RM03

DRIVE COMPATIBILITY TEST

MD-11-DZRMI-A

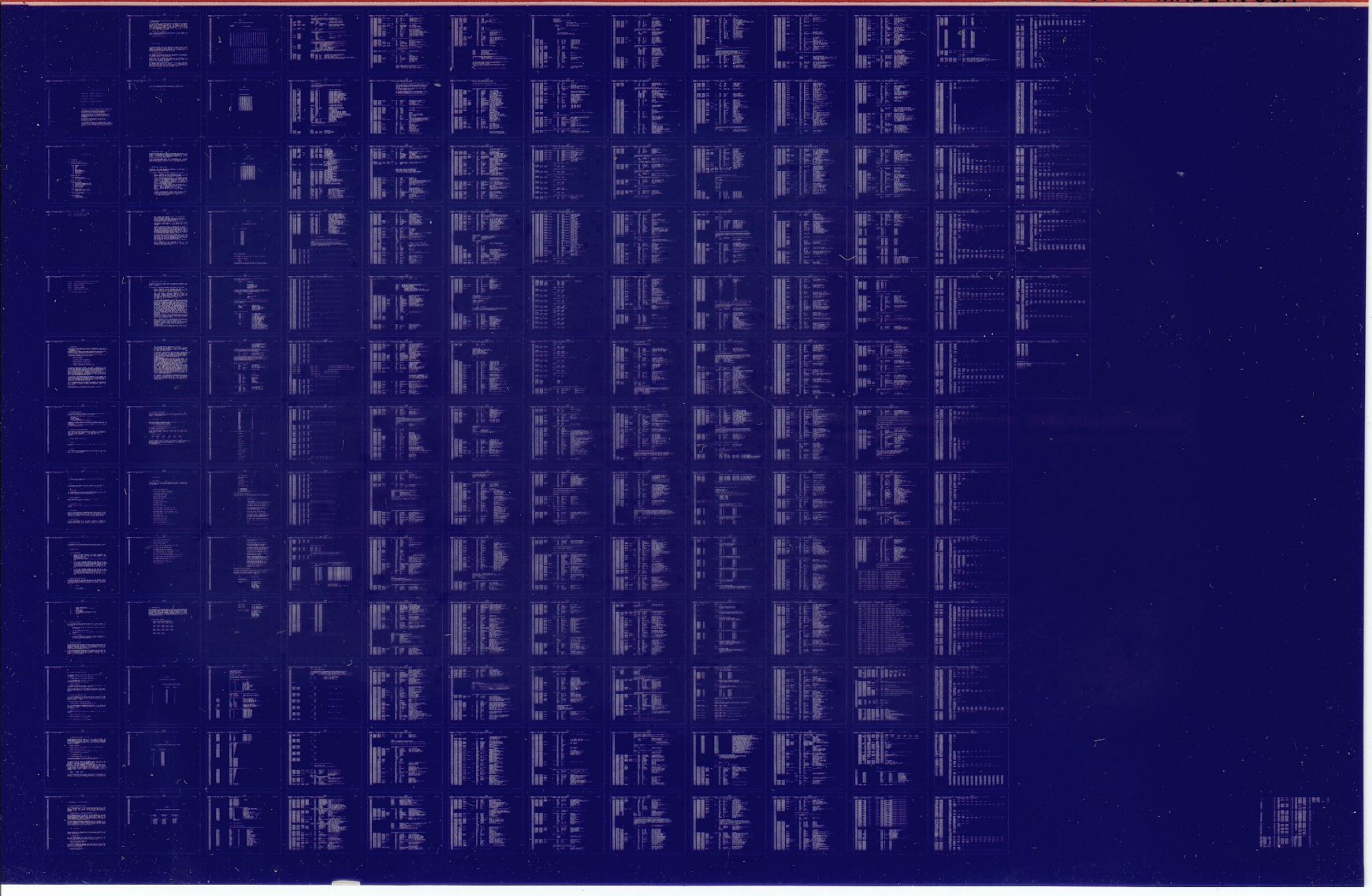
EP-DZRMI-A-DL-A

FICHE 1 OF 1

OCT 1977

digital

MADE IN USA



B01

```
                                .REM    @
```

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

PRODUCT CODE: MAINDEC-11-DZRMI-A-D

PRODUCT NAME: RM03 DRIVE COMPATIBILITY TEST

DATE CREATED: AUGUST 1977

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: C. CHEN

TABLE OF CONTENTS

```
   105                              TABLE A -  BASIC READ/WRITE TEST SECTORS
   106
   107                              TABLE B -  WORSE CASE DATA PATTERN
```

```
                                TABLE C -  CYLINDER BLOCK ASSIGNMENT FOR A GIVEN SURFACE
                                TABLE D -  BASIC CYLINDER BLOCK LAYOUT EXAMPLE
                                TABLE E -  OVERWRITE CYLINDERS
                                TABLE F -  SELF-TEST CYLINDERS
                                TABLE G -  PSEUDO-RANDOM DATA PATTERN

                                12.0  RM03 SOFTWARE DRIVER DOCUMENT
```

1.0 INTRODUCTION

THE PURPOSE OF THIS PROGRAM IS TO VERIFY THE COMPATIBILITY OF UP TO 16 RM03$ DRIVES WHICH MAY RESIDE ON ONE / MORE RH11(70)/RM03 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 RM03 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 OPERAT_ONAL 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:

(MAINDEC DZRMI-A) - RM03 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:

(MAINDEC DZRMI-A)- RM03 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 PROGRAM 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 :

(MAINDEC DZRMI-A) - 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 DECREMENTED. 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 NO.	DRIVE READ	OVRWRT OFST-	OVRWRT OFST+	READ OFST-	READ 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 FULLOWING 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	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2
115	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5
116	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9
117	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
118	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4
119	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11
120	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3
121	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12
122	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6
123	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1
124	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13
125	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10
126	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8
127	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7

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	!
3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	!
6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	!
10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	!
15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	!
5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	!
12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	!
4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	!
13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	!
7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	!
2	3	4	5	6	7	8	9	10	1	!1	12	13	14	15	0	1
14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	!
11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	!
9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	!
8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	!

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

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	177070
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 MCI
                                .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   ERRTBS          ;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   3,0
                .WORD   400,ERRTBS,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
RECALIRATE	107	P
DRIVE CLEAR	111	N
RELEASE	113	N
OFFSET	115	P
RETURN TO CENTER	117	P
READIN PRESET	121	N
PACK ACKNOWLEDGE	123	N
SEARCH	131	P
GET REGISTER(S)	141	S
SET FORMAT	143	S
SELECT DRIVE	145	S

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

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

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
9(3)(4)	SOFTWARE TIMEOUT OCCURRED ON THIS DRIVE
8(4)	SOFTWARE TIMEOUT OCCURRED ON ANOTHER DRIVE
7	DONE
6(2)	ERROR OCCURRED DURING AN I/O OPERATION

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

                                        4(2)            CORRECTABLE UNSAFE CONDITION OCCURRED

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

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

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

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

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

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

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

        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
```

```
1508                                            ERROR (MCPE=1)          RD.WRD= WORD READ
1509
1510                                    4       MASSBUS PARITY          WRT.AD= ADDRESS OF REG. WRITTEN
1511                                            ERROR (PAR=1)           WRT.WD= WORD WRITTEN
1512                                                                    RD.WRD= WORD READ BACK
1513
1514                                    5       ADDRESS PLUG CHANGE     R1= DRIVE NUMBER
1515                                            BIT SET ('OPE' ERROR)   R3= ATA BIT
1516                                                                   *R4= RMCS1'S ADDRESS
1517                                                                    R5= (RMAS)
1518                                                                    RMERRS  =RMDS
1519                                                                    RMERRS+2=RMER1
1520                                                                    RMERRS+4=RMER2
1521                                                                    RMERRS+6=RMMR2
1522
1523                                    * THIS IS THE ACTUAL UNIBUS ADDRESS (176700)
1524
1525                                    .
```

```
                                .TITLE  MAINDEC-11-DZRMI A0/00, RM03 DRIVE COMPATIBILITY  TEST
                                ;*COPYRIGHT (C) 1976
                                ;*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,SW1
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
                000174                  .=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

                000220                  .=220                   ;SECOND PASS STARTING ADDRESS
000220  000137  003102                  JMP     @#START2
                                .SBTTL  ACT11 HOOKS

                                ;;**************************************************************
                                ;HOOKS REQUIRED BY ACT11
                000224                  $SVPC=.                 ;SAVE PC
                000046                  .=46
000046  017312                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                000052                  .=52
000052  040000                          .WORD   40000           ;;2)SET LOC.52 TO 40000
                000224                  .=$SVPC                 ;; RESTORE PC
                001100                  .=1100
                                .SBTTL  APT PARAMETER BLOCK

                                ;;**************************************************************
                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                ;;**************************************************************
                001100                  .$X=.   ;;SAVE CURRENT LOCATION
                000024                  .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024  000200                          200     ;;FOR APT START UP
                000044                  .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044  001100                          $APTHDR ;;POINT TO APT HEADER BLOCK
                001100                  .=.$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
001160  177560                  $TKS:   177560                  ;;TTY KBD STATUS
001162  177562                  $TKB:   177562                  ;;TTY KBD BUFFER
001164  177564                  $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001166  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                                  ;*                              11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
 1810                                  ;*                              11/70=06,PDQ=07,Q=10
 1811                                  ;*                      BIT 10=REAL TIME CLOCK
 1812                                  ;*                      BIT  9=FLOATING POINT PROCESSOR
 1813                                  ;*                      BIT  8=MEMORY MANAGEMENT
 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                                  ;*                              900 NSEC CORE=001
 1818                                  ;*                              300 NSEC BIPOLAR=002
 1819                                  ;*                              500 NSEC MOS=003
 1820  001242  000000                  $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
 1821                                  ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
 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               ;'CONTFOL 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                    SYSA??: .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  RDN0:   .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  036506          BLKADR: .WORD   DRIV0            ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 0
2164  002602  036530                  .WORD   DRIV1            ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 1
2165  002604  036552                  .WORD   DRIV2            ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 2
2166  002606  036574                  .WORD   DRIV3            ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 3
2167  002610  036616                  .WORD   DRIV4            ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 4
2168  002612  036640                  .WORD   DRIV5            ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 5
2169  002614  036662                  .WORD   DRIV6            ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 6
2170  002616  036704                  .WORD   DRIV7            ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 7
2171  002620  036726                  .WORD   DRIV10           ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 10
2172  002622  036750                  .WORD   DRIV11           ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 11
2173  002624  036772                  .WORD   DRIV12           ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 12
2174  002626  037014                  .WORD   DRIV13           ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 13
2175  002630  037036                  .WORD   DRIV14           ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 14
2176  002632  037060                  .WORD   DRIV15           ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 15
2177  002634  037102                  .WORD   DRIV16           ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 16
2178  002636  037124                  .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  031616                          EM1                     ;RH70 INTERRUPT OCCURRED (RMAS = 0)
002744  034244                          DH1
002746  034666                          DT1
002750  035012                          DF1

                                ;ERROR 2

002752  031665                          EM2                     ;UNEXPECTED ATTENTION OCCURRED
002754  034253                          DH2
002756  034672                          DT2
002760  035013                          DF2

                                ;ERROR 3

002762  031723                          EM3                     ;MASSBUS PARITY ERROR (MCPE=1)
002764  034335                          DH3
002766  034710                          DT3
002770  035021                          DF3

                                ;ERROR 4

002772  031761                          EM4                     ;MASSBUS PARITY ERROR (PAR=1)
002774  034366                          DH4
002776  034720                          DT4
003000  035024                          DF4

                                ;ERROR 5

003002  032016                          EM5                     ;ADDRESS PLUG BIT CHANGED
003004  034253                          DH2
003006  034672                          DT2
003010  035013                          DF2

                                ;ERROR 6

003012  032052                          EM6                     ;RH11 DIDN'T RESPOND TO ADDRESSING
003014  034430                          DH6
```

```
2280  003016  034732                        DT6
2281  003020  035012                        DF1
2282
2283                                  ;ERROR 7
2284
2285  003022  000000                        0
2286  003024  034441                        DH7
2287  003026  034736                        DT7
2288  003030  000000                        0
2289
2290                                  ;ERROR 10
2291
2292  003032  000000                        0
2293  003034  034512                        DH10
2294  003036  034752                        DT10
2295  003040  000000                        0
2296
2297                                  ;ERROR 11
2298
2299  003042  000000                        0
2300  003044  034563                        DH11
2301  003046  034766                        DT11
2302  003050  000000                        0
2303
2304                                  ;ERROR 12
2305
2306  003052  000000                        0
2307  003054  034634                        DH12
2308  003056  035002                        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  022362  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
2335  003136  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
```

```
2336  003144  012737  017366  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
2337  003152  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
2338  003160  012737  022642  000034          MOV     #STRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
2339  003166  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
2340  003174  012737  020452  000024          MOV     #SPWRON,@#PWRVEC ;;POWER FAILURE VECTOR
2341  003202  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
2342  003210  013737  017260  017252          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 "$RTRN", IN
2347                                  ;;THE "END-OF-PASS" (SEOP) ROUTINE, WITH A "RTI" OR "RTT".
2348  003234  012737  017356  000014          MOV     #$RTRN,@#TBITVEC ;;SET "T" BIT VECTOR TO $RTRN
2349  003242  012737  000340  000016          MOV     #340,@#TBITVEC+2 ;;LEVEL 7
2350  003250  012737  000002  017356          MOV     #RTI,$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$,-(SP)      ;;AND PC
2354  003272  000006                          RTT                     ;;TRY THE RTT
2355  003274  012737  000006  017356  64$:    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  017364                  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  001002                          BNE     1$              ;BR IF NOT
2387  003470  104401  037262                  TYPE    ,TITLE          ;NAME AND MANDEC NUMBER
2388  003474  004737  016462          1$:     JSR     PC,$TKINT       ;TURN ON THE KEYBOARD INTERRUPT
2389                          .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
2390  003500  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
2391  003504  001012                          BNE     71$             ;;BRANCH IF YES
```

```
2392  003506  123727  001230  000001          CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
2393  003514  001406                          BEQ     71$             ;;BRANCH IF YES
2394  003516  023727  001154  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
2395  003524  001005                          BNE     72$             ;;BRANCH IF NO
2396  003526  104406                          GTSWR                   ;;GET SOFT-SWR SETTINGS
2397  003530  000403                          BR      72$
2398  003532  112737  000001  001150  71$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
2399  003540                          72$:
2400  003540  013737  001274  023076          MOV     $RMADR,RMADR    ;RH11 ADDRESS
2401  003546  013737  001276  023100          MOV     $RMVEC,RMVEC    ;RH11 VECTOR ADDRESS
2402  003554  012705  001776          2$:     MOV     #ASNLST,R5      ;START OF AREA TO CLEAR
2403  003560  005025                  3$:     CLR     (R5)+
2404  003562  022705  002560                  CMP     #INDST,R5       ;LOOK FOR END OF CLEAR AREA
2405  003566  001374                          BNE     3$              ;BR IF NOT FINISHED
2406  003570  012706  001100                  MOV     #STACK,SP       ;SETUP THE STACK POINTER
2407  003574  005037  177776                  CLR     PS              ;CLEAR THE PROCESSOR STATUS WORD
2408  003600  013737  001316  001350          MOV     HZ,SIXTEE       ;1/60 TH OR 1/50 TH SECOND COUNTER VALUE
2409  003606  005037  001342                  CLR     HOUR            ;CLEAR THE HOUR'S COUNTER
2410  003612  005037  001344                  CLR     MINUTE          ;CLEAR THE MINUTE'S COUNTER
2411  003616  005037  001346                  CLR     SECOND          ;CLEAR THE SECOND'S COUNTER
2412  003622  005037  001334                  CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
2413
2414                                  ;ROUTINE TO DETERMINE BUFFER AREA SIZE
2415
2416  003626  005227  177777          SIZMEM: INC     #-1             ;SEE IF TIME TO SIZE MEMORY
2417  003632  001002                          BNE     1$              ;BR IF NOT
2418  003634  004737  031216                  JSR     PC,$SIZE        ;SEE HOW MUCH MEMORY ON SYSTEM
2419  003640  013737  031312  001330  1$:     MOV     $LSTAD,LSTAD    ;SAVE THE LAST ADDRESS
2420  003646  023727  001330  160000          CMP     LSTAD,#160000   ;OVER 28K ?
2421  003654  101403                          BLOS    2$              ;NO,THEN DON'T SET THE NEW LIMIT
2422  003656  012737  160000  001330          MOV     #160000,LSTAD   ;SET NEW LIMIT
2423  003664  162737  005670  001330  2$:     SUB     #1500.*2,LSTAD  ;SAVE XXDP LOADER AND ABSOLUTE LOADER
2424
2425                                  ;       SET UP THE OTHER SYSTEM DEVICES THAT
2426                                  ;       THE PROGRAM WILL USE
2427
2428  003672  004737  014740          SETVEC: JSR     PC,CKCLK        ;START THE CLOCK
2429  003676  012737  177777  023036          MOV     #-1,SAVEFG      ;SET THE SAVE REGISTERS FLAG
2430
2431                                  ;SETUP IF 'XXDP' OR 'ACT11' OPERATION
2432
2433  003704  005001                  MONTR:  CLR     R1              ;DRIVE #
2434  003706  005002                          CLR     R2              ;AVAIL TABLE INDEX
2435  003710  005003                          CLR     R3              ;DRIVE# X 2
2436  003712  016300  002600          1$:     MOV     BLKADR(R3),R0   ;LOAD DPB ADDRESS
2437  003716  004737  015426                  JSR     PC,CLRDPB       ;CLEAR DPB BLOCK
2438  003722  022322                  2$:     CMP     (R3)+,(R2)+     ;INCREMENT INDEX
2439  003724  005201                          INC     R1              ;NEXT DRIVE
2440  003726  022701  000007                  CMP     #7,R1           ;ALL DRIVE ASSIGN ?
2441  003732  002367                          BGE     1$              ;NO
2442
2443
2444                                  ;       ASSIGN LOGICAL DRIVES TO BE TEST IN THE PASS1 AND PASS 2
2445                                  ;       THREE WORDS ARE USED IN THE BIT MAPS:
2446                                  ;       ASNLST:   SPECIFIES THE LOGICAL DRIVES ASSIGNED
2447                                  ;       ASSGN1:   SPECIFIES THE LOGICAL DRIVES WILL BE TESTED IN PASS1.
```

```
                                ;       ASSGN2:   SPECIFIES THE LOGICAL DRIVES WILL BE TESTED IN PASS2.
                                ;
                                ;       EACH LOGICAL DRIVE HAS A HISTORY FILE LABELED DIRV'Z (Z=0 TO 17)
                                ;       THE LOCATIONS LABELED $SYSNM AND $PHYRD IN THE HISTORY FILE
                                ;       STORE THE SYSTEM NAME (A TO H) AND PHYSICAL DRIVE NUMBER
                                ;       (0 TO 7).
                                ;       THE SUB-SYSTEM ADDRESS AND INTERRUPT VECTOR ARE STOREED
                                ;       IN THE TABLE LABELED "SYSADR:".
                                ;
                                ;       THE LOCATIONS SYSADR AND SYSADR+2 FOR SUB-SYSTEM A,SYSADR+4
                                ;       AND SYSADR+6 FOR SUBSYSTEM B , ETC,THE FIRST WORD
                                ;       IS THE SUB SYSTEM ADDRESS WHILE THE SECOND WORD IS THE VECTOR
                                ;
                                ;       THE LOGICAL DRIVES ARE ASSIGNED FROM CONSOLE KEYBOARD.

003734  012700  001776          MOD00:  MOV     #ASNLST,R0      ;ADDRESS OF 1ST BIT MAPS IN R0
003740  012701  002560                  MOV     #INDST,R1       ;LAST ADDRESS TO CLEAR
003744  005020                  1$:     CLR     (R0)+           ;CLEAR CURRENT POINTED ADDRESS
003746  020100                          CMP     R1,R0           ;ALL DONE ?
003750  101375                          BHI     1$              ;NO,THEN BRANCH BACK
003752  012737  000101  001272          MOV     #'A,$CDW2       ;TEMP STORATE OF SYS NAME

003760  005037  001222          MOD21:  CLR     DRIVE           ;TEM STORAGE OF PHYSICAL DRIVE BIT MAP
003764  104401  001205                  TYPE    ,$CRLF
003770  104401  035030                  TYPE    ,MESG1          ;SUB-SYSTEM
003774  104401  001272                  TYPE    ,$CDW2          ;A TO H (ONE OF THEM)
004000  104401  036121                  TYPE    ,LINSP
004004  104401  035041                  TYPE    ,MESG2          ;DRIVES
004010  104401  036201                  TYPE    ,COLON          ;COLON
004014  104411                          RDLIN                   ;READ IN THE DRIVE NUMBERS
004016  012601                          MOV     (SP)+,R1        ;GET THE INPUT LINE ADDRESS
004020  105711                  1$:     TSTB    (R1)            ;END OF STRING ?
004022  001415                          BEQ     2$              ;YES
004024  004537  016116                  JSR     R5,CK.OCT       ;CHECK THE DIGIT MUST 0 TO 7,RETURN VALUE IN R2
004030  000753                          BR      MOD21           ;INCORRECT DRIVE NUMBER,ENTER AGAIN
004032  156237  023064  001222          BISB    ATABIT(R2),DRIVE  ;SET THE ASSIGNED PHYSICAL DRIVE BIT
                                                                ;R2 THE DRIVE NUMBER FROM CK.OCT RT.
004040  005201                          INC     R1              ;NEXT BYTE OF INPUT LINE
004042  105711                          TSTB    (R1)            ;END OF STRING ?
004044  001404                          BEQ     2$              ;YES
004046  122721  000054                  CMPB    #',,(R1)+       ;MUST BE A ,
004052  001342                          BNE     MOD21           ;ENTER AGAIN ,IF NOT
004054  000761                          BR      1$              ;LOCATE NEXT DRIVE
004056  005737  001332          2$:     TST     CHGADR          ;START AT 200 ?
004062  003036                          BGT     MOD22           ;BRANCH IF SO
004064  013701  001272                  MOV     $CDW2,R1        ;SYS NAME ASCII FROM A TO H
004070  042701  177760                  BIC     #177760,R1      ;LEFT ONLY 4 BITS
004074  005301                          DEC     R1              ;ADJUST INDEX VALUE
004076  006301                          ASL     R1              ;2 WORD INDEX VALUE
004100  006301                          ASL     R1              ;
004102  062701  002004                  ADD     #SYSADR,R1      ;SYS ADDRESS TABLE ADDRESS
```

```
2504  004106  010137  001270            MOV     R1,$CDW1        ; SYS ADDRESS TABLE'S ENTRY TO CDW1
2505  004112  000400            3$:     BR      MOD23           ;
2506
2507  004114  005737  001222    MOD23:  TST     DRIVE           ;CHECK IF ANY PHYSICAL DRIVE(S) ASSIGNED
2508  004120  001416                    BEQ     2$              ;BRANCH IF NONE
2509  004122  013700  001270    1$:     MOV     $CDW1,R0        ;SYS ADDRESS TABLE ENTRY
2510  004126  011037  001274            MOV     (R0),$RMADR     ; SYS ADDRESS
2511  004132  016037  000002  001276    MOV     2(R0),$RMVEC    ; SYS VECTOR
2512  004140  004737  031314            JSR     PC,BUSADR       ;CHECK THE ADDRESS WITH THE OPERATOR
2513  004144  013710  001274            MOV     $RMADR,(R0)     ;NEW RH11/70 ADDRESS  INTO TABLE
2514  004150  013760  001276  000002    MOV     $RMVEC,2(R0)    ;NEW VECTOR OF RH11/70 INTO TABLE
2515  004156  000407            2$:     BR      MOD11           ;BRANCH TO NEXT MODULE
2516
2517
2518  004160            MOD22:
2519  004160  013737  001274  002004  3$:     MOV     $RMADR,SYSADR   ;LOAD THE SYSTEM ADDRESS TABLE
2520  004166  013737  001276  002006          MOV     $RMVEC,SYSADR+2 ;LOAD THE VECTOR
2521  004174  000400            4$:     BR      MOD11
2522
2523
2524
2525
2526                              ;       $CDW2-- ASCII NAME OF SUB SYSTEM (A TO H)
2527                              ;       $CDW1--ENTRY TO THE SYS ADDRESS TABLE
2528                              ;       DRIVE--PHYSICAL DRIVES TO BE ASSIGNED
2529                              ;       THIS SECTION OF CODING USES THE ABOVE PARAMETERS
2530                              ;       TO SET UP THE BIT MAP OF ASNLST,ASSGN1,ASSGN2
2531                              ;
2532                              ;
2533
2534
2535
2536  004176  005737  001222    MOD11:  TST     DRIVE   ;ANY DRIVE ASSIGN ?
2537  004202  001002                    BNE     MOD30           ;BRANCH IF ANY
2538  004204  000137  004360            JMP     MOD12           ;BRANCH,IF NONE
2539  004210  022737  177777  001776  MOD30:  CMP     #-1,ASNLST      ;ALL 16 LOGICAL DRIVES HAS BEEN ASSIGNED ?
2540  004216  001457                    BEQ     5$              ;YES,THEN DON'T ASSIGN NEW DRIVES
2541  004220  013700  001776            MOV     ASNLST,R0       ;FOUND THE AVAILABLE LOGICAL DRIVE LOCATION
2542  004224  012701  000001            MOV     #1,R1           ;START FROM LOGICAL DRIVE 0
2543  004230  005002                    CLR     R2              ;INDEX VALUE
2544  004232  030100            1$:     BIT     R1,R0           ;IS THE LOGICAL DRIVE AVAILABLE ?
2545  004234  001406                    BEQ     2$              ;YES
2546  004236  000241                    CLC
2547  004240  006101                    ROL     R1              ;NEXT LOGICAL DRIVE
2548  004242  103445                    BCS     5$              ;BRANCH IF NONE IS AVAILABLE
2549  004244  062702  000002            ADD     #2,R2           ;INCREMENT INDEX VALUE
2550  004250  000770                    BR      1$              ;LOCATE NEXT LOGICAL DRIVE
2551  004252  016204  002100    2$:     MOV     BLKADR(R2),R4   ;GET THE LOGICAL DRIVE'S HISTORY FILE
2552  004256  113764  001272  000014    MOVB    $CDW2,$SYSNM(R4)  ;LOAD THE ASCII SYS NAME
2553  004264  050137  001776            BIS     R1,ASNLST               ;SET LOGICAL DRIVE ASSIGN BIT
2554  004270  050137  002000            BIS     R1,ASSGN1       ;SET PASS 1 BIT
2555  004274  050137  002002            BIS     R1,ASSGN2       ;SET PASS2 BIT
2556  004300  013700  001222            MOV     DRIVE,R0        ;LOAD THE ASCII PHYSICAL DRIVE #
2557  004304  012701  000001            MOV     #1,R1           ;START FROM DRIVE 0
2558  004310  005002                    CLR     R2              ;DRIVE #
2559  004312  030100            3$:     BIT     R1,R0           ;IS THE PHYSICAL DRIVE ASSIGNED
```

```
2560  004314  001005                        BNE     4$              ;YES,THEN BRANCH
2561  004316  000241                        CLC                     ;LOCATE THE NEXT DRIVE
2562  004320  006101                        ROL     R1              ;SHIFT ONE BIT LEFT
2563  004322  103415                        BCS     5$              ;BRANCH IF NO MORE DRIVES
2564  004324  005202                        INC     R2              ;PHYSICAL DRIVE NUMBER
2565  004326  000771                        BR      3$              ;LOOPING BACK
2566  004330  110214                4$:     MOVB    R2,(R4)         ;LOAD THE PHYSICAL DRIVE # INTO HISTORY FILE
2567  004332  062702  000060              ADD     #'0,R2          ;ASCII CODE OF DRIVE  #
2568  004336  110264  000015              MOVB    R2,$PHYDR(R4)   ;PHYSICAL DRIVE NUMBER
2569  004342  112764  000011  000016      MOVB    #HT,$GAP(R4)    ;MAKE UP FOR SCORE TYPE
2570  004350  040137  001222              BIC     R1,DRIVE        ;RESET THE ASSIGNED DRIVE BIT
2571  004354  001315                        BNE     MOD30           ;BRANCH IF NOT ALL DONE
2572  004356  000400                5$:     BR      MOD12           ;CHECK OTHER SUB SYSTEM
2573
2574
2575  004360  005737  001332        MOD12:  TST     CHGADR  ;200 START ?
2576  004364  003066                        BGT     6$              ;YES,THEN EXIT
2577  004366  022737  177777  001776  1$:   CMP     #-1,ASNLST      ;FULL HOUSE
2578  004374  001462                        BEQ     6$              ;YES,THEN EXIT
2579  004376  104401  035737          2$:   TYPE    ,MESG5          ;WILL TEST
2580  004402  104401  035041                TYPE    ,MESG2          ;DRIVES
2581  004406  005003                        CLR     R3              ;INDEX TO LOAGICAL DRIVE HISTORY FILE
2582  004410  012704  000001                MOV     #1,R4           ;BIT MAP OF ASNLST
2583  004414  030437  001776          3$:   BIT     R4,ASNLST       ;ASSIGNED LOGICAL DRIVE ?
2584  004420  001417                        BEQ     5$              ;NO
2585  004422  016305  002600                MOV     BLKADR(R3),R5   ;LOAD THE HISTORY FILE ADDRESS
2586  004426  126537  000014  001272        CMPB    $SYSNM(R5),$CDW2  ;ON THE SAME SYSTEM ?
2587  004434  001003                        BNE     4$              ;NO
2588  004436  111546                        MOVB    (R5),-(SP)      ;TYPE THE PHYSICAL DRIVE #
2589  004440  104403                        TYPOS
2590  004442     002                        .BYTE   2
2591  004443     000                        .BYTE   0
2592  004444  062703  000002          4$:   ADD     #2,R3           ;INCREMENT TO NEXT LOGICAL DRIVE
2593  004450  000241                        CLC
2594  004452  006104                        ROL     R4              ;BIT MAP OF THE NEXT LOGICAL DRIVE
2595  004454  103401                        BCS     5$              ;BRANCH IF ALL LOGICAL DRIVE'S CHECKED
2596  004456  000756                        BR      3$              ;BRANCH BACK
2597  004460  104401  036121          5$:   TYPE    ,LINSP
2598  004464  104401  035752                TYPE    ,MESG6          ;ON
2599  004470  104401  035030                TYPE    ,MESG1          ;SUB SYSTEM
2600  004474  104401  001272                TYPE    ,$CDW2          ; A TO H
2601
2602  004500  005237  001272                INC     $CDW2           ;CHECK NEXT SUB SYSTEM
2603  004504  122737  000110  001272        CMPB    #'H,$CDW2               ;ALL EIGHT SUB SYSTEMS CHECKED?
2604  004512  103413                        BLO     6$              ;YES
2605  004514  104401  001205                TYPE    ,$CRLF
2606  004520  104401  035756                TYPE    ,MESG7          ;ASK FOR OTHER SUB SYSTEM
2607  004524  104411                        RDLIN                   ;READ IN A LINE
2608  004526  012605                        MOV     (SP)+,R5        ;CHECK THE FIRST CHARACTER
2609  004530  122715  000131                CMPB    #'Y,(R5)        ;ENTER Y ?
2610  004534  001002                        BNE     6$              ;BRANCH IF NOT
2611  004536  000137  003760                JMP     MOD21           ;SET UP OTHER SUB SYSTEM
2612  004542  005737  001332          6$:   TST     CHGADR          ;START AT 220
2613  004546  001402                        BEQ     7$              ;YES,EXECUTE PASS2 ONLY
2614  004550  000137  004564                JMP     XPASS1          ;TO PASS1
2615  004554  005037  002000          7$:   CLR     ASSGN1          ;CLEAR THE PASS1 BIT MAP
```

```
2616  004560  000137  006772                  JMP     XPASS2          ;BRANCH TO PASS2
2617
2618
2619
2620
2621
2622                              ;       PASS ONE , THE VARIAVLES ARE ASSIGNED AS FOLLOWS:
2623                              ;       $CDW1   :  ADDRESS OF THE CURRENT LOGICAL DRIVE HISTORY FILE
2624                              ;       $CDW2   :  SYSTEM NAME A THROUGH H
2625                              ;       $DEVM   :   CURRENT LOGICAL DRIVE #
2626                              ;       ASNLST  :       ASSIGNED LOGICAL DRIVES
2627                              ;       ASSGN1  :       ASSIGNED LOGICAL DRIVES IN THIS PASS
2628                              ;       R0      :       ADDRESS OF DPB BLOCK FEEDED INTO DRIVER-HANDLER
2629                              ;       R1      :       TEMP STORAGE OF ADDRESS OF THE LOGICAL BLOCK
2630
2631
2632
2633
2634  004564  005737  002000      XPASS1: TST     ASSGN1          ;ANY DRIVE ASSIGNED FOR PASS1
2635  004570  001002                  BNE     1$              ;BRANCH IF THEY ARE
2636  004572  000137  005432          JMP     ENDX1           ;END OF PASS 1
2637  004576  005037  001266      1$: CLR     $DEVM           ;INDEX OF LOGICAL BLOCK
2638  004602  013737  002600  001270  MOV     BLKADR,$CDW1    ;ADDRESS OF LOGICAL BLOCK 0
2639  004610  112737  000101  001272  MOVB    #'A,$CDW2       ;SYS NAME STARTS FROM A
2640  004616  012777  013256  016254  MOV     #IDLEX,@RMVEC   ;RESET ALL INTERRUPT VERTOR
2641  004624  005077  016252          CLR     @RMVEC+2        ;CLEAR THE INTERRUPT LEVEL
2642  004630  013737  002004  023076  MOV     SYSADR,RMADR    ;LOAD SYS-TEM A INTO
2643  004636  013737  002006  023100  MOV     SYSADR+2,RMVEC  ;DIRVER-HANDLER
2644  004644  013701  001270          MOV     $CDW1,R1        ;R1=ADDRESS OF LOGICAL BLOCK 1
2645  004650  012777  013256  016222  MOV     #IDLEX,@RMVEC   ;RESET ALL INTERRUPT VECTOR
2646  004656  005077  016220          CLR     @RMVEC+2        ;CLEAR THE INTERRUPT LEVEL
2647  004662  104401  001205          TYPE    ,$CRLF
2648  004666  104401  035053          TYPE    ,MESG3
2649  004672  104401  001205          TYPE    ,$CRLF
2650  004676  104401  035101          TYPE    ,MESG4
2651  004702  104401  001272          TYPE    ,$CDW2
2652  004706  111137  001222          MOVB    (R1),DRIVE      ;LOAD THE PHYSICAL DRIVE #
2653  004712  111146                  MOVB    (R1),-(SP)      ;TYPE THE DRIVE #
2654  004714  104403                  TYPOS
2655  004716     002                  .BYTE   2
2656  004717     000                  .BYTE   0
2657  004720  104401  036121          TYPE    ,LINSP
2658  004724  104401  035127          TYPE    ,MESG8
2659  004730  104401  001205          TYPE    ,$CRLF
2660  004734  104401  035142          TYPE    ,MESG9
2661  004740  104401  036121          TYPE    ,LINSP
2662
2663
2664
2665
2666  004744  104411              2$: RDLIN                   ;CHECK IF O.P. READY
2667  004746  012602                  MOV     (SP)+,R2        ;LOCATE THE INPUT LINE
2668  004750  122722  000122          CMPB    #'R,(R2)+       ;FIRST CHARACTER IS A "R" ?
2669  004754  001373                  BNE     2$              ;BRANCH IF NOT
2670  004756  105712                  TSTB    (R2)            ;FOLLOW BY <CR> ?
2671  004760  001371                  BNE     2$              ;BRANCH IF NOT
```

```
2672  004762  004737  023114          JSR     PC,RMINIT       ;INITIALIZE THE SUB SYSTEM
2673  004766  012737  177777  023036  MOV     #-1,SAVEFG      ;SAVE ALL RH/RM REGISTERS
2674  004774  012737  177777  023040  MOV     #-1,SEEKFG      ;DON'T DO IMPLY SEEK
2675  005002  013700  001222          MOV     DRIVE,R0        ;R0=PHYSICAL DRIVE # OF THE SUB SYSTEM
2676  005006  105760  022750          TSTB    DRVSTA(R0)      ;DIRVE EXIST  AND ON LINE ?
2677  005012  003452                  BLE     5$              ;BRANCH IF NOT
2678  005014  105760  022760          TSTB    DRVTYP(R0)      ;DRIVE IS AN RM03 ?
2679  005020  003447                  BLE     5$              ;BRANCH IF NOT
2680                                                          ;SET UP TO RETRIEVE THE BAD SPOT FILE
2681                                                          ;FROM THE LAST TRACK
2682  005022  012700  037146          MOV     #FMTDPB,R0      ;DPB ADDRESS
2683  005026  113710  001222          MOVB    DRIVE,(R0)      ;LOAD THE DRIVE NUMBER
2684  005032  012760  037262  000006  MOV     #ENDPGM,$BUF(R0)  ;LOAD BUFFER ADDRESS
2685  005040  012760  037166  000014  MOV     #RM.REG,14(R0)    ;AREA TO SAVE ALL RH/RM03 REG'S
2686  005046  012760  177400  000004  MOV     #-256.,$WRDM(R0) ;WORD COUNT (NEG)
2687  005054  013760  001372  000012  MOV     CYLIMT,$CYL(R0) ;CYLINDER 822.
2688  005062  113760  001370  000011  MOVB    TRKLMT,$TRK(R0) ;TRACK ADDRESS 4.
2689  005070  105060  000010          CLRB    $SEC(R0)        ;SEC 0
2690  005074  112760  000171  000002  MOVB    #RDDAT,$COMND(R0) ;READ DATA COMMAND
2691
2692  005102  004037  023620  3$:     JSR     R0,RM03         ;CALL THE DRIVER-HANDLER
2693  005106  037146                  FMTDPB                  ;PARAMETER ADDRESS
2694  005110  000774                  BR      3$              ;LOOPING IF QUEUE IS NOT SUCCESSFUL
2695  005112  005737  037164  4$:     TST     FMTDPB+16       ;COMMAND DONE ?
2696  005116  001775                  BEQ     4$              ;BRANCH IF NOT
2697  005120  100013                  BPL     6$              ;BRANCH IF DONE, WITHOUT ERROR
2698  005122  062737  000002  037156  ADD     #2,FMTDPB+10    ;TRY NEXT SECTOR (0,2,4,6,8)
2699  005130  122737  000010  037156  CMPB    #8.,FMTDPB+10   ;ALL FIVE SECTORS CHECKED ?
2700  005136  101361                  BHI     3$              ;NO,THEN TRY AGAIN
2701
2702  005140  104401  035201  5$:     TYPE    ,MESG10         ;DRIVE IS NOT READY
2703  005144  000137  005432          JMP     ENDX1           ;STOP THE TEST
2704
2705
2706                                                          ;BAD SPOT FILE IS RETRIVED,IS STORED
2707                                                          ;FROM ENDPGM+4 TO ENDPGM+256.
2708                                                          ;FIRST WORD CYLINDER #,SECOND WORD
2709                                                          ;TRK AND SEC NUMBERS,FILE IS TERMINATED
2710                                                          ;BY A -1 IN THE CYLINDER NUMBER
2711
2712
2713  005150  012704  037272  6$:     MOV     #ENDPGM+10,R4   ;R4 ADDRESS OF BAD SPOT FILE
2714  005154  022714  177777  7$:     CMP     #-1,(R4)        ;END OF BAD SPOT FILE ?
2715  005160  001411                  BEQ     8$              ;YES
2716  005162  022704  040262          CMP     #ENDPGM+1000,R4 ;END OF BAD SPOT FILE ?
2717  005166  101406                  BLOS    8$              ;BRANCH IF IT IS
2718  005170  004537  005264          JSR     R5,SPOTX        ;CHECK THE CYLINDER POINTED BY (R4)
2719  005174  000415                  BR      10$             ;BRANCH IF BAD SPOT IN THE TEST ZONES
2720  005176  062704  000004          ADD     #4,R4           ;NEXT BAD SPOT ADDRESS
2721  005202  000764                  BR      7$              ;LOOPING BACK
2722
2723  005204  032777  000200  173742  8$:     BIT     #BIT7,@SWR      ;SWITCH 7 SET ?
2724  005212  001404                  BEQ     9$              ;BRANCH IF NOT SET
2725  005214  012700  037146          MOV     #FMTDPB,R0      ;DPB ADDRESS
2726  005220  004737  014406          JSR     PC,PRTBAD       ;PRINT THE BAD SPOT FILE
2727  005224  000137  005450  9$:     JMP     TST1            ;PACK IS ACCEPTABLE
```

```
2728  005230  104401  035245          10$:    TYPE    ,MESG11         ;PACK NOT ACCEPTABLE
2729  005234  104401  001205                  TYPE    ,$CRLF
2730  005240  032777  000200  173706          BIT     #BIT7,@SWR      ;SWITCH 7 SET ?
2731  005246  001404                          BEQ     11$             ;BRANCH IF NOT SET
2732  005250  012700  037146                  MOV     #FMTDPB,R0      ;DPB ADDRESS
2733  005254  004737  014406                  JSR     PC,PRTBAD       ;TYPE THE BAD SPOT FILE
2734  005260  000137  004564          11$:    JMP     XPASS1          ;RESTART PASS 1
2735
2736
2737                                  ;       SUBROUTINE SPOTX
2738                                  ;       CHECK THE CYLINDER POINTED BY (R4) IS IN THE TESTING ZONES
2739                                  ;       (0-15,128-143,256-271,384-399,512-527,640-655,768-783
2740                                  ;       112-127,240-255,368-383,496-511,624-639,752-767,
2741                                  ;       17-32,145-160,273-288,401-416,529-544,657-672,785-800
2742                                  ;       AND 620 )
2743                                  ;       CALLING SEQ:
2744                                  ;       JSR     R5,SPOTX        ;R4=POINT TO CYLINDER NUMBER
2745                                  ;       RET1                    ;ERROR RET
2746                                  ;       RET2                    ;NORMAL RET1
2747
2748
2749  005264  010146          SPOTX:  MOV     R1,-(SP)        ;SAVE R1 THROUGH R3
2750  005266  010246                  MOV     R2,-(SP)
2751  005270  010346                  MOV     R3,-(SP)
2752
2753  005272  005003                  CLR     R3              ;ERROR FLAG
2754  005274  005046                  CLR     -(SP)           ;DUMMY PAIR
2755  005276  005046                  CLR     -(SP)           ;DUMMY PAIR
2756  005300  005046                  CLR     -(SP)           ;ZONE STARTING ADDRESS
2757  005302  012746  000007          MOV     #7,-(SP)        ;SEGMENT NUMER
2758  005306  012746  000021          MOV     #17.,-(SP)      ;ZONE STARTING ADDRESS
2759  005312  012746  000007          MOV     #7,-(SP)        ;SEGMENT NUMBER
2760  005316  012701  000160          MOV     #112.,R1        ;R1=ZONE STARTING ADDRESS
2761  005322  012702  000006          MOV     #6,R2           ;R2=SEGMENT NUMBER
2762  005326  021401          1$:     CMP     (R4),R1         ;CYL IN THE ZONE ?
2763  005330  103410                  BLO     3$              ;BRANCH IF NOT
2764  005332  062701  000017          ADD     #15.,R1         ;CHECK WITH THE UPPER BOND
2765  005336  021401                  CMP     (R4),R1         ;CYL IN THE ZONE ?
2766  005340  101002                  BHI     2$              ;BRANCH IF NOT
2767  005342  052703  000002          BIS     #BIT1,R3        ;SET THE ERROR FLAG
2768
2769  005346  162701  000017  2$:     SUB     #15.,R1         ;RESTORE TO THE LOWER BOND
2770  005352  005302          3$:     DEC     R2              ;DECREMENT THE SEGMENT COUNT
2771  005354  001403                  BEQ     4$              ;ALL SEGMENT CHECKED ?
2772  005356  062701  000200          ADD     #128.,R1        ;ADJUST ZONE STARTING ADDRESS
2773  005362  000761                  BR      1$              ;LOOPING BACK UNTIL ALL SEGMENTS ARE CHECKED
2774
2775  005364  012602          4$:     MOV     (SP)+,R2        ;POP THE NEXT SET OF ZONE PARAMETERS
2776  005366  012601                  MOV     (SP)+,R1
2777  005370  005702                  TST     R2              ;DUMMY PAIR
2778  005372  001355                  BNE     1$              ;BRANCH IF NOT
2779  005374  005701                  TST     R1              ;DUMMY PAIR
2780  005376  001353                  BNE     1$              ;BRANCH IF NOT
2781
2782  005400  021427  001154  5$:     CMP     (R4),#620.      ;ON CYLDER 620 ?
2783  005404  001002                  BNE     6$              ;NO
```

```
                        GET VALUE FOR SOFTWARE SWITCH REGISTER

2784  005406  052703  000002            BIS     #BIT1,R3        ;SET ERROR FLAG
2785  005412  005703                6$: TST     R3              ;ANY ERROR ?
2786  005414  001002                    BNE     7$              ;YES
2787  005416  062705  000002            ADD     #2,R5           ;ADJUST FOR NORMAL RETURN
2788  005422  012603                7$: MOV     (SP)+,R3        ;RESTORE R3 THROUGH R1
2789  005424  012602                    MOV     (SP)+,R2
2790  005426  012601                    MOV     (SP)+,R1
2791  005430  000205                    RTS     R5              ;EXIT
2792
2793
2794  005432  104401  001205      ENDX1:  TYPE    ,$CRLF  ;
2795  005436  104401  036424              TYPE    ,MESG21         ;DRIVE NOT ONLINE OR NOT ASSIGNED
2796  005442  000000                      HALT            ;TEMP HALT FOR DEBUG
2797  005444  000137  000200              JMP     @#200   ;RESTART IF DESIRED
2798                                ;
2799                                ;
2800                                ;
2801                                ;THE FOLLOWING CODING FOR TEST 1 THROUGH TEST 4
2802                                ;PARAMETER IN TST 1
2803                                ;$CDW1  :       ADDRESS OF LOGICAL DRIVE BLOCK
2804                                ;DRIVE  :       PHYSICAL DRIVE #
2805                                ;$DEVM  :       LOGICAL DRIVE # 0-17
2806                                ;ASSGN1 :       ASSIGN LOGICAL DRIVE BIT MAP
2807                                ;ASNLST :       ASSIGNED LOGICAL DRIVE MAP INDICATOR
2808                                ;$CDW2: SYS-NAME
2809                                ;
2810                                ;$CDW2,DRIVE ARE ONLY CHANGED IN TST1 DURING PASS 1
2811                                ;$DEVM,ASSGN1,$CDW1 ARE CHANGED BY THE TEST 4
2812
2813
2814
2815
2816
2817
2818                                ;TST 1:DIRECT OPERATOR TO MOUNT AND LOAD PACKS
2819                                ;
2820                                ;
2821
2822
2823                                ;;*****************************************************************
2824  005450  000004            TST1:   SCOPE
2825  005452  012737  000001  001174    MOV     #1,$TIMES       ;;DO 1 ITERATION
2826  005460  012737  000001  001336    MOV     #1,TSTNM        ;LOAD THE TEST NUMBER
2827  005466  012706  001100            MOV     #STACK,SP       ;INITIALIZE THE STACK POINT
2828  005472  023737  001270  002600    CMP     $CDW1,BLKADR    ;LOGICAL DRIVE 0
2829  005500  001524                    BEQ     3$              ;THEN EXIT
2830  005502  013701  001270            MOV     $CDW1,R1        ;R1=LOGICAL DRIVE BLOCK ADDRESS
2831  005506  126137  000014  001272    CMPB    $SYSNM(R1),$CDW2 ;STILL ON THE SAME SYSTEM ?
2832  005514  001426                    BEQ     1$              ;THEN ,EXIT
2833  005516  116137  000014  001272    MOVB    $SYSNM(R1),$CDW2 ;LOAD THE NEW SUB SYSTEM NAME
2834  005524  012777  013256  015346    MOV     #IDLEX,@RMVEC   ;RESET THE INTERRUPT VECTOR
2835  005532  005077  015344            CLR     @RMVEC+2        ;CLEAR THE INTERRUPT LEVEL
2836  005536  104401  001205            TYPE    ,$CRLF
2837  005542  104401  035053            TYPE    ,MESG3          ;** STARTING PASS 1 **
2838  005546  104401  001205            TYPE    ,$CRLF
2839  005552  104401  035752            TYPE    ,MESG6          ;ON--
```

```
2840  005556  104401  035030                  TYPE    ,MESG1          ;SUB-SYSTEM
2841  005562  104401  001272                  TYPE    ,$CDW2          ;A TO H
2842  005566  104401  036121                  TYPE    ,LINSP
2843  005572  111137  001222          1$:     MOVB    (R1),DRIVE      ;LOAD THE PHYSICAL DRIVE #
2844  005576  104401  001205                  TYPE    ,$CRLF
2845  005602  104401  035101                  TYPE    ,MESG4          ;MOUNT PACK ON THE DRIVE
2846  005606  104401  001272                  TYPE    ,$CDW2          ;SYS-NAME
2847  005612  113746  001222                  MOVB    DRIVE,-(SP)     ;THE PHYSICAL DRIVE #
2848  005616  104403                          TYPOS
2849  005620     002                          .BYTE   2
2850  005621     000                          .BYTE   0
2851  005622  104401  036121                  TYPE    ,LINSP
2852  005626  104401  035127                  TYPE    ,MESG8          ;AND LOAD
2853  005632  104401  001205                  TYPE    ,$CRLF
2854  005636  104401  035142                  TYPE    ,MESG9
2855  005642  104411                  2$:     RDLIN
2856  005644  012605                          MOV     (SP)+,R5                ;LOCATE THE READIN LINE
2857  005646  122725  000122                  CMPB    #'R,(R5)+               ;CHECK IF O.P. READY ?
2858  005652  001373                          BNE     2$              ;IF NOT
2859  005654  105715                          TSTB    (R5)            ;NOT CORRECT INPUT LINE FORMAT
2860  005656  001371                          BNE     2$              ;BRANCH IF NOT
2861  005660  113701  001272                  MOVB    $CDW2,R1        ;LOCATE THE SYSTEM ADDRESS TABLE
2862  005664  042701  177760                  BIC     #177760,R1      ;LEFT ON 4 BITS
2863  005670  005301                          DEC     R1              ;ADDJUST THE INDEX VALUE
2864  005672  006301                          ASL     R1              ;FOUR WORD INDEX VALUE
2965  005674  006301                          ASL     R1
2866  005676  016137  002004  023076          MOV     SYSADR(R1),RMADR  ;LOAD THE SYSTEM ADDRESS
2867  005704  016137  002006  023100          MOV     SYSADR+2(R1),RMVEC ;LOAD THE SYSTEM INTERRUPT VECTOR
2868  005712  004737  023114                  JSR     PC,RMINIT       ;INITIALIZE THE SYSTEM
2869  005716  012737  177777  023036          MOV     #-1,SAVEFG      ;SAVE ALL RH/RM REGISTER
2870  005724  012737  177777  023040          MOV     #-1,SEEKFG      ;DON'T DO ANY IMPLY SEEK
2871  005732  013700  001222                  MOV     DRIVE,R0        ;R0=PHYSICAL DRIVE #
2872  005736  105760  022750                  TSTB    DRVSTA(R0)      ;ON-LINE ?
2873  005742  003404                          BLE     4$              ;BRANCH IF NOT
2874  005744  105760  022760                  TSTB    DRVTYP(R0)      ;AN RM03
2875  005750  003401                          BLE     4$              ;BRANCH IF NOT AN RM03
2876  005752  000403                  3$:     BR      TST2            ;ALL DONE PROCEED TO TEST 2
2877  005754  104401  035201          4$:     TYPE    ,MESG10         ;DRIVE NOT READY
2878  005760  000633                          BR      TST1            ;TRY AGAIN
2879
2880
2881                                  ;TEST 2
2882                                  ;BASIC READ AND WRITE TEST
2883                                  ;ALL LOGICAL DRIVE ACCESS CYLINDER 620
2884                                  ;AND SECTOR ADDRESS IS COORESPONDING TO THE LOGICAL DRIVE #
2885                                  ;EACH LOGICAL DRIVE PERFORM WRITE AND WRITE CHECK ON ALL FIVE SURFACES(TRK0 - TRK4)
2886
2887
2888
2889                                  ;;*****************************************************************
2890  005762  000004                  TST2:   SCOPE
2891  005764  012737  000001  001174          MOV     #1,$TIMES       ;;DO 1 ITERATION
2892  005772  012737  000002  001336          MOV     #2,TSTNM        ;LOAD TEST NUMBER
2893  006000  012706  001100                  MOV     #STACK,SP       ;INITIAL THE STACK POINTER
2894  006004  012700  037146                  MOV     #FMTDPB,R0      ;DPB BLOCK ADDRESS
2895  006010  013701  001270                  MOV     $CDW1,R1        ;ADDRESS OF THE LOGICAL DRIVE BLOCK
```

