

RK11/RK05

RK11 UTILITY PACKAGE
CZRKIFO

AH-9238F-MC

COPYRIGHT © 74-78

FICHE 1 OF 1

JUN 1978

digital

MADE IN USA

This microfiche card contains a grid of frames. The first column on the left contains frames with text, likely serving as a table of contents or index. The subsequent columns contain frames with data, including tables and lists. The data is organized into columns and rows, with some frames containing headers and footers. The text is small and difficult to read due to the high resolution of the microfiche.

100-100000-000000

PRODUCT CODE: AC-9236F-MC
PRODUCT NAME: CZRKIFD 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

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
9.6	RK05 CONTROL PANEL TEST # 2
9.7	HEAD ALIGNMENT ROUTINE
9.8	(DISK) POWER FAILURE TEST
9.9	SECTION SPECIAL
9.10	COMPATIBILITY ERROR RECOVERY

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 175 (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

VARIES 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) .EST	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

GO1

SEQ 0006

COMPATIBILITY OF UP TO EIGHT (8) DRIVES SIMPLY BY STATING THE DRIVE NUMBERS TO BE TESTED. THE TEST DOES THE REST, INSTRUCTING THE USER WHERE TO PLACE THE PACK. THE LIMITATIONS OF TESTING ARE IF THERE ARE (2) TWO PROCESSORS, FROM ONE (1) TO SEVEN (7) DRIVES MAY BE ON SYSTEM ONE, AND ONLY ONE (1) DRIVE (ANY DRIVE NUMBER) MAY BE ON SYSTEM TWO.

COMPATIBILITY-A DEFINITION. COMPATIBILITY INFERS MORE THAN THE FACT THAT INFORMATION WHICH WAS WRITTEN ON ONE DRIVE CAN BE READ ON ANOTHER.

FOR DRIVES TO BE CONSIDERED TRULY COMPATIBLE ANY DRIVE SHOULD BE ABLE TO READ WHAT WAS WRITTEN BY ANY OTHER DRIVE AND ALSO MUST BE ABLE TO OVERWRITE A PORTION OF INFORMATION WRITTEN BY ANOTHER DRIVE, WITH NEW INFORMATION, AND READ IT BACK. THIS IS A VERY BROAD DEFINITION BUT IS THE BASIC PREMISE OF TRUE COMPATIBILITY.

THE BELOW IS AN EXAMPLE OF ACTUAL OUTPUT. THE USER WANTS TO RUN SINGLE PROCESSOR MODE AND TEST COMPATIBILITY ON THREE (3) DRIVES WHOSE UNIT NUMBERS ARE 0,1,3.....

USE:

EXAMPLE 1

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 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

SEQ 0007

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	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 SYSTEM 1=1,0

IS THERE A SECOND SYSTEM?Y
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
	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
		7

TYPE=

THE USER NOW PRESSES 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 OSCILLISCOPE. IT SHOULD BE NOTED THAT THE OSCILLATING SEEKS BETWEEN THE SPECIFIED CYLINDERS ARE DONE ON ALL AVAILABLE DRIVES. THE ONLY WAY TO EXIT IS HALT!, 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 TYPEOUT
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. THUS 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
	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
		7

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

RK11 UTILITY PACKAGE

INDEX	NAME	TYPE
	COMPATIBILITY PACKAGE	0
		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
	COMPATABILITY 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
		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

NAME	TYPE
INDEX	0
COMPATABILITY 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=

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.
 SW2-15=0
 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 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
RKDS CONTROL PANEL TEST        4

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

IS THERE A SECOND SYSTEM?N      :NO SECOND SYSTEM
MOUNT PACK ON DRIVE #0         :CONTINUE PRESSED BUT
MAKE PACK WRITE ENABLE         :WRITE PROTECT ON
PRESS CONTINUE WHEN DRIVE RDY  :CLEAR WRITE PROTECT SWITCH
DRIVE WRITE PROTECTED.         :NOW RUNS TO FINISH
DRIVE WRITE PROTECTED         :THIS DOES NOT EFFECT
DONE!                          :OUTCOME OF TEST

      RK11 UTILITY PACKAGE

      NAME          TYPE
INDEX          0
COMPATIBILITY PACKAGE 1
OSCILLATING SEEK PACKAGE 2
    
```

ERROR EXAMPLE 2

```

      RK11 UTILITY PACKAGE

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

RK11 UTILITY PACKAGE

INDEX	NAME	TYPE
	COMPATIBILITY PACKAGE	0
		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

