRB    DRVSTA(R1)      ;SET DRIVE STATUS TO OFFLINE
024030  032764  004000  000000          BIT     #BIT11,RMCS1(R4)  ;SEE IF DRIVE AVAILABLE
024036  001750                          BEQ     DULP            ;BR IF DRIVE NOT AVAILABLE
024040  004037  030352                  JSR     R0,RD.RM        ;READ THE DRIVE TYPE REG.
024044  000026                          RMDT
024046  024266                          8$                      ;ERROR RETURN ADDRESS
024050  012605                          MOV     (SP)+,R5        ;PUT DRIVE TYPE IN R5
024052  112761  000004  023410          MOVB    #4,DRVTYP(R1)   ;SET RM03 INDICATOR
024060  022705  020024                  CMP     #20024,R5       ;SINGLE PORT RM03 ?
024064  001420                          BEQ     2$              ;BR IF YES
024066  022705  024024                  CMP     #24024,R5       ;DUAL PORT RM03 ?
024072  001415                          BEQ     2$              ;BR IF YES
024074  112761  000005  023410          MOVB    #5,DRVTYP(R1)   ;SET RM02 INDICATOR
024102  022705  020025                  CMP     #20025,R5       ;SINGLE PORT RM02 ?
024106  001407                          BEQ     2$              ;BRANCH IF SO
024110  022705  024025                  CMP     #24025,R5       ;DUAL PORT RM02 ?
024114  001404                          BEQ     2$              ;BRANCH IF SO
024116  112761  177777  023410          MOVB    #-1,DRVTYP(R1)  ;SET INDICATOR TO 'OTHER'
024124  000446                          BR      6$              ;EXIT
024126  012746  000121          2$:     MOV     #121,-(SP)      ;DO A "READ-IN PRESET"
024132  004037  030532                  JSR     R0,WRT.RM
024136  000000                          RMCS1
024140  024266                          8$
024142  012746  010000                  MOV     #BIT12,-(SP)    ;SET FMT22=1
024146  004037  030532                  JSR     R0,WRT.RM
024152  000032                          RMOF
024154  024266                          8$
024156  004037  030352                  JSR     R0,RD.RM        ;READ RMDS
024162  000012                          RMDS
024164  024266                          8$
024166  012605                          MOV     (SP)+,R5        ;AND SAVE IT IN R5
024170  100015                          BPL     4$              ;BRANCH IF ATA=0
024172  116164  023514  000016          MOVB    ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
024200  004037  030352                  JSR     R0,RD.RM        ;FIND OUT WHY ATA=1
024204  000014                          RMER1
```

```
6144  024206  024266                        8$:
6145  024210  006126                              ROL     (SP)+           ;IS IT UNSAFE?
6146  024212  100004                              BPL     4$              ;BR IF NOT
6147  024214  112761  177777  023400              MOVB    #-1,DRVSTA(R1)  ;SET UNSAFE INDICATOR
6148  024222  000407                              BR      6$              ;EXIT
6149  024224  005105                        4$:   COM     R5              ;CHECK MOL, DPR, DRY, AND VV
6150  024226  042705  167077                      BIC     #^C<BIT12!BIT08!BIT07!BIT06>,R5
6151  024232  001003                              BNE     6$              ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
6152  024234  112761  000001  023400              MOVB    #1,DRVSTA(R1)   ;SET DRIVE STATUS TO ONLINE
6153  024242  005720                        6$:   TST     (R0)+           ;STEP OVER THE ERROR RETURN
6154  024244  000410                              BR      8$              ;EXIT
6155  024246  006301                        7$:   ASL     R1              ;CHANGE INDEX TO ADDRESS WORDS
6156  024250  012761  060000  023472              MOV     #60000,TIMER(R1)  ;START 2 SEC TIMER
6157  024256  006201                              ASR     R1              ;RESTORE R1
6158  024260  112761  177777  023420              MOVB    #-1,DPINT(R1)   ;SET PORT INITIALIZE INIDICATOR
6159  024266  012605                        8$:   MOV     (SP)+,R5        ;RESTORE R5
6160  024270  000200                              RTS     R0              ;EXIT
6161
6162                                  ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
6163                                  ;
6164                                  ;CALL
6165                                  ;
6166                                  ;       JSR     R0,@#RM03       ;CALL THE RM03 DRIVER
6167                                  ;       PNTADR                  ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
6168                                  ;       RETURN1                 ;RETURN HERE IF QUEUE IS FULL
6169                                  ;       RETURN2                 ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
6170                                  ;                               ;IS AN ERROR CONDITION
6171
6172  024272  013746  177776        RM03:   MOV     @#PS,-(SP)      ;SAVE THE CALLING STATUS
6173  024276  013737  023532  177776        MOV     RMVEC+2,@#PS    ;DON'T ALLOW ANY RM03 INTERRUPTS
6174  024304  112737  000001  023444        MOVB    #1,ACTDRV       ;SET "ACTIVE DRIVER" FLAG
6175  024312  104412                        SAVREG                  ;SAVE R0 - R5
6176  024314  011002                        MOV     (R0),R2         ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
6177  024316  005062  000016                CLR     16(R2)          ;CLEAR THE STATUS/ERROR INDICATOR
6178  024322  111201                        MOVB    (R2),R1         ;PICKUP THE DRIVE NUMBER
6179  024324  013704  023526                MOV     RMADR,R4        ;UNIBUS ADDRESS OF RMCS1
6180  024330  105761  023400                TSTB    DRVSTA(R1)      ;CHECK DRIVES STATUS
6181  024334  003014                        BGT     1$              ;BRANCH IF ONLINE
6182  024336  105761  023446                TSTB    ULDFLG(R1)      ;UNLOAD COMMAND IN QUEUE?
6183  024342  001036                        BNE     3$              ;BRANCH IF YES
6184  024344  105761  023420                TSTB    DPINT(R1)       ;TRYING TO INIT THE DRIVE
6185  024350  001042                        BNE     5$              ;BR IF YES
6186  024352  004037  023756                JSR     R0,DRVINT       ;GO INIT. THE DRIVE
6187  024356  000434                        BR      4$              ;ERROR RETURN
6188  024360  105761  023400                TSTB    DRVSTA(R1)      ;IS DRIVE STATUS ONLINE?
6189  024364  003445                        BLE     6$              ;BR IF NOT
6190  024366  105761  023430          1$:   TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
6191  024372  001031                        BNE     5$              ;BR IF YES
6192  024374  010164  000010                MOV     R1,RMCS2(R4)    ;SELECT THE DRIVE
6193  024400  004037  031504                JSR     R0,DRVQUE       ;PUT THIS REQUEST IN QUEUE
6194  024404  000460                        BR      9$              ;QUEUE IS FULL
6195  024406  122762  000103  000002        CMPB    #103,2(R2)      ;IS THIS REQ. FOR AN UNLOAD?
6196  024414  001003                        BNE     2$              ;BR IF NO
6197  024416  112761  177777  023446        MOVB    #-1,ULDFLG(R1)  ;SET THE "UNLOAD IN QUEUE" FLAG
6198  024424  105761  023370          2$:   TSTB    DRVACT(R1)      ;IS THIS DRIVE ACTIVE?
6199  024430  001043                        BNE     8$              ;BR IF YES
```

```
6200  024432  004737  024564                        JSR     PC OPT          ;CALL THE OPTIMIZER
6201  024436  000440                                BR      8$
6202  024440  012762  120000  000016  3$:           MOV     #BIT15!BIT13,16(R2) ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
6203  024446  000434                                BR      8$              ;EXIT
6204  024450  004737  025644                  4$:   JSR     PC,CI7          ;GO HANDLE THE PARITY ERROR
6205  024454  000431                                BR      8$
6206  024456  004037  031504                  5$:   JSR     R0,DRVQUE       ;PUT REQUEST IN QUEUE
6207  024462  000431                                BR      9$              ;QUEUE IS FULL
6208  024464  032714  000100                        BIT     #BIT06,(R4)     ;IE BIT SET ?
6209  024470  001023                                BNE     8$              ;YES
6210  024472  004737  031042                        JSR     PC,SET.IE       ;SET THE INTERRUPT
6211  024476  000420                                BR      8$              ;RETURN
6212  024500  105761  023400                  6$:   TSTB    DRVSTA(R1)      ;SEE IF DRIVE OFFLINE OR UNSAFE
6213  024504  002412                                BLT     7$              ;BR IF UNSAFE
6214  024506  012762  140000  000016                MOV     #BIT15!BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
6215  024514  105761  023410                        TSTB    DRVTYP(R1)      ;SEE IF OFFLINE OR NONEXISTENT
6216  024520  001007                                BNE     8$              ;BR IF OFFLINE
6217  024522  012762  100002  000016                MOV     #BIT15!BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
6218  024530  000403                                BR      8$              ;GO TO EXIT
6219  024532  012762  110000  000016  7$:           MOV     #BIT15!BIT12,16(R2) ;DRIVE IS UNSAFE
6220  024540  104413                          8$:   RESREG                  ;RESTORE R0 - R5
6221  024542  005720                                TST     (R0)+           ;SETUP FOR NORMAL RETURN
6222  024544  000401                                BR      10$             ;FINISH UP, THEN EXIT
6223  024546  104413                          9$:   RESREG                  ;RESTORE R0 - R5
6224  024550  005720                          10$:  TST     (R0)+           ;CORRECT THE RETURN ADDRESS
6225  024552  105037  023444                        CLRB    ACTDRV          ;CLEAR "ACTIVE DRIVER" FLAG
6226  024556  012637  177776                        MOV     (SP)+,@#PS      ;RETURN "PS" TO USER LEVEL
6227  024562  000200                                RTS     R0              ;RETURN TO CALLER
6228
6229                                          ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
6230                                          ;
6231                                          ;CALL
6232                                          ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
6233                                          ;       JSR     PC,OPT          ;SETUP A COMMAND
6234                                          ;
6235  024564  104412                          OPT:  SAVREG                  ;SAVE R0 - R5
6236  024566  013746  177776                        MOV     @#PS,-(SP)      ;SAVE PROC. STATUS
6237  024572  146137  023514  023442                BICB    ATABIT(R1),SRCHWT       ;CLEAR LA SEACH FLAG
6238  024600  105061  023430                        CLRB    DPRQS(R1)       ;RESET THE PORT REQ FLAG ****
6239  024604  004737  031560                        JSR     PC,GETREQ       ;GET "DPB" POINTER OF REQUEST
6240  024610  005702                                TST     R2              ;IS THERE A REQUEST IN QUEUE?
6241  024612  001472                                BEQ     7$              ;NO--BRANCH TO EXIT
6242  024614  010164  000010                        MOV     R1,RMCS2(R4)    ;LOAD THE DRIVE ADDRESS *******
6243  024620  012764  000111  000000                MOV     #111,RMCS1(R4)  ;CLEAR THE DRIVE
6244  024626  032764  004000  000000                BIT     #BIT11,RMCS1(R4)        ;DVA SET ?
6245  024634  001446                                BEQ     5$              ;TO PROT REQUEST ,IF NOT
6246  024636  105761  023400                  10$:  TSTB    DRVSTA(R1)      ;IS DRIVE ONLINE?
6247  024642  003014                                BGT     1$              ;YES--BRANCH
6248  024644  004737  031602                        JSR     PC,POPQUE       ;NO--REMOVE REQUEST FROM QUEUE
6249  024650  012762  140000  000016                MOV     #BIT15!BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
6250  024656  105761  023400                        TSTB    DRVSTA(R1)      ;IS DRIVE UNSAFE ?
6251  024662  100053                                BPL     8$              ;BR TO EXIT IF NOT
6252  024664  012762  110000  000016                MOV     #BIT15!BIT12,16(R2) ;SET UNSAFE STATUS/ERROR INDICATOR
6253  024672  000447                                BR      8$              ;BRANCH TO EXIT
6254  024674                                  1$:
6255                                          ;       MOV     #111,-(SP)      ;LOAD COMMAND ONTO THE STACK
```

```
6256                                            ;       JSR     R0,WRT.RM       ;LOAD THE REGISTER
6257                                            ;       RMCS1                   ;REGISTER INCREMENT
6258                                            ;       6$                      ;ERROR RETURN ADDRESS
6259                                            ;       BIT     #BIT11,(R4)     ;DRIVE AVAILABLE ?
6260                                            ;       BEQ     9$              ;BR IF NOT
6261  024674  122762  000150  000002            ;       CMPB    #150,2(R2)      ;IS THE REQUEST FOR I/O?
6262  024702  002403                            ;       BLT     2$              ;YES--BRANCH
6263  024704  004737  025230                    ;       JSR     PC,CI4          ;CALL THE COMMAND INITIATOR
6264  024710  000440                            ;       BR      8$              ;BRANCH TO EXIT
6265  024712  005737  023512          2$:       ;       TST     DTUW            ;DATA TRANSFER UNDERWAY?
6266  024716  002012                            ;       BGE     4$              ;YES--GO START A SEARCH
6267  024720  005737  023470                    ;       TST     SEEKFG          ;DO IMPLIED SEEKS?
6268  024724  100404                            ;       BMI     3$              ;YES---BRANCH
6269  024726  004037  026202                    ;       JSR     R0,LA           ;NO--DO LOOK AHEAD
6270  024732  000427                            ;       BR      8$              ;RETURN HERE ON A PARITY ERROR
6271  024734  000403                            ;       BR      4$              ;GO START A SEARCH
6272  024736  004737  025022          3$:       ;       JSR     PC,CI1          ;START A DATA TRANSFER
6273  024742  000423                            ;       BR      8$
6274  024744  004737  025130          4$:       ;       JSR     PC,CI3          ;START A SEARCH
6275  024750  000420                            ;       BR      8$              ;GO TO THE EXIT
6276  024752  112761  177777  023430  5$:       ;       MOVB    #-1,DPRQS(R1)   ;SET PORT REQUEST INDICATOR
6277  024760  010103                            ;       MOV     R1,R3           ;SET UP TO ADDRESS WORDS
6278  024762  006303                            ;       ASL     R3              ;CONVERT TO WORD INDEX
6279  024764  012763  060000  023472            ;       MOV     #60000,TIMER(R3)  ;START 10 SEC TIMER
6280  024772  000402                            ;       BR      7$              ;EXIT
6281  024774  004737  025644          6$:       ;       JSR     PC,CI7          ;PROCESS THE PARITY ERROR
6282  025000  032714  000100          7$:       ;       BIT     #BIT06,(R4)     ;SEE IF 'IE' ALREADY SET
6283  025004  001002                            ;       BNE     8$              ;BR IF SET
6284  025006  004737  031042                    ;       JSR     PC,SET.IE       ;SET "IE" WITHOUT A "TRE"
6285  025012  012637  177776          8$:       ;       MOV     (SP)+,@#PS      ;RESTORE PROC. STATUS
6286  025016  104413                            ;       RESREG                  ;RESTORE R0 - R5
6287  025020  000207                            ;       RTS     PC
6288
6289                                            ;COMMAND INITIATOR
6290                                            ;
6291                                            ;CALL
6292                                            ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
6293                                            ;       MOV     #DPB,R2         ;ADDRESS OF DPB
6294                                            ;       JSR     PC,CI?          ;CI?= CI1,CI3, OR CI4
6295                                            ;                               ;WHERE:
6296                                            ;                               ;CI1=DATA TRANSFER
6297                                            ;                               ;CI2=SEARCH REQUESTED BY DATA XFER
6298                                            ;                               ;CI4=NOT DATA TRANSFER
6299
6300  025022  004737  031602          CI1:      JSR     PC,POPQUE       ;REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
6301  025026  010237  023440                    MOV     R2,TRNSWT       ;PUT REQ. IN TRANSFER WAIT QUEUE
6302  025032  010203                            MOV     R2,R3           ;DPB ADDRESS TO R3
6303  025034  013704  023526                    MOV     RMADR,R4        ;RMCS1 ADDRESS
6304  025040  010164  000010                    MOV     R1,RMCS2(R4)    ;SELECT DRIVE
6305  025044  062703  000004                    ADD     #4,R3           ;DESIRED WORD COUNT
6306  025050  062704  000002                    ADD     #2,R4           ;RMWC ADDRESS
6307  025054  012324                            MOV     (R3)+,(R4)+     ;LOAD WORD COUNT
6308  025056  012324                            MOV     (R3)+,(R4)+     ;LOAD BUFFER ADDRESS
6309  025060  012346                            MOV     (R3)+,-(SP)     ;LOAD SECTOR AND TRACK
6310  025062  004037  030532                    JSR     R0,WRT.RM       ;CALL THE LOAD(WRITE) ROUTINE
6311  025066  000006                            RMDA                    ;INDEX OF REGISTER TO LOAD
```

```
6312  025070  025644                        CI7                     ;ERROR RETURN ADDRESS
6313  025072  012346                        MOV     (R3)+,-(SP)     ;LOAD CYLINDER ADDRESS
6314  025074  004037  030532                JSR     R0,WRT.RM
6315  025100  000034                        RMDC
6316  025102  025644                        CI7
6317  025104  016246  000002                MOV     2(R2),-(SP)     ;LOAD "COMMAND+GO", "A17&A16", AND "PSEL"
6318  025110  004037  030532                JSR     R0,WRT.RM
6319  025114  000000                        RMCS1
6320  025116  025644                        CI7
6321  025120  010137  023512                MOV     R1,DTUW         ;SET "DATA TRANSFER UNDERWAY"
6322  025124  000137  025606                JMP     CI5
6323  025130  013704  023526        CI3:    MOV     RMADR,R4        ;RMCS1 ADDRESS
6324  025134  010164  000010                MOV     R1,RMCS2(R4)    ;SELECT DRIVE
6325  025140  016246  000012                MOV     12(R2),-(SP)    ;DESIRED CYLINDER ADDRESS
6326  025144  004037  030532                JSR     R0,WRT.RM
6327  025150  000034                        RMDC
6328  025152  025644                        CI7
6329  025154  116203  000010                MOVB    10(R2),R3       ;PICKUP SECTOR ADDRESS
6330                                        SUB     MXWNDW,R3       ;BACKUP BY MAX. SEARCH FOR I O WINDOW
6331                                        BGE     1$
6332                                        ADD     #32.,R3
6333  025160  010346                1$:     MOV     R3,-(SP)        ;COMBINE THE ADJUSTED SECTOR WITH
6334  025162  042716  177740                BIC     #177740,(SP)    ;CHOP OF ALL EXENTED BITS ,IF ANY
6335  025166  116266  000011  000001        MOVB    11(R2),1(SP)    ;THE DESIRED TRACK
6336  025174  004037  030532                JSR     R0,WRT.RM       ;LOAD DESIRED TRACK & SECTOR
6337  025200  000006                        RMDA
6338  025202  025644                        CI7
6339  025204  012746  000131                MOV     #131,-(SP)      ;START A SEARCH
6340  025210  004037  030532                JSR     R0,WRT.RM
6341  025214  000000                        RMCS1
6342  025216  025644                        CI7
6343  025220  156137  023514  023442        BISB    ATABIT(R1),SRCHWT       ;SET "SEARCH WAIT" KEY
6344  025226  000567                        BR      CI5
6345  025230  013704  023526        CI4:    MOV     RMADR,R4        ;RMCS1 ADDRESS
6346  025234  010164  000010                MOV     R1,RMCS2(R4)    ;SELECT DRIVE
6347  025240  116203  000002                MOVB    2(R2),R3        ;PICKUP THE REQUESTED COMMAND
6348  025244  122703  000131                CMPB    #131,R3         ;IS IT A SEARCH COMMAND?
6349  025250  001007                        BNE     1$              ;BRANCH IF NO
6350  025252  016246  000010                MOV     10(R2),-(SP)    ;LOAD DESIRED TRACK & SECTOR
6351  025256  004037  030532                JSR     R0,WRT.RM
6352  025262  000006                        RMDA
6353  025264  025644                        CI7
6354  025266  000403                        BR      2$              ;GO LOAD CYLINDER
6355  025270  122703  000105        1$:     CMPB    #105,R3         ;IS IT A SEEK COMMAND
6356  025274  001007                        BNE     3$              ;BRANCH IF NO
6357  025276  016246  000012        2$:     MOV     12(R2),-(SP)    ;LOAD DESIRED CYLINDER
6358  025302  004037  030532                JSR     R0,WRT.RM
6359  025306  000034                        RMDC
6360  025310  025644                        CI7
6361  025312  000546                        BR      CI6
6362  025314  122703  000115        3$:     CMPB    #115,R3         ;IS IT AN "OFFSET" COMMAND?
6363  025320  001013                        BNE     4$              ;BR IF NO
6364  025322  004037  030352                JSR     R0,RD.RM        ;MERGE THE OFFSET VALUE INTO RMOF
6365  025326  000032                        RMOF                    ;BUT DON'T CHANGE THE UPPER
6366  025330  025644                        CI7
6367  025332  116216  000001                MOVB    1(R2),(SP)      ;BYTE WHEN LOADING THE
```

