

IDENTIFICATION

SEQ 0001

PRODUCT CODE: AC-9236F-MC  
PRODUCT NAME: CZRKIFO RK11 UTILITY PACKAGE  
DATE CREATED: MARCH, 1978  
MAINTAINER: DIAGNOSTIC GROUP  
AUTHOR: BOB COLLINS  
REVISED BY: JIM KAPADIA  
TOM SAWYER  
CHUCK HESS

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS MANUAL.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

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

COPYRIGHT (C) 1974, 1978 BY DIGITAL EQUIPMENT CORPORATION

700  
10 000 00  
10000 000 / 00 00 00 00 00  
1 0 0 0

TABLE OF CONTENTS

1.0	ABSTRACT
2.0	REQUIREMENTS
2.1	EQUIPMENT
2.2	STORAGE
2.3	PRELIMINARY PROGRAMS
3.0	LOADING PROCEDURE
4.0	STARTING PROCEDURE
5.0	OPERATING PROCEDURE
6.0	ERRORS
7.0	RESTRICTIONS
8.0	EXECUTION TIME
9.0	PROGRAM DESCRIPTION
9.1	PROGRAM INDEX
9.2	COMPATIBILITY PACKAGE
9.3	OSCILLATING SEEK PACKAGE
9.4	FORMATTER SURFACE VERIFIER
9.5	RK05 CONTROL PANEL TEST # 2
9.6	HEAD ALIGNMENT ROUTINE
9.7	(DISK) POWER FAILURE TEST
9.8	SECTION SPECIAL
9.9	COMPATIBILITY ERROR RECOVERY
9.10	

## 1. ABSTRACT

1.1 THIS PACKAGE CONTAINS 4 INDIVIDUAL UTILITY PROGRAMS FOR THE RKXX PLUS A MINI-MONITOR WHICH ALLOWS TEST SELECTION AND PARAMETER INPUT VIA THE CONSOLE DEVICE. ALL UTILITY PACKAGES ARE EXPLAINED IN DETAIL IN PARAGRAPH 9.

## 2. REQUIREMENTS

2.1 EQUIPMENT  
PDP-11 PROCESSOR  
8K MEMORY  
RK11 OR RKV11 CONTROLLER  
1-8 RK05 OR RK05F DISK DRIVES (DRIVE TYPES MAY BE MIXED)

2.2 STORAGE  
THIS PROGRAM REQUIRES 8K

2.3 PRELIMINARY PROGRAMS  
THIS IS NOT A DIAGNOSTIC, PACKAGE IT IS ASSUMED THAT ALL EQUIPMENT IS FUNCTIONAL

## 3. LOADING PROCEDURE

3.1 METHOD  
PROCEDURE FOR NORMAL BINARY TAPES SHOULD BE FOLLOWED  
A. ABSOLUTE LOADER MUST BE IN MEMORY.  
B. PLACE BINARY TAPE IN READER.  
C. LOAD ADDRESS \*7500 (\*DETERMINED BY LOCATION OF LOADER).  
D. PRESS "START" PROGRAM WILL LOAD.

## 4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS  
NONE

4.2 STARTING ADDRESS  
200 MINI MONITOR

4.3 PROGRAM AND/OR OPERATOR ACTION  
LOAD PROGRAM INTO MEMORY  
SET SWITCH REGISTER TO STARTING ADDRESS (200)  
LOAD ADDRESS  
PRESS START

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E. AN 11/34) THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE 'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176 (8). THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' whenever the program enters the scope routine or begins a new test. the 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

'SWR = NNNNNN NEW ='

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

ON PROCESSORS WITH HARDWARE SWITCH REGISTERS, THE 'SOFTWARE' SWITCH REGISTER MAY BE USED. IF THE PROGRAM FINDS ALL 16 SWITCHES IN THE 'UP' POSITION, ALL SWITCH REGISTER REFERENCES WILL BE TO THE 'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST BE FOLLOWED.

PROGRAM WILL TYPE MINI MONITOR ROUTINE

5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS  
SEE SEC. 9.0 FOR SWITCHES APPLICABLE TO INDIVIDUAL ROUTINES.

5.2 SUBROUTINE ABSTRACTS  
NOT APPLICABLE

5.3 PROGRAM AND/OR OPERATOR ACTOR  
SEE INDIVIDUAL PACKAGE DESCRIPTION (PARAGRAPH 9)

6. ERRORS

6.1 ERROR HALTS AND DESCRIPTION  
IF HALTED A MAJOR PROBLEM EXIST CHECK CODE AT HALT PC TO DETERMINE WHAT OCCURRED.

6.2 ERROR RECOVERY  
EXPLAINED IN DETAIL IN INDIVIDUAL PACKAGE DESCRIPTION (PARAGRAPH 9)

7. RESTRICTIONS

7.1 STARTING RESTRICTIONS  
IT IS NOT RECOMMENDED THAT YOU START AT AN ADDRESS OTHER THAN 200, (REASON EXPLAINED IN PARAGRAPH 9.1) UNLESS DIRECTED TO BY THE PROGRAM.

7.2 OPERATIONAL RESTRICTIONS  
EXPLAINED IN DETAIL IN INDIVIDUAL PACKAGE DESCRIPTIONS (PARAGRAPH. 9)

8. EXECUTION TIME

VARIABLES WITH SELECTED ROUTINE, NUMBER OF DRIVES, ETC.

9. PROGRAM DESCRIPTION

THE RK11 UTILITY PACKAGE IS DIVIDED INTO EIGHT SECTIONS WHICH ALLOW COMPATABILITY TESTING, OSCILLATING SEEKS FOR SERVO ADJUSTMENT AND SEEK LOGIC WAVEFORM ANALYSIS, PACK FORMATTING AND SURFACE VERIFICATION, AND FRONT PANEL TESTING (INDICATOR LAMPS, SWITCHES, INTERLOCKS, ETC) AND VERIFICATION. THE PACKAGE IS DIVIDED INTO FIVE SECTIONS

SECTION	NAME
0	INDEX
1	COMPATIBILITY TEST
2	OSCILLATING SEEK PACKAGE
3	FORMATTER SURFACE VERIFIER
4	FRONT PANEL TEST
5	RK05 CONTROL PANEL TEST #2
6	HEAD ALIGNMENT ROUTINE
7	POWER FAILURE (DURING WRITE) TEST

NOTE: NORMAL LINKAGE TO ANY OF THESE PACKAGES IS THRU SECTION 0 (SEE PARAGRAPH 9.1)

9.1 SECTION 0 INDEX

**PURPOSE:** TO ALLOW THE USER TO SELECT AND RUN TESTS VIA THE CONSOLE DEVICE IN AN EFFORT TO FREE HIM FROM REMEMBERING VARIOUS SWITCH SETTINGS.

**DESCRIPTION:** LOAD START ADDRESS 200, A TABLE IS PRODUCED WHICH TELLS THE USER THE NAME AND TYPE OF THE TEST. (TYPE IS AN OCTAL CODE BY WHICH THE USER SELECTS THE TEST). AFTER THE TABLE IS TYPED, THE QUESTION "TYPE =" IS ASKED, THE USER THEN TYPES THE NUMERAL 0-4 TO SELECT A TEST.

**USE:** THIS IS EXAMPLE OF THE ACTUAL OUTPUT:

RK11 UTILITY PACKAGE

NAME	TYPE
INDEX	0
COMPATIBILITY PACKAGE	1
OSCILLATING SEEK PACKAGE	2
FORMATTER-SURFACE VERIFIER	3
RK05 CONTROL PANEL TEST	4
RK05 CONTROL PANEL TEST #2	5
HEAD ALIGNMENT ROUTINE	6
POWER FAILURE (WRITE) TEST	7

TYPE=X

WERE "X" IS THE RESPONSE (0-7) BY THE USER

**ERROR INFO:** ANY ILLEGAL INPUT IS HANDLED, A QUESTION MARK IS TYPED AND THE QUESTION "TYPE =" IS RE-ASKED.

9.2 SECTION 1 COMPATIBILITY PACKAGE

**PURPOSE:** TO CONFIRM THE FACT THAT A GROUP OF DRIVES (A MAXIMUM OF EIGHT) ARE TRULY COMPATIBLE.

**DESCRIPTION:** THIS PACKAGE DOES NOT APPLY TO RK-05F DRIVES. THIS PACKAGE ALLOWS A USER TO AUTOMATICALLY TEST



RK11 UTILITY PACKAGE

```

INDEX      NAME      TYPE
COMPATIBILITY PACKAGE  0
                                     1
*****

```

...THE USER SELECTED TYPE ONE (1) AND RECEIVED THE MESSAGE RKXX COMPATIBILITY PACKAGE AND WAS THEN ASKED FOR SYSTEM 1 DRIVES HE TYPES EACH SELECTED DRIVE NUMBER SEPARATED BY COMMAS HE TERMINATES THE STRING WITH A PERIOD THEN A CARRIAGE RETURN HE IS ASKED IF THERE IS A SECOND SYSTEM, HE TYPES N FOR NO. HE NOW RECEIVES A STRING OF MOVE DIRECTIVES TELLING HIM EXACTLY WHERE TO MOVE THE TEST PACK AND WHAT TO DO. FINALLY THE USER RECEIVES THE MESSAGE "DONE!" INDICATING A SUCCESSFUL PASS. AT THIS POINT ANY DRIVE WHICH HAS NOT BEEN DECLARED DOWN AND DID NOT RECEIVE AN ERROR\* MESSAGE IS COMPATIBLE WITH ANY OTHER SELECTED DRIVE MEETING THE SAME CONDITIONS. FINALLY THE INDEX ROUTINE IS AUTOMATICALLY RE-ENTERED AND USER IS READY TO MAKE ANOTHER SELECTION. \*SEE ERROR INFO TO DETERMINE THE TYPE OF ERROR WHICH CONSTITUTES INCOMPATIBILITY. \*\*\*\*\*

EXAMPLE 2

THE USER NOW DESIRES TO TEST COMPATIBILITY ON TWO SYSTEMS HE HAS UNITS 0,1 ON SYSTEM ONE AND UNIT 0 ON SYSTEM 2, IT GOES LIKE THIS....  
 RK11 UTILITY PACKAGE

```

INDEX      NAME      TYPE
COMPATIBILITY PACKAGE  0
OSCILLATING SEEK PACKAGE  1
FORMATTER-SURFACE VERIFIER  3
RK05 CONTROL PANEL TEST  4
RK05 CONTROL PANEL TEST #2  5
HEAD ALIGNMENT ROUTINE  6
POWER FAILURE (WRITE) TEST  7

```

TYPE=1  
 DRIVE NUMBERS ON SYSTEM 1=1,0

IS THERE A SECOND SYSTEM?  
 DRIVE # =0  
 MOUNT PACK ON DRIVE #1  
 MAKE PACK WRITE ENABLE  
 PRESS CONTINUE WHEN DRIVE RDY  
 MOUNT PACK ON DRIVE #0  
 MAKE PACK WRITE ENABLE  
 PRESS CONTINUE WHEN DRIVE RDY  
 LOAD AND START ADDRESS 210 ON SYSTEM #2  
 AND TYPE THE BELOW WHEN ASKED ON SYSTEM #2

AND TYPE THE BELOW WHEN ASKED FOR IT.

WORD 1=000002  
WORD 2=000200

\*\*\*\*\*

....THE ONLY DIFFERENCE BETWEEN THIS AND SINGLE SYSTEM IS THE NEW DIRECTIVE TO LOAD START 210 ETC. THE USER NOW LOADS AND STARTS SYSTEM TWO AND THE BELOW IS TYPED...

\*\*\*\*\*

COMPATIBILITY-SYSTEM#2  
WORD 1=000002  
WORD 2=000200

MOUNT PACK ON DRIVE #0  
MAKE PACK WRITE ENABLE  
PRESS CONTINUE WHEN DRIVE RDY  
DONE SYSTEM 2 RESTART SYSTEM 1, TYPE WORD 000077

\*\*\*\*\*

.....THE USER RESPONSE TO THE QUESTION WORD 1 = BY TYPING WORD 1 FROM PROCESSOR ONE AND WORD 2 =, BY TYPING WORD TWO FROM PROCESSOR 1 HE RECEIVES THE MOUNT COMMAND MOVES THE TEST PACK TO SYSTEM TWO, DRIVE NUMBER (0), AND PRESSES CONTINUE. NOW THE MESSAGE TO RETURN TO SYSTEM ONE\*

\*SYSTEM ONE HAS BEEN IN A HALT STATE AND SHOULD BE LEFT THAT WAY UNTIL THE RETURN FROM SYSTEM TWO SO THAT TABLES, ETC. BUILT FOR THE TEST WILL NOT BE DISTURBED.

\*\*\*\*\*

WORD=000077  
MOUNT PACK ON DRIVE #1  
MAKE PACK WRITE ENABLE  
PRESS CONTINUE WHEN DRIVE RDY  
MOUNT PACK ON DRIVE #0  
MAKE PACK WRITE ENABLE  
PRESS CONTINUE WHEN DRIVE RDY  
DONE!

RK11 UTILITY PACKAGE

INDEX	NAME	TYPE
0	COMPATIBILITY PACKAGE	0
1	OSCILLATING SEEK PACKAGE	1
2	FORMATTER-SURFACE VERIFIER	2
3	RK05 CONTROL PANEL TEST	3
4	RK05 CONTROL PANEL TEST #2	4
5	HEAD ALIGNMENT ROUTINE	5
6	POWER FAILURE (WRITE) TEST	6
7		7

TYPE=

\*\*\*\*\*



THE USER NOW PASSES CONTINUE ON PROCESSOR ONE AND IN RESPONSE TO THE QUESTION, WORD =, TYPES THE WORD GIVEN TO HIM FROM PROCESSOR TWO THEN EVERYTHING BECOMES THE SAME AS A SINGLE SYSTEM. THE USER NEARLY FOLLOWS DIRECTIONS. ERROR INFO: SEE PARAGRAPH 9.6 SPECIAL SECTION

9.3 SECTION 2 OSCILLATING SEEK PACKAGE

PURPOSE: TO ALLOW THE USER TO MAKE SERVO ADJUSTMENTS AND/OR SEEK LOGIC CHECKOUT BY PERFORMING SEEKS BETWEEN USER SPECIFIED ADDRESS

DESCRIPTION: SELECT TYPE 2, THE USER THEN INSERTS THE DRIVES TO BE TESTED IN SW0 TO SW7 OF THE SWITCH REGISTER. A SWITCH IS SET FOR EACH DRIVE (E.G. SW2 TO TEST DRIVE 2. THE USER THEN INSERTS THE ADDRESS TO SEEK IN THE SWR. IF BOTH ADDRESS ARE LEGAL, 50 CYCLES (100 SEEKS) WILL BE MADE BETWEEN THE SPECIFIED ADDRESS THEN THE PROGRAM WILL LOOK AT THE SWR FOR POSSIBLE CHANGES THIS SHOULD ALLOW FOR GOOD STABLE TRACES ON AN OSCILLOSCOPE. IT SHOULD BE NOTED THAT THE OSCILLATING SEEKS BETWEEN THE SPECIFIED CYLINDERS ARE DONE ON ALL AVAILABLE DRIVES. THE ONLY WAY TO EXIT IS HALTI, LOAD ADDRESS 200, HIT START.

USE: SELECT TYPE 2, RESPOND TO QUESTION WITH UNIT NUMBER... TYPE=2

OSCILLATING SEEK PACKAGE  
 SET SW0 TO SW7 TO SELECT THE DRIVES TO TEST AND CONTINUE. IF ALL SWITCHES ARE RESET, ALL AVAILABLE DRIVES WILL BE TESTED.  
 TOGGLE THE "FIRST CYLINDER ADDRESS" (OUTER LIMIT) INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST CYLINDER ADDRESS" (INNER LIMIT) INTO THE HIGH BYTE (BIT8-15), THEN PRESS CONTINUE  
 ...FOLLOW INSTRUCTIONS TYPED

ERROR INFO: IF AN ILLEGAL ADDRESS IS SELECTED A MESSAGE IS TYPED AND USER NEARLY SELECTS LEGAL ADDRESS AND DEPRESSES CONTINUE

EXAMPLE TYPEDOUT  
 INVALID ADDRESS IN SWITCH REGISTER TRY AGAIN  
 INVALID ADDRESS IN SWITCH REGISTER TRY AGAIN  
 INVALID ADDRESS IN SWITCH REGISTER TRY AGAIN

\*\*NOTE:\*\* BOTH DRIVES OF AN RK-05F SHOULD NOT BE SELECTED FOR TESTING AT THE SAME TIME.

9.4 SECTION 3 FORMATTER-SURFACE VERIFIER

PURPOSE: TO FORMAT VIRGIN PACKS OR REFORMAT AN OLDER PACK AND VERIFY ITS SURFACE

DESCRIPTION: SELECT TYPE 3, RESPOND TO THE QUESTION BY SETTING SWITCHES CORRESPONDING TO DRIVE NUMBERS TO BE FORMATTED. THIS IF DRIVES 0,1,2 ARE TO BE FORMATTED SET SWITCHES 0,1,2. THE DRIVES ARE FORMATTED ONE AFTER ANOTHER AT COMPLETION PACK GOOD

USE: MESSAGE IS TYPED AND PACK IS FORMATTED.  
 SELECT TYPE 3, RESPOND TO QUESTION WITH  
 SETTING OF SWITCH REGISTER.  
 \*\*\*\*\*

RK11 UTILITY PACKAGE

INDEX	NAME	TYPE
0	COMPATIBILITY PACKAGE	0
1	OSCILLATING SEEK PACKAGE	2
2	FORMATTER-SURFACE VERIFIER	3
3	RK05 CONTROL PANEL TEST	4
4	RK05 CONTROL PANEL TEST #2	5
5	HEAD ALIGNMENT ROUTINE	6
6	POWER FAILURE (WRITE) TEST	7

TYPE=3  
 FORMATTER-SURFACE VERIFIER, SET SW REG WITH DRIVE #'S

PACK GOOD.

RK11 UTILITY PACKAGE

INDEX	NAME	TYPE
0	COMPATIBILITY PACKAGE	0
1	*****	1

AFTER THE PACK IS FORMATTED A GOOD MESSAGE IS  
 GIVEN AND A CHECK IS MADE TO SEE IF THERE ARE  
 ANY MORE PACKS TO BE FORMATTED. IF THERE ARE  
 NONE CONTROL IS TRANSFERRED TO THE MINI-MONITOR  
 ERROR INFO: DRIVE PROBLEM, IF THE MESSAGE....  
 SYSTEM ERROR

....IS TYPED IT INDICATES A FAULTY DRIVE OR  
 CONTROLLER, RUN DIAGNOSTICS, THE PROCESSOR WILL HALT  
 PRESS CONTINUE TO RETURN TO MINI MONITOR.  
 BAD SPOT, OR SURFACE PROBLEM, ETC.

PACK FAILED AT (IN OCTAL) CYLINDER SECTOR SURFACE

9.5 SECTION 4 RK05 CONTROL PANEL TEST

PURPOSE: TO INSURE ALL SWITCHES INDICATOR LAMPS, AND INTERLOCKS  
 ARE FUNCTIONAL IN THE RK05  
 DESCRIPTION: SELECT TYPE 4, RESPOND TO QUESTION WITH UNIT NUMBER, FOLLOW  
 DIRECTIONS GIVEN. AT COMPLETION MESSAGE "DONE!" IS GIVEN  
 USE: SELECT TYPE 4, RESPOND TO QUESTION WITH THE UNIT NUMBER....  
 \*\*\*\*\*

INDEX	NAME	TYPE
0	COMPATIBILITY PACKAGE	0
1	OSCILLATING SEEK PACKAGE	2
2	FORMATTER-SURFACE VERIFIER	3
3	RK05 CONTROL PANEL TEST	4
4	RK05 CONTROL PANEL TEST #2	5
5	HEAD ALIGNMENT ROUTINE	6
6	POWER FAILURE (WRITE) TEST	7

TYPE=4

RK05 CONTROL PANEL TEST, WHICH DRIVE?0  
MOUNT PACK ON DRIVE#0  
PLACE DRIVE IN RUN ;SHOULD SEE THE RUN,  
POWER, AND ON CYLINDER LAMPS LIGHT.  
MAKE DRIVE WRITE ENABLE PRESS CONTINUE

WRITE PROTECT THE DRIVE THEN PRESS CONTINUE

CLEAR WRITE PROTECT THEN PRESS CONTINUE

CAUTION! TRY TO OPEN THE DOOR, DO NOT FORCE:  
DOOR SHOULD NOT OPEN!  
PRESS CONTINUE WHEN FINISHED

PUT DRIVE IN LOAD, WAIT FOR LOAD LIGHT  
PRESS CONTINUE WHEN FINISHED

OPEN THE DOOR, PUT DRIVE IN RUN  
CAUTION! IF RUN LIGHT ON ERROR! DEPRESS  
LOAD IMMEDIATELY, CONTINUE WHEN FINISHED

REMOVE THE PACK, CLOSE THE DOOR  
PUT DRIVE IN RUN, DRIVE SHOULD NOT  
RUN...INTERLOCKS HAVE BEEN CHECKED  
DONE!

RK11 UTILITY PACKAGE

INDEX	NAME	TYPE
0	COMPATABILITY PACKAGE	0
1	OSCILLATING SEEK PACKAGE	1
2	FORMATTER-SURFACE VERIFIER	2
3	RK05 CONTROL PANEL TEST	3
4	RK05 CONTROL PANEL TEST #2	4
5	HEAD ALIGNMENT ROUTINE	5
6	POWER FAILURE (WRITE) TEST	6
7		7

TYPE=

\*\*\*\*\*

9.6 SECTION 5 RK05 CONTROL PANEL TEST #2

PURPOSE: TO GIVE A CONTINUOUS MONITORING AND  
CHECKING CAPABILITY FOR THE FOLLOWING  
CONDITIONS ON THE VARIOUS DRIVES:  
OFF LINE (RDY CLR)/ON LINE (RDY SET)  
WRITE PROTECTED/WRITE ENABLED  
POWER LOW/POWER UP  
SEEK INCOMPLETE/SEEK OK

DESCRIPTION: SELECT TYPE 5, PUT ALL THE DRIVES THAT  
ARE TO BE MONITORED AND CHECKED ON 'RUN'.  
NOTE THAT THIS IS IMPORTANT BECAUSE THE  
PROGRAM HAS TO KNOW WHICH DRIVES ARE TO  
BE CHECKED.

USE: AFTER HAVING SELECTED TYPE 5 AND PUTTING  
THE DRIVES THAT ARE TO BE MONITORED ON  
'RUN', THE PROGRAM PRINTS OUT ALL THE  
DRIVES THAT ARE 'ON LINE'.

DRIVE 0 ON LINE  
 DRIVE 1 ON LINE  
 DRIVE 2 ON LINE

THE PROGRAM, THEN STARTS SCANNING ALL DRIVES, ONE AFTER THE OTHER. CHECKS IF THE DRIVE IS ON LINE OR OFF LINE (DRY SET OR CLEAR). THEN IT CHECKS IF THE DRIVE IS WRITE ENABLED OR WRITE PROTECTED. THEN A SEEK (TO CYLINDER 1) IS DONE AND 'DPL' BIT IS CHECKED TO SEE IF DRIVE POWER IS LOW OR OK. IF THE DRIVE IS POWERED, IT IS CHECKED IF THE SEEK IS DONE OR SEEK INCOMPLETE OCCURS. WHEN EVER ANY CHANGE IN THE STATUS IS FOUND, IT IS REPORTED. IF THE DRIVES PUT ON 'LOAD' AND BACK TO 'RUN', THE PROGRAM CHECKS IF THE DRIVE COMES ON LINE IN THE WRITE ENABLED MODE. IF NOT, AN ERROR MESSAGE (ERROR, NOT WRITE ENABLED) IS REPORTED. THEN THE DRIVE IS WRITE PROTECTED.  
 EX: IN A SYSTEM UNDER TEST, IF A DRIVE IS PUT ON 'LOAD' BY THE USER IT GETS REPORTED, IF THE USER SET 'WRITE PROT' IT GETS REPORTED. THE MESSAGES APPEAR AS FOLLOWING:

DRIVE 0 OFF LINE  
 DRIVE 1 WRITE PROTECTED  
 DRIVE 2 SIN  
 DRIVE 1 WRITE ENABLED  
 DRIVE 0 POWER LO  
 DRIVE 2 SEEK OK  
 DRIVE 0 POWER OK

NOTE THAT ONLY CHANGES IN STATUS ARE REPORTED. THESE CHANGES HAVE TO BE AFFECTED BY THE USER, IF ANY CHANGE IN STATUS IS NOT DETECTED AND REPORTED BY THE PROGRAM IT MIGHT IMPLY AN ERROR CONDITION.

9.7 SECTION 6 HEAD ALIGNMENT ROUTINE

PURPOSE: TO PROVIDE A FACILITY FOR HEAD ALIGNMENT, WITH DYNAMIC SELECTION OF THE UPPER OR LOWER HEAD.

DESCRIPTION: WHEN THE ROUTINE IS SELECTED THE FOLLOWING MESSAGE APPEARS:  
 SET SW0=0 FOR SURFACE 0, SW0=1 FOR SURFACE 1,  
 SET SW1=1 TO TEST CYL 64, SET SW1=0 TO TEST CYLINDER 105.  
 PUT ANY SW FROM 2-15 HI TO SELECT NEW DRIVE

THEN THE FOLLOWING QUESTION IS ASKED:  
 DRIVE? THE USER  
 SHOULD TYPE IN THE DRIVE NUMBER THAT HE WANTS TO SELECT. THE DRIVE NUMBER IS SUFFIXED WITH AN 'F' TO TEST RK-05F TYPE DRIVES.

TYPE=6  
 DRIVE=0<CR>

THE UPPER OR THE LOWER HEAD CAN BE SELECTED BY SWITCH 0. IF SURFACE 0 IS TO BE SELECTED, PUT SW 0 TO 0. IF SURFACE 1 IS TO BE SELECTED PUT SW 0 ON 1. THE HEADS MAY BE POSITIONED AT CYLINDER 64 OR CYLINDER 105. SET SW1=0 FOR CYLINDER 105, SW1=1 FOR CYLINDER 64. THE PROGRAM POSITIONS THE HEADS ON THE SELECTED CYLINDER AND CONTINUOUSLY READS FROM THE SURFACE SELECTED. IF THE USER WISHES TO SELECT THE OTHER HEAD OR CYLINDER IT CAN BE DYNAMICALLY DONE BY FLIPPING SW 0 OR SW 1. IF SOME OTHER DRIVE IS TO BE SELECTED, ANY SWITCH BETWEEN SW 2 AND SW 15 SHOULD BE PUT UP. THE QUESTION - DRIVE? IS ASKED AGAIN. THIS IS A CONTINUOUS ROUTINE, HENCE TO EXIT A HALT HAS TO BE DONE.

\*\*NOTE\*\* ALIGNMENT IS DONE WITH AN RK-05J CARTRIDGE SO IF AN F TYPE DRIVE IS SELECTED, CYLINDER 64 OF THE RK-05J IS CYLINDER 130 OF THE F DRIVE (EVEN DRIVE). CYLINDER 105 BECOMES CYLINDER 5 OF THE ODD DRIVE ON THE RK-05F.

9.8 SECTION 7 (DISK) POWER FAILURE (DURING WRITE) TEST  
 PURPOSE: THIS TEST CHECKS THAT DATA WRITTEN ON THE DISK IS NOT DESTROYED WHEN THE DISK SENSES A LOSS OF POWER (POWER FAILS) WHILE DOING A WRITE.  
 DESCRIPTION: UPON SELECTING THIS TEST, THE PROGRAM FINDS OUT THE FIRST AVAILABLE DRIVE AND INDICATES IT TO THE USER BY TYPING A MESSAGE:  
 DRIVE X X=DRIVE NUMBER 0,1,..,7  
 THEN IT PROCEEDS TO WRITE UNIQUE PATTERNS ON CYLINDERS 0 TO 15 (DECIMAL) OF THAT DRIVE. THE HEADS ARE THEN POSITIONED ON CYLINDER 10 AND THE USER IS ASKED TO DROP POWER ON THAT DRIVE:  
 DROP POWER  
 MEANWHILE WRITE IS BEING DONE ON CYLINDER 10. ON GETTING THE ABOVE MESSAGE THE USER SHOULD DROP THE POWER ON THAT DRIVE. ON SENSING A LOSS OF POWER, THE PROGRAM WILL ASK THE USER TO PUT THE POWER ON AGAIN:  
 POWER ON  
 ON RECEIVING THE ABOVE MESSAGE THE USER SHOULD PUT THE POWER ON. ON DETECTING POWER UP THE PROGRAM PROCEEDS TO CHECK THAT THE DATA WRITTEN ON CYLINDERS 0 TO 15 WAS INTACT. IF A WRITE CHECKS ERROR OCCURS (POSSIBLY MEANING THAT SOME OF THE DATA WAS DESTROYED DURING THE LOSS OF POWER) IT IS REPORTED AS FOLLOWING:  
 ERROR, ON POWER-UP, RKDA=XXXX  
 XXXX IS THE CONTENTS OF RKDA AT THE TIME OF ERROR.

THE PROGRAM DOES THE ABOVE POWER FAIL TEST

ON ALL DRIVES THAT ARE PRESENT, ONE AFTER THE OTHER IN A ROUND BOBBIN FASHION. EXIT IS THROUGH HALT.

9.9 SECTION SPECIAL

FOR THE BELOW EXAMPLES THE FOLLOWING FORMAT WILL BE USED.