```
2896  006014  113710  001222                  MOVB    DRIVE,(R0)          ;PHYSICAL DRIVE #
2897  006020  013760  001266  000010          MOV     $DEVM,$SEC(R0)      ;LOAD THE SECTOR #,FROM THE LOGICAL DRIVE NUMBER.
2898  006026  012760  001154  000012          MOV     #620.,$CYL(R0)      ;LOAD CYLINDER NUMBER
2899  006034  012760  177400  000004          MOV     #-256.,$WRDM(R0)    ;LOAD NEG WORD COUNT
2900  006042  012760  037262  000006          MOV     #ENDPGM,$BUF(R0)    ;LOAD BUFFER ADDRESS
2901  006050  012760  037166  000014          MOV     #RM.REG,14(R0)      ;ADDRESS TO SAVE ALL RH/RM03 REG'S
2902  006056  105060  000011                  CLRB    $TRK(R0)            ;START FROM TRACK 0
2903  006062  112760  000161  000002          MOVB    #WRTDAT,$COMND(R0)      ;WRITE DATA COMMAND
2904  006070  004737  014546                  JSR     PC,FILBUF           ;FILL THE BUFFER WITH STANDARD PATTERN
2905  006074  004037  023620          2$:     JSR     R0,RM03             ;CALL THE DRIVER
2906  006100  037146                          FMTDPB
2907  006102  000774                          BR      2$                  ;BRANCH IF NOT QUEUE SUCCESSFULLY
2908  006104  005737  037164          3$:     TST     FMTDPB+16
2909  006110  001775                          BEQ     3$                  ;BRANCH IF NOT DONE
2910  006112  012700  037146                  MOV     #FMTDPB,R0          ;R0=FMTDPB ADDRESS
2911  006116  004737  013264                  JSR     PC,PROCESS          ;CHECK THE TERMINATION
2912  006122  112760  000151  000002          MOVB    #WCKD,$COMND(R0)    ;CHANGE TO THE WRITE CHECK DATA COMMAND
2913  006130  004037  023620          4$:     JSR     R0,RM03             ;CLLL THE DRIVER
2914  006134  037146                          FMTDPB
2915  006136  000774                          BR      4$                  ;BRANCH IF NOT QUEU SUCCESSFULLY
2916  006140  005737  037164          5$:     TST     FMTDPB+16           ;DONE ?
2917  006144  001775                          BEQ     5$                  ;BRANCH IF NOT DONE
2918  006146  012700  037146                  MOV     #FMTDPB,R0          ;
2919  006152  004737  013264                  JSR     PC,PROCES           ;PROCESS IF ANY ERROR HAPPENS ?
2920  006156  112760  000161  000002          MOVB    #WRTDAT,$COMND(R0)  ;RESET TO WRITE DATA COMMAND
2921  006164  105260  000011                  INCB    $TRK(R0)            ;INCREMENT TO THE NEXT TRACK
2922  006170  122760  000005  000011          CMPB    #5.,$TRK(R0)        ;ALL TRACKS DONE ?
2923  006176  101336                          BHI     2$                  ;NO
2924
2925
2926
2927                                  ;TEST 3
2928                                  ;WRITE 7 ZONES FOR WRITE TEST IN PASS 2
2929                                  ;WRITE 6 ZONES FOR READ TEST IN PASS2
2930                                  ;$DEVM  :       LOGICAL DRIVE #
2931                                  ;$CDW1  :       ADDRESS OF LOGICAL DRIVE HISTORY BLOCK FILE
2932                                  ;$CDW2  :       SYS NAME
2933                                  ;DRIVE  :       PHYSICAL DRIVE # OF THIS LOGICAL DRIVE
2934                                  ;PACK   :       ENTRY OF TABLE-D
2935                                  ;CMCNT  :       ZONE COUNT
2936                                  ;CMSEC  :       DELTA CYLINDER COUNT
2937                                  ;R4     :       ENTRY POINTER OF TABLE-D,CANNOT BE DESTORIED.
2938                                  ;R0     :       ADDRESS OF FMTDPB
2939
2940
2941
2942                                  ;;*****************************************************************
2943  006200  000004                  TST3:   SCOPE
2944  006202  012737  000001  001174          MOV     #1,$TIMES           ;;DO 1 ITERATION
2945  006210  012737  000003  001336          MOV     #3,TSTNM            ;LOAD THE TEST NUMBER
2946  006216  012706  001100                  MOV     #STACK,SP           ;INITIAL THE STACK POINTER
2947  006222  012704  001376                  MOV     #LOGO,R4            ;ADDRESS OF TABLE-D
2948  006226  013700  001266                  MOV     $DEVM,R0            ;LOGICAL DRIVE #
2949  006232  001404                          BEQ     2$                  ;BRANCH IF LOGICAL DRIVE 0
2950  006234  062704  000020          1$:     ADD     #16.,R4             ;EACH LOGICAL DRIVE TAKES 16 BYTES IN THE TABLE
2951  006240  005300                          DEC     R0                  ;DECREMENT THE DRIVE # COUNT
```

```
2952  006242  001374                        BNE     1$              ;BRANCH ,UNTIL THE ENTRY IS LOCATED
2953  006244  010437  001322        2$:     MOV     R4,PACK         ;SAVE THE TABLE-D ENTRY IN PACK
2954                                                                ;SET UP THE FMTDPB BLOCK
2955  006250  012700  037146                MOV     #FMTDPB,R0      ;ADDRESS OF FMTDPB
2956  006254  113710  001222                MOVB    DRIVE,(R0)      ;PHYSICAL DRIVE NUMBER
2957  006260  112760  000161  000002        MOVB    #WRTDAT,$COMND(R0)      ;WRITE DATA COMMAND
2958  006266  012760  037262  000006        MOV     #ENDPGM,$BUF(R0) ;BUFFER ADDRESS
2959  006274  012760  037166  000014        MOV     #RM.REG,14(R0)  ;ADDRESS TO SAVE ALL RH/RM03 REG'S
2960  006302  012760  177400  000004        MOV     #-256.,$WRDM(R0)        ;NEG WORD COUNT
2961  006310  013701  001266                MOV     $DEVM,R1        ;LOGICAL DRIVE #
2962  006314  006301                        ASL     R1              ;WORD INDEX
2963  006316  016104  002702                MOV     PSEUDO(R1),R4   ;LOAD THE DATA PATTERN
2964  006322  016002  000006                MOV     $BUF(R0),R2     ;BUFFER ADDRESS IN R2
2965  006326  012703  000400                MOV     #256.,R3        ;POS WORD COUNT
2966  006332  010422                3$:     MOV     R4,(R2)+        ;FULL THE BUFFER WITH SIGLE WORD PATTERN
2967  006334  005303                        DEC     R3              ;DECREMENT THE WORD COUNT
2968  006336  001375                        BNE     3$              ;BRANCH,UNTIL IT IS FULL
2969
2970
2971  006340  005046                        CLR     -(SP)           ;DUMY PAIR OF ZONE COUNT
2972  006342  005046                        CLR     -(SP)           ;DUMY STARTING CYLINDER NUMBER
2973  006344  012746  000006                MOV     #6,-(SP)        ;ZONE COUNT
2974  006350  012746  000160                MOV     #112.,-(SP)     ;STARTING CYLINDER NUMBER
2975  006354  012737  000007  001352        MOV     #7,CMCNT        ;ZONE COUNT
2976  006362  005037  001354                CLR     CMCYL           ;STARTING CYLINDER NUMBER
2977  006366  012737  000020  001360  4$:     MOV     #16.,CMSEC      ;DELTA CYLINDER NUMBER
2978  006374  012700  037146                MOV     #FMTDPB,R0      ;R0 DPB ADDRESS
2979  006400  013704  001322                MOV     PACK,R4         ;R4 ENTRY TO TABLE-D
2980  006404  013760  001354  000012  5$:     MOV     CMCYL,$CYL(R0)  ;STARTING CYLINDER
2981  006412  105060  000011                CLRB    $TRK(R0)        ;STARTS FROM TRACK 0
2982  006416  111460  000010          6$:     MOVB    (R4),$SEC(R0)   ;LOAD SECTOR NUMBER FROM TABLE-D
2983  006422  004037  023620          7$:     JSR     R0,RM03         ;CALL THE DRIVER
2984  006426  037146                        FMTDPB
2985  006430  000774                        BR      7$              ;BRANCH IF QUEU IS NOTSUCCESSFUL
2986  006432  005737  037164          8$:     TST     FMTDPB+16       ;DONE ?
2987  006436  001775                        BEQ     8$              ;BRANCH IF NOT
2988  006440  012700  037146                MOV     #FMTDPB,R0
2989  006444  004737  013264                JSR     PC,PROCES       ;PROCESS TO CHECK IF ANY ERROR
2990  006450  062760  000020  000010        ADD     #16.,$SEC(R0)   ;WRITE TWO SECTORS ON ONE CYLINDER
2991  006456  122760  000037  000010        CMPB    #31.,$SEC(R0)   ;ALL DONE-TWO SECTORS
2992  006464  103346                        BHIS    7$              ;BRANCH IF NOT
2993  006466  111460  000010                MOVB    (R4),$SEC(R0)   ;RESTORE SECTOR #
2994  006472  105260  000011                INCB    $TRK(R0)        ;INCREMENT TO NEXT TRACK
2995  006476  122760  000004  000011        CMPB    #4.,$TRK(R0)    ;LAST TRACK ?
2996  006504  103346                        BHIS    7$              ;NO, THEN BRANCH
2997  006506  105060  000011                CLRB    $TRK(R0)        ;RESTORE TO TRACK-0
2998  006512  005260  000012                INC     $CYL(R0)        ;INCREMENT CYLINDER NUMBER
2999  006516  005204                        INC     R4              ;INCREMENT TABLE-D ENTRY
3000  006520  053737  001360                BIS     CMSEC           ;DECREMENT THE DELTA CYLINDER COUNT
3001  006524  001334                        BNE     6$              ;BRANCH IF NOT END OF THIS BLOCK
3002  006526  062760  000160  000012        ADD     #112.,$CYL(R0)  ;INCREMENTTHE CYLINDER NUMBER TO NEXT ZONE
3003  006534  016037  000012  001354        MOV     $CYL(R0),CMCYL  ;INITAIL THE STARTING CYLINDER IN THE BLOCK
3004  006542  005337  001352                DEC     CMCNT   ;DECREMENT THE ZONE COUNT
3005  006546  001307                        BNE     4$              ;LOOPING IF NOT END OF ZONE
3006  006550  012637  001354                MOV     (SP)+,CMCYL     ;LOAD NEW PAIR OF STARTING CYLINDER
3007  006554  012637  001352                MOV     (SP)+,CMCNT     ;AND ZONE COUNT
```

```
3008  006560  005737  001354                  TST     CMCYL           ;NOT END YET ?
3009  006564  001300                          BNE     4$              ;BRANCH IF NOT
3010  006566  005737  001352                  TST     CMCNT           ;BRANCH IF NOT END
3011  006572  001275                          BNE     4$              ;LOOPING BACK
3012
3013
3014                                  ;TEST 4
3015                                  ;UPDATE THE PARAMETERS,$CDW1,$DEVM,ASSGN1
3016                                  ;DIRECT THE OPERATOR TO DISMOUNT PACK AND LOAD TO OTHER DRIVE
3017                                  ;
3018                                  ;$CDW2,DRIVE ARE CHANGED BY TEST ONE ONLY AFTER THE TEST LOOPING TO TEST1
3019
3020
3021                                  ;;*****************************************************************
3022  006574  000004                  TST4:   SCOPE
3023  006576  012737  000001  001174          MOV     #1,$TIMES       ;;DO 1 ITERATION
3024  006604  012737  000004  001336          MOV     #4,TSTNM        ;LOAD THE TES, NUMBER
3025  006612  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
3026  006616  012777  013256  014254          MOV     #IDLEX,@RMVEC   ;RESET THE INTERRUPT VECTOR
3027  006624  005077  014252                  CLR     @RMVEC+2        ;CLEAR THE INTERRUPT LEVEL
3028  006630  104401  001205                  TYPE    ,$CRLF
3029  006634  104401  035323                  TYPE    ,P   312
3030  006640  104401  001272                  TYPE    ,    W2
3031  006644  013746  001222                  MOV     DRIVE,-(SP)
3032  006650  104403                          TYPOS
3033  006652      002                         .BYTE   2
3034  006653      000                         .BYTE   0
3035  006654  104401  036121                  TYPE    ,LINSP
3036  006660  104401  035341                  TYPE    ,MESG13
3037  006664  012701  000001                  MOV     #1,R1
3038  006670  005002                          CLR     R2
3039  006672  020237  001266          1$:     CMP     R2,$DEVM        ;LOCATE THE COORESPONDING BIT MAP
3040  006676  001404                          BEQ     2$              ;BRANCH IF LOCATED
3041  006700  000241                          CLC                     ;LOCATE NEXT DRIVE
3042  006702  006101                          ROL     R1              ;
3043  006704  005202                          INC     R2              ;NEXT DRIVE #
3044  006706  000771                          BR      1$              ;LOCATE THE BIT MAP
3045  006710  040137  002000          2$:     BIC     R1,ASSGN1       ;DEASSIGN THE LOGICAL DRIVE FOR PASS 1
3046  006714  001410                          BEQ     4$              ;NO MORE DRIVES
3047  006716  005202                  3$:     INC     R2              ;GET NEXT LOGICAL DRIVE #
3048  006720  006302                          ASL     R2              ;WORD INDEX
3049  006722  016237  002600  001270          MOV     BLKADR(R2),$CDW1        ;LOAD THE NEW DPB ADDRESS
3050  006730  006202                          ASR     R2              ;RESTORE R2
3051  006732  010237  001266                  MOV     R2,$DEVM        ;LOAD THE NEW LOCICAL DRIVE #
3052  006736  104411                  4$:     RDLIN                   ;WAIT UNTIL IT IS DONE
3053  006740  012605                          MOV     (SP)+,R5        ;LOCATE THE INPUT LINE
3054  006742  122725  000122                  CMPB    #'R,(R5)+       ;
3055  006746  001373                          BNE     4$              ;
3056  006750  105715                          TSTB    (R5)            ;TERMINATOR ?
3057  006752  001371                          BNE     4$              ;BRANCH IF NOT
3058  006754  005737  002000                  TST     ASSGN1          ;OTHER DRIVES ?
3059  006760  001002                          BNE     5$              ;BRANCH IF MORE DRIVES IN TEST
3060  006762  000137  006772                  JMP     XPASS2          ;BRANCH TO PASS 2
3061  006766  000137  005450          5$:     JMP     TST1            ;JUMP TO TEST 1
3062
3063
```

```
3064                                    ;XPASS2         INITILIZE FOR PASS 2 TEST
3065                                    ;$CDW1  :       ADDRESS OF THE CURRENT LOGICAL DRIVE HISTORY FILE
3066                                    ;$CDW2  :       SYSTEM NAME A THROUGH H
3067                                    ;$DEVM  :       CURRENT LOGICAL DRIVE # 0 TO 15.
3068                                    ;ASSGN2 :       ASSIGNED LOGICAL DRIVE FOR PASS 2
3069                                    ;ASNLST :       ASSIGNED LOGICAL DRIVE
3070                                    ;DRIVE  :       PHYSICAL DRIVE # OF CURRENT RH/RM SYSTEM
3071
3072
3073
3074   006772  005737  002002          XPASS2: TST     ASSGN2          ;ANYTHING IN TEST FOR PASS 2
3075   006776  001002                          BNE     1$              ;YES,THEN GO ON
3076   007000  000137  012762                  JMP     XEND2           ;JUMP TO END OF PASS 2
3077   007004  005037  001266          1$:     CLR     $DEVM           ;START FROM LOGICAL DRIVE 0
3078   007010  013737  002600  001270          MOV     BLKADR,$CDW1    ;ADDRESS OF LOGICAL BLOCK DRIVE 0
3079   007016  112737  000101  001272          MOVB    #'A,$CDW2       ;LOAD SYSTEM NAME "A"
3080   007024  013737  002004  023076          MOV     SYSADR,RMADR    ;LOAD SYSTEM-A ADDRESS TO DRIVER
3081   007032  013737  002006  023100          MOV     SYSADR+2,RMVEC  ;LOAD SYSTEM-A VECTOR TO DRIVER
3082   007040  013701  001270                  MOV     $CDW1,R1        ;R1=ADDRESS OF LOGICAL BLOCK
3083   007044  104401  001205                  TYPE    ,$CRLF
3084                                  ;         TYPE    ,MESG:4 ;STARTING PASS 2
3085   007050  005037  001374                  CLR     FAULT           ;RESSET THE NOT COMPATIBLE FLAG
3086   007054  000400                          BR      TST5            ;BRANCH TO TEST 5
3087
3088
3089
3090                                    ;$CDW1  :       ADDRESS OF CURRENT LOGICAL DRIVE BLOCK,ONLY CHANGED BY TST9
3091                                    ;$CDW2  :       SYSTEM-NAME             ,ONLY CHANGED BY TEST 5
3092                                    ;$DEVM  :       CURRENT LOGICAL DRIVE #
3093                                    ;ASSGN2 :       ASSIGNED LOGICAL DRIVE'S BIT MAP FOR PASS 2
3094                                    ;DRIVE  :       PHYSICAL DRIVE #
3095                                    ;ASNLST :       ASSIGNED LOGICAL DRIVE IN THE TEST
3096                                    ;IN TEST 5 DIRECT OPERATOR TO CHANGE,LOAD THE PACK
3097                                    ;
3098                                    ;
3099
3100                                    ;;*****************************************************************
3101   007056  000004                  TST5:   SCOPE
3102   007060  012737  000001  001174          MOV     #1,$TIMES       ;;DO 1 ITERATION
3103   007066  012737  000005  001336          MOV     #5,TSTNM        ;LOAD THE TEST NUMBER
3104   007074  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
3105   007100  012777  013256  013772          MOV     #IDLEX,@RMVEC   ;RESET THE INTERRUPT VECTOR
3106   007106  005077  013770                  CLR     @RMVEC+2        ;CLEAR THE INTERRUPT LEVEL
3107   007112  013701  001270                  MOV     $CDW1,R1        ;ADDRESS OF THE HISTORY FILE
3108   007116  126137  000014  001272          CMPB    $SYSNM(R1),$CDW2        ;ON THE SAME SUB-SYSTEM
3109   007124  001420                          BEQ     1$              ;THEN DON'T UPDATE SYSTEM ADDRESS
3110   007126  116137  000014  001272          MOVB    $SYSNM(R1),$CDW2
3111   007134  013700  001272                  MOV     $CDW2,R0        ;LOCATE SYSTEM ADDRESS TABLE
3112   007140  005300                          DEC     R0              ;ADJUST FOR INDEX FORM 0
3113   007142  042700  177760                  BIC     #177760,R0      ;LEFT ON FOUR BITS
3114   007146  006300                          ASL     R0
3115   007150  006300                          ASL     R0              ;INDEX FOR TWO WORD
3116   007152  016037  002004  023076          MOV     SYSADR(R0),RMADR ;SYSTEM ADDRESS
3117   007160  016037  002006  023100          MOV     SYSADR+2(R0),RMVEC ;SYSTEM INTERRUP VECTOR
3118                                  ;1$:    MOV     #IDLEX,@RMVEC   ;RESET THE INTERRUPT VECTOR
3119                                  ;       CLR     @RMVEC+2        ;CLEAR THE INTERRUPT LEVEL
```

```
3120  007166  104401  001205          1$:     TYPE    ,$CRLF
3121  007172  104401  035414                  TYPE    ,MESG14         ;START THE PASS 2
3122  007176  104401  001205                  TYPE    ,$CRLF          ;
3123  007202  104401  035101                  TYPE    ,MESG4  ;MOUNT PACK ON DRIVE
3124  007206  104401  001272                  TYPE    ,$CDW2  ;SYSTEM NAME
3125  007212  111137  001222                  MOVB    (R1),DRIVE      ;LOCATE THE PHYSICAL DRIVE #
3126  007216  111146                          MOVB    (R1),-(SP)
3127  007220  104403                          TYPOS
3128  007222     002                          .BYTE   2
3129  007223     000                          .BYTE   0
3130  007224  104401  036121                  TYPE    ,LINSP
3131  007230  104401  035127                  TYPE    ,MESG8  ;AND LOAD
3132  007234  104401  001205                  TYPE    ,$CRLF
3133  007240  104401  035142                  TYPE    ,MESG9  ;TYPE R<CR> WHEN DRIVE IS READY
3134  007244  104411                  2$:     RDLIN
3135  007246  012602                          MOV     (SP)+,R2        ;LOCATE THE READ IN LINE
3136  007250  122722  000122                  CMPB    #'R,(R2)+       ;FIRST CHAR IS A "R"
3137  007254  001373                          BNE     2$              ;BRANCH IF NOT
3138  007256  105712                          TSTB    (R2)            ;FOLLOWED BY A TERMINATOR
3139  007260  001371                          BNE     2$              ;BRANCH IF NOT
3140  007262  004737  023114                  JSR     PC,RMINIT
3141  007266  012737  177777  023036          MOV     #-1,SAVEFG      ;SAVE ALL RH/RM REGISTERS
3142  007274  012737  177777  023040          MOV     #-1,SEEKFG      ;DON'T DO IMPLY SEEK
3143  007302  013700  001222                  MOV     DRIVE,R0        ;LOAD THE PHYSICAL DRIVE NUMBER
3144  007306  105760  022750                  TSTB    DRVSTA(R0)      ;DRIVE EXISTS AND ON-LINE ?
3145  007312  003405                          BLE     3$              ;BRANCH IF NOT
3146  007314  105760  022760                  TSTB    DRVTYP(R0)      ;AN RM03 ?
3147  007320  003402                          BLE     3$              ;NOT AN RM03
3148  007322  000137  007340                  JMP     TST6            ;BRANCH IF IT IS,(NEXT TEST)
3149  007326  104401  001205          3$:     TYPE    ,$CRLF
3150  007332  104401  035201                  TYPE    ,MESG10 ;DRIVE IS NOT READY
3151  007336  000647                          BR      TST5            ;TRY AGAIN
3152
3153
3154
3155                                  ;DRIVE  :       PHYSICAL DRIVE NUMBER
3156                                  ;$CDW1  :       DPB BLOCK OF THIS LOGICAL DRIVE
3157                                  ;$DEVM  :       LOGICAL DRIVE #
3158                                  ;$CDW2  :       SUB-SYSTEM NAME
3159                                  ;RMADR  :       SUB-SYSTEM BASE REGISTER ADDRESS
3160                                  ;RMVEC  :       SUB-SYSTEM INTERRUPT VECTOR
3161
3162                                  ;THE ABOVE PARAMETERS CANNOT BE MODIFIED BY THIS TEST (TST6)
3163                                  ;THE FOLLOWING REG'S ARE ASSIGNED AS:
3164                                  ;R0     :       ADDRESS OF DPB FEEDED INTO DRIVER HANDLER=FMTDPB
3165                                  ;R1     :       ADDRESS OF THE HISTORY FILE BLOCK OF THE LOGICAL DRIVE=$CDW1
3166
3167
3168                                  ;IN TEST 6,EACH LOGICAL DRIVE WRITES 7 CYLINDERS ON EACH SURFACE WITH
3169                                  ;AN UNIQUE DATA PATTERN.  THESE 7 CYLINDERS HAS BEEN
3170                                  ;WRITTEN BY OTHER DRIVES IN PASS 1.
3171                                  ;THEN,THIS LOGICAL DRIVE EXECUTES "WRITE CHECK DATA" TO SEE IF
3172                                  ;THIS LOGICAL DRIVE CAN OVER-WRITE ALL DATA WRITTEN BY OTHER
3173                                  ;DRIVES.
3174
3175                                  ;THESE 7 CYLINDERS ARE SPECIFIED AS $DEVM,$DEVM+128,$DEVM+256
```

```
                                        ;$DEVM+384,$DEVM+512,$DEVM+640 AND $DEVM+768.
                                        ;
                                        ;THE OVER-WRITE TEST IS PERFORMED WITH OFFSETS IN BOTH DIRECTIONS.
                                        ;
                                        ;;*****************************************************************
007340  000004                  TST6:   SCOPE
007342  012737  000001  001174          MOV     #1,$TIMES       ;;DO 1 ITERATION
007350  012737  000006  001336          MOV     #6,TSTNM        ;LOAD TEST NUMBER
007356  012706  001100                  MOV     #STACK,SP       ;INITIAL THE STACK POINT
007362  012703  002046                  MOV     #OVWNO,R3       ;1 ST ADDRESS TO CLEAR
007366  012704  002306                  MOV     #RDNO,R4        ;LAST ADDRESS+2
007372  005023                  1$:     CLR     (R3)+           ;RESET ALL SCORE BOARD
007374  020403                          CMP     R4,R3           ;ALL LOCATIONS ARE CLEARED ?
007376  101375                          BHI     1$              ;BRANCH IF NOT
007400  012700  037146                  MOV     #FMTDPB,R0      ;SET UP PARAMETER BLOCK
007404  013701  001270                  MOV     $CDW1,R1        ;HISTORY FILE BLOCK
007410  113710  001222                  MOVB    DRIVE,(R0)      ;LOAD THE PHYSICAL DRIVE #
007414  012760  160000  000004          MOV     #-8192.,$WRDM(R0) ;LOAD THE WORD COUNT
007422  005060  000010                  CLR     $SEC(R0)        ;START FROM  TRK0,SEC0
007426  112760  000161  000002          MOVB    #WRTDAT,$COMND(R0)  ;WRITE DATA COMMAND
007434  013760  001266  000012          MOV     $DEVM,$CYL(R0)  ;LOAD THE STARTING CYLINDER
007442  012760  037262  000006          MOV     #ENDPGM,$BUF(R0) ;LOAD THE BUFFER ADDRESS
007450  012760  037166  000014          MOV     #RM.REG,14(R0)  ;REG'S SAVE ADDRESS
007456  013702  001266                  MOV     $DEVM,R2        ;LOCATE THE DATA PATTERN
007462  006302                          ASL     R2              ;WORD INDEX
007464  016202  002702                  MOV     PSEUDO(R2),R2   ;LOAD R2 THE DATA PATTERN POINTED
                                                                ;BY ITSELF
007470  016003  000006                  MOV     $BUF(R0),R3     ;BUFFER ADDRESS
007474  012704  020000                  MOV     #8192.,R4       ;WORD COUNT
007500  010223                  2$:     MOV     R2,(R3)+        ;FILL THE BUFFER
007502  005304                          DEC     R4              ;DECREMENT WORD CTR
007504  001375                          BNE     2$              ;BRANCH IF NOT DONE
                                        ;                       START TO WRITE ONE CYLINDER
                                        ;                       ON EACH SURFACE OF EACH WRITE
                                        ;                       CURRENT ZONE ( 7 ZONES ON ONE PACK)
007506  004037  023620          3$:     JSR     R0,RM03         ;CALL THE DRIVER
007512  037146                          FMTDPB                  ;PARAMETER BLOCK
007514  000774                          BR      3$              ;BRANCH IF QUEUE FAIL
007516  005737  037164          4$:     TST     FMTDPB+16       ;WRITE COMMAND DONE ?
007522  001775                          BEQ     4$              ;NO,THEN WAIT
007524  012700  037146                  MOV     #FMTDPB,R0      ;PROCESS IF ANY ERROR
007530  004737  013264                  JSR     PC,PROCES
007534  105260  000011                  INCB    $TRK(R0)        ;NEXT TRACK
007540  122760  000005  000011          CMPB    #5,$TRK(R0)     ;LAST TRACK IS DONE ?
007546  101357                          BHI     3$              ;NO,THEN BRANCH
007550  105060  000011                  CLRB    $TRK(R0)        ;RESET TRACK NUMBER
007554  062760  000200  000012          ADD     #128.,$CYL(R0)  ;MOVE TO NEXT ZONE
007562  022760  001417  000012          CMP     #783.,$CYL(R0)  ;LAST ZONE IS DONE ?
007570  103346                          BHIS    3$              ;BRANCH IF NOT
                                        ;
                                        ;
                                        ;                       ;RESET THE FMTDPB BLOCK AND
                                        ;                       ;EXECUTE WRITE-CHECK COMMAND
                                        ;                       ;TO DETECT ANY COMPATIBLE PROBLEM
```

```
                                                        ;THE FOLLOWING SUBROUTINE ARE CALLED
                                                        ;SCORE,OFFST,MAKEUP.
                                                        ;
007572  005037  002640            CLR     OFFCOD        ;SET NEGATIVE OFFSET DIRECTION FLAG
007576  012700  037146      LOOP1: MOV     #FMTDPB,R0    ;R0=FMTDPB ADDRESS
007602  005060  000010            CLR     $SEC(R0)      ;START FROM SECTOR 0,SURFACE 0
007606  013760  001266  000012    MOV     $DEVM,$CYL(R0) ;STARTING CYLINDER NUMBER
                                                        ;TOTAL 7 CYLINDERS ON ONE SURFACE
007614  112760  000151  000002 LOOP2: MOVB  #WCKD,$COMND(R0)      ;WRITE-CHECK-DATA COMMAND
007622  012760  037262  000006    MOV     #ENDPGM,$BUF(R0)       ;RESET BUFFER ADDRESS
007630  012760  037166  000014    MOV     #RM.REG,14(R0) ;ADDRESS TO SAVE RH/RM03 REG'S
007636  004037  023620      1$:   JSR     R0,RM03       ;CALL THE DRIVER
007642  037146                    FMTDPB                ;PARAMETER BLOCK ADDRESS
007644  000774                    BR      1$            ;BRANCH IF QUEUE FAILURE
007646  005737  037164      2$:   TST     FMTDPB+16     ;TEST IF COMMAND IS DONE ?
007652  001775                    BEQ     2$            ;BRANCH , IF NOT
007654  012700  037146            MOV     #FMTDPB,R0    ;LOAD THE PARAMETER BLOCK ADDRESS
007660  004737  013264            JSR     PC,PROCES     ;REPORT IF ANY ERROR
007664  004737  010732            JSR     PC,LABAD      ;LOCATE STARTING AND ENDING SECTORS
007670  005760  000016            TST     16(R0)        ;ANY ERROR ?
007674  100006                    BPL     3$            ;BRANCH IF NONE
007676  023737  001360  001356    CMP     CMSEC,STARSC  ;ON THE SAME SECTOR FOR STARTING AND ENDING
007704  101404                    BLOS    4$            ;DON'T INCREMENT SCORES
007706  005337  001360            DEC     CMSEC         ;DECREMENT THE SECTOR ADDRESS
007712  004737  010512      3$:   JSR     PC,SCORE      ;INCREMENT SCORE
007716  005760  000022      4$:   TST     $RMWC(R0)     ;WORD COUNT = 0 /
007722  001407                    BEQ     5$            ;BRANCH, IF WORD COUNT IS 0
007724  116060  000026  000010    MOVB    $RMDA(R0),$SEC(R0)  ;UPDATE STARTING SECTOR
007732  016060  000022  000004    MOV     $RMWC(R0),$WRDM(R0) ;UPDATE WORD COUNT
007740  000736                    BR      1$            ;TO READ THE REST SECTORS
                                                        ;
                                                        ;
                                                        ;THE FOLLOWING CODING TEST THE
                                                        ;COMPATIBLE PROBLEM IN OFFSET MODE
                                                        ;OFFCOD=0,NEGATIVE ;OFFCOD=1 ,POSITIVE
                                                        ;
007742  012700  037146      5$:   MOV     #FMTDPB,R0    ;RESET THE DPB BLOCK
007746  012760  160000  000004    MOV     #-8192.,$WRDM(R0)  ;FULL TRACK WORD COUNT
007754  105060  000010            CLRB    $SEC(R0)      ;RESET TO SECTOR 0,SURFACE NOT CHANGED
007760  012760  037262  000006    MOV     #ENDPGM,$BUF(R0)       ;BUFFER ADDRESS
007766  012760  037166  000014    MOV     #RM.REG,14(R0) ;ADDRESS TO SAVE ALL RH/RM03 REG'S
007774  004537  010176      6$:   JSR     R5,OFFST      ;CALL OFFSET
010000  000440                    BR      10$           ;BRANCH TO NEXT CYLINDER,IF OFFSET FAILS
010002  004737  010400            JSR     PC,MAKEUP     ;CALL WRITE CHECK IN OFFSET MODE
010006  005737  037164      7$:   TST     FMTDPB+16     ; OFFSET WRITE CHECK IS DONE ?
010012  001775                    BEQ     7$            ;BRANCH IF NOT
                                                        ;
010014  012700  037146            MOV     #FMTDPB,R0    ;LOAD THE DPB ADDRESS
010020  004737  013264            JSR     PC,PROCES     ;REPORT , IF ANY ERROR
010024  004737  010732            JSR     PC,LABAD      ;LOCATE STARTINF AND ENDING SECTORS
010030  005760  000016            TST     16(R0)        ;ANY ERROR ?
010034  100006                    BPL     8$            ;BRANCH, IF NONE
010036  023737  001360  001356    CMP     CMSEC,STARSC  ;STARTING AND ENDING ADDRESSES O.K ?
010044  101404                    BLOS    9$            ;BRANCH IF NOT
010046  005337  001360            DEC     CMSEC         ;EXCLUDING THE BAD SECTOR
010052  004737  010512      8$:   JSR     PC,SCORE      ;UPDATE THE TEST SCORE
```

```
3288  010056  005760  000022          9$:     TST     $RMWC(R0)           ;WORD CTR = 0 ?
3289  010062  001407                          BEQ     10$                 ;IF WORD CTR=0 ,BRANCH TO NEXT OP
3290  010064  116060  000026  000010          MOVB    $RMDA(R0),$SEC(R0) ;UPDATE THE NEW STARTING ADDRESS
3291  010072  016060  000022  000004          MOV     $RMWC(R0),$WRDM(R0) ;UPDATE THE NEW WORD COUNT
3292  010100  000735                          BR      6$
3293  010102  062760  000200  000012  10$:     ADD     #128.,$CYL(R0)  ;ADJUST CYLINDER ADDRESS TO NEXT ZONE
3294  010110  022760  001417  000012          CMP     #783.,$CYL(R0)  ;ALL 7 ZONES HAVE BEEN TESTED ?
3295  010116  103402                          BLO     11$             ;BRANCH, IF ALL DONE
3296  010120  000137  007614                  JMP     LOOP2           ;TO NEXT WRITE CURRENT ZONE
3297  010124  013760  001266  000012  11$:     MOV     $DEVM,$CYL(R0)  ;RESET CYLINDER ADDRESS
3298  010132  105260  000011                  INCB    $TRK(R0)        ;INCREMENT TO NEXT SURFACE
3299  010136  122760  000005  000011          CMPB    #5,$TRK(R0)     ;ALL FIVE SURFACES ARE TESTED ?
3300  010144  101402                          BLOS    12$             ;BRANCH IF ALL DONE
3301  010146  000137  007614                  JMP     LOOP2           ;TO NEXT SURFACE
3302  010152  005737  002640          12$:    TST     OFFCOD          ;FINISHING TEST THE POSITIVE OFFSET ?
3303  010156  001005                          BNE     13$             ;BRANCH IF ALL DONE
3304  010160  012737  000001  002640          MOV     #1,OFFCOD       ;SET POSITIVE OFFSET DIRECTION FLAG
3305  010166  000137  007576                  JMP     LOOP1           ;RESTART LOCATION
3306  010172  000137  011030          13$:    JMP     TST7            ;BRANCH TO NEXT TEST
3307
3308
3309
3310
3311                                          ;OFFST ROUTINE
3312                                          ;OFFSET THE HEAD IN THE DIRECTION TOWARD SPINDLE OR AWAY FROM THE SPINDLE
3313                                          ;OFFCOD = 1      TOWARD SPINDLE (POSITIVE)
3314                                          ;OFFCOD = 0      AWAY FROM SPINDLE (NEGATIVE)
3315                                          ;DRIVE           PHICAL DRIVE NUMBER
3316                                          ;R0             DPB ADDRESS
3317                                          ;RETRY  3 TIMES
3318                                          ;CALLING SEQUENCE
3319                                          ;       JSR     R5,OFFST
3320                                          ;       RET1                    OFFSET FAIL RETURN ADDRESS
3321                                          ;       RET2                    OFFSET SUCCESSFUL RETURN
3322
3323  010176  010146                  OFFST:  MOV     R1,-(SP)                        ;SAVE ALL REGISTERS
3324  010200  010246                          MOV     R2,-(SP)
3325  010202  010346                          MOV     R3,-(SP)
3326  010204  010446                          MOV     R4,-(SP)
3327  010206  013704  023076                  MOV     RMADR,R4        ;RH70/RM BASE ADDRESS
3328  010212  013701  001222                  MOV     DRIVE,R1        ;PHYSICAL DRIVE #,R1 FOR INDEXING
3329  010216  012702  000003                  MOV     #3,R2           ;TRY TO OFFSET 3 TIMES
3330  010222  005003                  1$:     CLR     R3              ;DECREMENT COUNT
3331  010224  010164  000010                  MOV     R1,RMCS2(R4)    ;LOAD DRIVE ADDRESS
3332  010230  012764  000011  000000          MOV     #11,RMCS1(R4)   ;CLEAR DRIVE ,WITH INTERRUPT DISABLE
3333  010236  016064  000012  000034          MOV     $CYL(R0),RMDC(R4) ;LOAD DESIRED CYLINDER ADDRESS
3334  010244  016064  000010  000006          MOV     $SEC(R0),RMDA(R4)       ;LOAD DESIRED ADDRESS
3335  010252  016064  000004  000002          MOV     $WRDM(R0),RMWC(R4)  ;LOAD WORD COUNT
3336  010260  016064  000006  000004          MOV     $BUF(R0),RMBA(R4)       ;LOAD BUFFER ADDRESS
3337  010266  005737  002640                  TST     OFFCOD          ;WHICH DIRECTION ?
3338  010272  001404                          BEQ     2$              ;BRANCH IF NEGATIVE
3339  010274  012764  010200  000032          MOV     #BIT12!BIT7,RMOF(R4)  ;LOAD RMOF WITH FMT AND OFT
3340  010302  000403                          BR      3$              ;
3341  010304  012764  010000  000032  2$:     MOV     #BIT12,RMOF(R4) ;LOAD FMT ONLY
3342  010312  012764  000015  000000  3$:     MOV     #15,RMCS1(R4)   ;ISSUE OFFSET COMMAND
3343  010320  136164  023064  000016  4$:     BITB    ATABIT(R1),RMAS(R4)     ;ATA= 1
```

```
3344  010326  001005                          BNE     5$              ; OFFSET IS DONE
3345  010330  005303                          DEC     R3              ;COUNT = 0
3346  010332  001372                          BNE     4$              ;BRANCH IF NOT
3347  010334  005302                          DEC     R2              ;DECREMENT RETRY COUNT
3348  010336  003413                          BLE     6$              ;ERROR EXIT
3349  010340  000730                          BR      1$              ;RETRY
3350  010342  032764  040000  000012  5$:     BIT     #BIT14,RMDS(R4) ;ANY ERROR DURING OFFSET ?
3351  010350  001324                          BNE     1$              ;BRANCH IF ERROR
3352  010352  032764  000001  000012          BIT     #BIT0,RMDS(R4)  ;OFSET MODE BIT SET ?
3353  010360  001720                          BEQ     1$              ;BRANCH IF NOT
3354  010362  062705  000002                  ADD     #2,R5           ;ADJUST RETURN ADDRESS
3355  010366  012604                  6$:     MOV     (SP)+,R4        ;RESTORE REG
3356  010370  012603                          MOV     (SP)+,R3
3357  010372  012602                          MOV     (SP)+,R2
3358  010374  012601                          MOV     (SP)+,R1
3359  010376  000205                          RTS     R5              ;EXIT
3360
3361
3362                                  ;MAKEUP ROUTINE
3363                                  ;THIS ROUTINE ISSUES A WRITE CHECK OR READ  COMMAND TO THE SELECTED DRIVE
3364                                  ;IN OFFSET MODE.
3365                                  ;AND SET UP THE FOLLOWING PARAMETERS
3366                                  ;
3367                                  ; DTUW = PHYSICAL DRIVE NUMBER
3368                                  ; TRNSWT  =  FMTDPB
3369                                  ; DRVACT (DRIVE)  = 1
3370                                  ;TIMER (DRIVE)  = 1 SECOND
3371                                  ;
3372                                  ;CALLING SEQ
3373                                  ;
3374                                  ;       JSR     PC,MAKEUP
3375                                  ;       RET
3376                                  ;MAIN PURPOSE OF THIS ROUTINE,TO EXECUTE A COMMAND WHILE ASSURE THAT
3377                                  ;THIS READ OR WRITE-CHECK COMMAND BEING EXECUTED IN OFFSET MODE.
3378                                  ;ROUTINES USED TD,SC,STO,( IN DRIVE HANDLER )
3379                                  ;
3380
3381
3382  010400  010146                  MAKEUP: MOV     R1,-(SP)
3383  010402  010246                          MOV     R2,-(SP)
3384  010404  010446                          MOV     R4,-(SP)
3385  010406  012702  000151                  MOV     #WCKD,R2        ;WRITE CHECK DATA IN TEST 6
3386  010412  022737  000010  001336          CMP     #10,TSTNM       ;ON TEST 8 ?
3387  010420  001002                          BNE     1$              ;BRANCH IF NOT
3388  010422  012702  000171                  MOV     #RDDAT,R2       ;READ DATA COMMAND IN TEST 8
3389  010426  005037  037164          1$:     CLR     FMTDPB+16       ;CLEAR THE STATUS WORD
3390  010432  013737  001222  023062          MOV     DRIVE,DTUW      ;ACTIEV DRIVE NUMBER
3391  010440  012737  037146  023010          MOV     #FMTDPB,TRNSWT  ;TRANSFER UNDERWAY FLAG
3392  010446  013701  001222                  MOV     DRIVE,R1        ;DRIVE NUMBER
3393  010452  013704  023076                  MOV     RMADR,R4        ;RH/RM BASE ADDRESS
3394  010456  112761  000001  022740          MOVB    #1,DRVACT(R1)   ;ACTIVE DRIVE FLAG
3395  010464  006301                          ASL     R1              ;WORD INDEX
3396  010466  012761  001750  023042          MOV     #1000.,TIMER(R1) ; ONE SECOND TIMER
3397  010474  006201                          ASR     R1
3398  010476  010264  000000                  MOV     R2,RMCSi(R4)    ;ISSURE WRITE CHECK OR READ COMMAND
3399  010502  012604                          MOV     (SP)+,R4        ;RESTORE REG 4, 1
```

```
3400  010504  012602                        MOV     (SP)+,R2
3401  010506  012601                        MOV     (SP)+,R1
3402  010510  000207                        RTS     PC              ;EXIT
3403
3404
3405                                  ;SCORE ROUTINE
3406                                  ;ROUTINE TO UPDATE THE TEST SCORE
3407                                  ;(TABLEX):ADDRESS OF CRRENT SCORE BOARD
3408                                  ;$DEVM: LOGICAL DRIVE #
3409                                  ;DRIVE: PHYSICAL DRIVE NUMBER
3410                                  ;CMSEC: END SECTOR ADDRESS
3411                                  ;STARSC: START SECTOR ADDRESS
3412                                  ;CALL SEQ
3413                                  ;
3414                                  ;       JSR     PC,SCORE
3415                                  ;       RET
3416
3417
3418  010512                          SCORE:
3419  010512  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3420  010514  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3421  010516  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3422  010520  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3423  010522  023737  001360  001356        CMP     CMSEC,STARSC    ;CORRECT START AND STOP ADDRESSES
3424  010530  003473                        BLE     9$              ;BRANCH IF NOT
3425  010532  022737  000010  001336        CMP     #10,TSTNM       ;ON TEST 8
3426  010540  001011                        BNE     2$              ;BRANCH IF NOT (MUST BE TEST 6)
3427  010542  005737  002640                TST     OFFCOD          ;NEGATIVE OFFSET ?
3428  010546  001403                        BEQ     1$              ;BRANCH IF NEGATIVE OFFSET
3429  010550  012703  002426                MOV     #RDPO,R3        ;SCORE BOARD ADDRESS
3430  010554  000413                        BR      4$
3431  010556  012703  002306          1$:   MOV     #RDNO,R3        ;SCORE BOARD ADDRESS
3432  010562  000410                        BR      4$
3433  010564  005737  002640          2$:   TST     OFFCOD          ;NEGATIVE OFFSET
3434  010570  001403                        BEQ     3$              ;BRANCH IF NEGATIVE OFFSET
3435  010572  012703  002166                MOV     #OVWPO,R3       ;SCORE BOARD ADDRESS
3436  010576  000402                        BR      4$
3437  010600  012703  002046          3$:   MOV     #OVWNO,R3       ;SCORE BOARD ADDRESS
3438  010604  113702  037157          4$:   MOVB    @#FMTDPB+$TRK,R2 ;LOAD THE SURFACE NUMBER
3439  010610  005702                        TST     R2              ;ON SURFACE 0
3440  010612  001404                        BEQ     6$              ;BRANCH IF IT IS
3441  010614  062703  000020          5$:   ADD     #16.,R3         ;EACH SCORE BOARD TAKES 16 BYTES
3442  010620  005302                        DEC     R2              ;LOCATED ?
3443  010622  001374                        BNE     5$              ;BRANCH IF NOT
3444  010624  010337  002044          6$:   MOV     R3,TABLEX       ;STORE THE TABLE STARTING ADDRESS
3445  010630  010301                        MOV     R3,R1           ;RE ASSIGN REGISTERS
3446  010632  013702  001266                MOV     $DEVM,R2        ;LOGICAL DRIVE #
3447  010636  013703  001356                MOV     STARSC,R3       ;START SECTOR
3448  010642  116204  002560                MOVB    INDST(R2),R4    ;LOCATE THE STARTING PRONT FOR SCORE BOARD
3449  010646  060304                        ADD     R3,R4           ;UPDATE POINTER
3450  010650  022704  000017          11$:  CMP     #15.,R4         ;SHOULD ADJUST POINTER ?
3451  010654  003403                        BLE     7$              ;NO
3452  010656  162704  000020                SUB     #16.,R4         ;ENTRY POINT AT THE SCORE BOARD
3453  010662  000772                        BR      11$             ;
3454  010664  060401          7$:   ADD     R4,R1           ;
3455  010666  023703  001360          8$:   CMP     CMSEC,R3        ;ENDING SECTOR REACHED ?
```

```
3456  010672  002412                          BLT     9$              ;BRANCH IF IT IS
3457  010674  105221                          INCB    (R1)+           ;INCREMENT SCORE AND POINT TO NEXT
3458                                                                  ;LOGICAL DRIVE
3459  010676  005204                          INC     R4              ;INCREMENT LOGICAL DRIVE #
3460  010700  005203                          INC     R3              ;INCREMENT SECTOR COUNT
3461  010702  022704  000017                  CMP     #15.,R4         ;TIME TO RESET TABLE ?
3462  010706  002367                          BGE     8$              ;BRANCH IF NOT
3463  010710  013701  002044                  MOV     TABLEX,R1       ;RESET TABLE ADDRESS
3464  010714  005004                          CLR     R4              ;LOGICAL DRIVE 0
3465  010716  000763                          BR      8$              ;LOOPING BACK
3466  010720                          9$:
3467  010720  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
3468  010722  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
3469  010724  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
3470  010726  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
3471  010730  000207                          RTS     PC
3472
3473
3474                                  ;LABAD ROUTINE
3475                                  ;LOCATE THE START SECTOR AND THE TERMINATE SECTOR OF THE PREVIOUS
3476                                  ;OPERATION
3477                                  ;
3478                                  ;STARSC :       STARTING SECTOR
3479                                  ;CMSEC  :       ENDING SECTOR
3480                                  ;INFORMATION FROM $RMDA(FMTDPB),$RMWC(FMTDPB),$SEC(FMTDPB)
3481                                  ;               $SEC(FMTDPB),$WRDM(FMTDPB)
3482                                  ;CALLING SEQ
3483                                  ;
3484                                  ;       JSR     PC,LABAD
3485                                  ;       RET
3486
3487
3488  010732                          LABAD:
3489  010732  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3490  010734  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3491  010736  012700  037146                  MOV     #FMTDPB,R0      ;DPB ADDRESS
3492  010742  116002  000026                  MOVB    $RMDA(R0),R2    ;HARDWARE TERMINATING SECTOR
3493  010746  116037  000010  001356          MOVB    $SEC(R0),STARSC ;STARTING SECTOR ADDRESS
3494  010754  005702                          TST     R2              ;TERMINATOR AT 0 SECTOR
3495  010756  001404                          BEQ     1$              ;BRANCH IF IT IS
3496  010760  005302                          DEC     R2              ;DECREMENT ONE SECTOR COUNT
3497  010762  010237  001360                  MOV     R2,CMSEC        ;ENDING SECTOR ADDRESS
3498  010766  000415                          BR      3$              ;EXIT
3499  010770  005760  000022          1$:     TST     $RMWC(R0)       ;WORD COUNT = 0
3500  010774  001407                          BEQ     2$              ;BRANCH IF IT IS
3501  010776  026060  000022  000004          CMP     $RMWC(R0),$WRDM(R0) ; WORD COUNT CHANGED AT ALL ?
3502  011004  001003                          BNE     2$              ;BRANCH IF CHANGED
3503  011006  116037  000010  001360          MOVB    $SEC(R0),CMSEC ;END SECTOR = START SECTOR
3504  011014  112737  000037  001360  2$:     MOVB    #31.,CMSEC      ;END AT SECTOR 31
3505  011022                          3$:
3506  011022  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
3507  011024  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
3508  011026  000207                          RTS     PC
3509
3510                                  ;TEST 7
3511                                  ;SELF TEST:WRITE CYLINDERS $DEVM+17,$DEVM+17+128,$DEVM+17+128X2,ETC.
```

```
                                        ;THEN EXECUTE WRITE CHECK TO DECTECT ANY DATA OR HEADER PROBLEM
                                        ;FOR THE SELECTED LOGICAL DRIVE
                                        ;$DEVM: LOGICAL DRIVE #
                                        ;DRIVE:         PHYSICAL DRIVE #
                                        ;
                                        ;