RK11 UTILITY PACKAGE

INDEX	NAME	TYPE
	COMPATABILITY PACKAGE	0
	OSCILLATING SEEK PACKAGE	1
	FORMATTER-SURFACE VERIFIER	2
	RK05 CONTROL PANEL TEST	3
	RK05 CONTROL PANEL TEST ROUTINE	4
	HEAD ALIGNMENT ROUTINE	5
	POWER FAILURE (WRITE) TEST	6
		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 RECD=177000
ADDR=002764 EXPCTD=077400 RECD=077600
ADDR=002764 EXPCTD=077400 RECD=037600
ADDR=002764 EXPCTD=077400 RECD=037600
ADDR=002764 EXPCTD=077400 RECD=037600
ERROR! DATA WRITTEN BY DRIVE 1 CANNOT BE READ.
ADDR=007624 EXPCTD=077400 RECD=177000
ERROR! DATA WRITTEN BY DRIVE 1 CANNOT BE READ.
ADDR=007633 EXPCTD=077400 RECD=177000
ADDR=007633 EXPCTD=077400 RECD=177000
DONE!

RK11 UTILITY PACKAGE

INDEX	NAME	TYPE
	COMPATABILITY 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
		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
MAKE PACK WRITE ENABLE
PRESS CONTINUE WHEN DRIVE RDY
ERROR! DATA WRITTEN BY DRIVE 1 CANNOT BE READ.
  ADDR=000367 EXPCTD=077400 RECVD=077600
  ADDR=000367 EXPCTD=077400 RECVD=037600
  ADDR=000367 EXPCTD=077400 RECVD=037600
  ADDR=000367 EXPCTD=077400 RECVD=037600
  ADDR=000367 EXPCTD=077400 RECVD=037600
ERROR! DATA WRITTEN BY DRIVE 1 CANNOT BE READ.
  ADDR=002564 EXPCTD=077400 RECVD=077600
  ADDR=002564 EXPCTD=077400 RECVD=037600
  ADDR=002564 EXPCTD=077400 RECVD=037600
  ADDR=002564 EXPCTD=077400 RECVD=037600
  ADDR=002564 EXPCTD=077400 RECVD=037600
ERROR! DATA WRITTEN BY DRIVE 1 CANNOT BE READ.
  ADDR=002764 EXPCTD=077400 RECVD=077600
  ADDR=002764 EXPCTD=077400 RECVD=037600
  ADDR=002764 EXPCTD=077400 RECVD=037600
  ADDR=002764 EXPCTD=077400 RECVD=037600
  ADDR=002764 EXPCTD=077400 RECVD=037600
ERROR! DATA WRITTEN BY DRIVE 1 CANNOT BE READ.
  ADDR=002767 EXPCTD=077400 RECVD=177000
5 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
	COMPATABILITY 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
		7

TYPE=0

RK11 UTILITY PACKAGE

INDEX	NAME	TYPE
	COMPATABILITY 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
		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
	COMPATABILITY 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
		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
	COMPATABILITY 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
		7

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

RK11 UTILITY PACKAGE

INDEX	NAME	TYPE
	COMPATABILITY 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
		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!

MAINDEC-11-DZRKI-E RK11 UTILITY PACKAGE
DZRKIF.P11 24-MAR-78 09:20 MACY11 30A(1052) 24-MAR-78 09:23
 TABLE OF CONTENTS

22	BASIC DEFINITIONS
132	TRAP CATCHER
141	STARTING ADDRESS(ES)
146	ACT11 HOOKS
156	COMMON TAGS
296	ERROR POINTER TABLE
324	INITIALIZE THE COMMON TAGS
353	TYPE PROGRAM NAME
358	GET VALUE FOR SOFTWARE SWITCH REGISTER
440	COMPATIBILITY TEST
1155	OSCILLATING SEEK ROUTINE
1307	FORMATTER-SURFACE VERIFIER
1571	RK05 CONTROL PANEL TEST
1841	CONTROL PANEL TEST # 2
2165	HEAD ALIGNMENT ROUTINE
2288	DISK POWER FAILURE TEST
2412	TYPE ROUTINE
2482	BINARY TO OCTAL (ASCII) AND TYPE
2559	TTY INPUT ROUTINE
2799	READ AN OCTAL NUMBER FROM THE TTY
2837	TRAP DECODER
2860	TRAP TABLE