```
6368  025336  004037  030532          JSR     R0,WRT.RM       ;REGISTER (RMOF)
6369  025342  000032                  RMOF
6370  025344  025644                  CI7
6371  025346  000530                  BR      CI6             ;GO START THE COMMAND
6372  025350  122703  000107  4$:     CMPB    #107,R3         ;IS IT A "RECALIBRATE" COMMAND?
6373  025354  001525                  BEQ     CI6             ;BRANCH IF YES
6374  025356  122703  000117          CMPB    #117,R3         ;IS IT A RETURN TO CENTER?
6375  025362  001522                  BEQ     CI6             ;BRANCH IF YES
6376  025364  122703  000103          CMPB    #103,R3         ;IS IT AN "UNLOAD" COMMAND?
6377  025370  001016                  BNE     5$              ;BRANCH IF NO
6378  025372  112761  000001  023370  MOVB    #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE INDICATOR
6379  025400  105061  023400          CLRB    DRVSTA(R1)      ;PUT DRIVE STATUS TO OFFLINE
6380  025404  112761  000001  023446  MOVB    #1,ULDFLG(R1)   ;SET "UNLOAD IN PROGRESS" FLAG
6381  025412  010346                  MOV     R3,-(SP)        ;START THE "UNLOAD" COMMAND
6382  025414  004037  030532          JSR     R0,WRT.RM
6383  025420  000000                  RMCS1
6384  025422  025644                  CI7
6385  025424  000207                  RTS     PC              ;RETURN TO USER
6386  025426  122703  000143  5$:     CMPB    #143,R3         ;IS IT A "SET FORMAT" COMMAND?
6387  025432  001014                  BNE     6$              ;BRANCH IF NO
6388  025434  004037  030352          JSR     R0,RD.RM        ;READ THE OFFSET REGISTER
6389  025440  000032                  RMOF
6390  025442  025644                  CI7
6391  025444  116266  000001  000001  MOVB    1(R2),1(SP)     ;COMBINE "FMT22","ECI", AND "HCI"
6392  025452  004037  030532          JSR     R0,WRT.RM       ;LOAD "FMT22", "ECI", AND/OR "HCI".
6393  025456  000032                  RMOF
6394  025460  025644                  CI7
6395  025462  000436                  BR      12$
6396  025464  122703  000141  6$:     CMPB    #141,R3         ;IS IT A "GET REGISTER" COMMAND?
6397  025470  001023                  BNE     10$             ;BRANCH IF NO
6398  025472  016203  000006  7$:     MOV     6(R2),R3        ;POINTS TO 1ST ADDRESS OF WHERE
6399                                                          ;TO PUT THE REGISTER(S)
6400  025476  116237  000010  025514  MOVB    10(R2),9$       ;INIT. THE INDEX FOR THE FIRST REG.
6401  025504  116205  000011          MOVB    11(R2),R5       ;INDEX OF LAST REG. TO MOVE
6402  025510  004037  030352  8$:     JSR     R0,RD.RM        ;READ RH70/RM03 REGISTER
6403  025514  000000  9$:     RMCS1                           ;INDEX OF REG. TO READ
6404  025516  025644                  CI7
6405  025520  012623                  MOV     (SP)+,(R3)+     ;GET THE CONTENTS OF RH70/RM03 REG.
6406  025522  023705  025514          CMP     9$,R5           ;LAST REG. BEEN READ?
6407  025526  001414                  BEQ     12$             ;GET OUT IF YES
6408  025530  062737  000002  025514  ADD     #2,9$           ;INCREASE THE INDEX BY 2
6409  025536  000764                  BR      8$              ;LOOP--MORE TO READ
6410  025540  122703  000145  10$:     CMPB    #145,R3         ;IS IT A "SELECT DRIVE" COMMAND?
6411  025544  001405                  BEQ     12$             ;BRANCH IF YES
6412  025546  010346  11$:     MOV     R3,-(SP)        ;LOAD THE COMMAND
6413  025550  004037  030532          JSR     R0,WRT.RM
6414  025554  000000                  RMCS1
6415  025556  025644                  CI7
6416  025560  004737  031602  12$:     JSR     PC,POPQUE       ;REMOVE REQ. FROM QUEUE
6417  025564  052762  000200  000016  BIS     #BIT07,16(R2)   ;SET THE "DONE" BIT
6418  025572  005737  023466          TST     SAVEFG          ;SAVE THE RH70/RM03 REGISTERS?
6419  025576  100002                  BPL     13$             ;BRANCH IF NO
6420  025600  004737  030724          JSR     PC,SVRH70       ;YES--GO SAVE THE REGISTERS
6421  025604  000207  13$:     RTS     PC              ;RETURN TO USER
6422  025606  006301  CI5:     ASL     R1
6423  025610  012761  060000  023472  MOV     #60000,TIMER(R1)        ;SET A ONE SECOND TIMER
```

```
6424  025616  006201                          ASR     R1
6425  025620  112761  000001  023370          MOVB    #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE
6426  025626  000207                          RTS     PC              ;RETURN TO THE USER
6427  025630  010346                  CI6:    MOV     R3,-(SP)        ;LOAD THE COMMAND
6428  025632  004037  030532                  JSR     R0,WRT.RM
6429  025636  000000                          RMCS1
6430  025640  025644                          CI7
6431  025642  000761                          BR      CI5
6432  025644  032764  010000  000010  CI7:    BIT     #BIT12,RMCS2(R4)  ;DRIVE NON-EXISTENT ?
6433                                  ;       BNE     CI8             ;BR IF YES
6434  025652  005702                  1$:     TST     R2              ;ANYTHING IN QUEUE ?
6435                                  ;       BEQ     CI7B            ;BR IF NOT
6436  025654  001001                          BNE     2$              ;BRANCH IF QUEUE IS THERE
6437  025656  000207                          RTS     PC              ;OTHERWISE EXIT
6438  025660  012762  104000  000016  2$:     MOV     #BIT15!BIT11,16(R2)     ;SET "PARITY" ERROR INDICATOR
6439                                  ;       JSR     PC,SVRH70       ;GO SAVE THE RH70/RM03 REGISTERS
6440  025666  012746  000111          CI7B:   MOV     #111,-(SP)      ;DO A "DRIVE CLEAR"
6441  025672  004037  030532                  JSR     R0,WRT.RM
6442  025676  000000                          RMCS1
6443  025700  025744                          CI8
6444  025702  004737  031464          2$:     JSR     PC,EMPTYQ       ;EMPTY THE QUEUE
6445  025706  105061  023430                  CLRB    DPRQS(R1)       ;CLEAR THE PORT REQUEST FLAG
6446  025712  105061  023446                  CLRB    ULDFLG(R1)      ;CLEAR THE UNLOAD IN QUEUE FLAG
6447  025716  105061  023370                  CLRB    DRVACT(R1)      ;DRIVE IS IDLE
6448  025722  020237  023440                  CMP     R2,TRNSWT       ;IF THIS DRIVE HAD AN I/O REQUEST
6449                                  ;       CMP     R1,DTUW         ;IF THIS DRIVE HAD AN I/O REQUEST
6450  025726  001005                          BNE     1$              ;IN PROGRESS CLEAR ALL OF THE FLAGS
6451  025730  005037  023440                  CLR     TRNSWT
6452  025734  012737  177777  023512          MOV     #-1,DTUW
6453  025742  000207                  1$:     RTS     PC
6454  025744  104412                  CI8:    SAVREG                  ;SAVE R0 - R5
6455  025746  032764  010000  000010          BIT     #BIT12,RMCS2(R4)  ;IS 'NED' SET ?
6456                                  ;       BNE     1$              ;BR IF YES
6457  025754  005001                          CLR     R1
6458  025756  005003                          CLR     R3
6459  025760  105761  023370          1$:     TSTB    DRVACT(R1)      ;DRIVE ACTIVE?
6460                                  ;       BEQ     5$              ;BRANCH IF NO
6461  025764  001003                          BNE     22$             ;BRANCH IF IN ACTIVE
6462  025766  105761  023430                  TSTB    DPRQS(R1)       ;PORT REQUEST
6463  025772  001443                          BEQ     5$              ;BRANCH IF NOT
6464  025774  013702  023440          22$:    MOV     TRNSWT,R2       ;GET THE "TRANSFER WAIT" QUEUE
6465  026000  020137  023512                  CMP     R1,DTUW         ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
6466  026004  001402                          BEQ     2$              ;BRANCH IF YES
6467  026006  004737  031560                  JSR     PC,GETREQ       ;GET THE DPB POINTER
6468  026012  005702                  2$:     TST     R2              ;QUEUE ENTRY FOR DRIVE ?
6469  026014  001413                          BEQ     4$              ;BR IF NOT
6470  026016  032764  010000  000010          BIT     #BIT12,RMCS2(R4)  ;'NED' SET ?
6471  026024  001404                          BEQ     3$              ;BR IF NOT
6472  026026  012762  100002  000016          MOV     #BIT15!BIT01,16(R2)  ;SET 'DRIVE NON-EXISTENT' INDICATOR
6473  026034  000403                          BR      4$              ;CONTINUE
6474  026036  012762  102000  000016  3$:     MOV     #BIT15!BIT10,16(R2) ;SET "NON-CLEARABLE PARITY" ERROR INDICATOR
6475                                  ;       JSR     PC,SVRH70       ;SAVE RH70/RM03 REGISTERS
6476  026044  012763  177777  023472  4$:     MOV     #-1,TIMER(R3)   ;STOP THE TIMER
6477  026052  105061  023370                  CLRB    DRVACT(R1)      ;SET "DRIVE ACTIVE" TO IDLE
6478  026056  105061  023430                  CLRB    DPRQS(R1)       ;CLEAR PORT REQUEST FLAG
6479  026062  020137  023512                  CMP     R1,DTUW         ;IS THIS DRIVE SETUP FOR A TRANSFER
```

```
6480  026066  001005                        BNE     5$                ;BR IF NOT
6481  026070  012737  177777  023512        MOV     #-1,DTUW          ;RESET THE INDICATOR
6482  026076  005037  023440                CLR     TRNSWT            ;CLEAR THE TRANSFER QUEUE
6483  026102  105061  023446        5$:     CLRB    ULDFLG(R1)        ;CLEAR UNLOAD FLAG
6484  026106  032764  010000  000010        BIT     #BIT12,RMCS2(R4)  ;'NED' SET ?
6485                                ;       BNE     6$                ;BR IF YES
6486  026114  005201                        INC     R1                ;MOVE TO THE NEXT DRIVE
6487  026116  062703  000002                ADD     #2,R3
6488  026122  042701  177770                BIC     #^C7,R1
6489  026126  001314                        BNE     1$                ;BRANCH IF MORE DRIVES
6490  026130  012737  177777  023512        MOV     #-1,DTUW          ;NO DATA TRANSFERS UNDERWAY
6491  026136  005037  023440                CLR     TRNSWT            ;CLEAR THE 'TRANSFER WAIT' QUEUE
6492  026142  004737  031406                JSR     PC,CLRQUE         ;CLEAR ALL OF THE REQUEST QUEUES
6493  026146  012764  000040  000010        MOV     #BIT05,RMCS2(R4)  ;DO A MASSBUS INIT.
6494  026154  000406                        BR      7$                ;CONTINUE
6495  026156  004737  031464        6$:     JSR     PC,EMPTYQ         ;CLEAR THE DRIVE'S QUEUE
6496  026162  105061  023400                CLRB    DRVSTA(R1)        ;SET DRIVE TO OFFLINE
6497  026166  105061  023410                CLRB    DRVTYP(R1)        ;CLEAR THE DRIVE TYPE INDICATOR
6498  026172  004737  031042        7$:     JSR     PC,SET.IE         ;SET "IE" WITHOUT "TRE"
6499  026176  104413                        RESREG                    ;RESTORE R0 - R5
6500  026200  000207                        RTS     PC                ;RETURN
6501
6502                                ;LOOK AHEAD ROUTINE
6503                                ;
6504                                ;CALL
6505                                ;       MOV     #DRVNUM,R1        ;DRIVE NUMBER
6506                                ;       MOV     #DPB,R2           ;POINT TO DPB
6507                                ;       JSR     R0,LA             ;GO CHECK THE WINDOW
6508                                ;       RETURN1                   ;ERROR RETURN
6509                                ;       RETURN2                   ;START A SEARCH
6510                                ;       RETURN3                   ;START A DATA TRANSFER
6511
6512  026202  013704  023526        LA:     MOV     RMADR,R4          ;GET RMCS1'S ADDRESS
6513  026206  010164  000010                MOV     R1,RMCS2(R4)      ;SELECT DRIVE
6514  026212  004037  030352                JSR     R0,RD.RM          ;READ DRIVE STATUS
6515  026216  000012                        RMDS
6516  026220  026350                        4$                        ;ERROR RETURN ADDRESS
6517  026222  042716  157577                BIC     #^C0200,(SP)      ;ON CYLINDER ?
6518  026226  022726  000200                CMP     #200,(SP)+        ;PIP=0,DRY=1?
6519  026232  001044                        BNE     3$                ;NO
6520  026234  105261  023456                INCB    LACNT(R1)         ;INCREMENT THE LOOK AHEAD COUNT
6521  026240  126137  023456  023534        CMPB    LACNT(R1),MXLACT  ;EXCEED MAX?
6522  026246  003033                        BGT     2$                ;BRANCH IF YES
6523  026250  116203  000010                MOVB    10(R2),R3         ;GET DESIRED SECTOR ADDRESS AND
6524  026254  000303                        SWAB    R3                ;MULT. BY 64--ALIGN WITH
6525  026256  006203                        ASR     R3                ;LOOK AHEAD REGISTER
6526  026260  006203                        ASR     R3
6527  026262  012737  000340  177776        MOV     #340,@#PS         ;PRIORITY LEVEL "7"
6528  026270  004037  030352        6$:     JSR     R0,RD.RM          ;READ LOOK AHEAD REGISTER
6529  026274  000020                        RMLA
6530  026276  026350                        4$
6531  026300  021664  000020                CMP     (SP),RMLA(R4)     ;CORRECT LA NUMBER ?
6532  026304  001402                        BEQ     7$                ;YES
6533  026306  005726                        TST     (SP)+             ;NO,CLEAR STACK
6534  026310  000415                        BR      3$
6535  026312  162603        7$:     SUB     (SP)+,R3          ;CALCULATE THE DELTA
```

```
6536  026314  002002                        BGE     1$
6537  026316  062703  004000                ADD     #(32.*64.),R3   ;MAKE THE DELTA POSITIVE
6538  026322  023703  023536        1$:     CMP     MXDLTA,R3       ;CHECK THE DELTA TO SEE
6539  026326  002406                        BLT     3$              ;IF IT IS WITHIN THE
6540  026330  023703  023540                CMP     MNDLTA,R3       ;WINDOW---IF YES, ZERO
6541  026334  002003                        BGE     3$              ;THE LOOK AHEAD COUNT
6542  026336  105061  023456        2$:     CLRB    LACNT(R1)       ;AND TAKE THE I/O EXIT
6543  026342  005720                        TST     (R0)+
6544  026344  005720                3$:     TST     (R0)+           ;ADJUST THE RETURN ADDRESS
6545  026346  000402                        BR      5$              ;EXIT
6546  026350  004737  025644        4$:     JSR     PC,CI7          ;PROCESS THE ERROR
6547  026354  000200                5$:     RTS     R0              ;RETURN
6548
6549                                ;INTERRUPT SERVICE ROUTINE
6550
6551  026356  112737  000001  023444  ISR:  MOVB    #1,ACTDRV       ;SET "ACTIVE DRIVER" FLAG
6552  026364  104412                        SAVREG                  ;SAVE R0 - R5
6553  026366  013704  023526                MOV     RMADR,R4        ;ADDRESS OF RHSCS1
6554  026372  013701  023512                MOV     DTUW,R1         ;GET "DATA TRANSFER UNDERWAY" INDICATOR
6555  026376  002403                        BLT     1$              ;BRANCH IF NO DATA TRANSFER UNDERWAY
6556  026400  004737  026422                JSR     PC,TD           ;CALL TRANSFER DONE
6557  026404  000402                        BR      2$              ;EXIT
6558  026406  004737  026572        1$:     JSR     PC,SC           ;CALL SPECIAL CONDITIONS
6559  026412  104413                2$:     RESREG                  ;RESTORE R0 - R5
6560  026414  105037  023444                CLRB    ACTDRV          ;CLEAR "ACTIVE DRIVER" FLAG
6561  026420  000002                        RTI                     ;RETURN
6562
6563                                ;TRANSFER DONE ROUTINE
6564
6565  026422  105061  023370        TD:     CLRB    DRVACT(R1)      ;SET DRIVE ACTIVE INDICATOR TO IDLE
6566  026426  012737  177777  023512        MOV     #-1,DTUW        ;NO DATA TRANSFERS UNDERWAY
6567  026434  006301                        ASL     R1
6568  026436  012761  177777  023472        MOV     #-1,TIMER(R1)   ;CANCEL TIMEOUT
6569  026444  006201                        ASR     R1
6570  026446  013702  023440                MOV     TRNSWT,R2       ;GET "DPB" ADDRESS FROM THE
6571  026452  005037  023440                CLR     TRNSWT          ;TRANSFER WAIT QUEUE--CLEAR QUEUE
6572  026456  052762  000200  000016        BIS     #BIT07,16(R2)   ;SET DONE
6573  026464  010164  000010                MOV     R1,RMCS2(R4)    ;SELECT THE DRIVE
6574  026470  004037  030352                JSR     R0,RD.RM        ;TRANSFER ERROR(TRE=1)?
6575  026474  000000                        RMCS1
6576  026476  025644                        CI7
6577  026500  006126                        ROL     (SP)+
6578  026502  100417                        BMI     3$              ;BR IF YES
6579  026504  005737  023466                TST     SAVEFG          ;SAVE THE RH70/RM03 REGISTERS?
6580  026510  100002                        BPL     1$              ;BRANCH IF NO
6581  026512  004737  030724                JSR     PC,SVRH70       ;YES--SAVE THE REGISTERS
6582  026516                        1$:
6583  026516  004737  031560                JSR     PC,GETREQ       ;GET DPB POINTER
6584  026522  005702                        TST     R2              ;ENTRY FOR DRIVE ?
6585  026524  001403                        BEQ     2$              ;BR IF NOT
6586  026526  004737  024564                JSR     PC,OPT          ;CALL OPTIMIZER
6587  026532  000417                        BR      SC              ;CHECK OTHER DRIVES
6588                                ;THE RELEASE DRIVE COMMAND IS FORECD TO ENTER FOR DUAL PORT OPERATION
6589  026534  012714  000113        2$:     MOV     #113,(R4)       ;RELEASE THE DRIVE
6590  026540  000414                        BR      SC              ;CHECK FOR OTHER DRIVES
6591  026542  052762  100100  000016  3$:   BIS     #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
```

```
 6592  026550  004737  031464            JSR     PC,EMPTYQ       ;EMPTY THE "DRIVE'S WAIT" QUEUE
 6593  026554  004737  030724            JSR     PC,SVRH70       ;SAVE THE RH70/RM03 REGISTERS
 6594  026560  012714  040111            MOV     #40111,(R4)     ;ISSUE A "DRIVE CLEAR"
 6595  026564  012714  000113            MOV     #113,(R4)       ;ISSUE A RELEASE TO THE DRIVE
 6596  026570  000400                    BR      SC              ;CHECK FOR OTHER DRIVES
 6597
 6598
 6599
 6600                                    ;SPECIAL CONDITION ROUTINE
 6601
 6602  026572  116403  000016    SC:     MOVB    RMAS(R4),R3     ;READ "RMAS"
 6603  026576  001014                    BNE     2$              ;BRANCH IF ANY 'ATA' BITS SET
 6604  026600  004037  030352            JSR     R0,RD.RM        ;READ CONTROL AND STATUS REGISTER
 6605  026604  000000                    RMCS1
 6606                            ;       CIB
 6607  026606  026626                    1$                      ;EXIT IF FAIL TO READ
 6608  026610  106126                    ROLB    (SP)+           ;IS "IE"=1?
 6609  026612  100405                    BMI     1$              ;YES, NO DRIVES TO CHECK
 6610  026614  004037  031650            JSR     R0,ES.SAV       ;SAVE THE ADDRESS IN '$ESCAPE'
 6611  026620  104001                    ERROR   1               ;REPORT AN ILLEGAL INTERRUPT
 6612  026622  004737  031042            JSR     PC,SET.IE       ;SET INTERRUPT ENABLE
 6613  026626  000207            1$:     RTS     PC              ;RETURN
 6614  026630  005046            2$:     CLR     -(SP)           ;PROCESS ALL DRIVES THAT HAVE
 6615  026632  110316                    MOVB    R3,(SP)         ;AN "ATA"=1
 6616  026634  012703  000001            MOV     #1,R3
 6617  026640  005001                    CLR     R1
 6618  026642  030316            SC3:    BIT     R3,(SP)         ;ATA=1?
 6619  026644  001005                    BNE     SC5             ;YES--BRANCH
 6620  026646  005201            SC4:    INC     R1              ;MOVE TO THE NEXT DRIVE
 6621  026650  106303                    ASLB    R3
 6622  026652  001373                    BNE     SC3             ;BRANCH IF MORE TO CHECK?
 6623  026654  005726                    TST     (SP)+           ;CLEAN OFF THE STACK
 6624  026656  000207                    RTS     PC              ;RETURN TO USER
 6625  026660  105761  023420    SC5:    TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
 6626  026664  001402                    BEQ     1$              ;BR IF NOT
 6627  026666  000137  027602            JMP     SC13            ;PROCESS THE DRIVE
 6628  026672  105761  023430    1$:     TSTB    DPRQS(R1)       ;PORT REQUEST OUTSTANDING ?
 6629  026676  001402                    BEQ     2$              ;BR IF NOT
 6630  026700  000137  027602            JMP     SC13            ;START THE OUTSTANDING COMMAND
 6631  026704  105761  023400    2$:     TSTB    DRVSTA(R1)      ;CHECK THE DRIVE STATUS
 6632  026710  003025                    BGT     5$              ;BRANCH IF ONLINE
 6633  026712  105761  023446            TSTB    ULDFLG(R1)      ;UNLOAD IN PROGRESS?
 6634  026716  003422                    BLE     5$              ;BRANCH IF NOT
 6635  026720  004737  031560            JSR     PC,GETREQ       ;GET DPB POINTER
 6636  026724  004737  030724            JSR     PC,SVRH70       ;SAVE THE RH70/RM03 REGISTERS
 6637  026730  004737  027540            JSR     PC,SC12         ;SAVE RMDS, RMER1, RMER2, AND RMMR2
 6638                                                            ;ALSO DO A DRIVE INIT (DRVINT)
 6639  026734  105761  023400            TSTB    DRVSTA(R1)      ;DID DRIVE COME ONLINE?
 6640  026740  003416                    BLE     6$              ;NO---BRANCH
 6641  026742  032737  040000  023360    BIT     #BIT14,RMERRS   ;WAS THERE AN ERROR?
 6642  026750  001002                    BNE     3$              ;BR IF ERROR
 6643  026752  000137  027430            JMP     SC11            ;NO ERROR
 6644  026756  013705  023362    3$:     MOV     RMERRS+2,R5     ;YES -- PICKUP RMER1 AND
 6645  026762  000504                    BR      SC6A            ;GO PROCESS THE ERROR
 6646  026764  105761  023370    5$:     TSTB    DRVACT(R1)      ;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
 6647  026770  001033                    BNE     SC6             ;BR IF EITHER
```