                                        ;;*****************************************************************
011030  000004                        TST7:   SCOPE
011032  012737  000001  001174                MOV     #1,$TIMES       ;;DO 1 ITERATION
011040  012737  000007  001336                MOV     #7,TSTNM        ;LOAD TEST NUMBER
011046  012706  001100                        MOV     #STACK,SP       ;INITIAL THE STACK
011052  012702  002546                        MOV     #SELF0,R2       ;CLEAR THE SELF TEST SCORE BOARDS
011056  005022                        1$:     CLR     (R2)+           ;
011060  022702  002560                        CMP     #SELF0+12,R2    ;ALL DONE ?
011064  101374                              BHI     1$              ;BRANCH IF NOT
011066  012700  037146                        MOV     #FMTDPB,R0              ;SET UP DPB
011072  012760  160000  000004                MOV     #-8192.,$WRDM(R0) ;LOAD FULL TRACK WORD COUNT
011100  112760  000161  000002                MOVB    #WRTDAT,$COMND(R0)      ;WRITE DATA COMMAND
011106  005060  000010                        CLR     $SEC(R0)                ;SECTOR 0,SURFACE 0
011112  113710  001222                        MOVB    DRIVE,(R0)              ;LOAD PHY. DRIVE
011116  013702  001266                        MOV     $DEVM,R2                ;LOCATE THE STARTING CYLINDER
011122  062702  000021                        ADD     #17.,R2
011126  010260  000012                        MOV     R2,$CYL(R0)             ;CYLINDER ADDRESS
011132  012760  037262  000006                MOV     #ENDPGM,$BUF(R0)        ;RESET BUFFER ADDRESS
011140  012760  037166  000014                MOV     #RM.REG,14(R0)          ;ADDRESS TO SAVE ALL RH/RM03 REG'S
011146  004737  014546                        JSR     PC,FILBUF               ;FILL THE BUFFER WITH WORSE CASE PATTERN
011152  004037  023620                2$:     JSR     R0,RM03                 ;CALL DRIVER HANDLER
011156  037146                              FMTDPB
011160  000774                              BR      2$                      ;BRANCH IF QUEUE FAILS
011162  005737  037164                3$:     TST     FMTDPB+16               ;ALL DONE ?
011166  001775                              BEQ     3$                      ;BRANCH IF NOT
011170  012700  037146                        MOV     #FMTDPB,R0              ;REPORT IF ANY ERROR
011174  004737  013264                        JSR     PC,PROCES
011200  105260  000011                        INCB    $TRK(R0)                ;NEXT SURFACE
011204  122760  000005  000011                CMPB    #5,$TRK(R0)             ;ALL FIVE SURFACE ARE DONE ?
011212  101357                              BHI     2$                      ;BRANCH IF NOT
011214  005060  000010                        CLR     $SEC(R0)                ;RESET SECTOR AND TRACK
011220  062760  000200  000012                ADD     #128.,$CYL(R0)          ;ADVANCE TO NEXT ZONE
011226  022760  001440  000012                CMP     #800.,$CYL(R0)          ; ALL 7 ZONES ARE DONE ?
011234  103346                              BHIS    2$                      ;BRANCH IF NOT
                                                                            ;EXECUTE WRITE CHECK
                                                                            ;TO DETECT ANY DATA OR HEADER
                                                                            ;PROBLEM
011236  112760  000151  000002                MOVB    #WCKD,$COMND(R0)        ;CHANGE TO WRITE CHECK COMMAND
011244  012702  002546                        MOV     #SELF0,R2               ;R2 POINTS TO SCORE BOARD
                                                                            ;CAN NOT BE DESTORIED
011250  013703  001266                        MOV     $DEVM,R3                ;LOACATE STARTING ADDRESS
011254  062703  000021                        ADD     #17.,R3
011260  010360  000012                        MOV     R3,$CYL(R0)             ;STARTING CYLINDER
011264  005037  002640                        CLR     OFFCOD                  ;SET NEGATIVE OFFSET FLAG
011270  004037  023620                4$:     JSR     R0,RM03                 ;CALL DRIVE HANDLER
011274  037146                              FMTDPB
011276  000774                              BR      4$                      ;BRANCH IF QUEUE FAILS
011300  005737  037164                5$:     TST     FMTDPB+16               ;ALL DONE ?
```

```
3568  011304  001775                        BEQ     5$                      ;BRANCH IF NOT
3569  011306  012700  037146                MOV     #FMTDPB,R0
3570  011312  004737  013264                JSR     PC,PROCES
3571  011316  005760  000016                TST     16(R0)                  ;ANY ERROR
3572  011322  100401                        BMI     6$                      ;YES,THEN DON'T INCREMENT SCORE
3573  011324  105212                        INCB    (R2)                    ;INCREMENT SCORE
3574  011326  004537  010176        6$:     JSR     R5,OFFST                ;OFFSET
3575  011332  000415                        BR      8$                      ;BRANCH IF OFFSET FAILS
3576  011334  004737  010400                JSR     PC,MAKEUP               ;EXECUTE WRITE CHECK IN OFFSET MODE
3577  011340  005737  037164        7$:     TST     FMTDPB+16               ;ALL DONE /
3578  011344  001775                        BEQ     7$                      ;BRANCH IF NOT
3579  011346  012700  037146                MOV     #FMTDPB,R0
3580  011352  004737  013264                JSR     PC,PROCES               ;REPORT IF ANY ERROR
3581  011356  005760  000016                TST     16(R0)                  ;ERROR BIT SET ?
3582  011362  100401                        BMI     8$                      ;DON'T INCREMENT SCORE
3583  011364  105212                        INCB    (R2)                    ;INCREMENT SCORE
3584  011366  005737  002640        8$:     TST     OFFCOD                  ;POSITIVE OFFSET IS DONE ?
3585  011372  001006                        BNE     9$                      ;BRANCH ,IF DONE
3586  011374  005202                        INC     R2                      ;INCREMENT SCORE BOARD POINTER
3587  011376  012737  000001  002640        MOV     #1,OFFCOD               ;SET POSITIVE OFFSET FLAG
3588  011404  000137  011270                JMP     4$                      ;POSITIVE OFFSET TEST
3589  011410  105260  000011        9$:     INCB    $TRK(R0)                ;INCREMENT TO NEXT SURFACE
3590  011414  005202                        INC     R2                      ;ADVANCE SCORE BOARD POINTER
3591  011416  122760  000005  000011        CMPB    #5,$TRK(R0)             ;ALL FIVE SURFACES ARE DONE ?
3592  011424  101404                        BLOS    10$                     ;BRANCH , IF ALL DNOE
3593  011426  005037  002640                CLR     OFFCOD                  ;RESET OFFSET DIRECTION
3594  011432  000137  011270                JMP     4$                      ;TRY THE NEXT SURFACE
3595  011436  005060  000010        10$:    CLR     $SEC(R0)                ;SURFACE 0, SECTOR 0
3596  011442  012702  002546                MOV     #SELF0,R2               ;RESET SCORE BOARD
3597  011446  062760  000200  000012        ADD     #128.,$CYL(R0)  ;ADVANCE TO NEXT ZONE
3598  011454  022760  001440  000012        CMP     #800.,$CYL(R0)  ;ALL 7 ZONES ARE DONE ?
3599  011462  103404                        BLO     11$                     ;BRANCH ,IF ALL DONE
3600  011464  005037  002640                CLR     OFFCOD                  ;RESET OFFSET FLAG
3601  011470  000137  011270                JMP     4$                      ;TO NEXT ZONE
3602  011474  000240                11$:    NOP                             ;TO NEXT TEST
3603
3604
3605                                ;TEST10  TEST 8      READ COMPATIBLE TEST
3606                                ;CYLINDERS TESTED : $DEVM+112,$DEVM+112+128XN   N=1 TO 5
3607                                ;THIS TEST SELECT ONE CYLINDER FOR EACH LOGICAL DRIVE FROM
3608                                ;ZONE 1 TO ZONE 6
3609                                ;$DEVM :   LOGICAL DRIVE #
3610                                ;DRIVE :         PHYSICAL DRIVE #
3611
3612
3613                                ;;*****************************************************************
3614  011476  000004                TST10:  SCOPE
3615  011500  012737  000001  001174        MOV     #1,$TIMES               ;;DO 1 ITERATION
3616  011506  012737  000010  001336        MOV     #10,TSTNM               ;LOAD TEST NUMBER
3617  011514  012706  001100                MOV     #STACK,SP               ;INITAIL THE STACK POINTER
3618  011520  012702  002306                MOV     #RDNO,R2                ;CLEAR THE SCORE BOARD OF READ TEST
3619  011524  005022                1$:     CLR     (R2)+
3620  011526  022702  002546                CMP     #RDP4+16.,R2    ;ALL DONE
3621  011532  101374                        BHI     1$                      ;BRANCH IF NOT
3622  011534  005037  002640                CLR     OFFCOD                  ;SET NEGATIVE OFFSET FLAG
3623  011540  012700  037146        LOOP3:  MOV     #FMTDPB,R0              ;SET UP DPB BLOCK
```

```
3624  011544  012760  160000  000004          MOV     #-8192.,$WRDM(R0) ;LOAD THE WORD CTR,FULL TRACK
3625  011552  005060  000010                  CLR     $SEC(R0)          ;SURFACE 0 AND SECTOR 0
3626  011556  112760  000171  000002          MOVB    #RDDAT,$COMND(R0) ;LOAD THE READ DATA COMMAND
3627  011564  012760  037262  000006          MOV     #ENDPGM,$BUF(R0)  ;RESET BUFFER ADDRESS
3628  011572  012760  037166  000014          MOV     #RM.REG,14(R0)    ;ADDRESS TO SAVE ALL RH/RM03 ADDRESS
3629  011600  113710  001222                  MOVB    DRIVE,(R0)        ;LOAD PHY. DRIVE ADDRESS
3630  011604  013703  001266                  MOV     $DEVM,R3          ;LOCATE STARTING CYL
3631  011610  062703  000160                  ADD     #112.,R3
3632  011614  010360  000012                  MOV     R3,$CYL(R0)       ;STARTING CYL ADDRESS
3633  011620  004037  023620          1$:     JSR     R0,RM03           ;CALL THE DRIVE HANDLER
3634  011624  037146                          FMTDPB
3635  011626  000774                          BR      1$
3636  011630  005737  037164          2$:     TST     FMTDPB+16         ;COMMAND DONE ?
3637  011634  001775                          BEQ     2$                ;BRANCH IF NOT
3638  011636  012700  037146                  MOV     #FMTDPB,R0
3639  011642  004737  013264                  JSR     PC,PROCES         ;REPORT IF ANY ERROR
3640  011646  004737  010732                  JSR     PC,LABAD          ;LOACATE START AND STOP ADDRESS
3641  011652  005760  000016                  TST     16(R0)            ;ANY ERROR ?
3642  011656  100006                          BPL     3$                ;BRANCH,IF NONE
3643  011660  023737  001360  001356          CMP     CMSEC,STARSC      ;ON THE SAME SECTOR ADDRESS
3644  011666  101404                          BLOS    4$                ;
3645  011670  005337  001360                  DEC     CMSEC             ;DECREMENT ONE SECTOR COUNT
3646  011674  004737  010512          3$:     JSR     PC,SCORE          ;UPDATE THE SCORE
3647  011700  005760  000022          4$:     TST     $RMWC(R0)         ;WORD COUNT= 0
3648  011704  001407                          BEQ     5$                ;BRANCH IIF IT IS
3649  011706  116060  000026  000010          MOVB    $RMDA(R0),$SEC(R0) ;UPDATE THE NEW STARTING SECTOR
3650  011714  016060  000022  000004          MOV     $RMWC(R0),$WRDM(R0) ;UPDATE WORD COUNT
3651  011722  000736                          BR      1$                ;CONTINUE
3652  011724  012700  037146          5$:     MOV     #FMTDPB,R0        ;RESET THE DPB BLOCK
3653  011730  012760  160000  000004          MOV     #-8192.,$WRDM(R0)  ;FULL TRACK WORD COUNT
3654  011736  105060  000010                  CLRB    $SEC(R0)          ;RESET TO SECTOR 0,TRACK NOT CHANGED
3655  011742  004537  010176          6$:     JSR     R5,OFFST          ;CALL OFFSET
3656  011746  000440                          BR      10$               ;BRANCH IF OFFSET FAILS
3657  011750  004737  010400                  JSR     PC,MAKEUP         ;EXECUTE READ DATA IN OFFSET MODE
3658  011754  005737  037164          7$:     TST     FMTDPB+16         ;OFFSET READ IS DONE ?
3659  011760  001775                          BEQ     7$                ;BRANCH IF NOT
3660  011762  012700  037146                  MOV     #FMTDPB,R0        ;REPORT IF ANY ERROR
3661  011766  004737  013264                  JSR     PC,PROCES
3662  011772  004737  010732                  JSR     PC,LABAD          ;LOCATE THE START AND STOP ADDRESSES
3663  011776  005760  000016                  TST     16(R0)            ;ANY ERROR ?
3664  012002  100006                          BPL     8$                ;BRANCH,IF NONE
3665  012004  023737  001360  001356          CMP     CMSEC,STARSC      ;START AND STOP SECTOR ADDRESSES IS O.K  ?
3666  012012  101404                          BLOS    9$                ;BRANCH IF NOT
3667  012014  005337  001360                  DEC     CMSEC             ;END SECTOR IS ERROR SECTOR
3668  012020  004737  010512          8$:     JSR     PC,SCORE          ;UPDATE THE SCORE
3669  012024  005760  000022          9$:     TST     $RMWC(R0)         ;WORD COUNT IS 0
3670  012030  001407                          BEQ     10$               ;BRANCH IF IT IS
3671  012032  116060  000026  000010          MOVB    $RMDA(R0),$SEC(R0)  ;UPDATE SECTOR ADDRESS
3672  012040  016060  000022  000004          MOV     $RMWC(R0),$WRDM(R0) ;UPDATE WORD COUNT
3673  012046  000735                          BR      6$                ;LOOPING UNTIL CURRENT TRACK IS DONE
3674  012050  062760  000200  000012  10$:     ADD     #128.,$CYL(R0)    ;ADJUST CYLINDER TO NEXT ZONE
3675  012056  022760  001377  000012          CMP     #767.,$CYL(R0)            ;ALL 6 ZONES ARE DONE ?
3676  012064  103402                          BLO     11$               ;BRANCH IF ALL DONE
3677  012066  000137  011620                  JMP     1$
3678  012072  013703  001266          11$:    MOV     $DEVM,R3          ;LOCATE STARTING CYLINDER
3679  012076  062703  000160                  ADD     #112.,R3          ;
```

```
                                GET VALUE FOR SOFTWARE SWITCH REGISTER

3680  012102  010360  000012                MOV     R3,$CYL(R0)     ;STARTING CYLINDER
3681  012106  105260  000011                INCB    $TRK(R0)        ;TO NEXT SURFACE
3682  012112  122760  000005  000011        CMPB    #5,$TRK(R0)     ;ALL 5 SRUFACE ARE DONE
3683  012120  101402                        BLOS    12$             ;BRANCH IF ALL DONE
3684  012122  000137  011620                JMP     1$              ;LOOPING UNTIL CURRENT TRACK IS DONE
3685  012126  005737  002640        12$:    TST     OFFCOD          ;OFFSET CODE = POSITIVE ?
3686  012132  001005                        BNE     13$             ;IF EQUAL,THEN ALL DONE
3687  012134  012737  000001  002640        MOV     #1,OFFCOD       ;SET POSITIVE OFFSET FLAG
3688  012142  000137  011540                JMP     LOOP3
3689  012146  000240                13$:    NOP                     ;TO NEXT TEST
3690
3691
3692
3693
3694                                  ;TEST11  TEST 9
3695                                  ;REPORT THE TEST SCORES
3696                                  ;DIRECT THE OPERATOR TO CHANGE PACK AND MOUNT TO OTHER DIRVE
3697
3698                                  ;$DEVM   :  LOGICAL DRIVE #, SHOULD NOT BE UPDATED BEFORE THE REPORT IS COMPLETED.
3699                                  ;
3700                                  ;BADSEC:        ACCEPTANCE FLAG, BADSEC = 0, ALL DRIVES ARE COMPATIBLE
3701                                  ;                                BADSEC = -1, DRIVES ARE NOT COMPATIBLE
3702                                  ;CMTRK :   TRACK (HEAD) NUMBER FOR CONTROLLING THE SCORE TYPING.
3703                                  ;
3704
3705
3706
3707                                  ;;*****************************************************************
3708  012150  000004                TST11:  SCOPE
3709  012152  012737  000001  001174        MOV     #1,$TIMES       ;;DO 1 ITERATION
3710  012160  012737  000011  001336        MOV     #11,TSTNM       ;LOAD THE TEST NUMBER
3711  012166  012706  001100                MOV     #STACK,SP       ;LOAD THE STACK POINTER
3712                                                                 ;ADJUST THE READ COMPATIBLE TEST SCORE
3713                                                                 ;IF THE SELF READ TEST SCORE IS TOO LOW
3714  012172  005046                        CLR     -(SP)           ;DUMMY SCORE BOARD ADDRESSES
3715  012174  005046                        CLR     -(SP)           ;
3716  012176  012746  002547                MOV     #SELF0+1,-(SP)  ;POSITIVE OFFSET READ TEST SCORE ADDRESS
3717  012202  012746  002426                MOV     #RDP0,-(SP)     ;POSITIVE OFFSET TEST SCORE
3718  012206  012702  002546                MOV     #SELF0,R2       ;NEGATIVE OFFSET READ TEST SCORE
3719  012212  012703  002306                MOV     #RDN0,R3        ;NEGATIVE OFFSET READ COMPATIBLE TEST SCORE
3720  012216  012701  000005        1$:     MOV     #5,R1           ;R1= SURFACE (TRACK, NUMBER
3721  012222  122712  000007        2$:     CMPB    #7,(R2)         ;SELF READ TEST SCORE IS TOO LOW
3722  012226  101411                        BLOS    4$              ;BRANCH IF NOT
3723  012230  012705  000020                MOV     #16.,R5         ;INCREMENT READ COMPATIBLE SCORE FOR ALL 16 DIRVES
3724  012234  111304                3$:     MOVB    (R3),R4         ;ADJUST SCORE BY ADDING 6
3725  012236  062704  000006                ADD     #6,R4
3726  012242  110423                        MOVB    R4,(R3)+        ;UPDATE SCORE AND POINTS TO NEXT DRIVE
3727  012244  005305                        DEC     R5              ;ALL DRIVES ARE UPDATED ?
3728  012246  001372                        BNE     3$              ;BRANCH IF NOT
3729  012250  000402                        BR      5$
3730  012252  062703  000020        4$:     ADD     #16.,R3         ;ADJUST POINTER OF SCORE BOARD ADDRESS
3731  012256  062702  000002        5$:     ADD     #2,R2           ;UPDATE THE SELF TEST SCORE ADDRESS
3732  012262  005301                        DEC     R1              ;ALL FIVE SURFACES ARE DONE ?
3733  012264  001356                        BNE     2$              ;BRANCH IF NOT
3734  012266  012603                        MOV     (SP)+,R3        ;GET NEXT PAIR
3735  012270  012602                        MOV     (SP)+,R2        ;EXIT THE THEY ARE 0
```

```
3736  012272  001351                          BNE     1$
3737                                                    ;
3738                                                    ; SET ACCEPTANCE FLAG,INITIALIZE THE
3739                                                    ; TABLE POINTERS AND TRACK NUMBER
3740  012274  005037  001340                   CLR     BADSEC          ;SET ACCEPTANCE FLAG
3741  012300  005037  001362                   CLR     CMTRK           ;START FROM TRACK 0
3742
3743  012304  005001                          CLR     R1              ;START FROM LOG DRV 0
3744  012306  012702  002046                   MOV     #OVWNO,R2       ;SCORE BOARD BASE ADDRESS
3745  012312  012703  000001                   MOV     #BIT0,R3        ;BIT POSITION FOR LOG DRV 0
3746  012316  030337  001776          LOOP4:  BIT     R3,ASNLST       ;IS THE LOG DRV UNDER TEST ?
3747  012322  001502                          BEQ     LOOP5           ;BRANCH IF NOT
3748  012324  005037  001364                   CLR     NULINE          ;NEW LINE INDICATOR
3749  012330  123701  001266                   CMPB    $DEVM,R1        ;SELF SCORE ?
3750  012334  001434                          BEQ     SPATH           ;BRANCH IF IT IS
3751  012336  122712  000016          KPATH:  CMPB    #14.,(R2)               ;OVERWRITE NEG OFFSET < 14 ?
3752  012342  101403                          BLOS    1$              ;BRANCH IF NOT
3753  012344  004537  012766                   JSR     R5,PRINT        ;REPORT EXCEPTIONS
3754  012350  000001                          000001                  ;COLUMN POSITION ON REPORT
3755  012352  122762  000016  000120  1$:     CMPB    #14.,80.(R2)    ;OVERWRITE POS OFFSET < 14 ?
3756  012360  101403                          BLOS    2$              ;BRANCH IF NOT
3757  012362  004537  012766                   JSR     R5,PRINT        ;REPORT EXECPTIONS
3758  012366  000002                          000002                  ;COLUMN POSITION ON REPORT
3759  012370  122762  000014  000240  2$:     CMPB    #12.,160.(R2)   ;READ NEG OFFSET < 6
3760  012376  101403                          BLOS    3$              ;BRANCH IF NOT
3761  012400  004537  012766                   JSR     R5,PRINT        ;REPORT EXCEPTIONS
3762  012404  000003                          000003                  ;COLUMN POSITION ON REPORT
3763  012406  122762  000014  000360  3$:     CMPB    #12.,240 (R2)   ;READ POS OFFSET < 12 ?
3764  012414  101445                          BLOS    LOOP5           ;BRANCH IF NOT
3765  012416  004537  012766                   JSR     R5,PRINT        ;REPORT EXECPTIONS
3766  012422  000004                          000004                  ;COLUMN POSITION ON REPORT
3767  012424  000441                          BR      LOOP5           ;EXIT
3768  012426  122712  000016          SPATH:  CMPB    #14.,(R2)               ;OVERWRITE NEG OFF < 14
3769  012432  101403                          BLOS    1$              ;BRANCH IF NOT
3770  012434  004537  012766                   JSR     R5,PRINT        ;REPORT EXECPTION
3771  012440  000001                          000001                  ;COLUMN POSITION ON REPORT
3772  012442  122762  000016  000120  1$:     CMPB    #14.,80.(R2)    ;OVERWRITE POS OFFSET < 14
3773  012450  101403                          BLOS    2$              ;BRANCH IF NOT
3774  012452  004537  012766                   JSR     R5,PRINT        ;REPORT EXCEPTION
3775  012456  000002                          000002                  ;COLUMN POSITION ON REPORT
3776  012460  013704  001362          2$:     MOV     CMTRK,R4        ;LOCATE THE SELF TEST SCORE
3777  012464  006304                          ASL     R4              ;WORD INDEX
3778  012466  122764  000006  002546          CMPB    #6,SELF0(R4)    ;SELF READ NEG OFFSET < 6 ?
3779  012474  101403                          BLOS    3$              ;BRANCH IF NOT
3780  012476  004537  012766                   JSR     R5,PRINT
3781  012502  000003                          000003                  ;REPORT EXCEPTIONS
3782  012504  013704  001362          3$:     MOV     CMTRK,R4        ;LOCATE THE SELF TEST SCORE
3783  012510  006304                          ASL     R4              ;WORD INDEX
3784  012512  122764  000006  002547          CMPB    #6,SELF0+1(R4)  ;SELF READ POS OFFSET < 6 ?
3785  012520  101403                          BLOS    LOOP5           ;BRANCH IF NOT
3786  012522  004537  012766                   JSR     R5,PRINT        ;REPORT EXCEPTIONS
3787  012526  000004                          000004                  ;COLUMN POSITION FOR REPORT PRINTING
3788  012530  005201                  LOOP5:  INC     R1              ;INCREASE THE LOGICAL DRIVE #
3789  012532  000241                          CLC                     ;ADJUST THE BIT POSITION
3790  012534  006103                          ROL     R3              ;POINTS TO NEXT DRIVE
3791  012536  005202                          INC     R2              ;UPDATE SCORE BOARD BASE ADDRESS
```

```
3792  012540 022701 000017          CMP    #15.,R1          ;ALL DRIVES ARE CHECK ?
3793  012544 103264                 BHIS   LOOP4            ;BRANCH IF NOT ALL DONE
3794  012546 005237 001362 LOOP6:   INC    CMTRK            ;NEXT SURFACE
3795  012552 012702 002046         MOV    #OVWNO,R2        ;RESET SCORE BOARD BASE ADDRESS
3796  012556 005001                CLR    R1               ;LOGICAL DRIVE START FROM 0
3797  012560 013703 001362         MOV    CMTRK,R3         ;LOCATE THE SCORE BOARD BASE ADDRESS
3798  012564 001404                BEQ    2$               ;BRANCH IF ON SURFACE 0
3799  012566 062702 000020 1$:      ADD    #16.,R2          ;FOR EACH SURFACE,16 BYTES
3800  012572 005303                DEC    R3               ;ALL SURFACE DONE ?
3801  012574 001374                BNE    1$               ;BRANCH IF NOT
3802  012576 012703 000001 2$:      MOV    #BIT0,R3                 ;BIT MAP POSITION FOR DRIVE 0
3803                                                        ;AT THIS POINT:
3804                                                        ;R1=0,LOGICAL DRIVE # START FROM 0
3805                                                        ;R2 ADDRESS OF SCORE BOARD BASE ADDRESS
3806                                                        ;R3=1,BIT MAP ,INDICATE LOGICAL DRIVE 0
3807  012602 122737 000005 001362   CMPB   #5,CMTRK         ;ALL SURFACES ARE DONE ?
3808  012610 101242                BHI    LOOP4            ;CONTINUE
3809  012612 005737 001340         TST    BADSEC           ;ACCEPTANCE FLAG SET ?
3810  012616 100402                BMI    DISMNT           ;NO,THEN BRANCH
3811  012620 104401 001205         TYPE   ,$CRLF
3812                               TYPE   ,MESG15 ;** ALL DRIVES ARE COMPATIBLE **
3813  012624 104401 001205 DISMNT:  TYPE   ,$CRLF            ;CR-LF
3814  012630 012777 013256 010242   MOV    #IDLEX,@RMVEC    ;RESET THE INTERRUPT VECTOR
3815  012636 005077 010240         CLR    @RMVEC+2         ;CLEAR THE INTERRUPT VECTOR
3816  012642 104401 035323         TYPE   ,MESG12          ;UNLOAD AND DISMOUNT MESSAGE
3817  012646 104401 001272         TYPE   ,$CDW2           ;SYSTEM NAME
3818  012652 013746 001222         MOV    DRIVE,-(SP)      ;PHYSICAL DRIVE NUMBER
3819  012656 104403                TYPOS
3820  012660    002                .BYTE  2
3821  012661    000                .BYTE  0
3822  012662 104401 035341         TYPE   ,MESG13          ;MESSAGE TYPE R<CR> WHEN READY
3823  012666 104411          1$:    RDLIN                  ;READ IN ONE LINE
3824  012670 012605                MOV    (SP)+,R5         ;LOCATE THE READ IN LINE
3825  012672 122527 000122         CMPB   (R5)+,#'R        ;READY ?
3826  012676 001373                BNE    1$               ;TRY AGAIN IF NOT
3827  012700 105715                TSTB   (R5)             ;TERMINATOR ?
3828  012702 001371                BNE    1$               ;BRANCH IF NOT
3829  012704 005002          DEASG: CLR    R2               ;LOGICAL DRIVE NUMBER
3830  012706 012703 000001         MOV    #BIT0,R3         ;BIT MAP OF ASSIGNED DRIVE
3831  012712 020237 001266  1$:     CMP    R2,$DEVM         ;IS THE LOGICAL DRIVE UNDER TEST ?
3832  012716 001404                BEQ    2$               ;BRANCH IF IT IS
3833  012720 000241                CLC                     ;SHIFT THE BIT MAP FOR
3834  012722 006103                ROL    R3               ;NEXT DRIVE
3835  012724 005202                INC    R2               ;INCREMENT THE LOGICAL DRIVE #
3836  012726 000771                BR     1$               ;LOOPING ,UNTIL THE LOGICAL DRIVE LOCATED
3837  012730 040337 002002  2$:     BIC    R3,ASSGN2        ;CLEAR THE ASSIGNED BIT
3838  012734 001412                BEQ    XEND2            ;BRANCH IF NO MORE DRIVES
3839  012736 005237 001266         INC    $DEVM            ;INCREMENT THE LOGICAL DRIVE #
3840  012742 013702 001266         MOV    $DEVM,R2         ;UPDATE SYSTEM BLOCK ADDRESS
3841  012746 006302                ASL    R2
3842  012750 016237 002600 001270  MOV    BLKADR(R2),$CDW1         ;SYSTEM BLOCK ADDRESS
3843  012756 000137 007056         JMP    TST5             ;JUMP TO TEST 5 FOR OTHER DRIVE
3844
3845
3846  012762 000137 017224 XEND2:   JMP    $EOP     ;END OF PASS
3847
```

```
3848
3849                                    ;PRINT ROUTINE
3850                                    ;NAME PRINT
3851                                    ;PRINT EXCEPTIONS FOR TEST SCORE
3852                                    ;PARAMETER USED
3853                                    ;   NULINE = 0          A NEW LINE TO PRINT
3854                                    ;          = 1 - 4      COLUMN NUMBER TO PRINT THE EXCEPTION MARK
3855                                    ;
3856                                    ;   BADSEC = 0          FIRST EXCEPTION DETECTED
3857                                    ;    BADSEC = -1        NOT FIRST EXCEPTION (DON'T PRINT THE TITLE)
3858                                    ;
3859                                    ;       CALLING SEQ:
3860                                    ;       JSR     R5,PRINT
3861                                    ;       NUMBER                  COLUMN NUMBER
3862                                    ;       RET
3863                                    ;
3864                                    ;       R1: LOGICAL DRIVE #
3865                                    ;       $DEVM: LOGICAL DRIVE UNDER TEST
3866                                    ;       R3: BIT POSITION OF LOGICAL DRIVE IN R1
3867                                    ;
3868
3869
3870
3871  012766                            PRINT:
3872  012766  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3873  012770  010246                            MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3874  012772  010346                            MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3875  012774  010446                            MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3876  012776  005737  001340                    TST     BADSEC          ;FIRST EXCEPTION
3877  013002  100432                            BMI     1$              ;BRANCH IF NOT,DON'T HAVE TO PRINT
3878                                                                    ;TITLE
3879  013004  104401  001205                    TYPE    ,$CRLF          ;CRLF
3880  013010  104401  035503                    TYPE    ,MESG16         ;SCORES FOR DRIVE --
3881  013014  104401  001272                    TYPE    ,$CDW2          ;SYSTEM NAME
3882  013020  013746  001222                    MOV     DRIVE,-(SP)     ;TYPE THE PHYSICAL DRIVE #
3883  013024  104403                            TYPOS
3884  013026     002                            .BYTE   2
3885  013027     000                            .BYTE   0
3886  013030  104401  001205                    TYPE    ,$CRLF          ;CR-LF
3887  013034  104401  035526                    TYPE    ,MESG17         ;SUB TITLE 1
3888  013040  104401  001205                    TYPE    ,$CRLF
3889  013044  104401  035622                    TYPE    ,MESG18         ;SUB TITLE 2
3890  013050  104401  001205                    TYPE    ,$CRLF
3891  013054  012737  177777  001340            MOV     #-1,BADSEC      ;RESET THE ACCEPTANCE FLAG
3892  013062  012737  000001  001374            MOV     #1,FAULT        ;NOT COMPATIBLE FLAG
3893  013070  005737  001364            1$:     TST     NULINE          ;NEW LINE ?
3894  013074  001045                            BNE     5$              ;BRANCH IF NOT
3895  013076  104401  001205                    TYPE    ,$CRLF
3896  013102  104401  001205                    TYPE    ,$CRLF
3897  013106  013746  001362                    MOV     CMTRK,-(SP)     ;TYPE THE SURFACE NUMBER
3898  013112  104403                            TYPOS
3899  013114     002                            .BYTE   2
3900  013115     000                            .BYTE   0
3901  013116  104401  035725                    TYPE    ,TAB            ;TYPE "TABLE" CHARACTER
3902  013122  123701  001266                    CMPB    $DEVM,R1        ;SCORE FOR $DEVM ITSELF ?
3903  013126  001005                            BNE     2$              ;BRANCH IF NOT
```

```
3904  013130  104401  035717          TYPE    ,SELFX          ;MESSAGE: SELF  .
3905  013134  104401  035725          TYPE    ,TAB
3906  013140  000420                  BR      4$              ;NEXT STEP
3907  013142  006301              2$: ASL     R1              ;LOCATE THE SYS HISTORY FILE
3908  013144  016137  002600  013174  MOV     BLKADR(R1),-(SP) ;LOCATE THE SYSTEM NAME AND DRIVE #
3909  013152  006201                  ASR     R1              ;RESTORE DRIVE #
3910  013154  062737  000014  013174  ADD     #$SYSNM,(SP)    ;LOCATE THE SYSTEM NAME AND DRIVE #
3911  013162  104401  036121          TYPE    ,LINSP          ;
3912  013166  104401  036121          TYPE    ,LINSP          ;
3913  013172  104401                  TYPE                    ;TYPE THE SYSTEM AND DRIVE
3914  013174  000000              3$: .WORD   0               ;ADDRESS FOR TYPING MESSAGE
3915  013176  104401  035725          TYPE    ,TAB            ;
3916  013202  012737  000001  001364  4$: MOV     #1,NULINE       ;INDICATE PRINTER STOPS AT COULMN 1
3917  013210  012504              5$: MOV     (R5)+,R4        ;RETRIEVE THE DESIRED COLUMN # FROM
3918                                                          ;CALLING ROUTINE
3919  013212  020437  001364      6$: CMP     R4,NULINE       ;ON THE RIGHT COLUMN ?
3920  013216  103412                  BLO     8$              ;IF LOW NOT PRINT
3921  013220  001405                  BEQ     7$              ;BRANCH IF LOCATED
3922  013222  104401  035725          TYPE    ,TAB            ;ADVANCE TO NEXT COLUMN
3923  013226  005237  001364          INC     NULINE          ;INCREMENT COLUMN CTR
3924  013232  000767                  BR      6$              ;CHECK AGAIN
3925  013234  104401  035727      7$: TYPE    ,MARKX          ;THE EXCEPTION MARK * 0
3926  013240  005237  001364          INC     NULINE          ;NEXT COLUMN
3927  013244              8$:
3928  013244  012604                  MOV     (SP)+,R4        ;;POP STACK INTO R4
3929  013246  012603                  MOV     (SP)+,R3        ;;POP STACK INTO R3
3930  013250  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
3931  013252  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
3932  013254  000205                  RTS     R5              ;EXIT
3933
3934
3935
3936  013256  000240          IDLEX:  NOP                     ;RESET ALL RH VECTOR FOR MOUNT AND DISMOUNT
3937  013260  000240                  NOP
3938  013262  000002                  RTI                     ;EXIT
3939                          ;PROCESS THE ORDER TERMINATION
3940
3941  013264  111037  001326  PROCES: MOVB    (R0),UNIT       ;DRIVE NUMBER FOR ANY ERROR MESSAGES
3942  013270  005760  000016          TST     $TATUS(R0)      ;SEE IF DRIVER SIGNALED AN ERROR
3943  013274  100421                  BMI     ERPROC          ;BR IF ERROR
3944  013276  032760  100000  000020  BIT     #BIT15,$RMCS1(R0)  ;SEE IF 'SC' SET
3945  013304  001410                  BEQ     1$              ;BR IF NOT SET
3946  013306  032760  040000  000020  BIT     #BIT14,$RMCS1(R0)  ;SEE IF 'TRE' SET
3947  013314  001011                  BNE     ERPROC          ;BR IF SET
3948  013316  032760  040000  000032  BIT     #BIT14,$RMDS(R0)  ;SEE IF 'ERR' SET
3949  013324  001005                  BNE     ERPROC          ;BR IF SET
3950  013326  004737  014276      1$: JSR     PC,CKERR        ;NO ERROR, CHECK ERROR BITS ANYWAY
3951  013332  004737  014344          JSR     PC,CKBUS        ;NO ERROR, CHECK BUS ADDR & WC
3952  013336  000207              2$: RTS     PC              ;RETURN
3953
3954                          ;ORDER TERMINATED WITH AN ERROR - PROCESS THE ERROR
3955
3956  013340  032760  000200  000016  ERPROC: BIT     #BIT07,$TATUS(R0) ;DONE BIT SET ?
3957  013346  001402                  BEQ     ERPRC1          ;BR IF ORDER DIDN'T COMPLETE NORMALLY
3958  013350  000137  013516          JMP     DONE            ;PROCESS ERROR WITH 'DONE' BIT SET
3959
```

```
                                        ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE NOT' BITS
                                        
013354  032760  010000  000016  ERPRC1: BIT     #BIT12,$STATUS(R0) ;SEE IF DRIVE WAS UNSAFE
013362  001025                          BNE     PUNSAF             ;BR IF YES
013364  032760  004000  000016          BIT     #BIT11,$STATUS(R0) ;PARITY ERROR OCCURRED
013372  001025                          BNE     UCPAR              ;BR IF IT DID
013374  032760  002000  000016          BIT     #BIT10,$STATUS(R0) ;FATAL PARITY ERROR?
013402  001025                          BNE     FALPAR             ;BR IF THERE IS ONE
013404  032760  001000  000016          BIT     #BIT09,$STATUS(R0) ;TIMEOUT?
013412  001025                          BNE     SWTIM              ;BR IF YES
013414  032760  040002  000016          BIT     #BIT14!BIT01,$STATUS(R0)  ;DRIVE WENT OFFLINE ?
013422  001025                          BNE     OFLIN              ;BR IF IT DID
013424  032760  000004  000016          BIT     #BIT2,$STATUS(R0)  ;PORT REQUEST TIME OUT ?
013432  001025                          BNE     PRTIM              ;BR IF IT DID
013434  000207                          RTS     PC                 ;ERROR. RETURN
                                        
                                        ;DRIVE IS PERSISTENTLY UNSAFE
                                        
013436                                  PUNSAF:
013436  104414  032216                          DISPLY  ,EM12
013442  000137  014264                          JMP     DUMP
                                        
                                        ;UNCORRECTABLE MASSBUS PARITY ERROR OCCURRED
                                        
013446                                  UCPAR:
013446  104414  032120                          DISPLY  ,EM10
013452  000137  014264                          JMP     DUMP
                                        
                                        ;'FATAL' MASSBUS PARITY ERROR OCCURRED
                                        
013456                                  FALPAR:
013456  104414  032163                          DISPLY  ,EM11
013462  000137  014264                          JMP     DUMP
                                        
                                        ;SOFTWARE TIMEOUT OCCURRED
                                        
013466                                  SWTIM:
013466  104414  032247                          DISPLY  ,EM13
013472  000137  014264                          JMP     DUMP
                                        
                                        ;DRIVE WENT OFFLINE
                                        
013476                                  OFLIN:
013476  104414  032321                          DISPLY  ,EM14
013502  000137  014264                          JMP     DUMP
                                        ;PORT REQUEST TIMEOUT ERROR
                                        
013506                                  PRTIM:
013506  104414  032344                          DISPLY  ,EM15
013512  000137  014264                          JMP     DUMP
                                        
                                        ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE' BITS SET
                                        
013516  032760  000030  000016  DONE:   BIT     #BIT04!BIT03,$STATUS(R0) ;UNSAFE OCCURRED
013524  001402                          BEQ     .+6                ;BR IF NOT
013526  000137  014254                  JMP     UNSAF              ;REPORT UNSAFE
```

```
4016  013532  032760  040000  000030          BIT     #BIT14,$RMCS2(R0)  ;IS 'WCE' SET ?
4017  013540  001402                          BEQ     .+6                ;BRANCH IF NOT SET
4018  013542  000137  014034                  JMP     WCKER              ;WRITE CHECK ERROR
4019  013546  032760  040000  000032          BIT     #BIT14,$RMDS(R0)   ;CHECK 'ERR'
4020  013554  001002                          BNE     1$                 ;BR IF SET
4021  013556  000137  014224                  JMP     TRFER              ;PROCESS 'TRE'
4022  013562  032760  000400  000034  1$:     BIT     #BIT08,$RMER1(R0)  ;'HCRC' SET?
4023  013570  001402                          BEQ     .+6                ;BR IF NOT
4024  013572  000137  014064                  JMP     HCRCER             ;PROCESS 'HCRC'
4025  013576  032760  000020  000034          BIT     #BIT04,$RMER1(R0)  ;'FMT' SET?
4026  013604  001402                          BEQ     .+6                ;BR IF NOT SET
4027  013606  000137  014104                  JMP     CKFMT              ;CHECK FORMAT ERROR
4028  013612  032760  000200  000034          BIT     #BIT07,$RMER1(R0)  ;'HCE' SET?
4029  013620  001402                          BEQ     .+6                ;BR IF NOT SET
4030  013622  000137  014114                  JMP     CKHCE              ;CHECK 'HCE' ERROR
4031  013626  032760  020000  000034          BIT     #BIT13,$RMER1(R0)  ;'OPI' SET?
4032  013634  001402                          BEQ     .+6                ;BR IF NOT SET
4033  013636  000137  014134                  JMP     OPIER              ;REPORT 'OPI'
4034  013642  032760  000010  000034          BIT     #BIT3,$RMER1(R0)   ;'PAR' SET?
4035  013650  001402                          BEQ     .+6                ;BR IF NOT SET
4036  013652  000137  014154                  JMP     PARER              ;REPORT 'PAR'
4037  013656  032760  000040  000034          BIT     #BIT5,$RMER1(R0)   ;'WCF' SET?
4038  013664  001402                          BEQ     .+6                ;BR IF NOT SET
4039  013666  000137  014244                  JMP     WCFER              ;REPORT 'WCF'
4040  013672  032760  002000  000034          BIT     #BIT10,$RMER1(R0)  ;'IAE' SET?
4041  013700  001402                          BEQ     .+6                ;BR IF NOT SET
4042  013702  000137  014164                  JMP     IAEER              ;REPORT 'IAE'
4043  013706  032760  004000  000034          BIT     #BIT11,$RMER1(R0)  ;'WLE' SET?
4044  013714  001402                          BEQ     .+6                ;BR IF NOT SET
4045  013716  000137  014174                  JMP     WLEER              ;REPORT 'WLE'
4046  013722  032760  001000  000034          BIT     #BIT9,$RMER1(R0)   ;'AOE' SET?
4047  013730  001405                          BEQ     2$                 ;BR IF NOT SET
4048  013732  032760  002000  000032          BIT     #BIT10,$RMDS(R0)   ;'LST' SET?
4049  013740  001401                          BEQ     2$                 ;BR IF NOT SET
4050  013742  000207                          RTS     PC                 ;'AOE' & 'LST' SET, EXIT
4051  013744  032760  010000  000034  2$:     BIT     #BIT12,$RMER1(R0)  ;SEE IF 'DTE' SET
4052  013752  001402                          BEQ     .+6                ;BR IF NOT
4053  013754  000137  014144                  JMP     DTEER              ;REPORT 'DTE' ERROR
4054  013760  005760  000034                  TST     $RMER1(R0)         ;SEE IF 'DCK' SET
4055  013764  100002                          BPL     .+6                ;BR IF NOT
4056  013766  000137  014024                  JMP     DCKER              ;PROCESS 'DCK'
4057  013772  032760  060000  000062          BIT     #BIT14!BIT13,$RMER2(R0) ;'SKI' OR 'OCYL' SET
4058  014000  001006                          BNE     3$                 ;BR IF IT IS
4059  014002  032760  100000  000062          BIT     #BIT15,$RMER2(R0)        ;BAD SPOT ?
4060  014010  001004                          BNE     4$                 ;BRANCH IF SO
4061  014012  000137  014074                  JMP     DRVER              ;REPORT ERROR
4062  014016  000137  014234          3$:     JMP     SKIER              ;REPORT DRIVE ERROR
4063  014022  000207                  4$:     RTS     PC                 ;
4064
4065                                  ;PROCESS DATA ('DCK') CHECK ERROR
4066
4067  014024                          DCKER:
4068  014024  104414  032421                  DISPLY  ,EM21
4069  014030  000137  014264                  JMP     DUMP
4070
4071                                  ;WRITE CHECK ERROR PROCESSING
```

```
4072
4073  014034                          WCKER:
4074  014034  032760  100000  000034          BIT     #BIT15,$RMER1(R0)       ;DCK BIT SET ?
4075  014042  001004                          BNE     1$                      ;BRANCH IF SET
4076  014044  104414  032525                  DISPLY  ,EM23
4077  014050  000137  014264                  JMP     DUMP
4078  014054  104414  032452          1$:     DISPLY  ,EM22
4079  014060  000137  014264                  JMP     DUMP
4080
4081                                  ;REPORT 'HCRC' ERROR
4082
4083  014064                          HCRCER:
4084  014064  104414  032400                  DISPLY  ,EM20
4085  014070  000137  014264                  JMP     DUMP
4086
4087                                  ;REPORT DRIVE ERROR
4088
4089  014074                          DRVER:
4090  014074  104414  033015                  DISPLY  ,EM30
4091  014100  000137  014264                  JMP     DUMP
4092
4093                                  ;PROCESS FORMAT ('FER') ERROR
4094
4095  014104                          CKFMT:
4096  014104  104414  032604                  DISPLY  ,EM24
4097  014110  000137  014264                  JMP     DUMP
4098
4099                                  ;PROCESS HEADER COMPARE ('HCE') ERROR
4100
4101  014114                          CKHCE:
4102  014114  104414  032652                  DISPLY  ,EM25
4103  014120  000137  014264                  JMP     DUMP
4104                                  ;POSSIBLE POSITIONING ERROR
4105
4106  014124  104414  034153          POSER:  DISPLY  ,EM51
4107  014130  000137  014264                  JMP     DUMP
4108                                  ;REPORT 'OPI' ERROR
4109  014134                          OPIER:
4110  014134  104414  033047                  DISPLY  ,EM31
4111  014140  000137  014264                  JMP     DUMP
4112                                  ;REPORT 'DTE' ERROR
4113
4114  014144                          DTEER:
4115  014144  104414  033112                  DISPLY  ,EM32
4116  014150  000137  014264                  JMP     DUMP
4117                                  ;REPORT 'PAR' ERROR
4118
4119  014154                          PARER:
4120  014154  104414  033145                  DISPLY  ,EM33
4121  014160  000137  014264                  JMP     DUMP
4122                                  ;REPORT 'IAE' ERROR
4123
4124  014164                          IAEER:
4125  014164  104414  033264                  DISPLY  ,EM35
4126  014170  000137  014264                  JMP     DUMP
4127
```

```
                                        ;REPORT WLE ERROR
                                        WLEER:
014174                                          DISPLY  ,EM36
014174  104414  033322                           JMP     DUMP
014200  000137  014264
                                        ;REPORT FORMAT ERROR
                                        FMTER:
014204                                          DISPLY  ,EM26
014204  104414  032733                           JMP     DUMP
014210  000137  014264
                                        ;REPORT HEADER COMPARE ERROR
014214  104414  032760                HCEER:  DISPLY  ,EM27
014220  000137  014264                        JMP     DUMP

                                        ;PROCESS CONTROL/INTERFACE TRANSFER ERROR
                                        TRFER:
014224                                          DISPLY  ,EM40
014224  104414  033435                           JMP     DUMP
014230  000137  014264

                                        ;PROCESS 'SKI' OR 'OCYL'
                                        SKIER:
014234                                          DISPLY  ,EM50
014234  104414  034115                           JMP     DUMP
014240  000137  014264
                                        ;REPORT WRITE CLOCK FAILURE
                                        WCFER:
014244                                          DISPLY  ,EM34
014244  104414  033222                           JMP     DUMP
014250  000137  014264

                                        ;REPORT DRIVE UNSAFE ERROR
                                        UNSAF:
014254                                          DISPLY  ,EM60
014254  104414  034216                           JMP     DUMP
014260  000137  014264


                                        DUMP:
014264                                          ERROR   7
014264  104007                                  ERROR   10
014266  104010                                  ERROR   11
014270  104011                                  ERROR   12
014272  104012                                  RTS     PC
014274  000207


                                        ;CKECK ERROR BITS IN THE RH11 AND RM03 REGSTERS

014276  032760  060000  000020        CKERR:  BIT     #60000,$RMCS1(R0)       ;SEE IF 'TRE' OR 'MCPE'
014304  001012                                BNE     1$                      ;YES
014306  032760  177400  000030                BIT     #177400,$RMCS2(R0)      ;ERROR BITS IN CS2 /
014314  001006                                BNE     1$                      ;YES
014316  005760  000034                        TST     $RMER1(R0)              ;ANY ERROR IN ER1
014322  001003                                BNE     1$                      ;YES
014324  005760  000062                        TST     $RMER2(R0)              ;ANY ERROR IN ER2
```

```
4184  014330  001404                        BEQ     2$              ;BRANCH IF NO ERROR
4185  014332  104414  033671        1$:     DISPLY  ,EM44
4186  014336  000137  014264                JMP     DUMP            ;TYPE ALL REGISTERS
4187  014342  000207                2$:     RTS     PC
4188
4189                                ;CHECK BUS ADDRESS REGISTER AND WORD COUNT REGISTER
4190
4191  014344  005760  000022        CKBUS:  TST     $RMWC(R0)       ;WORD COUNT = 0
4192  014350  001011                        BNE     1$              ;NO
4193  014352  016046  000004                MOV     $WRDM(R0),-(SP) ;WORD LENGTH
4194  014356  005416                        NEG     (SP)            ;GET THE POSITIVE NUMBER OF WORD COUNT
4195  014360  006316                        ASL     (SP)            ;BYTE COUNT
4196  014362  066016  000006                ADD     $BUF(R0),(SP)
4197  014366  022660  000024                CMP     (SP)+,$RMBA(R0)
4198  014372  001404                        BEQ     2$
4199  014374  104414  033477        1$:     DISPLY  ,EM41
4200  014400  000137  014264                JMP     DUMP            ;TYPE ALL REGISTERS
4201  014404  000207                2$:     RTS     PC
4202                                ;ROUTINE TO DISPLAY THE SECTOR WHICH GAVE THE HARD ERROR
4203
4204  014406  104401  001205        PRTBAD: TYPE    ,$CRLF
4205  014412  104401  036351                TYPE    ,MESG19
4206  014416  104401  001205                TYPE    ,$CRLF
4207  014422  016001  000024                MOV     $RMBA(R0),R1    ;PUT THE END ADDRESS INTO R1
4208  014426  016046  000004                MOV     $WRDM(R0),-(SP) ;FIND THE BEGINNING OF THE SECTOR
4209  014432  005416                        NEG     (SP)            ;GET THE POSITIVE NUMBER OF WORD COUNT
4210  014434  066016  000022                ADD     $RMWC(R0),(SP)  ;SUBTRACT THE WORDS NOT TRANSFERED
4211  014440  005046                        CLR     -(SP)           ;MAKE THE UPPER DIVIDEND 0
4212  014442  012746  000400                MOV     #256.,-(SP)  ;DIVDE THE WORDS TRANSFERED BY THE SECTOR SIZE
4213  014446  004737  015472                JSR     PC,LINKDV       ;DIVIDE
4214  014452  005716                        TST     (SP)            ;REMANDER = 0 ?
4215  014454  001403                        BEQ     1$              ;BR IF IT IS - COMPLETE SECTOR TRANSFERED
4216  014456  006316                        ASL     (SP)            ;CONVERT THE RESIDUAL SECTOR SIZE INTO BYTE COUNT
4217  014460  161601                        SUB     (SP),R1         ;SUBTRACT IT FROM THE END ADDRESS
4218  014462  000402                        BR      2$              ;FINISH THE SIZING
4219  014464  162701  001000        1$:     SUB     #1000,R1        ;SUBTRACT FULL SECTOR SIZE FROM END ADDR
4220  014470  062706  000004        2$:     ADD     #4,SP           ;RESTORE THE STACK POINTER
4221  014474  012702  000007        3$:     MOV     #7,R2           ;R2 CONTAINS THE WORDS/LINE COUNT
4222  014500  010146                        MOV     R1,-(SP)        ;PUT THE ADDRESS ON THE STACK
4223  014502  004737  014666                JSR     PC,LINOCT       ;TYPE THE ADDRESS
4224  014506  020160  000024        4$:     CMP     R1,$RMBA(R0)    ;PRINTED ALL THE SECTOR ?
4225  014512  001412                        BEQ     5$              ;BR IF ALL PRINTED
4226  014514  104414  036121                DISPLY  ,LINSP          ;SPACES
4227  014520  012146                        MOV     (R1)+,-(SP)     ;PUT THE DATA ON THE STACK
4228  014522  004737  014666                JSR     PC,LINOCT       ;TYPE THE DATA
4229  014526  005302                        DEC     R2              ;DECREMENT THE HORIZONTAL COUNT
4230  014530  001366                        BNE     4$              ;BR IF NOT AT THE END OF THE LINE
4231  014532  104414  001205                DISPLY  ,$CRLF          ;CR-LF
4232  014536  000756                        BR      3$              ;RESTORE THE WORDS/LINE COUNT
4233  014540  104414  001205        5$:     DISPLY  ,$CRLF          ;PRINT WHAT REMAINS IN THE BUFFER
4234  014544  000207                6$:     RTS     PC              ;RETURN
4235
4236
4237
4238  014546  104412                FILBUF: SAVREG                  ;SAVE THE REGISTERS
4239  014550  016001  000006                MOV     $BUF(R0),R1     ;BUFFER ADDRESS
```