THE ACTUAL TYPEOUT ; COMMENTS ON WHAT AND RESPONSE ; OCCURRED OR WHAT TO DO \*NOTES IF NECESSARY FOR CLARITY

ERROR EXAMPLE 1

```

FORMATTER-SURFACE VERIFIER      3
RK05 CONTROL PANEL TEST        4

TYPE=1
DRIVE NUMBERS ON SYTEM 1=0.
;TYPE 1 SELECTION
;DRIVE #0 SELECTED

IS THERE A SECOND SYSTEM?N
MOUNT PACK ON DRIVE #0
MAKE PACK WRITE ENABLE
PRESS CONTINUE WHEN DRIVE RDY
DRIVE WRITE PROTECTED.
DRIVE WRITE PROTECTED
DONE!
    RK11 UTILITY PACKAGE

```

```

INDEX      NAME      TYPE
COMPATIBILITY PACKAGE  0
OSCILLATING SEEK PACKAGE 1

```

ERROR EXAMPLE 2

RK11 UTILITY PACKAGE

```

INDEX      NAME      TYPE
COMPATIBILITY PACKAGE  0
OSCILLATING SEEK PACKAGE 1
FORMATTER-SURFACE VERIFIER 2
RK05 CONTROL PANEL TEST 3
RK05 CONTROL PANEL TEST #2 4
HEAD ALIGNMENT ROUTINE 5
POWER FAILURE (WRITE) TEST 6

```

```

TYPE=1
DRIVE NUMBERS ON SYTEM 1=0.

```

```

IS THERE A SECOND SYSTEM?N
MOUNT PACK ON DRIVE #0
MAKE PACK WRITE ENABLE
PRESS CONTINUE WHEN DRIVE
DRIVE NOT READY
DRIVE NOT READY
DRIVE NOT READY
DRIVE NOT READY
DRIVE NOT READY

```

```

;CONTINUE PRESSED BUT
;DRIVE NOT READY. IF UP
;TO SPEED ETC. AND MESSAGE
;OCCURRING - STATIC SHOULD BE

```



DRIVE NUMBERS ON SYTEM 1=0.

```

IS THERE A SECOND SYSTEM?N      ;SAME AS ABOVE BUT FUNCTION
MOUNT PACK ON DRIVE #0          ;WAS A CONTROL RESET
MAKE PACK WRITE ENABLE
PRESS CONTINUE WHEN DRIVE RDY   ;ALL COMMENTS ARE THE SAME
CONTROL RESET TIMED OUT        ;AS EXAMPLE 3
CONTROL RESET TIMED OUT
CONTROL RESET TIMED OUT
CONTROL RESET TIMED OUT
    
```

\*A SINGULAR OCCURANCE AS ABOVE IS NOT A PROBLEM AND WILL NOT EFFECT COMPATABILITY

ERROR EXAMPLE 5

THE BELOW ERRORS DO, ALWAYS, EFFECT COMPATABILITY. IN THE FIRST TYPE THE DRIVE IS DOWN INDICATING THAT (5) FIVE HARD OR SOFT ERRORS OCCURRED. THE TEST WILL CONTINUE AGAINST THE OTHER DRIVES BUT THERE IS A PROBLEM IN THIS DRIVE AND IT SHOULD BE CONSIDERED NON EXISTENT AS FAR AS COMPATABILITY GOES. THAT IS TO SAY IT IS NOT TESTED, THEREFORE NOT NECESSARILY COMPATABLE OR INCOMPATABLE.

RK11 UTILITY PACKAGE

INDEX	NAME	TYPE
0	COMPATIBILITY PACKAGE	0
1	OSCILLATING SEEK PACKAGE	1
2	FORMATTER-SURFACE VERIFIER	2
3	RK05 CONTROL PANEL TEST	3
4	RK05 CONTROL PANEL TEST #2	4
5	HEAD ALIGNMENT ROUTINE	5
6	POWER FAILURE (WRITE) TEST	6
7		7

TYPE=1

DRIVE NUMBERS ON SYTEM 1=0.

```

IS THERE A SECOND SYSTEM?N
MOUNT PACK ON DRIVE #0
MAKE PACK WRITE ENABLE
PRESS CONTINUE WHEN DRIVE RDY
5 ERRORS OCCURRED DRIVE DECLARED DOWN!! NOT TESTED I
DONE!
    
```

RK11 UTILITY PACKAGE

INDEX	NAME	TYPE
0	COMPATIBILITY PACKAGE	0
1		1

\*IN THE ABOVE CASE THE MESSAGE "3 SEEK INCOMPLETE ERRORS OCCURRED DRIVE DECLARED DOWN!! NOT TESTED!" MAY OCCUR IT IS THE SAME ERROR AS DESCRIBED ABOVE EXCEPT THAT IT IS CAUSED BY 3 SEEK ERRORS OCCURRING ON ONE DRIVE.

ERROR EXAMPLE 6



INDEX	NAME	TYPE
0	COMPATABILITY PACKAGE	0
1	OSCILLATING SEEK PACKAGE	1
2	FORMATTER-SURFACE VERIFIER	2
3	RK05 CONTROL PANEL TEST	3
4	RK05 CONTROL PANEL TEST ROUTINE	4
5	HEAD ALIGNMENT ROUTINE	5
6	POWER FAILURE (WRITE) TEST	6
7		7

TYPE=1  
 DRIVE NUMBERS ON SYSTEM 1=0:

IS THERE A SECOND SYSTEM?Y

DRIVE # =1  
 MOUNT PACK ON DRIVE #0  
 MAKE PACK WRITE ENABLE  
 PRESS CONTINUE WHEN DRIVE RDY  
 LOAD AND START ADDRESS 210 ON SYSTEM #2  
 AND TYPE THE BELOW WHEN ASKED FOR IT.  
 WORD  
 1=101000

WORD=000177

MOUNT PACK ON DRIVE #0  
 MAKE PACK WRITE ENABLE  
 PRESS CONTINUE WHEN DRIVE RDY  
 ERROR! DATA WRITTEN BY DRIVE 1 CANNOT BE READ.

ADDR=002764 EXPCTD=077400 RECVD=177000  
 ADDR=002764 EXPCTD=077400 RECVD=077600  
 ADDR=002764 EXPCTD=077400 RECVD=037600  
 ADDR=002764 EXPCTD=077400 RECVD=037600  
 ERROR! DATA WRITTEN BY DRIVE 1 CANNOT BE READ.  
 ADDR=007624 EXPCTD=077400 RECVD=177000  
 ERROR! DATA WRITTEN BY DRIVE 1 CANNOT BE READ.  
 ADDR=007633 EXPCTD=077400 RECVD=177000  
 ADDR=007633 EXPCTD=077400 RECVD=177000  
 DONE!

RK11 UTILITY PACKAGE

INDEX	NAME	TYPE
0	COMPATABILITY PACKAGE	0
1	OSCILLATING SEEK PACKAGE	1
2	FORMATTER-SURFACE VERIFIER	2
3	RK05 CONTROL PANEL TEST	3
4	RK05 CONTROL PANEL TEST #2	4
5	HEAD ALIGNMENT ROUTINE	5
6	POWER FAILURE (WRITE) TEST	6
7		7

TYPE=  
 THE ABOVE ERROR MESSAGE SHOWS A COMPATABILITY  
 PROBLEM. ALL ERRORS OCCURRED ON HEAD ONE OF  
 DRIVE 0 TRYING TO READ INFORMATION WRITTEN BY  
 DRIVE 1.

ERROR EXAMPLE 7

```

MOUNT PACK ON DRIVE #0
WAKKE PACK WRITE ENABLE
PRESS CONTINUE WHEN DRIVE RDY
ERROR! DATA WRITTEN BY DRIVE 1 CANNOT BE READ.
  ADDR=000367  EXPCID=077400  RECVD=077600
  ADDR=000367  EXPCID=077400  RECVD=037600
  ADDR=000367  EXPCID=077400  RECVD=037600
  ADDR=000367  EXPCID=077400  RECVD=037600
  ADDR=000367  EXPCID=077400  RECVD=037600
  ADDR=000367  EXPCID=077400  RECVD=037600
  ADDR=002564  EXPCID=077400  RECVD=077600
  ADDR=002564  EXPCID=077400  RECVD=037600
  ADDR=002564  EXPCID=077400  RECVD=037600
  ADDR=002564  EXPCID=077400  RECVD=037600
  ADDR=002564  EXPCID=077400  RECVD=037600
  ADDR=002764  EXPCID=077400  RECVD=077600
  ADDR=002764  EXPCID=077400  RECVD=037600
  ADDR=002764  EXPCID=077400  RECVD=037600
  ADDR=002764  EXPCID=077400  RECVD=037600
  ADDR=002764  EXPCID=077400  RECVD=037600
  ADDR=002767  EXPCID=077400  RECVD=177000
S ERRORS OCCURRED DRIVE DECLARED DOWN!! NOT TESTED!
DONE!

```

IN THE ABOVE EXAMPLE THE PROBLEM IS EXTREME. THE DRIVE WAS DECLARED DOWN DO TO CHECKSUM ERRORS. (TO SEE HOW THIS WAS DETERMINED SEE PARAGRAPH 9.7). NOTICE ALSO THE PROBLEM DID NOT START APPEARING UNTIL CYLINDER 7, AND WAS NOT FATAL UNTIL CYLINDER 57, AGAIN HEAD #1 WAS A COMMON FACTOR.

\*\*\*\*\*

9.10 COMPATIBILITY ERROR RECOVERY

ALTHOUGH A UTILITY PACKAGE IS NOT A TRUE DIAGNOSTIC IT IS OF BENEFIT TO THE USER TO AT TIMES, BE ABLE TO MODIFY THE PROGRAM TO RECIEVE MORE INFORMATION OR CONTROL PARAMETERS

1. THERE ARE TWO STRATEGICALLY PLACED NO-OPS, WHICH IF CHANGED TO HALTS, MAY BE OF HELP TO THE USER. ONE IS IN THE 'EXECUTE' ROUTINE WHICH ALLOWS THE USER TO EXAMINE THE DISK ADDRESS, BUS ADDRESS, WORD COUNT AND CONTROL REGISTERS IN TEMPORARY LOCATIONS JUST PRIOR TO LOADING AND EXECUTION. THE SECOND IS IN THE 'ERRCHK' ROUTINE WHICH ALLOW THE USER TO EXAMINE THE RKR REGISTER BEFORE THE PROGRAM CORRECTS ANY ERRORS WHICH WHICH MAY HAVE OCCURRED.
2. IF PLAGED BY CHECKSUM ERRORS AND THE USER WISHES MORE ERROR MAPING THEN HE MAY MODIFY THE MASK WORD AT LOCATION 'ERRCHK+2' TO ONLY RECOGNIZE HARD ERRORS.
3. TO INCREASE OR DECREASE THE NUMBER OF RETRYS ALLOWED

BEFORE A DRIVE IS DECLARED DOWN, GO TO THE 'MOUNT' ROUTINE, MODIFY THE SETUP OF LOCATIONS 'ECNT' AND 'CNTSIN' AND YOU HAVE IT!

4. IF THE USER DECIDES, SAY BECAUSE OF A LARGE NUMBER OF FAILURES, TO ALTER THE NUMBER OF PRINTOUTS PER SECTOR ON FAILURES (THE TYPE IN ERROR EXAMPLE 6 AND 7) HE MAY MODIFY THE SETUP OF 'CHKCNT' IN THE 'RDCHK' ROUTINE.

A FINAL LOOK; THE FOLLOWING SECTION SHOWS ALL PACKAGES CALLED IN SEQUENCE, NONE WITH ERRORS.  
RK11 UTILITY PACKAGE

INDEX	NAME	TYPE
0	COMPATABILITY PACKAGE	0
1	OSCILLATING SEEK PACKAGE	1
2	FORMATTER-SURFACE VERIFIER	2
3	RK05 CONTROL PANEL TEST	3
4	RK05 CONTROL PANEL TEST #2	4
5	HEAD ALIGNMENT ROUTINE	5
6	POWER FAILURE (WRITE) TEST	6
7		7

TYPE=0

RK11 UTILITY PACKAGE

INDEX	NAME	TYPE
0	COMPATABILITY PACKAGE	0
1	OSCILLATING SEEK PACKAGE	1
2	FORMATTER-SURFACE VERIFIER	2
3	RK05 CONTROL PANEL TEST	3
4	RK05 CONTROL PANEL TEST #2	4
5	HEAD ALIGNMENT ROUTINE	5
6	POWER FAILURE (WRITE) TEST	6
7		7

TYPE=1

DRIVE NUMBERS ON SYSTEM 1=0.1.3.

IS THERE A SECOND SYSTEM?N

- MOUNT PACK ON DRIVE #0
- MAKE PACK WRITE ENABLE
- PRESS CONTINUE WHEN DRIVE RDY
- MOUNT PACK ON DRIVE #1
- MAKE PACK WRITE ENABLE
- PRESS CONTINUE WHEN DRIVE RDY
- MOUNT PACK ON DRIVE #3
- MAKE PACK WRITE ENABLE
- PRESS CONTINUE WHEN DRIVE RDY
- MOUNT PACK ON DRIVE #0
- MAKE PACK WRITE ENABLE
- PRESS CONTINUE WHEN DRIVE RDY
- MOUNT PACK ON DRIVE #1
- MAKE PACK WRITE ENABLE
- PRESS CONTINUE WHEN DRIVE RDY
- MOUNT PACK ON DRIVE #3

MAKE PACK WRITE ENABLE  
PRESS CONTINUE WHEN DRIVE RDY  
DONE!

RK11 UTILITY PACKAGE

INDEX	NAME	TYPE
0	COMPATABILITY PACKAGE	0
1	OSCILLATING SEEK PACKAGE	1
2	FORMATTER-SURFACE VERIFIER	2
3	RK05 CONTROL PANEL TEST	3
4	RK05 CONTROL PANEL TEST #2	4
5	HEAD ALIGNMENT ROUTINE	5
6	POWER FAILURE (WRITE) TEST	6
7		7

TYPE=2  
OSCILLATING SEEK PACKAGE, WHICH DRIVE?0  
TOGGLE THE "FIRST CYLINDER ADDRESS" (OUTER LIMIT)  
INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST  
CYLINDER ADDRESS" (INNER LIMIT) INTO THE HIGH  
BYTE (BIT8-15), THEN PRESS CONTINUE.

RK11 UTILITY PACKAGE

INDEX	NAME	TYPE
0	COMPATABILITY PACKAGE	0
1	OSCILLATING SEEK PACKAGE	1
2	FORMATTER-SURFACE VERIFIER	2
3	RK05 CONTROL PANEL TEST	3
4	RK05 CONTROL PANEL TEST #2	4
5	HEAD ALIGNMENT ROUTINE	5
6	POWER FAILURE (WRITE) TEST	6
7		7

TYPE=3  
FORMATTER-SURFACE VERIFIER, WHICH DRIVE?0

PACK GOOD.

RK11 UTILITY PACKAGE

INDEX	NAME	TYPE
0	COMPATABILITY PACKAGE	0
1	OSCILLATING SEEK PACKAGE	1
2	FORMATTER-SURFACE VERIFIER	2
3	RK05 CONTROL PANEL TEST	3
4	RK05 CONTROL PANEL TEST #2	4
5	HEAD ALIGNMENT ROUTINE	5
6	POWER FAILURE (WRITE) TEST	6
7		7

TYPE=4  
RK05 CONTROL PANEL TEST, WHICH DRIVE?0  
MOUNT PACK ON DRIVE #0  
PLACE DRIVE IN RUN ; SHOULD SEE THE RUN,  
POWER, AND ON CYLINDER LAMPS LIGHT.  
MAKE DRIVE WRITE ENABLE PRESS CONTINUE

WRITE PROTECT THE DRIVE THEN PRESS CONTINUE



```

1      .TITLE MAINDEC-11-DZRKI-E
2      ;*COPYRIGHT (C) 1974,1977
3      ;*DIGITAL EQUIPMENT CORP.
4      ;*MAYNARD, MASS. 01754
5      ;*
6      ;*PROGRAM BY BOB COLLINS
7      ;*
8      ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
9      ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
10     ;*
11     000001      $TN=1
12     160000      $SWR=160000      ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYPQUT
13
14
15     ;*REVISED BY JIM KAPADIA
16     ;*REVISED BY TOM SAWYER FEB 27, 1976
17     ;*REVISED BY CHUCK HESS AUGUST, 1976
18
19     .SBTTL BASIC DEFINITIONS
20
21     ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
22     001100      STACK= 1100
23     .EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
24     .EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
25
26     ;*MISCELLANEOUS DEFINITIONS
27     000011      HT= 11      ;;CODE FOR HORIZONTAL TAB
28     000012      LF= 12      ;;CODE FOR LINE FEED
29     000015      CR= 15      ;;CODE FOR CARRIAGE RETURN
30     000200      CRLF= 200    ;;CODE FOR CARRIAGE RETURN-LINE FEED
31     177776      PS= 177776   ;;PROCESSOR STATUS WORD
32     .EQUIV PS,PSW
33     177774      STKLMT= 177774 ;;STACK LIMIT REGISTER
34     177772      PIRO= 177772  ;;PROGRAM INTERRUPT REQUEST REGISTER
35     177570      DSWR= 177570  ;;HARDWARE SWITCH REGISTER
36     177570      DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
37
38     ;*GENERAL PURPOSE REGISTER DEFINITIONS
39     000000      R0= %0      ;;GENERAL REGISTER
40     000001      R1= %1      ;;GENERAL REGISTER
41     000002      R2= %2      ;;GENERAL REGISTER
42     000003      R3= %3      ;;GENERAL REGISTER
43     000004      R4= %4      ;;GENERAL REGISTER
44     000005      R5= %5      ;;GENERAL REGISTER
45     000006      R6= %6      ;;GENERAL REGISTER
46     000007      R7= %7      ;;GENERAL REGISTER
47     000006      SP= %6      ;;STACK POINTER
48     000007      PC= %7      ;;PROGRAM COUNTER
49
50     ;*PRIORITY LEVEL DEFINITIONS
51     000000      PR0= 0      ;;PRIORITY LEVEL 0
52     000040      PR1= 40     ;;PRIORITY LEVEL 1
53     000100      PR2= 100    ;;PRIORITY LEVEL 2
54     000140      PR3= 140    ;;PRIORITY LEVEL 3
55     000200      PR4= 200    ;;PRIORITY LEVEL 4
56     000240      PR5= 240    ;;PRIORITY LEVEL 5
  
```

```

57     000300      PR6= 300    ;;PRIORITY LEVEL 6
58     000340      PR7= 340    ;;PRIORITY LEVEL 7
59
60     ;*SWITCH REGISTER SWITCH DEFINITIONS
61     100000      SW15= 100000
62     040000      SW14= 40000
63     020000      SW13= 20000
64     010000      SW12= 10000
65     004000      SW11= 4000
66     002000      SW10= 2000
67     001000      SW09= 1000
68     000400      SW08= 400
69     000200      SW07= 200
70     000100      SW06= 100
71     000040      SW05= 40
72     000020      SW04= 20
73     000010      SW03= 10
74     000004      SW02= 4
75     000002      SW01= 2
76     000001      SW00= 1
77     .EQUIV SW09,SW9
78     .EQUIV SW08,SW8
79     .EQUIV SW07,SW7
80     .EQUIV SW06,SW6
81     .EQUIV SW05,SW5
82     .EQUIV SW04,SW4
83     .EQUIV SW03,SW3
84     .EQUIV SW02,SW2
85     .EQUIV SW01,SW1
86     .EQUIV SW00,SW0
87
88     ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
89     100000      BIT15= 100000
90     040000      BIT14= 40000
91     020000      BIT13= 20000
92     010000      BIT12= 10000
93     004000      BIT11= 4000
94     002000      BIT10= 2000
95     001000      BIT09= 1000
96     000400      BIT08= 400
97     000200      BIT07= 200
98     000100      BIT06= 100
99     000040      BIT05= 40
100    000020      BIT04= 20
101    000010      BIT03= 10
102    000004      BIT02= 4
103    000002      BIT01= 2
104    000001      BIT00= 1
105    .EQUIV BIT09,BIT9
106    .EQUIV BIT08,BIT8
107    .EQUIV BIT07,BIT7
108    .EQUIV BIT06,BIT6
109    .EQUIV BIT05,BIT5
110    .EQUIV BIT04,BIT4
111    .EQUIV BIT03,BIT3
112    .EQUIV BIT02,BIT2
  
```

```

113 .EQUIV BIT01,BIT1
114 .EQUIV BIT00,BIT0
115
116 ;*BASIC "CPU" TRAP VECTOR ADDRESSES
117 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
118 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
119 TBITVEC=14 ;: "T" BIT
120 TRTVEC= 14 ;:TRACE TRAP
121 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
122 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
123 PWRVEC= 24 ;:POWER FAIL
124 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
125 TRAPVEC=34 ;:"TRAP" TRAP
126 TKVEC= 60 ;:TTY KEYBOARD VECTOR
127 TPVEC= 64 ;:TTY PRINTER VECTOR
128 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
129 .SBTTL TRAP CATCHER
130
131 .=0
132 ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
133 ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
134 ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
135 .=174
136 000174 000000 DISPREG: .WORD 0 ;:SOFTWARE DISPLAY REGISTER
137 000176 000000 SWREG: .WORD 0 ;:SOFTWARE SWITCH REGISTER
138 .SBTTL STARTING ADDRESS(ES)
139 000200 000137 001434 JMP @#STARTR ;:JUMP TO STARTING ADDRESS OF PROGRAM
140 000210 .=210
141 000210 112737 000377 001312 MOVB #377,@#MODE
142 000216 000137 001440 JMP @#START
143 .SBTTL ACT11 HOOKS
144
145 ;:*****
146 ;:HOOKS REQUIRED BY ACT11
147 000222 $SVPC=. ;:SAVE PC
148 000046 .=46
149 000046 001400 $ENDAD ;:1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
150 000052 .=52
151 000052 000000 .WORD 0 ;:2)SET LOC.52 TO ZERO
152 000222 .=$SVPC ;: RESTORE PC
    
```

```

153 .SBTTL COMMON TAGS
154
155 ;:*****
156 ;:THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
157 ;:USED IN THE PROGRAM.
158
159 .=1100
160 001100 $CMTAG: .WORD 0 ;:START OF COMMON TAGS.
161 001100 000000 $PASS: .WORD 0 ;:CONTAINS PASS COUNT
162 001102 000 $STNM: .BYTE 0 ;:CONTAINS THE TEST NUMBER
163 001103 000 $ERFLG: .BYTE 0 ;:CONTAINS ERROR FLAG
164 001104 000000 $ICNT: .WORD 0 ;:CONTAINS SUBTEST ITERATION COUNT
165 001106 000000 $LPADR: .WORD 0 ;:CONTAINS SCOPE LOOP ADDRESS
166 001110 000000 $LPERR: .WORD 0 ;:CONTAINS SCOPE RETURN FOR ERRORS
167 001112 000000 $ERTTL: .WORD 0 ;:CONTAINS TOTAL ERRORS DETECTED
168 001114 000 $ITEMB: .BYTE 0 ;:CONTAINS ITEM CONTROL BYTE
169 001115 001 $ERRMAX: .BYTE 1 ;:CONTAINS MAX. ERRORS PER TEST
170 001116 000000 $ERRPC: .WORD 0 ;:CONTAINS PC OF LAST ERROR INSTRUCTION
171 001120 000000 $GDADR: .WORD 0 ;:CONTAINS ADDRESS OF 'GOOD' DATA
172 001122 000000 $BDADR: .WORD 0 ;:CONTAINS ADDRESS OF 'BAD' DATA
173 001124 000000 $GGDAT: .WORD 0 ;:CONTAINS 'GOOD' DATA
174 001126 000000 $BDDAT: .WORD 0 ;:CONTAINS 'BAD' DATA
175 001130 000000 .WORD 0 ;:RESERVED--NOT TO BE USED
176 001132 000000 .WORD 0
177 001134 000 $AUTOB: .BYTE 0 ;:AUTOMATIC MODE INDICATOR
178 001135 000 $INTAG: .BYTE 0 ;:INTERRUPT MODE INDICATOR
179 001136 000000 .WORD 0
180 001140 177570 SWR: .WORD DSWR ;:ADDRESS OF SWITCH REGISTER
181 001142 177570 DISPLAY: .WORD DDISP ;:ADDRESS OF DISPLAY REGISTER
182 001144 177560 STKS: 177560 ;:TTY KBD STATUS
183 001146 177562 STKB: 177562 ;:TTY KBD BUFFER
184 001150 177564 STPS: 177564 ;:TTY PRINTER STATUS REG. ADDRESS
185 001152 177566 STPB: 177566 ;:TTY PRINTER BUFFER REG. ADDRESS
186 001154 000 $NULL: .BYTE 0 ;:CONTAINS NULL CHARACTER FOR FILLS
187 001155 002 $FILLS: .BYTE 2 ;:CONTAINS # OF FILLER CHARACTERS REQUIRED
188 001156 012 $FILL: .BYTE 12 ;:INSERT FILL CHARS. AFTER A "LINE FEED"
189 001157 000 $TPFLG: .BYTE 0 ;:"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
190 001160 077 $QUES: .ASCII ?/ ;:QUESTION MARK
191 001161 015 $CRFL: .ASCII <15> ;:CARRIAGE RETURN
192 001162 000012 $LF: .ASCIIZ <12> ;:LINE FEED
193 ;:*****
194 001164 000000 DRACTV: .WORD 0 ;:ACTIVE DRIVE WORD
195
196 001166 000010 LOGA: .BLKW 10 ;:TABLE OF ACTIVE DRIVE WORDS
197
198 001206 152525 DRVO: .WORD 152525 ;:TABLE OF PATTERN # TO DRIVE #'S
199 001210 077777 .WORD 077777
200 001212 000000 .WORD 000000
201 001214 012345 .WORD 012345
202 001216 125252 .WORD 125252
203 001220 000001 .WORD 000001
204 001222 177777 .WORD 177777
205 001224 154320 .WORD 154320
206
207 001226 000010 RDTBL: .BLKW 10 ;:TABLE OF READ ADDRESS
208 001246 000004 PASTBL: .BLKW 4 ;:TABLE OF PARAMETERS FOR SYSTEM #2
    
```

209					
210	001256	377	MSKTBL: .BYTE	377	:TABLE OF CYLINDER BASE FOR AUTO MODE
211	001257	177	.BYTE	177	
212	001260	077	.BYTE	077	
213	001261	037	.BYTE	037	
214	001262	017	.BYTE	017	
215	001263	007	.BYTE	007	
216	001264	003	.BYTE	003	
217	001265	001	.BYTE	001	
218					
219	001266	000	BASE: .BYTE	0	:CYL 0 BASE CYLINDER ADDRESS
220	001267	050	.BYTE	50	:CYL 40 BASE CYLINDER ADDRESS
221	001270	120	.BYTE	120	:CYL 80 BASE CYLINDER ADDRESS
222	001271	170	.BYTE	170	:CYL 120 BASE CYLINDER ADDRESS
223	001272	240	.BYTE	240	:CYL 160 BASE CYLINDER ADDRESS
224	001273	303	.BYTE	303	:CYL 195 BASE CYLINDER ADDRESS
225					
226	001274	000011	CYLTBL: .BLKB	11	:TABLE OF SELECTED BASES
227					
228	001305	000	SECTBL: .BYTE	0	:SECTOR 0
229	001306	004	.BYTE	4	:SECTOR 4
230	001307	007	.BYTE	7	:SECTOR 7
231	001310	013	.BYTE	13	:SECTOR 12
232					
233	001311	000	DRCNT1: .BYTE	0	:COUNT OF NUMBER OF DRIVES ON SYS. 1
234	001312	000	MODE: .BYTE	0	:IF -1 START 210 SELECTED
235	001313	000	PRONUM: .BYTE	0	:IF 0 1 PROCESSOR SELECTED
236	001314	000	DRIVE: .BYTE	0	:DRIVE # UNDER TEST (MAN+AUTO MODE)
237	001315	000	CYLBAS: .BYTE	0	:BASE SELECTED (MANUAL MODE)
238	001316	000	COMND: .BYTE	0	:IF 0 WRITE COMMAND
239	001317	000	WRTNBY: .BYTE	0	:DRIVE WHICH DID WRITE (READ OPERATION)
240	001320	000	HDRFLG: .BYTE	0	:FLAG FOR ONE HEADER PRINTOUT
241	001321	000	ECNT: .BYTE	0	:ERROR COUNTER
242	001322	000	CNTSIN: .BYTE	0	:SEEK INCOM. COUNTER
243	001323	000	TIMR2: .BYTE	0	:SECOND PASS TIMER
244	001324	000	IDEX: .BYTE	0	:CURRENT INDEX #
245	001325	000	STFLG: .BYTE	0	
246	001325	000	OSPFLG: .BYTE	0	
247					
248		001330	.EVEN		
249					
250	001330	000000	KYTEMP: .WORD	0	:TEMP. KEYBOARD BUFFER
251	001332	000000	CONTRL: .WORD	0	:TEMP. CONTROL+STATUS WORD
252	001334	000000	DSKADR: .WORD	0	:TEMP. DISK ADDRESS WORD
253	001336	000000	BUSADR: .WORD	0	:TEMP. BUS ADDRESS WORD
254	001340	000000	WRDCNT: .WORD	0	:TEMP. WORD COUNT
255	001342	172000	CYLCNT: .WORD	-6000	:WORD COUNT OF 1 CYLINDER
256	001344	177400	SECCNT: .WORD	-400	:WORD COUNT OF 1 SECTOR
257	001346	000000	TIMR: .WORD	0	:TIMER FOR OPERATIONS
258	001350	000000	CHKCNT: .WORD	0	:NUMBER OF ERROR PRINTOUTS
259	001352	000000	DSKTMP: .WORD	0	:SAVE OF CURRENT DISK #
260	001354	004003	WRITCS: .WORD	4003	:IBA+WRITE+GO
261	001356	000005	READCS: .WORD	5	:READ+GO
262	001360	000000	ERRFLG: .WORD	0	:ERROR FLAG INHIBIT ADDRESS CHANGE
263	001362	000000	PATRN: .WORD	0	:DATA PATTERN
264	001364	177400	RKDS: .WORD	177400	