```
6648  026772  004737  027540                  JSR     PC,SC12         ;SAVE RMDS, RMER1, RMER2, AND RMMR2
6649                                                                    ;ALSO DO A DRVINT
6650  026776  105761  023420          6$:     TSTB    D?INT(R1)       ;TRYING TO INIT THE DRIVE ?
6651  027002  001321                          BNE     SC4             ;BR IF YES, CHECK ON MORE DRIVES
6652  027004  105761  023400                  TSTB    DRVSTA(R1)      ;CHECK ON DRIVE'S STATUS
6653  027010  100412                          BMI     7$              ;BR IF UNSAFE
6654  027012  032737  020000  023364          BIT     #BIT13,RMERRS+4 ;ADDRESS PLUG CHANGED ?
6655  027020  001013                          BNE     8$              ;BR IF YES
6656                                  ;       MOV     #113,-(SP)      ;RELEASE COMMAND
6657  027022  012746  000111                  MOV     #111,-(SP)      ;DRIVE CLEAR
6658  027026  004037  030532                  JSR     R0,WRT.RM       ;WRITE THE COMMAND INTO RMCS1
6659  027032  000000                          RMCS1                   ;REGISTER INDEX
6660  027034  027400                          SC8                     ;PARITY EXIT ADDRESS
6661  027036  011605          7$:     MOV     (SP),R5         ;PICKUP (RMAS) BEFORE THE ERROR CALL
6662  027040  004037  031650                  JSR     R0,ES.SAV       ;SAVE THE ADDRESS IN '$ESCAPE'
6663  027044  104002                          ERROR   2               ;REPORT THE UNEXPECTED ATTENTION
6664  027046  000677                          BR      SC4             ;GO CHECK FOR MORE ATA'S
6665  027050                          8$:
6666  027050  004037  031650                  JSR     R0,ES.SAV       ;SAVE THE ADDRESS IN '$ESCAPE'
6667  027054  104005                          ERROR   5               ;REPORT THE ADDRESS PLUG CHANGE
6668  027056  000673                          BR      SC4             ;CHECK FOR MORE DRIVES
6669  027060  006301          SC6:    ASL     R1              ;SETUP TO ADDRESS WORDS
6670  027062  012761  177777  023472          MOV     #-1,TIMER(R1)   ;STOP THE TIMER
6671  027070  006201                          ASR     R1              ;RESTORE THE DRIVE ADDRESS
6672  027072  004737  031560                  JSR     PC,GETREQ       ;GET THE DPB POINTER FROM THE QUEUE
6673  027076  010164  000010                  MOV     R1,RMCS2(R4)    ;SELECT DRIVE
6674  027102  000137  027430                  JMP     SC11            ;PROCESS THE SEARCH
6675  027106  004037  030352                  JSR     R0,RD.RM        ;READ THE RM03'S STATUS REG.
6676  027112  000012                          RMDS
6677  027114  027400                          SC8
6678  027116  011605                          MOV     (SP),R5         ;AND PUT IT IN R5
6679  027120  006126                          ROL     (SP)+           ;WAS THERE AN ERROR?
6680  027122  100407                          BMI     1$              ;BR IF ERROR
6681  027124  105761  023370                  TSTB    DRVACT(R1)      ;CHECK DRIVE'S STATE
6682  027130  003137                          BGT     SC11            ;BR IF DRIVE ACTIVE WITH ORDER
6683  027132  052762  100210  000016          BIS     #BIT15!BIT07!BIT03,16(R2)  ;INFORM USER OF ERROR RECOVER COMPLETION
6684  027140  000470                          BR      SC7
6685  027142  004037  030352          1$:     JSR     R0,RD.RM        ;READ ERROR REGISTER #1
6686  027146  000014                          RMER1
6687  027150  027400                          SC8
6688  027152  012605                          MOV     (SP)+,R5        ;AND SAVE IT IN R5
6689  027154  004737  030724                  JSR     PC,SVRH70       ;SAVE RH70/RM03 REGISTERS
6690  027160  012746  000111                  MOV     #111,-(SP)      ;ISSUE A DRIVE CLEAR
6691  027164  004037  030532                  JSR     R0,WRT.RM
6692  027170  000000                          RMCS1
6693  027172  027400                          SC8
6694  027174  006105          SC6A:   ROL     R5              ;WAS "UNSAFE" CONDITION =1?
6695  027176  100406                          BMI     1$              ;BRANCH IF YES
6696  027200  005702                          TST     R2              ;ANYTHING IN QUEUE ?
6697  027202  001447                          BEQ     SC7             ;BR IF NOT
6698  027204  052762  100240  000016          BIS     #BIT15!BIT07!BIT05,16(R2)       ;INFORM USER OF ERROR
6699  027212  000443                          BR      SC7
6700  027214  004037  030352          1$:     JSR     R0,RD.RM        ;READ DRIVE STATUS REG. #1
6701  027220  000012                          RMDS
6702  027222  027400                          SC8
6703  027224  011605                          MOV     (SP),R5         ;SAVE RMDS IN R5
```

```
6704  027226  006126                          ROL     (SP)+           ;"ERR"=1?
6705  027230  100011                          BPL     2$              ;BR IF NO--UNSAFE CLEARED
6706  027232  112761  177777  023400          MOVB    #-1,DRVSTA(R1)  ;DRIVE IS UNSAFE
6707  027240  004737  030724                  JSR     PC,SVRH70       ;SAVE RH70/RM03 REGISTERS
6708  027244  052762  110000  000016          BIS     #BIT15!BIT12,16(R2)     ;INFORM USER OF UNSAFE ERROR
6709  027252  000423                          BR      SC7
6710  027254  032705  010000          2$:     BIT     #BIT12,R5       ;"MOL" = 1 ?
6711  027260  001015                          BNE     3$              ;BR IF YES
6712  027262  112761  177777  023370          MOVB    #-1,DRVACT(R1)  ;ACTIVE ERROR RECOVER
6713  027270  112761  000001  023400          MOVB    #1,DRVSTA(R1)   ;ONLINE
6714  027276  006301                          ASL     R1
6715  027300  012761  072460  023472          MOV     #30000.,TIMER(R1) ;START 30 SECOND TIMER
6716  027306  006201                          ASR     R1
6717  027310  000137  026646                  JMP     SC4
6718  027314  052762  100220  000016  3$:     BIS     #BIT15!BIT07!BIT04,16(R2)       ;INFORM USER OF ERROR
6719  027322  105061  023370          SC7:    CLRB    DRVACT(R1)      ;DRIVE IS IDLE
6720                                  ;       JSR     PC,EMPTYQ       ;DUMP THE QUEUE
6721  027326  004737  031602                  JSR     PC,POPQUE       ;REMOVE THE QUEUE
6722  027332  105761  023446                  TSTB    ULDFLG(R1)      ;UNLOAD IN RMOGRESS OR QUEUE?
6723  027336  003002                          BGT     1$              ;BR IF NOT
6724  027340  105061  023446                  CLRB    ULDFLG(R1)      ;CLEAR UNLOAD FLAG
6725  027344  116164  023514  000016  1$:     MOVB    ATABIT(R1),RMAS(R4)  ;CLEAR ATTENTION BIT
6726  027352  105761  023400                  TSTB    DRVSTA(R1)      ;IS THE DRIVE UNSAFE ?
6727  027356  100406                          BMI     2$              ;BR IF IT IS
6728                                  ;       MOV     #113,-(SP)      ;RELEASE COMMAND
6729  027360  012746  000111                  MOV     #111,-(SP)      ;DRIVE CLEAR COMMAND
6730  027364  004037  030532                  JSR     R0,WRT.RM       ;WRITE THE COMMAND INTO RPCS1
6731  027370  000000                          RMCS1                   ;REGISTER INDEX
6732  027372  027400                          SC8                     ;RMS'TY EXIT ADDRESS
6733  027374  000137  026646          2$:     JMP     SC4             ;CHECK FOR MORE DRIVES
6734  027400  105761  023370          SC8:    TSTB    DRVACT(R1)      ;IS DRIVE IDLE?
6735  027404  001405                          BEQ     1$              ;YES--BRANCH
6736  027406  004737  031560                  JSR     PC,GETREQ       ;GET DPB POINTER
6737  027412  004737  025644                  JSR     PC,CI7          ;PROCESS THE PARITY ERROR
6738  027416  000402                          BR      2$              ;CONTINUE
6739  027420                          1$:
6740                                  ;       JSR     PC,CI7          ;PROCESS THE PARITY ERROR
6741  027420  004737  025666                  JSR     PC,CI7B         ;PROCESS THE UNCORRECTABLE PARITY ERROR
6742  027424  000137  026646          2$:     JMP     SC4             ;CHECK MORE DRIVES
6743  027430  105761  023446          SC11:   TSTB    ULDFLG(R1)      ;"UNLOAD IN PROGRESS"?
6744  027434  003402                          BLE     1$              ;BRANCH IF NO
6745  027436  105061  023446                  CLRB    ULDFLG(R1)      ;CLEAR UNLOAD FLAG
6746  027442  105061  023370          1$:     CLRB    DRVACT(R1)      ;SET DRIVE IDLE
6747  027446  136137  023514  023442          BITB    ATABIT(R1),SRCHWT       ;DOING A SEARCH OPERATION FOR
6748                                                                  ;AN I/O COMMAND?
6749  027454  001012                          BNE     2$              ;BRANCH IF YES
6750  027456  004737  031602                  JSR     PC,POPQUE       ;REMOVE REQUEST FROM QUEUE
6751  027462  052762  000200  000016          BIS     #BIT07,16(R2)   ;SET "DONE" BIT
6752  027470  005737  023466                  TST     SAVEFG          ;SAVE THE REGISTERS?
6753  027474  100002                          BPL     2$              ;BRANCH IF NO
6754  027476  004737  030724                  JSR     PC,SVRH70       ;YES--SAVE ALL OF THE RH70/RM03 REG'S
6755  027502  116164  023514  000016  2$:     MOVB    ATABIT(R1),RMAS(R4)  ;CLEAR ATTENTION BIT
6756  027510  146137  023514  023442          BICB    ATABIT(R1),SRCHWT       ;CLEAR IMPLIED SEEK SET
6757  027516  006301                          ASL     R1              ;WORD INDEX
6758  027520  012761  177777  023472          MOV     #-1,TIMER(R1)   ;STOP CLOCK
6759  027526  006201                          ASR     R1              ;RESTORE R1
```

```
6760  027530  004737  024564                  JSR     PC,OPT          ;START A REQUEST
6761  027534  000137  026646                  JMP     SC4             ;CHECK FOR MORE DRIVES
6762  027540  010164  000010          SC12:   MOV     R1,RMCS2(R4)    ;SELECT DRIVE
6763  027544  016437  000012  023360          MOV     RMDS(R4),RMERRS ;SAVE THE FOUR REGISTERS THAT
6764  027552  016437  000014  023362          MOV     RMER1(R4),RMERRS+2      ;WILL TELL US SOMETHING
6765  027560  016437  000042  023364          MOV     RMER2(R4),RMERRS+4
6766  027566  016437  000040  023366          MOV     RMMR2(R4),RMERRS+6
6767                                  ;       JSR     R0,DRVINT       ;INIT. THE STATE OF THE DRIVE
6768                                  ;       BR      1$              ;TAKE ERROR EXIT
6769  027574  000207                          RTS     PC              ;RETURN
6770  027576  005726                  1$:     TST     (SP)+           ;POP PC OFF OF THE STACK
6771  027600  000677                          BR      SC8             ;PROCESS THE PARITY ERROR
6772  027602  006301                  SC13:   ASL     R1              ;SETUP TO ADDRESS WORDS
6773  027604  012761  177777  023472          MOV     #-1,TIMER(R1)   ;STOP THE TIMER
6774  027612  006201                          ASR     R1              ;
6775  027614  010164  000010                  MOV     R1,RMCS2(R4)    ;SELECT THE DRIVE
6776  027620  116164  023514  000016          MOVB    ATABIT(R1),RMAS(R4)     ;CLEAR THE ATTENTION BIT
6777  027626  105761  023420          1$:     TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
6778  027632  001424                          BEQ     2$              ;BR IF NOT
6779  027634  105061  023420                  CLRB    DPINT(R1)       ;CLEAR THE INIT INDICATOR
6780  027640  004037  023756                  JSR     R0,DRVINT       ;GO INIT THE DRIVE
6781  027644  000240                          NOP                     ;DUMMY PARITY ERROR RETURN
6782  027646  105761  023400                  TSTB    DRVSTA(R1)      ;DRIVE ONLINE ?
6783  027652  003014                          BGT     2$              ;BR IF YES -- START ORDER
6784  027654  005702                          TST     R2              ;QUEUE ENTRY FOR THE DRIVE
6785  027656  001426                          BEQ     3$              ;BR IF NOT
6786  027660  004737  031560                  JSR     PC,GETREQ       ;GET DPB ADDRESS
6787  027664  052762  140000  000016          BIS     #BIT15!BIT14,16(R2)     ;INFORM USER THAT DRIVE OFFLINE
6788  027672  004737  030724                  JSR     PC,SVRH70       ;SAVE THE REGISTERS
6789                                  ;       JSR     PC,EMPTYQ       ;EMPTY THE REQUEST QUEUE
6790  027676  004737  031602                  JSR     PC,POPQUE       ;REMOVE THE QUEUE
6791  027702  000414                          BR      3$
6792  027704  032764  004000  000000  2$:     BIT     #BIT11,RMCS1(R4)        ;DVA SET ?
6793  027712  001006                          BNE     4$              ;SET THEN CALL OPT
6794  027714  006301                          ASL     R1
6795  027716  012761  060000  023472          MOV     #60000,TIMER(R1)
6796  027724  006201                          ASR     R1
6797  027726  000402                          BR      3$
6798  027730  004737  024564          4$:     JSR     PC,OPT          ;START THE PENDING REQUEST
6799  027734  000137  026646          3$:     JMP     SC4             ;PROCESS OTHER DRIVES
6800
6801                                  ;/RM03 TIMER ROUTINE
6802                                  ;CALL
6803                                  ;       MOV     #TIME,-(SP)     ;ELASPED TIME IN MILLISECONDS ON THE STACK
6804                                  ;       JSR     PC,RPTMR        ;CALL RM03 TIME ROUTINE
6805
6806  027740  005737  023444          RPTMR:  TST     ACTDRV          ;CHECK "ACTDRV & ACTSTR"
6807  027744  001027                          BNE     4$              ;IF NON ZERO EXIT
6808  027746  112737  000001  023445          MOVB    #1,ACTSTR       ;SET "ACTSTR"
6809  027754  104412                          SAVREG                  ;SAVE R0 - R5
6810  027756  005001                          CLR     R1              ;START WITH DRIVE 0
6811  027760  005003                          CLR     R3
6812  027762  005763  023472          1$:     TST     TIMER(R3)       ;IS THE TIMER RUNNING?
6813  027766  002406                          BLT     2$              ;BRANCH IF NO
6814  027770  166663  000002  023472          SUB     2(SP),TIMER(R3) ;COUNT THE INTERVAL
6815  027776  003002                          BGT     2$              ;BR IF NO SOFTWARE TIMEOUT
```

```
  6816  030000  004737  030030          JSR     PC,STO          ;CALL SOFTWARE TIMEOUT ROUTINE
  6817  030004  005201                  2$:     INC     R1              ;MOVE TO NEXT DRIVE
  6818  030006  005723                          TST     (R3)+
  6819  030010  022701  000010                  CMP     #8.,R1          ;OUT OF DRIVES?
  6820  030014  003362                          BGT     1$              ;BRANCH IF NO
  6821  030016  104413                  3$:     RESREG                  ;RESTORE R0 - R5
  6822  030020  105037  023445                  CLRB    ACTSTR          ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
  6823  030024  012616                  4$:     MOV     (SP)+,(SP)      ;ADJUST THE STACK
  6824  030026  000[illegible]                  RTS     PC              ;RETURN
  6825
  6826                                  ;SOFTWARE TIMEOUT ROUTINE
  6827                                  ;
  6828                                  ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
  6829                                  ;       OR GREATER
  6830                                  ;
  6831                                  ;CALL:  STO
  6832                                  ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
  6833                                  ;       JSR     PC,STO          ;CALL
  6834                                  ;       RETURN
  6835
  6836  030030  010146                  STO:    MOV     R1,-(SP)        ;SAVE R1
  6837  030032  010246                          MOV     R2,-(SP)        ;SAVE R2
  6838  030034  010346                          MOV     R3,-(SP)        ;SAVE R3
  6839  030036  010446                          MOV     R4,-(SP)        ;SAVE R4
  6840  030040  013704  023526                  MOV     RMADR,R4        ;GET ADDRESS OF "RMCS1"
  6841  030044  010164  000010                  MOV     R1,RMCS2(R4)    ;SELECT THE DRIVE
  6842  030050  004037  030352                  JSR     R0,RD.RM        ;READ "DRIVE STATUS REG"
  6843  030054  000012                          RMDS
  6844                                  ;       STO5
  6845  030056  030340                          STO9
  6846  030060  105726                          TSTB    (SP)+           ;IS "DRY"=1?
  6847  030062  100436                          BMI     STO2            ;BR IF YES
  6848  030064  105761  023420          STO1:   TSTB    DPINT(R1)       ;TRYING TO INTIALIZE THE DRIVE ?
  6849  030070  001033                          BNE     STO2            ;BR IF YES
  6850  030072  105761  023430                  TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
  6851  030076  001030                          BNE     STO2            ;BR IF YES
  6852  030100  013702  023440                  MOV     TRNSWT,R2       ;PICKUP TRANSFER UNIT QUEUE
  6853  030104  020137  023512                  CMP     R1,DTUW         ;TRANSFER UNDERWAY ON THIS DRIVE?
  6854  030110  001404                          BEQ     1$              ;BRANCH IF YES
  6855  030112  000137  030340                  JMP     STO9            ;IF NOT DON'T BOTHER DRIVES
  6856  030116  004737  031560                  JSR     PC,GETREQ       ;GET DPB ADDRESS
  6857  030122  052762  101000  000016  1$:     BIS     #BIT15!BIT09,16(R2) ;SET THE ERROR FLAGS
  6858  030130  004737  030724                  JSR     PC,SVRH70       ;SAVE RH70/RM03 REGISTERS
  6859                                  ;       MOV     #BIT05,RMCS2(R4) ;"INIT" THE MASS BUS
  6860  030134  105061  023370                  CLRB    DRVACT(R1)      ;DRIVE IS IDLE
  6861  030140  105061  023446                  CLRB    ULDFLG(R1)      ;CLEAR THE UNLOAD FLAG
  6862  030144  005037  023440                  CLR     TRNSWT          ;CLEAR DPB ADDRESS
  6863  030150  012737  177777  023512          MOV     #-1,DTUW        ;CLEAR THE TRANSFER DRIVE #
  6864  030156  000470                          BR      STO9            ;DON'T BOTHER OTHER DRIVES
  6865  030160  116405  000016          STO2:   MOVB    RMAS(R4),R5     ;READ ATTENTION REG
  6866  030164  136105  023514                  BITB    ATABIT(R1),R5   ;IS ATTENTION FOR THIS DRIVE UP ?
  6867  030170  001007                          BNE     STO3            ;YES--BRANCH
  6868  030172  105761  023420                  TSTB    DPINT(R1)       ;TRYING TO INTIALIZE THE DRIVE ?
  6869  030176  001021                          BNE     STO6            ;BR IF YES - DRIVE NOT ONLINE
  6870  030200  105761  023430                  TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
  6871  030204  001035                          BNE     STO7            ;BR IF YES - NO RESPONSE TO REQUEST
```