```
4240  014554  016002  000004                MOV     $WRDM(R0),R2    ;POSITIVE WORD COUNT
4241  014560  005402                       NEG     R2              ;
4242  014562  012705  002642                MOV     #$TNDAT,R5      ;PATTERN ADDRESS
4243  014566  012703  000020                MOV     #20,R3          ;PATTERN COUNT
4244  014572  012521                3$:     MOV     (R5)+,(R1)+     ;MOVE THE PATTERN INTO THE BUFFER
4245  014574  005302                       DEC     R2              ;DECREMENT THE WORD COUNT
4246  014576  003407                       BLE     4$              ;BR IF DONE (WORD COUNT = 0)
4247  014600  005303                       DEC     R3              ;DECREMENT THE PATTERN COUNT
4248  014602  001373                       BNE     3$              ;BR IF MORE PATTERN
4249  014604  012703  000020                MOV     #20,R3          ;RESTORE PATTERN COUNT
4250  014610  012705  002642                MOV     #$TNDAT,R5      ;RESTORE THE ADDRESS
4251  014614  000766                       BR      3$              ;CONTINUE DISTRIBUTING THE PATTERN
4252  014616  104413                4$:     RESREG                  ;RESTORE THE REGISTERS
4253  014620  000207                       RTS     PC              ;RETURN
4254
4255                                  ;;**************************************************************
4256
4257                                  .SBTTL  ERROR MESSAGE GENERATION ROUTINES
4258
4259                                  ;;**************************************************************
4260
4261                                  ;PRINT LINE 1 OF ERROR MESSAGE:
4262                                  ;'HH:MM:SS'
4263
4264  014622  032777  002000  164324  LINE1:  BIT     #SW10,@SWR      ;SWITCH 10 SET ?
4265  014630  001402                       BEQ     1$              ;BR IF NOT
4266  014632  104401  001200                TYPE    ,$BELL          ;RING THE BELL
4267  014636  032777  020000  164310  1$:     BIT     #SW13,@SWR      ;INHIBIT TYPEOUT ?
4268  014644  001403                       BEQ     2$              ;BR IF NOT
4269  014646  104414  001205                DISPLY  ,$CRLF          ;CR-LF
4270  014652  000404                       BR      3$              ;EXIT
4271  014654  004737  015136          2$:     JSR     PC,$TIME        ;TYPE THE TIME
4272  014660  104414  036332                DISPLY  ,LINSPO         ;SPACES
4273  014664  000207                 3$:     RTS     PC              ;RETURN & TYPE DESCRIPTION
4274
4275                                  ;OCTAL TYPEOUT ROUTINE
4276                                  ;CALL:
4277                                  ;       MOV     NUM,-(SP)       ;PUT THE NUMBER ON THE STACK
4278                                  ;       JSR     PC,LINOCT
4279                                  ;       RETURN
4280
4281  014666  016646  000002          LINOCT: MOV     2(SP),-(SP)     ;PUT NUMBER IN PROPER LOCATION ON STACK
4282  014672  004737  016432                JSR     PC,$$B20        ;CONVERT THE NUMBER TO OCTAL
4283  014676  012637  014712                MOV     (SP)+,1$        ;GET THE ADDRESS OF THE ASCII STRING
4284  014702  062737  000005  014712        ADD     #5.,1$          ;ADDRESS THE LAST 6 ASCII DIGITS
4285  014710  104414                       DISPLY                  ;TYPE IT
4286  014712  000000                 1$:     .WORD   0               ;ADDRESS
4287  014714  012616                       MOV     (SP)+,(SP)      ;CORRECT THE STACK
4288  014716  000207                       RTS     PC              ;RETURN
4289
4290                                  ;ROUTINE TO CONVERT THE INPUT NUMBER TO DECIMAL AND TYPE IT WITH
4291                                  ;LEADING ZERO SUPRESSION
4292                                  ;CALL:
4293                                  ;       MOV     NUM,-(SP)       ;PUT THE NUMBER ON THE STACK
4294                                  ;       JSR     PC,LINDEC
4295                                  ;       RETURN
```

```
4296
4297  014720  016646  000002          LINDEC: MOV     2(SP),-(SP)     ;SET UP STACK FOR CONVERT
4298  014724  004737  016402                  JSR     PC,$SB2D        ;CONVERT IT TO DECIMAL
4299  014730  004737  016006                  JSR     PC,$SUPRS       ;TYPE IT (WITH LEADING ZEROS SUPRESSED)
4300  014734  012616                          MOV     (SP)+,(SP)      ;RESTORE STACK POINTER
4301  014736  000207                          RTS     PC
4302
4303                                  ;;*****************************************************************
4304
4305                                  .SBTTL  GENERAL SUPPORT SUBROUTINES
4306
4307                                  ;;*****************************************************************
4308
4309
4310                                  ;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS
4311
4312  014740  012737  177777  001314  CKCLK:  MOV     #-1,CLKFLG      ;CLEAR CLOCK AVAILABILITY FLAG
4313  014746  012737  177777  001312          MOV     #-1,PCLOCK      ;CLEAR KW11-P CLOCK AVAILABILITY FLAG
4314  014754  012737  015034  000004          MOV     #CKCLK1,ERRVEC  ;SET UP VECTOR FOR CLOCK CHECK
4315  014762  005037  000006                  CLR     @#ERRVEC+2        ;NEW PSW
4316  014766  005777  164306                  TST     @$LKCSR         ;CHECK FOR KW11-P
4317  014772  005037  001314                  CLR     CLKFLG          ;SET CLOCK AVAILABILITY FLAG
4318  014776  005037  001312                  CLR     PCLOCK          ;SET KW11-P CLOCK FLAG
4319  015002  013701  001304                  MOV     $LPVEC,R1       ;KW11-P VECTOR ADDRESS
4320  015006  012721  015234                  MOV     #CLOCK,(R1)+    ;SET UP KW11-P VECTOR
4321  015012  012711  000300                  MOV     #300,(R1)       ;PSW - PRI 6
4322  015016  012777  174575  164256          MOV     #-1667.,@$LKCSB ;LOAD COUNTER BUFFER WITH 16.67
4323  015024  012777  000131  164246          MOV     #131,@$LKCSR    ;SET CLOCK - CNT UP, 10US, CONT INT
4324  015032  000435                          BR      CKCLK3
4325  015034  062706  000004          CKCLK1: ADD     #4,SP           ;RESTORE THE STACK POINTER
4326  015040  012737  015102  000004          MOV     #CKCLK2,@#ERRVEC  ;CHANGE ERROR VECTOR TO CHECK FOR KW11-L
4327  015046  005777  164234                  TST     @$LKS           ;LOOK FOR KW11-L
4328  015052  005037  001314                  CLR     CLKFLG          ;SET CLOCK FLAG
4329  015056  013701  001310                  MOV     $LLVEC,R1       ;KW11-L VECTOR ADDRESS
4330  015062  012721  015234                  MOV     #CLOCK,(R1)+    ;SET UP KW11-L VECTOR
4331  015066  012711  000300                  MOV     #300,(R1)       ;PSW - PRI 6
4332  015072  012777  000100  164206          MOV     #100,@$LKS      ;SET KW11-L INTERRUPT
4333  015100  000412                          BR      CKCLK3
4334  015102  062706  000004          CKCLK2: ADD     #4,SP           ;RESTORE THE STACK POINTER
4335  015106  104401  036130                  TYPE    ,NEDCLK         ;'F OR L CLOCK MUST BE ON SYSTEM'
4336  015112  005737  000042                  TST     42              ;UNDER MONITOR CONTROL ?
4337  015116  001400                          BEQ     1$              ;BR IF NOT
4338  015120  000000                  1$:     HALT                    ;HALT
4339  015122  000137  003062                  JMP     START           ;TRY AGAIN
4340  015126  012737  000006  000004  CKCLK3: MOV     #6,@#ERRVEC     ;RESTORE THE ERROR VECTOR
4341  015134  000207                          RTS     PC
4342
4343
4344                                  ;ROUTINE TO TYPE THE TIME
4345
4346  015136  005737  001314          $TIME:  TST     CLKFLG          ;CLOCK ON THE SYSTEM ?
4347  015142  001033                          BNE     1$              ;BR IF NOT
4348  015144  104401  001205                  TYPE    ,$CRLF          ;CR-LF
4349  015150  013746  001342                  MOV     HOUR,-(SP)      ;PUT 'HOURS' ON THE STACK
4350  015154  004737  016402                  JSR     PC,$SB2D        ;CONVERT TO DECIMAL
4351  015160  004537  015716                  JSR     R5,REPLZ        ;TYPE IT
```

```
4352  015164  000002                          .WORD   2               ;TYPE 2 DIGITS
4353  015166  104401  036201                  TYPE    ,COLON          ;':'
4354  015172  013746  001344                  MOV     MINUTE,-(SP)    ;PUT 'MINUTES' ON THE STACK
4355  015176  004737  016402                  JSR     PC,$SB2D        ;CONVERT TO DECIMAL
4356  015202  004537  015716                  JSR     R5,REPLZ        ;TYPE IT
4357  015206  000002                          .WORD   2               ;TYPE 2 DIGITS
4358  015210  104401  036201                  TYPE    ,COLON          ;':'
4359  015214  013746  001346                  MOV     SECOND,-(SP)    ;PUT SECONDS ON THE STACK
4360  015220  004737  016402                  JSR     PC,$SB2D        ;CONVERT TO DECIMAL
4361  015224  004537  015716                  JSR     R5,REPLZ        ;TYPE IT
4362  015230  000002                          .WORD   2               ;TYPE 2 DIGITS
4363  015232  000207                  1$:     RTS     PC
4364
4365                                  ;CLOCK HANDLER ROUTINE
4366
4367  015234  005337  001350          CLOCK:  DEC     SIXTEE          ;INCREMENT THE 1/60 SECOND COUNTER
4368  015240  001033                          BNE     1$              ;BR IF A SECOND NOT COUNTED
4369  015242  013737  001316  001350          MOV     HZ,SIXTEE       ;RESTORE THE VALUE
4370  015250  005237  001346                  INC     SECOND          ;COUNT THE SECOND
4371  015254  022737  000074  001346          CMP     #60.,SECOND     ;AT MAXIMUM ?
4372  015262  001022                          BNE     1$              ;BR IF NOT
4373  015264  005037  001346                  CLR     SECOND          ;CLEAR THE SECOND'S COUNTER
4374  015270  005237  001344                  INC     MINUTE          ;COUNT THE MINUTE
4375  015274  022737  000074  001344          CMP     #60.,MINUTE     ;AT MAXIMUM ?
4376  015302  001012                          BNE     1$              ;BR IF NOT
4377  015304  005037  001344                  CLR     MINUTE          ;CLEAR THE MINUTE'S COUNTER
4378  015310  005237  001342                  INC     HOUR            ;COUNT THE HOURS
4379  015314  022737  001747  001342          CMP     #999.,HOUR      ;AT MAXIMUM
4380  015322  103002                          BHIS    1$              ;BR IF NOT
4381  015324  005037  001342                  CLR     HOUR            ;CLEAR THE HOURS
4382  015330  012746  000021          1$:     MOV     #17.,-(SP)      ;17 MS ON THE STACK
4383  015334  004737  027266                  JSR     PC,RPTMR        ;DRIVER TIMER ROUTINE
4384  015340  000002                  2$:     RTI
4385
4386                                  ;COMMAND DECODE ROUTINE
4387                                  ;CALL:
4388                                  ;       MOV     #-1,CFLAG       ;'CFLAG' IS NORMALLY SET BY THE TTY SERVICE
4389                                  ;                               ;ROUTINE IN INTERRUPT MODE
4390                                  ;       JSR     PC,KSR
4391                                  ;       RETURN1                 ;SYSTEM BUSY RETURN
4392                                  ;       RETURN2                 ;RETURN AFTER KEYBOARD SERVICED
4393
4394  015342  104412                  KSR:    SAVREG                  ;SAVE THE REGISTERS
4395  015344  012737  000200  177776          MOV     #PR4,PS         ;SET PRIORITY TO 4
4396  015352  005037  001334                  CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
4397  015356  004737  015136                  JSR     PC,$TIME        ;TYPE THE TIME
4398  015362  005777  163574                  TST     @$TKB           ;CLEAR ANY GARBAGE IN THE TTY BUFFER
4399  015366  005737  001334                  TST     CFLAG           ;CHECK THE CONTROL C FLAG
4400  015372  001002                          BNE     7$              ;EXIT IF 'CONTROL C' ENTERED
4401  015374  000240                          NOP                     ;DUMP CODING FOR LATER USE 3/8/77
4402  015376  000240                          NOP
4403  015400  104413                  7$:     RESREG                  ;RESTORE R0 - R5
4404  015402  062716  000002                  ADD     #2,(SP)         ;INCREMENT THE RETURN ADDRESS
4405  015406  005777  163550                  TST     @$TKB           ;CLEAR THE TTY BUFFER
4406  015412  052777  000100  163540          BIS     #BIT06,@$TKS    ;SET TTY INTERRUPT ENABLE
4407  015420  005037  177776                  CLR     PS              ;SET PRIORITY BACK TO ZERO
```

```
4408  015424  000207                    RTS     PC              ;RETURN
4409
4410                                    ;ROUTINE TO CLEAR THE DPB FOR THE ASSIGNED DRIVE
4411                                    ;CALL:
4412                                    ;       MOV     #DPB,R0         ;DPB ADDRESS
4413                                    ;       JSR     PC,CLRDPB
4414                                    ;       RETURN
4415
4416  015426                            CLRDPB:
4417  015426  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
4418  015430  010346                            MOV     R3,-(SP)        ;;PUSH R3 ON STACK
4419  015432  010446                            MOV     R4,-(SP)        ;;PUSH R4 ON STACK
4420  015434  010546                            MOV     R5,-(SP)        ;;PUSH R5 ON STACK
4421  015436  010004                            MOV     R0,R4           ;GET THE DPB ADDRESS
4422  015440  062704  000002                    ADD     #2,R4           ;ADDRESS OF FIRST LOCN TO BE CLEARED
4423  015444  012703  000020                    MOV     #$EMTAB-2,R3    ;NUMBER OF LOCATIONS TO CLEAR
4424  015450  005024                    1$:     CLR     (R4)+           ;CLEAR THE STORAGE LOCATION
4425  015452  162703  000002                    SUB     #2,R3           ;DECREMENT THE BYTE COUNT
4426  015456  001374                            BNE     1$              ;LOOPING BACK
4427  015460  012605                            MOV     (SP)+,R5        ;;POP STACK INTO R5
4428  015462  012604                            MOV     (SP)+,R4        ;;POP STACK INTO R4
4429  015464  012603                            MOV     (SP)+,R3        ;;POP STACK INTO R3
4430  015466  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
4431  015470  000207                            RTS     PC              ;RETURN
4432
4433  015472  104412                    LINKDV: SAVREG                  ;STORE R0 - R5
4434  015474  016605  000026                    MOV     26(SP),R5       ;DIVISOR
4435  015500  005004                            CLR     R4              ;OTHER DIVISOR WORD
4436  015502  016602  000030                    MOV     30(SP),R2       ;UPPER DIVIDEND WORD
4437  015506  016603  000032                    MOV     32(SP),R3       ;LOWER DIVIDEND WORD
4438  015512  005000                            CLR     R0              ;CLEAR OTHER DIVIDEND REGISTERS
4439  015514  005001                            CLR     R1
4440  015516  004737  015540                    JSR     PC,M.DPID       ;GO TO THE DIVIDE ROUTINE
4441  015522  010166  000030                    MOV     R1,30(SP)       ;REMAINDER ON THE STACK
4442  015526  010366  000032                    MOV     R3,32(SP)       ;QUOTIENT ON THE STACK
4443  015532  104413                            RESREG                  ;RESTORE R0 - R5
4444  015534  012616                            MOV     (SP)+,(SP)      ;MOVE RETURN UP THE STACK
4445  015536  000207                            RTS     PC
4446
4447                                    ;       DIVISION UTILITY SUBROUTINE
4448                                    ;       R0-R1-R2-R3=DIVIDEND
4449                                    ;       R4-R5=DIVISOR
4450                                    ;       R0-R1=REMAINDER AFTER DIVISION
4451                                    ;       R2-R3=QUOTIENT AFTER DIVISION
4452                                    ;       ENTER WITH JSR  PC,M.DPID
4453                                    ;
4454  015540  012746  000040           M.DPID: MOV     #40,-(SP)       ;COUNTER FOR DIVISION CYCLES
4455  015544  010446                            MOV     R4,-(SP)        ;HIGH ORDER
4456  015546  010546                            MOV     R5,-(SP)        ;LOW ORDER DIVISOR TO THE STACK
4457  015550  005466  000002                    NEG     2(SP)           ;FORM NEGATIVE
4458  015554  005416                            NEG     @SP             ;VERSION OF THE DIVISOR
4459  015556  005666  000002                    SBC     2(SP)
4460  015562  061601                            ADD     @SP,R1
4461  015564  005500                            ADC     R0              ;PERFORM THE INITIAL SUBTRACTION
4462  015566  066600  000002                    ADD     2(SP),R0
4463  015572  103445                            BCS     M.DP50          ;IF CARRY THEN OVERFLOW HAS OCCURRED
```

```
4464  015574  005046                          CLR     -(SP)           ;THIS IS A LONGER LASTING CARRY BIT
4465  015576  006103                M.DP40:   ROL     R3
4466  015600  006102                          ROL     R2
4467  015602  006101                          ROL     R1
4468  015604  006100                          ROL     R0
4469  015606  005716                          TST     @SP             ;TEST "CARRY" INDICATOR
4470  015610  001410                          BEQ     M.DP41          ;IF NO "CARRY" THEN ADD ELSE SUBTRACT
4471  015612  005016                          CLR     @SP             ;CLEAR UP FOR NEXT TIME
4472  015614  066601  000002                ADD     2(SP),R1
4473  015620  005500                          ADC     R0              ;ADD -(DIVISOR)
4474  015622  005516                          ADC     @SP     ;  I    ;SET "CARRY"
4475  015624  066600  000004                ADD     4(SP),R0;<-
4476  015630  000404                          BR      M.DP42
4477  015632  060501                M.DP41:   ADD     R5,R1
4478  015634  005500                          ADC     R0              ;ADD +(DIVISOR)
4479  015636  005516                          ADC     @SP     ;  I    ;SET "CARRY"
4480  015640  060400                          ADD     R4,R0   ;<-
4481  015642  005516                M.DP42:   ADC     @SP             ;SET "CARRY"
4482  015644  005716                          TST     @SP             ;TEST THE UPDATE INDICATOR
4483  015646  001401                          BEQ     .+4     ;->     ;IF ZERO FORGET IT
4484  015650  005203                          INC     R3      ;  I    ;NO CARRY POSSIBLE HERE
4485  015652  005366  000006                DEC     6(SP)   ;<-     ;DECREMENT COUNTER
4486  015656  003347                          BGT     M.DP40          ;BRANCH IF MORE TO DO
4487  015660  006003                          ROR     R3
4488  015662  103404                          BCS     M.DP44
4489  015664  060501                          ADD     R5,R1
4490  015666  005500                          ADC     R0
4491  015670  060400                          ADD     R4,R0
4492  015672  000241                          CLC
4493  015674  006103                M.DP44:   ROL     R3
4494  015676  062706  000010                ADD     #10,SP          ;ADJUST STACK BY 4 WORDS
4495  015702  000242                          CLV
4496  015704  000207                          RTS     PC
4497  015706  062706  000006        M.DP50:   ADD     #6,SP
4498  015712  000262                          SEV
4499  015714  000207                          RTS     PC
4500
4501
4502                                  ;ROUTINE TO REPLACE LEADING ZEROS IN A NUMERIC STRING WITH SPACES
4503                                  ;CALL
4504                                  ;       MOV     #ADR,-(SP)      ;ADDRESS OF NUMBER (IN ASCII)
4505                                  ;       JSR     R5,REPLZ
4506                                  ;       .WORD   N               ;'N' IS NUMBER OF DIGITS TO BE TYPED
4507
4508  015716  010046                REPLZ:    MOV     R0,-(SP)        ;SAVE R0
4509  015720  012746  000012                MOV     #10.,-(SP)      ;MAXIMUM NUMBER OF DIGITS TO BE TYPED
4510  015724  162516                          SUB     (R5)+,(SP)      ;SUBTACT DIGITS TO FORM INDEX
4511  015726  016600  000006                MOV     6(SP),R0        ;ADDRESS OF NUMBER TO R0
4512  015732  122710  000060        1$:       CMPB    #'0,(R0)        ;BYTE EQUAL TO ASCII '0' ?
4513  015736  001004                          BNE     2$              ;BR IF NOT
4514  015740  112710  000040                MOVB    #40,(R0)        ;REPLACE THE ZERO WITH A SPACE
4515  015744  005200                          INC     R0              ;INCREMENT THE BYTE ADDRESS
4516  015746  000771                          BR      1$              ;GO BACK AND LOOK FOR MORE LEADING ZEROS
4517  015750  105710                2$:       TSTB    (R0)            ;SEE IF ZERO BYTE TERMINATOR
4518  015752  001003                          BNE     3$              ;BR IF NOT
4519  015754  005300                          DEC     R0              ;BACKUP STRING POINTER
```

```
4520  015756  112710  000060                  MOVB    #'0,(R0)        ;PUT A ZERO BACK IN
4521  015762  016637  000006  015776  3$:     MOV     6(SP),4$        ;PUT ADDRESS IN LOCATION FOR TYPEOUT
4522  015770  062637  015776                  ADD     (SP)+,4$        ;BEGINNING OF SIGNIFICANT DIGITS
4523  015774  104401                          TYPE                    ;TYPE THE NUMBER
4524  015776  000000                  4$:     .WORD   0               ;ADDRESS OF NUMBER
4525  016000  012600                          MOV     (SP)+,R0        ;RESTORE R0
4526  016002  012616                          MOV     (SP)+,(SP)      ;MOVE RETURN ADDRESS
4527  016004  000205                          RTS     R5              ;RETURN
4528
4529                                  ;TYPE NUMERICAL ASCIZ STRING SUPRESS LEADING ZEROS
4530
4531                                  ;CALL
4532                                  ;       MOV     #NUMADR,-(SP)   ;FIRST ADDRESS OF ASCIZ STRING
4533                                  ;       JSR     PC,$SUPRS
4534
4535  016006  010046                  $SUPRS: MOV     R0,-(SP)        ;SAVE R0
4536  016010  016600  000004                  MOV     4(SP),R0        ;PICKUP THE POINTER
4537  016014  105710                  1$:     TSTB    (R0)            ;TERMINATOR ?
4538  016016  001403                          BEQ     2$              ;BR IF YES
4539  016020  122720  000060                  CMPB    #'0,(R0)+       ;IS THIS AN ASCII '0' ?
4540  016024  001773                          BEQ     1$              ;BR IF YES
4541  016026  005300                  2$:     DEC     R0              ;BACKUP BY '1'
4542  016030  010037  016036                  MOV     R0,3$           ;SAVE FOR TYPING
4543  016034  104414                          DISPLY                  ;GO PRINT
4544  016036  000000                  3$:     .WORD   0               ;ASCIZ POINTER GOES HERE
4545  016040  012600                          MOV     (SP)+,R0        ;RESTORE R0
4546  016042  012616                          MOV     (SP)+,(SP)      ;RESTORE THE STACK
4547  016044  000207                          RTS     PC              ;RETURN
4548
4549                                  ;ROUTINE TO TYPE AT PRIORITY 4
4550
4551  016046  013746  177776          TYPRI4: MOV     @#PS,-(SP)      ;SAVE THE PRESENT STATUS
4552  016052  012737  000200  177776          MOV     #200,@#PS       ;CHANGE THE PRIORITY TO 4
4553  016060  012537  016070                  MOV     (R5)+,1$        ;MESSAGE ADDRESS
4554  016064  004737  017722                  JSR     PC,$TYPE        ;TYPE THE MESSAGE
4555  016070  000000                  1$:     .WORD   0               ;MESSAGE ADDRESS GOES HERE
4556  016072  000205                          RTS     R5              ;RETURN
4557
4558                                  ;ROUTINE TO TYPE ERRORS
4559                                  ;CALL
4560                                  ;       DISPLY                  ;MUST DEFINED IN 'TRAP' TABLE
4561                                  ;       MESADR                  ;ADDRESS OF MESSAGE
4562                                  ;       RETURN
4563
4564  016074  032777  020000  163052  $DSPLY: BIT     #BIT13,@SWR     ;INHIBIT ERROR TYPEOUT ?
4565  016102  001002                          BNE     1$              ;BR IF YES
4566  016104  000137  017722                  JMP     $TYPE           ;TYPE THE MESSAGE
4567  016110  062716  000002          1$:     ADD     #2,(SP)         ;INCREMENT THE RETURN
4568  016114  000002                          RTI                     ;RETURN
4569
4570                                  ;THIS ROUTINE IS USED TO CHECK IF AN
4571                                  ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
4572                                  ;CALL
4573                                  ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
4574                                  ;       JSR     R5,CK.OCT       ;CHECK THE CHARACTER
4575                                  ;       RETURN1                 ;CHARACTER IS NOT BETWEEN 0-7
```

```
                                ;       RETURN2                 ;CHARACTER IS IN R2 AS A
                                ;                               ;OCTAL DIGIT
                                
016116  121127  000060          CK.OCT: CMPB    (R1),#'0        ;LESS THAN ZERO?
016122  103407                          BLO     1$              ;YES -- BRANCH
016124  121127  000067                  CMPB    (R1),#'7        ;GREATER THAN SEVEN?
016130  101004                          BHI     1$              ;YES -- BRANCH
016132  111102                          MOVB    (R1),R2         ;GET THE CHARACTER
016134  042702  177770                  BIC     #^C7,R2         ;STRIP AWAY THE ASCII
016140  005725                          TST     (R5)+           ;ADJUST FOR RETURN
016142  000205                  1$:     RTS     R5              ;RETURN

                                ;THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
                                ;AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
                                ;CALL
                                ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
                                ;       JSR     R5,CK.DEC       ;CHECK THE CHARACTER
                                ;       RETURN1                 ;NOT BETWEEN 0 AND 9
                                ;       RETURN2                 ;BETWEEN 0 AND 9
                                ;                               ;R2 = DIGIT

016144  121127  000060          CK.DEC: CMPB    (R1),#'0        ;LESS THAN ZERO?
016150  103407                          BLO     1$              ;YES -- BRANCH
016152  121127  000071                  CMPB    (R1),#'9        ;GREATER THAN NINE?
016156  101004                          BHI     1$              ;YES -- BRANCH
016160  111102                          MOVB    (R1),R2         ;GET THE CHARACTER
016162  042702  000060                  BIC     #'0,R2          ;STRIP AWAY THE ASCII
016166  005725                          TST     (R5)+           ;ADJUST FOR RETURN
016170  000205                  1$:     RTS     R5              ;RETURN

                                ;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

016172  105711                  CK.CHR: TSTB    (R1)            ;"CARRIAGE RETURN"?
016174  001417                          BEQ     3$              ;YES -- BRANCH
016176  121127  000054                  CMPB    (R1),#',        ;"COMMA"?
016202  001413                          BEQ     2$              ;YES -- BRANCH
016204  121127  000056                  CMPB    (R1),#'.        ;"PERIOD"?
016210  001407                          BEQ     1$              ;YES -- BRANCH
016212  004537  016144                  JSR     R5,CK.DEC       ;"DIGIT"?
016216  000410                          BR      4$              ;NO -- BRANCH
016220  004537  016116                  JSR     R5,CK.OCT       ;OCTAL ?
016224  005725                          TST     (R5)+           ;DIGIT BETWEEN 8-9
016226  005725                          TST     (R5)+           ;DIGIT BETWEEN 0-7
016230  005725                  1$:     TST     (R5)+           ;PERIOD
016232  005725                  2$:     TST     (R5)+           ;COMMA
```

```
4632  016234  005725                  3$:     TST     (R5)+           ;CARRIAGE RETURN
4633  016236  005201                          INC     R1              ;MOVE POINTER TO NEXT CHARACTER
4634  016240  011505                  4$:     MOV     (R5),R5         ;UNKNOWN CHARACTER
4635  016242  000205                          RTS     R5              ;RETURN
4636
4637                                  ;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
4638                                  ;CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
4639                                  ;CALL
4640                                  ;       MOV     #ADR,R1         ;ADDRESS OF ASCIZ STRING
4641                                  ;       MOV     #NUM,R2         ;MAX. MAGNITUDE OF INPUT NUMBER
4642                                  ;       JSR     R5,CK.DIG       ;CHECK DIGITS
4643                                  ;       RETURN  ADR1            ;"CR" ONLY ENTERED -- R2=0
4644                                  ;       RETURN  ADR2            ;"PERIOD" ONLY ENTERED -- R2=0
4645                                  ;       RETURN  ADR3            ;ILLEGAL CHARACTER OR INPUT TOO LARGE -- R2=?
4646                                  ;       RETURN  ADR4            ;"CR" -- R2 = NUMBER
4647                                  ;       RETURN  ADR5            ;"COMMA" -- R2 = NUMBER
4648                                  ;       RETURN  ADR6            ;"PERIOD" -- R2 = NUMBER
4649
4650  016244  010446                  CK.DIG: MOV     R4,-(SP)        ;SAVE R4
4651  016246  010346                          MOV     R3,-(SP)        ;SAVE R3
4652  016250  010246                          MOV     R2,-(SP)        ;SAVE THE MAX. SIZE ON THE STACK
4653  016252  005002                          CLR     R2              ;START WITH 0
4654  016254  005003                          CLR     R3
4655  016256  005004                          CLR     R4
4656  016260  004537  016172                  JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
4657  016264  016360                          6$                      ;ILLEGAL CHARACTER
4658  016266  016366                          9$                      ;CARRIAGE RETURN
4659  016270  016360                          6$                      ;","
4660  016272  016362                          7$                      ;"."
4661  016274  016300                          1$                      ;DIGIT 0-7
4662  016276  016300                          1$                      ;DIGIT 8-9
4663  016300  062705  000004          1$:     ADD     #4,R5           ;STEP RETURN POINTER PAST "CR" & "PERIOD" RETURNS
4664  016304  006303          2$:     ASL     R3                      ;INPUT NUMBER *2
4665  016306  010346                  MOV     R3,-(SP)                ;SAVE *2
4666  016310  006303                  ASL     R3                      ;*4
4667  016312  006303                  ASL     R3                      ;*8
4668  016314  062603                  ADD     (SP)+,R3                ;(*2)+(*8) = *10
4669  016316  060203                  ADD     R2,R3                   ;UPDATE THE INPUT NUMBER
4670  016320  004537  016172          JSR     R5,CK.CHR               ;CHECK ONE CHARACTER
4671  016324  016364                  8$                              ;ILLEGAL CHARACTER
4672  016326  016350                  5$                              ;CARRIAGE RETURN
4673  016330  016346                  4$                              ;","
4674  016332  016340                  3$                              ;"."
4675  016334  016304                  2$                              ;DIGIT 0-7
4676  016336  016304                  2$                              ;DIGIT 8-9
4677  016340  105711          3$:     TSTB    (R1)                    ;DOES A "CR" FOLLOW THE "PERIOD"
4678  016342  001010                  BNE     8$                      ;BR IF NOT
4679  016344  005724                  TST     (R4)+                   ;INCREMENT THE RETURN
4680  016346  005724          4$:     TST     (R4)+                   ;INCREMENT THE RETURN
4681  016350  005724          5$:     TST     (R4)+                   ;INCREMENT THE RETURN
4682  016352  020316                  CMP     R3,(SP)                 ;CHECK THE MAGNITUDE OF THE NUMBER
4683  016354  101004                  BHI     9$                      ;BR IF ENTERED NUMBER TOO LARGE
4684  016356  000402                  BR      8$                      ;BYPASS INCREMENT
4685  016360  005725          6$:     TST     (R5)+                   ;INCREMENT RETURN PAST INVALID RETURN
4686  016362  005725          7$:     TST     (R5)+                   ;INCREMENT RETURN
4687  016364  060405          8$:     ADD     R4,R5                   ;SETUP RETURN POINTER
```

```
                                        ;
016366  010302                  9$:     MOV     R3,R2           ;ENTERED VALUE
016370  005726                          TST     (SP)+           ;CLEAN MAX. SIZE OFF OF STACK
016372  012603                          MOV     (SP)+,R3        ;RESTORE R3
016374  012604                          MOV     (SP)+,R4        ;RESTORE R4
016376  011505                          MOV     (R5),R5         ;GET RETURN ADDRESS
016400  000205                          RTS     R5              ;RETURN
                                        ;
                                        ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
                                        ;UNSIGNED DECIMAL ASCIZ NUMBER.
                                        ;CALL
                                        ;       MOV     NUMBER,-(SP)    ;PUT THE NUMBER ON THE STACK
                                        ;       JSR     PC,$SB2D        ;CALL
                                        ;       RETURN                  ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
                                        ;
                                        ;NOTE:  THE PROGRAM REQUIRES THIS FORM OF '$SB2D', NOT THE VERSION ON
                                        ;       THE SYSMAC LIBRARY, REV C AND LATER
                                        ;
016402  016637  000002  016426  $SB2D:  MOV     2(SP),1$        ;SAVE THE BINARY NUMBER
016410  012746  016426                  MOV     #1$,-(SP)       ;SET THE POINTER
016414  004737  022046                  JSR     PC,$DB2D        ;CALL THE DOUBLE LENGTH CONVERT
016420  012666  000002                  MOV     (SP)+,2(SP)     ;PICKUP THE POINTER
016424  000207                          RTS     PC              ;RETURN
016426  000000  000000          1$:     .WORD   0,0
                                        ;
                                        ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
                                        ;UNSIGNED OCTAL ASCIZ NUMBER.
                                        ;CALL
                                        ;       MOV     NUMBER,-(SP)    ;PUT THE NUMBER ON THE STACK
                                        ;       JSR     PC,$SB2O        ;CALL
                                        ;       RETURN                  ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
                                        ;
                                        ;NOTE:  THE PROGRAM REQUIRES THIS FORM OF '$SB2O', NOT THE VERSION ON
                                        ;       THE SYSMAC LIBRARY, REV C AND LATER
                                        ;
016432  016637  000002  016456  $SB2O:  MOV     2(SP),1$        ;SAVE THE BINARY NUMBER
016440  012746  016456                  MOV     #1$,-(SP)       ;SET THE POINTER
016444  004737  022242                  JSR     PC,$DB2O        ;CALL THE DOUBLE LENGTH CONVERT
016450  012666  000002                  MOV     (SP)+,2(SP)     ;PICKUP THE POINTER
016454  000207                          RTS     PC              ;RETURN
016456  000000  000000          1$:     .WORD   0,0
                                        ;
                                        ;KEYBOARD INTERRUPT INITIALIZATION ROUTINE
                                        ;CALL
                                        ;       JSR     PC,$TKINT
                                        ;       RETURN
                                        ;
016462  012737  016512  000060  $TKINT: MOV     #$TKSRV,TKVEC   ;SETUP VECTOR
016470  012737  000240  000062          MOV     #PR5,TKVEC+2    ;PRIORITY TO 5
016476  005777  162460                  TST     @$TKB           ;CLEAR THE BUFFER
016502  012777  000100  162450          MOV     #BIT06,@$TKS    ;SET INTERRUPT ENABLE
016510  000207                          RTS     PC              ;RETURN
                                        ;
                                        ;KEYBOARD INTERRUPT SERVICE ROUTINE
                                        ;CALL
                                        ;       ENTER VIA INTERRUPT
                                        ;
```

```
4744  016512  104410                  $TKSRV: RDCHR                   ;READ THE KEYBOARD
4745  016514  112637  016642                  MOVB    (SP)+,5$        ;GET THE CHARACTER
4746  016520  023727  016642  000003          CMP     5$,#3           ;'CONTROL C' ?
4747  016526  001012                          BNE     1$              ;BR IF NOT
4748  016530  104401  001205                  TYPE    ,$CRLF          ;CR-LF
4749  016534  104401  017146                  TYPE    ,$CNTLC         ;'↑C'
4750  016540  012737  177777  001334          MOV     #-1,CFLAG       ;SET THE 'CONTROL C' FLAG
4751  016546  005077  162406                  CLR     @$TKS           ;CLEAR THE TTY INTERRUPT
4752  016552  000432                          BR      4$              ;EXIT
4753  016554  023727  001154  000176  1$:     CMP     SWR,#SWREG      ;SOFTWARE SWITCH REGISTER IN USE ?
4754  016562  001024                          BNE     3$              ;BR IF NOT
4755  016564  023727  016642  000007          CMP     5$,#7           ;'CONTROL G' ?
4756  016572  001020                          BNE     3$              ;BR IF NOT
4757  016574  104401  001205                  TYPE    ,$CRLF          ;CR-LF
4758  016600  104401  021723                  TYPE    ,$CNTLG         ;'↑G'
4759  016604  013746  177776                  MOV     PS,-(SP)        ;PUT THE STATUS WORD ON THE STACK
4760  016610  012746  016624                  MOV     #2$,-(SP)       ;RETURN ADDRESS
4761  016614  005077  162340                  CLR     @$TKS           ;CLEAR THE TTY INTERRUPT ENABLE
4762  016620  000137  021364                  JMP     $GTSWR          ;GET THE SWITCH REGISTER ENTRY
4763  016624  012777  000100  162326  2$:     MOV     #100,@$TKS      ;ENABLE TTY KEYBOARD INTERRUPT
4764  016632  000402                          BR      4$              ;EXIT
4765  016634  104401  016642          3$:     TYPE    ,5$             ;ECHO THE CHARACTER
4766  016640  000002                  4$:     RTI                     ;RETURN
4767
4768  016642  000000                  5$:     .WORD   0               ;ENTERED CHARACTER
4769
4770                                  ;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
4771                                  ;CALL:
4772                                  ;       RDLIN                   ;;INPUT A STRING FROM THE TTY
4773                                  ;       RETURN HERE             ;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
4774                                  ;                               ;TERMINATOR WILL BE A BYTE OF ALL 0'S
4775
4776  016644  010346                  $RDLIN: MOV     R3,-(SP)        ;SAVE R3
4777  016646  005046                          CLR     -(SP)           ;CLEAR THE RUBOUT KEY
4778  016650  012703  017122          1$:     MOV     #$TTYIN,R3      ;GET ADDRESS
4779  016654  022703  017134          2$:     CMP     #$TTYIN+10.,R3  ;BUFFER FULL?
4780  016660  101467                          BLOS    4$              ;BR IF YES
4781  016662  104410                          RDCHR                   ;GO READ ONE CHARACTER FROM THE TTY
4782  016664  112613                          MOVB    (SP)+,(R3)      ;GET CHARACTER
4783  016666  122713  000177                  CMPB    #177,(R3)       ;IS IT A RUBOUT
4784  016672  001022                          BNE     5$              ;BR IF NO
4785  016674  005716                          TST     (SP)            ;IS THIS THE FIRST RUBOUT?
4786  016676  001007                          BNE     6$              ;BR IF NO
4787  016700  112737  000134  017120          MOVB    #'\,9$          ;TYPE A BACK SLASH
4788  016706  104401  017120                  TYPE    ,9$
4789  016712  012716  177777                  MOV     #-1,(SP)        ;SET THE RUBOUT KEY
4790  016716  005303                  6$:     DEC     R3              ;BACKUP BY ONE
4791  016720  020327  017122                  CMP     R3,#$TTYIN      ;STACK EMPTY?
4792  016724  103445                          BLO     4$              ;BR IF YES
4793  016726  111337  017120                  MOVB    (R3),9$         ;SETUP TO TYPEOUT THE DELETED CHAR.
4794  016732  104401  017120                  TYPE    ,9$             ;GO TYPE
4795  016736  000746                          BR      2$              ;GO READ ANOTHER CHAR.
4796  016740  005716                  5$:     TST     (SP)            ;RUBOUT KEY SET?
4797  016742  001406                          BEQ     7$              ;BR IF NO
4798  016744  112737  000134  017120          MOVB    #'\,9$          ;TYPE A BACK SLASH
4799  016752  104401  017120                  TYPE    ,9$
```

```
4800  016756  005016                        CLR     (SP)            ;CLEAR THE RUBOUT KEY
4801  016760  122713  000025        7$:     CMPB    #25,(R3)        ;IS CHARACTER A CTRL U?
4802  016764  001003                        BNE     10$             ;BR IF NO
4803  016766  104401  021716                TYPE    ,$CNTLU         ;TYPE A CONTROL "U"
4804  016772  000726                        BR      1$              ;GO START OVER
4805  016774  122713  000003        10$:    CMPB    #3,(R3)         ;IS CHARACTER A CTRL C ?
4806  017000  001006                        BNE     8$              ;BR IF NOT
4807  017002  012737  177777  001334        MOV     #-1,CFLAG       ;SET CNTRL C FLAG
4808  017010  104401  017146                TYPE    ,$CNTLC         ;ECHO IT
4809  017014  000427                        BR      11$             ;EXIT
4810  017016  122713  000012        8$:     CMPB    #12,(R3)        ;IS CHARACTER A "LF"?
4811  017022  001011                        BNE     3$              ;BRANCH IF NO
4812  017024  105013                        CLRB    (R3)            ;CLEAR THE CHARACTER
4813  017026  104401  001205                TYPE    ,$CRLF          ;TYPE A "CR" & "LF"
4814  017032  104401  017122                TYPE    ,$TTYIN         ;TYPE THE INPUT STRING
4815  017036  000706                        BR      2$              ;GO PICKUP ANOTHER CHACTER
4816  017040  104401  001204        4$:     TYPE    ,$QUES          ;TYPE A '?'
4817  017044  000701                        BR      1$              ;CLEAR THE BUFFER AND LOOP
4818  017046  111337  017120        3$:     MOVB    (R3),9$         ;ECHO THE CHARACTER
4819  017052  104401  017120                TYPE    ,9$
4820  017056  122723  000015                CMPB    #15,(R3)+       ;CHECK FOR RETURN
4821  017062  001274                        BNE     2$              ;LOOP IF NOT RETURN
4822  017064  105063  177777                CLRB    -1(R3)          ;CLEAR RETURN (THE 15)
4823  017070  104401  001206                TYPE    ,$LF            ;TYPE A LINE FEED
4824  017074  005726                11$:    TST     (SP)+           ;CLEAN RUBOUT KEY FROM THE STACK
4825  017076  012603                        MOV     (SP)+,R3        ;RESTORE R3
4826  017100  011646                        MOV     (SP),-(SP)      ;ADJUST THE STACK AND PUT ADDRESS OF THE
4827  017102  016666  000004  000002        MOV     4(SP),2(SP)     ;       FIRST ASCII CHARACTER ON IT
4828  017110  012766  017122  000004        MOV     #$TTYIN,4(SP)
4829  017116  000002                        RTI                     ;RETURN
4830  017120     000                9$:     .BYTE   0               ;STORAGE FOR ASCII CHAR. TO TYPE
4831  017121     000                        .BYTE   0               ;TERMINATOR
4832  017122  000024                $TTYIN: .BLKB   20.             ;RESERVE 20 BYTES FOR TTY INPUT
4833  017146  041536  005015  000   $CNTLC: .ASCIZ  /^C/<CR><LF>    ;CONTROL "C"
4834
4835          017154                        .EVEN
4836
4837
4838
4839  017154  005737  001374        FINISH: TST     FAULT           ;COMPATIBLE ?
4840  017160  001006                        BNE     1$              ;BRANCH IF NOT
4841  017162  104401  001205                TYPE    ,$CRLF
4842  017166  104401  001205                TYPE    ,$CRLF
4843  017172  104401  035442                TYPE    ,MESG15         ;MESSAGE: ALL DRIVE COMPATIBLE
4844  017176  104401  001205        1$:     TYPE    ,$CRLF
4845  017202  104401  001205                TYPE    ,$CRLF
4846  017206  104401  036375                TYPE    ,MESG20
4847  017212  104401  001205                TYPE    ,$CRLF
4848  017216  000000                        HALT
4849  017220  000137  000200                JMP     @#200           ;BRANCH TO 200 START IF DESIRED
4850
4851                                        .SBTTL  END OF PASS ROUTINE
4852
4853                                        ;;*****************************************************************
4854                                        ;*INCREMENT THE PASS NUMBER ($PASS)
4855                                        ;*IF SW12=1 INHIBIT TRACE TRAP
```

```
                                        ;*IF THERES A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO FINISH

017224                                  $EOP:
017224  000240                                NOP
017226  005037  001116                      CLR     $TSTNM          ;;ZERO THE TEST NUMBER
017232  005037  001174                      CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
017236  005237  001216                      INC     $PASS           ;;INCREMENT THE PASS NUMBER
017242  042737  100000  001216              BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
017250  005327                            1$: DEC     (PC)+           ;;LOOP?
017252  000001                      $EOPCT: .WORD   1
017254  003022                              BGT     $DOAGN          ;;YES
017256  012737                              MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
017260  000001                      $ENDCT: .WORD   1
017262  017252                              $EOPCT
017264  013700  000042              $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
017270  001414                              BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
017272  005046                              CLR     -(SP)           ;;INSURE THE "T" BIT IS CLEAR
017274  012746  017302                      MOV     #$CLR.T,-(SP)   ;;SETUP FOR AN RTI OR RTT
017300  000426                              BR      $RTRN           ;;GO DO AN RTI OR RTT TO LOAD THE PSW
                                                                    ;;WITH A CLEARED "T" BIT
017302                                  $CLR.T:
017302  013700  000042                      MOV     @#42,R0         ;;INSURE R0 CONTAINS THE MONITORS
017306  001405                              BEQ     $DOAGN          ;;RETURN ADDRESS
017310  000005                              RESET                   ;;CLEAR THE WORLD
017312  004710                      $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
017314  000240                              NOP                     ;;SAVE ROOM
017316  000240                              NOP                     ;;FOR
017320  000240                              NOP                     ;;ACT11
017322                                  $DOAGN:
017322  104400                              TRAP                    ;;PUSH OLD PSW AND PC ON STACK
017324  042716  000020                      BIC     #20,(SP)        ;;CLEAR THE "T" BIT
017330  032777  010000  161616              BIT     #BIT12,@SWR     ;;RUN WITH TRACE TRAP?
017336  001005                              BNE     1$              ;;BR IF NO
017340  005137  017364                      COM     $TBIT           ;;IS IT TIME FOR TRACE TRAP
017344  100402                              BMI     1$              ;;BR IF NO
017346  052716  000020                      BIS     #20,(SP)        ;;SET TRACE TRAP
017352  012746  017360              1$:     MOV     #$LOOP,-(SP)    ;;JUMP TO START OF TEST
017356  000002                      $RTRN:  RTI                     ;;RETURN--THIS IS CHANGED TO
                                                                    ;;AN "RTT" IF "RTT" IS A LEGAL
                                                                    ;;INSTRUCTION
017360                                  $LOOP:
017360  000137                              JMP     @(PC)+          ;;RETURN
017362  017154                      $RTNAD: .WORD   FINISH
017364  000000                      $TBIT:  .WORD   0               ;;"T" BIT STATE INDICATOR