265	001366	177402	RKER: .WORD	177402
266	001370	177404	RKCS: .WORD	177404
267	001372	177406	RKWC: .WORD	177406
268	001374	177410	RKBA: .WORD	177410
269	001376	177412	RKDA: .WORD	177412
270	001400	000000	SENDAD: .WORD	0
271	001402	000000	SEEKI: .WORD	0
272	001404	000000	SEEKO: .WORD	0
273				
274		105212	LFLF=	105212
275	001406	013700	BA: .WORD	0
276	001410	000000	DA: .WORD	0
277	001412	000000	WC: .WORD	0
278	001414	013702	RBA: .WORD	0
279	001416	000000	RWC: .WORD	0
280	001420	000000	EXTR: .WORD	0
281	001422	000000	ERRWF: .WORD	0
282	001424	000000	ERRRF: .WORD	0
283	001426	000000	ERRRFC: .WORD	0
284	001430	000000	ERRWCH: .WORD	0
285	001432	000000	ERRWCS: .WORD	0
286				
287			:BIT DEFINITIONS	
288				
289	010000		DPL=BIT12	
290	000100		RWS=BIT8	
291	000040		WPS=BITS	
292	001000		SIN=BIT9	



```

293 .SBTTL ERROR POINTER TABLE
294
295 ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
296 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
297 ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
298 ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
299 ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
300
301 ;* EM ;;POINTS TO THE ERROR MESSAGE
302 ;* DH ;;POINTS TO THE DATA HEADER
303 ;* DT ;;POINTS TO THE DATA
304 ;* DF ;;POINTS TO THE DATA FORMAT
305
306
307 001434 $ERRTB:
308 ;*****
309
310 ;THE ERROR TABLE IS UNUSED IN THIS PROGRAM
311
312 ;*****
313

```

```

314 001434 105037 001312 STARTR: CLR8 @#MODE
315 001440 000005 START: RESET ;CLEAR THE BUS
316 001442 012706 001100 MOV #STACK,SP
317 001446 012746 000000 MOV #0,-(SP) ;SET UP STACK FOR PSW=0
318 001452 012746 001460 MOV #2$,-(SP) ;RETURN FOR RTI
319 001456 000002 RTI
320 001460 000240 2$: NOP
321 .SBTTL INITIALIZE THE COMMON TAGS
322 ;:CLEAR THE COMMON TAGS ($CMTAG) AREA
323 001462 012706 001100 MOV #SCMTAG,R6 ;:FIRST LOCATION TO BE CLEARED
324 001466 005026 CLR (R6)+ ;:CLEAR MEMORY LOCATION
325 001470 022706 001140 CMP #SWR,R6 ;:DONE?
326 001474 001374 BNE -.6 ;:LOOP BACK IF NO
327 001476 012706 001100 MOV #STACK,SP ;:SETUP THE STACK POINTER
328 ;:INITIALIZE A FEW VECTORS
329 001502 012737 024230 000034 MOV #STRAP,@#TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
330 001510 012737 000340 000036 MOV #340,@#TRAPVEC+2;LEVEL 7
331 001516 012737 024310 000024 MOV #SPWRDN,@#PWRVEC ;:POWER FAILURE VECTOR
332 001524 012737 000340 000026 MOV #340,@#PWRVEC+2 ;:LEVEL 7
333 ;:SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
334 ;:EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
335 001532 013746 000004 MOV @#ERRVEC,-(SP) ;:SAVE ERROR VECTOR
336 001536 012737 001572 000004 MOV #64$,@#ERRVEC ;:SET UP ERROR VECTOR
337 001544 012737 177570 001140 MOV #DSWR,SWR ;:SETUP FOR A HARDWARE SWICH REGISTER
338 001552 012737 177570 001142 MOV #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
339 001560 022777 177777 177352 CMP #-1,@SWR ;:TRY TO REFERENCE HARDWARE SWR
340 001566 001012 BNE 66$ ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
341 ;:AND THE HARDWARE SWR IS NOT = -1
342 001570 000403 BR 65$ ;:BRANCH IF NO TIMEOUT
343 001572 012716 001600 64$: MOV #85$,(SP) ;:SET UP FOR TRAP RETURN
344 001576 000002 RTI
345 001600 012737 000176 001140 65$: MOV #SWREG,SWR ;:POINT TO SOFTWARE SWR
346 001606 012737 000174 001142 66$: MOV #DISPREG,DISPLAY ;:SOFTWARE SWITCH REG SELECTED?
347 001614 012637 000004 (SP)+,@#ERRVEC ;:RESTORE ERROR VECTOR
348
349 001620 004737 022746 JSR PC,$TKINT ;:INITIALIZE THE TTY INTERRUPT HANDLER
350 .SBTTL TYPE PROGRAM NAME
351 ;:TYPE THE NAME OF THE PROGRAM IF FIRST PASS
352 001624 005227 177777 INC #-1 ;:FIRST TIME?
353 001630 001045 BNE 67$ ;:BRANCH IF NO
354 001632 104401 001670 TYPE ,68$ ;:TYPE ASCIZ STRING
355 ;:TYPE ASCIZ STRING
356 001636 005737 000042 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
357 001642 001006 TST @#42 ;:ARE WE RUNNING UNDER XXDP/ACT?
358 001644 023727 001140 000176 BNE 69$ ;:BRANCH IF YES
359 001652 001005 CMP SWR,#SWREG ;:SOFTWARE SWITCH REG SELECTED?
360 001654 104405 BNE 70$ ;:BRANCH IF NO
361 001656 000403 GTSWR ;:GET SOFT-SWR SETTINGS
362 001660 112737 000001 001134 69$: BR 70$
363 001666 MOVB #1,$AUTOB ;:SET AUTO-MODE INDICATOR
364 001666 000426 BR 67$
365 ;:GET OVER THE ASCIZ
366 001744 ;:68$: .ASCIZ <RFL>/RK11 UTILITY PACKAGE/<15><12>/MAINDEC-11-DZRKI-E/<CRLF>
367 001744 105737 001312 67$:
368 001750 100002 TSTB @#MODE
369 001752 000137 003452 BPL 1$
JMP @#SECOND

```

```

370 001756 105737 001325 1$: TSTB STFLG ;PRINT ONLY THE FIRST TIME
371 001762 001402 BEQ 105
372 001764 000137 002554 JMP TABLTY
373 001770 105237 001325 10$: INCB STFLG
374 001774 105037 001326 STRT1: CLRB DSPFLG
375 002000 104401 002006 TYPE ,65$ ;:TYPE ASCIZ STRING
376 002004 000423 BR 64$ ;:GET OVER THE ASCIZ
;:65$: .ASCIZ <15><12><15><12>/ NAME TYPE/
64$:
378 002054 TYPE ,67$ ;:TYPE ASCIZ STRING
379 002054 104401 002062 BR 66$ ;:GET OVER THE ASCIZ
380 002060 000421 ;:67$: .ASCIZ <15><12>/INDEX 0/
381
382 002124 66$: TYPE ,69$ ;:TYPE ASCIZ STRING
383 002124 104401 002132 BR 68$ ;:GET OVER THE ASCIZ
384 002130 000421 ;:69$: .ASCIZ <15><12>/COMPATIBILITY PACKAGE 1/
385
386 002174 68$: TYPE ,71$ ;:TYPE ASCIZ STRING
387 002174 104401 002202 BR 70$ ;:GET OVER THE ASCIZ
388 002200 000421 ;:71$: .ASCIZ <15><12>/OSCILLATING SEEK PACKAGE 2/
389
390 002244 70$: TYPE ,73$ ;:TYPE ASCIZ STRING
391 002244 104401 002252 BR 72$ ;:GET OVER THE ASCIZ
392 002250 000421 ;:73$: .ASCIZ <15><12>/FORMATTER-SURFACE VERIFIER 3/
393
394 002314 72$: TYPE ,75$ ;:TYPE ASCIZ STRING
395 002314 104401 002322 BR 74$ ;:GET OVER THE ASCIZ
396 002320 000421 ;:75$: .ASCIZ <15><12>/RK05 CONTROL PANEL TEST 4/
397
398 002364 74$: TYPE ,77$ ;:TYPE ASCIZ STRING
399 002364 104401 002372 BR 76$ ;:GET OVER THE ASCIZ
400 002370 000421 ;:77$: .ASCIZ <15><12>/RK05 CONTROL PANEL TEST #2 5/
401
402 002434 76$: TYPE ,79$ ;:TYPE ASCIZ STRING
403 002434 104401 002442 BR 78$ ;:GET OVER THE ASCIZ
404 002440 000421 ;:79$: .ASCIZ <15><12>/HEAD ALIGNMENT ROUTINE 6/
405
406 002504 78$: TYPE ,81$ ;:TYPE ASCIZ STRING
407 002504 104401 002512 BR 80$ ;:GET OVER THE ASCIZ
408 002510 000421 ;:81$: .ASCIZ <15><12>/POWER FAILURE (WRITE) TEST 7/
409
410 002554 80$:
411 002554 TABLTY:
412 002554 104401 002562 TYPE ,65$ ;:TYPE ASCIZ STRING
413 002560 000405 BR 64$ ;:GET OVER THE ASCIZ
414 ;:65$: .ASCIZ <15><12><15><12>/TYPE=/
415 64$:
416 002574 104411 RDOCT ;GET THE TEST NUMBER FROM THE OPERATOR
417 002576 012600 MOV (SP)+,R0 ;STORE IT IN R0
418 002600 022700 CMP #10,R0 ;VALID NUMBER ?
419 002604 003403 BLE NG ;BR IF NOT
420 002606 006100 ROL R0 ;ALIGN THE NUMBER FOR DISPATCHING
421 002610 000170 002622 JMP @BEGIN(R0) ;GO TO THE SELECTED TEST
422 002614 104401 001160 NG: TYPE ,80$ ;:TYPE ASCIZ STRING
423 002620 000755 BR TABLTY ;:GET OVER THE ASCIZ
424 ;:80$: .ASCIZ <15><12><15><12>/TYPE=?
425 ;:80$: .ASCIZ <15><12><15><12>/TYPE=?

;TEST ENTRY DISPATCH TABLE
  
```

```

426
427 002622 001774 BEGIN: STRT1
428 002624 002642 SECT.3
429 002626 010370 SECT.2
430 002630 012002 SECT.1
431 002632 013772 SECT.0
432 002634 017300 SECT.4
433 002636 020634 SECT.5
434 002640 021544 SECT.6
435
436
437 .SBTTL COMPATIBILITY TEST
438
439 ;ROUTINE TO PICK UP THE DRIVE NUMBER TO BE TESTED
440 ;ON SYSTEM 1.
441
442 002642 000240 SECT.3: NOP ;NO-OP
443 002644 AUTSL2:
444 002644 104401 002652 TYPE ,65$ ;:TYPE ASCIZ STRING
445 002650 000415 BR 64$ ;:GET OVER THE ASCIZ
;:65$: .ASCIZ <15><12>/TERMINATE WITH ',<CR>' /
64$:
447 002704 TYPE ,67$ ;:TYPE ASCIZ STRING
448 002704 104401 002712 BR 66$ ;:GET OVER THE ASCIZ
449 002710 000417 ;:67$: .ASCIZ <15><12>/DRIVE NUMBERS ON SYSTEM 1=/
66$:
451 002750 RDLIN
452 002750 104410 MOV (SP)+,R0 ;PICK UP THE ADDRESS OF THE INPUT BUFFER
453 002752 012600 MOV #LOGA,R1 ;GET THE ADDRESS OF THE LOGICAL UNIT TBL.
454 002754 012701 001166 CLRB @#DRCNT1 ;CLEAR THE DRIVE COUNTER
455 002760 105037 001311 CLR @#KYTEMP ;CLEAR TEMP
456 002764 005037 001330 1$: MOV (R0)+,@#KYTEMP ;GET THE FIRST DRIVE #
457 002770 112037 001330 CMPB #54,@#KYTEMP ;IS IT A COMMA THAT WAS TYPED?
458 002774 122737 000054 001330 BEQ 1$ ;IF YES GO BACK
459 003002 001770 SUB #60,@#KYTEMP ;MAKE ASCII A DRIVE #
460 003004 162737 000060 001330 BMI 2$ ;IF RESULT NEGATIVE BRANCH
461 003012 104403 JSR PC,STORE ;IF RESULT POSITIVE JUMP
462 003014 004737 004072 BR 1$ ;AFTER STORING GET NEXT #
463 003020 000761 2$: CMPB #56,-(R0) ;WAS NEGATIVE RESULT A TERMINATOR?
464 003022 122740 000056 BEQ 3$ ;IF YES BRANCH
465 003026 001402 JSR PC,ILEGAL ;IF NO BAD CHARACTER JUMP
466 003030 004737 004142 3$: CMP #DRVO,R1 ;IS THE TABLE FULL
467 003034 022701 001208 BEQ SECSYS ;IF YES BRANCH
468 003040 001403 MOV #100000,(R1)+ ;IF NO FILL TABLE WITH DOWN INDICATOR.
469 003042 012721 100000 BR 3$ ;GO BACK AND CHECK
470 003046 000772
471
472 ;ROUTINE TO DETERMINE IF THERE IS A SECOND
473 ;SYSTEM AND IF SO TO GET THE NUMBER OF THE
474 ;DRIVE ON THIS SYSTEM
475
476 003050 SECSYS:
477 003050 104401 003056 TYPE ,65$ ;:TYPE ASCIZ STRING
478 003054 000416 BR 64$ ;:GET OVER THE ASCIZ
;:65$: .ASCIZ <15><12>/IS THERE A SECOND SYSTEM?/
64$:
479
480 003112 NOP
481 003112 000240 ;***
  
```

```

482 003114 104407          RDCHR          ;READ A CHARACTER
483 003116 012637 001330  MOV          (SP)+,@#KYTEMP ;GET THE RESPONSE
484 003122 104401 001330  TYPE          ,KYTEMP      ;ECHO
485 003126 022737 000131 001330  CMP          #131,@#KYTEMP ;WAS IT A "Y" (FOR YES)?
486 003134 001411          BEQ          PRO2          ;IF YES BRANCH TO PROCESSOR 2
487 003136 022737 000116 001330  CMP          #116,@#KYTEMP ;WAS RESPONSE LEGAL (N FOR NO)?
488 003144 001460          BEQ          PRO1          ;IF LEGAL BRANCH
489 003146 104401 003154  TYPE          ,67$         ;TYPE ASCIZ STRING
490 003152 000401          BR           66$          ;GET OVER THE ASCIZ
491          ;:67$: .ASCIZ  /?/
492          66$:
493 003156          BR           SECSYS        ;GO BACK ASK AGAIN
494 003160 152737 000377 001313  PRO2: BISB    #377,@#PRONUM ;SET FLAG TWO PROCESSORS
495 003166 000240          NOP
496 003170 104401 003176  TYPE          ,65$         ;TYPE ASCIZ STRING
497 003174 000406          BR           64$          ;GET OVER THE ASCIZ
498          ;:65$: .ASCIZ <15><12>/DRIVE # =/
499          64$:
500 003212          RDCHR          ;READ A CHARACTER
501 003214 104407 012637 001330  MOV          (SP)+,@#KYTEMP ;PICK UP THE RESPONSE
502 003220 104401 001330  TYPE          ,KYTEMP      ;ECHO
503 003224 162737 000060 001330  SUB          #60,@#KYTEMP   ;MAKE IT A NUMBER
504 003232 100420          BMI          BADINP       ;IF NOT A NUMBER, BRANCH
505 003234 022737 000010 001330  CMP          #10,@#KYTEMP   ;IS IT A LEGAL #?
506 003242 003414          BLE          BADINP       ;IF NO BRANCH
507 003244 000337 001330  SWAB        @#KYTEMP       ;GET THE DRIVE # TO THE HIGH BYTE
508 003250 052737 040000 001330  BIS         #BIT14,@#KYTEMP ;SET THE SECOND SYSTEM BIT
509 003256 113705 001311  MOVVB       @#DRCNT1,R5    ;GET THE DRIVE COUNT
510 003262 006105          ROL         R5             ;MAKE IT AN INDEX
511 003264 013765 001330 001166  MOV          @#KYTEMP,LOGA(R5) ;STORE THE SYSTEM #2 WORD
512 003272 000407          BR           GO           ;GO DO THE TEST
513          BADINP:
514 003274 104401 003302  TYPE          ,65$         ;TYPE ASCIZ STRING
515 003300 000401          BR           64$          ;GET OVER THE ASCIZ
516          ;:65$: .ASCIZ  /?/
517          64$:
518 003304          BR           PRO2          ;GO BACK ASK AGAIN
519 003306 105037 001313  PRO1: CLRB    @#PRONUM     ;CLEAR THE FLAG ONE PROCESSOR
520
521          ;THIS IS THE ACTUAL PROGRAM
522
523 003312 012700 001166  GO:  MOV      #LOGA,R0      ;GET THE TABLE ADDRESS TO R0
524 003316 105037 001324  CLRB     @#INDEX          ;CLRB THE INDEX
525 003322 000405          BR         GO2           ;BRANCH AROUND INCREMENT ROUTINE
526
527 003324 062700 000002  GO1: ADD      #2,R0         ;INDEX THRU THE TABLE
528 003330 022700 001206  CMP      #DRVO,R0        ;DONE?
529 003334 001414          BEQ      EXIT           ;IF YES GET OUT
530 003336 005710  GO2: TST     (R0)         ;IS THE DRIVE ACTIVE
531 003340 100001          BPL     GO3            ;IF YES BRANCH
532 003342 000770          BR      GO1            ;NO TRY THE NEXT ONE
533 003344 011037 001164  GO3: MOV     (R0),@#DRACTV ;PICK UP THE ACTIVE DRIVE WORD
534 003350 004737 004174  JSR     PC,CYCLE         ;CALL CYCLE (PICK UP DRIVE #, CYL BASE, AND CALL MOUNT)
535 003354 004737 005104  JSR     PC,WRLINK        ;CALL WRITE LINK TO LOAD REGISTERS FOR WRITE
536 003360 004737 005632  JSR     PC,RDLINK        ;CALL READ LINK TO LOAD REGISTERS FOR READ
537 003364 000757          BR      GO1            ;GO GET NEXT DRIVE
    
```

```

538 003366 012700 001166  EXIT: MOV      #LOGA,R0
539 003372 022700 001206  EXTFR2: CMP     #DRVO,R0
540 003376 001414          BEQ     EXITX
541 003400 032710 040000  BIT     #BIT14,(R0)
542 003404 001011          BNE    EXITX
543 003406 005720          TST    (R0)+
544 003410 100770          BMI    EXTFR2
545 003412 014001          MOV    -(R0),R1
546 003414 004737 004250  JSR     PC,DO2
547 003420 004737 005632  JSR     PC,RDLINK
548 003424 005720          TST    (R0)+
549 003426 000761          BR     EXTFR2
550 003430
551 003430 104401 003436  EXITX: TYPE    ,65$         ;TYPE ASCIZ STRING
552 003434 000404          BR     64$             ;GET OVER THE ASCIZ
553          ;:65$: .ASCIZ <15><12>/DONE!/
554          64$:
555 003446 000137 001440  JMP     @#START         ;RESTART
556
557          ;THIS IS PROCESSOR #2 CODE. THIS ROUTINE ASKS FOR AND UNPACKS THE CONTROL
558          ;WORDS FROM THE FIRST PROCESSOR.
559
560          SECOND:
561 003452          TYPE    ,65$         ;TYPE ASCIZ STRING
562 003456 104401 003460  BR     64$             ;GET OVER THE ASCIZ
563          ;:65$: .ASCIZ <15><12>/COMPATIBILITY-SYSTEM#2/
564          64$:
565 003512 012704 000200  MOV     #200,R4         ;SET UP MASK BIT IN R4
566 003516 012703 000001  MOV     #1,R3          ;SET UP WORD COUNTER IN R3
567 003522 012702 001166  MOV     #LOGA,R2        ;GET THE TABLE ADDRESS
568 003526          INSYS2:
569 003526 104401 003534  TYPE    ,65$         ;TYPE ASCIZ STRING
570 003532 000405          BR     64$             ;GET OVER THE ASCIZ
571          ;:65$: .ASCIZ <15><12>/WORD /
572          64$:
573 003546          NOP
574 003550 010346  MOV     R3,-(SP)        ;***
575 003552 104403  TYPOS   .BYTE 6         ;GET READY TO TYPE WORD COUNTER
576 003554          .BYTE 0
577 003555          TYPE    ,67$         ;TYPE ASCIZ STRING
578 003556 104401 003564  BR     66$             ;GET OVER THE ASCIZ
579 003562 000401          ;:67$: .ASCIZ  /=/
580          66$:
581 003566          RDOCT
582 003566 104411  MOV     (SP)+,R0        ;PICK UP THE OCTAL WORD
583 003570 012600  MOVVB   R0,R1          ;GET THE FIRST DRIVE TO R1
584 003572 110001  SWAB    R0             ;GET THE SECOND DRIVE TO R0 LOW BYTE
585 003574 000300  BIC     #177400,R0     ;CLEAR THE UNUSED BITS
586 003576 042700 177400  BIC     #177400,R1     ;CLEAR THE UNUSED BITS
587 003602 042701 177400  ROR     R0             ;ROTATE RIGHT R0
588 003606 006000  ROR     R0             ;ROTATE RIGHT R0
589 003610 103003  BCC     2$            ;IF CARRY IS CLEAR BRANCH
590 003612 052700 000200  BIS     #BIT7,R0       ;SET DOWN BIT IF CARRY SET
591 003616 000241  CLC
592 003620 006001  ROR     R1             ;AND CLEAR THE CARRY BIT
593 003622 103002  BCC     3$            ;NOW DO THE SAME FOR R1
                    ;IF NO ERROR, BRANCH
    
```

```

594 003624 052701 000200      BIS      #BIT7,R1      ;IF ERROR SET THE BIT
595 003630 110162 000001      MOVVB   R1,1(R2)     ;SET DRIVE FIRST IN TABLE
596 003634 004737 004030      JSR     PC,MASKER   ;CALL THE MASK CONTROL SUBROUTINE
597 003640 110062 000001      MOVVB   R0,1(R2)     ;SET DRIVE SECOND IN TABLE (NEXT WORD)
598 003644 004737 004030      JSR     PC,MASKER   ;CALL THE MASK CONTROL SUBROUTINE
599 003650 005203              INC      R3          ;INCREMENT THE WORD COUNTER
600 003652 000725              BR      INSYS2     ;GO BACK AND GET NEXT WORD
601 003654 050412      EXITA:  BIS      R4,(R2)   ;FILL THE TABLE (LAST WORD)
602 003656 006004              ROR     R4          ;WITH ALL BITS SET
603 003660 103375              BCC     EXITA      ;GO BACK IF NOT DONE
604 003662 042712 174000      EXITB:  BIC     #174000,(R2) ;CLEAR DOWN AND SECOND SYSTEM BITS AT TABLE
605 003666 010200      MOV     R2,R0      ;GET ADDRESS TO RO (CURRENT TABLE)
606 003670 005722      TST    (R2)+      ;ADD 2 TO THE POINTER
607 003672 022702 001206      FIL.DN: CMP    #DRVO,R2 ;TABLE FULL?
608 003676 001403      BEQ    2$         ;IF YES BRANCH
609 003700 012722 100000      MOV     #BIT15,(R2)+ ;FILL THE TABLE
610 003704 000772      BR     FIL.DN     ;GO BACK TRY AGAIN
611 003706 011001      2$:    MOV     (R0),R1 ;GET THE WORD TO R1
612 003710 110102      MOVVB  R1,R2      ;PC.MASK
613 003712 004737 005234      JSR     PC,MASK   ;FORM DRIVE # FOR MOUNT
614 003716 004737 004250      JSR     PC,DO2    ;WRITE NEW INFO
615 003722 004737 005104      JSR     PC,WRLINK ;READ SAMPLE (ALL DRIVES)
616 003726 004737 005632      JSR     PC,RDLINK ;TYPE ASCIZ STRING
617 003732 104401      TYPE   ,65$      ;GET OVER THE ASCIZ
618 003736 000427      BR     64$       ;GET OVER THE ASCIZ
619
620 004016              ;;65$: .ASCIZ <15><12>/DONE SYSTEM 2 RESTART SYSTEM 1, TYPE WORD /
621 004018 011046      64$:    MOV     (R0),-(SP) ;GET WORD FOR SYSTEM 1 AND TYPE
622 004020 104403      MOV     TYPOS     ;
623 004022 006      .BYTE 6
624 004023 001      .BYTE 1
625 004024 000000      1$:    HALT      ;HALT (FINISHED SYSTEM #2)
626 004026 000776      BR     1$        ;GO BACK TO HALT
627
628
629 ;THIS ROUTINE SETS UP THE MASK BITS IN THE LOG TABLE FOR
630 ;SYSTEM #2, IF A DRIVE IS DOWN NO BIT IS SET BUT THE MASK
631 ;IS SHIFTED.
632 004030 005712      MASKER: TST    (R2)    ;IS THE DRIVE UP
633 004032 100401      BMI    RETRN4      ;IF NO, BRANCH
634 004034 110412      MOVVB  R4,(R2)     ;MOVE THE MASK BIT IN THE TABLE
635 004036 006004      RETRN4: ROR    R4    ;ROTATE THE MASK
636 004040 103003      BCC    1$         ;DONE?, IF NO, BRANCH
637 004042 012716 003662      MOV     #EXITB,(SP) ;SET UP FOR RETURN
638 004046 000207      RTS    PC
639 004050 032712 040000      1$:    BIT     #BIT14,(R2) ;IS THIS SYSTEM # 2'S DRIVE?
640 004054 001403      BEQ    2$         ;IF NO, BRANCH
641 004056 012716 003654      MOV     #EXITA,(SP) ;SET UP FOR RETURN
642 004062 000207      RTS    PC
643 004064 062702 000002      2$:    ADD    #2,R2    ;INDEX THRU LOGA TABLE
644 004070 000207      RTS    PC          ;RETURN
645
646 ;ROUTINE TO BUILD THE ACTIVE LOGICAL UNIT TABLE
647
648 004072 022737 000010 001330 STORE: CMP    #10,@@KYTEMP ;IS INPUT A LEGAL NUMBER?
649 004100 000240      NOP                ;***

```

```

650 004102 003417      BLE    ILEGAL     ;IF NOT BRANCH
651 004104 000337 001330      SWAB   @@KYTEMP   ;ALIGN DRIVE # FOR TABLE
652 004110 013721 001330      MOV     @@KYTEMP,(R1)+ ;PUT THE WORD IN THE TABLE
653 004114 005037 001330      CLR    @@KYTEMP   ;CLEAR THE TEMP WORD
654 004120 105237 001311      INCB   @DRCNT1    ;INCREMENT THE COUNTER
655 004124 022701 001206      CMP    #DRVO,R1   ;IS THE TABLE FULL?
656 004130 001401      BEQ    TBLFUL     ;IF YES BRANCH
657 004132 000207      RTS    PC         ;IF NO RETURN
658 004134 012716 003050      TBLFUL: MOV    #SECSYS,(SP) ;IF TABLE FULL SET UP FOR SYSTEM #2
659 004140 000207      RTS    PC         ;RETURN
660 004142 012716 002644      ILEGAL: MOV    #AUTSL2,(SP) ;IF ILLEGAL RESPONSE, GO BACK
661 004146 104401 004154      TYPE   ,65$      ;TYPE ASCIZ STRING
662 004152 000407      BR     64$       ;GET OVER THE ASCIZ
663
664 004172              ;;65$: .ASCIZ /ILLEGAL INPUT/
665 004172 000207      64$:    RTS    PC          ;RETURN
666
667 ;THIS ROUTINE GETS A DRIVE # AND BUILDS A TABLE OF
668 ;CYLINDER ADDRESS FOR USE BY WRITE, (R0)=ADDRESS OF ACTIVE
669 ;WORD IN LOGICAL TABLE, IT ALSO SETS AND CLEARS THE MASK BITS
670 ;OF THE TABLE AS OPERATIONS INDICATE
671
672 004174 011001      CYCLE:  MOV     (R0),R1 ;GET THE LOGICAL UNIT ACTIVE WORD TO R1
673 004176 032701 040000      BIT     #BIT14,R1   ;IS IT ON SYSTEM #2
674 004202 001402      BEQ    CYCL2      ;IF NO BRANCH
675 004204 000137 006752      JMP     @SECON     ;GO TO SYSTEM #2
676 004210 113703 001324      CYCL2: MOVVB  @INDEX,R3 ;GET THE INDEX VALUE
677 004214 116302 001256      MOVVB  MSKTBL(R3),R2 ;GET THE MASK TO R2
678 004220 004737 005234      CYCLE2: JSR    PC,MASK ;CALL THE MASK SUBROUTINE
679 004224 012703 001166      LDPLG: MOV    #LOGA,R3 ;GET TABLE ADDRESS TO R3
680 004230 020003      2$:    CMP    R0,R3     ;IS ACTIVE ADDRESS=TO FIRST ADDRESS
681 004232 001002      BNE    3$        ;IF NO BRANCH
682 004234 050223      BIS    R2,(R3)+   ;IF YES SET BITS TO SHOW WRITE
683 004236 000401      BR     4$        ;BRANCH TO SEE IF DONE
684 004240 040223      3$:    BIC    R2,(R3)+ ;CLEAR BITS TO SHOW OVER-WRITE
685 004242 022703 001206      4$:    CMP    #DRVO,R3 ;DONE
686 004246 001370      BNE    2$        ;IF NO GO BACK
687 004250 000301      DO2:   SWAB   R1    ;GET DRIVE # TO LOW BYTE
688 004252 000240      NOP                ;***
689 004254 110102      MOVVB  R1,R2      ;GET IT TO R2
690 004256 042702 000370      BIC    #370,R2    ;CLEAR THE UNUSED BITS
691 004262 110237 001314      MOVVB  R2,@DRIVE  ;GET THE DRIVE #
692 004266 006102      ROL    R2        ;SHIFT THE DRIVE # TO
693 004270 006102      ROL    R2        ;ALIGN IT FOR THE DRIVE ADDR.
694 004272 006102      ROL    R2        ;KEEP IT MOVING!
695 004274 006102      ROL    R2        ;A LITTLE MORE!
696 004276 006102      ROL    R2        ;THERE IT IS
697 004300 110237      MOVVB  R2,@DSKTMP+1 ;GET IT TO DISK ADDR. TEMP.
698 004304 110237 001335      MOVVB  R2,@DSKADR+1 ;CALL MOUNT
699 004310 004737 004316      JSR     PC,MOUNT  ;RETURN
700 004314 000207      RTS    PC
701
702 004316 105237 001324      MOUNT:  INCB   @INDEX ;INCREMENT THE INDEX
703 004322 000240      NOP                ;***
704 004324 112737 000005 001321      MOVVB  #5,@#ECNT  ;SET ERROR CNTR TO 5
705 004332 112737 000003 001322      MOVVB  #3,@#CNTSIN ;SET SIN CNTR TO 3