```
                                        BR      ST09            ;OTHER WISE EXIT
                                ST03:   TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
                                        BNE     1$              ;BR IF INIT PENDING
                                        TSTB    DPRQS(R1)       ;PORT REQUEST PENDING ?
                                        BEQ     ST09            ;BR IF NOT
                                1$:     MOV     #-1,TIMER(R3)   ;STOP THE TIMER
                                        BR      ST09            ;EXIT
                                ST05:   JSR     PC,CI8          ;GO HANDLE THE PARITY ERROR
                                        BR      ST09
                                ST06:   CLRB    DPINT(R1)       ;CLEAR THE INITIALIZE INDICATOR
                                        CLRB    DRVSTA(R1)      ;SET UNIT OFFLINE
                                        MOV     #-1,TIMER(R3)   ;STOP THE TIMER
                                        JSR     PC,GETREQ       ;GET THE DPB ADDRESS
                                        TST     R2              ;REQUEST IN QUEUE ?
                                        BEQ     ST09            ;BR IF NOT
                                        BIS     #BIT15!BIT14,16(R2)  ;INFORM THE USER DRIVE NOT AVAILABLE
                                        BR      ST08            ;FINISH
                                ST07:   MOV     #-1,TIMER(R3)   ;STOP THE TIMER
                                        CLRB    DPRQS(R1)       ;CLEAR PORT REQUEST INDICATOR
                                        JSR     PC,GETREQ       ;GET DPB ADDRESS
                                        TST     R2              ;QUEUE ENTRY FOR DRIVE ?
                                        BEQ     ST09            ;BR IF NONE
                                        MOV     #BIT15!BIT2,16(R2)  ;INFORM USER OF PORT REQUEST ERROR
                                ST08:   JSR     PC,EMPTYQ       ;CLEAR THE QUEUE FOR THE DRIVE
                                        JSR     PC,SVRH70       ;SAVE THE REGISTERS
                                ST09:   MOV     (SP)+,R4        ;RESTORE R4
                                        MOV     (SP)+,R3        ;RESTORE R3
                                        MOV     (SP)+,R2        ;RESTORE R2
                                        MOV     (SP)+,R1        ;RESTORE R1
                                        RTS     PC              ;RETURN

                                ;ROUTINE TO READ A RH70/RM03 REGISTER
                                ;
                                ;CALL
                                ;       JSR     R0,RD.RM        ;GO READ A REGISTER
                                ;       INDEX                   ;REG. INDEX FROM BASE
                                ;       ERRADR                  ;ERROR ADDRESS--PROCESS ERROR STARTING
                                ;                               ;AT THIS ADDRESS
                                ;       RETURN                  ;CONTENTS OF REG. IS ON THE STACK
                                ;
030352  013737  023524  030520  RD.RM:   MOV     MCPEMX,RD.RM2   ;MAX. RETRYS ALLOWED
030360  011646                          MOV     (SP),-(SP)      ;SAVE R0 FOR RETURN
030362  013737  023526  030376          MOV     RMADR,RD.ADR    ;FORM THE DESIRED ADDRESS
030370  062037  030376                  ADD     (R0)+,RD.ADR    ;USING THE BASE AND THE INDEX
030374  013727                  RD.RM1: MOV     @(PC)+,(PC)+    ;READ THE DESIRED REGISTER OF THE RM03
030376  000000                  RD.ADR: .WORD   0               ;ADDRESS IS FORMED HERE
030400  000000                  RD.WRD: .WORD   0               ;REG. CONTENTS PUT HERE
030402  013766  030400  000002          MOV     RD.WRD,2(SP)    ;RETURN IT TO THE USER
030410  013746  023526                  MOV     RMADR,-(SP)     ;PUT THE ADDRESS ON THE STACK
030414  062716  000010                  ADD     #RMCS2,(SP)     ;FORM THE ADDRESS OF RMCS2
030420  032736  010000                  BIT     #BIT12,@(SP)+   ;CHECK THE 'NED' BIT
030424  001037                          BNE     RD.RM3          ;BR IF DRIVE NON-EXISTENT
030426  017746  173074                  MOV     @RMADR,-(SP)    ;READ RMCS1
030432  032716  020000                  BIT     #BIT13,(SP)     ;DID MCPE SET?
030436  001002                          BNE     1$              ;BRANCH IF YES
030440  022620                          CMP     (SP)+,(R0)+     ;ADJUST FOR RETURN
```

```
6928  030442  000432                        BR      RD.RM4          ;EXIT
6929  030444                          1$:
6930  030444  004037  031650                JSR     R0,ES.SAV       ;SAVE THE ADDRESS IN '$ESCAPE'
6931  030450  104003                        ERROR   3               ;REPORT "MCPE" ERROR
6932  030452  005737  023512                TST     DTUW            ;DATA TRANSFER UNDERWAY?
6933  030456  100405                        BMI     2$              ;NO--BRANCH
6934  030460  032716  040000                BIT     #BIT14,(SP)     ;NO--"TRE"=1?
6935  030464  001402                        BEQ     2$              ;NO--BRANCH
6936  030466  005726                        TST     (SP)+           ;YES--CLEAN OFF THE STACK AND
6937  030470  000415                        BR      RD.RM3          ;TAKE THE FATAL ERROR EXIT
6938  030472  052716  040000          2$:   BIS     #BIT14,(SP)     ;CLEAR "MCPE" BY SENDING A "1" TO "TRE"
6939  030476  000316                        SWAB    (SP)            ;POSITION BEFORE WRITING
6940  030500  013737  023526  030514        MOV     RMADR,3$        ;FORM ADDRESS OF HIGH BYTE
6941  030506  005237  030514                INC     3$
6942  030512  112637                        MOVB    (SP)+,@(PC)+    ;WRITE THE HIGH BYTE OF RMCS1
6943  030514  000000                  3$:   .WORD   0               ;ADDRESS STORAGE
6944  030516  005327                        DEC     (PC)+           ;EXCEEDED MAX. RETRYS
6945  030520  000003                RD.RM2: .WORD   3
6946  030522  002324                        BGE     RD.RM1          ;BRANCH IF NO
6947  030524  011000                RD.RM3: MOV     (R0),R0         ;FATAL ERROR EXIT
6948  030526  012616                        MOV     (SP)+,(SP)
6949  030530  000200                RD.RM4: RTS     R0
6950
6951                                  ;ROUTINE TO WRITE A REGISTER
6952                                  ;
6953                                  ;CALL
6954                                  ;       MOV     DATA,-(SP)      ;DATA TO BE LOADED ON THE STACK
6955                                  ;       JSR     R0,WRT.RM       ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
6956                                  ;       INDEX                   ;INDEX OF THE REGISTER TO BE LOADED
6957                                  ;       ERRADR                  ;ADDRESS TO RETURN TO ON AN ERROR
6958                                  ;       RETURN                  ;ERROR FREE RETURN
6959
6960  030532  013737  023524  030710  WRT.RM: MOV     MCPEMX,WRT.R2   ;MAX RETRYS ALLOWED
6961  030540  016637  000002  030620          MOV     2(SP),WRT.WD    ;SAVE THE WORD TO WRITE
6962  030546  012616                        MOV     (SP)+,(SP)      ;ADJUST THE STACK
6963  030550  012037  030622                MOV     (R0)+,WRT.AD    ;GET INDEX OF REGISTER TO BE WRITTEN
6964  030554  001015                        BNE     1$              ;BRANCH IF NOT RMCS1
6965  030556  122737  000150  030620        CMPB    #150,WRT.WD     ;IS THE COMMAND FOR DATA TRANSFERS?
6966  030564  002411                        BLT     1$              ;YES--DON'T GET THE OLD A16 & A17, & PSEL
6967  030566  004037  030352                JSR     R0,RD.RM        ;NO---COMBINE A16&A17, & PSEL WITH
6968  030572  000000                        RMCS1                   ;THE COMMAND BEFORE SENDING IT TO
6969  030574  030714                        WRT.R3                  ;THE RH70/RM03
6970  030576  000316                        SWAB    (SP)
6971  030600  042716  177770                BIC     #^C7,(SP)
6972  030604  112637  030621                MOVB    (SP)+,WRT.WD+1
6973  030610  063737  023526  030622  1$:   ADD     RMADR,WRT.AD    ;FORM THE ADDRESS OF THE DISK REG.
6974  030616  012737                WRT.R1: MOV     (PC)+,@(PC)+    ;LOAD THE DESIRED REG.
6975  030620  000000                WRT.WD: .WORD   0               ;WORD TO WRITE GOES HERE
6976  030622  000000                WRT.AD: .WORD   0               ;ADDRESS IS FORMED HERE
6977  030624  013746  023526                MOV     RMADR,-(SP)     ;PUT THE ADDRESS ON THE STACK
6978  030630  062716  000010                ADD     #RMCS2,(SP)     ;FORM THE ADDRESS OF RMCS2
6979  030634  032736  010000                BIT     #BIT12,@(SP)+   ;CHECK THE 'NED' BIT
6980  030640  001025                        BNE     WRT.R3          ;BR IF DRIVE NON-EXISTENT
6981  030642  004037  030352                JSR     R0,RD.RM        ;CHECK FOR PARITY ERROR ON WRITE
6982  030646  000014                        RMER1
6983  030650  030714                        WRT.R3
```

```
030652  032726  000010                  BIT     #BIT03,(SP)+
030656  001420                          BEQ     WRT.R4          ;BRANCH IF "PAR=0"
030660  016037  177776  030672          MOV     -2(R0),1$       ;PICKUP THE INDEX
030666  004037  030352                  JSR     R0,RD.RM        ;READ THE REG.
030672  000000                  1$:     .WORD   0               ;REG. INDEX
030674  030714                          WRT.R3                  ;RETURN TO THIS ADDRESS ON ERROR
030676  004037  031650                  JSR     R0,ES.SAV       ;SAVE THE ADDRESS IN '$ESCAPE'
030702  104004                          ERROR   4               ;REPORT THE PARITY ON WRITE ERROR
030704  005726                          TST     (SP)+           ;CLEAR OFF THE STACK
030706  005327                          DEC     (PC)+           ;DECREMENT THE ERROR COUNT
030710  000003                  WRT.R2: .WORD   3               ;RETRY COUNTER
030712  002341                          BGE     WRT.R1          ;TRY AGAIN IF NOT FINISHED
030714  011000                  WRT.R3: MOV     (R0),R0         ;TAKE THE "PARITY ON WRITE" ERROR EXIT
030716  000401                          BR      WRT.R5          ;EXIT
030720  005720                  WRT.R4: TST     (R0)+           ;ADJUST FOR ERROR FREE EXIT
030722  000200                  WRT.R5: RTS     R0

                                ;ROUTINE TO SAVE THE RH70/RP04/5/RM03 REGISTERS AS PER DPB+14
                                ;
                                ;CALL
                                ;       MOV     #DPBNUM,R2      ;DPB POINTER TO R2
                                ;       JSR     PC,SVRH70       ;SAVE THE DRIVES REG'S

030724  104412                  SVRH70: SAVREG                  ;SAVE R0 - R5
030726  005702                          TST     R2              ;QUEUE ENTRY FOR THE DRIVE ?
030730  001442                          BEQ     6$              ;BR IF NONE
030732  013704  023526                  MOV     RMADR,R4
030736  111264  000010                  MOVB    (R2),RMCS2(R4)  ;SELECT DRIVE
030742  016203  000014                  MOV     14(R2),R3       ;GET THE ERROR TABLE POINTER
030746  001433                          BEQ     6$              ;EXIT IF NO ADDRESS
030750  005037  031004                  CLR     3$              ;COUNTER & POINTER
030754  023727  031004  000022  1$:     CMP     3$,#RMDB        ;REACHED THE BUFFER REGISTER ?
030762  001006                          BNE     2$              ;BR IF NOT
030764  032764  000200  000010          BIT     #BIT07,RMCS2(R4)  ;'OR' SET ?
030772  001002                          BNE     2$              ;BR IF SET
030774  005023                          CLR     (R3)+           ;STORE RMDB AS ZEROES
030776  000405                          BR      4$              ;CONTINUE
031000  004037  030352          2$:     JSR     R0,RD.RM        ;READ THE SELECTED REGISTER
031004  000000                  3$:     .WORD   0               ;REGISTER INDEX
031006  031032                          5$                      ;ERROR RETURN ADDRESS
031010  012623                          MOV     (SP)+,(R3)+     ;STORE THE REGISTER CONTENTS
031012  023727  031004  000046  4$:     CMP     3$,#RMEC2       ;REACHED THE END ?
031020  001406                          BEQ     6$              ;BR IF YES
031022  062737  000002  031004          ADD     #2,3$           ;INCREMENT THE REGISTER INDEX
031030  000751                          BR      1$              ;CONTINUE READING THE REGISTERS
031032  004737  025644          5$:     JSR     PC,CI7          ;PROCESS THE UNCORRECTABLE PARITY ERROR
031036  104413                  6$:     RESREG                  ;RESTORE R0 - R5
031040  000207                          RTS     PC              ;RETURN

                                ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
                                ;CALL
                                ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
                                ;       JSR     PC,SET.IE       ;SET "IE"
                                ;       RETURN

031042  010446                  SET.IE: MOV     R4,-(SP)        ;SAVE R4
```

```
 7040  031044  013704  023526                MOV     RMADR,R4          ;PICKUP ADDRESS OF RMCS1
 7041  031050  010164  000010                MOV     R1,RMCS2(R4)      ;SELECT DRIVE
 7042  031054  011446                        MOV     (R4),-(SP)        ;READ RMCS1
 7043  031056  052716  040000                BIS     #BIT14,(SP)       ;SET THE "TRE" BIT OF THE WORD READ
 7044  031062  000316                        SWAB    (SP)              ;ADJUST FOR DATO
 7045  031064  112714  000100                MOVB    #BIT06,(R4)       ;SET "IE"
 7046  031070  032764  010000  000010        BIT     #BIT12,RMCS2(R4)  ;IS "NED"=1?
 7047  031076  001002                        BNE     1$                ;YES--CLEAR "TRE"
 7048  031100  005726                        TST     (SP)+             ;CLEAN OFF THE STACK
 7049  031102  000402                        BR      2$
 7050  031104  112664  000001        1$:     MOVB    (SP)+,1(R4)       ;CLEAR "TRE"
 7051  031110  012604                2$:     MOV     (SP)+,R4          ;RESTORE R4
 7052  031112  000207                        RTS     PC                ;RETURN TO CALLER
 7053
 7054                                ;QUEUE COUNT
 7055  031114     000                QCNT:   .BYTE   0                 ;DRIVE 0
 7056  031115     000                        .BYTE   0                 ;DRIVE 1
 7057  031116     000                        .BYTE   0                 ;DRIVE 2
 7058  031117     000                        .BYTE   0                 ;DRIVE 3
 7059  031120     000                        .BYTE   0                 ;DRIVE 4
 7060  031121     000                        .BYTE   0                 ;DRIVE 5
 7061  031122     000                        .BYTE   0                 ;DRIVE 6
 7062  031123     000                         BYTE   0                 ;DRIVE 7
 7063
 7064                                ;QUEUE INPUT POINTERS
 7065
 7066  031124  031206                QINPT:  .WORD   QDRV0              ;DRIVE 0
 7067  031126  031226                        .WORD   QDRV1             ;DRIVE 1
 7068  031130  031246                        .WORD   QDRV2             ;DRIVE 2
 7069  031132  031266                        .WORD   QDRV3             ;DRIVE 3
 7070  031134  031306                        .WORD   QDRV4             ;DRIVE 4
 7071  031136  031326                        .WORD   QDRV5             ;DRIVE 5
 7072  031140  031346                        .WORD   QDRV6             ;DRIVE 6
 7073  031142  031366                        .WORD   QDRV7             ;DRIVE 7
 7074
 7075                                ;QUEUE OUTPUT POINTERS
 7076
 7077  031144  031206                QOUTPT: .WORD   QDRV0              ;DRIVE 0
 7078  031146  031226                        .WORD   QDRV1             ;DRIVE 1
 7079  031150  031246                        .WORD   QDRV2             ;DRIVE 2
 7080  031152  031266                        .WORD   QDRV3             ;DRIVE 3
 7081  031154  031306                        .WORD   QDRV4             ;DRIVE 4
 7082  031156  031326                        .WORD   QDRV5             ;DRIVE 5
 7083  031160  031346                        .WORD   QDRV6             ;DRIVE 6
 7084  031162  031366                        .WORD   QDRV7             ;DRIVE 7
 7085
 7086  031164  031206                QSTART: .WORD   QDRV0              ;DRIVE 0 START ADDRESS
 7087  031166  031226                QSTOP:  .WORD   QDRV1              ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
 7088  031170  031246                        .WORD   QDRV2             ;STOP DRIVE 1--START DRIVE 2
 7089  031172  031266                        .WORD   QDRV3             ;STOP DRIVE 2--START DRIVE 3
 7090  031174  031306                        .WORD   QDRV4             ;STOP DRIVE 3--START DRIVE 4
 7091  031176  031326                        .WORD   QDRV5             ;STOP DRIVE 4--START DRIVE 5
 7092  031200  031346                        .WORD   QDRV6             ;STOP DRIVE 5--START DRIVE 6
 7093  031202  031366                        .WORD   QDRV7             ;STOP DRIVE 6--START DRIVE 7
 7094  031204  031406                        .WORD   QTERM             ;STOP DRIVE 7
 7095
```

```
                                        ;DRIVE REQUEST QUEUES

031206  000010                  QDRV0:  .BLKW   10
031226  000010                  QDRV1:  .BLKW   10
031246  000010                  QDRV2:  .BLKW   10
031266  000010                  QDRV3:  .BLKW   10
031306  000010                  QDRV4:  .BLKW   10
031326  000010                  QDRV5:  .BLKW   10
031346  000010                  QDRV6:  .BLKW   10
031366  000010                  QDRV7:  .BLKW   10
        031406                  QTERM=.

                                        ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
                                        ;
                                        ;CALL
                                        ;       JSR     PC,CLRQUE

031406  104412                  CLRQUE: SAVREG                  ;SAVE R0 - R5
031410  012702  031114                  MOV     #QCNT,R2        ;ZERO THE QUEUE COUNTS
031414  005022                          CLR     (R2)+           ;DRIVES 0 & 1
031416  005022                          CLR     (R2)+           ;DRIVES 2 & 3
031420  005022                          CLR     (R2)+           ;DRIVES 4 & 5
031422  005022                          CLR     (R2)+           ;DRIVES 6 & 7
031424  012703  000010                  MOV     #8.,R3          ;MOVE THE STARTING
031430  012701  031164                  MOV     #QSTART,R1      ;ADDRESS OF THE QUEUE INTO
031434  012122                  1$:     MOV     (R1)+,(R2)+     ;THE QUEUE INPUT POINTER
031436  005303                          DEC     R3
031440  001375                          BNE     1$
031442  012703  000010                  MOV     #8.,R3          ;MOVE THE STARTING ADDRESS
031446  012701  031164                  MOV     #QSTART,R1      ;OF THE QUEUE INTO THE
031452  012122                  2$:     MOV     (R1)+,(R2)+     ;QUEUE OUTPUT POINTER
031454  005303                          DEC     R3
031456  001375                          BNE     2$
031460  104413                          RESREG                  ;RESTORE R0 - R5
031462  000207                          RTS     PC

                                        ;EMPTY THE QUEUE SPECIFIED BY R1
                                        ;
                                        ;CALL
                                        ;       MOV     DRVNUM,R1       ;DRIVE NUMBER TO R1
                                        ;       JSR     PC,EMPTYQ

031464  105061  031114          EMPTYQ: CLRB    QCNT(R1)        ;CLEAR NUMBER OF ITEMS IN QUEUE
031470  006301                          ASL     R1
031472  016161  031124  031144          MOV     QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
031500  006201                          ASR     R1
031502  000207                          RTS     PC

                                        ;ROUTINE TO PUT A REQUEST IN QUEUE
                                        ;
                                        ;CALL
                                        ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
                                        ;       MOV     #DPB,R2         ;ADDRESS OF PARAMETER BLOCK
                                        ;       JSR     R0,DRVQUE       ;GO PUT REQUEST IN QUEUE
                                        ;       RETURN1                 ;RETURN HERE IF QUEUE IS FULL
                                        ;       RETURN2                 ;RETURN HERE IF REQUEST IS IN QUEUE
```