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55

```

.TITLE MAINDEC-11-DZRKI-E
.*COPYRIGHT (C) 1974,1977
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754
.*
.*PROGRAM BY BOB COLLINS
.*
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
.*
$TN=1
$$SWR=160000      ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT

.*REVISD BY JIM KAPADIA
.*REVISD BY TOM SAWYER FEB 27, 1976
.*REVISD BY CHUCK HESS AUGUST, 1976

.SBTTL BASIC DEFINITIONS

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

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

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

.*PRIORITY LEVEL DEFINITIONS
000000  PR0= 0                ;;PRIORITY LEVEL 0
000040  PR1= 40               ;;PRIORITY LEVEL 1
000100  PR2= 100              ;;PRIORITY LEVEL 2
000140  PR3= 140              ;;PRIORITY LEVEL 3
000200  PR4= 200              ;;PRIORITY LEVEL 4

```

```

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

```



```

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
130      .SBTTL TRAP CATCHER
131
132      .=0
133      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
134      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
135      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
136      DISPREG: .WORD 0          ;; SOFTWARE DISPLAY REGISTER
137      SWREG:   .WORD 0          ;; SOFTWARE SWITCH REGISTER
138
139      .SBTTL STARTING ADDRESS(ES)
140      JMP      @#STARTR          ;; JUMP TO STARTING ADDRESS OF PROGRAM
141      .=210
142      MOV8    #377,@#MODE
143      JMP      @#START
144      .SBTTL ACT11 HOOKS
145
146      ;*****
147      ;HOOKS REQUIRED BY ACT11
148      $SVPC=.          ;SAVE PC
149      $ENDAD          ;; 1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
150      .=52
151      .WORD 0          ;; 2)SET LOC.52 TO ZERO

```

153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207

001100
001100 000000
001102 000
001103 000
001104 000000
001106 000000
001110 000000
001112 000000
001114 000
001115 001
001116 000000
001120 000000
001122 000000
001124 000000
001126 000000
001130 000000
001132 000000
001134 000
001135 000
001136 000000
001140 177570
001142 177570
001144 177560
001146 177562
001150 177564
001152 177566
001154 000
001155 002
001156 012
001157 000
001160 077
001161 015
001162 000012
001164 000000
001166 000010
001206 152525
001210 077777
001212 000000
001214 012345
001216 125252
001220 000001
001222 177777
001224 154320
001226 000010

.SBTTL COMMON TAGS

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

.=1100

SCMTAG: .WORD 0
\$PASS: .WORD 0
\$STNM: .BYTE 0
\$ERFLG: .BYTE 0
\$ICNT: .WORD 0
\$LPADR: .WORD 0
\$LPERR: .WORD 0
\$ERTTL: .WORD 0
\$ITEMB: .BYTE 0
\$ERMAX: .BYTE 1
\$ERRPC: .WORD 0
\$GDADR: .WORD 0
\$BDADR: .WORD 0
\$GDDAT: .WORD 0
\$BDDAT: .WORD 0
\$AUTOB: .BYTE 0
\$INTAG: .BYTE 0
\$SWR: .WORD DSWR
\$DISPLAY: .WORD DDISP
\$TKS: 177560
\$TKB: 177562
\$TPS: 177564
\$TPB: 177566
\$NULL: .BYTE 0
\$FILLS: .BYTE 2
\$FILLC: .BYTE 12
\$TPFLG: .BYTE 0
\$QUES: .ASCII /?/
\$CRLF: .ASCII <15>
\$LF: .ASCIZ <12>
\$DRACTV: .WORD 0
LOGA: .BLKW 10
DRVO: .WORD 152525
.WORD 077777
.WORD 000000
.WORD 012345
.WORD 125252
.WORD 000001
.WORD 177777
.WORD 154320
RDTBL: .BLKW 10

START OF COMMON TAGS
CONTAINS PASS COUNT
CONTAINS THE TEST NUMBER
CONTAINS ERROR FLAG
CONTAINS SUBTEST ITERATION COUNT
CONTAINS SCOPE LOOP ADDRESS
CONTAINS SCOPE RETURN FOR ERRORS
CONTAINS TOTAL ERRORS DETECTED
CONTAINS ITEM CONTROL BYTE
CONTAINS MAX. ERRORS PER TEST
CONTAINS PC OF LAST ERROR INSTRUCTION
CONTAINS ADDRESS OF 'GOOD' DATA
CONTAINS ADDRESS OF 'BAD' DATA
CONTAINS 'GOOD' DATA
CONTAINS 'BAD' DATA
RESERVED--NOT TO BE USED
AUTOMATIC MODE INDICATOR
INTERRUPT MODE INDICATOR
ADDRESS OF SWITCH REGISTER
ADDRESS OF DISPLAY REGISTER
TTY KBD STATUS
TTY KBD BUFFER
TTY PRINTER STATUS REG. ADDRESS
TTY PRINTER BUFFER REG. ADDRESS
CONTAINS NULL CHARACTER FOR FILLS
CONTAINS # OF FILLER CHARACTERS REQUIRED
INSERT FILL CHARS. AFTER A "LINE FEED"
"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
QUESTION MARK
CARRIAGE RETURN
LINE FEED