                                        .SBTTL  ERROR HANDLER ROUTINE

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

```
                                        ;*SW09=1        LOOP ON ERROR
                                        ;*CALL
                                        ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER

017366                                  $ERROR:
017366  104407                                  CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
017370  105237  001117                  7$:     INCB    $ERFLG                  ;;SET THE ERROR FLAG
017374  001775                                  BEQ     7$                      ;;DON'T LET THE FLAG GO TO ZERO
017376  013777  001116  161552                  MOV     $TSTNM,@DISPLAY         ;;DISPLAY TEST NUMBER AND ERROR FLAG
017404  032777  002000  161542                  BIT     #BIT10,@SWR             ;;BELL ON ERROR?
017412  001402                                  BEQ     1$                      ;;NO - SKIP
017414  104401  001200                          TYPE    ,$BELL                  ;;RING BELL
017420  005237  001126                  1$:     INC     $ERTTL                  ;;COUNT THE NUMBER OF ERRORS
017424  011637  001132                          MOV     (SP),$ERRPC             ;;GET ADDRESS OF ERROR INSTRUCTION
017430  162737  000002  001132                  SUB     #2,$ERRPC
017436  117737  161470  001130                  MOVB    @$ERRPC,$ITEMB          ;;STRIP AND SAVE THE ERROR ITEM CODE
017444  032777  020000  161502                  BIT     #BIT13,@SWR             ;;SKIP TYPEOUT IF SET
017452  001004                                  BNE     20$                     ;;SKIP TYPEOUTS
017454  004737  017566                          JSR     PC,$ERRTYP              ;;GO TO USER ERROR ROUTINE
017460  104401  001205                          TYPE    ,$CRLF
017464                                  20$:
017464  122737  000001  001230                  CMPB    #APTENV,$ENV            ;;RUNNING IN APT MODE
017472  001007                                  BNE     2$                      ;;NO,SKIP APT ERROR REPORT
017474  113737  001130  017506                  MOVB    $ITEMB,21$              ;;SET ITEM NUMBER AS ERROR NUMBER
017502  004737  020222                          JSR     PC,$ATY4                ;;REPORT FATAL ERROR TO APT
017506     000                          21$:    .BYTE   0
017507     000                                  .BYTE   0
017510  000777                          22$:    BR      22$                     ;;APT ERROR LOOP
017512  005777  161436                  2$:     TST     @SWR                    ;;HALT ON ERROR
017516  100002                                  BPL     3$                      ;;SKIP IF CONTINUE
017520  000000                                  HALT                            ;;HALT ON ERROR!
017522  104407                                  CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
017524  032777  001000  161422          3$:     BIT     #BIT09,@SWR             ;;LOOP ON ERROR SWITCH SET?
017532  001402                                  BEQ     4$                      ;;BR IF NO
017534  013716  001124                          MOV     $LPERR,(SP)             ;;FUDGE RETURN FOR LOOPING
017540  005737  001176                  4$:     TST     $ESCAPE                 ;;CHECK FOR AN ESCAPE ADDRESS
017544  001402                                  BEQ     5$                      ;;BR IF NONE
017546  013716  001176                          MOV     $ESCAPE,(SP)            ;;FUDGE RETURN ADDRESS FOR ESCAPE
017552                                  5$:
017552  022737  017312  000042                  CMP     #$ENDAD,@#42            ;;ACT-11 AUTO-ACCEPT?
017560  001001                                  BNE     6$                      ;;BRANCH IF NO
017562  000000                                  HALT                            ;;YES
017564                                  6$:
017564  000002                                  RTI                             ;;RETURN

                                        .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE

                                        ;;**************************************************************
                                        ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
                                        ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
                                        ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

017566                                  $ERRTYP:
017566  104401  001205                          TYPE    ,$CRLF                  ;;"CARRIAGE RETURN" & "LINE FEED"
017572  010046                                  MOV     R0,-(SP)                ;;SAVE R0
017574  005000                                  CLR     R0                      ;;PICKUP THE ITEM INDEX
```

```
4968  017576  153700  001130          BISB    @#$ITEMB,R0
4969  017602  001004                  BNE     1$                      ;;IF ITEM NUMBER IS ZERO, JUST
4970                                                                  ;;TYPE THE PC OF THE ERROR
4971  017604  013746  001132          MOV     $ERRPC,-(SP)            ;;SAVE $ERRPC FOR TYPEOUT
4972                                                                  ;;ERROR ADDRESS
4973  017610  104402                  TYPOC                           ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
4974  017612  000426                  BR      6$                      ;;GET OUT
4975  017614  005300          1$:     DEC     R0                      ;;ADJUST THE INDEX SO THAT IT WILL
4976  017616  006300                  ASL     R0                      ;;      WORK FOR THE ERROR TABLE
4977  017620  006300                  ASL     R0
4978  017622  006300                  ASL     R0
4979  017624  062700  002742          ADD     #$ERRTB,R0              ;;FORM TABLE POINTER
4980  017630  012037  017640          MOV     (R0)+,2$                ;;PICKUP "ERROR MESSAGE" POINTER
4981  017634  001404                  BEQ     3$                      ;;SKIP TYPEOUT IF NO POINTER
4982  017636  104401                  TYPE                            ;;TYPE THE "ERROR MESSAGE"
4983  017640  000000          2$:     .WORD   0                       ;;"ERROR MESSAGE" POINTER GOES HERE
4984  017642  104401  001205          TYPE    ,$CRLF                  ;;"CARRIAGE RETURN" & "LINE FEED"
4985  017646  012037  017656  3$:     MOV     (R0)+,4$                ;;PICKUP "DATA HEADER" POINTER
4986  017652  001404                  BEQ     5$                      ;;SKIP TYPEOUT IF 0
4987  017654  104401                  TYPE                            ;;TYPE THE "DATA HEADER"
4988  017656  000000          4$:     .WORD   0                       ;;"DATA HEADER" POINTER GOES HERE
4989  017660  104401  001205          TYPE    ,$CRLF                  ;;"CARRIAGE RETURN" & "LINE FEED"
4990  017664  011000          5$:     MOV     (R0),R0                 ;;PICKUP "DATA TABLE" POINTER
4991  017666  001004                  BNE     7$                      ;;GO TYPE THE DATA
4992  017670  012600          6$:     MOV     (SP)+,R0                ;;RESTORE R0
4993  017672  104401  001205          TYPE    ,$CRLF                  ;;"CARRIAGE RETURN" & "LINE FEED"
4994  017676  000207                  RTS     PC                      ;;RETURN
4995  017700                  7$:
4996  017700  013046                  MOV     @(R0)+,-(SP)            ;;SAVE @(R0)+ FOR TYPEOUT
4997  017702  104402                  TYPOC                           ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
4998  017704  005710                  TST     (R0)                    ;;IS THERE ANOTHER NUMBER?
4999  017706  001770                  BEQ     6$                      ;;BR IF NO
5000  017710  104401  017716          TYPE    ,8$                     ;;TYPE TWO(2) SPACES
5001  017714  000771                  BR      7$                      ;;LOOP
5002  017716  020040      000 8$:     .ASCIZ  /  /                    ;;TWO(2) SPACES
5003          017722                  .EVEN
5004
5005                                  .SBTTL  TYPE ROUTINE
5006
5007                                  ;;*****************************************************************
5008                                  ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
5009                                  ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
5010                                  ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
5011                                  ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
5012                                  ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
5013                                  ;*
5014                                  ;*CALL:
5015                                  ;*1) USING A TRAP INSTRUCTION
5016                                  ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
5017                                  ;*OR
5018                                  ;*      TYPE
5019                                  ;*      MESADR
5020                                  ;*
5021
5022  017722  105737  001173  $TYPE:   TSTB    $TPFLG                  ;;IS THERE A TERMINAL?
5023  017726  100002                  BPL     1$                      ;;BR IF YES
```

```
5024  017730  000000                        HALT                    ;;HALT HERE IF NO TERMINAL
5025  017732  000430                        BR      3$              ;;LEAVE
5026  017734  010046                1$:     MOV     R0,-(SP)        ;;SAVE R0
5027  017736  017600  000002                MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
5028  017742  122737  000001  001230        CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
5029  017750  001011                        BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
5030  017752  132737  000100  001231        BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
5031  017760  001405                        BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
5032  017762  010037  017772                MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
5033  017766  004737  020212                JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
5034  017772  000000                61$:    .WORD   0               ;;MESSAGE ADDRESS
5035  017774  132737  000040  001231  62$:  BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
5036  020002  001003                        BNE     60$             ;;YES,SKIP TYPE OUT
5037  020004  112046                2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
5038  020006  001005                        BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
5039  020010  005726                        TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
5040  020012  012600                60$:    MOV     (SP)+,R0        ;;RESTORE R0
5041  020014  062716  000002          3$:   ADD     #2,(SP)         ;;ADJUST RETURN PC
5042  020020  000002                        RTI                     ;;RETURN
5043  020022  122716  000011          4$:   CMPB    #HT,(SP)        ;;BRANCH IF <HT>
5044  020026  001430                        BEQ     8$
5045  020030  122716  000200                CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
5046  020034  001006                        BNE     5$
5047  020036  005726                        TST     (SP)+           ;;POP  <CR><LF> EQUIV
5048  020040  104401                        TYPE                    ;;TYPE A CR AND LF
5049  020042  001205                        $CRLF
5050  020044  105037  020200                CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
5051  020050  000755                        BR      2$              ;;GET NEXT CHARACTER
5052  020052  004737  020134          5$:   JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
5053  020056  123726  001172          6$:   CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
5054  020062  001350                        BNE     2$              ;;IF NO GO GET NEXT CHAR.
5055  020064  013746  001170                MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
5056                                        ;;AND THE NULL CHAR.
5057  020070  105366  000001          7$:   DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
5058  020074  002770                        BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
5059  020076  004737  020134                JSR     PC,$TYPEC       ;;GO TYPE A NULL
5060  020102  105337  020200                DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
5061  020106  000770                        BR      7$              ;;LOOP
5062
5063                                ;HORIZONTAL TAB PROCESSOR
5064
5065  020110  112716  000040          8$:   MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
5066  020114  004737  020134          9$:   JSR     PC,$TYPEC       ;;TYPE A SPACE
5067  020120  132737  000007  020200        BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
5068  020126  001372                        BNE     9$              ;;TAB STOP
5069  020130  005726                        TST     (SP)+           ;;POP SPACE OFF STACK
5070  020132  000724                        BR      2$              ;;GET NEXT CHARACTER
5071  020134  105777  161024          $TYPEC: TSTB  @$TPS           ;;WAIT UNTIL PRINTER IS READY
5072  020140  100375                        BPL     $TYPEC
5073  020142  116677  000002  161016        MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
5074  020150  122766  000015  000002        CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
5075  020156  001003                        BNE     1$              ;;BRANCH IF NO
5076  020160  105037  020200                CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
5077  020164  000406                        BR      $TYPEX          ;;EXIT
5078  020166  122766  000012  000002  1$:   CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
5079  020174  001402                        BEQ     $TYPEX          ;;BRANCH IF YES
```

```
5080  020176  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
5081  020200  000000                $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
5082  020202  000207                $TYPEX: RTS     PC
5083
5084
5085                                .SBTTL  APT COMMUNICATIONS ROUTINE
5086
5087                                ;;**************************************************************
5088  020204  112737  000001  020450 $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
5089  020212  112737  000001  020446 $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
5090  020220  000403                        BR      $ATYC
5091  020222  112737  000001  020450 $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
5092  020230                        $ATYC:
5093  020230  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
5094  020232  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
5095  020234  105737  020446                TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
5096  020240  001450                        BEQ     5$              ;;IF NOT:  BR
5097  020242  122737  000001  001230        CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
5098  020250  001031                        BNE     3$              ;;IF NOT:  BR
5099  020252  132737  000100  001231        BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
5100  020260  001425                        BEQ     3$              ;;IF NOT:  BR
5101  020262  017600  000004                MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
5102  020266  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
5103  020274  005737  001210        1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
5104  020300  001375                        BNE     1$              ;;IF NOT:  WAIT
5105  020302  010037  001224                MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
5106  020306  105720                2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
5107  020310  001376                        BNE     2$
5108  020312  163700  001224                SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
5109  020316  006200                        ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
5110  020320  010037  001226                MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
5111  020324  012737  000004  001210        MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
5112  020332  000413                        BR      5$
5113  020334  017637  000004  020360 3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
5114  020342  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
5115  020350  013746  177776                MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
5116  020354  004737  017722                JSR     PC,$TYPE        ;;CALL TYPE MACRO
5117  020360  000000                4$:     .WORD   0
5118  020362                        5$:
5119  020362  105737  020450        10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
5120  020366  001416                        BEQ     12$             ;;IF NOT:  BR
5121  020370  005737  001230                TST     $ENV            ;;RUNNING UNDER APT?
5122  020374  001413                        BEQ     12$             ;;IF NOT:  BR
5123  020376  005737  001210        11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
5124  020402  001375                        BNE     11$             ;;IF NOT:  WAIT
5125  020404  017637  000004  001212        MOV     @4(SP),$FATAL   ;;GET ERROR #
5126  020412  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
5127  020420  005237  001210                INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
5128  020424  105037  020450        12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
5129  020430  105037  020447                CLRB    $LFLG           ;;CLEAR LOG FLAG
5130  020434  105037  020446                CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
5131  020440  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
5132  020442  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
5133  020444  000207                        RTS     PC              ;;RETURN
5134  020446     000                $MFLG:  .BYTE   0               ;;MESSG. FLAG
5135  020447     000                $LFLG:  .BYTE   0               ;;LOG FLAG
```

```
                                ;$FFLG:  .BYTE   0              ;;FATAL FLAG
                                         .EVEN
                                APTSIZE=200
                                APTENV=001
                                APTSPOOL=100
                                APTCSUP=040

                                .SBTTL  POWER DOWN AND UP ROUTINES

                                ;;**********************************************************************
                                ;POWER DOWN ROUTINE
020452  012737  020624  000024  $PWRON: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
020460  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
020466  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
020470  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
020472  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
020474  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
020476  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
020500  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
020502  017746  160446                MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
020506  010637  020630                MOV     SP,$SAVR6       ;;SAVE SP
020512  012737  020524  000024        MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
020520  000000                        HALT
020522  000776                        BR      .-2             ;;HANG UP

                                ;;**********************************************************************
                                ;POWER UP ROUTINE
020524  012737  020624  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
020532  013706  020630                MOV     $SAVR6,SP       ;;GET SP
020536  005037  020630                CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
020542  005237  020630        1$:     INC     $SAVR6          ;;WAIT FOR THE INC
020546  001375                        BNE     1$              ;;OF  WORD
020550  012677  160400                MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
020554  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
020556  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
020560  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
020562  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
020564  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
020566  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
020570  012737  020452  000024        MOV     #$PWRON,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
020576  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;PRIO:7
020604  104401                        TYPE                    ;REPORT THE POWER FAILURE
020606  020632          $PWRMG: .WORD   $POWER                  ;;POWER FAIL MESSAGE POINTER
020610  042766  000020  000002        BIC     #20,2(SP)       ;;CLEAR "T" BIT
020616  005037  017364                CLR     $TBIT           ;;CLEAR THE "T" BIT FLAG
020622  000002                        RTI
020624  000000          $ILLUP: HALT                            ;;THE POWER UP SEQUENCE WAS STARTED
020626  000776                        BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
020630  000000          $SAVR6: 0                               ;;PUT THE SP HERE
020632  005015  047520  042527  $POWER: .ASCIZ  <15><12>"POWER"
020640  000122
                                         .EVEN

                                .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

                                ;;**********************************************************************
```

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

020642  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
020646  116637  000001  021065          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
020654  112637  021067                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
020660  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
020664  000406                          BR      $TYPON
020666  112737  000001  021065  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
020674  112737  000006  021067          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
020702  112737  000005  021064  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
020710  010346                          MOV     R3,-(SP)        ;;SAVE R3
020712  010446                          MOV     R4,-(SP)        ;;SAVE R4
020714  010546                          MOV     R5,-(SP)        ;;SAVE R5
020716  113704  021067                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
020722  005404                          NEG     R4
020724  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
020730  110437  021066                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
020734  113704  021065                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
020740  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
020744  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
020746  006105          1$:     ROL     R5              ;;ROTATE MSB INTO "C"
020750  000404                  BR      3$              ;;GO DO MSB
020752  006105          2$:     ROL     R5              ;;FORM THIS DIGIT
020754  006105                  ROL     R5
020756  006105                  ROL     R5
020760  010503                  MOV     R5,R3
020762  006103          3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
020764  105337  021066          DECB    $OMODE          ;;TYPE THIS DIGIT?
020770  100016                  BPL     7$              ;;BR IF NO
020772  042703  177770          BIC     #177770,R3      ;;GET RID OF JUNK
020776  001002                  BNE     4$              ;;TEST FOR 0
021000  005704                  TST     R4              ;;SUPPRESS THIS 0?
021002  001403                  BEQ     5$              ;;BR IF YES
021004  005204          4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
021006  052703  000060          BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
021012  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
```

```
5248  021016  110337  021062              MOVB    R3,8$           ;;SAVE FOR TYPING
5249  021022  104401  021062              TYPE    ,8$             ;;GO TYPE THIS DIGIT
5250  021026  105337  021064      7$:     DECB    $OCNT           ;;COUNT BY 1
5251  021032  003347                      BGT     2$              ;;BR IF MORE TO DO
5252  021034  002402                      BLT     6$              ;;BR IF DONE
5253  021036  005204                      INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
5254  021040  000744                      BR      2$              ;;GO DO THE LAST DIGIT
5255  021042  012605              6$:     MOV     (SP)+,R5        ;;RESTORE R5
5256  021044  012604                      MOV     (SP)+,R4        ;;RESTORE R4
5257  021046  012603                      MOV     (SP)+,R3        ;;RESTORE R3
5258  021050  016666  000002  000004      MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
5259  021056  012616                      MOV     (SP)+,(SP)
5260  021060  000002                      RTI                     ;;RETURN
5261  021062     000      8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
5262  021063     000              .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
5263  021064     000      $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
5264  021065     000      $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
5265  021066  000000      $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
5266
5267                      .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
5268
5269                      ;;**************************************************************
5270                      ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
5271                      ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
5272                      ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
5273                      ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
5274                      ;*REPLACED WITH SPACES.
5275                      ;*CALL:
5276                      ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
5277                      ;*      TYPDS                   ;;GO TO THE ROUTINE
5278
5279  021070              $TYPDS:
5280  021070  010046              MOV     R0,-(SP)        ;;PUSH R0 ON STACK
5281  021072  010146              MOV     R1,-(SP)        ;;PUSH R1 ON STACK
5282  021074  010246              MOV     R2,-(SP)        ;;PUSH R2 ON STACK
5283  021076  010346              MOV     R3,-(SP)        ;;PUSH R3 ON STACK
5284  021100  010546              MOV     R5,-(SP)        ;;PUSH R5 ON STACK
5285  021102  012746  020200      MOV     #200,-(SP)      ;;SET BLANK SWITCH AND SIGN
5286  021106  016605  000020      MOV     20(SP),R5       ;;GET THE INPUT NUMBER
5287  021112  100004              BPL     1$              ;;BR IF INPUT IS POS.
5288  021114  005405              NEG     R5              ;;MAKE THE BINARY NUMBER POS.
5289  021116  112766  000055  000001  MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
5290  021124  005000      1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
5291  021126  012703  021304      MOV     #$OBLK,R3       ;;SETUP THE OUTPUT POINTER
5292  021132  112723  000040      MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
5293  021136  005002      2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
5294  021140  016001  021274      MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
5295  021144  160105      3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
5296  021146  002402              BLT     4$              ;;BR IF DONE
5297  021150  005202              INC     R2              ;;INCREASE THE BCD DIGIT BY 1
5298  021152  000774              BR      3$
5299  021154  060105      4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
5300  021156  005702              TST     R2              ;;CHECK IF BCD DIGIT=0
5301  021160  001002              BNE     5$              ;;FALL THROUGH IF 0
5302  021162  105716              TSTB    (SP)            ;;STILL DOING LEADING 0'S?
5303  021164  100407              BMI     7$              ;;BR IF YES
```

```
5304  021166  106316                  5$:     ASLB    (SP)            ;;MSD?
5305  021170  103003                          BCC     6$              ;;BR IF NO
5306  021172  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
5307  021200  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
5308  021204  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
5309  021210  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
5310  021212  005720                          TST     (R0)+           ;;JUST INCREMENTING
5311  021214  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
5312  021220  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
5313  021222  003002                          BGT     8$              ;;GO TO EXIT
5314  021224  010502                          MOV     R5,R2           ;;GET THE LSD
5315  021226  000764                          BR      6$              ;;GO CHANGE TO ASCII
5316  021230  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
5317  021232  100003                          BPL     9$              ;;BR IF NO
5318  021234  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
5319  021242  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
5320  021244  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
5321  021246  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
5322  021250  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
5323  021252  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
5324  021254  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
5325  021256  104401  021304                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
5326  021262  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
5327  021270  012616                          MOV     (SP)+,(SP)
5328  021272  000002                          RTI                     ;;RETURN TO USER
5329  021274  023420                  $DTBL:  10000.
5330  021276  001750                          1000.
5331  021300  000144                          100.
5332  021302  000012                          10.
5333  021304  000004                  $DBLK:  .BLKW   4
5334
5335                                  .SBTTL  TTY INPUT ROUTINE
5336
5337                                  ;;**********************************************************************
5338                                  .ENABL  LSB
5339
5340                                  ;;**********************************************************************
5341                                  ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
5342                                  ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
5343                                  ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
5344                                  ;*WHEN OPERATING IN TTY FLAG MODE.
5345  021314  022737  000176  001154  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
5346  021322  001074                          BNE     15$             ;;BRANCH IF NO
5347  021324  105777  157630                  TSTB    @$TKS           ;;CHAR THERE?
5348  021330  100071                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
5349  021332  117746  157624                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
5350  021336  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
5351  021342  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
5352  021346  001062                          BNE     15$             ;;NO, RETURN TO USER
5353  021350  123727  001150  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
5354  021356  001456                          BEQ     15$             ;;BRANCH IF YES
5355
5356  021360  104401  021723                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
5357  021364  104401  021730          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
5358  021370  013746  000176                  MOV     $WREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
5359  021374  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
```

```
5360  021376  104401  021741                    TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
5361  021402  005046                  19$:      CLR     -(SP)           ;;CLEAR COUNTER
5362  021404  005046                            CLR     -(SP)           ;;THE NEW SWR
5363  021406  105777  157546          7$:       TSTB    @$TKS           ;;CHAR THERE?
5364  021412  100375                            BPL     7$              ;;IF NOT TRY AGAIN
5365
5366  021414  117746  157542                    MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
5367  021420  042716  177600                    BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
5368
5369
5370
5371  021424  021627  000025          9$:       CMP     (SP),#25        ;;IS IT A CONTROL-U?
5372  021430  001005                            BNE     10$             ;;BRANCH IF NOT
5373  021432  104401  021716                    TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
5374  021436  062706  000006          20$:      ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
5375  021442  000757                            BR      19$             ;;LET'S TRY IT AGAIN
5376
5377
5378  021444  021627  000015          10$:      CMP     (SP),#15        ;;IS IT A <CR>?
5379  021450  001022                            BNE     16$             ;;BRANCH IF NO
5380  021452  005766  000004                    TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
5381  021456  001403                            BEQ     11$             ;;BRANCH IF YES
5382  021460  016677  000002  157466            MOV     2(SP),@SWR      ;;SAVE NEW SWR
5383  021466  062706  000006          11$:      ADD     #6,SP           ;;CLEAR UP STACK
5384  021472  104401  001205          14$:      TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
5385  021476  123727  001151  000001            CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
5386  021504  001003                            BNE     15$             ;;BRANCH IF NOT
5387  021506  012777  000100  157444            MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
5388  021514  000002          15$:      RTI                             ;;RETURN
5389  021516  004737  020134          16$:      JSR     PC,$TYPEC       ;;ECHO CHAR
5390  021522  021627  000060                    CMP     (SP),#60        ;;CHAR < 0?
5391  021526  002420                            BLT     18$             ;;BRANCH IF YES
5392  021530  021627  000067                    CMP     (SP),#67        ;;CHAR > 7?
5393  021534  003015                            BGT     18$             ;;BRANCH IF YES
5394  021536  042726  000060                    BIC     #60,(SP)+       ;;STRIP-OFF ASCII
5395  021542  005766  000002                    TST     2(SP)           ;;IS THIS THE FIRST CHAR
5396  021546  001403                            BEQ     17$             ;;BRANCH IF YES
5397  021550  006316                            ASL     (SP)            ;;NO, SHIFT PRESENT
5398  021552  006316                            ASL     (SP)            ;;   CHAR OVER TO MAKE
5399  021554  006316                            ASL     (SP)            ;;   ROOM FOR NEW ONE.
5400  021556  005266  000002          17$:      INC     2(SP)           ;;KEEP COUNT OF CHAR
5401  021562  056616  177776                    BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
5402  021566  000707                            BR      7$              ;;GET THE NEXT ONE
5403  021570  104401  001204          18$:      TYPE    ,$QUES          ;;TYPE ?<CR><LF>
5404  021574  000720                            BR      20$             ;;SIMULATE CONTROL-U
5405                                .DSABL  LSB
5406
5407
5408                                ;;*****************************************************************
5409                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
5410                                ;*CALL:
5411                                ;*      RDCHR                       ;;INPUT A SINGLE CHARACTER FROM THE TTY
5412                                ;*      RETURN HERE                 ;;CHARACTER IS ON THE STACK
5413                                ;*                                  ;;WITH PARITY BIT STRIPPED OFF
5414                                ;
5415
```

```
5416  021576  011646                        $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
5417  021600  016666  000004  000002                MOV     4(SP),2(SP)     ;;SAVE THE PS
5418  021606  105777  157346                1$:     TSTB    @#TKS           ;;WAIT FOR
5419  021612  100375                              BPL     1$              ;;A CHARACTER
5420  021614  117766  157342  000004              MOVB    @#TKB,4(SP)     ;;READ THE TTY
5421  021622  042766  177600  000004              BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
5422  021630  026627  000004  000023              CMP     4(SP),#23       ;;IS IT A CONTROL-S?
5423  021636  001013                              BNE     3$              ;;BRANCH IF NO
5424  021640  105777  157314                2$:     TSTB    @#TKS           ;;WAIT FOR A CHARACTER
5425  021644  100375                              BPL     2$              ;;LOOP UNTIL ITS THERE
5426  021646  117746  157310                      MOVB    @#TKB,-(SP)     ;;GET CHARACTER
5427  021652  042716  177600                      BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
5428  021656  022627  000021                      CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
5429  021662  001366                              BNE     2$              ;;IF NOT DISCARD IT
5430  021664  000750                              BR      1$              ;;YES, RESUME
5431  021666  026627  000004  000140  3$:         CMP     4(SP),#140      ;;IS IT UPPER CASE?
5432  021674  002407                              BLT     4$              ;;BRANCH IF YES
5433  021676  026627  000004  000175              CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
5434  021704  003003                              BGT     4$              ;;BRANCH IF YES
5435  021706  042766  000040  000004              BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
5436  021714  000002                        4$:     RTI                     ;;GO BACK TO USER
5437  021716  052536  005015     000  $CNTLU: .ASCIZ  /^U/<15><12>        ;;CONTROL "U"
5438  021723     136  006507  000012  $CNTLG: .ASCIZ  /^G/<15><12>        ;;CONTROL "G"
5439  021730  005015  053523  020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
5440  021736  020075     000
5441  021741     040  047040  053505  $MNEW:  .ASCIZ  /  NEW = /
5442  021746  036440  000040
5443
5444
5445                                   .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
5446
5447                                   ;;******************************************************************
5448                                   ;*SAVE R0-R5
5449                                   ;*CALL:
5450                                   ;*      SAVREG
5451                                   ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
5452                                   ;*
5453                                   ;*TOP---(+16)
5454                                   ;* +2---(+18)
5455                                   ;* +4---R5
5456                                   ;* +6---R4
5457                                   ;* +8---R3
5458                                   ;*+10---R2
5459                                   ;*+12---R1
5460                                   ;*+14---R0
5461
5462  021752                           $SAVREG:
5463  021752  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
5464  021754  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
5465  021756  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
5466  021760  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
5467  021762  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
5468  021764  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
5469  021766  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
5470  021772  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
5471  021776  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
```

```
5472  022002  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
5473  022006  000002                          RTI
5474
5475                                  ;*RESTORE R0-R5
5476                                  ;*CALL:
5477                                  ;*      RESREG
5478  022010                          $RESREG:
5479  022010  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
5480  022014  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
5481  022020  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
5482  022024  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
5483  022030  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
5484  022032  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
5485  022034  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
5486  022036  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
5487  022040  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
5488  022042  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
5489  022044  000002                          RTI
5490
5491                                  .SBTTL  DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
5492
5493                                  ;;**************************************************************
5494                                  ;*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
5495                                  ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
5496                                  ;*POSITIVE.
5497                                  ;*CALL
5498                                  ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
5499                                  ;*      JSR     PC,@#$DB2D
5500                                  ;*      RETURN                  ;;THE FIRST ADDRESS OF ASCIZ
5501                                                                  ;;IS ON THE STACK
5502
5503
5504  022046  104412                  $DB2D:  SAVREG                  ;;SAVE REGISTERS
5505  022050  016602  000002                  MOV     2(SP),R2        ;;PICKUP THE DATA POINTER
5506  022054  012700  022226                  MOV     #$DECVL,R0      ;;GET ADDRESS OF "$DECVL" STRING
5507  022060  010066  000002                  MOV     R0,2(SP)        ;;PUT ADDRESS OF ASCIZ STRING ON STACK
5508  022064  012201                          MOV     (R2)+,R1        ;;PICKUP THE BINARY NUMBER
5509  022066  012202                          MOV     (R2)+,R2
5510  022070  012737  000012  022144          MOV     #10.,4$         ;;SET UP TO DO 10 CONVERSIONS
5511  022076  012704  022156                  MOV     #$TNPWR,R4      ;;ADDRESS OF TEN POWER
5512  022102  012705  022160                  MOV     #$TNPWR+2,R5
5513  022106  005003                  1$:     CLR     R3              ;;CLEAR PARTIAL
5514  022110  161401                  2$:     SUB     (R4),R1         ;;SUBTRACT TEN POWER
5515  022112  005602                          SBC     R2
5516  022114  161502                          SUB     (R5),R2
5517  022116  002402                          BLT     3$              ;;BR IF TEN POWER TO LARGE
5518  022120  005203                          INC     R3              ;;ADD 1 TO PARTIAL
5519  022122  000772                          BR      2$              ;;LOOP
5520  022124  062401                  3$:     ADD     (R4)+,R1        ;;RESTORE SUBTRACTED VALUE
5521  022126  005502                          ADC     R2
5522  022130  062402                          ADD     (R4)+,R2
5523  022132  022525                          CMP     (R5)+,(R5)+     ;;MOVE TO NEXT TEN POWER
5524  022134  052703  000060                  BIS     #'0,R3          ;;CHANGE PARTIAL TO ASCII
5525  022140  110320                          MOVB    R3,(R0)+        ;;SAVE IT
5526  022142  005327                          DEC     (PC)+           ;;DONE?
5527  022144  000000                  4$:     .WORD   0
```

```
5528  022146  001357                          BNE     1$              ;;BR IF NO
5529  022150  105020                          CLRB    (R0)+           ;;TERMINATOR
5530  022152  104413                          RESREG                  ;;RESTORE REGISTERS
5531  022154  000207                          RTS     PC              ;;RETURN
5532  022156  145000                $TNPWR:   145000                  ;;1.0E09
5533  022160  035632                          35632
5534  022162  160400                          160400                  ;;1.0E08
5535  022164  002765                          2765
5536  022166  113200                          113200                  ;;1.0E07
5537  022170  000230                          230
5538  022172  041100                          041100                  ;;1.0E06
5539  022174  000017                          17
5540  022176  103240                          103240                  ;;1.0E05
5541  022200  000001                          1
5542  022202  023420                          23420                   ;;1.0E04
5543  022204  000000                          0
5544  022206  001750                          1750                    ;;1.0E03
5545  022210  000000                          0
5546  022212  000144                          144                     ;;1.0E02
5547  022214  000000                          0
5548  022216  000012                          12                      ;;1.0E01
5549  022220  000000                          0
5550  022222  000001                          1                       ;;1.0E00
5551  022224  000000                          0
5552  022226  000014                $DECVL:   .BLKB   12.             ;;RESERVE STORAGE FOR ASCIZ STRING
5553
5554                                          .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
5555
5556                                          ;;**************************************************************
5557                                          ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
5558                                          ;*UNSIGNED OCTAL ASCIZ NUMBER.
5559                                          ;*CALL
5560                                          ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
5561                                          ;*      JSR     PC,@#$DB2O      ;;CALL THE ROUTINE
5562                                          ;*      RETURN                  ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK
5563
5564
5565  022242  104412                $DB2O:    SAVREG                  ;;SAVE ALL REGISTERS
5566  022244  016601  000002                  MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
5567  022250  012705  022361                  MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
5568  022254  012704  000014                  MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
5569  022260  012703  177770                  MOV     #^C7,R3         ;;MASK
5570  022264  012100                          MOV     (R1)+,R0        ;;LOWER WORD
5571  022266  012101                          MOV     (R1)+,R1        ;;HIGH WORD
5572  022270  005002                          CLR     R2              ;;TERMINATOR
5573  022272  110245                1$:       MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
5574  022274  010002                          MOV     R0,R2           ;;GET THIS DIGIT
5575  022276  005304                          DEC     R4              ;;COUNT THIS CHARACTER
5576  022300  003007                          BGT     3$              ;;BR IF NOT THE LAST DIGIT
5577  022302  001405                          BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
5578  022304  005205                          INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
5579  022306  010566  000002                  MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
5580  022312  104413                          RESREG                  ;;RESTORE ALL REGISTERS
5581  022314  000207                          RTS     PC              ;;RETURN TO USER
5582  022316  006203                2$:       ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
5583  022320  006001                3$:       ROR     R1              ;;POSITION THE BINARY NUMBER FOR
```

```
5584  022322  006000                          ROR     R0              ;;      THE NEXT OCTAL DIGIT
5585  022324  006001                          ROR     R1
5586  022326  006000                          ROR     R0
5587  022330  006001                          ROR     R1
5588  022332  006000                          ROR     R0
5589  022334  040302                          BIC     R3,R2           ;;MASK OUT ALL JUNK
5590  022336  062702  000060                  ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
5591  022342  000753                          BR      1$              ;;GO PUT IT IN THE DATA TABLE
5592  022344  000016              $OCTVL: .BLKB   14.             ;;RESERVE DATA TABLE
5593
5594                              .SBTTL  SCOPE HANDLER ROUTINE
5595
5596                              ;;**************************************************************
5597                              ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
5598                              ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
5599                              ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
5600                              ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
5601                              ;*SW14=1        LOOP ON TEST
5602                              ;*SW11=1        INHIBIT ITERATIONS
5603                              ;*SW09=1        LOOP ON ERROR
5604                              ;*SW08=1        LOOP ON TEST IN SWR<7:0>
5605                              ;*CALL
5606                              ;*      SCOPE           ;;SCOPE=IOT
5607
5608  022362                      $SCOPE:
5609  022362  104407                          CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
5610  022364  032777  040000  156562  1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
5611  022372  001114                          BNE     $OVER           ;;YES IF SW14=1
5612                              ;*****START OF CODE FOR THE XOR TESTER*****
5613  022374  000416              $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
5614                                                              ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
5615  022376  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
5616  022402  012737  022422  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
5617  022410  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
5618  022414  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
5619  022420  000463                          BR      $SVLAD          ;;GO TO THE NEXT TEST
5620  022422  022626              5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
5621  022424  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
5622  022430  000423                          BR      7$              ;;LOOP ON THE PRESENT TEST
5623  022432                      6$:;*****END OF CODE FOR THE XOR TESTER*****
5624  022432  032777  000400  156514          BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
5625  022440  001404                          BEQ     2$              ;;BR IF NO
5626  022442  127737  156506  001116          CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
5627  022450  001465                          BEQ     $OVER           ;;BR IF YES
5628  022452  105737  001117          2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
5629  022456  001421                          BEQ     3$              ;;BR IF NO
5630  022460  123737  001131  001117          CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
5631  022466  101015                          BHI     3$              ;;BR IF NO
5632  022470  032777  001000  156456          BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
5633  022476  001404                          BEQ     4$              ;;BR IF NO
5634  022500  013737  001124  001122  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
5635  022506  000446                          BR      $OVER
5636  022510  105037  001117          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
5637  022514  005037  001174                  CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
5638  022520  000415                          BR      1$              ;;ESCAPE TO THE NEXT TEST
5639  022522  032777  004000  156424  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
```

```
5640  022530  001011                          BNE     1$              ;;BR IF YES
5641  022532  005737  001216                  TST     $PASS           ;;IF FIRST PASS OF PROGRAM
5642  022536  001406                          BEQ     1$              ;;     INHIBIT ITERATIONS
5643  022540  005237  001120                  INC     $ICNT           ;;INCREMENT ITERATION COUNT
5644  022544  023737  001174  001120          CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
5645  022552  002024                          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
5646  022554  012737  000001  001120  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
5647  022562  013737  022640  001174          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
5648  022570  105237  001116          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
5649  022574  113737  001116  001214          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
5650  022602  011637  001122                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
5651  022606  011637  001124                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
5652  022612  005037  001176                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
5653  022616  112737  000001  001131          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
5654  022624  013777  001116  156324  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
5655  022632  013716  001122                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
5656  022636  000002                          RTI                     ;;FIXES PS
5657  022640  003720                  $MXCNT: 2000.                   ;;MAX. NUMBER OF ITERATIONS
5658
5659                                  .SBTTL  TRAP DECODER
5660
5661                                  ;;**************************************************************
5662                                  ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
5663                                  ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
5664                                  ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
5665                                  ;*GO TO THAT ROUTINE.
5666
5667  022642  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
5668  022644  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
5669  022650  005740                          TST     -(R0)           ;;BACKUP BY 2
5670  022652  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
5671  022654  006300                          ASL     R0              ;;POSITION FOR INDEXING
5672  022656  016000  022676                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
5673  022662  000200                          RTS     R0              ;;GO TO ROUTINE
5674
5675
5676                                  ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
5677
5678  022664  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
5679  022666  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
5680  022674  000002                          RTI                     ;;RESTORE THE PSW
5681
5682                                  .SBTTL  TRAP TABLE
5683
5684                                  ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
5685                                  ;*BY THE "TRAP" INSTRUCTION.
5686
5687                                  ;       ROUTINE
5688                                  ;       -------
5689  022676  022664                  $TRPAD: .WORD   $TRAP2
5690  022700  017722                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TYPE TYPEOUT ROUTINE
5691  022702  020666                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
5692  022704  020642                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
5693  022706  020702                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
5694  022710  021070                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
5695
```

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

                                        $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
                                        $^DCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
                                        $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
                                        $SAVREG ;;CALL=SAVREG   TRAP+12(104412) SAVE R0-R5 ROUTINE
                                        $RESREG ;;CALL=RESREG   TRAP+13(104413) RESTORE R0-R5 ROUTINE
                                        $DSPLY  ;;CALL=DISPLY   TRAP+14(104414) ROUTINE TO TYPE ERROR MESSAGES
                                $TERM=.-$TRPAD

                                ;;**************************************************************************



                                .SBTTL  SINGLE/DUAL PORT RH70/RM03 DRIVER (REV 1.0)

                                ;COPYRIGHT (C) 1976
                                ;DIGITAL EQUIPMENT CORP.
                                ;MAYNARD, MA 01754
                                ;AUTHOR(S): JIM LACEY/CHUCK HESS/C. CHEN

                                ;;**************************************************************************

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

022730  000000  000000  000000  RMERRS: .WORD   0,0,0,0
022736  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

022740      000                 DRVACT: .BYTE   0               ;DRIVE 0
022741      000                         .BYTE   0               ;DRIVE 1
022742      000                         .BYTE   0               ;DRIVE 2
022743      000                         .BYTE   0               ;DRIVE 3
022744      000                         .BYTE   0               ;DRIVE 4
022745      000                         .BYTE   0               ;DRIVE 5
022746      000                         .BYTE   0               ;DRIVE 6
022747      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

022750      000                 DRVSTA: .BYTE   0               ;DRIVE 0
022751      000                         .BYTE   0               ;DRIVE 1
022752      000                         .BYTE   0               ;DRIVE 2
022753      000                         .BYTE   0               ;DRIVE 3
022754      000                         .BYTE   0               ;DRIVE 4
```

```
5752  022755  000                     .BYTE   0               ;DRIVE 5
5753  022756  000                     .BYTE   0               ;DRIVE 6
5754  022757  000                     .BYTE   0               ;DRIVE 7
5755
5756                                ;TABLE OF DRIVE TYPES (DRVTYP=8 BYTES)
5757                                        ;DRVTYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
5758                                        ;DRVTYP=4 IF DRIVE IS RM03
5759                                        ;DRVTYP=-1 IF NOT RM03
5760
5761  022760  000                   DRVTYP: .BYTE   0               ;DRIVE 0
5762  022761  000                           .BYTE   0               ;DRIVE 1
5763  022762  000                           .BYTE   0               ;DRIVE 2
5764  022763  000                           .BYTE   0               ;DRIVE 3
5765  022764  000                           .BYTE   0               ;DRIVE 4
5766  022765  000                           .BYTE   0               ;DRIVE 5
5767  022766  000                           .BYTE   0               ;DRIVE 6
5768  022767  000                           .BYTE   0               ;DRIVE 7
5769
5770                                ;TABLE OF DUAL PORT INITIALIZATION INDICATORS
5771                                        ;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
5772                                        ;DPINT<0 IF INITIALIZATION IS IN PROGRESS
5773
5774  022770  000                   DPINT:  .BYTE   0               ;DRIVE 0
5775  022771  000                           .BYTE   0               ;DRIVE 1
5776  022772  000                           .BYTE   0               ;DRIVE 2
5777  022773  000                           .BYTE   0               ;DRIVE 3
5778  022774  000                           .BYTE   0               ;DRIVE 4
5779  022775  000                           .BYTE   0               ;DRIVE 5
5780  022776  000                           .BYTE   0               ;DRIVE 6
5781  022777  000                           .BYTE   0               ;DRIVE 7
5782
5783                                ;TABLE OF PENDING DUAL PORT REQUESTS
5784                                        ;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
5785                                        ;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE
5786
5787  023000  000                   DPRQS:  .BYTE   0               ;DRIVE 0
5788  023001  000                           .BYTE   0               ;DRIVE 1
5789  023002  000                           .BYTE   0               ;DRIVE 2
5790  023003  000                           .BYTE   0               ;DRIVE 3
5791  023004  000                           .BYTE   0               ;DRIVE 4
5792  023005  000                           .BYTE   0               ;DRIVE 5
5793  023006  000                           .BYTE   0               ;DRIVE 6
5794  023007  000                           .BYTE   0               ;DRIVE 7
5795
5796                                ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
5797                                        ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
5798                                        ;"DPB" OF THE I/O OPERATION.
5799
5800  023010  000000                TRNSWT: .WORD   0
5801
5802                                ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
5803                                        ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
5804                                        ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
5805                                        ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
5806                                        ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
5807
```

```
5808  023012  000000                  SRCHWT: .WORD   0
5809
5810                                  ;RM03 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
5811                                          ;ACTDRV=0 IF DRIVER IS INACTIVE
5812                                          ;ACTDRV>0 IF DRIVER IS ACTIVE
5813
5814  023014      000                 ACTDRV: .BYTE   0
5815
5816                                  ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
5817                                          ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
5818                                          ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
5819
5820  023015      000                 ACTSTR: .BYTE   0
5821
5822                                  ;UNLOAD FLAG (ULDFLG=8 BYTES)
5823                                          ;ULDFLG=0 IF NO UNLOAD COMMAND
5824                                          ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
5825                                          ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
5826
5827  023016      000                 ULDFLG: .BYTE   0               ;DRIVE 0
5828  023017      000                         .BYTE   0               ;DRIVE 1
5829  023020      000                         .BYTE   0               ;DRIVE 2
5830  023021      000                         .BYTE   0               ;DRIVE 3
5831  023022      000                         .BYTE   0               ;DRIVE 4
5832  023023      000                         .BYTE   0               ;DRIVE 5
5833  023024      000                         .BYTE   0               ;DRIVE 6
5834  023025      000                         .BYTE   0               ;DRIVE 7
5835
5836                                  ;LOOK AHEAD COUNT (LACNT=8 BYTES)
5837                                          ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
5838
5839  023026      000                 LACNT:  .BYTE   0               ;DRIVE 0
5840  023027      000                         .BYTE   0               ;DRIVE 1
5841  023030      000                         .BYTE   0               ;DRIVE 2
5842  023031      000                         .BYTE   0               ;DRIVE 3
5843  023032      000                         .BYTE   0               ;DRIVE 4
5844  023033      000                         .BYTE   0               ;DRIVE 5
5845  023034      000                         .BYTE   0               ;DRIVE 6
5846  023035      000                         .BYTE   0               ;DRIVE 7
5847
5848                                  ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
5849                                          ;SAVEFG <0 IF SAVE THE RH70/RM03 REGISTERS WHEN THE
5850                                          ;OPERATION IS COMPLETED AS PER (DPB+14).
5851                                          ;SAVEFG=0 IF SAVE THE RH70/RM03 REGISTERS, AS PER
5852                                          ;(DPB+14), AFTER AN ERROR.
5853
5854  023036  000000                  SAVEFG: .WORD   0
5855
5856                                  ;SEEK FLAG (SEEKFG=1 WORD)
5857                                          ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
5858                                          ;FOR A DATA TRANSFER START A SEARCH COMMAND
5859                                          ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
5860                                          ;DISREGARD THE WINDOW
5861
5862  023040  177777                  SEEKFG: .WORD   -1
5863
```

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

023042  177777                TIMER:  .WORD   -1      ;DRIVE 0
023044  177777                        .WORD   -1      ;DRIVE 1
023046  177777                        .WORD   -1      ;DRIVE 2
023050  177777                        .WORD   -1      ;DRIVE 3
023052  177777                        .WORD   -1      ;DRIVE 4
023054  177777                        .WORD   -1      ;DRIVE 5
023056  177777                        .WORD   -1      ;DRIVE 6
023060  177777                        .WORD   -1      ;DRIVE 7

                                ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
                                        ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
                                        ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N

023062  177777                DTUW:   .WORD   -1

                                ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
                                        ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
                                        ;ATTENTION BIT

023064      001                 ATABIT: .BYTE   1       ;DRIVE 0
023065      002                         .BYTE   2       ;DRIVE 1
023066      004                         .BYTE   4       ;DRIVE 2
023067      010                         .BYTE   10      ;DRIVE 3
023070      020                         .BYTE   20      ;DRIVE 4
023071      040                         .BYTE   40      ;DRIVE 5
023072      100                         .BYTE   100     ;DRIVE 6
023073      200                         .BYTE   200     ;DRIVE 7

                                ;FSRM03 TO RH70 "MASSBUS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE
                                ;CALLING IT FATAL (MCPEMX=1 WORD)

023074  000003                MCPEMX: .WORD   3

                                ;STORAGE FOR RMADR (THE FIRST ADDRESS (776700) OF THE RH70/RM03),
                                ;RMVEC (THE VECTOR ADDRESS (254)), AND RMVEC+2 (THE BR LEVEL (5)).

023076  176700                RMADR:  .WORD   176700
023100  000254  000240        RMVEC:  .WORD   254,5*32.

                                ;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)

023104  000004                MXLACT: .WORD   4
                                ;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)

023106  001000                MXDLTA: .WORD   8.*64.
                                ;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)

023110  000200                MNDLTA: .WORD   2*64.
                                ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)

023112  000005                MXWNDW: .WORD   5

                                ;DEFINITIONS OF THE RH70/RM03 ADDRESS INDEXES
```

```
                        RMCS1=0                 ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
                        RMWC=2                  ;WORD COUNT REGISTER (NOT A DRIVE REG)
                        RMBA=4                  ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
                        RMDA=6                  ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
                        RMCS2=10                ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
                        RMDS=12                 ;DRIVE STATUS REGISTER (DRIVE REG 01)
                        RMER1=14                ;ERROR REGISTER #1 (DRIVE REG. 02)
                        RMAS=16                 ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
                        RMLA=20                 ;LOOK AHEAD REGISTER (DRIVE REG. 07)
                        RMDB=22                 ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
                        RMMR1=24                ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
                        RMDT=26                 ;DRIVE TYPE REGISTER (DRIVE REG. 06)
                        RMSN=30                 ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
                        RMOF=32                 ;OFFSET REGISTER (DRIVE REG. 11)
                        RMDC=34                 ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
                        RMHR=36                 ;DUMMY  ADDRESS REGISTER (DRIVE REG. 13)
                        RMMR2=40                ;MAINTENANCE REGISTER #2
                        RMER2=42                ;ERROR REGISTER #2 (DRIVE REG. 15)
                        RMEC1=44                ;ECC POSITION REGISTER (DRIVE REG. 16)
                        RMEC2=46                ;ECC PATTERN REGISTER (DRIVE REG.  17)

                ;RH70/RM03 DRIVER INITIALIZATION CODE
                        ;THIS ROUTINE WILL DETERMINE WHICH RM03 DRIVES ARE
                        ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
                        ;TO THE PROPER STATE FOR EACH DRIVE.
                        ;NOTE: THIS ROUTINE CALLS DRVINT
                ;
                ;CALL
                ;
                        JSR     PC,RMINIT
                        RETURN
                ;
                ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
                ;
023114  104412  RMINIT: SAVREG                  ;SAVE R0 - R5
023116  013746  177776  MOV     @#PS,-(SP)      ;SAVE THE PRESENT PROCESSOR STATUS
023122  012737  000240  177776  MOV     #<5*32.),@#PS   ;CHANGE THE PRIORITY TO 5
023130  004737  030730          JSR     PC,CLRQUE       ;CLEAR ALL REQUEST QUEUES
023134  012701  022730          MOV     #RMERRS,R1      ;FIRST ADDRESS TO BE CLEARED
023140  012702  023040          MOV     #SEEKFG,R2      ;LAST ADDRESS TO BE CLEARED
023144  005021          1$:     CLR     (R1)+           ;CLEAR
023146  020102                  CMP     R1,R2           ;ARE WE DONE?
023150  103775                  BLO     1$              ;BRANCH IF NO
023152  012702  023062          MOV     #DTUW,R2        ;LAST ADDRESS
023156  012721  177777  2$:     MOV     #-1,(R1)+       ;INITIALIZE
023162  020102                  CMP     R1,R2           ;DONE?
023164  101774                  BLOS    2$              ;LOOP IF NO
023166  005037  022750          CLR     DRVSTA          ;SET ALL DRIVES TO OFFLINE
023172  005037  022752          CLR     DRVSTA+2
023176  005_    022754          CLR     DRVSTA+4
023202  005037  022756          CLR     DRVSTA+6
023206  013703  023100          MOV     RMVEC,R3        ;SETUP THE RH70/RM03 VECTOR
023212  012723  025702          MOV     #ISR,(R3)+
023216  013713  023102          MOV     RMVEC+2,(R3)
023222  013704  023076          MOV     RMADR,R4        ;FIRST ADDRESS OF RH70/RM03
```

```
5976  023226  012764  000040  000010          MOV     #BIT05,RMCS2(R4)  ;MASSBUS INIT
5977  023234  005001                          CLR     R1                ;START WITH DRIVE 0
5978  023236  004037  023326          3$:     JSR     R0,DRVINT         ;INIT THE DRIVE
5979  023242  000401                          BR      4$                ;'DVA' NOT SET OR PARITY ERROR
5980  023244  000402                          BR      5$                ;NORMAL RETURN
5991  023246  105061  022750          4$:     CLRB    DRVSTA(R1)        ;SET DRIVE STATUS TO OFFLINE
5982  023252  005201                  5$:     INC     R1                ;GO TO NEXT DRIVE
5983  023254  042701  177770                  BIC     #^C7,R1           ;MASK OUT UNUSED BITS
5984  023260  001366                          BNE     3$                ;BR IF MORE DRIVES TO GO
5985  023262  012701  000007                  MOV     #7,R1             ;START WITH DRIVE 7
5986  023266  012737  177776                  CLR     @#PS              ;CLEAR THE PROCESSOR STATUS
5987  023272  105761  022770          6$:     TSTB    DPINT(R1)         ;WAITING FOR DRIVE TO SWITCH PORTS ?
5988  023276  001405                          BEQ     8$                ;BR NOT WAITING
5989  023300  004737  030364                  JSR     PC,SET.IE         ;SET INTERRUPT
5990  023304  105761  022770          7$:     TSTB    DPINT(R1)         ;DRIVE SWITCHED PORTS ?
5991  023310  001375                          BNE     7$                ;BR IF NOT
5992  023312  005301                  8$:     DEC     R1                ;GO TO THE NEXT DRIVE
5993  023314  100366                          BPL     6$                ;CHECK NEXT DRIVE
5994  023316  012637  177776                  MOV     (SP)+,@#PS        ;RESTORE THE PROCESSOR STATUS
5995  023322  104413                          RESREG                    ;RESTORE R0 - R5
5996  023324  000207                          RTS     PC                ;BYE-BYE
5997
5998                                  ;DRIVE INITILIZATION ROUTINE
5999                                          ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
6000                                          ;AN RM03. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT22
6001                                          ;IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
6002                                          ;INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
6003                                          ;DRVSTA IS SET TO THE PROPER CONDITION.
6004                                  ;CALL
6005                                  ;       MOV     #DRVNUM,R1        ;DRIVE NUMBER TO R1
6006                                  ;       MOV     RMADR,R4          ;UNIBUS ADDRESS OF RH70/RM03 (RMCS1)
6007                                  ;       JSR     R0,DRVINT         ;CALLED BY A JSR
6008                                  ;       RETURN1                   ;ERROR OCCURRED (PARITY)
6009                                  ;       RETURN2                   ;NORMAL RETURN
6010                                  ;
6011
6012  023326  010546                  DRVINT: MOV     R5,-(SP)          ;SAVE R5
6013  023330  105061  022750          DULP:   CLRB    DRVSTA(R1)        ;START DRIVE STATUS AS OFFLINE
6014  023334  105061  022760                  CLRB    DRVTYP(R1)        ;CLEAR THE DRIVE TYPE INDICATOR
6015  023340  105061  023016                  CLRB    ULDFLG(R1)        ;CLEAR THE UNLOAD FLAG
6016  023344  010164  000010                  MOV     R1,RMCS2(R4)      ;SELECT A DRIVE
6017  023350  112764  000111  000000          MOVB    #111,RMCS1(R4)    ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
6018  023356  032764  010000  000010          BIT     #BIT12,RMCS2(R4)  ;NONEXISTENT DRIVE?
6019  023364  001403                          BEQ     1$                ;NO---BRANCH
6020  023366  004737  030364                  JSR     PC,SET.IE         ;GO SET "IE" WITHOUT A "TRE"
6021  023372  000476                          BR      6$                ;LEAVE THIS ROUTINE
6022  023374  105061  022750          1$:     CLRB    DRVSTA(R1)        ;SET DRIVE STATUS TO OFFLINE
6023  023400  032764  004000  000000          BIT     #BIT11,RMCS1(R4)  ;SEE IF DRIVE AVAILABLE
6024  023406  001750                          BEQ     DULP              ;BR IF DRIVE NOT AVAILABLE
6025  023410  004037  027674                  JSR     R0,RD.RM          ;READ THE DRIVE TYPE REG.
6026  023414  000026                          RMDT
6027  023416  023614                          8$                        ;ERROR RETURN ADDRESS
6028  023420  012605                          MOV     (SP)+,R5          ;PUT DRIVE TYPE IN R5
6029  023422  112761  000004  022760          MOVB    #4,DRVTYP(R1)     ;SET RM03 INDICATOR
6030  023430  022705  020024                  CMP     #20024,R5         ;SINGLE PORT RM03 ?
6031  023434  001407                          BEQ     2$                ;BR IF YES
```

```
6032  023436  022705  024024                    CMP     #24024,R5       ;DUAL PORT RM03 ?
6033  023442  001404                            BEQ     2$              ;BR IF YES
6034  023444  112761  177777  022760            MOVB    #-1,DRVTYP(R1)  ;SET INDICATOR TO 'OTHER'
6035  023452  000446                            BR      6$              ;EXIT
6036  023454  012746  000121          2$:       MOV     #121,-(SP)      ;DO A "READ-IN PRESET"
6037  023460  004037  030054                    JSR     R0,WRT.RM
6038  023464  000000                            RMCS1
6039  023466  023614                            B$
6040  023470  012746  010000                    MOV     #BIT12,-(SP)    ;SET FMT22=1
6041  023474  004037  030054                    JSR     R0,WRT.RM
6042  023500  000032                            RMOF
6043  023502  023614                            B$
6044  023504  004037  027674                    JSR     R0,RD.RM        ;READ RMDS
6045  023510  000012                            RMDS
6046  023512  023614                            B$
6047  023514  012605                            MOV     (SP)+,R5        ;AND SAVE IT IN R5
6048  023516  100015                            BPL     4$              ;BRANCH IF ATA=0
6049  023520  116164  023064  000016            MOVB    ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
6050  023526  004037  027674                    JSR     R0,RD.RM        ;FIND OUT WHY ATA=1
6051  023532  000014                            RMER1
6052  023534  023614                            B$
6053  023536  006126                            ROL     (SP)+           ;IS IT UNSAFE?
6054  023540  100004                            BPL     4$              ;BR IF NOT
6055  023542  112761  177777  022750            MOVB    #-1,DRVSTA(R1)  ;SET UNSAFE INDICATOR
6056  023550  000407                            BR      6$              ;EXIT
6057  023552  005105                    4$:     COM     R5              ;CHECK MOL, DPR, DRY, AND VV
6058  023554  042705  167077                    BIC     #^C<BIT12!BIT08!BIT07!BIT06>,R5
6059  023560  001003                            BNE     6$              ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
6060  023562  112761  000001  022750            MOVB    #1,DRVSTA(R1)   ;SET DRIVE STATUS TO ONLINE
6061  023570  005720                    6$:     TST     (R0)+           ;STEP OVER THE ERROR RETURN
6062  023572  000410                            BR      8$              ;EXIT
6063  023574  006301                    7$:     ASL     R1              ;CHANGE INDEX TO ADDRESS WORDS
6064  023576  012761  003720  023042            MOV     #2000.,TIMER(R1)  ;START 2 SEC TIMER
6065  023604  006201                            ASR     R1              ;RESTORE R1
6066  023606  112761  177777  022770            MOVB    #-1,DPINT(R1)   ;SET PORT INITIALIZE INIDICATOR
6067  023614  012605                    8$:     MOV     (SP)+,R5        ;RESTORE R5
6068  023616  000200                            RTS     R0              ;EXIT
6069
6070                                    ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
6071                                    ;
6072                                    ;CALL
6073                                    ;
6074                                    ;       JSR     R0,@#RM03       ;CALL THE RM03 DRIVER
6075                                    ;       PNTADR                  ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
6076                                    ;       RETURN1                 ;RETURN HERE IF QUEUE IS FULL
6077                                    ;       RETURN2                 ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
6078                                    ;                               ;IS AN ERROR CONDITION
6079
6080  023620  013746  177776            RM03:   MOV     @#PS,-(SP)      ;SAVE THE CALLING STATUS
6081  023624  013737  023102  177776            MOV     RMVEC+2,@#PS    ;DON'T ALLOW ANY RM03 INTERRUPTS
6082  023632  112737  000001  023014            MOVB    #1,ACTDRV       ;SET "ACTIVE DRIVER" FLAG
6083  023640  104412                            SAVREG                  ;SAVE R0 - R5
6084  023642  011002                            MOV     (R0),R2         ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
6085  023644  005062  000016                    CLR     16(R2)          ;CLEAR THE STATUS/ERROR INDICATOR
6086  023650  111201                            MOVB    (R2),R1         ;PICKUP THE DRIVE NUMBER
6087  023652  013704  023076                    MOV     RMADR,R4        ;UNIBUS ADDRESS OF RMCS1
```