```
7152
7153  031504  122761  000010  031114  DRVQUE: CMPB    #10,QCNT(R1)      ;IS QUEUE FULL?
7154  031512  001421                          BEQ     2$                ;BR IF YES-TAKE RETURN1
7155  031514  105261  031114                  INCB    QCNT(R1)          ;INCREMENT QUEUE COUNT
7156  031520  006301                          ASL     R1
7157  031522  010271  031124                  MOV     R2,@QINPT(R1)     ;PUT THIS REQUEST IN QUEUE
7158  031526  062761  000002  031124          ADD     #2,QINPT(R1)      ;UPDATE THE QUEUE POINTER
7159  031534  026161  031124  031166          CMP     QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
7160  031542  001003                          BNE     1$                ;BRANCH IF NO
7161  031544  016161  031164  031124          MOV     QSTART(R1),QINPT(R1) ;YES--RESET POINTER
7162  031552  006201                  1$:     ASR     R1
7163  031554  005720                          TST     (R0)+             ;TAKE RETURN 2
7164  031556  000200                  2$:     RTS     R0                ;RETURN TO USER
7165
7166                                          ;ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
7167                                          ;
7168                                          ;CALL
7169                                          ;       MOV     #DRVNUM,R1        ;DRIVE NUMBER TO R1
7170                                          ;       JSR     PC,GETREQ         ;GO GET THE REQUEST
7171                                          ;       RETURN                    ;R2="DPB" ADDRESS OF THE REQUEST
7172                                          ;                                 ;R2=0 IF NO REQUEST IN QUEUE
7173
7174  031560  005002                  GETREQ: CLR     R2
7175  031562  105761  031114                  TSTB    QCNT(R1)          ;IS THERE ANY REQUEST IN QUEUE?
7176  031566  001404                          BEQ     2$                ;NO---BRANCH
7177  031570  006301                  1$:     ASL     R1
7178  031572  017102  031144                  MOV     @QOUTPT(R1),R2    ;PICKUP "DPB" POINTER FOR THIS DRIVE
7179  031576  006201                          ASR     R1
7180  031600  000207                  2$:     RTS     PC                ;RETURN TO USER
7181
7182                                          ;ROUTINE TO "POP" THE REQUEST FROM QUEUE
7183                                          ;
7184                                          ;CALL
7185                                          ;       MOV     #DRVNUM,R1        ;DRIVE NUMBER TO R1
7186                                          ;       JSR     PC,POPQUE         ;CALL TO REMOVE REQUEST
7187                                          ;       RETURN                    ;R2=ADDRESS OF DPB REMOVED
7188
7189  031602  105361  031114          POPQUE: DECB    QCNT(R1)          ;DECREMENT QUEUE COUNT
7190  031606  006301                          ASL     R1
7191  031610  017102  031144                  MOV     @QOUTPT(R1),R2    ;GET THE "DPB" POINTER
7192  031614  005071  031144                  CLR     @QOUTPT(R1)       ;REMOVE DPB ADDRESS FROM THE QUEUE
7193  031620  062761  000002  031144          ADD     #2,QOUTPT(R1)     ;UPDATE THE QUEUE POINTER
7194  031626  026161  031144  031166          CMP     QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
7195  031634  001003                          BNE     1$                ;NO--BRANCH TO EXIT
7196  031636  016161  031164  031144          MOV     QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
7197  031644  006201                  1$:     ASR     R1
7198  031646  000207                          RTS     PC                ;RETURN TO USER
7199
7200                                          ;ROUTINE TO SAVE THE CONTENTS OF '$ESCAPE' WHEN THE DRIVER
7201                                          ;REPORTS AN ERROR DIRECTLY.
7202                                          ;
7203                                          ;CALL
7204                                          ;       JSR     R0,ES.SAV
7205                                          ;       ERROR   N                 ;;THE ERROR CALL
7206                                          ;       RETURN                    ;THE RETURN IS PAST THE ERROR CALL
7207
```

```
 7208  031650  012037  031664          ES.SAV: MOV     (R0)+,1$        ;GET THE ERROR CALL
 7209  031654  013746  001176                  MOV     $ESCAPE,-(SP)   ;SAVE THE ADDRESS IN '$ESCAPE'
 7210  031660  005037  001176                  CLR     $ESCAPE         ;CLEAR THE ESCAPE RETURN
 7211  031664  000000          1$:     .WORD   0               ;THE ERROR CALL IS MOVED HERE
 7212  031666  012637  001176                  MOV     (SP)+,$ESCAPE   ;RESTORE THE ESCAPE ADDRESS
 7213  031672  000200                          RTS     R0              ;RETURN
 7214
 7215                                  ;;*****************************************************************
 7216
 7217                                  .SBTTL  DATA, CONTROL, & STATUS BLOCKS
 7218
 7219                                  ;;*****************************************************************
 7220
 7221                                  .SBTTL  ROUTINE TO SIZE MEMORY
 7222
 7223                                  ;;*****************************************************************
 7224                                  ;*CALL:
 7225                                  ;*      JSR     PC,$SIZE
 7226                                  ;*      RETURN
 7227                                  ;*$LSTAD WILL CONTAIN THE LAST AVAILABLE MEMORY LOCATION
 7228
 7229  031674  010046          $SIZE:  MOV     R0,-(SP)        ;;SAVE R0 ON THE STACK
 7230  031676  010146                  MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
 7231  031700  013746  000004          MOV     @#ERRVEC,-(SP)  ;;SAVE PRESENT ERROR VECTOR PS & PC
 7232  031704  013746  000006          MOV     @#ERRVEC+2,-(SP)
 7233  031710  010600                  MOV     SP,R0           ;;SAVE THE STACK POINTER
 7234                                  ;;SET THE ERRVEC PS TO THE PRESENT PS
 7235  031712  104400                  TRAP                    ;;PUSH OLD PSW AND PC ON STACK
 7236  031714  012637  000006          MOV     (SP)+,@#ERRVEC+2        ;;SAVE THE PSW IN @#ERRVEC+2
 7237  031720  012737  031740  000004  MOV     #2$,@#ERRVEC    ;;SET FOR TIMEOUT
 7238  031726  012701  020000          MOV     #20000,R1       ;;FIRST ADDRESS
 7239  031732  005711          1$:     TST     (R1)            ;;TEST THIS ADDRESS
 7240  031734  005721                  TST     (R1)+           ;;STEP TO NEXT ADDRESS
 7241  031736  000775                  BR      1$              ;;TRY ANOTHER
 7242  031740  162701  000002  2$:     SUB     #2,R1           ;;DROP BACK
 7243  031744  010006                  MOV     R0,SP           ;;RESTORE THE STACK
 7244  031746  012637  000006          MOV     (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
 7245  031752  012637  000004          MOV     (SP)+,@#ERRVEC
 7246  031756  010137  031770          MOV     R1,$LSTAD       ;;LAST ADDRESS
 7247  031762  012601                  MOV     (SP)+,R1        ;;RESTORE R1
 7248  031764  012600                  MOV     (SP)+,R0        ;;RESTORE R0
 7249  031766  000207                  RTS     PC
 7250  031770  000000          $LSTAD: .WORD   0               ;;CONTAINS THE LAST ADDRESS
 7251
 7252                                  .SBTTL  BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH11
 7253                                  ;THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS
 7254                                  ;OF THE RH11 IS SETUP FOR THE PROPER ADDRESS.
 7255                                  ;IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
 7256                                  ;REQUIRED.
 7257                                  ;NOTE: THIS ROUTINE DESTROYS R0-R4
 7258                                  ;CALL
 7259                                  ;
 7260                                  ;       JSR     PC,BUSADR
 7261                                  ;       RETURN
 7262
 7263  031772  104412          BUSADR: SAVREG          ;SAVE ALL REG
```

```
7264  031774  012700  001274          1$:     MOV     #$RMADR,R0      ;FIRST ADDRESS
7265  032000  104401  032142                  TYPE    ,MRMCS1         ;"RMCS1="
7266  032004  011046                          MOV     (R0),-(SP)      ;PRESENT RMCS1 ADDRESS
7267  032006  104402                          TYPOC                   ;TYPE IT
7268  032010  104401  036604                  TYPE    ,LINSP          ;2 SPACES
7269  032014  104411                          RDLIN                   ;GET THE ENTRY
7270  032016  012601                          MOV     (SP)+,R1        ;ADDRESS OF ASCII TEXT
7271  032020  004537  032164                  JSR     R5,CK.NUM       ;ENTER AND STORE THE NEW ADDRESS
7272  032024  000763                          BR      1$              ;ERROR EXIT
7273  032026  012700  001276          2$:     MOV     #$RMVEC,R0      ;VECTOR ADDRESS
7274  032032  104401  032153                  TYPE    ,MRHVEC         ;"RHVEC="
7275  032036  011046                          MOV     (R0),-(SP)      ;PRESENT RH11 VECTOR ADDRESS ON THE STACK
7276  032040  104402                          TYPOC                   ;TYPE IT
7277  032042  104401  036604                  TYPE    ,LINSP          ;2 SPACES
7278  032046  104411                          RDLIN                   ;READ THE ENTRY
7279  032050  012601                          MOV     (SP)+,R1        ;ASCII TEXT ADDRESS
7280  032052  004537  032164                  JSR     R5,CK.NUM       ;ENTER AND STORE NEW ADDRESS
7281  032056  000763                          BR      2$              ;ERROR EXIT
7282  032060  012700  001274          3$:     MOV     #$RMADR,R0      ;FIRST ADDRESS OF NEW PARAMETERS
7283  032064  012701  023526                  MOV     #RMADR,R1       ;FIRST ADDRESS OF WHERE TO PUT THEM
7284  032070  013705  000004                  MOV     @#ERRVEC,R5     ;SAVE ERROR VECTOR
7285  032074  012737  032120  000004          MOV     #4$,@#ERRVEC    ;LOAD NEW VECTOR ADDRESS
7286  032102  005777  147166                  TST     @$RMADR         ;LEGAL I/O ADDRESS ?
7287  032106  010537  000004                  MOV     R5,@#ERRVEC     ;YES,IF NOT TRAP TO TIMEOUT
7288  032112  012021                          MOV     (R0)+,(R1)+     ;LOAD THE BUS ADDRESS
7289  032114  012021                          MOV     (R0)+,(R1)+     ;LOAD VECTOR ADDRESS
7290  032116  000407                          BR      6$              ;COMMON EXIT
7291  032120  012716  032126          4$:     MOV     #5$,(SP)        ;SET RETURN ADDRESS
7292  032124  000002                          RTI                     ;RETRUN FROM TIME OUT TRAP
7293  032126  104006          5$:     ERROR   6                       ;NO RESPONSE FROM RMADR
7294  032130  010537  000004                  MOV     R5,@#ERRVEC     ;RESTORE THE TIME OUT TRAP
7295  032134  000717                          BR      1$              ;TRY AGAIN
7296  032136  104413          6$:     RESREG                          ;RESTORE ALL REG
7297  032140  000207                          RTS     PC
7298
7299  032142  046522  051503  020061  MRMCS1: .ASCIZ  @RMCS1 = @
7300  032150  020075     000
7301  032153     122  053110  041505  MRHVEC: .ASCIZ  @RHVEC = @
7302  032160  036440  000040
7303
7304                                  .SBTTL  CK.NUM - CHECK NUMBER (OCTAL)
7305                                  ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
7306                                  ;AND FORMS AN OCTAL NUMBER IN R2
7307                                  ;CALL:
7308                                  ;       MOV     #ADR,R1         ;ADDRESS OF ASCIZ STRING
7309                                  ;       JSR     R5,CK.NUM       ;R5 CHANGED
7310                                  ;       RET                     ;ERROR EXIT
7311                                  ;       RET                     ;NORMAL EXIT
7312  032164  010246          CK.NUM: MOV     R2,-(SP)                ;SAVE R2
7313  032166  010346                  MOV     R3,-(SP)                ;SAVE R3
7314  032170  010446                  MOV     R4,-(SP)                ;SAVE R4
7315  032172  012703  000006          MOV     #6,R3                   ;MAX OCTAL DIGITS IN THE NUMBER
7316  032176  005002                  CLR     R2                      ;FINAL OCTAL VALUE
7317  032200  112104          1$:     MOVB    (R1)+,R4                ;GET CURRENT POINTED BYTE
7318  032202  001424                  BEQ     3$                      ;BRANCH IF TERMINATOR DETECTED
7319  032204  120427  000060          CMPB    R4,#'0                  ;SMALLER THAN ASCII-0 ?
```

```
7320  032210  103425                  BLO     5$              ;YES,ERROR EXIT
7321  032212  120427  000067          CMPB    R4,#'7          ;LARGER THAN ASCII-7 ?
7322  032216  101022                  BHI     5$              ;YES,ERROR EXIT
7323  032220  006302                  ASL     R2              ;SHIFT LEFT
7324  032222  103420                  BCS     5$              ;
7325  032224  006302                  ASL     R2              ;ONE
7326  032226  103416                  BCS     5$              ;
7327  032230  006302                  ASL     R2              ;OCTAL DIGIT
7328  032232  103414                  BCS     5$              ;ERROR IF CARRY BIT SET
7329  032234  042704  177770          BIC     #177770,R4      ;CHOP OFF HIGHER BITS
7330  032240  060402                  ADD     R4,R2           ;APPENDING CURRENT DIGIT TO NUMBER
7331  032242  005303                  DEC     R3              ;DECREMENT BYTE COUNT
7332  032244  001401                  BEQ     2$              ;BRANCH IF LAST BYTE
7333  032246  000754                  BR      1$              ;LOOPING BACK
7334  032250  112104          2$:     MOVB    (R1)+,R4        ;CHECK TERMINATOR
7335  032252  001004                  BNE     5$              ;ERROR EXIT
7336  032254  005702          3$:     TST     R2              ;FINAL VALUE = 0
7337  032256  001402                  BEQ     5$              ;YES, TAKE ERROR EXIT
7338  032260  010210                  MOV     R2,(R0)         ;REPLACE THE ORIGINAL VALUE
7339  032262  005725          4$:     TST     (R5)+           ;ADJUST FOR NORMAL RETURN
7340  032264  012604          5$:     MOV     (SP)+,R4        ;RESTORE R4
7341  032266  012603                  MOV     (SP)+,R3        ;RESTORE R3
7342  032270  012602                  MOV     (SP)+,R2        ;RESTORE R2
7343  032272  000205                  RTS     R5              ;EXIT
7344
7345                          ;;*****************************************************************
7346
7347                          .SBTTL  ERROR MESSAGES
7348
7349                          ;;*****************************************************************
7350
7351                          .NLIST  BEX

032274  044122  030067  030450  EM1:    .ASCIZ  /RH70(11) INTERRUPT OCCURRED (RMAS = 0)/
032343     125  042516  050130  EM2:    .ASCIZ  /UNEXPECTED ATTENTION OCCURRED/
032401     115  051501  041123  EM3:    .ASCIZ  /MASSBUS PARITY ERROR (MCPE=1)/
032437     115  051501  041123  EM4:    .ASCIZ  /MASSBUS PARITY ERROR (PAR=1)/
032474  042101  051104  051505  EM5:    .ASCIZ  /ADDRESS PLUG CHANGE BIT SET/
032530  044122  030461  033450  EM6:    .ASCIZ  /RH11(70) DIDN'T RESPOND TO ADDRESSING/
032576  047125  047503  051122  EM10:   .ASCIZ  /UNCORRECTABLE MASSBUS PARITY ERROR/
032641     106  052101  046101  EM11:   .ASCIZ  /FATAL MASSBUS PARITY ERROR/
032674  042520  051522  051511  EM12:   .ASCIZ  /PERSISTENT DEVICE UNSAFE/
032725     117  042520  040522  EM13:   .ASCIZ  /OPERATION NOT COMPLETED WITHIN TIME LIMIT/
032777     104  044522  042526  EM14:   .ASCIZ  /DRIVE WENT OFFLINE/
033022  047516  051040  051505  EM15:   .ASCIZ  /NO RESPONSE TO PORT REQUEST/
```

```
033056  042510  042101  051105  EM20:   .ASCIZ  /HEADER CRC ERROR/
033077     104  052101  020101  EM21:   .ASCIZ  /DATA CHECK ('DCK') ERROR/
033130  051127  052111  020105  EM22:   .ASCIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') SET/
033203     127  044522  042524  EM23:   .ASCIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET/
033262  042510  042101  051105  EM24:   .ASCIZ  /HEADER READ ERROR - 'FMT' BIT DROPPED/
033330  042510  042101  051105  EM25:   .ASCIZ  /HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR/
033411     106  051117  040515  EM26:   .ASCIZ  /FORMAT ERROR ('FER')/
033436  042510  042101  051105  EM27:   .ASCIZ  /HEADER COMPARE ('HCE') ERROR/
033473     115  051511  042503  EM30:   .ASCIZ  /MISCELLANEOUS DRIVE ERROR/
033525     117  042520  040522  EM31:   .ASCIZ  /OPERATION INCOMPLETE ('OPI') ERROR/
033570  051104  053111  020105  EM32:   .ASCIZ  /DRIVE TIMING ('DTE') ERROR/
033623     120  051101  052111  EM33:   .ASCIZ  /PARITY ('PAR') ERROR AFTER OPERATION STARTED/
033700  051127  052111  020105  EM34:   .ASCIZ  /WRITE CLOCK FAILURE ('WCF') ERROR/
033742  047111  040526  044514  EM35:   .ASCIZ  /INVALID ADDRESS ('IAE') ERROR/
034000  051127  052111  020105  EM36:   .ASCIZ  /WRITE LOCK ('WLE') ERROR/
034031     104  052101  020101  EM37:   .ASCIZ  /DATA CHECk ('DCK') SET DURING WRITE CHECK COMMAND/
034113     122  030510  024061  EM40:   .ASCIZ  /RH11(70) OR UNIBUS TRANSFER ERROR/
034155     102  051525  040440  EM41:   .ASCIZ  /BUS ADDRESS OR WORD COUNT INCORRECT/
034221     104  052101  020101  EM42:   .ASCIZ  /DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED/
034302  040503  023516  020124  EM43:   .ASCIZ  /CAN'T MATCH DATA READ WITH A PATTERN/
034347     105  051122  051117  EM44:   .ASCIZ  /ERROR BIT(S) SET, BUT NO ERROR SIGNALED BY THE RH11/
034433     105  041503  046040  EM45:   .ASCIZ  /ECC LOGIC FAILURE - POSITION REGISTER VALUE NOT VALID
034521     102  051525  040440  EM46:   .ASCIZ  /BUS ADDRESS AND WORD COUNT NOT CONSISTENT/
034573     123  042505  020113  EM50:   .ASCIZ  /SEEK INCOMPLETE ('SKI') ERROR/
034631     120  047522  051107  EM51:   .ASCIZ  /PROGRAM DETECTED POSITIONING ERROR/
034674  051104  053111  020105  EM60:   .ASCIZ  /DRIVE UNSAFE ERROR/
034717     000  000040          EM61:   .ASCIZ  /  /
                               .EVEN
034722  046522  051501  020040  DH1:    .ASCIZ  /RMAS
```

```
035731     104  044522  042526  DH2:    .ASCIZ  /DRIVE   RMDS    RMER1   RMER2   RMMR2    RMAS    /
035013     104  044522  042526  DH3:    .ASCIZ  /DRIVE   REG ADR   DATA  /
035044  051104  053111  020105  DH4:    .ASCIZ  /DRIVE   REG ADR   GOOD    BAD    /
035106  046522  042101  020122  DH6:    .ASCIZ  /RMADR   /
035117     104  044522  042526  DH7:    .ASCIZ  /DRIVE   RMCS1   RMWC    RMBA    RMDA    /
035170  046522  051503  020062  DH10:   .ASCIZ  /RMCS2   RMDS    RMER1   RMAS    RMDB    /
035241     122  046515  030522  DH11:   .ASCIZ  /RMMR1   RMDT    RMOF    RMDC    RMMR2   /
035312  046522  051105  020062  DH12:   .ASCIZ  /RMER2   RMEC1   RMEC2   /
        035344                          .EVEN

                                .LIST   BEX

7352
7353  035344  001324  000000          DT1:    .WORD   ATTN,0
7354  035350  001222  023360  023362  DT2:    .WORD   DRIVE,RMERRS,RMERRS+2,RMERRS+4,RMERRS+6,ATTN,0
7355  035356  023364  023366  001324
7356  035364  000000
7357  035366  001222  030376  030400  DT3:    .WORD   DRIVE,RD.ADR,RD.WRD,0
7358  035374  000000
7359  035376  001222  030622  030620  DT4:    .WORD   DRIVE,WRT.AD,WRT.WD,RD.WRD,0
7360  035404  030400  000000
7361  035410  023526  000000          DT6:    .WORD   RMADR,0
7362  035414  001222  040000  040002  DT7:    .WORD   DRIVE,RM.REG,RM.REG+2,RM.REG+4,RM.REG+6,0
7363  035422  040[illegible]  040006  000000
7364  035430  040[illegible]  040012  040014  DT10:   .WORD   RM.REG+10,RM.REG+12,RM.REG+14,RM.REG+16,RM.REG+22,0
7365  035436  040016  040022  000000
7366  035444  040024  040026  040032  DT11:   .WORD   RM.REG+24,RM.REG+26,RM.REG+32,RM.REG+34,RM.REG+40,0
7367  035452  040034  040040  000000
7368  035460  040042  040044  040046  DT12:   .WORD   RM.REG+42,RM.REG+44,RM.REG+46,0
7369  035466  000000
7370
7371                            ;
7372                            ;
7373  035470     000          DF1:    .BYTE   0
7374  035471     000     000     000  DF2:    .BYTE 0,0,0,0,0,0
7375  035474     000     000     000
7376  035477     000     000     000  DF3:    .BYTE   0,0,0
7377  035502     000     000     000  DF4:    .BYTE   0,0,0,0
7378  035505     000
7379                                    .EVEN
7380
7381                            .NLIST  BEX