ACTIVE DRIVE WORD
TABLE OF ACTIVE DRIVE WORDS
TABLE OF PATTERN = TO DRIVE #'S
TABLE OF READ ADDRESS

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	001326	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

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	SEEK0:	.WORD	0
273					
274		105212	LFLF=	105212	
275	001406	013700	BA:	BUFF	
276	001410	000000	DA:	.WORD	0
277	001412	000000	WC:	.WORD	0
278	001414	013702	RBA:	RBUFF	
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=	BIT6	
291		000040	WPS=	BITS	

293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312

001434

.SBTTL ERROR POINTER TABLE

;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
;*LOCATION SITE#B. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
;*NOTE1: IF SITE#B IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).
;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

;* EM ::POINTS TO THE ERROR MESSAGE
;* DH ::POINTS TO THE DATA HEADER
;* DT ::POINTS TO THE DATA
;* DF ::POINTS TO THE DATA FORMAT

\$ERRTB:
;*****
;THE ERROR TABLE IS UNUSED IN THIS PROGRAM
;*****

```

314 001434 105037 001312 START: CLRB 2#MODE ;CLEAR THE BUS
315 001440 000005 START: RESET
316 001442 012706 001100 MOV #STACK,SP ;SET UP STACK FOR PSW=0
317 001446 012746 000000 MOV #0,-(SP) ;RETURN FOR RTI
318 001452 012746 001460 MOV #2$,-(SP)
319 001456 000002 RTI
320 001460 000240 2$: NOP
321 .SBTTL INITIALIZE THE COMMON TAGS
322 ;;CLEAR THE COMMON TAGS (SCMTAG) AREA
323 001462 012706 001100 MOV #SCMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
324 001466 005026 001100 CLR (R6)+ ;;CLEAR MEMORY LOCATION
325 001470 022706 001140 CMP #SWR,R6 ;;DONE?
326 001474 001374 001140 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,2#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
330 001510 012737 000340 000036 MOV #340,2#TRAPVEC+2;LEVEL 7
331 001516 012737 024310 000024 MOV #SPHADN,2#PWAVEC ;;POWER FAILURE VECTOR
332 001524 012737 000340 000026 MOV #340,2#PWAVEC+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 2#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
336 001536 012737 001572 000004 MOV #64$ ,2#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 #-1,2#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$: MOV #65$,(SP) ;;SET UP FOR TRAP RETURN
344 001576 000002 RTI
345 001600 012737 000176 001140 65$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
346 001606 012737 000174 001142 MOV #DISPREG,DISPLAY
347 001614 012637 000004 66$: MOV (SP)+,2#ERRVEC ;;RESTORE ERROR VECTOR
348
349 001620 004737 022746 JSR PC,STKINT ;;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 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
356 001636 005737 000042 TST 2#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
357 001642 001006 BNE 69$ ;;BRANCH IF YES
358 001644 023727 001140 000176 CMP SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
359 001652 001005 BNE 70$ ;;BRANCH IF NO
360 001654 104405 GTSWR ;;GET SOFT-SWR SETTINGS
361 001656 000403 BR 70$
362 001660 112737 000001 001134 69$: MOVB #1,SAUTOB ;;SET AUTO-MODE INDICATOR
363 001666 70$:
364 001666 000426 BR 67$ ;;GET OVER THE ASCIZ
365 ;;68$: .ASCIZ <CRLF>/RK11 UTILITY PACKAGE/<15><12>/MAINDEC-11-DZRKI-E/<CRLF>
366 001744
367 001744 105737 001312 67$: TSTR 2#MODE
368 001750 100002 BPL 1$