```
6088  023656  105761  022750                    TSTB    DRVSTA(R1)          ;CHECK DRIVES STATUS
6089  023662  003014                            BGT     1$                  ;BRANCH IF ONLINE
6090  023664  105761  023016                    TSTB    ULDFLG(R1)          ;UNLOAD COMMAND IN QUEUE?
6091  023670  001036                            BNE     3$                  ;BRANCH IF YES
6092  023672  105761  022770                    TSTB    DPINT(R1)           ;TRYING TO INIT THE DRIVE
6093  023676  001042                            BNE     5$                  ;BR IF YES
6094  023700  004037  023326                    JSR     R0,DRVINT           ;GO INIT. THE DRIVE
6095  023704  000434                            BR      4$                  ;ERROR RETURN
6096  023706  105761  022750                    TSTB    DRVSTA(R1)          ;IS DRIVE STATUS ONLINE?
6097  023712  003445                            BLE     6$                  ;BR IF NOT
6098  023714  105761  023000          1$:       TSTB    DPRQS(R1)           ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
6099  023720  001031                            BNE     5$                  ;BR IF YES
6100  023722  010164  000010                    MOV     R1,RMCS2(R4)        ;SELECT THE DRIVE
6101  023726  004037  031026                    JSR     R0,DRVQUE           ;PUT THIS REQUEST IN QUEUE
6102  023732  000460                            BR      9$                  ;QUEUE IS FULL
6103  023734  122762  000103  000002            CMPB    #103,2(R2)          ;IS THIS REQ. FOR AN UNLOAD?
6104  023742  001003                            BNE     2$                  ;BR IF NO
6105  023744  112761  177777  023016            MOVB    #-1,ULDFLG(R1)      ;SET THE "UNLOAD IN QUEUE" FLAG
6106  023752  105761  022740          2$:       TSTB    DRVACT(R1)          ;IS THIS DRIVE ACTIVE?
6107  023756  001043                            BNE     8$                  ;BR IF YES
6108  023760  004737  024112                    JSR     PC,OPT              ;CALL THE OPTIMIZER
6109  023764  000440                            BR      8$
6110  023766  012762  120000  000016    3$:     MOV     #BIT15!BIT13,16(R2) ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
6111  023774  000434                            BR      8$                  ;EXIT
6112  023776  004737  025172          4$:       JSR     PC,CI7              ;GO HANDLE THE PARITY ERROR
6113  024002  000431                            BR      8$
6114  024004  004037  031026          5$:       JSR     R0,DRVQUE           ;PUT REQUEST IN QUEUE
6115  024010  000431                            BR      9$                  ;QUEUE IS FULL
6116  024012  032714  000100                    BIT     #BIT06,(R4)         ;IE BIT SET ?
6117  024016  001023                            BNE     8$                  ;YES
6118  024020  004737  030364                    JSR     PC,SET.IE           ;SET THE INTERRUPT
6119  024024  000420                            BR      8$                  ;RETURN
6120  024026  105761  022750          6$:       TSTB    DRVSTA(R1)          ;SEE IF DRIVE OFFLINE OR UNSAFE
6121  024032  002412                            BLT     7$                  ;BR IF UNSAFE
6122  024034  012762  140000  000016            MOV     #BIT15!BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
6123  024042  105761  022760                    TSTB    DRVTYP(R1)          ;SEE IF OFFLINE OR NONEXISTENT
6124  024046  001007                            BNE     8$                  ;BR IF OFFLINE
6125  024050  012762  100002  000016            MOV     #BIT15!BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
6126  024056  000403                            BR      8$                  ;GO TO EXIT
6127  024060  012762  110000  000016    7$:     MOV     #BIT15!BIT12,16(R2) ;DRIVE IS UNSAFE
6128  024066  104413                    8$:     RESREG                      ;RESTORE R0 - R5
6129  024070  005720                            TST     (R0)+               ;SETUP FOR NORMAL RETURN
6130  024072  000401                            BR      10$                 ;FINISH UP, THEN EXIT
6131  024074  104413                    9$:     RESREG                      ;RESTORE R0 - R5
6132  024076  005720                    10$:    TST     (R0)+               ;CORRECT THE RETURN ADDRESS
6133  024100  105037  023014                    CLRB    ACTDRV              ;CLEAR "ACTIVE DRIVER" FLAG
6134  024104  012637  177776                    MOV     (SP)+,@#PS          ;RETURN "PS" TO USER LEVEL
6135  024110  000200                            RTS     R0                  ;RETURN TO CALLER
6136
6137                                      ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
6138                                      ;
6139                                      ;CALL
6140                                      ;       MOV     #DRVNUM,R1          ;DRIVE NUMBER TO R1
6141                                      ;       JSR     PC,OPT              ;SETUP A COMMAND
6142                                      ;
6143  024112  104412                    OPT:    SAVREG                      ;SAVE R0 - R5
```

```
6144  024114  013746  177776                  MOV     @#PS,-(SP)      ;SAVE PROC. STATUS
6145  024120  146137  023064  023012          BICB    ATABIT(R1),SRCHWT       ;CLEAR LA SEACH FLAG
6146  024126  105061  023000                  CLRB    DPRQS(R1)       ;RESET THE PORT REQ FLAG ****
6147  024132  004737  031102                  JSR     PC,GETREQ       ;GET "DPR" POINTER OF REQUEST
6148  024136  005702                          TST     R2              ;IS THERE A REQUEST IN QUEUE?
6149  024140  001472                          BEQ     7$              ;NO--BRANCH TO EXIT
6150  024142  010164  000010                  MOV     R1,RMCS2(R4)    ;LOAD THE DRIVE ADDRESS *******
6151  024146  012764  000111  000000          MOV     #111,RMCS1(R4)  ;CLEAR THE DRIVE
6152  024154  032764  004000  000000          BIT     #BIT11,RMCS1(R4)        ;DVA SET ?
6153  024162  001446                          BEQ     5$              ;TO PROT REQUEST ,IF NOT
6154  024164  105761  022750          10$:    TSTB    DRVSTA(R1)      ;IS DRIVE ONLINE?
6155  024170  003014                          BGT     1$              ;YES--BRANCH
6156  024172  004737  031124                  JSR     PC,POPQUE       ;NO--REMOVE REQUEST FROM QUEUE
6157  024176  012762  140000  000016          MOV     #BIT15!BIT14,16(R2)  ;SET OFFLINE STATUS/ERROR INDICATOR
6158  024204  105761  022750                  TSTB    DRVSTA(R1)      ;IS DRIVE UNSAFE ?
6159  024210  100053                          BPL     8$              ;BR TO EXIT IF NOT
6160  024212  012762  110000  000016          MOV     #BIT15!BIT12,16(R2)  ;SET UNSAFE STATUS/ERROR INDICATOR
6161  024220  000447                          BR      8$              ;BRANCH TO EXIT
6162  024222                          1$:
6163                                  ;       MOV     #111,-(SP)      ;LOAD COMMAND ONTO THE STACK
6164                                  ;       JSR     R0,WRT.RM       ;LOAD THE REGISTER
6165                                  ;       RMCS1                   ;REGISTER INCREMENT
6166                                  ;       6$                      ;ERROR RETURN ADDRESS
6167                                  ;       BIT     #BIT11,(R4)     ;DRIVE AVAILABLE ?
6168                                  ;       BEQ     9$              ;BR IF NOT
6169  024222  122762  000150  000002          CMPB    #150,2(R2)      ;IS THE REQUEST FOR I/O?
6170  024230  002403                          BLT     2$              ;YES--BRANCH
6171  024232  004737  024556                  JSR     PC,CI4          ;CALL THE COMMAND INITIATOR
6172  024236  000440                          BR      8$              ;BRANCH TO EXIT
6173  024240  005737  023062          2$:     TST     DTUW            ;DATA TRANSFER UNDERWAY?
6174  024244  002012                          BGE     4$              ;YES--GO START A SEARCH
6175  024246  005737  023040                  TST     SEEKFG          ;DO IMPLIED SEEKS?
6176  024252  100404                          BMI     3$              ;YES---BRANCH
6177  024254  004037  025526                  JSR     R0,LA           ;NO--DO LOOK AHEAD
6178  024260  000427                          BR      8$              ;RETURN HERE ON A PARITY ERROR
6179  024262  000403                          BR      4$              ;GO START A SEARCH
6180  024264  004737  024350          3$:     JSR     PC,CI1          ;START A DATA TRANSFER
6181  024270  000423                          BR      8$
6182  024272  004737  024456          4$:     JSR     PC,CI3          ;START A SEARCH
6183  024276  000420                          BR      8$              ;GO TO THE EXIT
6184  024300  112761  177777  023000  5$:     MOVB    #-1,DPRQS(R1)   ;SET PORT REQUEST INDICATOR
6185  024306  010103                          MOV     R1,R3           ;SET UP TO ADDRESS WORDS
6186  024310  006303                          ASL     R3              ;CONVERT TO WORD INDEX
6187  024312  012763  023420  023042          MOV     #10000.,TIMER(R3)  ;START 10 SEC TIMER
6188  024320  000402                          BR      7$              ;EXIT
6189  024322  004737  025172          6$:     JSR     PC,CI7          ;PROCESS THE PARITY ERROR
6190  024326  032714  000100          7$:     BIT     #BIT06,(R4)     ;SEE IF 'IE' ALREADY SET
6191  024332  001002                          BNE     8$              ;BR IF SET
6192  024334  004737  030364                  JSR     PC,SET.IE       ;SET "IE" WITHOUT A "TRE"
6193  024340  012637  177776          8$:     MOV     (SP)+,@#PS      ;RESTORE PROC. STATUS
6194  024344  104413                          RESREG                  ;RESTORE R0 - R5
6195  024346  000207                          RTS     PC
6196
6197                                  ;COMMAND INITIATOR
6198                                  ;
6199                                  ;CALL
```

```
6200                                    ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
6201                                    ;       MOV     #DPB,R2         ;ADDRESS OF DPB
6202                                    ;       JSR     PC,CI?          ;CI?= CI1,CI3, OR CI4
6203                                    ;                               ;WHERE:
6204                                    ;                               ;CI1=DATA TRANSFER
6205                                    ;                               ;CI2=SEARCH REQUESTED BY DATA XFER
6206                                    ;                               ;CI4=NOT DATA TRANSFER
6207
6208  024350  004737  031124      CI1:  JSR     PC,POPQUE       ;REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
6209  024354  010237  023010            MOV     R2,TRNSWT       ;PUT REQ. IN TRANSFER WAIT QUEUE
6210  024360  010203                    MOV     R2,R3           ;DPB ADDRESS TO R3
6211  024362  013704  023076            MOV     RMADR,R4        ;RMCS1 ADDRESS
6212  024366  010164  000010            MOV     R1,RMCS2(R4)    ;SELECT DRIVE
6213  024372  062703  000004            ADD     #4,R3           ;DESIRED WORD COUNT
6214  024376  062704  000002            ADD     #2,R4           ;RMWC ADDRESS
6215  024402  012324                    MOV     (R3)+,(R4)+     ;LOAD WORD COUNT
6216  024404  012324                    MOV     (R3)+,(R4)+     ;LOAD BUFFER ADDRESS
6217  024406  012346                    MOV     (R3)+,-(SP)     ;LOAD SECTOR AND TRACK
6218  024410  004037  030054            JSR     R0,WRT.RM       ;CALL THE LOAD(WRITE) ROUTINE
6219  024414  000006                    RMDA                    ;INDEX OF REGISTER TO LOAD
6220  024416  025172                    CI7                     ;ERROR RETURN ADDRESS
6221  024420  012346                    MOV     (R3)+,-(SP)     ;LOAD CYLINDER ADDRESS
6222  024422  004037  030054            JSR     R0,WRT.RM
6223  024426  000034                    RMDC
6224  024430  025172                    CI7
6225  024432  016246  000002            MOV     2(R2),-(SP)     ;LOAD "COMMAND+GO", "A17&A16", AND "PSEL"
6226  024436  004037  030054            JSR     R0,WRT.RM
6227  024442  000000                    RMCS1
6228  024444  025172                    CI7
6229  024446  010137  023062            MOV     R1,DTUW         ;SET "DATA TRANSFER UNDERWAY"
6230  024452  000137  025134            JMP     CI5
6231  024456  013704  023076      CI3:  MOV     RMADR,R4        ;RMCS1 ADDRESS
6232  024462  010164  000010            MOV     R1,RMCS2(R4)    ;SELECT DRIVE
6233  024466  016246  000012            MOV     12(R2),-(SP)    ;DESIRED CYLINDER ADDRESS
6234  024472  004037  030054            JSR     R0,WRT.RM
6235  024476  000034                    RMDC
6236  024500  025172                    CI7
6237  024502  116203  000010            MOVB    10(R2),R3       ;PICKUP SECTOR ADDRESS
6238                                    ;       SUB     MXWNDW,R3       ;BACKUP BY MAX. SEARCH FOR I/O WINDOW
6239                                    ;       BGE     1$
6240                                    ;       ADD     #32.,R3
6241  024506  010346            1$:   MOV     R3,-(SP)        ;COMBINE THE ADJUSTED SECTOR WITH
6242  024510  042716  177740            BIC     #177740,(SP)    ;CHOP OF ALL EXENTED BITS ,IF ANY
6243  024514  116266  000011  000001    MOVB    11(R2),1(SP)    ;THE DESIRED TRACK
6244  024522  004037  030054            JSR     R0,WRT.RM       ;LOAD DESIRED TRACK & SECTOR
6245  024526  000006                    RMDA
6246  024530  025172                    CI7
6247  024532  012746  000131            MOV     #131,-(SP)      ;START A SEARCH
6248  024536  004037  030054            JSR     R0,WRT.RM
6249  024542  000000                    RMCS1
6250  024544  025172                    CI7
6251  024546  156137  023064  023012    BISB    ATABIT(R1),SRCHWT       ;SET "SEARCH WAIT" KEY
6252  024554  000567                    BR      CI5
6253  024556  013704  023076      CI4:  MOV     RMADR,R4        ;RMCS1 ADDRESS
6254  024562  010164  000010            MOV     R1,RMCS2(R4)    ;SELECT DRIVE
6255  024566  116203  000002            MOVB    2(R2),R3        ;PICKUP THE REQUESTED COMMAND
```

F10


```
6256  024572  122703  000131            CMPB    #131,R3         ;IS IT A SEARCH COMMAND?
6257  024576  001007                    BNE     1$              ;BRANCH IF NO
6258  024600  016246  000010            MOV     10(R2),-(SP)    ;LOAD DESIRED TRACK & SECTOR
6259  024604  004037  030054            JSR     R0,WRT.RM
6260  024610  000006                    RMDA
6261  024612  025172                    CI7
6262  024614  000403                    BR      2$              ;GO LOAD CYLINDER
6263  024616  122703  000105      1$:   CMPB    #105,R3         ;IS IT A SEEK COMMAND
6264  024622  001007                    BNE     3$              ;BRANCH IF NO
6265  024624  016246  000012      2$:   MOV     12(R2),-(SP)    ;LOAD DESIRED CYLINDER
6266  024630  004037  030054            JSR     R0,WRT.RM
6267  024634  000034                    RMDC
6268  024636  025172                    CI7
6269  024640  000546                    BR      CI6
6270  024642  122703  000115      3$:   CMPB    #115,R3         ;IS IT AN "OFFSET" COMMAND?
6271  024646  001013                    BNE     4$              ;BR IF NO
6272  024650  004037  027674            JSR     R0,RD.RM        ;MERGE THE OFFSET VALUE INTO RMOF
6273  024654  000032                    RMOF                    ;BUT DON'T CHANGE THE UPPER
6274  024656  025172                    CI7
6275  024660  116216  000001            MOVB    1(R2),(SP)      ;BYTE WHEN LOADING THE
6276  024664  004037  030054            JSR     R0,WRT.RM       ;REGISTER (RMOF)
6277  024670  000032                    RMOF
6278  024672  025172                    CI7
6279  024674  000530                    BR      CI6             ;GO START THE COMMAND
6280  024676  122703  000107      4$:   CMPB    #107,R3         ;IS IT A "RECALIBRATE" COMMAND?
6281  024702  001525                    BEQ     CI6             ;BRANCH IF YES
6282  024704  122703  000117            CMPB    #117,R3         ;IS IT A RETURN TO CENTER?
6283  024710  001522                    BEQ     CI6             ;BRANCH IF YES
6284  024712  122703  000103            CMPB    #103,R3         ;IS IT AN "UNLOAD" COMMAND?
6285  024716  001016                    BNE     5$              ;BRANCH IF NO
6286  024720  112761  000001  022740    MOVB    #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE INDICATOR
6287  024726  105061  022750            CLRB    DRVSTA(R1)      ;PUT DRIVE STATUS TO OFFLINE
6288  024732  112761  000001  023016    MOVB    #1,ULDFLG(R1)   ;SET "UNLOAD IN PROGRESS" FLAG
6289  024740  010346                    MOV     R3,-(SP)        ;START THE "UNLOAD" COMMAND
6290  024742  004037  030054            JSR     R0,WRT.RM
6291  024746  000000                    RMCS1
6292  024750  025172                    CI7
6293  024752  000207                    RTS     PC              ;RETURN TO USER
6294  024754  122703  000143      5$:   CMPB    #143,R3         ;IS IT A "SET FORMAT" COMMAND?
6295  024760  001014                    BNE     6$              ;BRANCH IF NO
6296  024762  004037  027674            JSR     R0,RD.RM        ;READ THE OFFSET REGISTER
6297  024766  000032                    RMOF
6298  024770  025172                    CI7
6299  024772  116266  000001  000001    MOVB    1(R2),1(SP)     ;COMBINE "FMT22","ECI", AND "HCI"
6300  025000  004037  030054            JSR     R0,WRT.RM       ;LOAD "FMT22", "ECI", AND/OR "HCI".
6301  025004  000032                    RMOF
6302  025006  025172                    CI7
6303  025010  000436                    BR      12$
6304  025012  122703  000141      6$:   CMPB    #141,R3         ;IS IT A "GET REGISTER" COMMAND?
6305  025016  001023                    BNE     10$             ;BRANCH IF NO
6306  025020  016203  000006      7$:   MOV     6(R2),R3        ;POINTS TO 1ST ADDRESS OF WHERE
6307                                                            ;TO PUT THE REGISTER(S)
6308  025024  116237  000010  025042    MOVB    10(R2),9$       ;INIT. THE INDEX FOR THE FIRST REG.
6309  025032  116205  000011            MOVB    11(R2),R5       ;INDEX OF LAST REG. TO MOVE
6310  025036  004037  027674      8$:   JSR     R0,RD.RM        ;READ RH70/RM03 REGISTER
6311  025042  000000                9$:   RMCS1                   ;INDEX OF REG. TO READ
```

```
6312  025044  025172                          CI7
6313  025046  012623                          MOV     (SP)+,(R3)+     ;GET THE CONTENTS OF RH70//RM03 REG.
6314  025050  023705  025042                  CMP     9$,R5           ;LAST REG. BEEN READ?
6315  025054  001414                          BEQ     12$             ;GET OUT IF YES
6316  025056  062737  000002  025042          ADD     #2,9$           ;INCREASE THE INDEX BY 2
6317  025064  000764                          BR      8$              ;LOOP--MORE TO READ
6318  025066  122703  000145          10$:    CMPB    #145,R3         ;IS IT A "SELECT DRIVE" COMMAND?
6319  025072  001405                          BEQ     12$             ;BRANCH IF YES
6320  025074  010346          11$:    MOV     R3,-(SP)        ;LOAD THE COMMAND
6321  025076  004037  030054                  JSR     R0,WRT.RM
6322  025102  000000                          RMCS1
6323  025104  025172                          CI7
6324  025106  004737  031124          12$:    JSR     PC,POPQUE       ;REMOVE REQ. FROM QUEUE
6325  025112  052762  000200  000016          BIS     #BIT07,16(R2)   ;SET THE "DONE" BIT
6326  025120  005737  023036                  TST     SAVEFG          ;SAVE THE RH70/RM03 REGISTERS?
6327  025124  100002                          BPL     13$             ;BRANCH IF NO
6328  025126  004737  030246                  JSR     PC,SVRH70       ;YES--GO SAVE THE REGISTERS
6329  025132  000207          13$:    RTS     PC              ;RETURN TO USER
6330  025134  006301          CI5:    ASL     R1
6331  025136  012761  001750  023042          MOV     #1000.,TIMER(R1)        ;SET A ONE SECOND TIMER
6332  025144  006201                          ASR     R1
6333  025146  112761  000001  022740          MOVB    #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE
6334  025154  000207                          RTS     PC              ;RETURN TO THE USER
6335  025156  010346          CI6:    MOV     R3,-(SP)        ;LOAD THE COMMAND
6336  025160  004037  030054                  JSR     R0,WRT.RM
6337  025164  000000                          RMCS1
6338  025166  025172                          CI7
6339  025170  000761                          BR      CI5
6340  025172  032764  010000  000010  CI7:    BIT     #BIT12,RMCS2(R4)  ;DRIVE NON-EXISTENT ?
6341  025200  001034                          BNE     CI8             ;BR IF YES
6342  025202  005702          1$:     TST     R2              ;ANYTHING IN QUEUE ?
6343  025204  001405                          BEQ     CI7B            ;BR IF NOT
6344  025206  012762  104000  000016          MOV     #BIT15!BIT11,16(R2)     ;SET "PARITY" ERROR INDICATOR
6345  025214  004737  030246                  JSR     PC,SVRH70       ;GO SAVE THE RH70/RM03 REGISTERS
6346  025220  012746  000111          CI7B:   MOV     #111,-(SP)      ;DO A "DRIVE CLEAR"
6347  025224  004037  030054                  JSR     R0,WRT.RM
6348  025230  000000                          RMCS1
6349  025232  025272                          CI8
6350  025234  004737  031006                  JSR     PC,EMPTYQ       ;EMPTY THE QUEUE
6351  025240  105061  023016                  CLRB    ULDFLG(R1)      ;CLEAR THE UNLOAD IN QUEUE FLAG
6352  025244  105061  022740                  CLRB    DRVACT(R1)      ;DRIVE IS IDLE
6353  025250  020137  023062                  CMP     R1,DTUW         ;IF THIS DRIVE HAD AN I/O REQUEST
6354  025254  001005                          BNE     1$              ;IN PROGRESS CLEAR ALL OF THE FLAGS
6355  025256  005037  023010                  CLR     TRNSWT
6356  025262  012737  177777  023062          MOV     #-1,DTUW
6357  025270  000207          1$:     RTS     PC
6358  025272  104412          CI8:    SAVREG                  ;SAVE R0 - R5
6359  025274  032764  010000  000010          BIT     #BIT12,RMCS2(R4)  ;IS 'NED' SET ?
6360  025302  001002                          BNE     1$              ;BR IF YES
6361  025304  005001                          CLR     R1
6362  025306  005003                          CLR     R3
6363  025310  105761  022740          1$:     TSTB    DRVACT(R1)      ;DRIVE ACTIVE?
6364  025314  001443                          BEQ     5$              ;BRANCH IF NO
6365  025316  013702  023010                  MOV     TRNSWT,R2       ;GET THE "TRANSFER WAIT" QUEUE
6366  025322  020137  023062                  CMP     R1,DTUW         ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
6367  025326  001402                          BEQ     2$              ;BRANCH IF YES
```

```
6368  025330  004737  031102            JSR     PC,GETREQ       ;GET THE DPB POINTER
6369  025334  005702                2$:   TST     R2              ;QUEUE ENTRY FOR DRIVE ?
6370  025336  001415                      BEQ     4$              ;BR IF NOT
6371  025340  032764  010000  000010      BIT     #BIT12,RMCS2(R4)  ;'NED' SET ?
6372  025346  001404                      BEQ     3$              ;BR IF NOT
6373  025350  012762  100002  000016      MOV     #BIT15!BIT01,16(R2)  ;SET 'DRIVE NON-EXISTENT' INDICATOR
6374  025356  000405                      BR      4$              ;CONTINUE
6375  025360  012762  102000  000016  3$: MOV     #BIT15!BIT10,16(R2) ;SET "NON-CLEARABLE PARITY" ERROR INDICATOR
6376  025366  004737  030246            JSR     PC,SVRH70       ;SAVE RH70/RM03 REGISTERS
6377  025372  012763  177777  023042  4$: MOV     #-1,TIMER(R3)   ;STOP THE TIMER
6378  025400  105061  022740            CLRB    DRVACT(R1)      ;SET "DRIVE ACTIVE" TO IDLE
6379  025404  020137  023062            CMP     R1,DTUW         ;IS THIS DRIVE SETUP FOR A TRANSFER
6380  025410  001005                      BNE     5$              ;BR IF NOT
6381  025412  012737  177777  023062      MOV     #-1,DTUW        ;RESET THE INDICATOR
6382  025420  005037  023010            CLR     TRNSWT          ;CLEAR THE TRANSFER QUEUE
6383  025424  105061  023016        5$: CLRB    ULDFLG(R1)      ;CLEAR UNLOAD FLAG
6384  025430  032764  010000  000010      BIT     #BIT12,RMCS2(R4)  ;'NED' SET ?
6385  025436  001021                      BNE     6$              ;BR IF YES
6386  025440  005201                      INC     R1              ;MOVE TO THE NEXT DRIVE
6387  025442  062703  000002            ADD     #2,R3
6388  025446  042701  177770            BIC     #^C7,R1
6389  025452  001316                      BNE     1$              ;BRANCH IF MORE DRIVES
6390  025454  012737  177777  023062      MOV     #-1,DTUW        ;NO DATA TRANSFERS UNDERWAY
6391  025462  005037  023010            CLR     TRNSWT          ;CLEAR THE 'TRANSFER WAIT' QUEUE
6392  025466  004737  030730            JSR     PC,CLRQUE       ;CLEAR ALL OF THE REQUEST QUEUES
6393  025472  012764  000040  000010      MOV     #BITO5,RMCS2(R4) ;DO A MASSBUS INIT.
6394  025500  000406                      BR      7$              ;CONTINUE
6395  025502  004737  031006        6$: JSR     PC,EMPTYQ       ;CLEAR THE DRIVE'S QUEUE
6396  025506  105061  022750            CLRB    DRVSTA(R1)      ;SET DRIVE TO OFFLINE
6397  025512  105061  022760            CLRB    DRVTYP(R1)      ;CLEAR THE DRIVE TYPE INDICATOR
6398  025516  004737  030364        7$: JSR     PC,SET.IE       ;SET "IE" WITHOUT "TRE"
6399  025522  104413                      RESREG                  ;RESTORE R0 - R5
6400  025524  000207                      RTS     PC              ;RETURN
6401
6402                                ;LOOK AHEAD ROUTINE
6403                                ;
6404                                ;CALL
6405                                ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
6406                                ;       MOV     #DPB,R2         ;POINT TO DPB
6407                                ;       JSR     R0,LA           ;GO CHECK THE WINDOW
6408                                ;       RETURN1                 ;ERROR RETURN
6409                                ;       RETURN2                 ;START A SEARCH
6410                                ;       RETURN3                 ;START A DATA TRANSFER
6411
6412  025526  013704  023076        LA:   MOV     RMADR,R4        ;GET RMCS1'S ADDRESS
6413  025532  010164  000010            MOV     R1,RMCS2(R4)    ;SELECT DRIVE
6414  025536  004037  027674            JSR     R0,RD.RM        ;READ DRIVE STATUS
6415  025542  000012                    RMDS
6416  025544  025674                    4$                      ;ERROR RETURN ADDRESS
6417  025546  042716  157577            BIC     #^C020200,(SP)  ;ON CYLINDER ?
6418  025552  022726  000200            CMP     #200,(SP)+      ;PIP=0,DRY=1?
6419  025556  001044                      BNE     3$              ;NO
6420  025560  105261  023026            INCB    LACNT(R1)       ;INCREMENT THE LOOK AHEAD COUNT
6421  025564  126137  023026  023104    CMPB    LACNT(R1),MXLACT ;EXCEED MAX?
6422  025572  003033                      BGT     2$              ;BRANCH IF YES
6423  025574  116203  000010            MOVB    10(R2),R3       ;GET DESIRED SECTOR ADDRESS AND
```

```
6424  025600  000303                          SWAB    R3              ;MULT. BY 64--ALIGN WITH
6425  025602  006203                          ASR     R3              ;LOOK AHEAD REGISTER
6426  025604  006203                          ASR     R3
6427  025606  012737  000340  177776          MOV     #340,@#PS       ;PRIORITY LEVEL "7"
6428  025614  004037  027674          6$:     JSR     R0,RD.RM        ;READ LOOK AHEAD REGISTER
6429  025620  000020                          RMLA
6430  025622  025674                          4$
6431  025624  021664  000020                  CMP     (SP),RMLA(R4)   ;CORRECT LA NUMBER ?
6432  025630  001402                          BEQ     7$              ;YES
6433  025632  005726                          TST     (SP)+           ;NO,CLEAR STACK
6434  025634  000415                          BR      3$
6435  025636  162603          7$:     SUB     (SP)+,R3        ;CALCULATE THE DELTA
6436  025640  002002                          BGE     1$
6437  025642  062703  004000                  ADD     #<32.*64.>,R3   ;MAKE THE DELTA POSITIVE
6438  025646  023703  023106          1$:     CMP     MXDLTA,R3       ;CHECK THE DELTA TO SEE
6439  025652  002406                          BLT     3$              ;IF IT IS WITHIN THE
6440  025654  023703  023110                  CMP     MNDLTA,R3       ;WINDOW---IF YES, ZERO
6441  025660  002003                          BGE     3$              ;THE LOOK AHEAD COUNT
6442  025662  105061  023026          2$:     CLRB    LACNT(R1)       ;AND TAKE THE I/O EXIT
6443  025666  005720                          TST     (R0)+
6444  025670  005720          3$:     TST     (R0)+           ;ADJUST THE RETURN ADDRESS
6445  025672  000402                          BR      5$              ;EXIT
6446  025674  004737  025172          4$:     JSR     PC,CI7          ;PROCESS THE ERROR
6447  025700  000200          5$:     RTS     R0              ;RETURN
6448
6449                                  ;INTERRUPT SERVICE ROUTINE
6450
6451  025702  112737  000001  023014  ISR:    MOVB    #1,ACTDRV       ;SET "ACTIVE DRIVER" FLAG
6452  025710  104412                          SAVREG                  ;SAVE R0 - R5
6453  025712  013704  023076                  MOV     RMADR,R4        ;ADDRESS OF RHSCS1
6454  025716  013701  023062                  MOV     DTUW,R1         ;GET "DATA TRANSFER UNDERWAY" INDICATOR
6455  025722  002403                          BLT     1$              ;BRANCH IF NO DATA TRANSFER UNDERWAY
6456  025724  004737  025746                  JSR     PC,TD           ;CALL TRANSFER DONE
6457  025730  000402                          BR      2$              ;EXIT
6458  025732  004737  026116          1$:     JSR     PC,SC           ;CALL SPECIAL CONDITIONS
6459  025736  104413          2$:     RESREG                  ;RESTORE R0 - R5
6460  025740  105037  023014                  CLRB    ACTDRV          ;CLEAR "ACTIVE DRIVER" FLAG
6461  025744  000002                          RTI                     ;RETURN
6462
6463                                  ;TRANSFER DONE ROUTINE
6464
6465  025746  105061  022740          TD:     CLRB    DRVACT(R1)      ;SET DRIVE ACTIVE INDICATOR TO IDLE
6466  025752  012737  177777  023062          MOV     #-1,DTUW        ;NO DATA TRANSFERS UNDERWAY
6467  025760  006301                          ASL     R1
6468  025762  012761  177777  023042          MOV     #-1,TIMER(R1)   ;CANCEL TIMEOUT
6469  025770  006201                          ASR     R1
6470  025772  013702  023010                  MOV     TRNSWT,R2       ;GET "DPB" ADDRESS FROM THE
6471  025776  005037  023010                  CLR     TRNSWT          ;TRANSFER WAIT QUEUE--CLEAR QUEUE
6472  026002  052762  000200  000016          BIS     #BIT07,16(R2)   ;SET DONE
6473  026010  010164  000010                  MOV     R1,RMCS2(R4)    ;SELECT THE DRIVE
6474  026014  004037  027674                  JSR     R0,RD.RM        ;TRANSFER ERR[illegible]TRE=1)?
6475  026020  000000                          RMCS1
6476  026022  025172                          CI7
6477  026024  006126                          ROL     (SP)+
6478  026026  100417                          BMI     3$              ;BR IF YES
6479  026030  005737  023036                  TST     SAVEFG          ;SAVE THE RH70/RM03 REGISTERS?
```

```
6480  026034  100002                          BPL     1$              ;BRANCH IF NO
6481  026036  004737  030246                  JSR     PC,SVRH70       ;YES--SAVE THE REGISTERS
6482  026042                          1$:
6483  026042  004737  031102                  JSR     PC,GETREQ       ;GET DPB POINTER
6484  026046  005702                          TST     R2              ;ENTRY FOR DRIVE ?
6485  026050  001403                          BEQ     2$              ;BR IF NOT
6486  026052  004737  024112                  JSR     PC,OPT          ;CALL OPTIMIZER
6487  026056  000417                          BR      SC              ;CHECK OTHER DRIVES
6488                                  ;THE RELEASE DRIVE COMMAND IS FORCED TO ENTER FOR DUAL PORT OPERATION
6489  026060  012714  000113          2$:     MOV     #113,(R4)       ;RELEASE THE DRIVE
6490  026064  000414                          BR      SC              ;CHECK FOR OTHER DRIVES
6491  026066  052762  100100  000016  3$:     BIS     #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
6492  026074  004737  031006                  JSR     PC,EMPTYQ       ;EMPTY THE "DRIVE'S WAIT" QUEUE
6493  026100  004737  030246                  JSR     PC,SVRH70       ;SAVE THE RH70/RM03 REGISTERS
6494  026104  012714  040111                  MOV     #40111,(R4)     ;ISSUE A "DRIVE CLEAR"
6495  026110  012714  000113                  MOV     #113,(R4)       ;ISSUE A RELEASE TO THE DRIVE
6496  026114  000400                          BR      SC              ;CHECK FOR OTHER DRIVES
6497
6498
6499
6500                                  ;SPECIAL CONDITION ROUTINE
6501
6502  026116  116403  000016          SC:     MOVB    RMAS(R4),R3     ;READ "RMAS"
6503  026122  001014                          BNE     2$              ;BRANCH IF ANY 'ATA' BITS SET
6504  026124  004037  027674                  JSR     R0,RD.RM        ;READ CONTROL AND STATUS REGISTER
6505  026130  000000                          RMCS1
6506  026132  025272                          CIB
6507  026134  106126                          ROLB    (SP)+           ;IS "IE"=1?
6508  026136  100405                          BMI     1$              ;YES, NO DRIVES TO CHECK
6509  026140  004037  031172                  JSR     R0,ES.SAV       ;SAVE THE ADDRESS IN '$ESCAPE'
6510  026144  104001                          ERROR   1               ;REPORT AN ILLEGAL INTERRUPT
6511  026146  004737  030364                  JSR     PC,SET.IE       ;SET INTERRUPT ENABLE
6512  026152  000207                  1$:     RTS     PC              ;RETURN
6513  026154  005046                  2$:     CLR     -(SP)           ;PROCESS ALL DRIVES THAT HAVE
6514  026156  110316                          MOVB    R3,(SP)         ;AN "ATA"=1
6515  026160  012703  000001                  MOV     #1,R3
6516  026164  005001                          CLR     R1
6517  026166  030316                  SC3:    BIT     R3,(SP)         ;ATA=1?
6518  026170  001005                          BNE     SC5             ;YES--BRANCH
6519  026172  005201                  SC4:    INC     R1              ;MOVE TO THE NEXT DRIVE
6520  026174  106303                          ASLB    R3
6521  026176  001373                          BNE     SC3             ;BRANCH IF MORE TO CHECK?
6522  026200  005726                          TST     (SP)+           ;CLEAN OFF THE STACK
6523  026202  000207                          RTS     PC              ;RETURN TO USER
6524  026204  105761  022770          SC5:    TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
6525  026210  001402                          BEQ     1$              ;BR IF NOT
6526  026212  000137  027130                  JMP     SC13            ;PROCESS THE DRIVE
6527  026216  105761  023000          1$:     TSTB    DPRQS(R1)       ;PORT REQUEST OUTSTANDING ?
6528  026222  001402                          BEQ     2$              ;BR IF NOT
6529  026224  000137  027130                  JMP     SC13            ;START THE OUTSTANDING COMMAND
6530  026230  105761  022750          2$:     TSTB    DRVSTA(R1)      ;CHECK THE DRIVE STATUS
6531  026234  003025                          BGT     5$              ;BRANCH IF ONLINE
6532  026236  105761  023016                  TSTB    ULDFLG(R1)      ;UNLOAD IN PROGRESS?
6533  026242  003422                          BLE     5$              ;BRANCH IF NOT
6534  026244  004737  031102                  JSR     PC,GETREQ       ;GET DPB POINTER
6535  026250  004737  030246                  JSR     PC,SVRH70       ;SAVE THE RH70/RM03 REGISTERS
```

```
6536  026254  004737  027060                  JSR     PC,SC12         ;SAVE RMDS, RMER1, RMER2, AND RMMR2
6537                                                                    ;ALSO DO A DRIVE INIT (DRVINT)
6538  026260  105761  022750                  TSTB    DRVSTA(R1)      ;DID DRIVE COME ONLINE?
6539  026264  003416                          BLE     6$              ;NO---BRANCH
6540  026266  032737  040000  022730          BIT     #BIT14,RMERRS   ;WAS THERE AN ERROR?
6541  026274  001002                          BNE     3$              ;BR IF ERROR
6542  026276  000137  026750                  JMP     SC11            ;NO ERROR
6543  026302  013705  022732          3$:     MOV     RMERRS+2,R5     ;YES -- PICKUP RMER1 AND
6544  026306  000502                          BR      SC6A            ;GO PROCESS THE ERROR
6545  026310  105761  022740          5$:     TSTB    DRVACT(R1)      ;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
6546  026314  001033                          BNE     SC6             ;BR IF EITHER
6547  026316  004737  027060                  JSR     PC,SC12         ;SAVE RMDS, RMER1, RMER2, AND RMMR2
6548                                                                    ;ALSO DO A DRVINT
6549  026322  105761  022770          6$:     TSTB    DPINT(R1)       ;TRYING TO INIT THE DRIVE ?
6550  026326  001321                          BNE     SC4             ;BR IF YES, CHECK ON MORE DRIVES
6551  026330  105761  022750                  TSTB    DRVSTA(R1)      ;CHECK ON DRIVE'S STATUS
6552  026334  100412                          BMI     7$              ;BR IF UNSAFE
6553  026336  032737  020000  022734          BIT     #BIT13,RMERRS+4 ;ADDRESS PLUG CHANGED ?
6554  026344  001013                          BNE     8$              ;BR IF YES
6555  026346  012746  000113                  MOV     #113,-(SP)      ;RELEASE COMMAND
6556  026352  004037  030054                  JSR     R0,WRT.RM       ;WRITE THE COMMAND INTO RMCS1
6557  026356  000000                          RMCS1                   ;REGISTER INDEX
6558  026360  026720                          SC8                     ;PARITY EXIT ADDRESS
6559  026362  011605          7$:     MOV     (SP),R5         ;PICKUP (RMAS) BEFORE THE ERROR CALL
6560  026364  004037  031172                  JSR     R0,ES.SAV       ;SAVE THE ADDRESS IN '$ESCAPE'
6561  026370  104002                          ERROR   2               ;REPORT THE UNEXPECTED ATTENTION
6562  026372  000677                          BR      SC4             ;GO CHECK FOR MORE ATA'S
6563  026374                          8$:
6564  026374  004037  031172                  JSR     R0,ES.SAV       ;SAVE THE ADDRESS IN '$ESCAPE'
6565  026400  104005                          ERROR   5               ;REPORT THE ADDRESS PLUG CHANGE
6566  026402  000673                          BR      SC4             ;CHECK FOR MORE DRIVES
6567  026404  006301          SC6:    ASL     R1              ;SETUP TO ADDRESS WORDS
6568  026406  012761  177777  023042          MOV     #-1,TIMER(R1)   ;STOP THE TIMER
6569  026414  006201                          ASR     R1              ;RESTORE THE DRIVE ADDRESS
6570  026416  004737  031102                  JSR     PC,GETREQ       ;GET THE DPB POINTER FROM THE QUEUE
6571  026422  010164  000010                  MOV     R1,RMCS2(R4)    ;SELECT DRIVE
6572  026426  004037  027674                  JSR     R0,RD.RM        ;READ THE RM03'S STATUS REG.
6573  026432  000012                          RMDS
6574  026434  026720                          SC8
6575  026436  011605                          MOV     (SP),R5         ;AND PUT IT IN R5
6576  026440  006126                          ROL     (SP)+           ;WAS THERE AN ERROR?
6577  026442  100407                          BMI     1$              ;BR IF ERROR
6578  026444  105761  022740                  TSTB    DRVACT(R1)      ;CHECK DRIVE'S STATE
6579  026450  003137                          BGT     SC11            ;BR IF DRIVE ACTIVE WITH ORDER
6580  026452  052762  100210  000016          BIS     #BIT15!BIT07!BIT03,16(R2)  ;INFORM USER OF ERROR RECOVER COMPLETION
6581  026460  000470                          BR      SC7
6582  026462  004037  027674          1$:     JSR     R0,RD.RM        ;READ ERROR REGISTER #1
6583  026466  000014                          RMER1
6584  026470  026720                          SC8
6585  026472  012605                          MOV     (SP)+,R5        ;AND SAVE IT IN R5
6586  026474  004737  030246                  JSR     PC,SVRH70       ;SAVE RH70/RM03 REGISTERS
6587  026500  012746  000111                  MOV     #111,-(SP)      ;ISSUE A DRIVE CLEAR
6588  026504  004037  030054                  JSR     R0,WRT.RM
6589  026510  000000                          RMCS1
6590  026512  026720                          SC8
6591  026514  006105          SC6A:   ROL     R5              ;WAS "UNSAFE" CONDITION =1?
```

```
6592  026516  100406                          BMI     1$              ;BRANCH IF YES
6593  026520  005702                          TST     R2              ;ANYTHING IN QUEUE ?
6594  026522  001447                          BEQ     SC7             ;BR IF NOT
6595  026524  052762  100240  000010          BIS     #BIT15!BIT07!BIT05,16(R2)       ;INFORM USER OF ERROR
6596  026532  000443                          BR      SC7
6597  026534  004037  027674          1$:     JSR     R0,RD.RM        ;READ DRIVE STATUS REG. #1
6598  026540  000012                          RMDS
6599  026542  026720                          SC8
6600  026544  011605                          MOV     (SP),R5         ;SAVE RMDS IN R5
6601  026546  006126                          ROL     (SP)+           ;"ERR"=1?
6602  026550  100011                          BPL     2$              ;BR IF NO--UNSAFE CLEARED
6603  026552  112761  177777  022750          MOVB    #-1,DRVSTA(R1)  ;DRIVE IS UNSAFE
6604  026560  004737  030246                  JSR     PC,SVRH70       ;SAVE RH70/RM03 REGISTERS
6605  026564  052762  110000  000016          BIS     #BIT15!BIT12,16(R2)     ;INFORM USER OF UNSAFE ERROR
6606  026572  000423                          BR      SC7
6607  026574  032705  010000          2$:     BIT     #BIT12,R5       ;"MOL" = 1 ?
6608  026600  001015                          BNE     3$              ;BR IF YES
6609  026602  112761  177777  022740          MOVB    #-1,DRVACT(R1)  ;ACTIVE ERROR RECOVER
6610  026610  112761  000001  022750          MOVB    #1,DRVSTA(R1)   ;ONLINE
6611  026616  006301                          ASL     R1
6612  026620  012761  072460  023042          MOV     #30000.,TIMER(R1) ;START 30 SECOND TIMER
6613  026626  006201                          ASR     R1
6614  026630  000137  026172                  JMP     SC4
6615  026634  052762  100220  000016  3$:     BIS     #BIT15!BIT07!BIT04,16(R2)       ;INFORM USER OF ERROR
6616  026642  105061  022740          SC7:    CLRB    DRVACT(R1)      ;DRIVE IS IDLE
6617  026646  004737  031006                  JSR     PC,EMPTYQ       ;DUMP THE QUEUE
6618  026652  105761  023016                  TSTB    ULDFLG(R1)      ;UNLOAD IN PROGRESS OR QUEUE?
6619  026656  003002                          BGT     1$              ;BR IF NOT
6620  026660  105061  023016                  CLRB    ULDFLG(R1)      ;CLEAR UNLOAD FLAG
6621  026664  116164  023064  000016  1$:     MOVB    ATABIT(R1),RMAS(R4)     ;CLEAR ATTENTION BIT
6622  026672  105761  022750                  TSTB    DRVSTA(R1)      ;IS THE DRIVE UNSAFE ?
6623  026676  100406                          BMI     2$              ;BR IF IT IS
6624  026700  012746  000113                  MOV     #113,-(SP)      ;RELEASE COMMAND
6625  026704  004037  030054                  JSR     R0,WRT.RM       ;WRITE THE COMMAND INTO RPCS1
6626  026710  000000                          RMCS1                   ;REGISTER INDEX
6627  026712  026720                          SC8                     ;PARITY EXIT ADDRESS
6628  026714  000137  026172          2$:     JMP     SC4             ;CHECK FOR MORE DRIVES
6629  026720  105761  022740          SC8:    TSTB    DRVACT(R1)      ;IS DRIVE IDLE?
6630  026724  001405                          BEQ     1$              ;YES--BRANCH
6631  026726  004737  031102                  JSR     PC,GETREQ       ;GET DPB POINTER
6632  026732  004737  025172                  JSR     PC,CI7          ;PROCESS THE PARITY ERROR
6633  026736  000402                          BR      2$              ;CONTINUE
6634  026740  004737  025220          1$:     JSR     PC,CI7B         ;PROCESS THE UNCORRECTABLE PARITY ERROR
6635  026744  000137  026172          2$:     JMP     SC4             ;CHECK MORE DRIVES
6636  026750  105761  023016          SC11:   TSTB    ULDFLG(R1)      ;"UNLOAD IN PROGRESS"?
6637  026754  003402                          BLE     1$              ;BRANCH IF NO
6638  026756  105061  023016                  CLRB    ULDFLG(R1)      ;CLEAR UNLOAD FLAG
6639  026762  105061  022740          1$:     CLRB    DRVACT(R1)      ;SET DRIVE IDLE
6640  026766  136137  023064  023012          BITB    ATABIT(R1),SRCHWT       ;DOING A SEARCH OPERATION FOR
6641                                                                  ;AN I/O COMMAND?
6642  026774  001012                          BNE     2$              ;BRANCH IF YES
6643  026776  004737  031124                  JSR     PC,POPQUE       ;REMOVE REQUEST FROM QUEUE
6644  027002  052762  000200  000016          BIS     #BIT07,16(R2)   ;SET "DONE" BIT
6645  027010  005737  023036                  TST     SAVEFG          ;SAVE THE REGISTERS?
6646  027014  100002                          BPL     2$              ;BRANCH IF NO
6647  027016  004737  030246                  JSR     PC,SVRH70       ;YES--SAVE ALL OF THE RH70/RM03 REG'S
```

```
6648  027022  116164  023064  000016  2$:     MOVB    ATABIT(R1),RMAS(R4)  ;CLEAR ATTENTION BIT
6649  027030  146137  023064  023012          BICB    ATABIT(R1),SRCHWT       ;CLEAR IMPLIED SEEK SET
6650  027036  006301                          ASL     R1                  ;WORD INDEX
6651  027040  012761  177777  023042          MOV     #-1,TIMER(R1)       ;STOP CLOCK
6652  027046  006201                          ASR     R1                  ;RESTORE R1
6653  027050  004737  024112                  JSR     PC,OPT              ;START A REQUEST
6654  027054  000137  026172                  JMP     SC4                 ;CHECK FOR MORE DRIVES
6655  027060  010164  000010          SC12:   MOV     R1,RMCS2(R4)        ;SELECT DRIVE
6656  027064  016437  000012  022730          MOV     RMOS(R4),RMERRS     ;SAVE THE FOUR REGISTERS THAT
6657  027072  016437  000014  022732          MOV     RMER1(R4),RMERRS+2      ;WILL TELL US SOMETHING
6658  027100  016437  000042  022734          MOV     RMER2(R4),RMERRS+4
6659  027106  016437  000040  022736          MOV     RMMR2(R4),RMERRS+6
6660  027114  004037  023326                  JSR     R0,DRVINT           ;INIT. THE STATE OF THE DRIVE
6661  027120  000401                          BR      1$                  ;TAKE ERROR EXIT
6662  027122  000207                          RTS     PC                  ;RETURN
6663  027124  005726                  1$:     TST     (SP)+               ;POP PC OFF OF THE STACK
6664  027126  000674                          BR      SC8                 ;PROCESS THE PARITY ERROR
6665  027130  006301                  SC13:   ASL     R1                  ;SETUP TO ADDRESS WORDS
6666  027132  012761  177777  023042          MOV     #-1,TIMER(R1)       ;STOP THE TIMER
6667  027140  006201                          ASR     R1                  ;
6668  027142  010164  000010                  MOV     R1,RMCS2(R4)        ;SELECT THE DRIVE
6669  027146  116164  023064  000016          MOVB    ATABIT(R1),RMAS(R4)  ;CLEAR THE ATTENTION BIT
6670  027154  105761  022770          1$:     TSTB    DPINT(R1)           ;INITIALIZING THE DRIVE ?
6671  027160  001424                          BEQ     2$                  ;BR IF NOT
6672  027162  105061  022770                  CLRB    DPINT(R1)           ;CLEAR THE INIT INDICATOR
6673  027166  004037  023326                  JSR     R0,DRVINT           ;GO INIT THE DRIVE
6674  027172  000240                          NOP                         ;DUMMY PARITY ERROR RETURN
6675  027174  105761  022750                  TSTB    DRVSTA(R1)          ;DRIVE ONLINE ?
6676  027200  003014                          BGT     2$                  ;BR IF YES -- START ORDER
6677  027202  005702                          TST     R2                  ;QUEUE ENTRY FOR THE DRIVE
6678  027204  001426                          BEQ     3$                  ;BR IF NOT
6679  027206  004737  031102                  JSR     PC,GETREQ           ;GET DPB ADDRESS
6680  027212  052762  140000  000016          BIS     #BIT15!BIT14,16(R2)  ;INFORM USER THAT DRIVE OFFLINE
6681  027220  004737  030246                  JSR     PC,SVRH70           ;SAVE THE REGISTERS
6682  027224  004737  031006                  JSR     PC,EMPTYQ           ;EMPTY THE REQUEST QUEUE
6683  027230  000414                          BR      3$
6684  027232  032764  004000  000000  2$:     BIT     #BIT11,RMCS1(R4)        ;DVA SET ?
6685  027240  001006                          BNE     4$                  ;SET THEN CALL OPT
6686  027242  006301                          ASL     R1
6687  027244  012761  023420  023042          MOV     #10000.,TIMER(R1)
6688  027252  006201                          ASR     R1
6689  027254  000402                          BR      3$
6690  027256  004737  024112          4$:     JSR     PC,OPT              ;START THE PENDING REQUEST
6691  027262  000137  026172          3$:     JMP     SC4                 ;PROCESS OTHER DRIVES
6692
6693                                  ;/RM03 TIMER ROUTINE
6694                                  ;CALL
6695                                  ;       MOV     #TIME,-(SP)         ;ELASPED TIME IN MILLISECONDS ON THE STACK
6696                                  ;       JSR     PC,RPTMR            ;CALL RM03 TIME ROUTINE
6697
6698  027266  005737  023014          RPTMR:  TST     ACTDRV              ;CHECK "ACTDRV & ACTSTR"
6699  027272  001027                          BNE     4$                  ;IF NON ZERO EXIT
6700  027274  112737  000001  023015          MOVB    #1,ACTSTR           ;SET "ACTSTR"
6701  027302  104412                          SAVREG                      ;SAVE R0 - R5
6702  027304  005001                          CLR     R1                  ;START WITH DRIVE 0
6703  027306  005003                          CLR     R3
```