035506  052523  051502  051531  MESG1:  .ASCIZ  /SUBSYS  /
035517     104  044522  042526  MESG2:  .ASCIZ  /DRIVE(S) /
035531      52  020052  052123  MESG3:  .ASCIZ  /** STARTING PASS 1 **/
035557     115  052517  052116  MESG4:  .ASCIZ  /MOUNT PACK ON DRIVE  /
035605     101  042116  046040  MESG8:  .ASCIZ  /AND LOAD. /
035620  054524  042520  051040  MESG9:  .ASCIZ  /TYPE R<CR> WHEN DRIVE READY : /
035657     104  044522  042526  MESG10: .ASCIZ  /DRIVE IS NOT READY, TEST IS ABORTED/
035723     120  041501  020113  MESG11: .ASCIZ  /PACK IS NOT ACCEPTABLE,CHANGE PACK ,TRY AGAIN
036001     125  046116  040517  MESG12: .ASCIZ  /UNLOAD DRIVE /
036017      40  040440  042116  MESG13: .ASCII  /  AND REMOVE PACK/<CR><LF>
036042  054524  042520  051040          .ASCIZ  /TYPE R<CR> WHEN DONE : /
036072  025052  051440  040524  MESG14: .ASCIZ  /** STARTING PASS 2 **/
036120  025040  020052  046101  MESG15: .ASCIZ  / ** ALL DRIVES ARE COMPATIBLE **
036161     123  047503  042522  MESG16: .ASCIZ  /SCORES FOR DRIVE  /
```

```
036204  051124  041501  020113  MESG17: .ASCIZ  /TRACK      DRIVE      OVRWRT     OVRWRT     READ       READ/
036300  047516  020056  020040  MESG18: .ASCIZ  /NO.        READ       OFST-      OFST+      OFST-      OFST+/
036375     040  051440  046105  SELFX:  .ASCII  /  SELF/
036403     011     000          TAB:    .BYTE   11,0              ;THE SEQUENCE OF SELFX,TAB MUST BE NOT REVERSED.
036405     040  020052     060  MARKX:  .ASCII  / # 0/
036411     011     000          .BYTE   11,0
036413     054     000          COMMA:  .ASCIZ  /,/
036424     127  046111  020114  MESG5:  .ASCIZ  /WILL TEST /
036430  047117  000040          MESG6:  .ASCIZ  /ON /
036434  051511  052040  042510  MESG7:  .ASCIZ  /IS THERE ANOTHER SUB-SYSTEM (Y OR N<CR>) ?/
036507     116  052117  040440  NOTRM:  .ASCIZ  @NOT AN RM03/RM02@
036530  047040  052117  051440  NOTSAF: .ASCIZ  / NOT SAFE/
036542  047040  052117  050040  NOTPRS: .ASCIZ  / NOT PRESENT/
036557     040  043117  046106  UNTOFF: .ASCIZ  / OFFLINE/
036570  047440  046116  047111  UNTON:  .ASCIZ  / ONLINE/
036600  020040  020040          LIN4SP: .ASCII  /    /
036604  000040                  LINSP:  .ASCIZ  / /
036606  046522  031460     000  RM03A:  .ASCIZ  /RM03/
036613     015  020012  046047  NEDCLK: .ASCIZ  <CR><LF>/ 'L' OR 'P' CLOCK REQUIRED ON SYSTEM/<CR><LF>
036664  000072                  COLON:  .ASCIZ  /:/
036666  047105  042524  020122  ENTDRV: .ASCIZ  /ENTER I.D. FOR DRIVE #/
036715     077  044440  053116  BADENT: .ASCIZ  /? INVALID ENTRY/
036735     105  052116  051105  ENTADR: .ASCIZ  @ENTER BAD TRK/SEC ADDREES FOR DRV #@
037001     040  020057     000  SLASH:  .ASCIZ  @ / @
037005     077     000          QUES:   .ASCIZ  /?/
037007     104  044522  042526  UNTMSG: .ASCIZ  /DRIVE/
037015     040     000          LINSP0: .ASCIZ  / /
037017     104  044522  042526  STATUS: .ASCIZ  /DRIVE STATUS/
037034  020040  020040  040502  MESG19: .ASCIZ  /     BAD SPOT FILE  /
037060  025040  020052  042440  MESG20: .ASCIZ  / **  END  OF  TEST  **/
037107     104  044522  042526  MESG21: .ASCII  /DRIVE NOT ON-LINE OR NOT ASSIGNED/<CR><LF>
037152  051120  043517  040522  HALTX:          .ASCIZ  /PROGRAM HALT/<CR><LF>
037171     104  044522  042526  HALT1:  .ASCIZ  /DRIVE FAILS ON BASIC READ & WRITE TEST/
037240  051104  053111  020105  HALT2:  .ASCIZ  /DRIVE FAILS ON WRITE TEST (TEST 4)/
037303     106  052101  046101  XFATL:  .ASCIZ  /FATAL ERROR/
037320                          .EVEN
                                .LIST   BEX
7382
7383
7384                            ;HISTORY FILE FOR 15 LOGICAL DRIVES:(0-17)
7385  000001                            $FMT    =       1
7386  000002                    $COMND  =       $FMT+1          ;COMMAND CODE
7387  000003                    $PSEL   =       $FMT+2          ;PROT SELECT AND A16,A17
7388  000004                    $WRDM   =       $FMT+3          ;WORD COUNT
7389  000006                    $BUF    =       $FMT+5          ;BUFFER ADDRESS
7390  000010                    $SEC    =       $FMT+7          ;SECTOR ADDRESS
7391  000011                    $TRK    =       $FMT+10         ;TRACK ADDRESS
7392  000012                    $CYL    =       $FMT+11         ;CYLINDER ADDRESS
7393  000014                    $SYSNM  =       $FMT+13         ;SUB SYSTEM A-H
7394  000015                    $PHYDR  =       $FMT+14         ;PHYSICAL DRIVE CODE (ASCII)
7395  000016                    $GAP    =       $FMT+15         ;LEFT TWO NULL BYTES
7396  000022                    $EMTAB  =       $GAP+4  ;END OF HISTORY TABLE
7397
```

```
7398  037320  000      000          DRIV0:  .BYTE 7&0,0     ;HISTORY BLOCK OF LOGICAL DRIVE 0
7399  037322  000020                       .BLKB   $EMTAB-$COMND
7400  037342  001      000          DRIV1:  .BYTE 7&1,0     ;HISTORY BLOCK OF LOGICAL DRIVE 1
7401  037344  000020                       .BLKB   $EMTAB-$COMND
7402  037364  002      000          DRIV2:  .BYTE 7&2,0     ;HISTORY BLOCK OF LOGICAL DRIVE 2
7403  037366  000020                       .BLKB   $EMTAB-$COMND
7404  037406  003      000          DRIV3:  .BYTE 7&3,0     ;HISTORY BLOCK OF LOGICAL DRIVE 3
7405  037410  000020                       .BLKB   $EMTAB-$COMND
7406  037430  004      000          DRIV4:  .BYTE 7&4,0     ;HISTORY BLOCK OF LOGICAL DRIVE 4
7407  037432  000020                       .BLKB   $EMTAB-$COMND
7408  037452  005      000          DRIV5:  .BYTE 7&5,0     ;HISTORY BLOCK OF LOGICAL DRIVE 5
7409  037454  000020                       .BLKB   $EMTAB-$COMND
7410  037474  006      000          DRIV6:  .BYTE 7&6,0     ;HISTORY BLOCK OF LOGICAL DRIVE 6
7411  037476  000020                       .BLKB   $EMTAB-$COMND
7412  037516  007      000          DRIV7:  .BYTE 7&7,0     ;HISTORY BLOCK OF LOGICAL DRIVE 7
7413  037520  000020                       .BLKB   $EMTAB-$COMND
7414  037540  000      000          DRIV10: .BYTE 7&10,0    ;HISTORY BLOCK OF LOGICAL DRIVE 10
7415  037542  000020                       .BLKB   $EMTAB-$COMND
7416  037562  001      000          DRIV11: .BYTE 7&11,0    ;HISTORY BLOCK OF LOGICAL DRIVE 11
7417  037564  000020                       .BLKB   $EMTAB-$COMND
7418  037604  002      000          DRIV12: .BYTE 7&12,0    ;HISTORY BLOCK OF LOGICAL DRIVE 12
7419  037606  000020                       .BLKB   $EMTAB-$COMND
7420  037626  003      000          DRIV13: .BYTE 7&13,0    ;HISTORY BLOCK OF LOGICAL DRIVE 13
7421  037630  000020                       .BLKB   $EMTAB-$COMND
7422  037650  004      000          DRIV14: .BYTE 7&14,0    ;HISTORY BLOCK OF LOGICAL DRIVE 14
7423  037652  000020                       .BLKB   $EMTAB-$COMND
7424  037672  005      000          DRIV15: .BYTE 7&15,0    ;HISTORY BLOCK OF LOGICAL DRIVE 15
7425  037674  000020                       .BLKB   $EMTAB-$COMND
7426  037714  006      000          DRIV16: .BYTE 7&16,0    ;HISTORY BLOCK OF LOGICAL DRIVE 16
7427  037716  000020                       .BLKB   $EMTAB-$COMND
7428  037736  007      000          DRIV17: .BYTE 7&17,0    ;HISTORY BLOCK OF LOGICAL DRIVE 17
7429  037740  000020                       .BLKB   $EMTAB-$COMND
7430
7431
7432
7433  037760  000                  FMTDPB: .BYTE   0       ;DRIVER PARAMETER BLOCK, DRIVE #
7434  037761  000                          .BYTE   0       ;OFFSET FMT16,HCI,ECI
7435  037762  000                          .BYTE   0       ;COMMAND CODE
7436  037763  000                          .BYTE   0       ;PSEL, A16 AND A17
7437  037764  000000                       .WORD   0       ;WORD COUNT (NEG)
7438  037766  040074                       .WORD   #ENDPGM ;BUFFER ADDRESS
7439  037770  000                          .BYTE   0       ;SECTOR ADDRESS
7440  037771  000                          .BYTE   0       ;TRACK ADDRESS
7441  037772  000000                       .WORD   0       ;CYLINDER ADDRESS
7442  037774  040000                       .WORD   RM.REG  ;ADDRESS TO SAVE ALL RH11/RM03 REG'S
7443  037776  000000                       .WORD   0       ;STATUS WORD
7444
7445  040000  000000               RM.REG:  .WORD   0       ;RMCS1
7446  040002  000000                       .WORD   0       ;RMWC
7447  040004  000000                       .WORD   0       ;RMBA
7448  040006  000000                       .WORD   0       ;RMDA
7449  040010  000000                       .WORD   0       ;RMCS2
7450  040012  000000                       .WORD   0       ;RMDS
7451  040014  000000                       .WORD   0       ;RMER1
7452  040016  000000                       .WORD   0       ;RMAS
7453  040020  000000                       .WORD   0       ;RMLA
```

```
7454  040022  000000                          .WORD   0       ;RMDB
7455  040024  000000                          .WORD   0       ;RMMR1
7456  040026  000000                          .WORD   0       ;RMDT
7457  040030  000000                          .WORD   0       ;RMSN
7458  040032  000000                          .WORD   0       ;RMOF
7459  040034  000000                          .WORD   0       ;RMCA
7460  040036  000000                          .WORD   0       ;RMDC
7461  040040  000000                          .WORD   0       ;RMER2
7462  040042  000000                          .WORD   0       ;RMMR2
7463  040044  000000                          .WORD   0       ;RMEC1
7464  040046  000000                          .WORD   0       ;RMEC2
7465
7466
7467                                  ;INDEX OF STATUS AND REGISTER WORDS RELATIVE TO FMTDPB
7468
7469          000016                  $TATUS  =       16
7470          000020                  $RMCS1  =       20
7471          000022                  $RMWC   =       $RMCS1+2
7472          000024                  $RMBA   =       $RMWC+2
7473          000026                  $RMDA   =       $RMBA+2
7474          000030                  $RMCS2  =       $RMDA+2
7475          000032                  $RMDS   =       $RMCS2+2
7476          000034                  $RMER1  =       $RMDS+2
7477          000036                  $RMAS   =       $RMER1+2
7478          000040                  $RMLA   =       $RMAS+2
7479          000042                  $RMDB   =       $RMLA+2
7480          000044                  $RMMR1  =       $RMDB+2
7481          000046                  $RMDT   =       $RMMR1+2
7482          000050                  $RMSN   =       $RMDT+2
7483          000052                  $RMOF   =       $RMSN+2
7484          000054                  $RMDC   =       $RMOF+2
7485          000056                  $RMHR   =       $RMDC+2
7486          000060                  $RMMR2  =       $RMHR+2
7487          000062                  $RMER2  =       $RMMR2+2
7488          000064                  $RMEC1  =       $RMER2+2
7489          000066                  $RMEC2  =       $RMEC1+2
7490
7491
7492  040050  000000  000000  177776  GENDPB: .WORD   0,0,-2,CYLDER
7493  040056  040070
7494  040060  000000  000000  040000          .WORD   0,0,RM.REG,0
7495  040066  000000
7496  040070  000002                  CYLDER: .BLKW   2
7497          040074                  ENDPGM  =       .       ;LAST LOCATION OF PROGRAM+2
7498                                  .NLIST  BEX

      040074  005015  055103  046522  TITLE:  .ASCII  <CR><LF>/CZRMIB0/<CR><LF>
      040107     122  030115  027463          .ASCIZ  @RM03/RM02 DRIVE COMPATIBILITY TEST@<CR><LF><LF>
      040155     015  052012  020117  LOADRV: .ASCII  <CR><LF>/TO TEST DRIVE 0, REPLACE THE 'XXDP' PACK ON DRIVE 0/<CR><LF>
      040244  044527  044124  040440          .ASCII  /WITH ANOTHER PACK, CLEAR MEMORY LOCATION 40,/<CR><LF>
      040322  047101  020104  042522          .ASCIZ  /AND RESTART THE PROGRAM/<CR><LF>

                                  .LIST   BEX
7499
7500          000001                          .END
```

```
ABASE = 176700           1660*   1792    1833
ACDW1 = 000000           1792    1835
ACDW2 = 000000           1792    1836
ACK   = 000123           1679*
ACPUO = 000000           1792    1307
ACTDF  023444            5901*   6174*   6225*   6551*   6560*   6806
AC   R  023445           5907*   6808*   6822*
ADDW0 = 000000           1792
ADDW1 = 000000           1792
ADDW10= 000000           1792
ADDW11= 000000           1792
ADDW12= 000000           1792
ADDW13= 000000           1792
ADDW14= 000000           1792
ADDW15= 000000           1792
ADDW2 = 000000           1792
ADDW3 = 000000           1792
ADDW4 = 000000           1792
ADDW5 = 000000           1792
ADDW6 = 000000           1792
ADDW7 = 000000           1792
ADDW8 = 000000           1792
ADDW9 = 000000           1792
ADEVCT= 000000           1792    1798
ADEVM = 000000           1792    1834
AENV  = 000000           1792    1803
AENVM = 000000           1792    1804
AFATAL= 000000           1792    1795
AMADR1= 000000           1792    1820
AMADR2= 000000           1792    1824
AMADR3= 000000           1792    1827
AMADR4= 000000           1792    1830
AMAMS1= 000000           1792    1814
AMAMS2= 000000           1792    1822
AMAMS3= 000000           1792    1825
AMAMS4= 000000           1792    1828
AMSGAD= 000000           1792    1800
AMSGLG= 000000           1792    1801
AMSGTY= 000000           1792    1794
AMTYP1= 000000           1792    1815
AMTYP2= 000000           1792    1823
AMTYP3= 000000           1792    1826
AMTYP4= 000000           1792    1829
APASS = 000000           1792    1797
APRIOR= 000000           1792
APTCSU= 000040           5119    5225*
APTENV= 000001           5017    5112    5181    5223*
APTSIZ= 000200           2379    5222*
APTSPO= 000100           5114    5183    5224*
ASNLST  001776           1989*   2405    2470    2542    2544    2556*   2580    2586    3808
ASSGN1  002000           1991*   2557*   2618*   2637    3068*   3081
ASSGN2  002002           1993*   2558*   3097    3899*
ASWREG= 000000           1792    1805
ATABIT  023514           2491    5973*   6141    6237    6343    6725    6747    6755    6756    6776    6866
ATESTN= 000000           1792    1796
ATTN    001324           1854*   7353    7354
```

```
AUNIT = 000000          1792    1799
AUSWR = 000000          1792    1806
AVECT1= 000254          1661#   1792    1831
AVECT2= 000000          1792    1832
BADENT  036715          7381#
BADSEC  001340          1861#   3802*   3871    3938    3953*
BIT0  = 000001          1645#   3807    3864    3892
BIT00 = 000001          1635#   1645
BIT01 = 000002          1634#   1644    4035    6217    6472
BIT02 = 000004          1633#   1643
BIT03 = 000010          1632#   1642    4083    6683    6984
BIT04 = 000020          1631#   1641    4083    4095    6718
BIT05 = 000040          1630#   1640    6063    6493    6698
BIT06 = 000100          1629#   1639    4487    4818    6150    6208    6282    6591    7045
BIT07 = 000200          1628#   1638    4021    4098    6150    6417    6572    6683    6698    6718    6751    7017
BIT08 = 000400          1627#   1637    4092    5708    6150
BIT09 = 001000          1626#   1636    4033    5028    5716    6857
BIT1  = 000002          1644#   2774    2791
BIT10 = 002000          1625#   4031    4110    4118    5005    6474
BIT11 = 004000          1624#   4029    4113    5723    6110    6244    6438    6792
BIT12 = 010000          1623#   4027    4121    4972    6105    6132    6150    6219    6252    6432    6455    6470    6484
                        6708    6710    6922    6979    7046
BIT13 = 020000          1622#   4101    4127    4645    5012    6202    6654    6925
BIT14 = 040000          1621#   4008    4010    4035    4086    4089    4127    5694    6214    6249    6641    6787    6887
                        6934    6938    7043
BIT15 = 100000          1620#   4006    4129    4152    6202    6214    6217    6219    6249    6252    6438    6472    6474
                        6591    6683    6698    6708    6718    6787    6857    6887    6894
BIT2  = 000004          1643#   4037    6894
BIT3  = 000010          1642#   4104
BIT4  = 000020          1641#
BIT5  = 000040          1640#   4107
BIT6  = 000100          1639#
BIT7  = 000200          1638#   2726    2736    3384
BIT8  = 000400          1637#
BIT9  = 001000          1636#   4116
BLKADR  002600          2163#   2439    2554    2588    2641    2835    3072    3101    3904    3970
BPTVEC= 000014          1652#
BUSADR  031772          2515    7263#
CFLAG   001334          1859#   2415*   4477*   4480    4831*   4891*
CHGADR  001332          1858#   2319*   2321*   2323*   2499    2578    2615
CI1     025022          6272    6300#
CI3     025130          6274    6323#
CI4     025230          6263    6345#
CI5     025606          6322    6344    6422#   6431
CI6     025630          6361    6371    6373    6375    6427#
CI7     025644          6204    6281    6312    6316    6320    6328    6338    6342    6353    6360    6366    6370    6384
                        6390    6394    6404    6415    6430    6432#   6546    6576    6737    7029
CI7B    025666          6440#   6741
CIB     025744          6443    6454#   6879
CKBUS   014760          4013    4272#
CKCLK   015354          2431    4393#
CKCLK1  015450          4395    4406#
CKCLK2  015516          4407    4415#
CKCLK3  015542          4405    4414    4421#
CKERR   014712          4012    4258#
CKFMT   014500          4097    4173#
```

```
CKHCE   014510          4100    4179#
CKSWR = 104407          5001    5027    5693    5782#
CK.CHR  016606          4700#   4737    4751
CK.DEC  016560          4678#   4706
CK.DIG  016660          4731#
CK NUM  032164          7271    7280    7312#
CK.OCT  016532          2489    4660#   4708
CLKFLG  001314          1849#   4393*   4398*   4409*   4427
CLOCK   015650          4401    4411    4448#
CLRDPB  016042          2440    4497#
CLRQUE  031406          6045    6492    7113#
CMCNT   001352          1866#   2991*   3022*   3025*   3028
CMCYL   001354          1867#   2992*   2996    3021*   3024*   3026
CMSEC   001360          1869#   2993*   3018*   3467    3499    3542*   3548*   3550*
CMTRK   001362          1870#   3803*   3838    3844    3856*   3859    3869    3959
COLON   036664          2484    4434    4439    7381#
COMMA   036413          7381#
CR      = 000015        1560#   1839#   4917    5158    5168    7381    7498
CRLF    = 000200        1561#   5129    5168
CYLDER  040070          7492    7496#
CYLIMT  001372          1874#   2690
DCKER   014420          4126    4145#
DDISP = 177570          1567#   1769    2367
DEASG   013206          3891#
DF1     035470          2246    2281    7373#
DF2     035471          2253    2274    7374#
DF3     035477          2260    7376#
DF4     035502          2267    7377#
DH1     034722          2244    7351#
DH10    035170          2293    7351#
DH11    035241          2300    7351#
DH12    035312          2307    7351#
DH2     034731          2251    2272    7351#
DH3     035013          2258    7351#
DH4     035044          2265    7351#
DH6     035106          2279    7351#
DH7     035117          2286    7351#
DISMNT  013126          3872    3875#
DISPLA  001156          1769#   2367*   2375*   5004*   5738#
DISPLY= 104414          4044    4050    4056    4062    4068    4073    4146    4154    4156    4162    4168    4174    4180
                        4184    4188    4193    4198    4203    4209    4214    4218    4224    4230    4235    4241    4266
                        4280    4305    4312    4314    4350    4353    4366    4624    5787#
DISPRE  000174          1698#   2375
DONE    014062          4023    4083#
DPINT   023420          5861#   6074    6077    6158*   6184    6625    6650    6777    6779*   6848    6868    6873    6881*
DPRQS   023430          5874#   6190    6238*   6276*   6445*   6462    6478*   6628    6850    6870    6875    6890*
DRIVE = 001222          1853#   2478*   2491*   2510    2539    2559    2573*   2655*   2678    2686    2850*   2854    2879
                        2903    2972    3054    3148*   3166    3217    3365    3377    3434    3436    3582    3683    3880
                        3944    7354    7357    7359    7362
DRIV0   037320          2163    7398#
DRIV1   037342          2164    7400#
DRIV10  037540          2171    7414#
DRIV11  037562          2172    7416#
DRIV12  037604          2173    7418#
DRIV13  037626          2174    7420#
DRIV14  037650          2175    7422#
```