```

```

370 001756 105737 001325 1S: TSTB STFLG ;PRINT ONLY THE FIRST TIME
371 001762 001402 BEQ 10S
372 001764 000137 002554 JMP TABLTY
373 001770 105237 001325 10S: INCB STFLG
374 001774 105037 001326 STRT1: CLRB OSPFLG
375 002000 104401 002006 TYPE 65S ;:TYPE ASCIZ STRING
376 002004 000423 BR 64S ;:GET OVER THE ASCIZ
;:65S: .ASCIZ <15><12><15><12>/ NAME TYPE/
377
378 002054 64S: TYPE 67S ;:TYPE ASCIZ STRING
379 002054 104401 002062 BR 66S ;:GET OVER THE ASCIZ
380 002060 000421 ;:67S: .ASCIZ <15><12>/INDEX 0/
381
382 002124 66S: TYPE 69S ;:TYPE ASCIZ STRING
383 002124 104401 002132 BR 68S ;:GET OVER THE ASCIZ
384 002130 000421 ;:69S: .ASCIZ <15><12>/COMPATIBILITY PACKAGE 1/
385
386 002174 68S: TYPE 71S ;:TYPE ASCIZ STRING
387 002174 104401 002202 BR 70S ;:GET OVER THE ASCIZ
388 002200 000421 ;:71S: .ASCIZ <15><12>/OSCILLATING SEEK PACKAGE 2/
389
390 002244 70S: TYPE 73S ;:TYPE ASCIZ STRING
391 002244 104401 002252 BR 72S ;:GET OVER THE ASCIZ
392 002250 000421 ;:73S: .ASCIZ <15><12>/FORMATTER-SURFACE VERIFIER 3/
393
394 002314 72S: TYPE 75S ;:TYPE ASCIZ STRING
395 002314 104401 002322 BR 74S ;:GET OVER THE ASCIZ
396 002320 000421 ;:75S: .ASCIZ <15><12>/RK05 CONTROL PANEL TEST 4/
397
398 002364 74S: TYPE 77S ;:TYPE ASCIZ STRING
399 002364 104401 002372 BR 76S ;:GET OVER THE ASCIZ
400 002370 000421 ;:77S: .ASCIZ <15><12>/RK05 CONTROL PANEL TEST #2 5/
401
402 002434 76S: TYPE 79S ;:TYPE ASCIZ STRING
403 002434 104401 002442 BR 78S ;:GET OVER THE ASCIZ
404 002440 000421 ;:79S: .ASCIZ <15><12>/HEAD ALIGNMENT ROUTINE 6/
405
406 002504 78S: TYPE 81S ;:TYPE ASCIZ STRING
407 002504 104401 002512 BR 80S ;:GET OVER THE ASCIZ
408 002510 000421 ;:81S: .ASCIZ <15><12>/POWER FAILURE (WRITE) TEST 7/
409
410 002554 80S:
411 002554 TABLTY:
412 002554 104401 002562 TYPE 65S ;:TYPE ASCIZ STRING
413 002560 000405 BR 64S ;:GET OVER THE ASCIZ
;:65S: .ASCIZ <15><12><15><12>/TYPE=/
414
415 002574 64S:
416 002574 104411 RDOCT ;:GET THE TEST NUMBER FROM THE OPERATOR
417 002576 012600 MOV (SP)+,R0 ;:STORE IT IN R0
418 002600 022700 000010 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 $QUES ;:
423 002620 000755 BR TABLTY
424

```

```

426
427 002622 001774
428 002624 002642
429 002626 010370
430 002630 012002
431 002632 013772
432 002634 017300
433 002636 020634
434 002640 021544
435
436
437
438
439
440
441
442 002642 000240
443 002644
444 002644 104401 002652
445 002650 000415
446
447 002704
448 002704 104401 002712
449 002710 000417
450
451 002750
452 002750 104410
453 002752 012600
454 002754 012701 001166
455 002760 105037 001311
456 002764 005037 001330
457 002770 112037 001330
458 002774 122737 000054 001330
459 003002 001770
460 003004 162737 000060 001330
461 003012 100403
462 003014 004737 004072
463 003020 000761
464 003022 122740 000056
465 003026 001402
466 003030 004737 004142
467 003034 022701 001206
468 003040 001403
469 003042 012721 100000
470 003046 000772
471
472
473
474
475
476 003050
477 003050 104401 003056
478 003054 000416
479
480 003112
    
```

```

BEGIN:  STRT1
        SECT.3
        SECT.2
        SECT.1
        SECT.0
        SECT.4
        SECT.5
        SECT.6

        .SBTTL  COMPATIBILITY TEST

        ;ROUTINE TO PICK UP THE DRIVE NUMBER TO BE TESTED
        ;ON SYSTEM 1.

        SECT.3: NOP ;NO-OP
        AUTSL2:
        TYPE 65$ ;:TYPE ASCIZ STRING
        BR 64$ ;:GET OVER THE ASCIZ
        ;:65$: .ASCIZ <15><12>/TERMINATE WITH '<CR>'/
        64$:
        TYPE 67$ ;:TYPE ASCIZ STRING
        BR 66$ ;:GET OVER THE ASCIZ
        ;:67$: .ASCIZ <15><12>/DRIVE NUMBERS ON SYSTEM 1=/
        66$:
        RDLIN
        MOV (SP)+,R0 ;:PICK UP THE ADDRESS OF THE INPUT BUFFER
        MOV #LOGA,R1 ;:GET THE ADDRESS OF THE LOGICAL UNIT TBL.
        CLRB #DRCNT1 ;:CLEAR THE DRIVE COUNTER
        CLR #KYTEMP ;:CLEAR TEMP
        1$: MOVB (R0)+,#KYTEMP ;:GET THE FIRST DRIVE #
        CMPB #54,#KYTEMP ;:IS IT A COMMA THAT WAS TYPED?
        BEQ 1$ ;:IF YES GO BACK
        SUB #60,#KYTEMP ;:MAKE ASCII A DRIVE #
        BMI 2$ ;:IF RESULT NEGATIVE BRANCH
        JSR PC,STORE ;:IF RESULT POSITIVE JUMP
        BR 1$ ;:AFTER STORING GET NEXT #
        2$: CMPB #56,-(R0) ;:WAS NEGATIVE RESULT A TERMINATOR?
        BEQ 3$ ;:IF YES BRANCH
        JSR PC,ILEGAL ;:IF NO BAD CHARACTER JUMP
        3$: CMP #DAVD,R1 ;:IS THE TABLE FULL
        BEQ SECSYS ;:IF YES BRANCH
        MOV #100000,(R1)+ ;:IF NO FILL TABLE WITH DOWN INDICATOR.
        BR 3$ ;:GO BACK AND CHECK

        ;ROUTINE TO DETERMINE IF THERE IS A SECOND
        ;SYSTEM AND IF SO TO GET THE NUMBER OF THE
        ;DRIVE ON THIS SYSTEM

        SECSYS:
        TYPE 65$ ;:TYPE ASCIZ STRING
        BR 64$ ;:GET OVER THE ASCIZ
        ;:65$: .ASCIZ <15><12>/IS THERE A SECOND SYSTEM?/
        64$:
    
```



```

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 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          MOVB    @#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 003274          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      @DRVD, R0        :DONE?
529 003334 001414          BEQ     EXIT            :IF YES GET OUT
530 003336 005710          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