```

```

706 004340 104401 004346          TYPE ,65$          ;;TYPE ASCIZ STRING
707 004344 000414          BR 64$           ;;GET OVER THE ASCIZ
708          ;;65$: .ASCIZ <15><12>/MOUNT PACK ON DRIVE #/
709 004376          64$:
710 004376 013746 001314          MOV DRIVE,-(SP)    ;;SAVE DRIVE FOR TYPEOUT
711 004402 104403          TYP0S            ;;GO TYPE--OCTAL ASCII
712 004404 001          .BYTE 1          ;;TYPE 1 DIGIT(S)
713 004405 000          .BYTE 0          ;;SUPPRESS LEADING ZEROS
714 004406 104401 004414          TYPE ,67$        ;;TYPE ASCIZ STRING
715 004412 000415          BR 66$           ;;GET OVER THE ASCIZ
716          ;;67$: .ASCIZ <15><12>/MAKE PACK WRITE ENABLE/
717 004446          66$:
718 004446 104401 004454          TYPE ,69$        ;;TYPE ASCIZ STRING
719 004452 000420          BR 68$           ;;GET OVER THE ASCIZ
720          ;;69$: .ASCIZ <15><12>/PRESS CONTINUE WHEN DRIVE RDY/
721 004514          68$:
722 004514 000000          HALT             ;;CALL INITIALIZER
723 004516 004737 004524          JSR PC,INITIL    ;;CALL INITIALIZER
724 004522 000207          RTS PC           ;;RETURN
725
726          ;THIS ROUTINE INITIALIZES A DRIVE AND INSURES THAT IT IS READY AND
727          ;WRITE ENABLED, IT IS ENTERED FROM MOUNT OR FROM EXECUTE IF OPERATION FAILS
728
729 004524 010046          INITIL: MOV R0,-(SP)      ;SAVE R0
730 004526 013700 001334          MOV @#DSKADR,R0   ;GET THE DRIVE # TO R0
731 004532 042700 001777          BIC #1777,R0      ;CLEAR THE UNUSED BITS
732 004536 005037 001346          INITI2: CLR @#TIMR ;CLEAR THE TIMER
733 004542 105037 001323          CLR @#TIMR2
734 004546 000240          NOP              ;***
735 004550 010077 174622          MOV R0,@RKDA      ;GET DRIVE # TO 'DA' REGISTER
736 004554 012777 000001 174608          MOV #1,@RKCS      ;ISSUE CONTROL RESET + GO
737 004562 004737 005614          JSR PC,SMTME
738 004566 105777 174576          1$: TSTB @RKCS     ;DID CONTROL READY SET
739 004572 100423          BMI 2$           ;IF YES BRANCH
740 004574 005237 001346          INC @#TIMR        ;IF NO INCREMENT THE TIMER
741 004600 001372          BNE 1$           ;IF TIMER NOT ZERO BRANCH
742 004602 104401 004610          TYPE ,65$        ;;TYPE ASCIZ STRING
743 004606 000415          BR 64$           ;;GET OVER THE ASCIZ
744          ;;65$: .ASCIZ <15><12>/CONTROL RESET TIME OUT/
745 004642          64$:
746 004642 010077 174530          2$: MOV R0,@RKDA      ;DRIVE NUMBER TO 'DA' REG.
747 004646 105777 174512          TSTB @RKDS        ;IS DRIVE READY
748 004652 100415          BMI 3$           ;IF YES BRANCH
749 004654 104401 004662          TYPE ,67$        ;;TYPE ASCIZ STRING
750 004660 000411          BR 66$           ;;GET OVER THE ASCIZ
751          ;;67$: .ASCIZ <15><12>/DRIVE NOT READY/
752 004704          66$:
753 004704 000714          BR INITI2        ;GO BACK TRY AGAIN
754 004706 032777 000040 174450          BIT #BITS,@RKDS   ;IS DRIVE WRITE LOCKED?
755 004714 001420          BEQ 4$           ;IF NO, BRANCH
756 004716 104401 004724          TYPE ,69$        ;;TYPE ASCIZ STRING
757 004722 000414          BR 68$           ;;GET OVER THE ASCIZ
758          ;;69$: .ASCIZ <15><12>/DRIVE WRITE PROTECTED/
759 004754          68$:
760 004754 000670          BR INITI2        ;YES, GO BACK TRY AGAIN
761 004756 005037 001346          CLR @#TIMR       ;CLEAR THE TIMER
    
```

```

762 004762 010077 174410          MOV R0,@RKDA      ;GET THE DRIVE # TO 'DA' REGISTER
763 004766 012777 000015 174374          MOV #15,@RKCS     ;ISSUE DRIVE RESET + GO
764 004774 004737 005614          JSR PC,SMTME
765 005000 105777 174364          5$: TSTB @RKCS      ;***
766 005004 100375          BPL 5$
767 005006 032777 000100 174350          BIT #100,@RKDS    ;READ/WRITE/SEEK READY BIT SET?
768 005014 001031          BNE 6$           ;IF YES, BRANCH
769 005016 005237 001346          INC @#TIMR        ;NO, INCREMENT THE TIMER
770 005022 001366          BNE 5$           ;GO BACK AND CHECK IF TIMER NOT 0
771 005024 105737 001323          TSTB @#TIMR2
772 005030 001003          BNE 7$           ;***
773 005032 105237 001323          INCB @#TIMR2
774 005036 000760          BR 5$
775 005040          7$:
776 005040 104401 005046          TYPE ,71$        ;;TYPE ASCIZ STRING
777 005044 000414          BR 70$           ;;GET OVER THE ASCIZ
778          ;;71$: .ASCIZ <15><12>/DRIVE RESET TIMED OUT/
779 005076          70$:
780 005076 000727          BR 4$           ;GO BACK, TRY AGAIN
781 005100 012600          6$: MOV (SP)+,R0      ;RESTORE R0
782 005102 000207          RTS PC           ;RETURN TO CALLER
783
784          ;THIS ROUTINE TAKES CARE OF ALL LINKAGES FOR THE
785          ;EXECUTE ROUTINE. IT FORMS THE ADDRESS AND SETS UP ALL THE
786          ;REGISTERS FOR THE WRITE OPERATION
787
788 005104 105037 001316          WRLINK: CLR @#COMND ;INDICATE WRITE OPERATION
789 005110 000240          NOP              ;***
790 005112 113701 001314          MOV @#DRIVE,R1    ;PICK UP THE DRIVE #
791 005116 006101          ROL R1           ;MAKE IT A WORD INDEX
792 005120 016137 001206 001362          1$: MOV DRV0(R1),@#PATRN ;PICK UP THE DATA PATTERN
793 005126 004737 005302          JSR PC,CYLADR     ;CALL CYLINDER ADDRESS
794 005132 000401          BR 2$           ;RETURN HERE IF NOT LAST BASE
795 005134 000207          RTS PC           ;RETURN HERE IF LAST BASE
796 005136 012737 001362 001336          2$: MOV #PATRN,@#BUSADR ;GET THE ADDRESS OF THE OUTPUT
797 005144 013737 001342 001340          MOV @#CYLCNT,@#WRDRCNT ;GET THE WORD COUNT
798 005152 013737 001354 001332          MOV @#WRITCS,@#CONTRL ;GET THE CONTROL + STATUS WORD
799 005160 004737 005402          JSR PC,EXECUTE    ;CALL EXECUTE
800 005164 032737 000020 001334          BIT #BIT4,@#DSKADR ;WAS THIS WRITE SURFACE "1"?
801 005172 001006          BNE 3$           ;IF YES BRANCH
802 005174 052737 000020 001334          BIS #BIT4,@#DSKADR ;SET SURFACE ONE BIT
803 005202 105137 001362          COMB @#PATRN      ;MAKE IT SURFACE ONE DATA
804 005206 000753          BR 2$           ;RELOAD REGISTERS AND EXECUTE
805 005210 042737 000020 001334          3$: BIC #BIT4,@#DSKADR ;SET UP FOR SURFACE "0"
806 005216 105137 001362          COMB @#PATRN      ;MAKE IT SURFACE 0 DATA
807 005222 005202          INC R2           ;INC. THRU SELECTED CYL. OFFSET TABLE
808 005224 004737 005312          JSR PC,CYLOFF     ;GET THE CYLINDER VALUE
809 005230 000742          BR 2$           ;RETURN HERE IF MORE TO READ
810 005232 000207          RTS PC           ;RETURN HERE IF FINISHED
811
812          ;THIS ROUTINE EXPECTS TO FIND A MASK IN R2, AND FROM THIS MASK
813          ;BUILDS A TABLE (AT CYLTBL) OF CYLINDER ADDRESS OFFSETS; THE TABLE
814          ;IS TERMINATED BY A #377
815
816 005234 010546          MASK: MOV R5,-(SP) ;SAVE R5
817 005236 042702 177400          BIC #177400,R2   ;CLR THE UNUSED BITS OF THE MASK
    
```

```

818 005242 000240                NOP                ;***
819 005244 012703 000200        MOV                #200,R3    ;SET UP THE COMPARE MASK
820 005250 005004                CLR                R4          ;CLR THE INDEX COUNTER
821 005252 012705 001274        MOV                #CYLTL,R5   ;GET THE CYLINDER TABLE ADDRESS
822 005256 030203                1$: BIT            R2,R3       ;IS THE MASK BIT SELECTED IN BASE
823 005260 001401                BEQ                2$          ;IF NO BRANCH
824 005262 110425                MOV#              R4,(R5)+     ;MOVE THE CYLINDER BASE TO THE TABLE
825 005264 105204                2$: INCB          R4          ;INCREMENT THE BASE
826 005266 006003                ROR                R3          ;ROTATE THE COMPARE MASK
827 005270 103372                BCC               1$          ;IF NOT DONE GO BACK
828 005272 112715 000377        MOV#              #377,(R5)   ;IF DONE LOAD FINISH FLAG
829 005276 012605                MOV                (SP)+,R5   ;RESTORE R5
830 005300 000207                RTS                 PC         ;RETURN TO CALLER
831
832 ;THIS ROUTINE FORMS A CYLINDER ADDRESS FOR BOTH THE READ AND WRITE ROUTINES
833 ;WHEN THE BASE TABLE IS FULLY INDEXED IT RETURNS TO PC+2 OF CALLER
834
835 005302 012703 001266        CYLADR: MOV        #BASE,R3    ;GET THE CYLINDER TABLE ADDRESS
836 005306 012702 001274        CYLAD2: MOV        #CYLTL,R2   ;GET THE SELECTED CYL BASE ADDR.
837 005312 111204                CYLOFF: MOV#       (R2),R4     ;GET THE SELECTED CYL VALUE TO R4
838 005314 000240                NOP                ;***
839 005316 122704 000377        CMPB              #377,R4     ;IS IT THE TABLE TERMINATOR?
840 005322 001416                BEQ                BASINC     ;IF YES BRANCH
841 005324 005046                CLR                -(SP)      ;INSURE CLEAN WORD
842 005326 111316                MOV#              (R3),(SP)   ;GET THE CYL ADDRESS ON THE STACK
843 005330 062604                ADD                (SP)+,R4   ;AND IT TO THE SELECTED OFFSET
844 005332 006104                ROL                R4          ;SHIFT THIS RESULT
845 005334 006104                ROL                R4          ;TO ALIGN THE NEWLY FORMED
846 005336 006104                ROL                R4          ;CYLINDER ADDRESS WITH BITS
847 005340 006104                ROL                R4          ;5 THRU 12 OF RKDA AND
848 005342 006104                ROL                R4          ;STORE THIS IN DSKADR
849 005344 042737 017777 001334  BIC                #017777,@DSKADR ;CLEAR ALL BUT DRIVE NUMBER
850 005352 050437 001334        BIS                R4,@DSKADR ;PUT IT IN DSKADR
851 005356 000207                RTS                 PC
852 005360 005203                BASINC: INC        R3         ;PICK UP ADDRESS OF NEXT BASE CYL.
853 005362 000240                NOP                ;***
854 005364 022703 001274        CMP                #CYLTL,R3  ;ARE YOU FINISHED?
855 005370 001401                BEQ                RETRN3     ;IF YES, BRANCH
856 005372 000745                BR                 CYLAD2     ;NO GO BACK
857 005374 062716 000002        RETRN3: ACD        #2,(SP)    ;SET-UP FOR PC+2
858 005400 000207                RTS                 PC         ;RETURN
859
860 ;ROUTINE TO PERFORM INDICATED FUNCTION AND CHECK FOR
861 ;DONE AND ERRORS
862
863 005402 005037 001346        EXECUT: CLR        @#TIMR     ;CLEAR THE TIMER
864 005406 000240                NOP                ;HALT HERE TO CHECK COMMAND ABOUT TO BE EXECUTED
865 005410 105037 001323        CLR#              @#TIMR2     ;CLEAR SECOND TIMER
866 005414 013777 001334 173754  MOV                @#DSKADR,@RKDA ;LOAD THE DISK ADDRESS REGISTER
867 005422 013777 001336 173744  MOV                @#BUSADR,@RKBA ;LOAD THE BUS ADDRESS REGISTER
868 005430 013777 001340 173734  MOV                @#WRDCNT,@RKWC ;LOAD THE WORD COUNT REGISTER
869 005436 013777 001332 173724  MOV                @#CONTRL,@RKCS ;LOAD THE CONTROL REGISTER
870 005444 004737 005614        JSR                PC,SMTME   ;KILL TIME FOR RK11-C
871 005450 105777 173714        CHECK1: TSTB      @RKCS       ;IS CONTROL READY SET
872 005454 100011                BPL                TIME       ;IF NO BRANCH
873 005456 004737 006102        JSR                PC,ERRCHK
    
```

```

874 005462 005737 001360                TST                @#ERRFLG   ;ERROR?
875 005466 001403                BEQ                1$         ;IF NO BRANCH
876 005470 005037 001360        CLR                @#ERRFLG   ;CLEAR THE FLAG
877 005474 000742                BR                 EXECUT     ;TRY AGAIN
878 005476 000207                1$: RTS            PC
879 005500 005237 001346        TIME: INC          @#TIMR     ;***
880 005504 000240                NOP
881 005506 001360                BNE                CHECK1     ;SECOND TIMEOUT?
882 005510 105737 001323        TSTB              @#TIMR2     ;IF YES BRANCH
883 005514 001003                BNE                1$         ;INDICATE SECOND TIMEOUT
884 005516 105237 001323        INCB              @#TIMR2     ;GO BACK
885 005522 000752                BR                 CHECK1
886 005524 004737 004524        1$: JSR            PC,INITIL  ;TYPE ASCII STRING
887 005530 104401 005536        TYPE              ,65$       ;GET OVER THE ASCII
888 005534 000426                BR                 64$
889 ;:65$: .ASCIZ <15><12>/TIMED OUT ON OPERATION RETRY IN PROGRESS/
890 005612                64$:
891 005612 000673                BR                 EXECUT
892 005614 012737 000500 001346  SMTME: MOV        #500,@#TIMR
893 005622 005337 001346        1$: DEC            @#TIMR
894 005626 001375                BNE                1$
895 005630 000207                RTS                 PC
896
897 ;THIS ROUTINE CHECKS TO SEE IF A DRIVE IS ACTIVE FROM A
898 ;WRITE OPERATION, IT GETS THE READ MASK TO R3, AND THE
899 ;EXPECTED DATA PATTERN TO "PATTERN", AND THEN CALLS ADDRESS
900 ;CONTROL.
901
902 005632 010046                RDLINK: MOV        R0,-(SP)    ;SAVE R0
903 005634 000240                NOP                ;***
904 005636 152737 000377 001316  BISB              #377,@#COMMND ;INDICATE READ OPERATION
905 005644 012701 001166        MOV                #LOGA,R1   ;GET THE TABLE ADDRESS TO R1
906 005650 012705 001305        MOV                #SECTBL,R5 ;GET THE SECTOR TABLE ADDRESS
907 005654 000405                BR                 RD2        ;SKIP OVER THE INDEX, FIRST PASS
908 005656 062701 000002        RD1: ADD          #2,R1        ;ADD 2 TO THE ADDRESS
909 005662 022701 001206        CMP                #DRVO,R1   ;ARE YOU THRU THE ENTIRE TABLE
910 005666 001503                BEQ                EXIT2     ;IF YES EXIT
911 005670 005711                RD2: TST          (R1)        ;IS THE DRIVE ACTIVE
912 005672 100001                BPL                RD3       ;IF YES BRANCH
913 005674 000770                BR                 RD1        ;IF NO GET NEXT WORD
914 005676 011100                RD3: MOV          (R1),R0     ;GET THE ACTIVE WORD TO R0
915 005700 010002                MOV                R0,R2     ;COPY THE WORD
916 005702 000300                SWAB              R0          ;GET THE DRIVE # TO THE LOW BYTE
917 005704 042700 177770        BIC                #177770,R0 ;CLR ALL BUT THE DRIVE #
918 005710 110037 001317        MOV#              R0,@#WRTNBY ;SAVE THE DRIVE #
919 005714 006100                ROL                R0          ;MAKE IT AN INDEX
920 005716 016037 001206 001362  MOV                DRVO(R0),@#PATRN ;PICK UP THE DATA PATTERN
921 005724 004737 005234        JSR                PC,MASK    ;CALL THE MASK SUBROUTINE
922 005730 004737 005302        JSR                PC,CYLADR  ;GO FORM A CYLINDER ADDRESS
923 005734 000401                BR                 RD1        ;RETURN HERE IF NOT LAST BASE ADDR.
924 005736 000747                BR                 RD1        ;IF LAST BASE ADDRESS, RETURN HERE
925 005740 053737 001352 001334  RD4: BIS            @#DSKTMP,@#DSKADR ;SET THE DRIVE # BITS IN DISK ADDR.
926 005746 152537 001334        RD5: BIS            (R5)+,@#DSKADR ;SET THE SECTOR BITS IN DISK ADDR.
927 005752 012737 007356 001336  RD6: MOV            @#RDBUFF,@#BUSADR ;GET THE BUFFER ADDR.
928 005760 000240                NOP                ;***
929 005762 013737 001344 001340  MOV                @#SECCNT,@#WRDCNT ;GET THE WORD COUNT
    
```

```

930 005770 013737 001356 001332 MOV @#READCS,@#CONTRL ;GET THE READ CONTROL WORD
931 005776 004737 005402 JSR PC,EXECUT ;DO THE READ
932 006002 004737 006450 JSR PC,RDCHK ;CHECK THE DATA
933 006006 042737 000017 001334 BIC #17,@#DSKADR ;CLR THE SECTOR BITS IN DISK ADDR.
934 006014 022705 001311 CMP #DRCNT1,R5 ;WAS THIS THE LAST SECTOR?
935 006020 001352 BNE R05 ;IF NO GO BACK
936 006022 012705 001305 MOV #SECTBL,R5 ;IF YES RESET SECTOR POINTER
937 006026 032737 000020 001334 BIT #BIT4,@#DSKADR ;WAS IT SURFACE "1" THAT WAS READ?
938 006034 001006 BNE R07 ;IF YES, BRANCH
939 006036 052737 000020 001334 BIS #BIT4,@#DSKADR ;NO, SET SURFACE "1" BIT
940 006044 105137 001362 CDMB @#PATRN ;MAKE HEAD "1" PATTERN
941 006050 000736 BR R05 ;GO BACK AND EXECUTE
942 006052 042737 000020 001334 R07: BIC #BIT4,@#DSKADR ;CLEAR THE SURFACE BIT
943 006060 105137 001362 CDMB @#PATRN ;MAKE IT SURFACE 0 DATA
944 006064 005202 INC R2 ;INCREMENT THE SELECTED CYL TABLE POINTER
945 006066 004737 005312 JSR PC,CYLOFF ;GO FORM NEXT ADDRESS
946 006072 000722 BR R04 ;IF HERE IT IS NOT THE LAST BASE ADDR
947 006074 000670 BR R01 ;IF HERE GET NEXT WORD-DRIVE FINISHED
948
949 006076 012600 EXIT2: MOV (SP)+,R0 ;RESTORE R0
950 006100 000207 RTS PC ;RETURN TO MAIN LINE CODE
951
952 ;THE ERROR CHECK ROUTINE CHECKS IF AN ERROR OCCURRED
953 ;ON WRITING OR READING.
954
955 006102 032777 140000 173260 ERRCHK: BIT #140000,@#RKCS ;HARD ERROR OR ERROR SET?
956 006110 000240 NOP ;HALT HERE TO EXAMINE ERROR REG., ECT.
957 006112 001420 BEQ TSTSN1 ;IF NO, GO TEST 'SIN' BIT
958 006114 012777 000001 173246 MOV #1,@#RKCS ;IF YES, ISSUE CNTRL RESET + GO
959 006122 004737 005614 JSR PC,SMTME
960 006126 012777 177777 173224 MOV #-1,@#ERRFLG ;FLAG AN ERROR (PREVENT UPDATE OF ADDR)
961 006134 105777 173230 1$: TSTB @#RKCS ;CNTRL READY BIT SET (FROM CNTRL RESET)
962 006140 100375 BPL 1$ ;IF NO WAIT. (IF HUNG HERE RUN STATIC)
963 006142 105337 001321 DECB @#ECNT ;DECREMENT ERROR COUNTER
964 006146 001002 BNE TSTSN1 ;HAVE ERROR BITS SET 5/TIMES?
965 006150 000137 006232 JMP @#RESTRT ;IF HERE 5 ERRORS HAVE OCCURRED
966 006154 032777 001000 173202 TSTSN1: BIT #1000,@#RKDS ;SEEK INCOMPLETE SET?
967 006162 000240 NOP ;***
968 006164 001530 BEQ RETRN2 ;BRANCH IF NO
969 006166 012777 000015 173174 MOV #15,@#RKCS ;IF YES, ISSUE DRIVE RESET, GO
970 006174 012777 177777 173156 MOV #-1,@#ERRFLG ;FLAG AN ERROR (PREVENT UPDATE OF ADDR)
971 006202 004737 005614 JSR PC,SMTME
972 006206 105777 173156 2$: TSTB @#RKCS
973 006212 100375 BBL 2$
974 006214 032777 000100 173142 BIT #100,@#RKDS ;"R/W/S READY" BIT SET?
975 006222 001771 BEQ 2$ ;IF NO WAIT! (IF HUNG HERE RUN STATIC)
976 006224 105337 001322 DECB @#CNTSN ;DECREMENT SEEK INCOMPLETE COUNTER
977 006230 001106 BNE RETRN2 ;IF 3 'SIN' ERRORS FALL THROUGH
978 006232 105737 RESTRT: TSTB @#ECNT
979 006236 000240 NOP ;***
980 006240 001421 BEQ 1$
981 006242 104401 006250 TYPE ,65$ ;:TYPE ASCIZ STRING
982 006246 000415 BR 64$ ;:GET OVER THE ASCIZ
983
984 006302 ;:65$: .ASCIZ <15><12>/3 'SIN' ERRORS OCCURRED/
985 006302 000415 64$: BR 2$
    
```

```

986 006304 1$: TYPE ,87$ ;:TYPE ASCIZ STRING
987 006304 104401 006312 BR 66$ ;:GET OVER THE ASCIZ
988 006310 000412 ;:87$: .ASCIZ <15><12>/5 ERRORS OCCURRED/
989
990 006336 66$:
991 006336 2$:
992 006336 104401 006344 TYPE ,89$ ;:TYPE ASCIZ STRING
993 006342 000422 BR 88$ ;:GET OVER THE ASCIZ
994
995 006410 ;:89$: .ASCIZ / DRIVE DECLARED DOWN!! NOT TESTED I/
996 006410 105737 001316 68$: TSTB @#COMND ;TEST THE COMMAND
997 006414 100006 BPL 3$ ;IF WRITE, BRANCH
998 006416 062706 000004 ADD #4,SP ;POINT TO SAVE R0
999 006422 012600 MOV (SP)+,R0 ;GET R0 BACK
1000 006424 052710 100000 BIS #BIT15,(R0) ;SET THE DOWN BIT
1001 006430 000207 RTS PC ;RETURN TO MAIN CODE
1002 006432 052710 100000 3$: BIS #BIT15,(R0) ;SET THE DOWN BIT
1003 006436 012706 001100 MOV #STACK,SP ;RESTORE THE STACK
1004 006442 000137 003324 JMP @#GOT
1005 006446 000207 RETRN2: RTS PC
1006
1007 ;THIS ROUTINE CHECKS A SECTORS WORTH OF DATA ON A READ OPERATION
1008 ;IT ALLOWS 5 ERROR PRINTOUTS PER SECTOR
1009
1010 006450 010446 RDCHK: MOV R4,-(SP) ;SAVE R4
1011 006452 010546 MOV R5,-(SP) ;SAVE R5 FOR RDLINK
1012 006454 105037 001320 CLR @#HDRFLG ;CLEAR THE PRINT HEADER FLAG
1013 006460 000240 NOP ;***
1014 006462 012737 000005 001350 MOV #5,@#CHKCNT ;PUT ERROR COUNT IN CHECK COUNT
1015 006470 012704 007356 MOV #RDBUFF,R4 ;GET THE TABLE ADDRESS TO R4
1016 006474 013705 001362 MOV @#PATRN,R5 ;GET THE EXPECTED DATA TO R5
1017 006500 020524 1$: CMP R5,(R4)+ ;ARE THEY THE SAME
1018 006502 001515 BEQ 3$ ;IF YES BRANCH
1019 006504 105737 001320 TSTB @#HDRFLG ;IS THE HEADER FLAG CLEAR
1020 006510 001046 BNE 2$ ;IF NO BRANCH
1021 006512 104401 006520 TYPE ,65$ ;:TYPE ASCIZ STRING
1022 006516 000420 BR 64$ ;:GET OVER THE ASCIZ
1023
1024 006560 ;:65$: .ASCIZ <15><12>/ERROR! DATA WRITTEN BY DRIVE /
1025 006560 152737 000377 001320 64$: BISB #377,@#HDRFLG ;SET THE HEADER FLAG
1026 006566 113746 001317 MOV @#WRTNBY,-(SP) ;PICK UP THE DRIVE # THAT WROTE
1027 006572 104403 TYPOS
1028 006574 001 .BYTE 1
1029 006575 000 .BYTE 0
1030 006576 104401 006504 TYPE ,87$ ;:TYPE ASCIZ STRING
1031 006602 000411 BR 66$ ;:GET OVER THE ASCIZ
1032
1033 ;:87$: .ASCIZ / CANNOT BE READ./
1034 66$:
1035 006626 2$:
1036 006626 104401 006634 TYPE ,89$ ;:TYPE ASCIZ STRING
1037 006632 000405 BR 68$ ;:GET OVER THE ASCIZ
1038
1039 ;:89$: .ASCIZ <15><12>/ ADDR=/
1040 68$:
1041 006646 1038 MOV @#DSKADR,-(SP) ;PICK UP THE ADDRESS THAT FAILED
1042 006646 013746 001334 TYPOS
1043 006652 104403 .BYTE 6
1044 006654 006
    
```

```

1042 006655 001 .BYTE 1
1043 006656 104401 006664 TYPE ,71$ ;;TYPE ASCIZ STRING
1044 006662 000405 BR 70$ ;;GET OVER THE ASCIZ
1045
1046 006676 .ASCIZ / EXPCTD=/
70$:
1047 006676 010546 MOV R5,-(SP) ;PICK UP THE EXPECTED DATA (GOOD)
1048 006700 104404 TYPON
1049 006702 104401 006710 TYPE ,73$ ;;TYPE ASCIZ STRING
1050 006706 000405 BR 72$ ;;GET OVER THE ASCIZ
1051
1052 006722 .ASCIZ / RECDV=/
72$:
1053 006722 016446 177776 MOV ~2(R4),-(SP) ;PICK UP THE RECEIVED DATA (BAD)
1054 006726 104404 TYPON
1055 006730 005337 001350 DEC @#CHKCNT ;DECREMENT THE CHECK COUNT
1056 006734 001403 BEQ 4$ ;IF ZERO, BRANCH
1057 006736 022704 010356 3$: CMP #MANSEL,R4 ;DONE ALL CHECKS?
1058 006742 001256 BNE 1$ ;IF NO, GO BACK
1059 006744 012605 4$: MOV (SP)+,R5 ;RESTORE R5
1060 006746 012604 MOV (SP)+,R4 ;RESTORE R4
1061 006750 000207 RTS PC ;RETURN TO CALLER
1062
1063 ;THIS ROUTINE BUILDS A TABLE OF PARAMETERS TO PASS TO THE SECOND SYSTEM
1064 ;IT PACKS THE INFO FOR TWO DRIVES INTO ONE WORD
1065
1066 006752 005003 SECONC: CLR R3 ;CLEAR THE WORD COUNTER
1067 006754 000240 NOP ;***
1068 006756 012702 001256 MOV #MSKTBL,R2
6$: CLR ~(R2)
1069 006762 005042 CMP #PASTBL,R2
1070 006764 022702 001246 BNE 6$
1071 006770 001374 MOV #LOGA,R0 ;GET THE ACTIVE TABLE ADDRESS
1072 006772 012700 001166 MOV (R0)+,R1 ;PICK UP THE WORD
1073 006776 012001 SWAB R1 ;GET THE DRIVE # TO THE LOW BYTE
1074 007000 000301 BIC #177400,R1 ;CLEAR THE UNWANTED BITS
1075 007002 042701 177400 ROLB R1 ;ROTATE THE BYTE, WAS DOWN SET?
1076 007006 106101 BCC 2$ ;IF NO BRANCH
1077 007010 103002 BIS #BIT0,R1 ;SHOW THE DRIVE AS DOWN IN THE TABLE
1078 007012 052701 000001 TSTB R1 ;IS THIS THE SYSTEM #2 DRIVE?
1079 007016 105701 BMI 3$ ;BRANCH IF YES
1080 007020 100404 BICB #360,R1 ;CLEAR THE UNUSED BITS
1081 007022 142701 000360 MOV R1,(R2)+ ;GET THIS # TO THE PASS TABLE
1082 007026 110122 BR 1$ ;GET THE NEXT WORD FROM ACTIVE TABLE
1083 007030 000762 3$: MOV R1,(R2) ;GET THE LAST DRIVE TO THE PASS TABLE
1084 007032 110112 MOV #PASTBL,R2 ;RESTORE THE TABLE POINTER
1085 007034 012702 001246 TYPE ,65$ ;;TYPE ASCIZ STRING
1086 007040 104401 007046 BR 64$ ;;GET OVER THE ASCIZ
1087 007044 000425
1088
1089 007120 .ASCIZ <15><12>/LOAD AND START ADDRESS 210 ON SYSTEM #2/
64$:
1090 007120 104401 007126 TYPE ,67$ ;;TYPE ASCIZ STRING
1091 007124 000424 BR 66$ ;;GET OVER THE ASCIZ
1092
1093 007176 .ASCIZ <15><12>/AND TYPE THE BELOW WHEN ASKED FOR IT./
66$:
1094 007176 005203 4$: INC R3 ;INCREMENT THE WORD COUNTER
1095 007200 104401 007206 TYPE ,69$ ;;TYPE ASCIZ STRING
1096 007204 000405 BR 68$ ;;GET OVER THE ASCIZ
1097
1098: .ASCIZ <15><12>/WORD /
    
```

```

1098 007220 68$: MOV R3,-(SP) ;GET THE WORD COUNT ON THE STACK
1099 007220 010346 TYPON
1100 007222 104403 .BYTE 6
1101 007224 006 .BYTE 0
1102 007225 000 007234 TYPE ,71$ ;;TYPE ASCIZ STRING
1103 007226 104401 BR 70$ ;;GET OVER THE ASCIZ
1104 007232 000401
1105
1106 007236 .ASCIZ /=/
70$:
1107 007236 012246 MOV (R2)+,-(SP) ;GET THE FIRST TO THE STACK
1108 007240 104403 TYPON
1109 007242 006 .BYTE 6
1110 007243 001 .BYTE 1
1111 007244 032762 100000 177776 BIT #BIT15,-2(R2) ;WAS THIS THE TABLE TERMINATOR
1112 007252 001004 BNE 5$ ;BRANCH IF YES
1113 007254 032762 000200 177776 BIT #BIT7,-2(R2) ;TERMINATOR?
1114 007262 001745 BEQ 4$ ;IF NO BRANCH
1115 007264 005740 5$: TST ~(R0)
1116 007266 000000 HALT
1117
1118 007270 RETFR2:
1119 007270 104401 007276 TYPE ,65$ ;;TYPE ASCIZ STRING
1120 007274 000404 BR 64$ ;;GET OVER THE ASCIZ
1121
1122 007306 .ASCIZ <15><12>/WORD=/
64$:
1123 007306 104411 RDOCT
1124 007310 012602 MOV (SP)+,R2 ;GET THE WORD FROM SYSTEM 2 TO TABLE
1125 007312 042702 177400 BIC #177400,R2
1126 007316 012704 001166 MOV #LOGA,R4 ;SET POINTER LOOK FOR FIRST "UP" DRIVE
1127 007322 005724 1$: TST (R4)+ ;DRIVE UP?
1128 007324 100776 BMI 1$ ;IF NO BRANCH
1129 007326 010437 001100 MOV R4,@#SPASS
1130 007332 01440+ MOV -(R4),R1
1131 007334 000240 NOP ;***
1132 007336 004737 004224 JSR PC,LDFLG ;CALL D02+
1133 007342 004737 005632 JSR PC,RDLINK ;CALL READ CHECK
1134 007346 013700 001100 MOV @#SPASS,R0
1135 007352 000137 003372 JMP @#EXTFR2 ;GO TO END OF TEST
1136
1137 007356 000400 RDBUFF: .BLKW 400
1138
1139 010356 000240 MANSEL: NOP ;TABLE TERMINATOR
1140
1141
1142
1143
1144
1145 010360 BADONE:
1146 010360 104401 010366 TYPE ,65$ ;;TYPE ASCIZ STRING
1147 010364 000401 BR 64$ ;;GET OVER THE ASCIZ
1148
1149 010370 .ASCIZ /?/
64$:
1150
1151
1152 .SBTTL OSCILLATING SEEK ROUTINE
1153
    
```



```

1154 010370          SECT.2:
1155 010370 104401 010376      TYPE      .65$      ;;TYPE ASCIZ STRING
1156 010374 000416          BR          64$      ;;GET OVER THE ASCIZ
1157          ;;65$: .ASCIZ <15><12>/OSCILLATING SEEK PACKAGE/
1158 010432          64$:
1159
1160 010432 012700 020614      MOV      #DRIVO,R0      ;FIND OUT WHICH DRIVES ARE
1161 010436 005001          CLR      R1              ;PRESENT AND PUT THE
1162 010440 01C177 170732      1$:     MOV      R1,@RKDA      ;DRIVE #'S IN A TABLE STARTING
1163 010444 105777 170714      TSTB    @RKDS           ;AT 'DRIVO'. BITS 15-13 CONTAINS
1164 010450 100001          BPL     2$              ;THE DRIVE #
1165 010452 010120          MOV      R1,(R0)+
1166 010454 062701 020000      2$:     ADD      #20000,R1
1167 010460 001367          BNE     1$
1168 010462 012710 177777      MOV      #-1,(R0)      ;SET THE TERMINATOR TO THE TABLE
1169
1170 010466 013702 001376      INIT.2: MOV      RKDA,R2      ;ISSUE CONTROL RESET + GO I
1171 010472 012777 000001 170870  MOV      #1,@RKCS
1172 010500 004737 005614      JSR     PC,SMTME
1173 010504 105777 170660      1$:     TSTB    @RKCS           ;DID CONTROL READY SET?
1174 010510 100375          BPL     1$              ;IF NO WAIT! (IF HUNG RUN STATIC)
1175 010512 012700 020614      MOV      #DRIVO,R0
1176 010516 012077 170654      3$:     MOV      (R0)+,@RKDA
1177 010522 012777 000015 170840  MOV      #15,@RKCS
1178 010530 004737 005614      JSR     PC,SMTME
1179 010534 105777 170630      2$:     TSTB    @RKCS
1180 010540 100375          BPL     2$
1181 010542 022710 177777      CMP     #-1,(R0)
1182 010546 001363          BNE     3$
1183 010550 104401 010556      TYPE     .65$           ;;TYPE ASCIZ STRING
1184 010554 000432          BR      64$           ;;GET OVER THE ASCIZ
1185          ;;65$: .ASCIZ <15><12>/SET SW0 TO SW7 TO SELECT DRIVES FOR TEST AND CONT/
1186 010642          64$:
1187 010642 104401 010650      TYPE     .67$           ;;TYPE ASCIZ STRING
1188 010646 000426          BR      66$           ;;GET OVER THE ASCIZ
1189          ;;67$: .ASCIZ <15><12>/RESET SW0 TO SW7 TO TEST ALL AVAIL DRIVES/
1190 010724          66$:
1191 010724 022737 000178 001140 7$:     CMP     #SWREG,SWR      ;SOFTWARE SWITCH REGISTER IN USE ?
1192 010732 001092          BNE     9$
1193 010734 104405          GTSWR   9$             ;BR IF NOT
1194 010736 000401          BR      8$             ;REQUEST NEW CONTENTS FOR SWITCH REG
1195 010740 000000          BR      8$             ;CONTINUE
1196 010742 117704 170172      9$:     HALT    @SWR,R4       ;WAIT FOR OPERATOR TO ENTER NEW SWR VALUE
1197 010746 001413          BEQ     4$             ;SW0 TO SW7 TO R4
1198 010750 012700 020614      8$:     MOV     #DRIVO,R0      ;NONE SET, SO TEST ALL
1199 010754 005001          CLR     R1              ;TABLE TO STORE DRIVE ADDRS
1200 010756 006004          ROR     R4              ;ADDR OF DRIVE
1201 010760 103001          BCC     5$             ;NEXT SWITCH TO CARRY
1202 010762 010120          MOV     R1,(R0)+
1203 010764 062701 020000      6$:     ADD     #20000,R1
1204 010770 001372          BNE     5$             ;SWITCH NOT SET
1205 010772 012720 177777      5$:     MOV     #-1,(R0)+
1206 010776 105737 001326      4$:     TSTB    QSPFLG        ;SWITCH SET, SO MOVE ADDR TO TABLE
1207 011002 001165          BNE     RWSRDY         ;ADDR OF NEXT DRIVE
1208 011004 104401 011012      TYPE     .69$           ;ALL DONE WHEN ZERO
1209 011010 000432          BR      68$           ;TABLE TERMINATOR
1210          ;;69$: .ASCIZ <15><12>/TYPE ASCIZ STRING
1211          ;;68$: .ASCIZ <15><12>/TYPE ASCIZ STRING
1212          ;;68$: .ASCIZ <15><12>/TYPE ASCIZ STRING
1213          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1214          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1215          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1216          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1217          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1218          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1219          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1220          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1221          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1222          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1223          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1224          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1225          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1226          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1227          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1228          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1229          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1230          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1231          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1232          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1233          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1234          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1235          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1236          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1237          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1238          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1239          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1240          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1241          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1242          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1243          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1244          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1245          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1246          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1247          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1248          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1249          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1250          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1251          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1252          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1253          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1254          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1255          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1256          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1257          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1258          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1259          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1260          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1261          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1262          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1263          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1264          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1265          ;;70$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST

```

```

1210          ;;69$: .ASCIZ <15><12>/TOGGLE THE "FIRST CYLINDER ADDRESS" (OUTER LIMIT)/
1211          68$:
1212          TYPE     .71$           ;;TYPE ASCIZ STRING
1213          BR      70$           ;;GET OVER THE ASCIZ
1214          ;;71$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1215          70$:
1216          TYPE     .73$           ;;TYPE ASCIZ STRING
1217          BR      72$           ;;GET OVER THE ASCIZ
1218          ;;73$: .ASCIZ <15><12>/CYLINDER ADDRESS" (INNER LIMIT) INTO THE HIGH/
1219          72$:
1220          TYPE     .75$           ;;TYPE ASCIZ STRING
1221          BR      74$           ;;GET OVER THE ASCIZ
1222          ;;75$: .ASCIZ <15><12>/
1223          74$:
1224          INCB   QSPFLG          ;BYTE (BIT8-15), THEN PRESS CONTINUE./
1225          RWSRDY: BIT      #100,@RKDS      ;"READ/WRITE/SEEK READY" BIT SET?
1226          BEQ     RWSRDY         ;IF NO WAIT! (IF HUNG RUN STATIC)
1227          TRYAGN: CMP     #SWREG,SWR      ;SOFTWARE SWITCH REGISTER IN USE ?
1228          BNE     1$             ;BR IF NOT
1229          GTSWR   1$             ;GET NEW CONTENTS
1230          BR      CONTIN         ;CONTINUE
1231          1$:     HALT
1232          CONTIN: MOV      #1,@RKCS       ;CONTROL RESET
1233          TSTB    @RKCS
1234          BPL     .-4
1235          MOV     SWR,R5          ;GET THE ADDRESS OF THE SWITCH REG. TO R5
1236          1$:     MOV     (R5)+,R1        ;GET A BYTE TO R1
1237          BIC     #177400,R1       ;CLEAR THE UNUSED BITS
1238          CMP     #312,R1          ;IS ADDRESS LEGAL?
1239          BPL     2$             ;BRANCH IF YES
1240          TYPE     .65$           ;;TYPE ASCIZ STRING
1241          BR      64$           ;;GET OVER THE ASCIZ
1242          ;;65$: .ASCIZ <15><12>/INVALID ADDRESS IN SWITCH REGISTER TRY AGAIN/
1243          64$:
1244          BR      TRYAGN         ;GO BACK FOR NEW PARAMETERS
1245          2$:     ROL     R1              ;ROTATING THIS REGISTER
1246          ROL     R1              ;MAKES THE BYTE FROM THE
1247          ROL     R1              ;SWITCH REGISTER LINE UP
1248          ROL     R1              ;WITH THE CYLINDER BITS OF
1249          ROL     R1              ;THE RKDA REGISTER.
1250          BIC     #160037,R1       ;CLEAR THE UNUSED BITS
1251          BIT     #1,R5           ;IS THIS THE LOW BYTE?
1252          BNE     3$             ;BRANCH IF YES
1253          MOV     R1,@SEEKI        ;STORE THE INNER LIMIT OF THE SEEK
1254          BR      SEKSET         ;START SEEKS
1255          3$:     MOV     R1,@SEEKO      ;STORE THE OUTER LIMIT OF THE SEEK
1256          BR      1$             ;GO BACK AND GET HIGH BYTE.
1257
1258          SEKSET: MOV     #50,R3       ;GET THE NUMBER OF SEEK CYCLES TO R3
1259          MOV     @SEEKO,R4        ;ADD THE OUTER LIMIT CYLINDER ADDRESS
1260          MOV     @SEEKI,R5        ;ADD THE INNER LIMIT CYLINDER ADDRESS
1261
1262          LDSEEK: MOV     #DRIVO,R0      ;INITIALIZE POINTER
1263          5$:     MOV     R4,@RKDA      ;GET INNER LIMIT CYL ADRES
1264          BIS     (R0)+,@RKDA      ;SET DRIVE ADRES
1265          MOV     #11,@RKCS        ;ISSUE SEEK+GO! (FOR INNER LIMIT)

```

```
1266 011626 004737 005614 JSR PC,SMTME
1267 011632 105777 167532 3$: TSTB @RKCS
1268 011636 100375 BPL 3$
1269
1270 011640 022710 177777 CMP #-1,(R0) ;ALL DRIVES DONE?
1271 011644 001361 BNE 5$ ;NO
1272 011646 012700 020614 MOV #DRIVO,R0
1273 011652 012077 167520 6$: MOV (R0)+,@RKDA ;ADRES THE DRIVE
1274 011656 032777 000100 167500 7$: BIT @RWS,@RKDS ;SEEK DONE?
1275 011664 001774 BEQ 7$ ;NO, WAIT
1276 011666 022710 177777 CMP #-1,(R0) ;ALL DRIVES DONE?
1277 011672 001361 BNE 6$ ;NO
1278
1279 011674 012700 020614 MOV #DRIVO,R0
1280 011700 010577 167472 8$: MOV R5,@RKDA ;SET OUTER CYL ADRES
1281 011704 052077 167466 BIT (R0)+,@RKDA ;SET DRIVE ADRES
1282 011710 012777 000011 167452 MOV #11,@RKCS ;ISSUE SEEK+GD! (FOR OUTER LIMIT)
1283 011716 004737 005614 JSR PC,SMTME
1284 011722 105777 167442 9$: TSTB @RKCS
1285 011726 100375 BPL 9$
1286
1287 011730 022710 177777 CMP #-1,(R0) ;ALL DONE?
1288 011734 001361 BNE 8$ ;NO
1289
1290 011736 012700 020614 MOV #DRIVO,R0
1291 011742 012077 167430 10$: MOV (R0)+,@RKDA ;SET DRIVE ADRES
1292 011746 032777 000100 167410 11$: BIT @RWS,@RKDS ;SEEK DONE?
1293 011754 001774 BEQ 11$
1294 011756 022710 177777 CMP #-1,(R0) ;ALL DONE?
1295 011762 001361 BNE 10$
1296 011764 005303 DEC R3 ;DDNE 50 SEEK CYCLES (100 SEEKS)
1297 011766 001306 BNE LDSEEK ;IF NO BRANCH (KEEP CYCLING)
1298 011770 000605 BR CONTIN ;CHECK SWR FOR CHANGE AND CONTINUE
1299
1300
1301
1302
1303
1304
1305
1306 011772
1307 011772 104401 012000 BAD.IN: TYPE ,65$ ;:TYPE ASCIZ STRING
1308 011776 000401 BR 64$ ;:GET OVER THE ASCIZ
1309
1310 012002 ;:65$: .ASCIZ /?/
1311 012002 64$: SECT.1:
1312 012002 104401 012010 TYPE ,65$ ;:TYPE ASCIZ STRING
1313 012006 000441 BR 64$ ;:GET OVER THE ASCIZ
1314
1315 012112 ;:65$: .ASCIZ <15><12>/FORMATTER-SURFACE VERIFIER,SET SW REG FOR DRV #'S. PRESS CONT./
1316 012112 022737 000176 001140 64$: CMP #SWREG,SWR ;SOFTWARE SWITCH REGISTER IN USE ?
1317 012120 001002 BNE 1$ ;BR IF NOT
1318 012122 104405 GTSWR ;GET SWITCH REGISTER VALUE
1319 012124 000401 BR 2$ ;CONTINUE
1320 012126 000000 1$: HALT ;WAIT FOR 'CONTINUE'
1321 012130 012737 000001 020444 2$: MOV #1,SHFCNT ;SET SHIFT COUNT
```

```
1322 012136 005037 020446 CLR DRVCNT ;CLEAR DRIVE COUNT
1323 012142 032777 020444 166770 S13: BIT SHFCNT,@SWR ;IS THIS SW SET?
1324 012150 001011 BNE S10 ;YES FORMAT THIS DRIVE
1325 012152 006337 020444 S11: ASL SHFCNT
1326 012156 005237 020446 INC DRVCNT
1327 012162 022737 000010 020446 CMP #10,DRVCNT ;ALL DONE?
1328 012170 001561 BEQ GD1
1329 012172 000763 BR S13
1330 012174 013700 020446 S10: MOV DRVCNT,R0
1331 012200 104401 001161 TYPE ,SCRLF
1332 012204 104401 020450 TYPE ,EM1 ;TYPE 'DRIVE'
1333 012210 010046 MOV RO,-(SP) ;TYPE DRIVE #
1334 012212 104402 TYPOC
1335 012214 000300 SWAB RO
1336 012216 006100 RDL RO
1337 012220 006100 ROL RO
1338 012222 006100 ROL RO
1339 012224 006100 ROL RO
1340 012226 006100 ROL RO
1341 012230 010037 001352 MOV RO,@#DSKTMP
1342 012234 012737 000000 001422 MOV #0,ERRWF
1343 012242 012737 000000 001424 MOV #0,ERRRF ;CLEAR OUT ERROR COUNTS.
1344 012250 012737 000000 001426 MOV #0,ERRRFC
1345 012256 012737 000000 001430 MOV #0,ERRWCH
1346 012264 012737 000000 001432 MOV #0,ERRWCS
1347 012272 012737 177750 001416 S12: MOV #-24,RWC ;SET WORD COUNT FOR READ FORMAT.
1348 012300 012737 164000 001412 MOV #-6144,WC ;SET UP WORD COUNT FOR WRITES.
1349 012306 012737 000000 001420 MOV #0,EXTR ;CLEAR EXTRA BIT FOR 12 SECTOR PACK.
1350 012314 012701 177772 COMMON: MOV #-6,R1 ;SET UP LOOP COUNT FOR THE CLEANER.
1351 012320 012777 014500 167050 COM: MOV #14500,@RKDA ;SET UP FOR A SEEK TO 202.
1352 012326 053777 001352 167042 BIT @#DSKTMP,@RKDA
1353 012334 105777 167030 1$: TSTB @RKCS ;IS THE CONTROLLER READY?
1354 012340 100375 BPL 1$ ;NO SO WAIT.
1355 012342 012777 000011 167020 MOV #11,@RKCS ;DO THE SEEK.
1356 012350 032777 000100 167006 2$: BIT #BIT6,@RKDS ;IS THE SEEK DONE?
1357 012356 001774 BEQ 2$ ;NO SO WAIT.
1358 012360 105777 167004 3$: TSTB @RKCS ;IS CONTROLLER READY?
1359 012364 100375 BPL 3$ ;NO
1360 012366 012777 000015 166774 MOV #15,@RKCS ;DO A DRIVE RESET.
1361 012374 032777 000100 166762 4$: BIT #BIT6,@RKDS ;IS DRIVE RESET DONE.
1362 012402 001774 BEQ 4$ ;NO
1363 012404 005201 INC R1 ;COUNT THE CLEANER LOOP.
1364 012406 001344 BNE COM ;MORE TO GO.
1365 012410 012737 000000 001410 MOV #0,DA ;START OUT AT CYL. 0.
1366 012416 012777 177777 166762 NEXT: MOV #177777,@BA ;PUT ALL ONE'S IN BUFFER.
1367 012424 004137 012540 JSR R1,IO ;GO DO THE DISK THING.
1368 012430 012777 000000 166750 MOV #0,@BA ;PUT ALL ZERO'S IN BUFFER.
1369 012436 004137 012540 JSR R1,IO ;GO DO IT AGAIN.
1370 012442 012777 125252 166736 MOV #125252,@BA ;PUT A ALT. PATTERN IN BUFFER.
1371 012450 004137 012540 JSR R1,IO ;ONCE MORE.
1372 012454 005177 166726 COM @BA ;COMPLEMENT THE LAST PATTERN.
1373 012460 004137 012540 JSR R1,IO ;AND AGAIN.
1374 012464 062737 000040 001410 ADD #40,DA ;INCREMENT TO THE NEXT CYL.
1375 012472 022737 014540 001410 CMP #14540,DA ;ARE WE DONE WITH THIS ONE?
1376 012500 001346 BNE NEXT ;NO SO DO THE NEXT CYL.
1377 012502
```

```

1378 012502 104401 012510
1379 012506 000411
1380
1381 012532
1382 012532 000607
1383 012534 000137 001440
1384
1385
1386
1387
1388
1389
1390 012540 013777 001412 166624
1391 012546 013777 001410 166622
1392 012554 053777 001352 166614
1393 012562 013777 001406 166604
1394 012570 012777 000000 166572
1395 012576 052777 006000 166564
1396 012604 052777 000002 166556
1397 012612 053777 001420 166550
1398 012620 052777 000001 166542
1399 012626 105777 166536
1400 012632 100375
1401 012634 005777 166530
1402 012640 100541
1403 012642 012737 000000 001422
1404
1405
1406
1407 012650 013777 001416 166514
1408 012656 013777 001410 166512
1409 012664 053777 001352 166504
1410 012672 013777 001414 166474
1411 012700 012777 000000 166462
1412 012706 052777 002000 166454
1413 012714 052777 000004 166446
1414 012722 053777 001420 166440
1415 012730 052777 000001 166432
1416 012736 105777 166426
1417 012742 100375
1418 012744 032777 040000 166416
1419 012752 001134
1420 012754 012737 000000 001424
1421
1422
1423
1424 012762 013777 001412 166402
1425 012770 013777 001410 166400
1426 012776 053777 001352 166372
1427 013004 013777 001406 166362
1428 013012 012777 000000 166350
1429 013020 052777 004400 166342
1430 013026 053777 001420 166334
1431 013034 052777 000006 166326
1432 013042 052777 000001 166320
1433

```

```

1434
1435
1436 013050 013703 001416
1437 013054 005403
1438 013056 063703 001414
1439 013062 013702 001414
1440 013066 023722 001410
1441 013072 001073
1442 013074 020302
1443 013076 001373
1444 013100 012737 000000 001426
1445
1446
1447
1448 013106 105777 166256
1449 013112 100375
1450 013114 005777 166250
1451 013120 100407
1452 013122 012737 000000 001430
1453 013130 012737 000000 001432
1454 013136 000201
1455 013140 000137 013624
1456
1457
1458
1459 013144 005237 001422
1460 013150 022737 000004 001422
1461 013156 001016
1462 013160
1463 013160 104401 013166
1464 013164 000410
1465
1466 013206
1467 013206 000000
1468 013210 000137 001440
1469 013214 012777 000000 166148
1470 013222 012777 000015 166140
1471 013230 032777 000100 166126
1472 013236 001774
1473 013240 000137 012540
1474
1475
1476
1477 013244 005237 001424
1478 013250 022737 000004 001424
1479 013256 001356
1480 013260 000737
1481
1482
1483
1484 013262 005237 001426
1485 013266 105777 166078
1486 013272 100375
1487 013274 022737 000004 001426
1488 013302 001401
1489 013304 000743

```

```

1490 013306 042777 000037 166062 FAILED: BIC #37,@RKDA ;PUT WHICH SECTORS HEADER
1491 013314 042702 177740 BIC #177740,R2
1492 013320 060277 166052 ADD R2,@RKDA ;FAILED IN RKDA FOR THE MESSAGE.
1493 013324
1494 013324 104401 013332 FAIL:
1495 013330 000417 TYPE ,65$ ;;TYPE ASCIZ STRING
64$: BR 64$ ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <15><12>/PACK FAILED AT (IN OCTAL) /
64$:
1497 013370 MOV @RKDA,R1 ;GENERAT THE CYL,SECTOR,SURFACE
1498 013370 017701 166002 MOV R1,R2 ;MESSAGE FROM RKDA
1499 013374 010102 BIC #177770,R2
1500 013376 042702 177770 ADD #260,R2
1501 013402 062702 000260 MOVB R2,SEC+1
1502 013406 110237 013565 JSR R3,SHF3
1503 013412 004337 013612 BIC #177776,R2
1504 013416 042702 177776 ADD #260,R2
1505 013422 062702 000260 MOVB R2,SEC
1506 013426 110237 013564 JSR R3,SHF1
1507 013432 004337 013616 BIC #177776,R2
1508 013436 042702 177776 ADD #260,R2
1509 013442 062702 000260 MOVB R2,SUR
1510 013446 110237 013576 JSR R3,SHF1
1511 013452 004337 013616 BIC #177770,R2
1512 013456 042702 177770 ADD #260,R2
1513 013462 062702 000260 MOVB R2,CYL+2
1514 013466 110237 013554 JSR R3,SHF3
1515 013472 004337 013612 BIC #177770,R2
1516 013476 042702 177770 ADD #260,R2
1517 013502 062702 000260 MOVB R2,CYL+1
1518 013506 110237 013553 JSR R3,SHF3
1519 013512 004337 013612 BIC #177774,R2
1520 013516 042702 177774 ADD #260,R2
1521 013522 062702 000260 MOVB R2,CYL
1522 013526 110237 013552 TYPE ,1$ ;TYPE OUT THE GENERATED MESSAGE
1523 013532 104401 013542 JMP STOP ;BYPASS THE INLINE MESSAGE
1524 013536 000137 013604
1525 013542 005015 054503 027114 1$: .ASCII <15><12>/CYL. /
1526 013550 020040
1527 013552 030060 020060 CYL: .ASCII /000 /
1528 013556 042523 027103 020040 .ASCII /SEC. /
1529 013564 030060 040 SEC: .ASCII /00 /
1530 013567 123 051125 027106 .ASCII /SURF. /
1531 013574 020040
1532 013576 020060 005015 000 SUR: .ASCIZ /0 /<15><12>
1533 013604 .EVEN
1534 013604 000000 STOP: HALT ;LET OPER DO HIS THING.
1535 013606 000137 001440 JMP START ;RESTART THE TEST.
1536
1537 ;SHIFT SUBROUTINES.
1538
1539 013612 006201 SHF3: ASR R1 ;HERE FOR A SHIFT OF 3.
1540 013614 006201 SHF2: ASR R1 ;HERE FOR A SHIFT OF 2.
1541 013616 006201 SHF1: ASR R1 ;HERE FOR A SHIFT OF 1.
1542 013620 010102 MOV R1,R2 ;PUT RESULTES IN THE WORKING REG.
1543 013622 000203 RTS R3
1544
1545 ;ERRORS FOR WRITE CHECK.
  