```
DRIV15  037672          2176    7424#
DRIV16  037714          2177    7426#
DRIV17  037736          2178    7428#
DRIV2   037364          2165    7402#
DRIV3   037406          2166    7404#
DRIV4   037430          2167    7406#
DRIV5   037452          2168    7408#
DRIV6   037474          2169    7410#
DRIV7   037516          2170    7412#
DRVACT  023370          3438#   5819#   6198    6378#   6425#   6447#   6459    6477#   6565#   6646    6681    6712#   6719#
                        6734    6746#   6860#
DRVCLR= 000111          1674#
DRVER   014470          4131    4167#
DRVINT  023756          6065    6099#   6186    6780
DRVQUE  031504          6193    6206    7153#
DRVSTA  023400          2679    2879    3167    5833#   6055#   6056#   6057#   6058#   6068#   6100#   6109#   6147#   6152#
                        6180    6188    6212    6246    6250    6379#   6496#   6631    6639    6652    6706#   6713#   6726
                        6782    6882#
DRVTYP  023410          2681    2881    3169    5848#   6101#   6116#   6121#   6126#   6215    6497#
DSWR    = 777570        1566#   1768    2366
DTEER   014540          4123    4192#
DTUW    023512          3434#   5967#   6051    6265    6321#   6452#   6465    6479    6481#   6490#   6554    6566#   6853
                        6863#   6932
DT1     035344          2245    7353#
DT10    035430          2294    7364#
DT11    035444          2301    7366#
DT12    035460          2308    7368#
DT2     035350          2252    2273    7354#
DT3     035366          2259    7357#
DT4     035376          2266    7359#
DT6     035410          2280    7361#
DT7     035414          2287    7362#
DULP    023760          6100#   6111
DUMP    014660          4147    4155    4157    4163    4169    4175    4181    4185    4189    4194    4199    4204    4210
                        4215    4219    4225    4231    4236    4242    4245#   4267    4281
DUMP2   014034          4045    4051    4057    4063    4069    4074#
EMPTYQ  031464          6444    6495    6592    6895    7138#
EMTVEC= 000030          1655#   2336#   2337#   2383#
EM1     032274          2243    7351#
EM10    032576          4050    7351#
EM11    032641          4056    7351#
EM12    032674          4044    7351#
EM13    032725          4062    7351#
EM14    032777          4068    7351#
EM15    033022          4073    7351#
EM2     032343          2250    7351#
EM20    033056          4162    7351#
EM21    033077          4146    7351#
EM22    033130          4156    7351#
EM23    033203          4154    7351#
EM24    033262          4174    7351#
EM25    033330          4180    7351#
EM26    033411          4214    7351#
EM27    033436          4218    7351#
EM3     032401          2257    7351#
EM30    033473          4168    7351#
```

```
EM31    033526  4188    7351#
EM32    033570  4193    7351#
EM33    033623  4198    7351#
EM34    033700  4235    7351#
EM35    033742  4203    7351#
EM36    034000  4209    7351#
EM37    034031  7351#
EM4     032437  2264    7351#
EM40    034113  4224    7351#
EM41    034155  4280    7351#
EM42    034221  7351#
EM43    034302  7351#
EM44    034347  4266    7351#
EM45    034433  7351#
EM46    034521  7351#
EM5     032474  2271    7351#
EM50    034573  4230    7351#
EM51    034631  4184    7351#
EM6     032530  2278    7351#
EM60    034674  4241    7351#
EM61    034717  7351#
ENDPGM= 040074  2687    2716    2719    2907    2974    3222    3269    3303    3586    3681    7438    7497#
ENDX1   005464  2639    2706    2801#
ENTADR  036735  7381#
ENTDRV  036666  7381#
ERPRC1  013676  4022    4027#
ERPROC  013662  4005    4009    4011    4021#
ERRVEC= 000004  1648#   2364    2365#   2376#   4395#   4396#   4407#   4421#   5699    5700#   5702#   5705#   7231
                7232    7236#   7237#   7244#   7245#   7284    7285#   7287#   7294#
ES.SAV  031650  6610    6662    6666    6930    6990    7208#
FALPAR  014000  4032    4055#
FAULT   001374  1875#   3108#   3954#   4923
FILBUF  015162  2911    3588    4319#
FINISH  017604  4923#   4983
FMTOPB  037760  2685    2696    2698    2701#   2702    2728    2738    2901    2913    2915    2917    2923    2925
                2927    2971    2994    3000    3002    3004    3215    3237    3239    3241    3264    3272    3274
                3276    3300    3308    3311    3363    3368    3370    3375    3390    3392    3397#   3432#   3433#
                3435    3482    3535    3578    3590    3592    3594    3619    3621    3623    3631    3633    3677
                3688    3690    3692    3710    3716    3718    7433#
FMTER   014600  4213#
GENOPB  040050  7492#
GETREG= 000141  1681#
GETREQ  031560  6239    6467    6583    6635    6672    6736    6786    6856    6884    6891    7174#
GNS     = ******  L     1697    5774    5775    5776    5777    5778    5780    5782    5783    5784    5795    5786    5787
G.SWR   = 104406        2399    5780#
HALTX   037152  2938    3034    4077    4139    4834    7381#
HALT1   037171  2937    7381#
HALT2   037240  3033    7381#
HCEER   014610  4218#
HCRCER  014460  4094    4161#
HOUR    001342  1862#   2412#   4430    4459#   4460    4462#
HT      = 000011  1558#   2572    5127    5168
HZ      001316  1850#   2411    4450
IAEER   014560  4112    4202#
IDLEX   013560  2643    2648    2841    3049    3128    3876    3998#
INDST   002560  2150#   2407    2471    3492
```

```
IOTVEC= 000020          1653#   2334#   2335#
ISR     026356          6060    6551#
KIPAR0= ****** U        7227
KPATH   012640          3813#
KSR     015756          4475#
LA      026202          6269    6512#
LABAD   011144          3278    3313    3532#   3694    3720
LACNT   023456          5926#   6520#   6521    6542#
LF    = 000012          1559#   1840#   4917    5162    5168    7381    7498
LINDEC  015334          4378#
LINE1   015236          4345#
LINKDV  016106          4294    4514#
LINOCT  015302          4307    4362#
LINSP   036604          2482    2600    2660    2664    2849    2858    3058    3153    3973    3974    4305    7268    7277
                        7381#
LINSP0  037015          4353    7381#
LIN4SP  036600          7381#
LOADRV  040155          2390    7498#
LOGO    001376          1892#   2963
LOG1    001416          1898#
LOG10   001636          1952#
LOG11   001656          1958#
LOG12   001676          1964#
LOG13   001716          1970#
LOG14   001736          1976#
LOG15   001756          1982#
LOG2    001436          1904#
LOG3    001456          1910#
LOG4    001476          1916#
LOG5    001516          1922#
LOG6    001536          1928#
LOG7    001556          1934#
LOG8    001576          1940#
LOG9    001616          1946#
LOOP1   007740          3264#   3341
LOOP2   007756          3268#   3332    3337
LOOP3   012026          3677#   3750
LOOP4   012620          3808#   3855    3870
LOOP5   013032          3809    3826    3829    3847    3850#
LOOP6   013050          3856#
LSTAD   001330          1857#   2422#   2423    2425#   2426#
MAKEUP  010606          3307    3425#   3630    3715
MARKX   036405          3987    7381#
MCPEMX  023524          5985#   6912    6960
MESG1   035506          2480    2602    2847    7381#
MESG10  035657          2705    2884    3173    7381#
MESG11  035723          2734    4134    7381#
MESG12  036001          3052    3878    7381#
MESG13  036017          3059    3884    7381#
MESG14  036072          3144    7381#
MESG15  036120          4927    7381#
MESG16  036161          3942    7381#
MESG17  036204          3949    7381#
MESG18  036300          3951    7381#
MESG19  037034          4286    7381#
MESG2   035517          2483    2583    7381#
```

```
MESG20  037060          4930    7381#
MESG21  037107          2802    7381#
MESG3   035531          2651    2844    7381#
MESG4   035557          2653    2852    3146    7381#
MESG5   036415          2582    7381#
MESG6   036430          2601    2846    7381#
MESG7   036434          2609    7381#
MESG8   035605          2661    2859    3154    7381#
MESG9   035620          2663    2861    3156    7381#
MINUTE  00134'          1863#   2413#   4435    4455#   4456    4458#
MMDLTA  023540          6001#   6540
MOD00   003720          2470#
MOD11   004212          2518    2524    2539#
MOD12   004374          2541    2575    2578#
MOD21   003774          2478#   2490    2497    2614
MOD22   004174          2500    2521#
MOD23   004130          2508    2510#
MOD30   004224          2540    2542#   2574
MONTR   003720          2436#
MRHVEC  032153          7274    7301#
MRMCS1  032142          7265    7299#
MXDLTA  023536          5998#   6538
MXLACT  023534          5995#   6521
MXWNDW  023542          6004#
M.DPID  016154          4521    4535#
M.DP40  016212          4546#   4567
M.DP41  016246          4551    4558#
M.DP42  016256          4557    4562#
M.DP44  016310          4569    4574#
M.DP50  016322          4544    4578#
NEDCLK  036613          4416    7381#
NOTPRS  036542          7381#
NOTRM   036507          7381#
NOTSAF  036530          7381#
NULINE  001364          1871#   3810#   3955    3978#   3981    3985#   3988#
OFFCOD  002640          2182#   3263#   3338    3340#   3385    3471    3477    3617#   3638    3641#   3647#   3654#   3676#
                        3747    3749#
OFFSET= 000115          1676#
OFFST   010354          3305    3359#   3628    3713
OFLIN   014020          4036    4067#
OPIER   014530          4103    4187#
OPT     024564          6200    6235#   6586    6760    6798
OVWN0   002046          2004#   3210    3481    3806    3857
OVWN1   002066          2010#
OVWN2   002106          2016#
OVWN3   002126          2022#
OVWN4   002146          2028#
OVWP0   002166          2038#   3479
OVWP1   002206          2045#
OVWP2   002226          2052#
OVWP3   002246          2059#
OVWP4   002266          2066#
PACK    001322          1852#   2969#   2995
PAPER   014550          4106    4197#
PCLOCK  001312          1848#   4394#   4399#
PIRQ  = 177772          1565#
```

```
PIRQVE= 000240  1659#
POPQUE  031602  6248    6300    6416    6721    6750    6790    7189#
POSER   014520  4184#
PRINT   013270  3815    3819    3823    3827    3832    3836    3842    3848    3933#
PROCES  013566  2918    2928    3005    3242    3277    3312    3595    3624    3634    3693    3719    4003#
PRTBAD  015022  2729    2739    4285#
PRTIM   014030  4038    4072#
PR0     = 000000  1582#
PR1     = 000040  1583#
PR2     = 000100  1584#
PR3     = 000140  1585#
PR4     = 000200  1586#  4476
PR5     = 000240  1587#  4816
PR6     = 000300  1588#
PR7     = 000340  1589#
PS      = 177776  1562#  1563    2410#   3373    3374#   3383#   4476#   4488#   4632    4633#   4843    6043    6044#
                  6073#  6081#  6172    6173#   6226#   6236    6285#   6527#
PSEUDO  002702  2204#   2979    3226
PSW     = 177776  1563#
PUNSAF  013760  4028    4043#
PWRVEC= 000024  1654#   2340#   2341#   5231#   5232#   5241#   5247#   5259#   5260#
QCNT    031114  7055#   7114    7138#   7153    7155#   7175    7189#
QDRV0   031206  7066    7077    7086    7098#
QDRV1   031226  7067    7078    7087    7099#
QDRV2   031246  7068    7079    7088    7100#
QDRV3   031266  7069    7080    7089    7101#
QDRV4   031306  7070    7081    7090    7102#
QDRV5   031326  7071    7082    7091    7103#
QDRV6   031346  7072    7083    7092    7104#
QDRV7   031366  7073    7084    7093    7105#
QINPT   031124  7066#   7140    7157#   7158#   7159    7161#
QOUTPT  031144  7077#   7140#   7178    7191    7192#   7193#   7194    7196#
QSTART  031164  7086#   7120    7125    7161    7196
QSTOP   031166  7087#   7159    7194
QTERM = 031406  7094    7106#
QUES    037005  7381#
RDCHR = 104410  4825    4865    5783#
RDDAT = 000171  1688#   2693    3431    3680
RDHD  = 000173  1689#
RDLIN = 104411  2485    2610    2669    2862    3075    3157    3885    5784#   7269    7278
RDN0    002306  2075#   3211    3475    3672    3781
RDN1    002326  2081#
RDN2    002346  2087#
RDN3    002366  2093#
RDN4    002406  2099#
RDP0    002426  2108#   3473    3779
RDP1    002446  2114#
RDP2    002466  2120#
RDP3    002506  2126#
RDP4    002526  2132#   3674
RD.ADR  030376  6914#   6915#   6917#   7357
RD.RM   030352  6112    6136    6142    6364    6388    6402    6514    6528    6574    6604    6675    6685    6700
                6842    6912#   6967    6981    6987    7021
RD.RM1  030374  6916#   6946
RD.RM2  030520  6912#   6945#
RD.RM3  030524  6923    6937    6947#
```

```
RD.RM4  030530  6928  6949#
RD.WRD  030400  6918#  6919  7357  7359
READIN=  000121  1678#
RECAL  =  000107  1673#
RELSE  =  000113  1675#
REPLZ  016332  4432  4437  4442  4589#
RESREG=  104413  4333  4484  4524  5614  5664  5786#  6082  6220  6223  6286  6499  6559  6821
                 7030  7129  7296
RESVEC=  000010  1649#  2351*  2358*
RMADR  023526  2403#  2645*  2873*  3103*  3139*  3376  3437  5990#  6062  6179  6303  6323  6345
                 6512  6553  6840  6914  6920  6924  6940  6973  6977  7010  7040  7283  7361
RMAS  =  000016  6015#  6141*  6602  6725*  6755*  6776*  6865
RMBA  =  000004  3382*  6010#
RMCS1  =  000000  3442*  6008#  6104*  6110  6130  6243*  6244  6319  6341  6383  6403  6414  6429
                 6442  6575  6605  6659  6692  6731  6792  6968
RMCS2  =  000010  3378*  6012#  6063*  6103*  6105  6192*  6242*  6304*  6324*  6346*  6432  6455  6470
                 6484  6493*  6513*  6573*  6673*  6762*  6775*  6841*  6921  6978  7011*  7017  7041*
                 7046
RMDA  =  000006  3380*  6011#  6311  6337  6352
RMDB  =  000022  6017#  7015
RMDC  =  000034  3379*  6022#  6315  6327  6359
RMDS  =  000012  6013#  6137  6515  6676  6701  6763  6843
RMDT  =  000026  6019#  6113
RMEC1  =  000044  6026#
RMEC2  =  000046  6027#  7025
RMERRS  023360  5811#  6046  6641  6644  6654  6763*  6764*  6765*  6766*  7354
RMER1  =  000014  6014#  6143  6686  6764  6982
RMER2  =  000042  6025#  6765
RMMR  =  000036  6023#
RMINIT  023544  2675  2875  3163  3245  3281  3316  3569  3598  3697  3723  4014  4250  6042#
RMLA  =  000020  6016#  6529  6531
RMMR1  =  000024  6018#
RMMR2  =  000040  6024#  6766
RMOF  =  000032  6021#  6134  6365  6369  6389  6393
RMSN  =  000030  6020#
RMVEC  023530  2404#  2643*  2644*  2646*  2648*  2649*  2841*  2842*  2874*  3049*  3050*  3104*  3128*
                 3129*  3140*  3876*  3877*  5991#  6059  6061  6173
RMWC  =  000002  3381*  6009#
RM.REG  040000  2688  2908  2975  3223  3270  3304  3587  3682  7362  7364  7366  7368  7442
                 7445#  7494
RM03  024272  2695  2912  2922  2999  3236  3271  3367  3389  3589  3618  3687  6172#
RM03A  036606  7381#
RNOP  =  000101  1671#
RPTMR  027740  4464  6806#
RTC  =  000117  1677#
SAVEFG  023466  2432*  2676*  2876*  3164*  3246*  3282*  3317*  3570*  3599*  3698*  3724*  4015*  4251*
                 5941#  6418  6579  6752
SAVREG=  104412  4319  4475  4514  5588  5649  5785#  6042  6175  6235  6454  6552  6809  7007
                 7113  7263
SC  026572  6558  6587  6590  6596  6602#
SCORE  010724  3288  3323  3462#  3704  3730
SC11  027430  6643  6674  6682  6743#
SC12  027540  6637  6648  6762#
SC13  027602  6627  6630  6772#
SC3  026642  6618#  6622
SC4  026646  6620#  6651  6664  6668  6717  6733  6742  6761  6799
```

```
SCS     026660          6619  6625#
SC6     027060          6647  6669#
SC6A    027174          6645  6694#
SC7     027322          6684  6697  6699  6709  6719#
SC8     027400          6660  6677  6687  6693  6702  6732  6734#  6771
SEARCH= 000131          1680#
SECLMT  001366          1872#
SECOND  001346          1864#  2414#  4440  4451#  4452  4454#
SEEK  = 000105          1672#
SEEKFG  023470          2677#  2877#  3165#  3247#  3283#  3318#  3571#  3600#  3699#  3725#  4016#  4252#  5949#
                        6047  6267
SELDRV= 000145          1683#
SELFX   036375          3966  7381#
SELF0   002546          2141#  3574  3576  3612  3650  3778  3780  3840  3846
SELF1   002550          2142#
SELF2   002552          2143#
SELF3   002554          2144#
SELF4   002556          2145#
SETFMT= 000143          1682#
SETVEC  003706          2431#
SET.IE  031042          6076  6107  6210  6284  6498  6612  7039#
SIXTEE  001350          1865#  2411#  4448#  4450#
SIZMEM  003642          2419#
SKIER   014630          4132  4229#
SLASH   037001          7381#
SPATH   012730          3812  3830#
SPOTX   005316          2721  2756#
SRCHWT  023442          5895#  6237#  6343#  6747  6756#
STACK = 001100          1553#  2332  2409  2834  2900  2962  3048  3127  3209  3573  3671  3773
STARSC  001356          1868#  3467  3491  3538#
START   003062          1701  2319#  4420
START1  003072          1702  2321#
START2  003102          1706  2323#
START3  003106          2320  2322  2325#
STATIN  001320          1851#
STATUS  037017          7381#
STKLMT= 177774          1564#
STNDAT  002642          2188#  4323  4331
ST0     030030          6816  6836#
ST01    030064          6848#
ST02    030160          6847  6849  6851  6865#
ST03    030210          6867  6873#
ST05    030234          6879#
ST06    030242          6869  6881#
ST07    030300          6871  6883#
ST08    030330          6888  6895#
ST09    030340          6845  6855  6864  6872  6876  6878  6880  6886  6893  6897#
SWRH70  030724          6420  6581  6593  6636  6689  6707  6754  6788  6858  6896  7007#
SWR     001154          1768#  2330  2366#  2368  2374#  2381#  2397  2726  2736  4345  4348  4645  4837
                        4972  5005  5012  5024  5028  5239  5252#  5429  5466#  5694  5708  5710  5716
                        5723
SWREG   000176          1699#  2374  2397  4837  5429  5442
SWTIM   014010          4034  4061#
SW0   = 000001          1617#
SW00  = 000001          1607#  1617
SW01  = 000002          1606#  1616
```