```

```

538 003366 012700 001166 EXIT: MOV #LOGA,R0
539 003372 022700 001206 EXTR2: CMP #DRVD,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 EXTR2
545 003412 014001 MOV -(R0),R1
546 003414 004737 004250 JSR PC,D02
547 003420 004737 005632 JSR PC,RDLINK
548 003424 005720 TST (R0)+
549 003426 000761 BR EXTR2
550 003430
551 003430 104401 003436 EXITX: TYPE 65$ ;;TYPE ASCIZ STRING
552 003434 000404 BR 64$ ;;GET OVER THE ASCIZ
553
554 003446 65$: .ASCIZ <15><12>/DONE!/
555 003446 000137 001440 64$: 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 003452 SECOND:
561 003452 104401 003460 TYPE 65$ ;;TYPE ASCIZ STRING
562 003456 000415 BR 64$ ;;GET OVER THE ASCIZ
563
564 003512 65$: .ASCIZ <15><12>/COMPATIBILITY-SYSTEM#2/
565 003512 012704 000200 64$: 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
569 003526 104401 003534 INSYS2: TYPE 65$ ;;TYPE ASCIZ STRING
570 003532 000405 BR 64$ ;;GET OVER THE ASCIZ
571
572 003546 65$: .ASCIZ <15><12>/WORD /
573 003546 000240 64$: NOP
574 003550 010346 MOV R3,-(SP) ;***
575 003552 104403 TYPOS ;GET READY TO TYPE WORD COUNTER
576 003554 006 .BYTE 6
577 003555 000 .BYTE 0
578 003556 104401 003564 TYPE 67$ ;;TYPE ASCIZ STRING
579 003562 000401 BR 66$ ;;GET OVER THE ASCIZ
580
581 003566 67$: .ASCIZ /=/
582 003566 104411 66$: RDOCT
583 003570 012600 MOV (SP)+,R0 ;PICK UP THE OCTAL WORD
584 003572 110001 MOV# R0,R1 ;GET THE FIRST DRIVE TO R1
585 003574 000300 SWAB R0 ;GET THE SECOND DRIVE TO R0 LOW BYTE
586 003576 042700 177400 BIC #177400,R0 ;CLEAR THE UNUSED BITS
587 003602 042701 177400 BIC #177400,R1 ;CLEAR THE UNUSED BITS
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 ;AND CLEAR THE CARRY BIT
592 003620 006001 2$: ROR R1 ;NOW DO THE SAME FOR R1