```

```

1546
1547 013624 032777 040000 165536 WCERR: BIT #BIT14,@RKCS ;WAS IT A HARD ERROR.
1548 013632 001010 BNE WCHERR ;YES GO PROCESSE IT.
1549 013634 005237 001432 INC ERRWCS ;ADD 1 TO THE SOFT ERROR COUNT.
1550 013640 022737 000004 001432 CMP #4,ERRWCS ;HAS THERE BEEN 4 OF THEM?
1551 013646 001626 BEQ FAIL ;YES PUT OUT FAILED MESSAGE.
1552 013650 000137 012540 JMP ID ;NO SO TRY AGAIN.
1553 013654 005237 001430 WCHERR: INC ERRWCH ;ADD 1 TO THE HARD ERROR COUNT.
1554 013660 022737 000004 001430 CMP #4,ERRWCH ;HAS THERE BEEN 4 OF THEM?
1555 013666 001402 BEQ SYSERR ;YES PUT OUT SYSTEM ERROR MESSAGE.
1556 013670 000137 013214 JMP RETRY ;NO SO TRY AGAIN.
1557 013674 000137 013160 SYSERR: JMP SYSER
1558
1559 ;BUFFERS.
1560
1561 013700 000000 BUFF: .WORD 0
1562 013702 000030 RBUFF: .BLKW 30
1563
1564
1565
1566
1567
1568
1569 .SBTTL RK05 CONTROL PANEL TEST
1570 013762
1571 013762 104401 013770 BAD.ON:
1572 013766 000401 TYPE ,65$ ;;TYPE ASCIZ STRING
64$: BR 64$ ;;GET OVER THE ASCIZ
;;65$: .ASCIZ /?/
64$:
SECT.0:
1574 013772 TYPE ,65$ ;;TYPE ASCIZ STRING
BR 64$ ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <15><12>/RK05 CONTROL PANEL TEST, WHICH DRIVE?/
64$:
1575 013772
1576 013772 104401 014000 RDCHR (SP),RO
1577 013776 000424 MOV MOV #60,RO
1578
1579 014050 TYPOS .BYTE 1
1580 014050 104407 .BYTE 0
1581 014052 011600 SUB #60,RO
1582 014054 104403 BMI BAD.ON
1583 014056 001 CMP #10,RO
1584 014057 000 BLE BAD.ON
1585 014060 162700 000060 MOVB RO,@#DRIVE
1586 014064 100736 SWAB RO
1587 014066 022700 000010 ROL RO
1588 014072 003733 ROL RO
1589 014074 110037 001314 ROL RO
1590 014100 000300 ROL RO
1591 014102 006100 ROL RO
1592 014104 006100 ROL RO
1593 014106 006100 ROL RO
1594 014110 006100 ROL RO
1595 014112 006100 ROL RO
1596 014114 010037 001352 MOV RO,@#DSKTMP
1597 014120 104401 014126 TYPE ,67$ ;;TYPE ASCIZ STRING
BR 66$ ;;GET OVER THE ASCIZ
;;67$: .ASCIZ <15><12>/MOUNT PACK ON DRIVE#/
66$:
1599
1600 014156 MOV DRIVE,-(SP) ;SAVE DRIVE FOR TYPEOUT
1601 014156 013746 001314
  
```

```

1602 014162 104403          TYPOS          ;;GO TYPE--OCTAL ASCII
1603 014164 001            .BYTE 1        ;;TYPE 1 DIGIT(S)
1604 014165 000            .BYTE 0        ;;SUPPRESS LEADING ZEROS
1605 014166 104401 014174  TYPE ,69$      ;;TYPE ASCIZ STRING
1606 014172 000425        BR 68$         ;;GET OVER THE ASCIZ
1607          ;;69$: .ASCIZ <15><12>/PLACE DRIVE IN RUN ;SHOULD SEE THE RUN./
1608 014246          68$:          TYPE ,71$      ;;TYPE ASCIZ STRING
1609 014246 104401 014254  BR 70$         ;;GET OVER THE ASCIZ
1610 014252 000423        ;;71$: .ASCIZ <15><12>/POWER, AND ON CYLINDER LAMPS LIGHT./
1611          70$:          TYPE ,73$      ;;TYPE ASCIZ STRING
1612 014322          BR 72$         ;;GET OVER THE ASCIZ
1613 014322 104401 014330  ;;73$: .ASCIZ <15><12>/MAKE DRIVE WRITE ENABLE PRESS CONTINUE/
1614 014326 000425        72$:          HALT          ;;
1615 014402          JSR PC,WRRDCK  ;;GO WRITE 0'S, RD 0'S, CHECK
1616 014402 000000        TYPE ,75$      ;;TYPE ASCIZ STRING
1617 014402 000000        BR 74$         ;;GET OVER THE ASCIZ
1618 014404 004737        ;;75$: .ASCIZ <15><12><15><12>/WRITE PROTECT THE DRIVE THEN PRESS CONTINUE/
1619 014410 104401 014416  74$:          HALT          ;;
1620 014414 000430        JSR PC,WPRD   ;;GO TRY OVERWRITE
1621          TYPE ,77$      ;;TYPE ASCIZ STRING
1622 014476          BR 76$         ;;GET OVER THE ASCIZ
1623 014476 000000        ;;77$: .ASCIZ <15><12><15><12>/CLEAR WRITE PROTECT THEN PRESS CONTINUE/
1624 014500 004737 164470  76$:          HALT          ;;
1625 014504 104401 014512  JSR PC,WRRDCK  ;;GO WRITE 0'S, RD 0'S, CHECK
1626 014510 000426        MOV @#DSKTMP,@RKDA ;SET UP DRIVE#
1627          MOV #17,@RKCS  ;;FUNCTION WRITE PROTECT
1628 014566          JSR PC,SMTME  ;;GO WAIST TIME FOR RK11-C
1629 014566 000000        TSTB @RKCS    ;;IS CONTROL READY SET
1630 014570 004737 164260  BPL 1$        ;;IF NO, BRANCH
1631 014574 013777 001352 164574 JSR PC,WPRD   ;;GO TRY OVERWRITE OF IERO'S
1632 014602 012777 000017 164560 1$:          PRTTWQ:
1633 014610 004737 005614  TYPE ,65$      ;;TYPE ASCIZ STRING
1634 014614 105777 164550  BR 64$         ;;GET OVER THE ASCIZ
1635 014620 100375        ;;65$: .ASCIZ <15><12><15><12>/CAUTION! TRY TO OPEN THE DOOR, DO NOT FORCE!
1636 014622 004737 164470  64$:          TYPE ,67$      ;;TYPE ASCIZ STRING
1637 014626          BR 66$         ;;GET OVER THE ASCIZ
1638 014626 104401 014634  ;;67$: .ASCIZ <15><12>/DOOR SHOULD NOT OPEN!
1639 014632 000431        66$:          TYPE ,69$      ;;TYPE ASCIZ STRING
1640          BR 68$         ;;GET OVER THE ASCIZ
1641 014716          ;;69$: .ASCIZ <15><12>/PRESS CONTINUE WHEN FINISHED/
1642 014716 104401 014724  68$:          HALT          ;;
1643 014722 000414        JSR PC,WRRDCK  ;;GO WRITE 0'S, RD 0'S, CHECK
1644          MOV #17,@RKCS  ;;FUNCTION WRITE PROTECT
1645 014754          JSR PC,SMTME  ;;GO WAIST TIME FOR RK11-C
1646 014754 104401 014762  TSTB @RKCS    ;;IS CONTROL READY SET
1647 014760 000420        BPL 1$        ;;IF NO, BRANCH
1648          JSR PC,WPRD   ;;GO TRY OVERWRITE OF IERO'S
1649 015022          TYPE ,71$      ;;TYPE ASCIZ STRING
1650 015022 000000        BR 70$         ;;GET OVER THE ASCIZ
1651 015024 104401 015032  ;;71$: .ASCIZ <15><12><15><12>/PUT DRIVE IN LOAD, WAIT FOR LOAD LIGHT/
1652 015030 000426        70$:          TYPE ,73$      ;;TYPE ASCIZ STRING
1653          BR 72$         ;;GET OVER THE ASCIZ
1654 015106          ;;73$: .ASCIZ <15><12>/PRESS CONTINUE WHEN FINISHED/
1655 015106 104401 015114  72$:          HALT          ;;
1656 015112 000420        JSR PC,WRRDCK  ;;GO WRITE 0'S, RD 0'S, CHECK
1657          MOV @#DSKTMP,@RKDA ;SET UP DRIVE#

```

```

1658 015154          72$:          HALT          ;;
1659 015154 000000        MOV @#DSKTMP,@RKDA ;SET UP DISK ADDRESS
1660 015156 013777 001352 164212 MOV #15,@RKCS  ;;ISSUE A DRIVE RESET
1661 015164 012777 000015 164176 JSR PC,SMTME  ;;KILL TIME FOR RK11-C
1662 015172 004737 005614 1$:          TSTB @RKCS    ;;CONTROL READY SET?
1663 015176 105777 164166  BPL 1$        ;;IF NO, BRANCH
1664 015202 100375        BIT #100200,@RKER ;DRE SET
1665 015204 032777 100200 164154 BNE 2$        ;;IF YES BRANCH
1666 015212 001057        TYPE ,75$      ;;TYPE ASCIZ STRING
1667 015214 104401 015222  BR 74$         ;;GET OVER THE ASCIZ
1668 015220 000427        ;;75$: .ASCIZ <15><12>/DRE=BIT15 OF RKER DID NOT SET IF AN RK11-C/
1669          74$:          TYPE ,77$      ;;TYPE ASCIZ STRING
1670 015300          BR 76$         ;;GET OVER THE ASCIZ
1671 015300 104401 015306  ;;77$: .ASCIZ <15><12>/OR BIT07 DID NOT SET IF AN RK11-D/
1672 015304 000422        76$:          BIT #140000,@RKCS ;DID HARD ERROR ON ERROR SET
1673          BNE 3$        ;;BRANCH IF YES
1674 015352          TYPE ,79$      ;;TYPE ASCIZ STRING
1675 015352 032777 140000 164010 BR 78$         ;;GET OVER THE ASCIZ
1676 015360 001015        ;;79$: .ASCIZ <15><12>/ERROR DID NOT SET/
1677 015362 104401 015370  78$:          MOV #1,@RKCS  ;;ISSUE A CONTROL RESET
1678 015366 000412        JSR PC,SMTME  ;;WAIST TIME
1679          TSTB @RKCS    ;;CONTROL READY SET
1680 015414          BPL 4$        ;;IF NO, BRANCH
1681 015414 012777 000001 163746 BIT #BIT15,@RKER ;'DRE' CLEAR
1682 015422 004737 005614  4$:          BEO 5$        ;;IF YES BRANCH
1683 015426 105777 163736  TYPE ,81$      ;;TYPE ASCIZ STRING
1684 015432 100375        BR 80$         ;;GET OVER THE ASCIZ
1685 015434 032777 100000 163724 ;;81$: .ASCIZ <15><12>/CONTROL RESET DID NOT CLEAR 'DRE'/
1686 015442 001425        80$:          MOV #140000,@RKCS ;ERROR BITS CLEAR
1687 015444 104401 015452  BEO X        ;;IF YES BRANCH
1688 015450 000422        TYPE ,83$      ;;TYPE ASCIZ STRING
1689          BR 82$         ;;GET OVER THE ASCIZ
1690 015516          ;;83$: .ASCIZ <15><12>/CONTROL RESET DID NOT CLEAR 'ERROR', RKCS/
1691 015516 032777 140000 183644 82$:          X:
1692 015524 001431        TYPE ,65$      ;;TYPE ASCIZ STRING
1693 015526 104401 015534  BR 64$         ;;GET OVER THE ASCIZ
1694 015532 000426        ;;65$: .ASCIZ <15><12><15><12>/OPEN THE DOOR, PUT DRIVE IN RUN/
1695          64$:          TYPE ,67$      ;;TYPE ASCIZ STRING
1696 015610          BR 66$         ;;GET OVER THE ASCIZ
1697 015610          ;;67$: .ASCIZ <15><12>/CAUTION! IF RUN LIGHT ON ERROR! DEPRESS/
1698 015610 104401 015616  66$:          TYPE ,69$      ;;TYPE ASCIZ STRING
1699 015614 000422        BR 68$         ;;GET OVER THE ASCIZ
1700          ;;69$: .ASCIZ <15><12>/LOAD IMMEDIATELY, CONTINUE WHEN FINISHED/
1701 015662          68$:          HALT          ;;
1702 015662 104401 015670  TYPE ,65$      ;;TYPE ASCIZ STRING
1703 015666 000425        BR 64$         ;;GET OVER THE ASCIZ
1704          ;;65$: .ASCIZ <15><12><15><12>/REMOVE THE PACK, CLOSE THE DOOR/
1705 015742          64$:          TYPE ,67$      ;;TYPE ASCIZ STRING
1706 015742 104401 015750  BR 66$         ;;GET OVER THE ASCIZ
1707 015746 000426        ;;67$: .ASCIZ <15><12>/CAUTION! IF RUN LIGHT ON ERROR! DEPRESS/
1708          66$:          TYPE ,69$      ;;TYPE ASCIZ STRING
1709 016024          BR 68$         ;;GET OVER THE ASCIZ
1710 016024 000000        XX:          HALT          ;;
1711 016026 104401 016034  TYPE ,65$      ;;TYPE ASCIZ STRING
1712 016032 000422        BR 64$         ;;GET OVER THE ASCIZ
1713          ;;65$: .ASCIZ <15><12><15><12>/REMOVE THE PACK, CLOSE THE DOOR/

```

```
1714 016100 64$: TYPE ,67$ ;;TYPE ASCIZ STRING
1715 016100 104401 016106 BR 66$ ;;GET OVER THE ASCIZ
1716 016104 000423 ;;67$: .ASCIZ <15><12>/PUT DRIVE IN RUN, DRIVE SHOULD NOT/
1717 66$:
1718 016154 TYPE ,69$ ;;TYPE ASCIZ STRING
1719 016154 104401 016162 BR 68$ ;;GET OVER THE ASCIZ
1720 016160 000423 ;;69$: .ASCIZ <15><12>/RUN...INTERCLOCKS HAVE BEEN CHECKED/
1721 68$:
1722 016230 TYPE ,71$ ;;TYPE ASCIZ STRING
1723 016230 104401 016236 BR 70$ ;;GET OVER THE ASCIZ
1724 016234 000407 ;;71$: .ASCIZ <15><12>/DONEI /
1725 70$:
1726 016254 WRRDCK: JMP @#START
1727 016254 000137 001440 CLR @#PATTRN ;MAKE A PATTERN OF ZERO'S
1728 016260 005037 001362 MOV @#DSKTMP,@RKDA ;SET UP DRIVE ADDRESS
1729 016264 013777 001352 163104 MOV #1,@RKCS ;ISSUE A CONTROL RESET
1730 016272 012777 000001 163070 JSR PC,SMTME
1731 016300 004737 005614 JSR @RKCS ;CONTROL READY SET
1732 016304 105777 163060 5$: TSTB ;IF NO BRANCH
1733 016310 BPL 5$ ;SET UP DRIVE ADDRESS
1734 016312 013777 001352 163056 MOV @#DSKTMP,@RKDA ;GET BUSS ADDRESS
1735 016320 012777 001362 163046 MOV #PATTRN,@RKBA ;WORD COUNT 1 SECTOR
1736 016326 013777 001344 163036 MOV @#SECCNT,@RKWC ;IBA + WRITE + GO
1737 016334 013777 001354 163026 MOV @#WRITCS,@RKCS ;KILL TIME FOR RK11-C
1738 016342 004737 005614 JSR PC,SMTME ;CONTROL READY SET
1739 016346 105777 163016 1$: TSTB @RKCS ;IF NO, BRANCH
1740 016352 100375 BPL 1$ ;SET UP RK REGISTERS
1741 016354 013777 001352 163014 MOV @#DSKTMP,@RKDA ;TO READ ONE SECTOR
1742 016362 012777 007356 163004 MOV #RDBUFF,@RKBA ;TO THE READ BUFFER
1743 016370 013777 001344 162774 MOV @#SECCNT,@RKWC
1744 016376 013777 001356 162764 MOV @#READCS,@RKCS
1745 016404 004737 005614 JSR PC,SMTME ;KILL TIME FOR RK11-C
1746 016410 105777 162754 2$: TSTB @RKCS ;CONTROL READY SET
1747 016414 100375 BPL 2$ ;IF NO, BRANCH
1748 016416 012704 007356 MOV #RDBUFF,R4 ;GET BUFFER TO R4
1749 016422 005005 CLR R5 ;SET UP TO COMPARE
1750 016424 020524 3$: CMP R5,(R4)+ ;FOR ZERO'S
1751 016426 001414 BEQ 4$ ;IF OK, BRANCH
1752 016430 104401 016438 TYPE ,65$ ;;TYPE ASCIZ STRING
1753 016434 000410 BR 64$ ;;GET OVER THE ASCIZ
1754 ;;65$: .ASCIZ <15><12>/WRITE FAILED/
1755 64$:
1756 016456 BR WRRDCK ;GO BACK TRY AGAIN
1757 016460 022704 010356 4$: CMP #MANSEL,R4 ;DONE ALL CHECKS
1758 016464 001357 BNE 3$ ;IF NO, BRANCH
1759 016466 000207 RTS PC ;IF YES, RETURN
1760 016470 032777 000040 162666 WRPRO: BIT #BIT5,@RKDS ;BIT 5 ON
1761 016476 001021 BNE 1$ ;IF YES, BRANCH
1762 016500 104401 016506 TYPE ,65$ ;;TYPE ASCIZ STRING
1763 016504 000416 BR 64$ ;;GET OVER THE ASCIZ
1764 ;;65$: .ASCIZ <15><12>/WPS=BITS OF RKDS NOT SET/
1765 64$:
1766 016542 1$ MOV #177777,@PATTRN ;GO LOAD ALL
1767 016550 013777 001352 162620 MOV @#DSKTMP,@RKDA ;RK REGISTERS
1768 016556 012777 001362 162610 MOV #PATTRN,@RKBA ;TO WRITE
1769 016564 013777 001344 162600 MOV @#SECCNT,@RKWC ;ALL ONES (WITH WRITE LOCK)
```

```
1770 016572 013777 001354 162570 MOV @#WRITCS,@RKCS ;OVER THE ZERO'S
1771 016600 004737 005614 JSR PC,SMTME ;KILL TIME FOR RK11-C
1772 016604 105777 162560 2$: TSTB @RKCS ;CONTROL READY SET?
1773 016610 00375 BPL 2$ ;IF NO, BRANCH
1774 016612 032777 020000 162548 BIT #BIT13,@RKER ;WLO BIT SET
1775 016620 001032 BNE 3$ ;IF YES BRANCH
1776 016622 104401 016630 TYPE ,67$ ;;TYPE ASCIZ STRING
1777 016626 000427 BR 66$ ;;GET OVER THE ASCIZ
1778 ;;67$: .ASCIZ <15><12>/EXPECTED WLO=BIT13 OF RKER BUT DID NOT SET/
1779 66$:
1780 016706 012777 000001 162454 3$: MOV #1,@RKCS ;DD A CONTROL RESET
1781 016714 004737 005614 JSR PC,SMTME ;KILL TIME FOR RK11-C
1782 016720 105777 162444 4$: TSTB @RKCS ;CONTROL READY SET?
1783 016724 100375 BPL 4$ ;IF NO BRANCH
1784 016726 032777 020000 162432 BIT #BIT13,@RKER ;WLO BIT CLEAR
1785 016734 001431 BEQ RDCHK0 ;IF YES, BRANCH
1786 016736 104401 016744 TYPE ,69$ ;;TYPE ASCIZ STRING
1787 016742 000426 BR 68$ ;;GET OVER THE ASCIZ
1788 ;;69$: .ASCIZ <15><12>/CONTROL RESET DID NOT CLEAR 'WLO' OF RKER/
1789 68$:
1790 017020 RDCHK0: MOV @#DSKTMP,@RKDA ;SET UP RK REGISTERS
1791 017026 013777 001352 162350 MOV #RDBUFF,@RKBA ;TO READ SECTOR 0,
1792 017034 013777 001344 162330 MOV @#SECCNT,@RKWC ;CYLINDER 0, HEAD 0
1793 017042 013777 001356 162320 MOV @#READCS,@RKCS ;TO ENSURE NO WRITE TOOK PLACE
1794 017050 004737 005614 JSR PC,SMTME ;KILL TIME
1795 017054 105777 162310 3$: TSTB @RKCS
1796 017060 100375 BPL 3$
1797 017062 012703 000005 MOV #5,R3
1798 017066 012704 007356 MOV #RDBUFF,R4 ;CHECK TO INSURE NO WRITE
1799 017072 005005 CLR R5 ;TOOK PLACE
1800 017074 020524 1$: CMP R5,(R4)+ ;WITH WRITE LOCK
1801 017076 001474 BEQ 2$
1802 017100 005303 DEC R3 ;DEC THE ERROR COUNT
1803 017102 001475 BEQ 4$ ;IF ZERO BRANCH
1804 017104 104401 017112 TYPE ,65$ ;;TYPE ASCIZ STRING
1805 017110 000422 BR 64$ ;;GET OVER THE ASCIZ
1806 ;;65$: .ASCIZ <15><12>/WRITE OCCURRED WITH WRITE PROTECT/
1807 64$:
1808 017156 TST -(R4)
1809 017160 104401 017168 TYPE ,67$ ;;TYPE ASCIZ STRING
1810 017164 000410 BR 66$ ;;GET OVER THE ASCIZ
1811 ;;67$: .ASCIZ <15><12>/BUFFER ADDR=/
1812 66$:
1813 017206 MOV R4,-(SP)
1814 017210 104403 TYPOS
1815 017212 006 .BYTE 6
1816 017213 001 .BYTE 1
1817 017214 104401 017222 TYPE ,69$ ;;TYPE ASCIZ STRING
1818 017220 000406 BR 68$ ;;GET OVER THE ASCIZ
1819 ;;69$: .ASCIZ / EXPCTD=/
1820 68$:
1821 017236 MOV R5,-(SP)
1822 017240 104404 TYPON
1823 017242 104401 017250 TYPE ,71$ ;;TYPE ASCIZ STRING
1824 017246 000406 BR 70$ ;;GET OVER THE ASCIZ
1825 ;;71$: .ASCIZ / RECDV=/
```

```

1826 017264          70$:   MOV      (R4)+,-(SP)
1827 017264 012446   TYPON
1828 017266 104404   2$:   CMP      #MANSEL,R4      ;FINISHED ALL CHECKS
1829 017270 022704 010356   BNE     1$                ;IF NO, BRANCH
1830 017274 001277   4$:   RTS      PC              ;RETURN
1831 017276 000207
1832
1833
1834
1835
  
```

```

1836          ;THE FOLLOWING REVISION WAS MADE BY JIM KAPADIA
1837
1838          .SBTTL CONTROL PANEL TEST # 2
1839
1840          ;THIS IS THE ENTRY POINT INTO CONTROL PANEL TEST #2. ALL
1841          ;THE DRIVES THAT ARE PRESENT AND IN 'RDY' CONDITION ARE
1842          ;REPORTED (ON LINE).
1843
1844 017300 000240   SECT.4: NOP
1845 017302 012777 000001 162060   MOV      #1,@RKCS
1846 017310 105777 162054   3$:   TSTB   @RKCS
1847 017314 100375   BPL     3$
1848 017316 012700 020614   MOV      #DRIVO,R0
1849 017322 005001   CLR     R1
1850 017324 005002   CLR     R2
1851
1852 017326 010210   1$:   MOV      R2,(R0)      ;SET UP ADDRESS TABLE
1853 017330 010277 162042   MOV      R2,@RKDA     ;ADDRESS THE DRIVE
1854 017334 105777 162024   TSTB   @RKDS         ;IS IT PRESENT?
1855 017340 100021   BPL     2$           ;NO
1856
1857 017342 104401 020450   TYPE    ,EM1         ;TYPE 'DRIVE'
1858 017346 010146   MOV      R1,-(SP)    ;TYPE OUT DRIVE #
1859 017350 104402   TYPON
1860 017352 104401 020461   TYPE    ,EM2         ;TYPE 'ON LINE'
1861 017356 052710 000300   BIS     #BIT6+BIT7,(R0) ;SET BITS INDICATING THIS
1862          ;DRIVE PRESENT
1863
1864 017362 012777 000015 162000   MOV      #15,@RKCS   ;ISSUE A DRIVE RESET
1865 017370 004737 005614   JSR     PC,SMTME     ;ALLOW SOME TIME
1866 017374 032777 000100 161782   4$:   BIT     #RWS,@RKDS  ;WAIT FOR RWS RDY
1867 017402 001774   BEQ     4$
1868
1869 017404 005720   2$:   TST     (R0)+
1870 017406 005201   INC     R1
1871 017410 062702 020000   ADD     #20000,R2    ;NXT DRIVE
1872 017414 001344   BNE     1$           ;ALL DONE?
1873
1874 017416 104401 001161   TYPE    ,SCLF
1875
1876          ;THIS CODE CHECKS THE CONDITION OF 'DRY' BIT IN RKDS FOR EVERY
1877          ;DRIVE. IF 'DRY' IS SET DRIVE IS SAID TO BE 'ON LINE', OTHERWISE IT
1878          ;IS OFFLINE. IF THE 'DRY' BIT HAS CHANGED FROM LAST TIME, THEN
1879          ;IT IS REPORTED. IF THERE IS NO CHANGE NOTHING IS REPORTED.
1880
1881 017422 012700 020614   BEGCT: MOV      #DRIVO,R0      ;INITIALIZE POINTERS
1882 017426 005001   CLR     R1
1883
1884 017430 011077 161742   BEGCT1: MOV      (R0),@RKDA    ;ADDRESS A DRIVE
1885 017434 042777 017777 161734   BIC     #17777,@RKDA ;MASK OUT NON DR# BITS
1886 017442 105777 161716   TSTB   @RKDS         ;IS THIS DRIVE ON LINE?
1887 017446 100044   BPL     1$           ;NO
1888          ;YES
1889 017450 105710   TSTB   (R0)         ;WAS IT 'ON LINE' LAST TIME?
1890 017452 100454   BMI     NXT1        ;YES, NO MESSAGE TO REPORT
1891 017454 052710 000200   BIS     #BIT7,(R0)   ;IT CHANGED FROM OFF LINE TO ON
  