```
6704  027310  005763  023042          1$:     TST     TIMER(R3)       ;IS THE TIMER RUNNING?
6705  027314  002406                          BLT     2$              ;BRANCH IF NO
6706  027316  166663  000002  023042          SUB     2(SP),TIMER(R3) ;COUNT THE INTERVAL
6707  027324  003002                          BGT     2$              ;BR IF NO SOFTWARE TIMEOUT
6708  027326  004737  027356                  JSR     PC,STO          ;CALL SOFTWARE TIMEOUT ROUTINE
6709  027332  005201                  2$:     INC     R1              ;MOVE TO NEXT DRIVE
6710  027334  005723                          TST     (R3)+
6711  027336  022701  000010                  CMP     #8.,R1          ;OUT OF DRIVES?
6712  027342  003362                          BGT     1$              ;BRANCH IF NO
6713  027344  104413                  3$:     RESREG                  ;RESTORE R0 - R5
6714  027346  105037  023015                  CLRB    ACTSTR          ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
6715  027352  012616                  4$:     MOV     (SP)+,(SP)      ;ADJUST THE STACK
6716  027354  000207                          RTS     PC              ;RETURN
6717
6718                                  ;SOFTWARE TIMEOUT ROUTINE
6719                                  ;
6720                                  ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
6721                                  ;       OR GREATER
6722                                  ;
6723                                  ;CALL:  STO
6724                                  ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
6725                                  ;       JSR     PC,STO          ;CALL
6726                                  ;       RETURN
6727
6728  027356  010146                  STO:    MOV     R1,-(SP)        ;SAVE R1
6729  027360  010246                          MOV     R2,-(SP)        ;SAVE R2
6730  027362  010346                          MOV     R3,-(SP)        ;SAVE R3
6731  027364  010446                          MOV     R4,-(SP)        ;SAVE R4
6732  027366  013704  023076                  MOV     RMADR,R4        ;GET ADDRESS OF "RMCS1"
6733  027372  010164  000010                  MOV     R1,RMCS2(R4)    ;SELECT THE DRIVE
6734  027376  004037  027674                  JSR     R0,RD.RM        ;READ "DRIVE STATUS REG"
6735  027402  000012                          RMDS
6736  027404  027556                          STOS
6737  027406  105726                          TSTB    (SP)+           ;IS "DRY"=1?
6738  027410  100434                          BMI     STO2            ;BR IF YES
6739  027412  105761  022770          STO1:   TSTB    DPINT(R1)       ;TRYING TO INTIALIZE THE DRIVE ?
6740  027416  001031                          BNE     STO2            ;BR IF YES
6741  027420  105761  023000                  TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
6742  027424  001026                          BNE     STO2            ;BR IF YES
6743  027426  013702  023010                  MOV     TRNSWT,R2       ;PICKUP TRANSFER WAIT QUEUE
6744  027432  020137  023062                  CMP     R1,DTUW         ;TRANSFER UNDERWAY ON THIS DRIVE?
6745  027436  001404                          BEQ     1$              ;BRANCH IF YES
6746  027440  000137  027662                  JMP     STO9            ;IF NOT DON'T BOTHER DRIVES
6747  027444  004737  031102                  JSR     PC,GETREQ       ;GET DPB ADDRESS
6748  027450  052762  101000  000016  1$:     BIS     #BIT15!BIT09,16(R2) ;SET THE ERROR FLAGS
6749  027456  004737  030246                  JSR     PC,SVRH70       ;SAVE RH70/RM03 REGISTERS
6750  027462  012764  000040  000010          MOV     #BIT05,RMCS2(R4) ;"INIT" THE MASS BUS
6751  027470  105061  022740                  CLRB    DRVACT(R1)      ;DRIVE IS IDLE
6752  027474  105061  023016                  CLRB    ULDFLG(R1)      ;CLEAR THE UNLOAD FLAG
6753  027500  000470                          BR      STO9            ;DON'T BOTHER OTHER DRIVES
6754  027502  116405  000016          STO2:   MOVB    RMAS(R4),R5     ;READ ATTENTION REG
6755  027506  136105  023064                  BITB    ATABIT(R1),R5   ;IS ATTENTION FOR THIS DRIVE UP ?
6756  027512  001007                          BNE     STO3            ;YES--BRANCH
6757  027514  105761  022770                  TSTB    DPINT(R1)       ;TRYING TO INTIALIZE THE DRIVE ?
6758  027520  001021                          BNE     STO6            ;BR IF YES - DRIVE NOT ONLINE
6759  027522  105761  023000                  TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
```

```
6760  027526  001035                          BNE     ST07            ;BR IF YES - NO RESPONSE TO REQUEST
6761  027530  000454                          BR      ST09            ;OTHER WISE EXIT
6762  027532  105761  022770          ST03:   TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
6763  027536  001003                          BNE     1$              ;BR IF INIT PENDING
6764  027540  105761  023000                  TSTB    DPRQS(R1)       ;PORT REQUEST PENDING ?
6765  027544  001446                          BEQ     ST09            ;BR IF NOT
6766  027546  012763  177777  023042  1$:     MOV     #-1,TIMER(R3)   ;STOP THE TIMER
6767  027554  000442                          BR      ST09            ;EXIT
6768  027556  004737  025272          ST05:   JSR     PC,CIB          ;GO HANDLE THE PARITY ERROR
6769  027562  000437                          BR      ST09
6770  027564  105061  022770          ST06:   CLRB    DPINT(R1)       ;CLEAR THE INITIALIZE INDICATOR
6771  027570  105061  022750                  CLRB    DRVSTA(R1)      ;SET UNIT OFFLINE
6772  027574  012763  177777  023042          MOV     #-1,TIMER(R3)   ;STOP THE TIMER
6773  027602  004737  031102                  JSR     PC,GETREQ       ;GET THE DPB ADDRESS
6774  027606  005702                          TST     R2              ;REQUEST IN QUEUE ?
6775  027610  001424                          BEQ     ST09            ;BR IF NOT
6776  027612  052762  140000  000016          BIS     #BIT15!BIT14,16(R2)  ;INFORM THE USER DRIVE NOT AVAILABLE
6777  027620  000414                          BR      ST08            ;FINISH
6778  027622  012763  177777  023042  ST07:   MOV     #-1,TIMER(R3)   ;STOP THE TIMER
6779  027630  105061  023000                  CLRB    DPRQS(R1)       ;CLEAR PORT REQUEST INDICATOR
6780  027634  004737  031102                  JSR     PC,GETREQ       ;GET DPB ADDRESS
6781  027640  005702                          TST     R2              ;QUEUE ENTRY FOR DRIVE ?
6782  027642  001407                          BEQ     ST09            ;BR IF NONE
6783  027644  012762  100004  000016          MOV     #BIT15!BIT2,16(R2)  ;INFORM USER OF PORT REQUEST ERROR
6784  027652  004737  031006          ST08:   JSR     PC,EMPTYQ       ;CLEAR THE QUEUE FOR THE DRIVE
6785  027656  004737  030246                  JSR     PC,SVRH70       ;SAVE THE REGISTERS
6786  027662  012604                  ST09:   MOV     (SP)+,R4        ;RESTORE R4
6787  027664  012603                          MOV     (SP)+,R3        ;RESTORE R3
6788  027666  012602                          MOV     (SP)+,R2        ;RESTORE R2
6789  027670  012601                          MOV     (SP)+,R1        ;RESTORE R1
6790  027672  000207                          RTS     PC              ;RETURN
6791
6792                                  ;ROUTINE TO READ A RH70/RM03 REGISTER
6793                                  ;
6794                                  ;CALL
6795                                  ;       JSR     R0,RD.RM        ;GO READ A REGISTER
6796                                  ;       INDEX                   ;REG. INDEX FROM BASE
6797                                  ;       ERRADR                  ;ERROR ADDRESS--PROCESS ERROR STARTING
6798                                  ;                               ;AT THIS ADDRESS
6799                                  ;       RETURN                  ;CONTENTS OF REG. IS ON THE STACK
6800                                  ;
6801  027674  013737  023074  030042  RD.RM:   MOV     MCPEMX,RD.RM2   ;MAX. RETRYS ALLOWED
6802  027702  011646                          MOV     (SP),-(SP)      ;SAVE R0 FOR RETURN
6803  027704  013737  023076  027720          MOV     RMADR,RD.ADR    ;FORM THE DESIRED ADDRESS
6804  027712  062037  027720                  ADD     (R0)+,RD.ADR    ;USING THE BASE AND THE INDEX
6805  027716  013727                  RD.RM1: MOV     @(PC)+,(PC)+    ;READ THE DESIRED REGISTER OF THE RM03
6806  027720  000000                  RD.ADR: .WORD   0               ;ADDRESS IS FORMED HERE
6807  027722  000000                  RD.WRD: .WORD   0               ;REG. CONTENTS PUT HERE
6808  027724  013766  027722  000002          MOV     RD.WRD,2(SP)    ;RETURN IT TO THE USER
6809  027732  013746  023076                  MOV     RMADR,-(SP)     ;PUT THE ADDRESS ON THE STACK
6810  027736  062716  000010                  ADD     #RMCS2,(SP)     ;FORM THE ADDRESS OF RMCS2
6811  027742  032736  010000                  BIT     #BIT12,@(SP)+   ;CHECK THE 'NED' BIT
6812  027746  001037                          BNE     RD.RM3          ;BR IF DRIVE NON-EXISTENT
6813  027750  017746  173122                  MOV     @RMADR,-(SP)    ;READ RMCS1
6814  027754  032716  020000                  BIT     #BIT13,(SP)     ;DID MCPE SET?
6815  027760  001002                          BNE     1$              ;BRANCH IF YES
```

```
                                                        CMP     (SP)+,(R0)+     ;ADJUST FOR RETURN
                                                        BR      RD.RM4          ;EXIT
                                                1$:
                                                        JSR     R0,ES.SAV       ;SAVE THE ADDRESS IN 'ESCAPE'
                                                        ERROR   3               ;REPORT "MCPE" ERROR
                                                        TST     DTUW            ;DATA TRANSFER UNDERWAY?
                                                        BMI     2$              ;NO--BRANCH
                                                        BIT     #BIT14,(SP)     ;NO--"TRE"=1?
                                                        BEQ     2$              ;NO--BRANCH
                                                        TST     (SP)+           ;YES--CLEAN OFF THE STACK A'ND
                                                        BR      RD.RM3          ;TAKE THE FATAL ERROR EXIT
                                                2$:     BIS     #BIT14,(SP)     ;CLEAR "MCPE" BY SENDING A "1" TO "TRE"
                                                        SWAB    (SP)            ;POSITION BEFORE WRITING
                                                        MOV     RMADR,3$        ;FORM ADDRESS OF HIGH BYTE
                                                        INC     3$
                                                        MOVB    (SP)+,@(PC)+    ;WRITE THE HIGH BYTE OF RMCS1
                                                3$:     .WORD   0               ;ADDRESS STORAGE
                                                        DEC     (PC)+           ;EXCEEDED MAX. RETRYS
                                                RD.RM2: .WORD   3
                                                        BGE     RD.RM1          ;BRANCH IF NO
                                                RD.RM3: MOV     (R0),R0         ;FATAL ERROR EXIT
                                                        MOV     (SP)+,(SP)
                                                RD.RM4: RTS     R0

                                                ;ROUTINE TO WRITE A REGISTER
                                                ;
                                                ;CALL
                                                ;       MOV     DATA,-(SP)      ;DATA TO BE LOADED ON THE STACK
                                                ;       JSR     R0,WRT.RM       ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
                                                ;       INDEX                   ;INDEX OF THE REGISTER TO BE LOADED
                                                ;       ERRADR                  ;ADDRESS TO RETURN TO ON AN ERROR
                                                ;       RETURN                  ;ERROR FREE RETURN

030054  013737  023074  030232  WRT.RM: MOV     MCPEMX,WRT.R2   ;MAX RETRYS ALLOWED
030062  016637  000002  030142          MOV     2(SP),WRT.WD    ;SAVE THE WORD TO WRITE
030070  012616                          MOV     (SP)+,(SP)      ;ADJUST THE STACK
030072  012037  030144                  MOV     (R0)+,WRT.AD    ;GET INDEX OF REGISTER TO BE WRITTEN
030076  001015                          BNE     1$              ;BRANCH IF NOT RMCS1
030100  122737  000150  030142          CMPB    #150,WRT.WD     ;IS THE COMMAND FOR DATA TRANSFERS?
030106  002411                          BLT     1$              ;YES--DON'T GET THE OLD A16 & A17, & PSEL
030110  004037  027674                  JSR     R0,RD.RM        ;NO---COMBINE A16&A17, & PSEL WITH
030114  000000                          RMCS1                   ;THE COMMAND BEFORE SENDING IT TO
030116  030236                          WRT.R3                  ;THE RH70/RM03
030120  000316                          SWAB    (SP)
030122  042716  177770                  BIC     #^C7,(SP)
030126  112637  030143                  MOVB    (SP)+,WRT.WD+1
030132  063737  023076  030144  1$:     ADD     RMADR,WRT.AD    ;FORM THE ADDRESS OF THE DISK REG.
030140  012737                  WRT.R1: MOV     (PC)+,@(PC)+    ;LOAD THE DESIRED REG.
030142  000000                  WRT.WD: .WORD   0               ;WORD TO WRITE GOES HERE
030144  000000                  WRT.AD: .WORD   0               ;ADDRESS IS FORMED HERE
030146  013746  023076                  MOV     RMADR,-(SP)     ;PUT THE ADDRESS ON THE STACK
030152  062716  000010                  ADD     #RMCS2,(SP)     ;FORM THE ADDRESS OF RMCS2
030156  032736  010000                  BIT     #BIT12,@(SP)+   ;CHECK THE 'NED' BIT
030162  0C 025                          BNE     WRT.R3          ;BR IF DRIVE NON-EXISTENT
030164  004037  027674                  JSR     R0,RD.RM        ;CHECK FOR PARITY ERROR ON WRITE
030170  000014                          RMER1
```

```
                                                WRT.R3
                                                BIT     #BIT03,(SP)+
                                                BEQ     WRT.R4          ;BRANCH IF "PAR=0"
                                                MOV     -2(R0),1$       ;PICKUP THE INDEX
                                                JSR     R0,RD.RM        ;READ THE REG.
                                        1$:     .WORD   0               ;REG. INDEX
                                                WRT.R3                  ;RETURN TO THIS ADDRESS ON ERROR
                                                JSR     R0,ES.SAV       ;SAVE THE ADDRESS IN '$ESCAPE'
                                                ERROR   4               ;REPORT THE PARITY ON WRITE ERROR
                                                TST     (SP)+           ;CLEAR OFF THE STACK
                                                DEC     (PC)+           ;DECREMENT THE ERROR COUNT
                                        WRT.R2: .WORD   3               ;RETRY COUNTER
                                                BGE     WRT.R1          ;TRY AGAIN IF NOT FINISHED
                                        WRT.R3: MOV     (R0),R0         ;TAKE THE "PARITY ON WRITE" ERROR EXIT
                                                BR      WRT.R5          ;EXIT
                                        WRT.R4: TST     (R0)+           ;ADJUST FOR ERROR FREE EXIT
                                        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

                                        SVRH70: SAVREG                  ;SAVE R0 - R5
                                                TST     R2              ;QUEUE ENTRY FOR THE DRIVE ?
                                                BEQ     6$              ;BR IF NONE
                                                MOV     RMADR,R4
                                                MOVB    (R2),RMCS2(R4)  ;SELECT DRIVE
                                                MOV     14(R2),R3       ;GET THE ERROR TABLE POINTER
                                                BEQ     6$              ;EXIT IF NO ADDRESS
                                                CLR     3$              ;COUNTER & POINTER
                                        1$:     CMP     3$,#RMDB        ;REACHED THE BUFFER REGISTER ?
                                                BNE     2$              ;BR IF NOT
                                                BIT     #BIT07,RMCS2(R4)  ;'OR' SET ?
                                                BNE     2$              ;BR IF SET
                                                CLR     (R3)+           ;STORE RMDB AS ZEROES
                                                BR      4$              ;CONTINUE
                                        2$:     JSR     R0,RD.RM        ;READ THE SELECTED REGISTER
                                        3$:     .WORD   0               ;REGISTER INDEX
                                                5$                      ;ERROR RETURN ADDRESS
                                                MOV     (SP)+,(R3)+     ;STORE THE REGISTER CONTENTS
                                        4$:     CMP     3$,#RMEC2       ;REACHED THE END ?
                                                BEQ     6$              ;BR IF YES
                                                ADD     #2,3$           ;INCREMENT THE REGISTER INDEX
                                                BR      1$              ;CONTINUE READING THE REGISTERS
                                        5$:     JSR     PC,CI7          ;PROCESS THE UNCORRECTABLE PARITY ERROR
                                        6$:     RESREG                  ;RESTORE R0 - R5
                                                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
```

```
6928  030364  010446                  SET.IE: MOV     R4,-(SP)        ;SAVE R4
6929  030366  013704  023076                  MOV     RMADR,R4        ;PICKUP ADDRESS OF RMCS1
6930  030372  010164  000010                  MOV     R1,RMCS2(R4)    ;SELECT DRIVE
6931  030376  011446                          MOV     (R4),-(SP)      ;READ RMCS1
6932  030400  052716  040000                  BIS     #BIT14,(SP)     ;SET THE "TRE" BIT OF THE WORD READ
6933  030404  000316                          SWAB    (SP)            ;ADJUST FOR DATO
6934  030406  112714  000100                  MOVB    #BIT06,(R4)     ;SET "IE"
6935  030412  032764  010000  000010          BIT     #BIT12,RMCS2(R4) ;IS "NED"=1?
6936  030420  001002                          BNE     1$              ;YES  CLEAR "TRE"
6937  030422  005726                          TST     (SP)+           ;CLEAN OFF THE STACK
6938  030424  000402                          BR      2$
6939  030426  112664  000001          1$:     MOVB    (SP)+,1(R4)     ;CLEAR "TRE"
6940  030432  012604                  2$:     MOV     (SP)+,R4        ;RESTORE R4
6941  030434  000207                          RTS     PC              ;RETURN TO CALLER
6942
6943                                  ;QUEUE COUNT
6944  030436      000                 QCNT:   .BYTE   0               ;DRIVE 0
6945  030437      000                         .BYTE   0               ;DRIVE 1
6946  030440      000                         .BYTE   0               ;DRIVE 2
6947  030441      000                         .BYTE   0               ;DRIVE 3
6948  030442      000                         .BYTE   0               ;DRIVE 4
6949  030443      000                         .BYTE   0               ;DRIVE 5
6950  030444      000                         .BYTE   0               ;DRIVE 6
6951  030445      000                         .BYTE   0               ;DRIVE 7
6952
6953                                  ;QUEUE INPUT POINTERS
6954
6955  030446  030530                  QINPT:  .WORD   QDRV0            ;DRIVE 0
6956  030450  030550                          .WORD   QDRV1           ;DRIVE 1
6957  030452  030570                          .WORD   QDRV2           ;DRIVE 2
6958  030454  030610                          .WORD   QDRV3           ;DRIVE 3
6959  030456  030630                          .WORD   QDRV4           ;DRIVE 4
6960  030460  030650                          .WORD   QDRV5           ;DRIVE 5
6961  030462  030670                          .WORD   QDRV6           ;DRIVE 6
6962  030464  030710                          .WORD   QDRV7           ;DRIVE 7
6963
6964                                  ;QUEUE OUTPUT POINTERS
6965
6966  030466  030530                  QOUTPT: .WORD   QDRV0            ;DRIVE 0
6967  030470  030550                          .WORD   QDRV1           ;DRIVE 1
6968  030472  030570                          .WORD   QDRV2           ;DRIVE 2
6969  030474  030610                          .WORD   QDRV3           ;DRIVE 3
6970  030476  030630                          .WORD   QDRV4           ;DRIVE 4
6971  030500  030650                          .WORD   QDRV5           ;DRIVE 5
6972  030502  030670                          .WORD   QDRV6           ;DRIVE 6
6973  030504  030710                          .WORD   QDRV7           ;DRIVE 7
6974
6975  030506  030530                  QSTART: .WORD   QDRV0            ;DRIVE 0 START ADDRESS
6976  030510  030550                  QSTOP:  .WORD   QDRV1           ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
6977  030512  030570                          .WORD   QDRV2           ;STOP DRIVE 1--START DRIVE 2
6978  030514  030610                          .WORD   QDRV3           ;STOP DRIVE 2--START DRIVE 3
6979  030516  030630                          .WORD   QDRV4           ;STOP DRIVE 3--START DRIVE 4
6980  030520  030650                          .WORD   QDRV5           ;STOP DRIVE 4--START DRIVE 5
6981  030522  030670                          .WORD   QDRV6           ;STOP DRIVE 5--START DRIVE 6
6982  030524  030710                          .WORD   QDRV7           ;STOP DRIVE 6--START DRIVE 7
6983  030526  030730                          .WORD   QTERM           ;STOP DRIVE 7
```

```
                                        ;DRIVE REQUEST QUEUES
                                        ;
030530  000010                  QDRV0:  .BLKW   10
030550  000010                  QDRV1:  .BLKW   10
030570  000010                  QDRV2:  .BLKW   10
030610  000010                  QDRV3:  .BLKW   10
030630  000010                  QDRV4:  .BLKW   10
030650  000010                  QDRV5:  .BLKW   10
030670  000010                  QDRV6:  .BLKW   10
030710  000010                  QDRV7:  .BLKW   10
        030730                  QTERM=.
                                        ;
                                        ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
                                        ;
                                        ;CALL
                                        ;       JSR     PC,CLRQUE
                                        ;
030730  104412                  CLRQUE: SAVREG                  ;SAVE R0 - R5
030732  012702  030436                  MOV     #QCNT,R2        ;ZERO THE QUEUE COUNTS
030736  005022                          CLR     (R2)+           ;DRIVES 0 & 1
030740  005022                          CLR     (R2)+           ;DRIVES 2 & 3
030742  005022                          CLR     (R2)+           ;DRIVES 4 & 5
030744  005022                          CLR     (R2)+           ;DRIVES 6 & 7
030746  012703  000010                  MOV     #8.,R3          ;MOVE THE STARTING
030752  012701  030506                  MOV     #QSTART,R1      ;ADDRESS OF THE QUEUE INTO
030756  012122                  1$:     MOV     (R1)+,(R2)+     ;THE QUEUE INPUT POINTER
030760  005303                          DEC     R3
030762  001375                          BNE     1$
030764  012703  000010                  MOV     #8.,R3          ;MOVE THE STARTING ADDRESS
030770  012701  030506                  MOV     #QSTART,R1      ;OF THE QUEUE INTO THE
030774  012122                  2$:     MOV     (R1)+,(R2)+     ;QUEUE OUTPUT POINTER
030776  005303                          DEC     R3
031000  001375                          BNE     2$
031002  104413                          RESREG                  ;RESTORE R0 - R5
031004  000207                          RTS     PC
                                        ;
                                        ;EMPTY THE QUEUE SPECIFIED BY R1
                                        ;
                                        ;CALL
                                        ;       MOV     DRVNUM,R1       ;DRIVE NUMBER TO R1
                                        ;       JSR     PC,EMPTYQ
                                        ;
031006  105061  030436          EMPTYQ: CLRB    QCNT(R1)        ;CLEAR NUMBER OF ITEMS IN QUEUE
031012  006301                          ASL     R1
031014  016161  030446  030466          MOV     QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
031022  006201                          ASR     R1
031024  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
```

```
7040                                    ;       RETURN2                 ;RETURN HERE IF REQUEST IS IN QUEUE
7041
7042  031026  122761  000010  030436   DRVQUE: CMPB    #10,QCNT(R1)    ;IS QUEUE FULL?
7043  031034  001421                           BEQ     2$              ;BR IF YES-TAKE RETURN1
7044  031036  105261  030436                   INCB    QCNT(R1)        ;INCREMENT QUEUE COUNT
7045  031042  006301                           ASL     R1
7046  031044  010271  030446                   MOV     R2,@QINPT(R1)   ;PUT THIS REQUEST IN QUEUE
7047  031050  062761  000002  030446           ADD     #2,QINPT(R1)    ;UPDATE THE QUEUE POINTER
7048  031056  026161  030446  030510           CMP     QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
7049  031064  001003                           BNE     1$              ;BRANCH IF NO
7050  031066  016161  030506  030446           MOV     QSTART(R1),QINPT(R1) ;YES--RESET POINTER
7051  031074  006201                   1$:     ASR     R1
7052  031076  005720                           TST     (R0)+           ;TAKE RETURN 2
7053  031100  000200                   2$:     RTS     R0              ;RETURN TO USER
7054
7055                                    ;ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
7056                                    ;
7057                                    ;CALL
7058                                    ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
7059                                    ;       JSR     PC,GETREQ       ;GO GET THE REQUEST
7060                                    ;       RETURN                  ;R2="DPB" ADDRESS OF THE REQUEST
7061                                    ;                               ;R2=0 IF NO REQUEST IN QUEUE
7062
7063  031102  005002                   GETREQ: CLR     R2
7064  031104  105761  030436                   TSTB    QCNT(R1)        ;IS THERE ANY REQUEST IN QUEUE?
7065  031110  001404                           BEQ     2$              ;NO---BRANCH
7066  031112  006301                   1$:     ASL     R1
7067  031114  017102  030466                   MOV     @QOUTPT(R1),R2  ;PICKUP "DPB" POINTER FOR THIS DRIVE
7068  031120  006201                           ASR     R1
7069  031122  000207                   2$:     RTS     PC              ;RETURN TO USER
7070
7071                                    ;ROUTINE TO "POP" THE REQUEST FROM QUEUE
7072                                    ;
7073                                    ;CALL
7074                                    ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
7075                                    ;       JSR     PC,POPQUE       ;CALL TO REMOVE REQUEST
7076                                    ;       RETURN                  ;R2=ADDRESS OF DPB REMOVED
7077
7078  031124  105361  030436           POPQUE: DECB    QCNT(R1)        ;DECREMENT QUEUE COUNT
7079  031130  006301                           ASL     R1
7080  031132  017102  030466                   MOV     @QOUTPT(R1),R2  ;GET THE "DPB" POINTER
7081  031136  005071  030466                   CLR     @QOUTPT(R1)     ;REMOVE DPB ADDRESS FROM THE QUEUE
7082  031142  062761  000002  030466           ADD     #2,QOUTPT(R1)   ;UPDATE THE QUEUE POINTER
7083  031150  026161  030466  030510           CMP     QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
7084  031156  001003                           BNE     1$              ;NO--BRANCH TO EXIT
7085  031160  016161  030506  030466           MOV     QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
7086  031166  006201                   1$:     ASR     R1
7087  031170  000207                           RTS     PC              ;RETURN TO USER
7088
7089                                    ;ROUTINE TO SAVE THE CONTENTS OF '$ESCAPE' WHEN THE DRIVER
7090                                    ;REPORTS AN ERROR DIRECTLY.
7091                                    ;
7092                                    ;CALL
7093                                    ;       JSR     R0,ES.SAV
7094                                    ;       ERROR   N               ;;THE ERROR CALL
7095                                    ;       RETURN                  ;THE RETURN IS PAST THE ERROR CALL
```

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

                                ;;**********************************************************************

                                .SBTTL  DATA, CONTROL, & STATUS BLOCKS

                                ;;**********************************************************************

                                .SBTTL  ROUTINE TO SIZE MEMORY

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

                                $SIZE:  MOV     R0,-(SP)        ;;SAVE R0 ON THE STACK
                                        MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
                                        MOV     @#ERRVEC,-(SP)  ;;SAVE PRESENT ERROR VECTOR PS & PC
                                        MOV     @#ERRVEC+2,-(SP)
                                        MOV     SP,R0           ;;SAVE THE STACK POINTER
                                ;;SET THE ERRVEC PS TO THE PRESENT PS
                                        TRAP                    ;;PUSH OLD PSW AND PC ON STACK
                                        MOV     (SP)+,@#ERRVEC+2        ;;SAVE THE PSW IN @#ERRVEC+2
                                        MOV     #2$,@#ERRVEC    ;;SET FOR TIMEOUT
                                        MOV     #20000,R1       ;;FIRST ADDRESS
                                1$:     TST     (R1)            ;;TEST THIS ADDRESS
                                        TST     (R1)+           ;;STEP TO NEXT ADDRESS
                                        BR      1$              ;;TRY ANOTHER
                                2$:     SUB     #2,R1           ;;DROP BACK
                                        MOV     R0,SP           ;;RESTORE THE STACK
                                        MOV     (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
                                        MOV     (SP)+,@#ERRVEC
                                        MOV     R1,$LSTAD       ;;LAST ADDRESS
                                        MOV     (SP)+,R1        ;;RESTORE R1
                                        MOV     (SP)+,R0        ;;RESTORE R0
                                        RTS     PC
                                $LSTAD: .WORD   0               ;;CONTAINS THE LAST ADDRESS

                                .SBTTL  BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH11
                                ;THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS
                                ;OF THE RH11 IS SETUP FOR THE PROPER ADDRESS.
                                ;IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
                                ;REQUIRED.
                                ;NOTE: THIS ROUTINE DESTROYS R0-R4
                                ;CALL
                                ;
                                ;       JSR     PC,BUSADR
                                ;       RETURN
```

```
7152  031314  104412                  BUSADR: SAVREG           ;SAVE ALL REG
7153  031316  012700  001274          1$:     MOV     #$RMADR,R0       ;FIRST ADDRESS
7154  031322  104401  031464                  TYPE    ,MRMCS1          ;"RMCS1="
7155  031326  011046                  MOV     (R0),-(SP)       ;PRESENT RMCS1 ADDRESS
7156  031330  104402                  TYPOC                    ;TYPE IT
7157  031332  104401  036121          TYPE    ,LINSP           ;2 SPACES
7158  031336  104411                  RDLIN                    ;GET THE ENTRY
7159  031340  012601                  MOV     (SP)+,R1         ;ADDRESS OF ASCII TEXT
7160  031342  004537  031506          JSR     R5,CK.NUM        ;ENTER AND STORE THE NEW ADDRESS
7161  031346  000763                  BR      1$               ;ERROR EXIT
7162  031350  012700  001276          2$:     MOV     #$RMVEC,R0       ;VECTOR ADDRESS
7163  031354  104401  031475          TYPE    ,MRHVEC          ;"RHVEC="
7164  031360  011046                  MOV     (R0),-(SP)       ;PRESENT RH11 VECTOR ADDRESS ON THE STACK
7165  031362  104402                  TYPOC                    ;TYPE IT
7166  031364  104401  036121          TYPE    ,LINSP           ;2 SPACES
7167  031370  104411                  RDLIN                    ;READ THE ENTRY
7168  031372  012601                  MOV     (SP)+,R1         ;ASCII TEXT ADDRESS
7169  031374  004537  031506          JSR     R5,CK.NUM        ;ENTER AND STORE NEW ADDRESS
7170  031400  000763                  BR      2$               ;ERROR EXIT
7171  031402  012700  001274          3$:     MOV     #$RMADR,R0       ;FIRST ADDRESS OF NEW PARAMETERS
7172  031406  012701  023076                  MOV     #RMADR,R1        ;FIRST ADDRESS OF WHERE TO PUT THEM
7173  031412  013705  000004                  MOV     @#ERRVEC,R5      ;SAVE ERROR VECTOR
7174  031416  012737  031442  000004          MOV     #4$,@#ERRVEC     ;LOAD NEW VECTOR ADDRESS
7175  031424  005777  147644                  TST     @$RMADR          ;LEGAL I/O ADDRESS ?
7176  031430  010537  000004                  MOV     R5,@#ERRVEC      ;YES,IF NOT TRAP TO TIMEOUT
7177  031434  012021                  MOV     (R0)+,(R1)+      ;LOAD THE BUS ADDRESS
7178  031436  012021                  MOV     (R0)+,(R1)+      ;LOAD VECTOR ADDRESS
7179  031440  000407                  BR      6$               ;COMMON EXIT
7180  031442  012716  031450          4$:     MOV     #5$,(SP)         ;SET RETURN ADDRESS
7181  031446  000002                  RTI                      ;RETRUN FROM TIME OUT TRAP
7182  031450  104006                  5$:     ERROR   6                ;NO RESPONSE FROM RMADR
7183  031452  010537  000004                  MOV     R5,@#ERRVEC      ;RESTORE THE TIME OUT TRAP
7184  031456  000717                  BR      1$               ;TRY AGAIN
7185  031460  104413                  6$:     RESREG                   ;RESTORE ALL REG
7186  031462  000207                  RTS     PC
7187
7188  031464  046522  051503  020061  MRMCS1: .ASCIZ  @RMCS1 = @
7189  031472  020075     000
7190  031475     122  053110  041505  MRHVEC: .ASCIZ  @RHVEC = @
7191  031502  036440  000040
7192
7193                                  .SBTTL  CK.NUM - CHECK NUMBER (OCTAL)
7194                                  ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
7195                                  ;AND FORMS AN OCTAL NUMBER IN R2
7196                                  ;CALL:
7197                                  ;       MOV     #ADR,R1          ;ADDRESS OF ASCIZ STRING
7198                                  ;       JSR     R5,CK.NUM        ;R5 CHANGED
7199                                  ;       RET                      ;ERROR EXIT
7200                                  ;       RET                      ;NORMAL EXIT
7201  031506  010246                  CK.NUM: MOV     R2,-(SP)         ;SAVE R2
7202  031510  010346                  MOV     R3,-(SP)         ;SAVE R3
7203  031512  010446                  MOV     R4,-(SP)         ;SAVE R4
7204  031514  012703  000006          MOV     #6,R3            ;MAX OCTAL DIGITS IN THE NUMBER
7205  031520  005002                  CLR     R2               ;FINAL OCTAL VALUE
7206  031522  112104                  1$:     MOVB    (R1)+,R4         ;GET CURRENT POINTED BYTE
7207  031524  001424                  BEQ     3$               ;BRANCH,IF TERMINATOR DETECTED
```

```
7208  031526  120427  000060                CMPB    R4,#'0          ;SMALLER THAN ASCII-0 ?
7209  031532  103425                        BLO     5$              ;YES,ERROR EXIT
7210  031534  120427  000067                CMPB    R4,#'7          ;LARGER THAN ASCII-7 ?
7211  031540  101022                        BHI     5$              ;YES,ERROR EXIT
7212  031542  006302                        ASL     R2              ;SHIFT LEFT
7213  031544  103420                        BCS     5$              ;
7214  031546  006302                        ASL     R2              ;ONE
7215  031550  103416                        BCS     5$              ;
7216  031552  006302                        ASL     R2              ;OCTAL DIGIT
7217  031554  103414                        BCS     5$              ;ERROR IF CARRY BIT SET
7218  031556  042704  177770                BIC     #177770,R4      ;CHOP OFF HIGHER BITS
7219  031562  060402                        ADD     R4,R2           ;APPENDING CURRENT DIGIT TO NUMBER
7220  031564  005303                        DEC     R3              ;DECREMENT BYTE COUNT
7221  031566  001401                        BEQ     2$              ;BRANCH IF LAST BYTE
7222  031570  000754                        BR      1$              ;LOOPING BACK
7223  031572  112104              2$:       MOVB    (R1)+,R4        ;CHECK TERMINATOR
7224  031574  001004                        BNE     5$              ;ERROR EXIT
7225  031576  005702              3$:       TST     R2              ;FINAL VALUE = 0
7226  031600  001402                        BEQ     5$              ;YES, TAKE ERROR EXIT
7227  031602  010210                        MOV     R2,(R0)         ;REPLACE THE ORIGINAL VALUE
7228  031604  005725              4$:       TST     (R5)+           ;ADJUST FOR NORMAL RETURN
7229  031606  012604              5$:       MOV     (SP)+,R4        ;RESTORE R4
7230  031610  012603                        MOV     (SP)+,R3        ;RESTORE R3
7231  031612  012602                        MOV     (SP)+,R2        ;RESTORE R2
7232  031614  000205                        RTS     R5              ;EXIT
7233
7234                          ;;**********************************************************************
7235
7236                          .SBTTL  ERROR MESSAGES
7237
7238                          ;;**********************************************************************
7239
7240                          .NLIST  BEX
      031616  044122  032067  030450  EM1:     .ASCIZ  /RH70(11) INTERRUPT OCCURRED (RMAS = 0)/
      031665     125  042516  050130  EM2:     .ASCIZ  /UNEXPECTED ATTENTION OCCURRED/
      031723     115  051501  041123  EM3:     .ASCIZ  /MASSBUS PARITY ERROR (MCPE=1)/
      031761     115  051501  041123  EM4:     .ASCIZ  /MASSBUS PARITY ERROR (PAR=1)/
      032016  042101  051104  051505  EM5:     .ASCIZ  /ADDRESS PLUG CHANGE BIT SET/
      032052  044122  030461  033450  EM6:     .ASCIZ  /RH11(70) DIDN'T RESPOND TO ADDRESSING/
      032120  047125  047503  051122  EM10:    .ASCIZ  /UNCORRECTABLE MASSBUS PARITY ERROR/
      032163     106  052101  046101  EM11:    .ASCIZ  /FATAL MASSBUS PARITY ERROR/
      032216  042520  051522  051511  EM12:    .ASCIZ  /PERSISTENT DEVICE UNSAFE/
      032247     117  042520  040522  EM13:    .ASCIZ  /OPERATION NOT COMPLETED WITHIN TIME LIMIT/
      032321     104  044522  042526  EM14:    .ASCIZ  /DRIVE WENT OFFLINE/
```

K11


```
032344  047516  051040  051505  EM15:   .ASCIZ  /NO RESPONSE TO PORT REQUEST/
032400  042510  042101  051105  EM20:   .ASCIZ  /HEADER CRC ERROR/
032421     104  052101  020101  EM21:   .ASCIZ  /DATA CHECK ('DCK') ERROR/
032452  051127  052111  020105  EM22:   .ASCIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') SET/
032525     127  044522  042524  EM23:   .ASCIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET/
032604  042510  042101  051105  EM24:   .ASCIZ  /HEADER READ ERROR - 'FMT' BIT DROPPED/
032652  042510  042101  051105  EM25:   .ASCIZ  /HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR/
032733     106  051117  040515  EM26:   .ASCIZ  /FORMAT ERROR ('FER')/
032760  042510  042101  051105  EM27:   .ASCIZ  /HEADER COMPARE ('HCE') ERROR/
033015     115  051511  042503  EM30:   .ASCIZ  /MISCELLANEOUS DRIVE ERROR/
033047     117  042520  040522  EM31:   .ASCIZ  /OPERATION INCOMPLETE ('OPI') ERROR/
033112  051104  053111  020105  EM32:   .ASCIZ  /DRIVE TIMING ('DTE') ERROR/
033145     120  051101  052111  EM33:   .ASCIZ  /PARITY ('PAR') ERROR AFTER OPERATION STARTED/
033222  051127  052111  020105  EM34:   .ASCIZ  /WRITE CLOCK FAILURE ('WCF') ERROR/
033264  047111  040526  044514  EM35:   .ASCIZ  /INVALID ADDRESS ('IAE') ERROR/
033322  051127  052111  020105  EM36:   .ASCIZ  /WRITE LOCK ('WLE') ERROR/
033353     104  052101  020101  EM37:   .ASCIZ  /DATA CHECK ('DCK') SET DURING WRITE CHECK COMMAND/
033435     122  030510  024061  EM40:   .ASCIZ  /RH11(70) OR UNIBUS TRANSFER ERROR/
033477     102  051525  040440  EM41:   .ASCIZ  /BUS ADDRESS OR WORD COUNT INCORRECT/
033543     104  052101  020101  EM42:   .ASCIZ  /DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED/
033624  040503  023516  020124  EM43:   .ASCIZ  /CAN'T MATCH DATA READ WITH A PATTERN/
033671     105  051122  051117  EM44:   .ASCIZ  /ERROR BIT(S) SET, BUT NO ERROR SIGNALED BY THE RH11/
033755     105  041503  046040  EM45:   .ASCIZ  /ECC LOGIC FAILURE - POSITION REGISTER VALUE NOT VALID/
034043     102  051525  040440  EM46:   .ASCIZ  /BUS ADDRESS AND WORD COUNT NOT CONSISTENT/
034115     123  042505  020113  EM50:   .ASCIZ  /SEEK INCOMPLETE ('SKI') ERROR/
034153     120  047522  051107  EM51:   .ASCIZ  /PROGRAM DETECTED POSITIONING ERROR/
034216  051104  053111  020105  EM60:   .ASCIZ  /DRIVE UNSAFE ERROR/
034241     040  000040          EM61:   .ASCIZ  /  /
                                .EVEN
```

```
034244 046522 051501 020040 DH1:    .ASCIZ  /RMAS  /
034253    104 044522 042526 DH2:    .ASCIZ  /DRIVE   RMDS    RMER1   RMER2   RMMR2    RMAS    /
034335    104 044522 042526 DH3:    .ASCIZ  /DRIVE   REG ADR   DATA  /
034366 051104 053111 020105 DH4:    .ASCIZ  /DRIVE   REG ADR   GOOD    BAD    /
034430 046522 042101 020122 DH6:    .ASCIZ  /RMADR   /
034441    104 044522 042526 DH7:    .ASCIZ  /DRIVE   RMCS1   RMWC    RMBA    RMDA    /
034512 046522 051503 020062 DH10:   .ASCIZ  /RMCS2   RMDS    RMER1   RMAS    RMDB    /
034563    122 046515 030522 DH11:   .ASCIZ  /RMMR1   RMDT    RMOF    RMDC    RMMR2   /
034634 046522 051105 020062 DH12:   .ASCIZ  /RMER2   RMEC1   RMEC2   /
       034666               .EVEN

                            .LIST   BEX

7241
7242 034666 001324 000000      DT1:    .WORD   ATTN,0
7243 034672 001222 022730 022732 DT2:  .WORD   DRIVE,RMERRS,RMERRS+2,RMERRS+4,RMERRS+6,ATTN,0
7244 034700 022734 022736 001324
7245 034706 000000
7246 034710 001222 027720 027722 DT3:  .WORD   DRIVE,RD.ADR,RD.WRD,0
7247 034716 000000
7248 034720 001222 030144 030142 DT4:  .WORD   DRIVE,WRT.AD,WRT.WD,RD.WRD,0
7249 034726 027722 000000
7250 034732 023076 000000      DT6:    .WORD   RMADR,0
7251 034736 001222 037166 037170 DT7:  .WORD   DRIVE,RM.REG,RM.REG+2,RM.REG+4,RM.REG+6,0
7252 034744 037172 037174 000000
7253 034752 037176 037200 037202 DT10: .WORD   RM.REG+10,RM.REG+12,RM.REG+14,RM.REG+16,RM.REG+22,0
7254 034760 037204 037210 000000
7255 034766 037212 037214 037220 DT11: .WORD   RM.REG+24,RM.REG+26,RM.REG+32,RM.REG+36,RM.REG+40,0
7256 034774 037224 037226 000000
7257 035002 037230 037232 037234 DT12: .WORD   RM.REG+42,RM.REG+44,RM.REG+46,0
7258 035010 000000
7259
7260                        ;
7261                        ;
7262 035012    000          DF1:    .BYTE   0
7263 035013    000    000   000  DF2:    .BYTE 0,0,0,0,0,0
7264 035016    000    000   000
7265 035021    000    000   000  DF3:    .BYTE   0,0,0
7266 035024    000    000   000  DF4:    .BYTE   0,0,0,0
7267 035027    000
7268                               .EVEN
7269
7270                        .NLIST  BEX

035030 052523 051502 051531 MESG1:  .ASCIZ  /SUBSYS  /
035041    104 044522 042526 MESG2:  .ASCIZ  /DRIVE(S) /
035053     52 020052 052123 MESG3:  .ASCIZ  /** STARTING PASS 1 **/
035101    115 052517 052116 MESG4:  .ASCIZ  /MOUNT PACK ON DRIVE  /
035127    101 042116 046040 MESG8:  .ASCIZ  /AND LOAD. /
035142 054524 042520 051040 MESG9:  .ASCIZ  /TYPE R<CR> WHEN DRIVE READY : /
035201    104 044522 042526 MESG10: .ASCIZ  /DRIVE IS NOT READY, TEST IS ABORTED/
035245    120 041501 020113 MESG11: .ASCIZ  /PACK IS NOT ACCEPTABLE,CHANGE PACK ,TRY AGAIN/
035323    125 046116 040517 MESG12: .ASCIZ  /UNLOAD DRIVE /
035341     40 040440 042116 MESG13: .ASCII  /  AND REMOVE PACK/<CR><LF>
035364 054524 042520 051040       .ASCIZ  /TYPE R<CR> WHEN DONE : /
035414 025052 051440 040524 MESG14: .ASCIZ  /** STARTING PASS 2 **/
035442 025040 020052 046101 MESG15: .ASCIZ  / ** ALL DRIVES ARE COMPATIBLE **/
```

```
035503     123  047503  042522  MESG16: .ASCIZ  /SCORES FOR DRIVE  /
035526  051124  041501  020113  MESG17: .ASCIZ  /TRACK      DRIVE      OVRWRT      OVRWRT      READ       READ/
035622  047516  020056  020040  MESG18: .ASCIZ  /NO.        READ       OFST-       OFST+       OFST-      OFST+/
035717     040  051440  046105  SELFX:  .ASCII  /  SELF/
035725     011     000          TAB:    .BYTE   11,0            ;THE SEQUENCE OF SELFX,TAB MUST BE NOT REVERSED.
035727     040  020052     060  MARKX:  .ASCII  / % 0/
035733     011     000          .BYTE   11,0
035735     054     000          COMMA:  .ASCIZ  /,/
035737     127  046111  020114  MESG5:  .ASCIZ  /WILL TEST /
035752  047117  000040          MESG6:  .ASCIZ  /ON /
035756  051511  052040  042510  MESG7:  .ASCIZ  /IS THERE ANOTHER SUB-SYSTEM (Y OR N<CR>) ?/
036031     116  052117  040440  NOTRM:  .ASCIZ  /NOT AN RM03/
036045     040  047516  020124  NOTSAF: .ASCIZ  / NOT SAFE/
036057     040  047516  020124  NOTPRS: .ASCIZ  / NOT PRESENT/
036074  047440  043106  044514  UNTOFF: .ASCIZ  / OFFLINE/
036105     040  047117  044514  UNTON:  .ASCIZ  / ONLINE/
036115     040  020040     040  LINMSP: .ASCII  /    /
036121     040     000          LINSP:  .ASCIZ  / /
036123     122  030115  000063  RM03A:  .ASCIZ  /RM03/
036130  005015  023440  023514  NEDCLK: .ASCIZ  <CR><LF>/ 'L' OR 'P' CLOCK REQUIRED ON SYSTEM/<CR><LF>
036201     072     000          COLON:  .ASCIZ  /:/
036203     105  052116  051105  ENTDRV: .ASCIZ  /ENTER I.D. FOR DRIVE #/
036232  020077  047111  040526  BADENT: .ASCIZ  /? INVALID ENTRY/
036252  047105  042524  020122  ENTADR: .ASCIZ  @ENTER BAD TRK/SEC ADDREES FOR DRV #@
036316  027440  000040          SLASH:  .ASCIZ  @ / @
036322  000077                  QUES:   .ASCIZ  /?/
036324  051104  053111  000105  UNTMSG: .ASCIZ  /DRIVE/
036332  000040                  LINSP0: .ASCIZ  / /
036334  051104  053111  020105  STATUS: .ASCIZ  /DRIVE STATUS/
036351     040  020040  041040  MESG19: .ASCIZ  /    BAD SPOT FILE  /
036375     040  025052  020040  MESG20: .ASCIZ  / **  END  OF  TEST  **/
036424  051104  053111  020105  MESG21: .ASCII  /DRIVE NOT ON-LINE OR NOT ASSIGNED/<CR><LF>
036467     120  047522  051107          .ASCIZ  /PROGRAM HALT/<CR><LF>


                                .EVEN

                                .LIST   BEX
7271
7272
7273                            ;HISTORY FILE FOR 15 LOGICAL DRIVES:(0-17)
7274            000001                  $FMT    =       1
7275            000002                  $COMND  =       $FMT+1          ;COMMAND CODE
7276            000003                  $PSEL   =       $FMT+2          ;PRO. SELECT AND A16,A17
7277            000004                  $WRDM   =       $FMT+3          ;WORD COUNT
7278            000006                  $BUF    =       $FMT+5          ;BUFFER ADDRESS
7279            000010                  $SEC    =       $FMT+7          ;SECTOR ADDRESS
7280            000011                  $TRK    =       $FMT+10         ;TRACK ADDRESS
7281            000012                  $CYL    =       $FMT+11         ;CYLINDER ADDRESS
7282            000014                  $SYSNM  =       $FMT+13         ;SUB SYSTEM A-H
7283            000015                  $PHYDR  =       $FMT+14         ;PHYSICAL DRIVE CODE (ASCII )
7284            000016                  $GAP    =       $FMT+15         ;LEFT TWO NULL BYTES
7285            000022                  $EMTAB  =       $GAP+4  ;END OF HISTORY TABLE
7286
7287  036506     000     000          DRIV0:  .BYTE 7&0,0     ;HISTORY BLOCK OF LOGICAL DRIVE 0
7288  036510  000020                          .BLKB   $EMTAB-$COMND
```

```
7289  036530  001     000     DRIV1:  .BYTE 7&1,0     ;HISTORY BLOCK OF LOGICAL DRIVE 1
7290  036532  000020          .BLKB   SEMTAB-$COMND
7291  036552  002     000     DRIV2:  .BYTE 7&2,0     ;HISTORY BLOCK OF LOGICAL DRIVE 2
7292  036554  000020          .BLKB   SEMTAB-$COMND
7293  036574  003     000     DRIV3:  .BYTE 7&3,0     ;HISTORY BLOCK OF LOGICAL DRIVE 3
7294  036576  000020          .BLKB   SEMTAB-$COMND
7295  036616  004     000     DRIV4:  .BYTE 7&4,0     ;HISTORY BLOCK OF LOGICAL DRIVE 4
7296  036620  000020          .BLKB   SEMTAB-$COMND
7297  036640  005     000     DRIV5:  .BYTE 7&5,0     ;HISTORY BLOCK OF LOGICAL DRIVE 5
7298  036642  000020          .BLKB   SEMTAB-$COMND
7299  036662  006     000     DRIV6:  .BYTE 7&6,0     ;HISTORY BLOCK OF LOGICAL DRIVE 6
7300  036664  000020          .BLKB   SEMTAB-$COMND
7301  036704  007     000     DRIV7:  .BYTE 7&7,0     ;HISTORY BLOCK OF LOGICAL DRIVE 7
7302  036706  000020          .BLKB   SEMTAB-$COMND
7303  036726  000     000     DRIV10: .BYTE 7&10,0    ;HISTORY BLOCK OF LOGICAL DRIVE 10
7304  036730  000020          .BLKB   SEMTAB-$COMND
7305  036750  001     000     DRIV11: .BYTE 7&11,0    ;HISTORY BLOCK OF LOGICAL DRIVE 11
7306  036752  000020          .BLKB   SEMTAB-$COMND
7307  036772  002     000     DRIV12: .BYTE 7&12,0    ;HISTORY BLOCK OF LOGICAL DRIVE 12
7308  036774  000020          .BLKB   SEMTAB-$COMND
7309  037014  003     000     DRIV13: .BYTE 7&13,0    ;HISTORY BLOCK OF LOGICAL DRIVE 13
7310  037016  000020          .BLKB   SEMTAB-$COMND
7311  037036  004     000     DRIV14: .BYTE 7&14,0    ;HISTORY BLOCK OF LOGICAL DRIVE 14
7312  037040  000020          .BLKB   SEMTAB-$COMND
7313  037060  005     000     DRIV15: .BYTE 7&15,0    ;HISTORY BLOCK OF LOGICAL DRIVE 15
7314  037062  000020          .BLKB   SEMTAB-$COMND
7315  037102  006     000     DRIV16: .BYTE 7&16,0    ;HISTORY BLOCK OF LOGICAL DRIVE 16
7316  037104  000020          .BLKB   SEMTAB-$COMND
7317  037124  007     000     DRIV17: .BYTE 7&17,0    ;HISTORY BLOCK OF LOGICAL DRIVE 17
7318  037126  000020          .BLKB   SEMTAB-$COMND
7319
7320
7321
7322  037146  000             FMTDPB: .BYTE   0       ;DRIVER PARAMETER BLOCK, DRIVE #
7323  037147  000                     .BYTE   0       ;OFFSET FMT16,HCI,ECI
7324  037150  000                     .BYTE   0       ;COMMAND CODE
7325  037151  000                     .BYTE   0       ;PSEL, A16 AND A17
7326  037152  000000                  .WORD   0       ;WORD COUNT (NEG)
7327  037154  037262                  .WORD   #ENDPGM ;BUFFER ADDRESS
7328  037156  000                     .BYTE   0       ;SECTOR ADDRESS
7329  037157  000                     .BYTE   0       ;TRACK ADDRESS
7330  037160  000000                  .WORD   0       ;CYLINDER ADDRESS
7331  037162  037166                  .WORD   RM.REG  ;ADDRESS TO SAVE ALL RH11/RM03 REG'S
7332  037164  000000                  .WORD   0       ;STATUS WORD
7333
7334  037166  000000          RM.REG: .WORD   0       ;RMCS1
7335  037170  000000                  .WORD   0       ;RMWC
7336  037172  000000                  .WORD   0       ;RMBA
7337  037174  000000                  .WORD   0       ;RMDA
7338  037176  000000                  .WORD   0       ;RMCS2
7339  037200  000000                  .WORD   0       ;RMDS
7340  037202  000000                  .WORD   0       ;RMER1
7341  037204  000000                  .WORD   0       ;RMAS
7342  037206  000000                  .WORD   0       ;RMLA
7343  037210  000000                  .WORD   0       ;RMDB
7344  037212  000000                  .WORD   0       ;RMMR1
```

```
                                                .WORD   0       ;RMDT
                                                .WORD   0       ;RMSN
                                                .WORD   0       ;RMOF
                                                .WORD   0       ;RMCA
                                                .WORD   0       ;RMDC
                                                .WORD   0       ;RMER2
                                                .WORD   0       ;RMMR2
                                                .WORD   0       ;RMEC1
                                                .WORD   0       ;RMEC2

                                        ;INDEX OF STATUS AND REGISTER WORDS RELATIVE TO FMTDPB

                                        $STATUS =       16
                                        $RMCS1  =       20
                                        $RMWC   =       $RMCS1+2
                                        $RMBA   =       $RMWC+2
                                        $RMDA   =       $RMBA+2
                                        $RMCS2  =       $RMDA+2
                                        $RMDS   =       $RMCS2+2
                                        $RMER1  =       $RMDS+2
                                        $RMAS   =       $RMER1+2
                                        $RMLA   =       $RMAS+2
                                        $RMDB   =       $RMLA+2
                                        $RMMR1  =       $RMDB+2
                                        $RMDT   =       $RMMR1+2
                                        $RMSN   =       $RMDT+2
                                        $RMOF   =       $RMSN+2
                                        $RMDC   =       $RMOF+2
                                        $RMHR   =       $RMDC+2
                                        $RMMR2  =       $RMHR+2
                                        $RMER2  =       $RMMR2+2
                                        $RMEC1  =       $RMER2+2
                                        $RMEC2  =       $RMEC1+2

037236  000000  000000  177776  GENDPB: .WORD   0,0,-2,CYLDER
037244  037256
037246  000000  000000  037166          .WORD   0,0,RM.REG,0
037254  000000
037256  000002                  CYLDER: .BLKW   2
        037262                  ENDPGM  =       .       ;LAST LOCATION OF PROGRAM+2
                                .NLIST  BEX

037262  005015  040515  047111  TITLE:  .ASCII  <CR><LF>/MAINDEC-11-DZRMI-A/<CR><LF>
037310  046522  031460  042040          .ASCIZ  @RM03 DRIVE COMPATIBILITY TEST@<CR><LF><LF>
037351     015  052012  020117  LOADRV: .ASCII  <CR><LF>/TO TEST DRIVE 0, REPLACE THE 'XXDP' PACK ON DRIVE 0/<CR><LF>
037440  044527  044124  040440          .ASCII  /WITH ANOTHER PACK, CLEAR MEMORY LOCATION 40,/<CR><LF>
037516  047101  020104  042522          .ASCIZ  /AND RESTART THE PROGRAM/<CR><LF>


                                .LIST   BEX

        000001                          .END
```

```
ABASE = 176700           1660*   1792    1833
ACDW1 = 000000           1792    1835
ACDW2 = 000000           1792    1836
ACK   = 000123           1679*
ACPUOP= 000000           1792    1807
ACTDRV  023014           5814*   6082*   6133*   6451*   6460*   6693
ACTSTR  023015           5820*   6700*   6714*
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
APFIOR= 000000           1792
APTCOD= 000040           5035    5141*
APTENV= 000001           4933    5028    5097    5139*
APTSIZ= 000200           2379    5138*
APTSPD= 000100           5030    5099    5140*
ASNLST  001776           1989*   2402    2467    2539    2541    2553*   2577    2583    3746
ASSGN1  002000           1991*   2554*   2615*   2634    3045*   3058
ASSGN2  002002           1993*   2555*   3074    3837*
ASWREG= 000000           1792    1805
ATNBIT  023064           2488    3343    5886*   6049    6145    6251    6621    6640    6648    6649    6669 - 6755
ATESTN= 000000           1792    1796
ATTN    001324           1854*   7242    7243
```

```
AUNIT = 000000          1792    1799
AUSWR = 000000          1792    1806
AVECT1= 000254          1661#   1792    1831
AVECT2= 000000          1792    1832
BADENT  036232          7270#
BADSEC  001340          1861#   3740*   3809    3876    3891*
BIT0  = 000001          1645#   3352    3745    3802    3830
BIT00 = 000001          1635#   1645
BIT01 = 000002          1634#   1644    3970    6125    6373
BIT02 = 000004          1633#   1643
BIT03 = 000010          1632#   1642    4013    6580    6873
BIT04 = 000020          1631#   1641    4013    4025    6615
BIT05 = 000040          1630#   1640    5976    6393    6595    6750
BIT06 = 000100          1629#   1639    4406    4737    6058    6116    6190    6491    6934
BIT07 = 000200          1628#   1638    3956    4028    6058    6325    6472    6580    6595    6615    6644    6906
BIT08 = 000400          1627#   1637    4022    5624    6058
BIT09 = 001000          1626#   1636    3968    4944    5632    6748
BIT1  = 000002          1644#   2767    2784
BIT10 = 002000          1625#   3966    4040    4048    4921    6375
BIT11 = 004000          1624#   3964    4043    5639    6023    6152    6344    6684
BIT12 = 010000          1623#   3339    3341    3962    4051    4888    6018    6040    6058    6127    6160    6340    6359
                        6371    6384    6605    6607    6811    6868    6935
BIT13 = 020000          1622#   4031    4057    4564    4928    6110    6553    6814
BIT14 = 040000          1621#   3350    3946    3948    3970    4016    4019    4057    5610    6122    6157    6540    6680
                        6776    6823    6827    6932
BIT15 = 100000          1620#   3944    4059    4074    6110    6122    6125    6127    6157    6160    6344    6373    6375
                        6491    6580    6595    6605    6615    6680    6748    6776    6783
BIT2  = 000004          1643#   3972    6783
BIT3  = 000010          1642#   4034
BIT4  = 000020          1641#
BIT5  = 000040          1640#   4037
BIT6  = 000100          1639#
BIT7  = 000200          1638#   2723    2730    3339
BIT8  = 000400          1637#
BIT9  = 001000          1636#   4046
BLKADR  002600          2163#   2436    2551    2585    2638    2828    3049    3078    3842    3908
BPTVEC= 000014          1652#
BUSADR  031314          2512    7152#
CFLAG   001334          1859#   2412*   4396*   4399    4750*   4807*
CHGADR  001332          1858#   2319*   2321*   2323*   2496    2575    2612
CI1     024350          6180    6208#
CI3     024456          6182    6231#
CI4     024556          6171    6253#
CI5     025134          6230    6252    6330#   6339
CI6     025156          6269    6279    6281    6283    6335#
CI7     025172          6112    6189    6220    6224    6228    6236    6246    6250    6261    6268    6274    6278    6292
                        6298    6302    6312    6323    6338    6340#   6446    6476    6632    6918
CI78    025220          6343    6346#   6634
CI8     025272          6341    6349    6358#   6506    6768
CKBUS   014344          3951    4191#
CKCLK   014740          2428    4312#
CKCLK1  015034          4314    4325#
CKCLK2  015102          4326    4334#
CKCLK3  015126          4324    4333    4340#
CKERR   014276          3950    4177#
CKFMT   014104          4027    4095#
```

```
CKHCE   014114          4030    4101#
CKSWR = 104407          4917    4943    5609    5698#
CK.CHR  016172          4619#   4656    4670
CK.DEC  016144          4597#   4625
CK.DIG  016244          4650#
CK.NUM  031506          7160    7169    7201#
CK.OCT  016116          2486    4579#   4627
CLKFLG  001314          1849#   4312*   4317*   4328*   4346
CLOCK   015234          4320    4330    4367#
CLRDPB  015426          2437    4416#
CLRQUE  030730          5958    6392    7002#
CMCNT   001352          1866#   2975*   3004*   3007*   3010
CMCYL   001354          1867#   2976*   2980    3003*   3006*   3008
CMSEC   001360          1869#   2977*   3000*   3253    3255*   3284    3286*   3423    3455    3497*   3503*   3504*   3643
                        3645*   3665    3667*
CMTRK   001362          1870#   3741*   3776    3782    3794*   3797    3807    3897
COLON   036201          2481    4353    4358    7270#
COMMA   035735          7270#
CR    = 000015          1560#   1839#   4833    5074    5084    7270    7387
CRLF  = 000200          1561#   5045    5084
CYLDER  037256          7381    7385#
CYLIMT  001372          1874#   2687
DCKER   014024          4056    4067#
DDISP = 177570          1567#   1769    2367
DEASG   012704          3829#
DF1     035012          2246    2281    7262#
DF2     035013          2253    2274    7263#
DF3     035021          2260    7265#
DF4     035024          2267    7266#
DH1     034244          2244    7240#
DH10    034512          2293    7240#
DH11    034563          2300    7240#
DH12    034634          2307    7240#
DH2     034253          2251    2272    7240#
DH3     034335          2258    7240#
DH4     034366          2265    7240#
DH6     034430          2279    7240#
DH7     034441          2286    7240#
DISMNT  012624          3810    3813#
DISPLA  001156          1769#   2367*   2375*   4920*   5654*
DISPLY= 104414          3979    3985    3991    3997    4003    4008    4068    4076    4078    4084    4090    4096    4102
                        4106    4110    4115    4120    4125    4131    4136    4140    4146    4152    4157    4163    4185
                        4199    4226    4231    4233    4269    4272    4285    4543    5703#
DISPRE  000174          1698#   2375
DONE    013516          3958    4013#
DPINT   022770          5774#   5987    5990    6066*   6092    6524    6549    6670    6672*   6739    6757    6762    6770*
DPRQS   023000          5787#   6098    6146*   6184*   6527    6741    6759    6764    6779*
DRIVE = 001222          1853#   2475*   2488*   2507    2536    2556    2570*   2652*   2675    2683    2843*   2847    2871
                        2896    2956    3031    3125*   3143    3194    3328    3390    3392    3533    3629    3818    3882
                        7243    7246    7248    7251
DRIV0   036506          2163    7287#
DRIV1   036530          2164    7289#
DRIV10  036726          2171    7303#
DRIV11  036750          2172    7305#
DRIV12  036772          2173    7307#
DRIV13  037014          2174    7309#
```