```

```

594 003624 052701 000200      BIS      #BIT7,R1      ;IF ERROR SET THE BIT
595 003630 110162 000001      3$:     MOVB     R1,1(R2) ;SET DRIVE FIRST IN TABLE
596 003634 004737 004030      JSR     PC,MASKER   ;CALL THE MASK CONTROL SUBROUTINE
597 003640 110062 000001      MOVB     R0,1(R2)   ;SET DRIVE SECOND IN TABLE (NEXT WORD)
598 003644 004737 004030      JSR     PC,MASKER   ;CALL THE MASK CONTROL SUBROUTINE
599 003650 005203          INC      R3          ;INCREMENT THE WORD COUNTER
600 003652 000725          BR      INSYS2      ;GO BACK AND GET NEXT WORD
601 003654 050412          EXITA:  BIS      R4,(R2) ;FILL THE TABLE (LAST WORD)
602 003656 006004          ROR     R4          ;WITH ALL BITS SET
603 003660 103375          BCC     EXITA       ;GO BACK IF NOT DONE
604 003662 042712 174000      EXITB:  BIC      #174000,(R2) ;CLEAR DOWN AND SECOND SYSTEM BITS AT TABLE
605 003666 010200          MOV     R2,R0       ;GET ADDRESS TO R0 (CURRENT TABLE)
606 003670 005722          TST     (R2)+       ;ADD 2 TO THE POINTER
607 003672 022702 001206      FIL.DN: CMP     #DRVO,R2   ;TABLE FULL?
608 003676 001403          BEQ     2$         ;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$:     MOV     (R0),R1 ;GET THE WORD TO R1
612 003710 110102          MOVB     R1,R2
613 003712 004737 005234      JSR     PC,MASK     ;FORM DRIVE # FOR MOUNT
614 003716 004737 004250      JSR     PC,DO2      ;WRITE NEW INFO
615 003722 004737 005104      JSR     PC,WRLINK   ;READ SAMPLE (ALL DRIVES)
616 003726 004737 005632      JSR     PC,RDLINK   ;TYPE ASCIZ STRING
617 003732 104401 003740      TYPE    65$        ;GET OVER THE ASCIZ
618 003736 000427          BR      64$
619          ;:65$: .ASCIZ <15><12>/DONE SYSTEM 2 RESTART SYSTEM 1, TYPE WORD /
620 004016          64$:     MOV     (R0),-(SP) ;GET WORD FOR SYSTEM 1 AND TYPE
621 004016 011046          TYPOS
622 004020 104403          .BYTE 6
623 004022 006          .BYTE 1
624 004023 001          1$:     HALT    ;HALT (FINISHED SYSTEM #2)
625 004024 000000          BR      1$        ;GO BACK TO HALT
626 004026 000776
627
628          ;THIS ROUTINE SETS UP THE MASK BITS IN THE LOG TABLE FOR
629          ;SYSTEM #2. IF A DRIVE IS DOWN NO BIT IS SET BUT THE MASK
630          ;IS SHIFTED.
631
632 004030 005712      MASKER: TST     (R2)      ;IS THE DRIVE UP
633 004032 100401      BMI     RETRN4      ;IF NO, BRANCH
634 004034 110412      MOVB     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?
    
```

```

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      @DRVD,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, (RD)=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    (RD),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      @#SECONE       ; GO TO SYSTEM #2
676 004210 113703 001324 CYCL2: MOV    @#INDEX,R3 ; GET THE INDEX VALUE
677 004214 116302 001256 MOV    @MSKTBL(R3),R2 ; GET THE MASK TO R2
678 004220 004737 005234 CYCLE2: JSR    PC,MASK   ; CALL THE MASK SUBROUTINE
679 004224 012703 001166 LDFLG: MOV    @LOGA,R3  ; GET TABLE ADDRESS TO R3
680 004230 020003 2$:      CMP      RD,R3        ; IS ACTIVE ADDRESS=TO FIRST ADDRESS
681 004232 001002          BNE     3$            ; 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      @DRVD,R3    ; DONE
686 004246 001370          BNE     2$            ; IF NO GO BACK
687 004250 000301          D02: SWAB    R1        ; GET DRIVE # TO LOW BYTE
688 004252 000240          NOP
689 004254 110102          MOV    R1,R2          ; GET IT TO R2
690 004256 042702 000370 BIC     @370,R2        ; CLEAR THE UNUSED BITS
691 004262 110237 001314 MOV    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 001353 MOV    R2,@#DSKTMP+1  ; GET IT TO DISK ADDR. TEMP.
698 004304 110237 001335 MOV    R2,@#DSKADR+1 ; CALL MOUNT
699 004310 004737 004316 JSR    PC,MOUNT