```

```

1892 017460 104401 020450          TYPE      ,EM1          ;LINE, REPORT MESSAGE
1893 017464 010146          MOV       R1,-(SP)
1894 017466 104402          TYPOC
1895 017470 104401 020461          TYPE      ,EM2          ;TYPE 'ON LINE'
1896 017474 032777 000040 161662  BIT       #WPS,@RKDS    ;WRITE ENABLED?
1897 017502 001417          BEQ      2$           ;YES, OK
1898 017504 104401 017512          TYPE      ,65$        ;TYPE ASCIZ STRING
1899 017510 000414          BR       64$         ;GET OVER THE ASCIZ
1900                                ;:65$: .ASCIZ <15><12>/EROR,NOT WRT ENABLED/
1901 017542                                64$:
1902 017542 012777 000017 161620  MOV       #17,@RKCS    ;WRITE PROT THE DISK
1903 017550 105777 161614          TSTB    @RKCS
1904 017554 100375          BPL     3$           ;
1905 017556 000412          BR      NXT1
1906
1907 017560 105710          1$: TSTB    (R0)       ;WAS THIS DRIVE OFF LINE LAST
1908 017562 100010          BPL     NXT1         ;TIME? BRNCH IF YES
1909 017564 104401 020450          TYPE      ,EM1          ;IF NOT, REPORT THE CHANGE
1910 017570 010146          MOV     R1,-(SP)    ;TYPE DRIVE #
1911 017572 104402          TYPOC
1912 017574 104401 020473          TYPE      ,EM3          ;TYPE 'OFF LINE'
1913 017600 042710 000200          BIC     #BIT7,(R0)   ;CLEAR BIT TO INDICATE THIS
1914                                ;DRIVE 'OFF LINE'
1915
1916                                ;THIS CODE CHECKS 'WPS' BIT FOR EVERY DRIVE THAT IS IN 'DRY'
1917                                ;CONDITION (ON LINE). IT REPORTS ANY CHANGE IN THE CONDITION OF
1918                                ;THE 'WPS' BIT. IF THERE WAS NO CHANGE FROM LAST TIME NOTHING
1919                                ;IS REPORTED. AT THE TIME OF ENTRY R0 POINTS TO DRIVE FLAG.
1920
1921 017604 105777 161554          NXT1: TSTB    @RKDS    ;IS THIS DRIVE PRESENT?
1922                                ;RKDA CONTAINS THE DRV #
1923 017610 100033          BPL     NXT2         ;NO, SKIP CHECKING
1924
1925 017612 032777 000040 161544  BIT       #WPS,@RKDS    ;WPS BIT SET?
1926 017620 001014          BNE     1$           ;YES
1927
1928 017622 032710 000004          BIT     #BIT2,(R0)    ;WPS BIT CLEAR
1929 017626 001424          BEQ     NXT2         ;WAS IT CLR LAST TIME ALSO?
1930                                ;YES, NOTHING TO REPORT.
1931 017630 104401 020450          TYPE      ,EM1          ;WPS CHANGED FROM 'SET'
1932 017634 010146          MOV     R1,-(SP)    ;TO 'CLR', REPORT IT
1933 017636 104402          TYPOC              ;TYPE DRIVE #
1934 017640 042710 000004          BIC     #BIT2,(R0)   ;INDICATE THAT 'WPS' IS CLEAR
1935 017644 104401 020521          TYPE      ,EM5          ;TYPE 'WPS CLEAR'
1936 017650 000413          BR      NXT2
1937
1938 017652 032710 000004          1$: BIT     #BIT2,(R0) ;WPS BIT IS SET
1939 017656 001010          BNE     NXT2         ;WAS IT SET LAST TIME ALSO?
1940                                ;YES, NOTHING TO REPORT.
1941                                ;WPS CHANGED, FROM 'CLR' TO
1942                                ;'SET', REPORT THIS CHANGE
1943 017660 104401 020450          TYPE      ,EM1          ;TYPE 'DRIVE'
1944 017664 010146          MOV     R1,-(SP)    ;TYPE DRIVE #
1945 017666 104402          TYPOC
1946 017670 104401 020506          TYPE      ,EM4          ;TYPE 'WPS SET'
1947 017674 052710 000004          BIS     #BIT2,(R0)   ;SET FLAG BIT INDICATING WPS SET
    
```

```

1948                                ;THIS CODE PERFORMS A SEEK FUNCTION ON A DRIVE AND CHECKS IF
1949                                ;THE 'DPL' BIT SET AS A RESULT. (IF THE POWER WAS CUT OFF
1950                                ;FROM THE DRIVE). NOTE THAT ONLY THOSE DRIVES ARE
1951                                ;CHECKED WHICH WERE FOUND TO BE PRESENT AT BEGINNING (WHEN
1952                                ;THIS TEST WAS ENTERED). SEEK IS DONE TO CYLINDER 1.
1953                                ;AT THE TIME OF ENTRY R0 POINTS TO THE DRIVE FLAG.
1954
1955 017700 032710 000100          NXT2: BIT     #BIT6,(R0) ;WAS THIS DRIVE PRESENT AT BEGNO
1956 017704 001403          BEQ     4$           ;NO
1957 017706 105777 161452          TSTB    @RKDS        ;IS IT PRESENT NOW?
1958 017712 100402          BMI     3$           ;YES
1959 017714 000137 020352          4$: JMP     DN1DRV    ;IF NOT SKIP THIS CHECK
1960
1961 017720 052777 000040 161450  3$: BIS     #40,@RKDA   ;RKDA ALREADY HAS THE DRV #
1962                                ;SET CYL 1 ADDRESS
1963 017726 012777 000011 161434  MOV     #11,@RKCS    ;SEEK, GO
1964
1965 017734 105777 161430          1$: TSTB    @RKCS        ;WAIT FOR CONTROL RDY?
1966 017740 100375          BPL     1$           ;SOMETHING WRONG IF CNTAL RDY
1967                                ;DOES NOT COME BACK
1968 017742 032777 010000 161414  BIT     #DPL,@RKDS    ;DPL BIT SET?
1969 017750 001414          BEQ     2$           ;NO
1970                                ;YES, DPL SET
1971 017752 032710 000001          BIT     #BIT0,(R0)   ;WAS 'DPL' SET LAST TIME ALSO?
1972 017756 001167          BNE     CLRDPPL     ;YES, NOTHING TO REPORT.
1973                                ;DPL CHANGED, GOT SET THIS
1974                                ;TIME, REPORT IT
1975 017760 104401 020450          TYPE      ,EM1          ;
1976 017764 010146          MOV     R1,-(SP)    ;
1977 017766 104402          TYPOC
1978 017770 104401 020537          TYPE      ,EM6          ;TYPE 'POWER LO'
1979 017774 052710 000001          BIS     #BIT0,(R0)   ;SET FLAG BIT INDICATING THAT
1980                                ;DPL SET THIS TIME
1981                                ;'DPL' BIT IS CLEAR
1982 020002 032710 000001          2$: BIT     #BIT0,(R0) ;WAS 'DPL' CLEAR LAST TIME ALSO?
1983 020006 001410          BEQ     WATSK       ;YES, NOTHING TO REPORT
1984                                ;REPORT THAT 'DPL' BIT CHANGED,
1985                                ;FROM SET TO CLEAR
1986 020010 104401 020450          TYPE      ,EM1          ;
1987 020014 010146          MOV     R1,-(SP)    ;
1988 020016 104402          TYPOC
1989 020020 104401 020560          TYPE      ,EM7          ;TYPE 'POWER UP'
1990 020024 042710 000001          BIC     #BIT0,(R0)   ;SET FLAG BIT INDICATING THAT DPL
1991                                ;IS CLEAR THIS TIME
1992
1993                                ;THIS CODE WAITS FOR THE SEEK (DONE ABOVE) TO FINISH. WAITING
1994                                ;TIME IS APPROX. 50 MS (FOR THE WORST CASE). IF R/W/S RDY
1995                                ;DOES NOT SET WITHIN 50 MS, THEN IT IS ASSUMED THAT A 'SIN'
1996                                ;IS POSSIBLE AND THE PROGRAM WAITS FOR 1450 MS MORE, SO THAT
1997                                ;THE 'SIN' CAN SET. IF 'SIN' DOES NOT SET WITHIN THIS
1998                                ;TIME AN ERROR IS REPORTED:
1999                                ; SIN DIDN'T OCCUR
2000                                ; IF R/W/S RDY SETS WITHIN 50 MS THE PROGRAM PROCEEDS TO
2001                                ;CHECK THE NEXT DRIVE.
2002 020030 012705 164220          WATSK: MOV     #-6000.,R5 ;SET COUNT TO WAIT FOR
2003                                ;50 MS
    
```



```

2004 020034 032777 000100 161322 1$: BIT #RWS,@RKDS ;R/W/S. RDY SET?
2005 020042 001042 BNE 3$ ;YES
2006 020044 005205 INC R5 ;WAIT
2007 020046 001372 BNE 1$ ;50 MS OVER. R/W/S RDY
2008 ;DIDN'T SET. WAIT FOR
2009 ;SIN TO SET.
2010
2011 020050 005004 CLR R4
2012 020052 012705 MOV #17777,R5 ;SET UP COUNT
2013 020056 032777 001000 161300 2$: BIT #SIN,@RKDS ;SIN SET?
2014 020064 001045 BNE SIN$ ;YES
2015 020066 005305 DEC R5 ;WAIT
2016 020070 001372 BNE 2$
2017 020072 005704 TST R4
2018 020074 001002 BNE 4$
2019 020076 005204 INC R4
2020 020100 000766 BR 2$
2021 ;1500 MS ELAPSED. BUT SIN
2022 ;DIDN'T SET. ERROR!
2023
2024 020102 104401 020110 4$: TYPE ,65$ ;;TYPE ASCIZ STRING
2025 020106 000415 BR 64$ ;;GET OVER THE ASCIZ
2026 ;;65$: .ASCIZ <15><12>/SIN DIDN'T OCCUR, DRVE/
2027 020142 64$: MOV R1,-(SP) ;TYPE DRIVE #
2028 020142 010146 TYPOC
2029 020144 104402 BR DN1DRV
2030 020146 000501 BIT #BIT1,(R0) ;DID R/W/S RDY SET LAST TIME?
2031 020150 032710 000002 3$: BEQ DN1DRV ;YES
2032 020154 001476 BIC #BIT1,(R0) ;CLR FLAG INDICATING THAT SEEK IS OK
2033 020156 042710 000002 TYPE ,EM1 ;REPORT THAT SEEK IS OK
2034 020162 104401 020450 MOV R1,-(SP)
2035 020166 010146 TYPOC
2036 020170 104402 TYPE ,EM9
2037 020172 104401 020601 BR DN1DRV
2038 020176 000465
2039 ;IF SIN SET, DO DRIVE RESET AND CLEAR IT
2040
2041
2042 020200 032710 000002 SIN$: BIT #BIT1,(R0) ;DID 'SIN' SET LAST TIME ALSO?
2043 020204 001010 BNE 4$ ;YES, NOTHING TO REPORT
2044 020206 052710 000002 BIS #BIT1,(R0) ;SET FLAG INDICATING THAT
2045 020212 104401 020450 TYPE ,EM1 ;'SIN' SET, AND REPORT THE CHANGE
2046 020216 010146 MOV R1,-(SP)
2047 020220 104402 TYPOC
2048 020222 104401 020573 TYPE ,EMB ;TYPE 'SIN'
2049
2050 020226 017705 161144 4$: MOV @RKDA,R5 ;SAVE RKDA
2051 020232 012777 000001 161130 MOV #1,@RKCS ;DO CONTROL RESET
2052 020240 105777 161124 1$: TSTB @RKCS ;WAIT FOR CONTROL RDY
2053 020244 100375 BPL 1$
2054 020246 010577 161124 MOV R5,@RKDA
2055 020252 012777 000015 161110 MOV #15,@RKCS ;DO DRIVE RESET. RKDA
2056 ;ALREADY HAS THE DRIVE #
2057 020260 105777 161104 2$: TSTB @RKCS ;WAIT FOR CNTRL RDY
2058 020264 100375 BPL 2$
2059 020266 005005 CLR R5
    
```

```

2060 020270 032777 000100 181066 3$: BIT #RWS,@RKDS ;R/W/S SET?
2061 020276 001025 BNE DN1DRV ;YES
2062 020300 005205 INC R5
2063 020302 001372 BNE 3$ ;WAIT FOR R/W/S RDY
2064 ;REPORT ERROR. R/W/S RDY CLR
2065 020304 104401 020312 TYPE ,65$ ;;TYPE ASCIZ STRING
2066 020310 000411 BR 64$ ;;GET OVER THE ASCIZ
2067 ;;65$: .ASCIZ <15><12>/RWS RDY NOT SET/
2068 020334 64$: BR DN1DRV
2069
2070 020334 000406
2071 ;IF DPL SET CLEAR THE ERROR BY DOING CONTROL RESET.
2072
2073
2074 020336 012777 000001 181024 CLR$PL: MOV #1,@RKCS ;CONTROL RESET
2075 020344 105777 161020 1$: TSTB @RKCS ;WAIT FOR CNTRL RDY
2076 020350 100375 BPL 1$
2077 ;AT THIS STAGE THE DRIVE (# IN RKDA) HAS BEEN CHECKED
2078 ;FOR DRY, WPS, DPL, & SIN. THE POINTERS ARE INCREMENTED
2079 ;AND THE SAME CHECKS WILL BE DONE ON THE NEXT
2080 ;DRIVE & THEN THE NEXT ONE & SO ON. NOTE THAT
2081 ;THIS SUB-PROGRAM KEEPS ON CYCLING THROUGH
2082 ;ALL THE DRIVES. AT THE TIME OF ENTRY (HERE)
2083 ;RO POINTS TO THE FLAG FOR THE DRIVE THAT WAS
2084 ;JUST CHECKED. BEFORE GOING ON TO THE NEXT
2085 ;DRIVE THE HEADS ARE BROUGHT BACK TO CYLINDER
2086 ;0 (FOR THE NEXT CYCLE).
2087
2088 020352 011077 161020 DN1DRV: MOV (R0),@RKDA ;GET DRIVE #
2089 020356 105777 161002 TSTB @RKDS ;DRIVE PRESENT?
2090 020362 100017 BPL 3$ ;NO
2091 020364 042777 017777 161004 BIC #17777,@RKDA ;CYL ADRES = 0
2092 020372 012777 000011 160770 MOV #11,@RKCS ;GO. SEEK
2093
2094 020400 105777 160764 1$: TSTB @RKCS ;WAIT FOR CNTRL RDY
2095 020404 100375 BPL 1$
2096
2097 020406 004737 005614 JSR PC,SMTME
2098 020412 032777 000100 160744 4$: BIT #RWS,@RKDS
2099 020420 001774 BEQ 4$
2100
2101 020422 005720 3$: TST (R0)+ ;INCREMENT POINTERS TO
2102 020424 005201 INC R1 ;NEXT DRIVE
2103 020426 020127 000010 CMP R1,#10 ;ALL DONE, THIS CYCLE?
2104 020432 001402 BEQ 2$ ;YES
2105 020434 000137 017430 JUMP BEGCT1 ;GO DO NEXT DRIVE
2106 020440 000137 017422 2$: JUMP BEGCT ;RESTART THE CYCLE OVER
2107 ;AGAIN
2108
2109
2110
2111 020444 000000 SHFCNT: .WORD 0
2112 020446 000000 DRVCNT: .WORD 0
2113 ;MESSAGES
2114 020450 005015 051104 053111 EM1: .ASCIZ <15><12>/DRIVE /
2115 020456 020105 000
    
```

```

2116 020461 040 047440 020116 EM2: .ASCIZ / ON LINE/
2117 020466 044514 042516 000
2118 020473 040 047440 043106 EM3: .ASCIZ / OFF LINE/
2119 020500 046040 047111 000105
2120 020506 020040 051127 020124 EM4: .ASCIZ / WRT PROT/
2121 020514 051120 052117 000
2122 020521 040 053440 052122 EM5: .ASCIZ / WRT ENABLED/
2123 020526 042440 040516 046102
2124 020534 042105 000
2125 020537 040 042040 044522 EM6: .ASCIZ / DRIVE POWER LO/
2126 020544 042526 050040 053517
2127 020552 051105 046040 000117
2128 020560 020040 047520 042527 EM7: .ASCIZ / POWER UP/
2129 020566 020122 050125 000
2130 020573 040 051440 047111 EM8: .ASCIZ / SIN/
2131 020600 000
2132 020601 040 051440 042505 EM9: .ASCIZ / SEEK OK/
2133 020606 020113 046517 000
    
```

```

2134
2135 ;DRIVE FLAGS FOR CONTROL PANEL TEST #2
2136 ;BITS 15,14,13 GIVE THE DRIVE NO (EX: 0,1,2,---)
2137 ;BIT 7 IS SET WHEN 'DRY' BIT IS SET FOR THE DRIVE (ON LINE)
2138 ;BIT 7 IS CLEAR WHEN 'DRY' IS CLEAR (WHEN DRIVE IS IN LOAD/OFF LINE.
2139 ;DRIVE POWER IS CUT OFF)
2140 ;BIT 6 IS SET IF A DRIVE IS FOUND TO BE PRESENT (DRY) AT
2141 ;THE BEGINING. UNLIKE BIT 7 THIS BIT DOES NOT GET SET OR
2142 ;CLEARED AS THE DRIVE CONDITIONS CHANGE. IT JUST INDICATES
2143 ;THAT THE DRIVE IS AVAILABLE FOR CHECKING.
2144 ;BIT 0 IS SET IF 'DPL' BIT GETS SET, IE: DRIVE POWER OFF
2145 ;BIT 0 IS CLEARED WHEN DRIVE POWER IS ON.
2146 ;BIT 1 IS SET WHEN SEEK INCOMPLETE 'SIN' OCCURS.
2147 ;BIT 1 IS CLEAR WHEN SEEK IS OK.
2148 ;BIT 2 IS SET WHEN WRT PROT IS SET FROM CONSOLE.
2149 ;BIT 2 IS CLEARED WHEN DRIVE IS WRITE ENABLED.
    
```

```

2150
2151 020614 .EVEN
2152 020614 000000 DRIV0: .WORD 0 ;DRIVE FLAGS
2153 020616 000000 DRIV1: .WORD 0
2154 020620 000000 DRIV2: .WORD 0
2155 020622 000000 DRIV3: .WORD 0
2156 020624 000000 DRIV4: .WORD 0
2157 020626 000000 DRIV5: .WORD 0
2158 020630 000000 DRIV6: .WORD 0
2159 020632 000000 DRIV7: .WORD 0
2160
2161
    
```

```

2162 .SBTTL HEAD ALIGNMENT ROUTINE
2163
2164 ;HEAD ALIGNMENT ROUTINE
2165 ;THIS MAINTAINANCE ROUTINE IS HELPFUL IN HEAD ALIGNMENT. UPON ENTRY
2166 ;THE QUESTION -- DRIVE? -- IS ASKED , THE USER SHOULD REPLY WITH THE
2167 ;DRIVE NUMBER THAT IS TO BE ALIGNED. IF THE DRIVE IS AN RK-05F
2168 ;THE LETTER 'F' IS ADDED AS A SUFFIX. FOR SELECTING SURFACE 0
2169 ;PUT SW0=0, FOR SELECTING SURFACE 1 PUT SW0=1. SET SW1 =1 TO SELECT
2170 ;CYLINDER 64. SET SW1=0 TO SELECT CYLINDER 105.
2171 ;IF THE DRIVE IS AN RK-05F, CYLINDER 64 BECOMES CYLINDER 130
2172 ;OF THE EVEN DRIVE, AND CYLINDER 105 BECOMES CYLINDER 5 OF THE ODD DRIVE
2173 ;THE HEADS ARE PLACED ON THE SELECTED CYLINDER AND DATA IS READ
2174 ;CONTINUOSLY FROM THE CYLINDER (SECTOR 0)
2175 ; THE UPPER OR LOWER HEAD AND CYLINDER CAN BE SELECTED
2176 ;DYNAMICALLY, IE. THE PROGRAM DOES NOT HAVE TO BE STOPPED TO SELECT THE
2177 ;UPPER OR LOWER HEAD OR CYLINDER.
2178 ;IN ORDER TO SELECT ANOTHER DRIVE, PUT ANY SWITCH FROM SW2 TO SW15 UP AND
2179 ;THE PROGRAM WILL AGAIN ASK THE QUESTION (DRIVE?).
2180
2181 020634 000240 SECT.5: NOP
2182 020636 104401 020644 TYPE ,65$ ;:TYPE ASCIZ STRING
2183 020642 000426 BR 64$ ;:GET OVER THE ASCIZ
2184
2185 020720 ;:65$: .ASCIZ <15><12>/SET SW0=0 FOR SURFACE 0, SW0=1 FOR SUR 1./
2186 020720 104401 020726 64$: TYPE ,67$ ;:TYPE ASCIZ STRING
2187 020724 000432 BR 66$ ;:GET OVER THE ASCIZ
2188
2189 021012 ;:67$: .ASCIZ <15><12>/SET SW1=0 FOR CYLINDER 105, SW1=1 FOR CYLINDER 64/
2190 021012 104401 021020 66$: TYPE ,69$ ;:TYPE ASCIZ STRING
2191 021016 000427 BR 68$ ;:GET OVER THE ASCIZ
2192
2193 021076 ;:69$: .ASCIZ <15><12>/PUT ANY SW FROM 2-15 HI TO SELECT NEW DRIVE/
2194 021076 104401 021532 68$: HDALGN: TYPE ,EM10 ;:ASK FOR DRIVE #
2195 021102 005037 021526 CLR FFLAG ;:FLAG FOR RK-05F
2196 021106 104410 RDLIN ;:GET OPR INPUT
2197 021110 012601 MOV (SP)+,R1 ;:ADDR OF COMMAND STRING
2198 021112 112100 MOVB #60,R0 ;:FIRST CHAR
2199 021114 162700 SUB #60,R0 ;:0 TO 7
2200 021120 002766 BLT HDALGN ;:TOO SMALL
2201 021122 022700 000067 CMP #67,R0 ;:MUST BE 7 OR LESS
2202 021126 002763 BLT HDALGN ;:TOO BIG
2203 021130 000241 CLC
2204 021132 006000 ROR RO
2205 021134 006000 ROR RO
2206 021136 006000 ROR RO
2207 021140 006000 ROR RO
2208 021142 010037 020614 MOV RO,DRIVO ;:ADDRESS OF DRIVE
2209 021146 112100 MOVB (R1)+,R0 ;:NEXT INPUT CHAR
2210 021150 001412 BEQ 5$ ;:ALL DONE IF C.R.
2211 021152 020027 000106 CMP RO,#'F ;:IS IT F?
2212 021156 001347 BNE HDALGN ;:NO, SO ERROR
2213 021160 105711 TSTB (R1) ;:NEXT CHAR MUST BE C.R.
2214 021162 001345 BNE HDALGN ;:ELSE, ERROR
2215 021164 042737 020000 020614 BIC #BIT13,DRIVO ;:USE EVEN DRIVE IF RK-05F
2216 021172 005237 021526 INC FFLAG ;:SHOW F TYPE DRIVE
2217 021176 013700 020614 5$: MOV DRIVO,R0 ;:DRIVE ADDR TO R0
    
```

2218	021202	022737	000176	001140	CMP	#SWREG,SWR	;SOFTWARE SWITCH REGISTER IN USE?
2219	021210	001002			BNE	15\$	;BR IF NOT
2220	021212	104405			GTSWR		;REQUEST NEW CONTENTS FOR SWITCH REG
2221	021214	000401			BR	16\$	;CONTINUE
2222	021216	000000			HALT		;WAIT FOR OPERATOR TO ENTER NEW SWR VALUE
2223	021220	017737	157714	021530	16\$: MOV	@SWR,SWTCH	;HOLD SWITCHES
2224	021226	042737	177775	021530	BIC	#177775,SWTCH	;WANT SW1 ONLY
2225	021234	001005			BNE	7\$	;SW1 SET, SO LOW CYLINDER
2226	021236	005737	021526		TST	FFLAG	;F DRIVE?
2227	021242	001402			BEQ	7\$	;NO
2228	021244	052700	020000		BIS	#BIT13,R0	;ODD DRIVE IF HIGH TRACK OF F
2229	021250	010077	160122		7\$: MOV	R0,@RKDA	;ADDRESS DRIVE
2230	021254	012777	000017	160108	MOV	#17,@RKCS	;WRITE PROTECT
2231	021262	105777	160102		8\$: TSTB	@RKCS	
2232	021266	100375			BPL	8\$	;WAIT FOR DRIVE READY
2233	021270	012777	000001	160072	MOV	#1,@RKCS	;RESET CONTROLLER
2234	021276	105777	160066		9\$: TSTB	@RKCS	
2235	021302	100375			BPL	9\$	;WAIT FOR READY
2236	021304	005737	021526		TST	FFLAG	;F DRIVE?
2237	021310	001410			BEQ	13\$	;NO
2238	021312	012701	000240		MOV	#5.*40,R1	;TRACK 5 OF HIGH
2239	021316	005737	021530		TST	SWTCH	;SW1 SET?
2240	021322	001412			BEQ	10\$	;YES SO TEST TRACK B OF DRIVE HIGH
2241	021324	062701	010100		ADD	#130.*40,R1	;TRACK 130. IF SW1 SET
2242	021330	000407			BR	10\$	
2243	021332	012701	004000		13\$: MOV	#64.*40,R1	;CYLINDER 64 IF NOT F
2244	021336	005737	021530		TST	SWTCH	;SW1 SET?
2245	021342	001002			BNE	10\$	;YES, SO CYLINDER 64
2246	021344	062701	002440		ADD	#41.*40,R1	;CYLINDER 105
2247	021350	005777	160014		10\$: TST	@RKCS	;ANY ERROR?
2248	021354	100006			BPL	11\$	;NO, CONTINUE
2249	021356	012777	000001	160004	MOV	#1,@RKCS	;RESET
2250	021364	105777	160000		12\$: TSTB	@RKCS	
2251	021370	100375			BPL	12\$	;WAIT FOR READY
2252	021372	017702	157542		11\$: MOV	@SWR,R2	;SWITCH REG TO R2
2253	021376	042702	177775		BIC	#177775,R2	;SW1 ONLY
2254	021402	020237	021530		CMP	R2,SWTCH	;ANY CHANGE SINCE LAST?
2255	021406	001273			BNE	5\$	;YES, GO SET-UP ADDR AGAIN
2256	021410	010077	157762		8\$: MOV	R0,@RKDA	;ADDRESS THE DRIVE
2257	021414	012777	000017	157746	MOV	#17,@RKCS	;WRITE PROTECT THE DRIVE
2258	021422	105777	157742		TSTB	@RKCS	;WAIT FOR CONTROL RDY
2259	021426	100375			BPL	.-4	
2260	021430	042700	000020		4\$: BIC	#20,R0	;CLEAR TRACK ADDR
2261	021434	032777	000001	157476	BIT	#1,@SWR	;SNO SET?
2262	021442	001402			BEQ	2\$	;NO TEST TRACK 0
2263	021444	052700	000020		BIS	#20,R0	;TEST TRACK 1
2264	021450	042700	017740		2\$: BIC	#17740,R0	;CLEAR CYLINDER ADDR
2265	021454	050100			BIS	R1,R0	;PUT CYLINDER ADDR IN ADDR
2266	021456	010077	157714		MOV	R0,@RKDA	;ADRES THE DRIVE
2267	021462	012777	177400	157702	MOV	#-400,@RKWC	;READ 1 SECTOR
2268	021470	012777	007356	157676	MOV	#RDBUFF,@RKBA	;INTO THIS BUFFER
2269	021476	012777	000005	157664	MOV	#5,@RKCS	;READ, GO
2270							
2271	021504	105777	157660		3\$: TSTB	@RKCS	;DONE?
2272	021510	100375			BPL	3\$	;NO
2273							

2274	021512	032777	177774	157420	BIT	#177774,@SWR	;EXIT OUT?
2275	021520	001713			BEQ	10\$	;NO, CONTINUE ON THIS DRIVE
2276	021522	000137	021076		JMP	HDALGN	;YES, GET NEW DRIVE
2277							
2278	021526	000000			FFLAG:	0	
2279	021530	000000			SWTCH:	0	
2280	021532	005015	051104	053111	EM10:	.ASCIZ <15><12>/DRIVE?/	
2281	021540	037505	000				
2282							
2283							
2284							
2285							
2286							
2287							
2288							
2289							
2290							
2291							
2292							
2293							
2294							
2295							
2296							
2297							
2298							
2299							
2300		021544					
2301							
2302	021544	005000			SECT.8:	CLR R0	;INITIALIZE DRIVE #
2303	021546	005037	020614		CLR	DRIVO	
2304	021552	010077	157620		8\$: MOV	R0,@RKDA	;IS IT PRESENT?
2305	021556	105777	157602		TSTB	@RKDS	;IF NOT SKIP
2306	021562	100406			BMI	12\$	
2307	021564	005237	020614		10\$: INC	DRIVO	
2308	021570	062700	020000		ADD	#20000,R0	
2309	021574	001366			BNE	8\$	
2310	021576	000762			BR	SECT.6	
2311	021600	013746	020614		12\$: MOV	DRIVO,-(SP)	;GET DRIVE #
2312	021604	032777	000040	157552	BIT	#40,@RKDS	
2313	021612	001406			BEQ	9\$	
2314	021614	104401	022234		TYPE	,EM11	
2315	021620	104402			TYPE	,EM12	
2316	021622	104401	022244		TYPE	10\$	
2317	021626	000756			BR	,EM1	
2318	021630	104401	020450		9\$: MOV	DRIVO,-(SP)	
2319	021634	013746	020614		TYPOC		
2320	021640	104402			MOV	#1,@RKCS	;CONTROL RESET
2321	021642	012777	000001	157520	TSTB	@RKCS	
2322	021650	105777	157514		BPL	.-4	
2323	021654	100375			CLR	R5	;INITIALIZE PATTERN TO BE WRITTEN
2324	021656	005005					;0 ON CYL 0, 1 ON CYL 1. ETC
2325							
2326	021660	013701	001376		MOV	RKDA,R1	
2327	021664	013702	001372		MOV	RKWC,R2	
2328	021670	013703	001374		MOV	RKBA,R3	
2329	021674	013704	001370		MOV	RKCS,R4	

```

2330 021700 010011          MOV    RO,@R1
2331 021702 010537 022232 1$:  MOV    R5,BUFR          ;FILL THE PATTERN IN DATA BUFFER.
2332 021706 012713 022232  MOV    #BUFR,@R3        ;BUS ADRES
2333 021712 012712 164000  MOV    #-14000,@R2      ;WRITE 1 CYL (256X12X2 WORDS),
2334 021716 012714 004003  MOV    #4003,@R4        ;WRITE, GO, IBA SET
2335 021722 105714          TSTB   @R4              ;WAIT FOR CONTROL READY
2336 021724 100376          BPL    .-2
2337                          ;DONE
2338 021726 005205          INC    R5              ;WRITTEN ALL 15 CYLINDERS?
2339 021730 020527 000020  CMP    R5,#20          ;IF NOT, GO BAK
2340 021734 001362          BNE    1$
2341                          ;
2342 021736 010005          MOV    RO,R5           ;DRIVE #
2343 021740 052705 000500  BIS    #500,R5         ;CYL 10
2344 021744 012737 000012 022232  MOV    #12,BUFR        ;PATTERN TO BE WRITTEN
2345 021752 010511          MOV    R5,@R1         ;ADRES THE DISK
2346 021754 012712 164000  MOV    #-14000,@R2     ;WORD COUNT= 1 CYLINDER
2347 021760 012713 022232  MOV    #BUFR,@R3      ;BUS ADRES
2348 021754 012714 004003  MOV    #4003,@R4      ;WRITE, GO, IBA
2349 021770 032777 000100 157366 2$:  BIT    #RMS,@RKDS     ;WAIT FOR THE HEADS TO SETTLE
2350 021776 001774          BEQ    2$             ;DN CYL 10
2351 022000 104401 022006  TYPE   ,65$           ;;TYPE ASCIZ STRING
2352 022004 000406          BR     64$           ;;GET OVER THE ASCIZ
2353                          ;
2354 022022          ;;65$: .ASCIZ /DROPP POWER/
2355 64$:
2356 022022 000407          BR     5$
2357 022024 010511          MOV    R5,@R1         ;ADRES THE DISK
2358 022026 012712 164000  MOV    #-14000,@R2     ;WORD COUNT= 1 CYLINDER
2359 022032 012713 022232  MOV    #BUFR,@R3      ;BUS ADRES
2360 022036 012714 004003  MOV    #4003,@R4      ;WRITE, GO, IBA
2361 022042 105714          TSTB   @R4           ;WAIT FOR CONTROL READY
2362 022044 100376          BPL    .-2
2363 022046 005714          TST    @R4           ;WAIT FOR DRIVE POWER TO GO DOWN,
2364                          ;OTHERWISE, KEEP ON WRITING ON CYL 10
2365 022050 100365          BPL    3$
2366                          ;
2367                          ;IF DRIVE POWER LOSS WAS SENSED,
2368 022052 104401 022060  TYPE   ,67$           ;ASK TO PUT POWER ON.
2369 022056 000406          BR     66$           ;;TYPE ASCIZ STRING
2370                          ;;GET OVER THE ASCIZ
2371 022074          ;;67$: .ASCIZ <15><12>/POWER ON/
2372 022074 105777 157264 66$:  TSTB   @RKDS         ;WAIT FOR DRIVE READY
2373 022100 100375          BPL    .-4
2374 022102 012714 000001  MOV    #1,@R4         ;CONTR: RESET, CLEAR ERROR
2375 022106 105714          TSTB   @R4
2376 022110 100376          BPL    .-2
2377 022112 010077 157260  MOV    RO,@RKDA
2378 022116 005005          CLR    R5             ;INITIALIZE PATTERN
2379 022120 010537 022232 6$:  MOV    R5,BUFR
2380 022124 012713 022232  MOV    #BUFR,@R3
2381 022130 012712 164000  MOV    #-14000,@R2
2382 022134 012714 004007  MOV    #4007,@R4      ;WRITE CHECK, GO, IBA
2383 022140 105714          TSTB   @R4
2384 022142 100376          BPL    .-2
2385