F12


```
DRIV14  037036          2175    7311#
DRIV15  037060          2176    7313#
DRIV16  037102          2177    7315#
DRIV17  037124          2178    7317#
DRIV2   036552          2165    7291#
DRIV3   036574          2166    7293#
DRIV4   036616          2167    7295#
DRIV5   036640          2168    7297#
DRIV6   036662          2169    7299#
DRIV7   036704          2170    7301#
DRVACT  022740          3394*   5733#   6106    6286*   6333*   6352*   6363    6378*   6465*   6545    6578    6609*   6616*
                        6629    6639*   6751*
DRVCLR= 000111          1674#
DRVER   014074          4061    4089#
DRVINT  023326          5978    6012#   6094    6660    6673
DRVQUE  031026          6101    6114    7042#
DRVSTA  022750          2676    2872    3144    5747#   5968*   5969*   5970*   5971*   5981*   6013*   6022*   6055*   6060*
                        6088    6096    6120    6154    6158    6287*   6396*   6530    6538    6551    6603*   6610*   6622
                        6675    6771*
DRVTYP  022760          2678    2874    3146    5761#   6014*   6029*   6034*   6123    6397*
DSWR    = 177570        1566#   1768    2366
DTEER   014144          4053    4114#
DTUW    023062          3390*   5880#   5964    6173    6229*   6353    6356*   6366    6379    6381*   6390*   6454    6466*
                        6744    6821
DT1     034666          2245    7242#
DT10    034752          2294    7253#
DT11    034766          2301    7255#
DT12    035002          2308    7257#
DT2     034672          2252    2273    7243#
DT3     034710          2259    7246#
DT4     034720          2266    7248#
DT6     034732          2280    7250#
DT7     034736          2287    7251#
DULP    023330          6013#   6024
DUMP    014264          3980    3986    3992    3998    4004    4009    4069    4077    4079    4085    4091    4097    4103
                        4107    4111    4116    4121    4126    4132    4137    4141    4147    4153    4158    4164    4167#
                        4186    4200
EMPTYQ  031006          6350    6395    6492    6617    6682    6784    7027#
EMTVEC= 000030          1655#   2336*   2337*   2383*
EM1     031616          2243    7240#
EM10    032120          3985    7240#
EM11    032163          3991    7240#
EM12    032216          3979    7240#
EM13    032247          3997    7240#
EM14    032321          4003    7240#
EM15    032344          4008    7240#
EM2     031665          2250    7240#
EM20    032400          4084    7240#
EM21    032421          4068    7240#
EM22    032452          4078    7240#
EM23    032525          4076    7240#
EM24    032604          4096    7240#
EM25    032652          4102    7240#
EM26    032733          4136    7240#
EM27    032760          4140    7240#
EM3     031723          2257    7240#
```

```
EM30    033015          4090    7240#
EM31    033047          4110    7240#
EM32    033112          4115    7240#
EM33    033145          4120    7240#
EM34    033222          4157    7240#
EM35    033264          4125    7240#
EM36    033322          4131    7240#
EM37    033353          7240#
EM4     031761          2264    7240#
EM40    033435          4146    7240#
EM41    033477          4199    7240#
EM42    033543          7240#
EM43    033624          7240#
EM44    033671          4185    7240#
EM45    033755          7240#
EM46    034043          7240#
EM5     032016          2271    7240#
EM50    034115          4152    7240#
EM51    034153          4106    7240#
EM6     032052          2278    7240#
EM60    034216          4163    7240#
EM61    034241          7240#
ENDPGM= 037262          2684    2713    2716    2900    2958    3199    3241    3271    3537    3627    7327    7386#
ENDX1   005432          2636    2703    2794#
ENTADR  036252          7270#
ENTDRV  036203          7270#
ERPRC1  013354          3957    3962#
ERPROC  013340          3943    3947    3949    3956#
ERRVEC= 000004          1648#   2364    2365*   2376*   4314*   4315*   4326*   4340*   5615    5616*   5618*   5621*   7120
                        7121    7125*   7126*   7133*   7134*   7173    7174*   7176*   7183*
ES.SAV  031172          6509    6560    6564    6819    6879    7097#
FALPAR  013456          3967    3990#
FAULT   001374          1875#   3085*   3892*   4839
FILBUF  014546          2904    3539    4238#
FINISH  017154          4839#   4899
FMTDPB  037146          2682    2693    2695    2698*   2699    2725    2732    2894    2906    2908    2910    2914    2916
                        2918    2955    2978    2984    2986    2988    3192    3214    3216    3218    3236    3244    3246
                        3248    3268    3276    3279    3389*   3391    3438    3491    3529    3541    3543    3545    3565
                        3567    3569    3577    3579    3623    3634    3636    3638    3652    3658    3660    7322#
FMTER   014204          4135#
GENDPB  037236          7381#
GETREG= 000141          1681#
GETREQ  031102          6147    6368    6483    6534    6570    6631    6679    6747    6773    6780    7063#
GMS   = ****** U        1697    5690    5691    5692    5693    5694    5696    5698    5699    5700    5701    5702    5703
GTSWR = 104406          2396    5696#
HCEER   014214          4140#
HCRCER  014064          4024    4083#
HOUR    001342          1862#   2409*   4349    4378*   4379    4381*
HT    = 000011          1558#   2569    5043    5084
HZ      001316          1850#   2408    4369
IAEER   014164          4042    4124#
IDLEX   013256          2640    2645    2834    3026    3105    3814    3936#
INDST   002560          2150#   2404    2468    3448
IOTVEC= 000020          1653#   2334*   2335*
ISR     025702          5973    6451#
KIPARO= ****** U        7116
```

```
KPATH   012336           3751#
KSR     015342           4394#
LA      025526           6177    6412#
LABAD   010732           3250    3281    3488#   3640    3662
LACNT   023026           5839#   6420*   6421    6442*
LF    = 000012           1559#   1840#   4833    5078    5084    7270    7387
LINDEC  014720           4297#
LINE1   014622           4264#
LINKDV  015472           4213    4433#
LINOCT  014666           4223    4228    4281#
LINSP   036121           2479    2597    2657    2661    2842    2851    3035    3130    3911    3912    4226    7157    7166
                         7270#
LINSP0  036332           4272    7270#
LINTSP  036115           7270#
LOADRV  037351           7387#
LOG0    001376           1892#   2947
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   007576           3236#   3305
LOOP2   007614           3240#   3296    3301
LOOP3   011540           3623#   3688
LOOP4   012316           3746#   3793    3808
LOOP5   012530           3747    3764    3767    3785    3788#
LOOP6   012546           3794#
LSTA0   001330           1857#   2419*   2420    2422*   2423*
MAKEUP  010400           3275    3382#   3576    3657
MARKX   035727           3925    7270#
MCPEMX  023074           5898#   6801    6849
MESG1   035030           2477    2599    2840    7270#
MESG10  035201           2702    2877    3150    7270#
MESG11  035245           2728    7270#
MESG12  035323           3029    3816    7270#
MESG13  035341           3036    3822    7270#
MESG14  035414           3121    7270#
MESG15  035442           4843    7270#
MESG16  035503           3880    7270#
MESG17  035526           3887    7270#
MESG18  035622           3889    7270#
MESG19  036351           4205    7270#
MESG2   035041           2480    2580    7270#
MESG20  036375           4846    7270#
MESG21  036424           2795    7270#
MESG3   035053           2648    2837    7270#
```

```
MESG4   035101          2650    2845    3123    7270#
MESG5   035737          2579    7270#
MESG6   035752          2598    2839    7270#
MESG7   035756          2606    7270#
MESG8   035127          2658    2852    3131    7270#
MESG9   035142          2660    2854    3133    7270#
MINUTE  001344          1863#   2410#   4354    4374#   4375    4377#
MNDLTA  023110          5914#   6440
MODOO   003734          2467#
MOD11   004176          2515    2521    2536#
MOD12   004360          2538    2572    2575#
MOD21   003760          2475#   2487    2494    2611
MOD22   004160          2497    2518#
MOD23   004114          2505    2507#
MOD30   004210          2537    2539#   2571
MONTR   003704          2433#
MRHVEC  031475          7163    7190#
MRMCS1  031464          7154    7188#
MXDLTA  023106          5911#   6438
MXLACT  023104          5908#   6421
MXWNDW  023112          5917#
M.DPID  015540          4440    4454#
M.DP40  015576          4465#   4486
M.DP41  015632          4470    4477#
M.DP42  015642          4476    4481#
M.DP44  015674          4488    4493#
M.DPSO  015706          4463    4497#
NEDCLK  036130          4335    7270#
NOTPRS  036057          7270#
NOTRM   036031          7270#
NOTSAF  036045          7270#
NULINE  001364          1871#   3748#   3893    3916#   3919    3923#   3926#
OFFCOO  002640          2182#   3235#   3302    3304#   3337    3427    3433    3563#   3584    3587#   3593#   3600#   3622#
                        3685    3687#
OFFSET= 000115          1676#
OFFST   010176          3273    3323#   3574    3655
OFLIN   013476          3971    4002#
OPIER   014134          4033    4109#
OPT     024112          6108    6143#   6486    6653    6690
OVWN0   002046          2004#   3187    3437    3744    3795
OVWN1   002066          2010#
OVWN2   002106          2016#
OVWN3   002126          2022#
OVWN4   002146          2028#
OVWP0   002166          2038#   3435
OVWP1   002206          2045#
OVWP2   002226          2052#
OVWP3   002246          2059#
OVWP4   002266          2066#
PACK    001322          1852#   2953#   2979
PWRER   014154          4036    4119#
PCLOCK  001312          1848#   4313#   4318#
PIRQ  = 177772          1565#
PIRQVE= 000240          1659#
POPQUE  031124          6156    6208    6324    6643    7078#
POSER   014124          4106#
```

```
PRINT   012766          3753    3757    3761    3765    3770    3774    3780    3786    3871#
PROCES  013264          2911    2919    2989    3219    3249    3280    3546    3570    3580    3639    3661    3941#
PRTBAD  014406          2726    2733    4204#
PRTIM   013506          3973    4007#
PR0   = 000000          1582#
PR1   = 000040          1583#
PR2   = 000100          1584#
PR3   = 000140          1585#
PR4   = 000200          1586#   4395
PR5   = 000240          1587#   4735
PR6   = 000300          1588#
PR7   = 000340          1589#
PS    = 177776          1562#   1563    2407*   4395*   4407*   4551    4552*   4759    5956    5957*   5986*   5994*   6080
                        6081*   6134*   6144    6193*   6427*
PSEUDO  002702          2204#   2963    3203
PSW   = 177776          1563#
PUNSAF  013436          3963    3978#
PWRVEC= 000024          1654#   2340*   2341*   5147*   5148*   5157*   5163*   5175*   5176*
QCNT    030436          6944#   7003    7027*   7042    7044*   7064    7078*
QDRV0   030530          6955    6966    6975    6987#
QDRV1   030550          6956    6967    6976    6988#
QDRV2   030570          6957    6968    6977    6989#
QDRV3   030610          6958    6969    6978    6990#
QDRV4   030630          6959    6970    6979    6991#
QDRV5   030650          6960    6971    6980    6992#
QDRV6   030670          6961    6972    6981    6993#
QDRV7   030710          6962    6973    6982    6994#
QINPT   030446          6955#   7029    7046*   7047*   7048    7050*
QOUTPT  030466          6966#   7029*   7067    7080    7081*   7082*   7083    7085*
QSTART  030506          6975#   7009    7014    7050    7085
QSTOP   030510          6976#   7048    7083
QTERM = 030730          6983    6995#
QUES    036322          7270#
RDCHR = 104410          4744    4781    5699#
RDDAT = 00017           1688#   2690    3388    3626
RDHD  = 000173          1689#
RDLIN = 104411          2482    2607    2666    2855    3052    3134    3823    5700#   7158    7167
RDN0    002306          2075#   3188    3431    3618    3719
RDN1    002326          2081#
RDN2    002346          2087#
RDN3    002366          2093#
RDN4    002406          2099#
RDP0    002426          2108#   3429    3717
RDP1    002446          2114#
RDP2    002466          2120#
RDP3    002506          2126#
RDP4    002526          2132#   3620
RD.ADR  027720          6803*   6804*   6806#   7246
RD.RM   027674          6025    6044    6050    6272    6296    6310    6414    6428    6474    6504    6572    6582    6597
                        6734    6801#   6856    6870    6876    6910
RD.RM1  027716          6805#   6835
RD.RM2  030042          6801*   6834#
RD.RM3  030046          6812    6826    6836#
RD.RM4  030052          6817    6838#
RD.WRD  027722          6807#   6908    7246    7248
READIN= 000121          1678#
```

```
RECAL = 000107          1673#
RELSE = 000113          1675#
REPLZ   015716          4351    4356    4361    4508#
RESREG= 104413          4252    4403    4443    5530    5580    5702#   5995    6128    6131    6194    6399    6459    6713
                        6919    7018    7185
RESVEC= 000010          1649#   2351*   2358*
RMADR   023076          2400*   2642*   2866*   3080*   3116*   3327    3393    5903#   5975    6087    6211    6231    6253
                        6412    6453    6732    6803    6809    6813    6829    6862    6866    6899    6929    7172    7250
RMAS  = 000016          3343    5928#   6049*   6502    6621*   6648*   6669*   6754
RMBA  = 000004          3336*   5923#
RMCS1 = 000000          3332*   3342*   3398*   5921#   6017*   6023    6038    6151*   6152    6227    6249    6291    6311
                        6322    6337    6348    6475    6505    6557    6589    6626    6684    6857
RMCS2 = 000010          3331*   5925#   5976*   6016*   6018    6100*   6150*   6212*   6232*   6254*   6340    6359    6371
                        6384    6393*   6413*   6473*   6571*   6655*   6668*   6733*   6750*   6810    6867    6900*   6906
                        6930*   6935
RMDA  = 000006          3334*   5924#   6219    6245    6260
RMDB  = 000022          5930#   6904
RMDC  = 000034          3333*   5931#   6223    6235    6267
RMDS  = 000012          3350    3352    5926#   6045    6415    6573    6598    6656    6735
RMDT  = 000026          5932#   6026
RMEC1 = 000044          5939#
RMEC2 = 000046          5940#   6914
RMERRS  022730          5725#   5959    6540    6543    6553    6656*   6657*   6658*   6659*   7243
RMER1 = 000014          5927#   6051    6583    6657    6871
RMER2 = 000042          5938#   6658
RMMR  = 000036          5936#
RMINIT  023114          2672    2868    3140    5955#
RMLA  = 000020          5929#   6429    6431
RMMR1 = 000024          5931#
RMMR2 = 000040          5937#   6659
RMOF  = 000032          3339*   3341*   5934#   6042    6273    6277    6297    6301
RMSN  = 000030          5933#
RMVEC   023100          2401*   2640*   2641*   2643*   2645*   2646*   2834*   2835*   2867*   3026*   3027*   3081*   3105*
                        3106*   3117*   3814*   3815*   5904#   5972    5974    6081
RMWC  = 000002          3335*   5922#
RM.REG  037166          2685    2901    2959    3200    3242    3272    3538    3628    7251    7253    7255    7257    7331
                        7334#   7383
RM03    023620          2692    2905    2913    2983    3213    3243    3540    3564    3633    6080#
RM03A   036123          7270#
RNOP  = 000101          1671#
RPTMR   027266          4383    6698#
RTC   = 000117          1677#
SAVEFG  023036          2429*   2673*   2869*   3141*   5854#   6326    6479    6645
SAVREG= 104412          4238    4394    4433    5504    5565    5701#   5955    6083    6143    6358    6452    6701    6896
                        7002    7152
SC      026116          6458    6487    6490    6496    6502#
SCORE   010512          3256    3287    3418#   3646    3668
SC11    026750          6542    6579    6636#
SC12    027060          6536    6547    6655#
SC13    027130          6526    6529    6665#
SC3     026166          6517#   6521
SC4     026172          6519#   6550    6562    6566    6614    6628    6635    6654    6691
SC5     026204          6518    6524#
SC6     026404          6546    6567#
SC6A    026514          6544    6591#
SC7     026642          6581    6594    6596    6606    6616#
```

L12


```
$C8      026720          6558    6574    6584    6590    6599    6627    6629#   6664
SEARCH=  000131          1680#
SECLMT   001366          1872#
SECOND   001346          1864#   2411*   4359    4370*   4371    4373*
SEEK  =  000105          1672#
SEEKFG   023040          2674*   2870*   3142*   5862#   5960    6175
SELDRV=  000145          1683#
SELFX    035717          3904    7270#
SELF0    002546          2141#   3525    3527    3558    3596    3716    3718    3778    3784
SELF1    002550          2142#
SELF2    002552          2143#
SELF3    002554          2144#
SELF4    002556          2145#
SETFMT=  000143          1682#
SETVEC   003672          2428#
SET.IE   030364          5989    6020    6118    6192    6398    6511    6928#
SIXTEE   001350          1865#   2408*   4367*   4369*
SIZMEM   003626          2416#
SKIER    014234          4062    4151#
SLASH    036316          7270#
SPATH    012426          3750    3768#
SPOTX    005264          2718    2749#
SRCHWT   023012          5808#   6145*   6251*   6640    6649*
STACK =  001100          1553#   2332    2406    2827    2893    2946    3025    3104    3186    3524    3617    3711
STARSC   001356          1868#   3253    3284    3423    3447    3493*   3643    3665
START    003062          1701    2319#   4339
START1   003072          1702    2321#
START2   003102          1706    2323#
START3   003106          2320    2322    2325#
STATIN   001320          1851#
STATUS   036334          7270#
STKLMT=  177774          1564#
STNDAT   002642          2188#   4242    4250
ST0      027356          6708    6728#
ST01     027412          6739#
ST02     027502          6738    6740    6742    6754#
ST03     027532          6756    6762#
ST05     027556          6736    6768#
ST06     027564          6758    6770#
ST07     027622          6760    6778#
ST08     027652          6777    6784#
ST09     027662          6746    6753    6761    6765    6767    6769    6775    6782    6786#
SVRH70   030246          6328    6345    6376    6481    6493    6535    6586    6604    6647    6681    6749    6785    6896#
SWR      001154          1768#   2330    2366*   2368    2374*   2381*   2394    2723    2730    4264    4267    4564    4753
                         4888    4921    4928    4940    4944    5155    5168*   5345    5382*   5610    5624    5626    5632
                         5639
SWREG    000176          1699#   2374    2394    4753    5345    5358
SWTIM    013466          3963    3996#
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#   4264
SW11  = 004000          1596#
SW12  = 010000          1595#
SW13  = 020000          1594#   4267
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#   2503    2519*   2520*   2642    2643    2866    2867    3080    3081    3116    3117
TAB     035725          3901    3905    3915    3922    7270#
TABLEX  002044          1997#   3444*   3463
TAB.XY= 001114          1740#   1747
TBITVE= 000014          1650#   2348*   2349*
TD      025746          6456    6465#
TIMER   023042          3396*   5867#   6064*   6187*   6331*   6377*   6468*   6568*   6612*   6651*   6666*   6687*   6704
                        6706*   6766*   6772*   6778*
TITLE   037262          2387    7387#
TKVEC = 000060          1657#   4734*   4735*
TPVEC = 000064          1658#
TRAPVE= 000034          1656#   2338*   2339*   2384*
TRFER   014224          4021    4145#
TRKLMT  001370          1873#   2688
TRNSWT  023010          3391*   5800#   6209*   6355*   6365    6382*   6391*   6470    6471*   6743
TRTVEC= 000014          1651#
TSTNM   001336          1860#   2826*   2892*   2945*   3024*   3103*   3185*   3386    3425    3523*   3616*   3710*
TST1    005450          2727    2824#   2878    3061
TST10   011476          3614#
TST11   012150          3708#
TST2    005762          2876    2890#
TST3    006200          2943#
TST4    006572          3022#
TST5    007056          3086    3101#   3151    3843
TST6    007340          3148    3183#
TST7    011030          3306    3521#
TYPDS = 104405          5694#
TYPE  = 104401          2387    2476    2477    2478    2479    2480    2481    2579    2580    2597    2598    2599    2600
                        2605    2606    2647    2648    2649    2650    2651    2657    2658    2659    2660    2661    2702
                        2728    2729    2794    2795    2836    2837    2838    2839    2840    2841    2842    2844    2845
                        2846    2851    2852    2853    2854    2877    3028    3029    3030    3035    3036    3083    3120
                        3121    3122    3123    3124    3130    3131    3132    3133    3149    3150    3811    3813    3816
                        3817    3822    3879    3880    3881    3886    3887    3888    3889    3890    3895    3896    3901
                        3904    3905    3911    3912    3913    3915    3922    3925    4204    4205    4206    4266    4335
                        4348    4353    4358    4523    4748    4749    4757    4758    4765    4788    4794    4799    4803
                        4808    4813    4814    4816    4819    4823    4841    4842    4843    4844    4845    4846    4847
                        4923    4931    4965    4982    4984    4987    4989    4993    5000    5048    5177    5249    5325
                        5356    5357    5360    5373    5384    5403    5690#   7154    7157    7163    7166
```

N12


```
TYPOC = 104402          4973    4997    5359    5691#   7156    7165
TYPON = 104404          5693#
TYPOS = 104403          2589    2654    2848    3032    3127    3819    3883    3898    5692#
TYPRI4  016046          4551#
UCPAR   013446          3965    3984#
ULDFLG  023016          5827#   6015*   6090    6105*   6288*   6351*   6383*   6532    6618    6620*   6636    6638*   6752*
UNIT    001326          1855#   3941*
UNSAF   014254          4015    4162#
UNTMSG  036324          7270#
UNTOFF  036074          7270#
UNTON   036105          7270#
WCFER   014244          4039    4156#
WCKD  = 000151          1684#   2912    3240    3385    3557
WCKER   014034          4018    4073#
WCKHD = 000153          1685#
WLEER   014174          4045    4130#
WRTDAT= 000161          1E 6#   2903    2920    2957    3197    3531
WRTHD = 000163          1687#
WRT.A0  030144          6852*   6862*   6865#   7248
WRT.RM  030054          6037    6041    6218    6222    6226    6234    6244    6248    6259    6266    6276    6290    6300
                        6321    6336    6347    6556    6588    6625    6849#
WRT.R1  030140          6863#   6884
WRT.R2  030232          6849*   6883#
WRT.R3  030236          6858    6869    6872    6878    6885#
WRT.R4  030242          6874    6887#
WRT.R5  030244          6886    6888#
WRT.W0  030142          6850*   6854    6861*   6864#   7248
XEND2   012762          3076    3838    3846#
XPASS1  004564          2614    2634#   2734
XPASS2  006772          2616    3060    3074#
$APTHD  001100          1727    1733#
$ASTAT= ****** U        5119    5134
$ATYC   020230          5090    5092#
$ATY1   020204          5088#
$ATY3   020212          5033    5089#
$ATY4   020222          4936    5091#
$AUTOB  001150          1765#   2398*   5353    5443
$BASE   001264          1833#
$BADR   001136          1760#
$BDAT   001142          1762#
$BELL   001200          1780#   4266    4923    4956
$BUF  = 000006          2684#   2900*   2958*   2964    3199*   3205    3241*   3271*   3336    3537*   3627*   4196    4239
                        7278#
$CDW1   001270          1635#   2504*   2509    2638*   2644    2828    2830    2895    3049*   3078*   3082    3107    3193
                        3842*
$CDW2   001272          1836#   2472*   2478    2498    2552    2586    2600    2602*   2603    2639*   2651    2831    2833*
                        2841    2846    2861    3030    3079*   3108    3110*   3111    3124    3817    3881
$CHARC  020200          5050*   5060*   5067    5076*   5081#
$CKSWR  021314          5345#   5698
$CLR.T  017302          4874    4877#
$CMTAG  001114          1748#   2327    2328    2336    2342    2343    2344
$CM3  = 000000          1778#
$CNTLC  017146          4749    4808    4833#
$CNTLG  021723          4758    5356    5438#
$CNTLU  021716          4803    5373    5437#
$COMND= 000002          2690*   2903*   2912*   2920*   2957*   3197*   3240*   3531*   3557*   3626*   7275#   7288    7290
```

```
                        7292    7294    7296    7298    7300    7302    7304    7306    7308    7310    7312    7314    7316
                        7318
$CPUOP  001236          1807#
$CRLF   001205          1782#   2476    2605    2647    2649    2659    2729    2794    2836    2858    2844    2853    3028
                        3083    3120    3122    3132    3149    3811    3813    3879    3886    3888    3890    3895    3896
                        4204    4206    4231    4233    4269    4348    4748    4757    4813    4841    4842    4844    4845
                        4847    4931    4956    4965    4984    4989    4993    5049    5084    5384
$CYL    = 000012        2687*   2898*   2980*   2998*   3002*   3003    3198*   3224*   3225    3238*   3293*   3294    3297*
                        3333    3536*   3551*   3552    3562*   3597*   3598    3632*   3674*   3675    3680*   7281#
$DBLK   021304          5291    5325    5333#
$DB2D   022046          4707    5504#
$DB20   022242          4724    5565#
$DECYL  022226          5506    5552#
$DEVCT  001220          1798#
$DEVM   001266          1834#   2637*   2897    2948    2961    3039    3051*   3077*   3198    3201    3238    3297    3446
                        3534    3560    3630    3678    3749    3831    3839*   3840    3902
$DOAGN  017322          4867    4872    4879    4885#
$DSPLY  016074          4564#   5703
$DTBL   021274          5294    5329#
$EMTAB= 000022          4423    7285#   7288    7290    7292    7294    7296    7298    7300    7302    7304    7306    7308
                        7310    7312    7314    7316    7318
$ENDAD  017312          1713    4881#   4951
$ENDCT  017260          2342    4869#
$ENV    001230          1803#   2392    4933    5028    5097    5121
$ENVM   001231          1804#   2379    5030    5035    5099
$EOP    017224          3846    4859#
$EOPCT  017252          2342*   4866#   4870
$ERFLG  001117          1751#   4918*   4956    5599    5628    5630    5636*   5658
$ERMAX  001131          1757#   2345*   5630    5653*   5658
$ERROR  017366          2336    4916#
$ERRPC  001132          1758#   4925*   4926*   4927    4956    497?
$ERRTB  002742          2238#   4979
$ERRTY  017566          4930    4964#
$ERTTL  001126          1755#   4924*   4956
$ESCAP  001176          1779#   2344*   4947    4949    4956    5652*   6509    6560    6564    6819    6879    7089    7098
                        7099*   7101*
$ETABL  001230          1802#
$ETEND  001274          1739    1837#
$FATAL  001212          1795#   5125*
$FFLG   020450          5088*   5091*   5119    5128*   5136#
$FILLC  001172          1776#   5053    5084
$FILLS  001171          1775#   5084
$FMT    = 000001        7274#   7275    7276    7277    7278    7279    7280    7281    7282    7283    7284
$GAP    = 000016        2569*   7284#   7285
$GDAOR  001134          1759#
$GDOAT  001140          1761#
$GET42  017264          4871#
$GTSWR  021364          4762    5357#   5696
$HD     = 000000        1537
$HIBTS  001100          1734#
$ICNT   001120          1752#   5643*   5644    5646*   5657
$ILLUP  020624          5147    5163    5182#
$INTAG  001151          1766#   5385    5443
$ITEMB  001130          1756#   4927*   4935    4956    4968
$LF     001206          1783#   4823    4956    5084
$LFLG   020447          5129*   5135#
```

```
$LKCSB  001302          1844#   4322*
$LKCSR  001300          1843#   4316    4323*
$LKS    001306          1846#   4327    4332*
$LLVEC  001310          1847#   4329
$LOOP   017360          4893    4897#
$LPADR  001122          1753#   2360*   5634*   5650*   5655    5657
$LPERR  001124          1754#   2361*   4946    5634    5651*   5657
$LPVEC  001304          1845#   4319
$LSTAD  031312          2419    7135*   7139#
$MADR1  001242          1820#
$MADR2  001246          1824#
$MADR3  001252          1827#
$MADR4  001256          1830#
$MAIL   001210          1735    1739    1793#   2378    2392    4933    5028    5649
$MAMS1  001240          1814#
$MAMS2  001244          1822#
$MAMS3  001250          1825#
$MAMS4  001254          1828#
$MBADR  001102          1735#
$MFLG   020046          5089*   5095    5130*   5134#
$MNEW   021741          5360    5441#
$MSGAD  001224          1800#   5105*   5108
$MSGLG  001226          1801#   5110*
$MSGTY  001210          1794#   5103    5111*   5123    5127*
$MSWR   021730          5357    5439#
$MTYP1  001241          1815#
$MTYP2  001245          1823#
$MTYP3  001251          1826#
$MTYP4  001255          1829#
$MXCNT  022640          5647    5657#
$NULL   001170          1774#   5055    5084
$NWTST= 000000          2823#   2889#   2942#   3021#   3100#   3182#   3520#   3613#   3707#
$OCNT   021064          5221*   5250*   5263#
$OCTVL  022344          5567    5592#
$OMODE  021066          5216*   5220*   5225    5228*   5239*   5265#
$OVER   022624          5611    5627    5635    5645    5654#
$PASS   001216          1797#   2378*   4863*   4864*   4900    5641    5658
$PASTM  001106          1737#
$PHYDR= 000015          2568*   7283#
$POWER  020632          5178    5185#
$PSEL = 000003          7276#
$PWRON  020452          2340    5147#   5175
$PWRMG  020606          5178#
$PWRUP  020524          5157    5163#
$QUES   001204          1781#   4816    4956    5084    5403
$ROCHR  021576          5416#   5699
$RODEC= ****** U        5701
$ROLIN  016644          4776#   5700
$ROOCT= ****** U        5701
$ROSZ = 000001          5437#
$RESRE  022010          5478#   5702
$RMADR  001274          1841#   2400    2510*   2513    2519    7153    7171    7175
$RMAS = 000036          7366#   7367
$RMBA = 000024          4197    4207    4224    7361#   7362
$RMCS1= 000020          3944    3946    4177    7359#   7360
$RMCS2= 000030          4016    4179    7363#   7364
```

```
$RMDA = 000026          3259    3290    3492    3649    3671    7362#   7363
$RMDB = 000042          7368#   7369
$RMDC = 000054          7373#   7374
$RMDS = 000032          3948    4013    4048    7364#   7365
$RMDT = 000046          7370#   7371
$RMEC1= 000064          7377#   7378
$RMEC2= 000066          7378#
$RMER1= 000034          4022    4025    4028    4031    4034    4037    4040    4043    4046    4051    4054    4074    4181
                        7365#   7366
$RMER2= 000062          4057    4059    4183    7376#   7377
$RMHR = 000056          7374#   7375
$RMLA = 000040          7367#   7368
$RMMR1= 000044          7369#   7370
$RMMR2= 000060          7375#   7376
$RMOF = 000052          7372#   7373
$RMSN = 000050          7371#   7372
$RMVEC  001276          1842#   2401    2511*   2514    2520    7162
$RMWC = 000022          3257    3260    3288    3291    3499    3501    3647    3650    3669    3672    4191    4210    7360#
                        7361
$RTNAD  017362          4899#
$RTRN   017356          2348    2350*   2355*   4875    4894#
$R2A  = ****** U        5703
$SAVRE  021752          5462#   5701
$SAVR6  020630          5156*   5164    5165*   5166*   5184#
$SB2D   016402          4298    4350    4355    4360    4705#
$SB2O   016432          4282    4722#
$SCOPE  022362          2334    5608#
$SEC  = 000010          2689*   2897*   2982*   2990*   2991    2993*   3196*   3237*   3259*   3270*   3290*   3334    3493
                        3503    3532*   3550*   3595*   3625*   3649*   3654*   3671*   7279#
$SETUP= 000137          2311#   2333    2334    2336    2338    2340    2342    2343    2344    2346    2360    2389    4861
                        4917    4943    4951    5180    5340    5443    5609
$SIZE   031216          2418    7118#
$STUP = 177777          2311#
$SUPRS  016006          4299    4535#
$SVLAD  022570          5619    5648#
$SVPC = 000224          1711#   1716
$SWR  = 177600          1526#   1537    1541    1542    1543    1544    1545    1546    1547    1548    1778    1779    1780
                        2343    2344    2346    2360    2361    2825    2891    2944    3023    3102    3184    3522    3615
                        3709    4855    4862    4873    4886    4900    4908    4909    4910    4911    4912    4921    4928
                        4940    4944    4956    5179    5600    5601    5602    5603    5604    5610    5622    5624    5625
                        5628    5629    5630    5637    5638    5639    5651    5654    5657
$SWREG  001232          1805#   2381
$SWRMK= 000000          1548    1549    5604    5605    5626
$SYSNM= 000014          2552*   2586    2831    2833    3108    3110    3910    7282#
$TATUS= 000016          3942    3956    3962    3964    3966    3968    3970    3972    4013    7358#
$TBIT   017364          2359*   4890*   4900#   5180*
$TERM = 000032          5704#
$TESTN  001214          1796#   5649*
$TIME   015136          4271    4346#   4397
$TIMES  001174          1778#   2343*   2825*   2891*   2944*   3023*   3102*   3184*   3522*   3615*   3709*   4862*   5637*
                        5644    5647*   5657
$TKB    001162          1771#   4398    4405    4736    5338    5349    5366    5420    5426
$TKINT  016462          2388    4734#
$TKS    001160          1770#   4406*   4737*   4751*   4761*   4763*   5338    5347    5363    5387*   5418    5424
$TKSRV  016512          4734    4744#
$TN   = 000012          1526#   1537    2823    2825#   2889    2891#   2942    2944#   3021    3023#   3100    3102#   3182
```

```
                        3184#   3520    3522#   3613    3615#   3707    3709#
$TNPWR  022156          5511    5512    5532#
$TPB    001166          1773#   5073#   5084
$TPFLG  001173          1777#   5022    5084
$TPS    001164          1772#   5071    5084
$TRAP   022642          2338    5667#
$TRAP2  022664          5678#   5689
$TRK  = 000011          2688#   2902#   2921#   2922    2981#   2994#   2995    2997#   3220#   3221    3223#   3298#   3299
                        3438    3547#   3548    3589#   3591    3681#   3682    7280#
$TRP  = 000015          5682#   5691#   5692#   5693#   5694#   5695#   5696    5697#   5698    5699#   5700#   5701#   5702#
                        5703#   5704#
$TRPAD  022676          5672    5689#   5704
$TSTM   001104          1736#
$TSTNM  001116          1750#   4861#   4920    4956    5599    5626    5648#   5649    5654    5658
$TTYIN  017122          4778    4779    4791    4814    4828    4832#
$TYPBN= ****** U        5695
$TYPDS  021070          5279#   5694
$TYPE   017722          4554    4566    5022#   5116    5682    5690
$TYPEC  020134          5052    5059    5066    5071#   5072    5389
$TYPEX  020202          5077    5079    5082#
$TYPOC  020666          5219#   5691
$TYPON  020702          5218    5221#   5693
$TYPOS  020642          5214#   5692
$UNIT   001222          1799#   1853
$UNITM  001110          1738#
$USWR   001234          1806#
$VECT1  001260          1831#
$VECT2  001262          1832#
$WROM = 000004          2686#   2899#   2960#   3195#   3260#   3269#   3291#   3335    3501    3530#   3624#   3650#   3653#
                        3672#   4193    4208    4240    7277#
$XTSTR  022374          5613#
$$GET4= 000001          4873#   4878
$OFILL  021065          5215#   5219#   5229    5264#
$40CAT= ****** U        4930    5610
.     = 037550          1693#   1697#   1705#   1711    1712#   1714#   1716#   1717#   1723    1724#   1726#   1728#   1740
                        1747#   1784    1995#   2331    2360    2361    4014    4017    4023    4026    4029    4032    4035
                        4038    4041    4044    4052    4055    4483    4832#   4835#   4900    4901    4956    5003#   5084
                        5137#   5159    5183    5333#   5338    5443    5552#   5592#   5657    5658    5704    6987#   6988#
                        6989#   6990#   6991#   6992#   6993#   6994#   6995    7240#   7288#   7290#   7292#   7294#   7296#
                        7298#   7300#   7302#   7304#   7306#   7308#   7310#   7312#   7314#   7316#   7318#   7385#   7386
.$ASTA= ****** U        5089    5092
.$X   = 001100          1723#   1728
```

```
CKCHR    1526#  4656   4670
CKDIG    1526#
CKNUM    1526#
COMMEN   1660#
ENDCOM   1660#
ERRCAL   5708#  6509   6560   6563   6818   6879
ERROR    1554#  4168   4169   4170   4171   6510   6561   6565   6820   6880   7182
ESCAPE   1660#
GETPRI   1660#  4886   7124
GETSWR   1526#  1660#  2389
MORETA   1717#  1839
MULT     1660#
NEWTST   1660#  2823   2889   2942   3021   3100   3182   3520   3613   3707
POP      1660#  3466   3505   3927   4427   5131   5132   5168   5169   5320   5483
PUSH     1660#  3418   3488   3871   4416   5092   5094   5115   5149   5155   5279   5463
REPORT   1660#
RM03.D      1#  5708
SCOPE    1555#  2824   2890   2943   3022   3101   3183   3521   3614   3708
SETPRI   1660#
SETTRA   5682#  5691   5692   5693   5694   5696   5698   5699   5700   5701   5702   5703
SETUP    1660#  2326
SKIP     1660#
SLASH    1660#
SPACE    1660#
STARS    1660#  1669   1709   1720   1722   1729   1743   1784   1787   1883   2222   2823   2889   2942   3021
         3100   3182   3520   3613   3707   4255   4259   4303   4307   4853   4904   4959   5007   5087   5145
         5161   5191   5269   5337   5340   5408   5447   5493   5556   5596   5661   5706   5717   7104   7108
         7112   7234   7238
SWRSU    1660#  2362#
TRMTRP   5682#  5704
TYPBIN   1660#
TYPDEC   1660#
TYPNAM   1660#
TYPNUM   1660#
TYPOCS   1660#
TYPOCT   1660#  4971   4995   5358
TYPTXT   1660#
$$CMRE   1741#
$$CMTM   1741#
$$ESCA   1660#
$$NEWT   1660#  2823   2889   2942   3021   3100   3182   3520   3613   3707
$$SET    5682#  5691   5692   5693   5694   5696   5698   5699   5700   5701   5702   5703
$$SETM   2378#
$$SETU   2346#
$$SKIP   1660#
.EQUAT   1526#  1550
.HEADE   1526#  1527
.SETUP   1526#  2311
.SWRHI   1526#  1537
.SWRLO   1526#  1549#
.$ACTI   1526#  1707
.$APTB   1526#  1785#
.$APTH   1526#  1718
.$APTY   1526#  5085
.$CATC   1526#  1691
.$CMTA   1526#  1741
```

```
.$D820   1526#   5491
.$D820   1526#   5554
.$EOP    1526#   4851
.$ERRO   1526#   4902
.$ERRT   1526#   4957
.$POWE   1526#   5143
.$READ   1526#   5335
.$SAVE   1526#   5445
.$SCOP   1526#   5594
.$SIZE   1526#   7110
.$TRAP   1526#   5659
.$TYPD   1526#   5267
.$TYPE   1526#   5005
.$TYPO   1526#   5189


. ABS.  037550     000


 ERRORS DETECTED:  0

 DSKZ:DZRMIA,DSKZ:DZRMIA.SEQ/NL:MD:MC:CND:TOC/LI:ME/DOC/SOL/CRF=DSKM:RMDRV4.P11,DSKM:DZRMIA.P11
 RUN-TIME:  29 20 2 SECONDS
 RUN-TIME RATIO: 717/52=13.7
 CORE USED:  40K  (79 PAGES)

 DOCUMENT PAGES:  161
```