```
SW02  = 000004          1605#   1615
SW03  = 000010          1604#   1614
SW04  = 000020          1603#   1613
SW05  = 000040          1602#   1612
SW06  = 000100          1601#   1611
SW07  = 000200          1600#   1610
SW08  = 000400          1599#   1609
SW09  = 001000          1598#   1608
SW1   = 000002          1616#
SW10  = 002000          1597#   4345
SW11  = 004000          1596#
SW12  = 010000          1595#
SW13  = 020000          1594#   4348
SW14  = 040000          1593#
SW15  = 100000          1592#
SW2   = 000004          1615#
SW3   = 000010          1614#
SW4   = 000020          1613#
SW5   = 000040          1612#
SW6   = 000100          1611#
SW7   = 000200          1610#
SW8   = 000400          1609#
SW9   = 001000          1608#
SYSADR  002004          1995#   2506    2522#   2523#   2645    2646    2873    2874    3103    3104    3139    3140
TAB     036403          3963    3967    3977    3984    7381#
TABLEX  002044          1997#   3488*   3507
TAB.XY= 001114          1740#   1747
TBITVE= 000014          1650#   2348*   2349*
TD      026422          6556    6565#
TIMER   023472          3440*   5954#   6156*   6279*   6423*   6476*   6568*   6670*   6715*   6758*   6773*   6795*   6812
                        6814*   6877*   6883*   6889*
TITLE   040074          2387    7498#
TKVEC = 000060          1657#   4815*   4816*
TPVEC = 000064          1658#
TRAPVE= 000034          1656#   2338*   2339*   2384*
TRFER   014620          4091    4223#
TRKLMT  001370          1873#   2691
TRNSWT  023440          3435*   5887#   6301*   6448    6451*   6464    6482*   6491*   6570    6571*   6852    6862*
TRTVEC= 000014          1651#
TSTNM   001336          1860#   2833*   2899*   2961*   3047*   3126*   3208*   3429    3469    3572*   3670*   3772*
TST1    005502          2741    2831#   2885    3084
TST10   011764          3668#
TST11   012452          3770#
TST2    006014          2883    2897#
TST3    006266          2935    2959#
TST4    006710          3031    3045#
TST5    007172          3109    3124#   3174    3905
TST6    007454          3171    3206#
TST7    011250          3342    3567#
TYPDS = 104405          5778#
TYPE  = 104401          2387    2390    2479    2480    2481    2482    2483    2484    2582    2583    2600    2601    2602
                        2603    2608    2609    2650    2651    2652    2653    2654    2660    2661    2662    2663    2664
                        2705    2734    2735    2801    2802    2843    2844    2845    2846    2847    2848    2849    2851
                        2852    2853    2858    2859    2860    2861    2884    2936    2937    2938    3032    3033    3034
                        3051    3052    3053    3058    3059    3106    3143    3144    3145    3146    3147    3153    3154
                        3155    3156    3172    3173    3873    3875    3878    3879    3884    3941    3942    3943    3948
```

```
                        3949    3950    3951    3952    3957    3958    3963    3966    3967    3973    3974    3975    3977
                        3984    3987    4074    4075    4076    4077    4133    4134    4139    4285    4286    4287    4347
                        4416    4429    4434    4439    4604    4829    4830    4833    4834    4841    4842    4849    4872
                        4878    4883    4887    4892    4897    4898    4900    4903    4907    4925    4926    4927    4928
                        4929    4930    4931    5007    5015    5049    5066    5068    5071    5073    5077    5084    5132
                        5261    5333    5409    5440    5441    5444    5457    5468    5487    5774#   7265    7268    7274
                        7277
TYPOC  = 104402         5057    5081    5443    5775#   7267    7276
TYPON  = 104404         5777#
TYPOS  = 104403         2592    2657    2855    3055    3150    3881    3945    3960    5776#
TYPRI4   016462         4632#
UCPAR    013770         4030    4049#
ULDFLG   023446         5914#   6102#   6182    6197#   6380#   6446#   6483#   6633    6722    6724#   6743    6745#   6861#
UNIT     001326         1855#   4003#
UNSAF    014650         4085    4240#
UNTMSG   037007         7381#
UNTOFF   036557         7381#
UNTON    036570         7381#
WCFER    014640         4109    4234#
WCKD   = 000151         1684#   2921    3268    3428    3611
WCKER    014430         4088    4151#
WCKHD  = 000153         1685#
WLEER    014570         4115    4208#
WRTDAT= 000161          1686#   2910    2931    2973    3220    3580
WRTHD  = 000163         1687#
WRT.AC   030622         6963*   6973*   6976#   7359
WRT.RM   030532         6129    6133    6310    6314    6318    6326    6336    6340    6351    6358    6368    6382    6392
                        6413    6428    6441    6658    6691    6730    6960#
WRT.R1   030616         6974#   6995
WRT.R2   030710         6960*   6994#
WRT.R3   030714         6969    6980    6983    6989    6996#
WRT.R4   030720         6985    6998#
WRT.R5   030722         6997    6999#
WRT.WD   030620         6961*   6965    6972*   6975#   7359
XEND2    013264         3099    3900    3908#
XFATL    037303         4075    7381#
XPASS1   004600         2617    2637#   2740
XPASS2   007106         2619    3083    3097#
$APTHD   001100         1727    1733#
$ASTAT= ****** U        5203    5218
$ATYC    020660         5174    5176#
$ATY1    020634         5172#
$ATY3    020642         5117    5173#
$ATY4    020652         5020    5 5#
$AUTOB   001150         1765#   2401#   5437    5527
$BASE    001264         1833#
$BDADR   001136         1760#
$BDDAT   001142         1762#
$BELL    001200         1780#   4347    5007    5040
$BUF   = 000006         2687#   2907#   2974#   2980    3222#   3228    3269#   3303#   3382    3586#   3681#   4277    4320
                        7389#
$CDW1    001270         1835#   2507#   2512    2641#   2647    2835    2837    2902    3072#   3101#   3105    3130    3216
                        3904#
$CDW2    001272         1836#   2475#   2481    2501    2555    2589    2603    2605#   2606    2642#   2654    2838    2840#
                        2848    2853    2868    3053    3102#   3131    3133#   3134    3147    3879    3943
$CHARC   020630         5134#   5144#   5151    5160#   5165#
```

```
$CKSWR  021744          5429#   5782
$CLR.T  017732          4958    4961#
$CMTAG  001114          1748#   2327    2328    2336    2342    2343    2344
$CM3    = 000000        1778#
$CNTLC  017576          4830    4892    4917#
$CNTLG  022352          4842    5440    5532#
$CNTLU  022346          4887    5457    5521#
$COMND= 000000          2693#   2910#   2921#   2931#   2973#   3220#   3268#   3354    3366#   3388#   3397#   3433#   3580#
                        3611#   3690#   7386#   7399    7401    7403    7405    7407    7409    7411    7413    7415    7417
                        7419    7421    7423    7425    7427    7429
$CPJOP  001236          1807#
$CRLF   001205          1782#   2479    2608    2650    2652    2662    2735    2901    2843    2845    2851    2860    2936
                        3023    3051    3106    3143    3145    3155    3172    3873    3875    3941    3948    3950    3952
                        3957    3958    4074    4076    4133    4285    4287    4312    4314    4350    4429    4829    4833
                        4841    4897    4925    4926    4928    4929    4931    5015    5040    5049    5058    5073    5077
                        5133    5168    5468
$CYL    = 000012        2690#   2905#   2996#   3016#   3020#   3021    3221#   3252#   3253    3266#   3329#   3330    3333#
                        3373    3585#   3605#   3606    3616#   3651#   3652    3686#   3736#   3737    3742#   7392#
$DBLK   021734          5375    5409    5417#
$DB2D   022476          4788    5588#
$DB20   022672          4805    5649#
$DECVL  022656          5590    5636#
$DEVCT  001220          1798#
$DEVM   001266          1834#   2640#   2904    2964    2977    3062    3074#   3100#   3221    3224    3266    3333    3490
                        3583    3614    3684    3740    3811    3893    3901#   3902    3964
$DOAGN  017752          4951    4956    4963    4969#
$DSPLY  016510          4645#   5787
$DTBL   021724          5378    5413#
$EMTAB= 000022          4504    7396#   7399    7401    7403    7405    7407    7409    7411    7413    7415    7417    7419
                        7421    7423    7425    7427    7429
$ENDAD  017742          1713    4965#   5035
$ENDCT  017710          2342    4953#
$ENV    001230          1803#   2395    5017    5112    5181    5205
$ENVM   001231          1804#   2379    5114    5119    5183
$EOP    017654          3908    4943#
$EOPCT  017702          2342#   4950#   4954
$ERFLG  001117          1751#   5002#   5040    5683    5712    5714    5720#   5742
$ERMAX  001131          1757#   2345#   5714    5737#   5742
$ERROR  020016          2336    5000#
$ERRPC  001132          1758#   5009#   5010#   5011    5040    5055
$ERRTB  002742          2238#   5063
$ERRTY  020216          5014    5048#
$ERTTL  001126          1755#   5008#   5040
$ESCAP  001176          1779#   2344#   5031    5033    5040    5736#   6610    6662    6666    6930    6990    7200    7209
                        7210#   7212#
$ETABL  001230          1802#
$ETEND  001274          1739    1837#
$FATAL  001212          1795#   5209#
$FFLG   021100          5172#   5175#   5203    5212#   5220#
$FILLC  001172          1776#   5137    5168
$FILLS  001171          1775#   5168
$FMT    = 000001        3384#   3387#   7385#   7386    7387    7388    7389    7390    7391    7392    7393    7394    7395
$GAP    = 000016        2572#   7395#   7396
$GDADR  001134          1759#
$GDDAT  001140          1761#
$GET42  017714          4955#
```

```
$GTSWR  022014          4846    5441#   5780
$HO   = 000000          1537
$HIBTS  001100          1734#
$ICNT   001120          1752#   5727*   5728    5730*   5741
$ILLUP  021254          5231    5247    5266#
$INTAG  001151          1766#   5469    5527
$ITEMB  001130          1756#   5011*   5019    5040    5052
$LF     001206          1783#   4907    5040    5168
$LFLG   021077          5213*   5219#
$LKCSB  001302          1844#   4403*
$LKCSR  001300          1843#   4397    4404*
$LKS    001306          1846#   4408    4413*
$LLVEC  001310          1847#   4410
$LOOP   020010          4977    4981#
$LPADR  001122          1753#   2360*   5718*   5734*   5739    5741
$LPERR  001124          1754#   2361*   5030    5718    5735*   5741
$LPVEC  001304          1845#   4400
$LSTAD  031770          [illegible]2    7246*   7250#
$MADR1  001242          1820#
$MADR2  001246          1824#
$MADR3  001252          1827#
$MADR4  001256          1830#
$MAIL   001210          1735    1739    1793#   2378    2395    5017    5112    5733
$MAMS1  001240          1814#
$MAMS2  001244          1822#
$MAMS3  001250          1825#
$MAMS4  001254          1828#
$MBADP  001102          1735#
$MFLG   021076          5173*   5179    5214*   5218#
$MNEW   022371          5444    5525#
$MSGAD  001224          1800#   5189*   5192
$MSGLG  001226          1801#   5194*
$MSGTY  001210          1794#   5187    5195*   5207    5211*
$MSWR   022360          5441    5523#
$MTYP1  001241          1815#
$MTYP2  001245          1823#
$MTYP3  001251          1826#
$MTYP4  001255          1829#
$MXCNT  023270          5731    5741#
$NULL   001170          1774#   5139    5168
$NWTST= 000000          2830#   2896#   2958#   3044#   3123#   3205#   3566#   3667#   3769#
$OCNT   021514          5305*   5334*   5347#
$OCTVL  022774          5651    5676#
$OMODE  021516          5300*   5304*   5309    5312*   5323*   5349#
$OVER   023254          5695    5711    5719    5729    5738#
$PASS   001216          1797#   2378*   4947*   4948*   4984    5725    5742
$PASTM  001106          1737#
$PHYDR= 000015          2571*   7394#
$POWER  021262          5262    5269#
$PSEL   = 000003        7387#
$PWRDN  021102          2340    5231#   5259
$PWRMG  021236          5262#
$PWRUP  021154          5241    5247#
$QUES   001204          1781#   4900    5040    5168    5487
$PDCHR  022226          5500#   5783
$PDDEC= ******          5785
```

```
$RDLIN  017274          4860#   5784
$RDOCT= ****** U        5785
$RDSZ = 000001          5521#
$RESRE  022440          5562#   5786
$RMAOR  001274          1841#   2403    2513#   2516    2522    7264    7282    7286
$RMAS = 000036          7477#   7478
$RMBA = 000024          4278    4288    4303    7472#   7473
$RMCS1= 000020          4006    4008    4258    7470#   7471
$RMCS2= 000030          4086    4260    7474#   7475
$RMDA = 000026          3291    3326    3536    3707    3733    7473#   7474
$RMDB = 000042          7479#   7480
$RMDC = 000054          7484#   7485
$RMDS = 000032          4010    4089    4118    7475#   7476
$RMDT = 000046          7481#   7482
$RMEC1= 000064          7488#   7489
$RMEC2= 000066          7489#
$RMER1= 000034          4092    4095    4098    4101    4104    4107    4110    4113    4116    4121    4124    4152    4262
                        7476#   7477
$RMER2= 000062          4127    4129    4264    7487#   7488
$RMHR = 000056          7485#   7486
$RMLA = 000040          7478#   7479
$RMMR1= 000044          7480#   7481
$RMMR2= 000060          7486#   7487
$RMOF = 000052          7483#   7484
$RMSN = 000050          7482#   7483
$RMVEC  001276          1842#   2404    2514#   2517    2523    7273
$RMWC = 000022          3289    3292    3324    3327    3544    3546    3705    3708    3731    3734    4272    4291    7471#
                        7472
$RTNAD  020012          4983#
$RTRN   020006          2348    2350#   2355#   4959    4978#
$R2A  = ****** U        5787
$SAVRE  022402          5546#   5785
$SAVR6  021260          5240#   5248    5249#   5250#   5268#
$SB2D   017016          4379    4431    4436    4441    4786#
$SB20   017046          4363    4803#
$SCOPE  023012          2334    5692#
$SEC  = 000010          2692#   2730    2732#   2904#   2998#   3008#   3009    3011#   3219#   3265#   3291#   3302#   3326#
                        3380    3538    3548    3581#   3604#   3649#   3679#   3707#   3712#   3733#   7390#
$SETUP= 000137          2311#   2333    2334    2336    2338    2340    2342    2343    2344    2346    2360    2392    4945
                        5001    5027    5035    5264    5424    5527    5693
$SIZE   031674          2421    7229#
$STUP = 177777          2311#
$SUPRS  016422          4380    4616#
$SVLAD  023220          5703    5732#
$SVPC = 000224          1711#   1716
$SWP  = 177600          1526#   1537    1541    1542    1543    1544    1545    1546    1547    1548    1778    1779    1780
                        2343    2344    2346    2360    2361    2832    2898    2960    3046    3125    3207    3568    3669
                        3771    4939    4946    4957    4970    4984    4992    4993    4994    4995    4996    5005    5012
                        5024    5028    5040    5263    5684    5685    5686    5687    5688    5694    5706    5708    5709
                        5712    5713    5714    5721    5722    5723    5735    5738    5741
$SWREG  001232          1805#   2381
$SWRMK= 000000          1548    1549    5688    5689    5710
$SYSNM= 000014          2555#   2589    2838    2840    3131    3133    3972    7393#
$TATUS= 000016          4004    4021    4027    4029    4031    4033    4035    4037    4083    7469#
$TBIT   020014          2359#   4974#   4984#   5264#
$TEMP = 000032          5788#
```

```
$TESTN  001214          1796#  5733#
$TIME   015552          4352   4427#  4478
$TIMES  001174          1778#  2343#  2832#  2898#  2960#  3046#  3125#  3207#  3568#  3669#  3771#  4946#  5721#
                        5728   5731#  5741
$TKB    001162          1771#  4479   4486   4817   5422   5433   5450   5504   5510
$TKINT  017076          2391   4815#
$TKS    001160          1770#  4487#  4818#  4832#  4845#  4847#  5422   5431   5447   5471#  5502   5508
$TKSRV  017126          4815   4825#
$TN     = 000012        1526#  1537   2830   2832#  2896   2898#  2958   2960#  3044   3046#  3123   3125#  3205
                        3207#  3566   3568#  3667   3669#  3769   3771#
$TNPWR  022606          5595   5596   5616#
$TPB    001166          1773#  5157#  5168
$TPFLG  001173          1777#  5106   5168
$TPS    001164          1772#  5155   5168
$TRAP   023272          2338   5751#
$TRAP2  023314          5762#  5773
$TRK    = 000011        2691#  2909#  2932#  2933   2997#  3012#  3013   3015#  3248#  3249   3251#  3334#  3335
                        3482   3601#  3602   3643#  3645   3743#  3744   7391#
$TRP    = 000015        5766#  5775#  5776#  5777#  5778#  5779#  5780   5781#  5782   5783#  5784#  5785#  5786#
                        5787#  5788#
$TRPAD  023326          5756   5773#  5788
$TSTM   001104          1736#
$TSTNM  001116          1750#  4945#  5004   5040   5683   5710   5732#  5733   5738   5742
$TTYIN  017552          4862   4863   4875   4898   4912   4916#
$TYPBN= ******  U       5779
$TYPDS  021520          5363#  5778
$TYPE   020352          4635   4647   5106#  5200   5766   5774
$TYPEC  020564          5136   5143   5150   5155#  5156   5473
$TYPEX  020632          5161   5163   5166#
$TYPOC  021316          5303#  5775
$TYPON  021332          5302   5305#  5777
$TYPOS  021272          5298#  5776
$UNIT   001222          1799#  1853
$UNITM  001110          1738#
$USWR   001234          1806#
$VECT1  001260          1831#
$VECT2  001262          1832#
$WROM   = 000004        2689#  2906#  2976#  3218#  3292#  3301#  3327#  3381   3546   3579#  3678#  3708#  3711#
                        3734#  4274   4289   4321   7388#
$XTSTR  023024          5697#
$$GET4= 000001          4957#  4962
$OFILL  021515          5299#  5303#  5313   5348#
$40CAT= ******  U       5014   5694
.       = 040354        1693#  1697#  1705#  1711   1712#  1714#  1716#  1717#  1723   1724#  1726#  1728#  1740
                        1747#  1784   1995#  2331   2360   2361   4084   4087   4093   4096   4099   4102   4105
                        4108   4111   4114   4122   4125   4564   4916#  4919#  4984   4985   5040   5087#  5168
                        5221#  5243   5267   5417#  5422   5527   5636#  5676#  5741   5742   5788   7098#  7099#
                        7100#  7101#  7102#  7103#  7104#  7105#  7106   7351#  7381#  7399#  7401#  7403#  7405#
                        7407#  7409#  7411#  7413#  7415#  7417#  7419#  7421#  7423#  7425#  7427#  7429#  7496#
                        7497
.$ASTA= ******  U       5173   5176
.$/     = 001100        1723#  1728
```

```
CKCHR   1526#  4737  4751
CKDIG   1526#
CKNUM   1526#
COMMEN  1660#
ENDCOM  1660#
ERRCAL  5792#  6610  6662  6665  6929  6990
ERROR   1554#  4135  4136  4137  4138  4246  4247  4248  4249  6611  6663  6667  6931  6991  7293
ESCAPE  1660#
GETPRI  1660#  4970  7235
GETSWR  1526#  1660#  2392
MORETA  1717#  1839
MULT    1660#
NEWTST  1660#  2830  2896  2958  3044  3123  3205  3566  3667  3769
POP     1660#  3510  3551  3989  4508  5215  5216  5252  5253  5404  5567
PUSH    1660#  3462  3532  3933  4497  5176  5178  5199  5233  5239  5363  5547
REPORT  1660#
RM03.D     1#  5792
SCOPE   1555#  2831  2897  2959  3045  3124  3206  3567  3668  3770
SETPRI  1660#
SETTRA  5766#  5775  5776  5777  5778  5780  5782  5783  5784  5785  5786  5787
SETUP   1660#  2326
SKIP    1660#
SLASH   1660#
SPACE   1660#
STARS   1660#  1669  1709  1720  1722  1729  1743  1784  1787  1883  2222  2830  2896  2958  3044
        3123   3205  3566  3667  3769  4336  4340  4384  4388  4937  4988  5043  5091  5171  5223
        5245   5275  5353  5421  5424  5492  5531  5577  5640  5680  5745  5790  5803  7215  7219
        7223   7345  7349
SWRSU   1660#  2362#
TRMTRP  5766#  5788
TYPBIN  1660#
TYPDEC  1660#
TYPNAM  1660#
TYPNUM  1660#
TYPOCS  1660#
TYPOCT  1660#  5055  5079  5442
TYPTXT  1660#
$$CMRE  1741#
$$CMTM  1741#
$$ESCA  1660#
$$NEWT  1660#  2830  2896  2958  3044  3123  3205  3566  3667  3769
$$SET   5766#  5775  5776  5777  5778  5780  5782  5783  5784  5785  5786  5787
$$SETM  2378#
$$SETU  2346#
$$SKIP  1660#
.EQUAT  1526#  1550
.HEADE  1526#  1527
.SETUP  1526#  2311
.SWRHI  1526#  1537
.SWRLO  1526#  1549#
.SACTI  1526#  1707
.SAPTB  1526#  1785#
.SAPTH  1526#  1718
.SAPTY  1526#  5169
.SCATC  1526#  1691
.SCMTA  1526#  1741
```

```
.$DB2D    1526#   5575
.$DB5O    1526#   5638
.$EOP     1526#   4935
.$ERRO    1526#   4986
.$ERRT    1526#   5041
.$POWE    1526#   5227
.$READ    1526#   5419
.$SAVE    1526#   5529
.$SCOP    1526#   5678
.$SIZE    1526#   7221
.$TRAP    1526#   5743
.$TYPD    1526#   5351
.$TYPE    1526#   5089
.$TYPO    1526#   5273

. ABS.   040354      000

 ERRORS DETECTED:  0

 RM03:CZRMIB,RM03:CZRMIB.SEQ/NL:MD:MC:CND:TOC/LI:ME/DOC/SOL/CRF=RM03:RMDRV5.P11,RM03:CZRMIB.P11
 RUN-TIME: 30 20 2 SECONDS
 RUN-TIME RATIO: 820/52=15.5
 CORE USED:  41K  (81 PAGES)

 DOCUMENT PAGES:  163
```