```

```

2386 022144 005714          TST    @R4           ;ANY ERROR?
2387 022146 100023          BPL    7$            ;NO
2388 022150 104401 022156  TYPE   ,69$           ;TYPE ASCIZ STRING
2389 022154 000416          BR     68$           ;GET OVER THE ASCIZ
2390                          ;
2391 022212          ;;69$: .ASCIZ <15><12>/ERROR, ON PWR-UP. RKDA=/
2392 022212 011146          68$:  MOV    @R1,-(SP)
2393 022214 104402          TPOC
2394 022216 005205          7$:  INC    R5
2395 022220 020527 000020  CMP    R5,#20
2396 022224 001335          BNE    6$
2397
2398 022226 000137 021564          JMP    10$
2399
2400 022232 000000          BUFR: .WORD 0
2401 022234 051104 053111 020105 EM11: .ASCIZ /DRIVE /
2402 022242 000040          EM12: .ASCIZ /IS WRITE PROTECTED /
2403 022244 051511 053440 044522
2404 022252 042524 050040 047522
2405 022260 042524 052103 042105
2406 022266 000040
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426 022270 105737 001157          $TYPE: TSTB   $TPFLG   ;:IS THERE A TERMINAL?
2427 022274 100002          BPL    1$           ;:BR IF YES
2428 022276 000000          HALT
2429 022300 000407          BR     3$           ;:HALT HERE IF NO TERMINAL
2430 022302 010046          BR     3$           ;:LEAVE
2431 022304 017600 000002 1$:  MOV    RO,-(SP)     ;:SAVE RO
2432 022310 112046          MOV    @2(SP),RO    ;:GET ADDRESS OF ASCIZ STRING
2433 022312 001005          MOVB   (RO)+,-(SP)  ;:PUSH CHARACTER TO BE TYPED ONTO STACK
2434 022314 005726          BNE    4$           ;:BR IF IT ISN'T THE TERMINATOR
2435 022316 012600          TST    (SP)+        ;:IF TERMINATOR POP IT OFF THE STACK
2436 022320 062716 000002 60$:  MOV    (SP)+,RO    ;:RESTORE RO
2437 022324 000002          ADD    #2,(SP)      ;:ADJUST RETURN PC
2438 022326 122716 000011          RTI
2439 022332 001430          CMPB   #HT,(SP)    ;:RETURN
2440 022334 122716 000200          BEQ    8$           ;:BRANCH IF <HT>
2441 022340 001006          CMPB   #CRLF,(SP)  ;:BRANCH IF NOT <CRLF>
2442                          BNE    5$

```



```

2554 022733 000          $OFILL: .BYTE 0          ;;ZERO FILL SWITCH
2555 022734 000000      $OMODE: .WORD 0         ;;NUMBER OF DIGITS TO TYPE
2556                      .SBTTL TTY INPUT ROUTINE
2557
2558                      ;*****
2559                      .ENABL LSB
2560 022736 000000      $TKCNT: .WORD 0          ;;NUMBER OF ITEMS IN QUEUE
2561 022740 000000      $TKQIN: .WORD 0          ;;INPUT POINTER
2562 022742 000000      $TKQOUT: .WORD 0         ;;OUTPUT POINTER
2563 022744 000001      $TKQSR: .BLKB 1         ;;TTY KEYBOARD QUEUE
2564                      $TKQEND=.
2565 022745
2566                      .EVEN
2567
2568                      ;*TK INITIALIZE ROUTINE
2569                      ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
2570                      ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
2571                      ;
2572                      ;*CALL:
2573                      ;* JSR PC,$TKINT
2574                      ;* RETURN
2575
2576 022746 005037 022736 $TKINT: CLR $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
2577 022752 012737 022744 022740 MOV $TKQSR,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
2578 022760 013737 022740 022742 MOV $TKQIN,$TKQOUT ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
2579 022766 012737 023016 000060 MOV $TKSRV,$TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
2580 022774 012737 000200 000062 MOV #200,$TKVEC+2 ;;"BR" LEVEL 4
2581 023002 005777 156140 TST @TKB          ;;CLEAR DONE FLAG
2582 023006 012777 000100 156130 MOV #100,@STKS ;;ENABLE TTY KEYBOARD INTERRUPT
2583 023014 000207 PC RTS PC          ;;RETURN TO CALLER
2584
2585                      ;*TK SERVICE ROUTINE
2586                      ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
2587                      ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
2588                      ;*IT IN THE QUEUE.
2589                      ;
2590 023016 117746 156124 $TKSRV: MOVB @TKB,-(SP) ;;PICKUP THE CHARACTER
2591 023022 042716 177600 BIC #'C177,(SP) ;;STRIP THE JUNK
2592 023026 021627 000007 1$: CMP (SP),#7 ;;IS IT A CONTROL G?
2593 023032 001004 BNE 2$ ;;BRANCH IF NO
2594 023034 022737 000176 001140 CMP #SWREG,SWR ;;IS SOFT-SWR SELECTED?
2595 023042 001500 BEQ 6$ ;;GO TO SWR CHANGE
2596
2597 023044 022737 000001 022736 2$: CMP #1,$TKCNT ;;IS THE QUEUE FULL?
2598 023052 001004 BNE 3$ ;;BRANCH IF NO
2599 023054 104401 024066 TYPE $BELL ;;RING THE TTY BELL
2600 023060 005726 TST (SP)+ ;;CLEAN CHARACTER OFF OF STACK
2601 023062 000451 BR 5$ ;;EXIT
2602 023064 021627 000023 3$: CMP (SP),#23 ;;IS IT A CONTROL-S?
2603 023070 001021 BNE 32$ ;;BRANCH IF NO
2604 023072 005077 156046 CLR @STKS ;;DISABLE TTY KEYBOARD INTERRUPTS
2605 023076 005726 TST (SP)+ ;;CLEAN CHAR OFF STACK
2606 023100 105777 156040 31$: TSTB @STKS ;;WAIT FOR A CHAR
2607 023104 100375 BPL 31$ ;;LOOP UNTIL ITS THERE
2608 023106 117746 156034 MOVB @TKB,-(SP) ;;GET THE CHARACTER
2609 023112 042716 177600 BIC #'C177,(SP) ;;MAKE IT 7-BIT ASCII

```

```

2610 023116 022627 000021 CMP (SP)+,#21 ;;IS IT A CONTROL-Q?
2611 023122 001366 BNE 31$ ;;BRANCH IF NO
2612 023124 012777 000100 156012 MOV #100,@STKS ;;REENABLE TTY KEYBOARD INTERRUPTS
2613 023132 000002 RTI ;;RETURN
2614 023134 005237 022736 32$: INC $TKCNT ;;COUNT THIS CHARACTER
2615 023140 021627 000140 CMP (SP),#140 ;;IS IT UPPER CASE?
2616 023144 002405 BLT 4$ ;;BRANCH IF YES
2617 023146 021627 000175 CMP (SP),#175 ;;IS IT A SPECIAL CHAR?
2618 023152 003002 BGT 4$ ;;BRANCH IF YES
2619 023154 042716 000040 BIC #40,(SP) ;;MAKE IT UPPER CASE
2620 023160 112677 177554 4$: MOVB (SP)+,@TKQIN ;;AND PUT IT IN QUEUE
2621 023164 005237 022740 INC $TKQIN ;;UPDATE THE POINTER
2622 023170 023727 022740 022745 CMP $TKQIN,$TKQEND ;;GO OFF THE END?
2623 023176 001003 BNE 5$ ;;BRANCH IF NO
2624 023200 012737 022744 022740 5$: MOV $TKQSR,$TKQIN ;;RESET THE POINTER
2625 023208 000002 RTI ;;RETURN
2626
2627                      ;*****
2628                      ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2629                      ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2630                      ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
2631                      ;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
2632 023210 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED
2633 023216 001104 BNE 15$ ;;EXIT IF NOT
2634 023220 105777 155720 TSTB @STKS ;;IS A CHAR WAITING?
2635 023224 100101 BPL 15$ ;;IF NOT, EXIT
2636 023226 117746 155714 MOVB @TKB,-(SP) ;;YES
2637 023232 042716 177600 BIC #'C177,(SP) ;;MAKE IT 7-BIT ASCII
2638 023236 021627 000007 CMP (SP),#7 ;;IS IT A CONTROL-G?
2639 023242 001300 BNE 2$ ;;IF NOT, PUT IT IN THE TTY QUEUE
2640                      ;AND EXIT
2641
2642                      ;*****
2643                      ;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
2644                      ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
2645                      ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
2646 023244 123727 001134 000001 6$: CMBP $AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
2647 023252 001674 BEQ 2$ ;;BRANCH IF YES
2648 023254 005726 TST (SP)+ ;;CLEAN CONTROL-G OFF STACK
2649 023256 004737 022746 JSR PC,$TKINT ;;FLUSH THE TTY INPUT QUEUE
2650 023262 005077 155656 CLR @STKS ;;DISABLE TTY KEYBOARD INTERRUPTS
2651 023266 112737 000001 001135 MOVB #1,$INTAG ;;SET INTERRUPT MODE INDICATOR
2652
2653 023274 104401 024077 $GTSWR: TYPE $CNTLG ;;ECHO THE CONTROL-G ("G")
2654 023300 104401 024104 TYPE $MSWR ;;TYPE CURRENT CONTENTS
2655 023304 013746 000176 MOV SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
2656 023310 104402 TYPEPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2657 023312 104401 024115 TYPE $MNEW ;;PROMPT FOR NEW SWR
2658 023316 005046 CLR -(SP) ;;CLEAR COUNTER
2659 023320 005046 CLR -(SP) ;;THE NEW SWR
2660 023322 105777 155616 7$: TSTB @STKS ;;CHAR THERE?
2661 023326 100375 BPL 7$ ;;IF NOT TRY AGAIN
2662
2663 023330 117746 155612 MOVB @TKB,-(SP) ;;PICK UP CHAR
2664 023334 042716 177600 BIC #'C177,(SP) ;;MAKE IT 7-BIT ASCII
2665

```

```

2666
2667
2668 023340 021627 000025      9$:  CMP      (SP),#25      ;;IS IT A CONTROL-U?
2669 023344 001005                BNE      10$           ;;BRANCH IF NOT
2670 023346 104401 024072        TYPE    ,%CNTLU      ;;YES, ECHO CONTROL-U (^U)
2671 023352 062706 000006      20$:  ADD      #8,SP      ;;IGNORE PREVIOUS INPUT
2672 023356 000757                BR       19$          ;;LET'S TRY IT AGAIN
2673
2674
2675 023360 021627 000015      10$:  CMP      (SP),#15      ;;IS IT A <CR>?
2676 023364 001022                BNE      16$          ;;BRANCH IF NO
2677 023366 005766 000004        TST     4(SP)         ;;YES, IS IT THE FIRST CHAR?
2678 023372 001403                BEQ     11$          ;;BRANCH IF YES
2679 023374 016677 000002 155536  MOV     2(SP),@SWR    ;;SAVE NEW SWR
2680 023402 062706 000006      11$:  ADD      #6,SP      ;;CLEAR UP STACK
2681 023406 104401 001161      14$:  TYPE    ,%CRLF      ;;ECHO <CR> AND <LF>
2682 023412 123727 001135 000001  CMPB   $INTAG,#1    ;;RE-ENABLE TTY KBD INTERRUPTS?
2683 023420 001003                BNE     15$          ;;BRANCH IF NOT
2684 023422 012777 000100 155514  MOV     #100,@$TKS  ;;RE-ENABLE TTY KBD INTERRUPTS
2685 023430 000002                RTI     ;           ;;RETURN
2686 023432 004737 022440      16$:  JSR     PC,$TYPEPC  ;;ECHO CHAR
2687 023436 021627 000060        CMP     (SP),#60    ;;CHAR < 0?
2688 023442 002420                BLT     18$          ;;BRANCH IF YES
2689 023444 021627 000067        CMP     (SP),#67    ;;CHAR > ??
2690 023450 003015                BGT     18$          ;;BRANCH IF YES
2691 023452 042726 000060        BIC     #60,(SP)+   ;;STRIP-OFF ASCII
2692 023456 005766 000002        TST     2(SP)       ;;IS THIS THE FIRST CHAR
2693 023462 001403                BEQ     17$          ;;BRANCH IF YES
2694 023464 006316                ASL     (SP)        ;;NO, SHIFT PRESENT
2695 023466 006316                ASL     (SP)        ;; CHAR OVER TO MAKE
2696 023470 006316                ASL     (SP)        ;; ROOM FOR NEW ONE.
2697 023472 005266 000002      17$:  INC     2(SP)       ;;KEEP COUNT OF CHAR
2698 023476 056616 177776        BIS     -2(SP), (SP) ;;SET IN NEW CHAR
2699 023502 000707                BR      7$          ;;GET THE NEXT ONE
2700 023504 104401 001160      18$:  TYPE    ,%SQUES     ;;TYPE ?<CR><LF>
2701 023510 000720                BR      20$          ;;SIMULATE CONTROL-U
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777

```

```

;*****
;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
;CALL:
;*
;* RDCHR
;* RETURN HERE
;*
;
;RDCHR: MOV      (SP),-(SP)      ;;PUSH DOWN THE PC AND
        MOV     4(SP),2(SP)     ;;THE PS
        CLR     4(SP)          ;;GET READY FOR A CHARACTER
        CLR     -(SP)         ;;PUT NEW PS ON STACK
        MOV     #64$,-(SP)    ;;PUT NEW PC ON STACK
        RTI     ;             ;;POP NEW PC AND PS
84$:
1$:    TST     $TKCNT         ;;WAIT ON A CHARACTER
        BEQ     1$

```

```

2722 023544 005337 022736      DEC     $TKCNT      ;;DECREMENT THE COUNTER
2723 023550 117766 177166 000004  MOVVB  @STKQOUT,4(SP) ;;GET ONE CHARACTER
2724 023556 005237 022742        INC     $TKQOUT    ;;UPDATE THE POINTER
2725 023562 023727 022742 022745  CMP     $TKQOUT,$TKQEND ;;DID IT GO OFF OF THE END?
2726 023570 001003                BNE     2$          ;;BRANCH IF NO
2727 023572 012737 022744 022742  MOV     #$TKQSRST,$TKQOUT ;;RESET THE POINTER
2728 023600 000002      2$:  RTI     ;           ;;RETURN
;*****
;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
;CALL:
;*
;* RDLIN
;* RETURN HERE
;*
;RDLIN: MOV     R3,-(SP)      ;;SAVE R3
        CLR     -(SP)        ;;CLEAR THE RUBOUT KEY
1$:    MOV     #$TTYIN,R3    ;;GET ADDRESS
2$:    CMP     #$TTYIN+30,R3 ;;BUFFER FULL?
        BLOS   4$          ;;BR IF YES
        RDCHR ;           ;;GO READ ONE CHARACTER FROM THE TTY
10$:  MOVVB  (SP)+,(R3)      ;;GET CHARACTER
        CMPB  #177,(R3)    ;;IS IT A RUBOUT
        BNE   5$          ;;BR IF NO
        TST  (SP)         ;;IS THIS THE FIRST RUBOUT?
        BNE   6$          ;;BR IF NO
        MOVVB #'\,$      ;;TYPE A BACK SLASH
        TYPE ,9$
8$:    MOV     #-1,(SP)     ;;SET THE RUBOUT KEY
        DEC  R3           ;;BACKUP BY ONE
        CMP  R3,$TTYIN   ;;STACK EMPTY?
        BLO  4$          ;;BR IF YES
        MOVVB (R3),9$    ;;SETUP TO TYPEDOUT THE DELETED CHAR.
        TYPE ,9$
        BR   2$          ;;GO TYPE
5$:    TST  (SP)         ;;RUBOUT KEY SET?
        BEQ  7$          ;;BR IF NO
        MOVVB #'\,$      ;;TYPE A BACK SLASH
        TYPE ,9$
7$:    CMPB  #25,(R3)     ;;IS CHARACTER A CTRL U?
        BNE  8$          ;;BR IF NO
        TYPE ,%CNTLU     ;;TYPE A CONTROL "U"
        BR   1$          ;;GO START OVER
8$:    CMPB  #22,(R3)     ;;IS CHARACTER A "R"?
        BNE  3$          ;;BRANCH IF NO
        CLRB (R3)        ;;CLEAR THE CHARACTER
        TYPE ,%CRLF      ;;TYPE A "CR" & "LF"
        TYPE , $TTYIN    ;;TYPE THE INPUT STRING
        BR   2$          ;;GO PICKUP ANOTHER CHACTER
4$:    TYPE ,%SQUES     ;;TYPE A '?'
        BR   1$          ;;CLEAR THE BUFFER AND LOOP
3$:    MOVVB (R3),9$    ;;ECHO THE CHARACTER
        TYPE ,9$
        CMPB #19,(R3)+   ;;CHECK FOR RETURN
        BNE  2$          ;;LOOP IF NOT RETURN
        CLRB -1(R3)     ;;CLEAR RETURN (THE 1$)

```

```

2778 024004 104401 001162          TYPE      ,SLF          ;;TYPE A LINE FEED
2779 024010 005726          TST        (SP)+      ;;CLEAN RUBOUT KEY FROM THE STACK
2780 024012 012603          MOV        (SP)+,R3  ;;RESTORE R3
2781 024014 011646          MOV        (SP)-,(SP) ;;ADJUST THE STACK AND PUT ADDRESS OF THE
2782 024016 016666 000004 000002  MOV        4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
2783 024024 012766 024036 000004  MOV        #TTYIN,4(SP)
2784 024032 000002          RTI              ;;RETURN
2785 024034 000          9S: .BYTE      0          ;;STORAGE FOR ASCII CHAR. TO TYPE
2786 024035 000          .BYTE      0          ;;TERMINATOR
2787 024036 000030          $TTYIN: .BLKB     30    ;;RESERVE 30 BYTES FOR TTY INPUT
2788 024066 177607          $BELL: .ASCIZ <207><377><377> ;;CODE FOR BELL
2789 024072 052536 005015 000          $CNTLW: .ASCIZ /"U/<15><12>    ;;CONTROL "U"
2790 024077 136 006507 000012 000012  $CNTLG: .ASCIZ /"G/<15><12>    ;;CONTROL "G"
2791 024104 005015 020122 053523  $MSWR: .ASCIZ <15><12>/SWR = /
2792 024112 020075 000
2793 024115 040 047040 053505  $MNEW: .ASCIZ / NEW = /
2794 024122 036440 000040
2795
2796          .SBTTL READ AN OCTAL NUMBER FROM THE TTY
2797
2798          ;*****
2799          ;THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2800          ;CHANGE IT TO BINARY.
2801          ;CALL:
2802          ;*   RDOCT          ;;READ AN OCTAL NUMBER
2803          ;*   RETURN HERE   ;;LOW ORDER BITS ARE ON TOP OF THE STACK
2804          ;*                   ;;HIGH ORDER BITS ARE IN $HI OCT
2805 024126 011646          $RDOCT: MOV        (SP)-,(SP) ;;PROVIDE SPACE FOR THE
2806 024130 016666 000004 000002  MOV        4(SP),2(SP) ;;INPUT NUMBER
2807 024136 010046          MOV        R0,-(SP)    ;;PUSH R0 ON STACK
2808 024140 010146          MOV        R1,-(SP)    ;;PUSH R1 ON STACK
2809 024142 010246          MOV        R2,-(SP)    ;;PUSH R2 ON STACK
2810 024144 104410          1$: RDLIN          ;;READ AN ASCII LINE
2811 024146 012600          MOV        (SP)+,R0    ;;GET ADDRESS OF 1ST CHARACTER
2812 024150 005001          CLR        R1          ;;CLEAR DATA WORD
2813 024152 005002          CLR        R2
2814 024154 112046          2$: MOVB       (R0)+,-(SP) ;;PICKUP THIS CHARACTER
2815 024156 001412          BEQ        3$          ;;IF ZERO GET OUT
2816 024160 006301          ASL        R1          ;;*2
2817 024162 006102          ROL        R2
2818 024164 006301          ASL        R1          ;;*4
2819 024166 006102          ROL        R2
2820 024170 006301          ASL        R1          ;;*8
2821 024172 006102          ROL        R2
2822 024174 042716 177770          BIC        #'C7,(SP)    ;;STRIP THE ASCII JUNK
2823 024200 062601          ADD        (SP)+,R1    ;;ADD IN THIS DIGIT
2824 024202 000764          BR         2$          ;;LOOP
2825 024204 005726          3$: TST        (SP)+    ;;CLEAN TERMINATOR FROM STACK
2826 024206 010166 000012 024226  MOV        R1,12(SP)    ;;SAVE THE RESULT
2827 024212 010237 024226          MOV        R2,$HI OCT
2828 024216 012602          MOV        (SP)+,R2    ;;POP STACK INTO R2
2829 024220 012601          MOV        (SP)+,R1    ;;POP STACK INTO R1
2830 024222 012600          MOV        (SP)+,R0    ;;POP STACK INTO R0
2831 024224 000002          RTI              ;;RETURN
2832 024226 000000          $HI OCT: WORD     0    ;;HIGH ORDER BITS GO HERE
2833          .SBTTL TRAP DECODER
    
```

```

2834
2835          ;*****
2836          ;THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
2837          ;AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
2838          ;OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
2839          ;GO TO THAT ROUTINE.
2840
2841 024230 010046          $TRAP: MOV        R0,-(SP) ;;SAVE R0
2842 024232 016600 000002  MOV        2(SP),R0    ;;GET TRAP ADDRESS
2843 024236 005740          TST        -(R0)      ;;BACKUP BY 2
2844 024240 111000          MOVB       (R0),R0    ;;GET RIGHT BYTE OF TRAP
2845 024242 006300          ASL        R0          ;;POSITION FOR INDEXING
2846 024244 016000 024264  MOV        $TRPAD(R0),R0 ;;INDEX TO TABLE
2847 024250 000200          RTS        R0          ;;GO TO ROUTINE
2848
2849
2850          ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
2851
2852 024252 011646          $TRAP2: MOV        (SP)-,(SP) ;;MOVE THE PC DOWN
2853 024254 016666 000004 000002  MOV        4(SP),2(SP) ;;MOVE THE PSW DOWN
2854 024262 000002          RTI              ;;RESTORE THE PSW
2855
2856          .SBTTL TRAP TABLE
2857
2858          ;THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
2859          ;BY THE "TRAP" INSTRUCTION.
2860
2861          ;
2862          ; ROUTINE
2863 024264 024252          $TRPAD: .WORD     $TRAP2
2864 024266 022270          $TYPE      ;;CALL=TYPE      TRAP+1(104401) TTY TYPEOUT ROUTINE
2865 024270 022534          $TYPOC    ;;CALL=TYPOC    TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
2866 024272 022510          $TYPOS    ;;CALL=TYPOS    TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
2867 024274 022550          $TYPON    ;;CALL=TYPON    TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
2868
2869 024276 023300          $GTSWR    ;;CALL=GTSWR    TRAP+5(104405) GET SOFT-SWR SETTING
2870
2871 024300 023210          $CKSWR    ;;CALL=CKSWR    TRAP+6(104406) TEST FOR CHANGE IN SOFT-SWR
2872 024302 023512          $RDCHR    ;;CALL=RDCHR    TRAP+7(104407) TTY TYPEIN CHARACTER ROUTINE
2873 024304 023602          $RDLIN    ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE
2874 024306 024126          $RDOCT    ;;CALL=RDOCT    TRAP+11(104411) READ AN OCTAL NUMBER FROM TTY
2875
2876          .SBTTL POWER DOWN AND UP ROUTINES
2877
2878          ;*****
2879          ;POWER DOWN ROUTINE
2880 024310 012737 024450 000024  $PWRDN: MOV        #SILLUP,@#PWRVEC ;;SET FOR FAST UP
2881 024316 012737 000340 000026  MOV        #340,@#PWRVEC+2 ;;PRIO:7
2882 024324 010046          MOV        R0,-(SP)    ;;PUSH R0 ON STACK
2883 024326 010146          MOV        R1,-(SP)    ;;PUSH R1 ON STACK
2884 024330 010246          MOV        R2,-(SP)    ;;PUSH R2 ON STACK
2885 024332 010346          MOV        R3,-(SP)    ;;PUSH R3 ON STACK
2886 024334 010446          MOV        R4,-(SP)    ;;PUSH R4 ON STACK
2887 024336 010546          MOV        R5,-(SP)    ;;PUSH R5 ON STACK
2888 024340 017746 154574  MOV        @SWR,-(SP)    ;;PUSH @SWR ON STACK
2889 024344 010637 024454  MOV        SP,$SAVR6    ;;SAVE SP
2890 024350 012737 024362 000024  MOV        #PWRUP,@#PWRVEC ;;SET UP VECTOR
    
```



```

2890 024356 000000          HALT
2891 024360 000776          BR      .-2          ;;HANG UP
2892
2893          ;;*****
2894          ;POWER UP ROUTINE
2895 024362 012737 024450 000024 $PWRUP: MOV  #SILLUP,@#PWRVEC ;;SET FOR FAST DOWN
2896 024370 013706 024454          MOV  $SAVR6,SP      ;;GET SP
2897 024374 005037 024454          CLR  $SAVR6        ;;WAIT LOOP FOR THE TTY
2898 024400 005237 024454          1$: INC $SAVR6     ;;WAIT FOR THE INC
2899 024404 001375          BNE  1$           ;;OF WORD
2900 024406 012677 154526          MOV  (SP)+,@SWR    ;;POP STACK INTO @SWR
2901 024412 012605          MOV  (SP)+,R5     ;;POP STACK INTO R5
2902 024414 012604          MOV  (SP)+,R4     ;;POP STACK INTO R4
2903 024416 012603          MOV  (SP)+,R3     ;;POP STACK INTO R3
2904 024420 012602          MOV  (SP)+,R2     ;;POP STACK INTO R2
2905 024422 012601          MOV  (SP)+,R1     ;;POP STACK INTO R1
2906 024424 012600          MOV  (SP)+,R0     ;;POP STACK INTO R0
2907 024426 012737 024310 000024 MOV  #SPWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
2908 024434 012737 000340 000026 MOV  #340,@#PWRVEC+2 ;;PRIO:7
2909 024442 104401          TYPE          ;;REPORT THE POWER FAILURE
2910 024444 024456          $PWRMG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
2911 024446 000002          RTI
2912 024450 000000          SILLUP: HALT
2913 024452 000776          BR      .-2          ;;THE POWER UP SEQUENCE WAS STARTED
2914 024454 000000          $SAVR6: 0        ;; BEFORE THE POWER DOWN WAS COMPLETE
2915 024456 005015 047520 042527 $POWER: .ASCIZ <15><12>"POWER" ;;PUT THE SP HERE
2916 024464 000122
2917          .EVEN
2918          .END
    
```

```

AUTSL2 002644          443# 660
BA      001406          275# 1366* 1370* 1372* 1393 1427
BADINP 003274          504 506 513#
BADONE 010360          1145#
BAD.IN  011772          1306#
BAD.ON  013762          1570# 1586 1588
BASE    001266          219# 835
BASINC  005360          840 852#
BEGCT   017422          1881# 2106
BEGCT1  017430          1884# 2105
BEGIN   002622          421 427#
BIT0 = 000001          114# 1078 1398 1415 1432 1971 1978 1982 1989
BIT00 = 000001          104# 114
BIT01 = 000002          103# 113
BIT02 = 000004          102# 112
BIT03 = 000010          101# 111
BIT04 = 000020          100# 110
BIT05 = 000040          99# 109
BIT06 = 000100          98# 108
BIT07 = 000200          97# 107
BIT08 = 000400          96# 106
BIT09 = 001000          95# 105
BIT1 = 000002          113# 1396 1431 2031 2033 2042 2044
BIT10 = 002000          94# 1395 1412
BIT11 = 004000          93# 1395 1429
BIT12 = 010000          92# 289
BIT13 = 020000          91# 1774 1784 2215 2228
BIT14 = 040000          90# 508 541 639 673 1418 1547
BIT15 = 100000          89# 609 1000 1002 1111 1885
BIT2 = 000004          112# 1413 1431 1928 1934 1938 1946
BIT3 = 000010          111#
BIT4 = 000020          110# 800 802 805 937 939 942
BIT5 = 000040          109# 291 754 1760
BIT6 = 000100          108# 290 1356 1361 1471 1881 1955
BIT7 = 000200          107# 590 594 1113 1881 1891 1913
BIT8 = 000400          106# 1429
BIT9 = 001000          105# 292
BPTVEC= 000014          121#
BUFF    013700          275 1561#
BUFR    022232          2331* 2332 2344* 2347 2359 2379* 2380 2400#
BUSADR  001336          253# 796* 867 927*
CHECK1  005450          871# 881 885
CHKCNT  001350          258# 1014* 1055*
CKSWR = 104406          2871#
CLRDPPL 020336          1972 1980 2074#
CNTSIN  001322          242# 705* 976*
COM      012320          1351# 1364
COMMON  012314          1350#
COMND   001316          238# 788* 904* 996
CONTIN  011404          1230 1232# 1298
CONTRL  001332          251# 796* 869 930*
CR = 000015          29# 1361 2469 2479
CRLF = 000200          50# 366 2440 2479
CYCLE   004174          534 672#
CYCLE2  004220          678#
CYCL2   004210          674 376#
    
```









.SDIV	1#	
.SEOP	1#	
.SERR0	1#	
.SERRT	1#	
.SMULT	1#	
.SPOWE	1#	2875
.SRAND	1#	
.SRDDE	1#	
.SRDOC	1#	2795
.SREAD	1#	2556
.SR2AZ	1#	
.SSAVE	1#	
.SSB2D	1#	
.SSB2O	1#	
.SSCOP	1#	
.SSIZE	1#	
.SSUPR	1#	
.STRAP	1#	2833
.STYPB	1#	
.STYPD	1#	
.STYPE	1#	2409
.STYPO	1#	2479
.S4OCA	1#	
.1170	1#	

. ABS. 024466 000

ERRORS DETECTED: 0

DZRKIF.BIN,DZRKIF.LST/CRF/SOL=DZRKKE.SML,DZRKIF.P11  
RUN-TIME: 12 16 1 SECONDS  
RUN-TIME RATIO: 102/31=3.3  
CORE USED: 33K (65 PAGES)