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 MOV    #5,@#ECNT     ; SET ERROR CNTR TO 5
    
```

```

706 004340 104401 004346          TYPE      65$          ;;TYPE ASCIZ STRING
707 004344 000414          BR        64$          ;;GET OVER THE ASCIZ
708          ;;65$: .ASCIZ <15><12>/MOUNT PACK ON DRIVE #/
709 004376          64$:          MOV        DRIVE,-(SP)      ;;SAVE DRIVE FOR TYPEOUT
710 004376 013746 001314          TYPOS          ;;GO TYPE--OCTAL ASCII
711 004402 104403          .BYTE      1          ;;TYPE 1 DIGIT(S)
712 004404 001          .BYTE      0          ;;SUPPRESS LEADING ZEROS
713 004405 000          TYPE      67$          ;;TYPE ASCIZ STRING
714 004406 104401 004414          BR        66$          ;;GET OVER THE ASCIZ
715 004412 000415          ;;67$: .ASCIZ <15><12>/MAKE PACK WRITE ENABLE/
716          66$:          TYPE      69$          ;;TYPE ASCIZ STRING
717 004446          BR        68$          ;;GET OVER THE ASCIZ
718 004446 104401 004454          ;;69$: .ASCIZ <15><12>/PRESS CONTINUE WHEN DRIVE RDY/
719 004452 000420          68$:          HALT
720          JSR      PC,INITIL      ;;CALL INITIALIZER
721 004514          RTS      PC          ;;RETURN
722 004514 000000          ;THIS ROUTINE INITIALIZES A DRIVE AND INSURES THAT IT IS READY AND
723 004516 004737 004524          ;WRITE ENABLED, IT IS ENTERED FROM MOUNT OR FROM EXECUTE IF OPERATION FAILS
724 004522 000207
725
726
727
728
729 004524 010046          INITIL: MOV      RD,-(SP)      ;;SAVE RD
730 004526 013700 001334          MOV      2#DSKADR,RD      ;;GET THE DRIVE # TO RD
731 004532 042700 001777          BIC      81777,RD          ;;CLEAR THE UNUSED BITS
732 004536 005037 001346          INITI2: CLR      2#TIMR      ;;CLEAR THE TIMER
733 004542 105037 001323          CLR      2#TIMR2
734 004546 000240          NOP
735 004550 010077 174622          MOV      RD,2#RKDA          ;;GET DRIVE # TO 'DA' REGISTER
736 004554 012777 000001 174606          MOV      81,2#RKCS          ;;ISSUE CONTROL RESET + GO
737 004562 004737 005614          JSR      PC,SMTME
738 004566 105777 174576          1$:      TSTB     2#RKCS          ;;DID CONTROL READY SET
739 004572 100423          BMI      2$              ;;IF YES BRANCH
740 004574 005237 001346          INC      2#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$: .ASCIZ <15><12>/CONTROL RESET TIME OUT/
745          64$:          MOV      RD,2#RKDA          ;;DRIVE NUMBER TO 'DA' REG.
746 004642 010077 174530          2$:      TSTB     2#RKDS          ;;IS DRIVE READY
747 004646 105777 174512          BMI      3$              ;;IF YES BRANCH
748 004652 100415          TYPE      67$          ;;TYPE ASCIZ STRING
749 004654 104401 004662          BR        66$          ;;GET OVER THE ASCIZ
750 004660 000411          ;;67$: .ASCIZ <15><12>/DRIVE NOT READY/
751          66$:          BR        INITI2          ;;GO BACK TRY AGAIN
752 004704          BIT      8BITS,2#RKDS      ;;IS DRIVE WRITE LOCKED?
753 004706 000714 000040 174450          BEQ      4$              ;;IF NO, BRANCH
754 004714 032777          TYPE      69$          ;;TYPE ASCIZ STRING
755 004716 001420 004724          BR        68$          ;;GET OVER THE ASCIZ
756 004716 104401          ;;69$: .ASCIZ <15><12>/DRIVE WRITE PROTECTED/
757 004722 000414          68$:          BR        INITI2          ;;YES, GO BACK TRY AGAIN
758
759 004754
760 004754 000670

```

```

762 004762 010077 174410      MOV      RD,DRKDA      ;GET THE DRIVE # TO 'DA' REGISTER
763 004766 012777 000015 174374  MOV      #15,DRKCS    ;ISSUE DRIVE RESET + GO
764 004774 004737 005614      JSR      PC,SMTME
765 005000 105777 174364      5$:     TSTB     DRKCS
766 005004 100375      BPL      5$
767 005006 032777 000100 174350  BIT      #100,DRKDS    ;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$: .ASCIZ <15><12>/DRIVE RESET TIMED OUT/
779      70$:
780 005076 000727      BR       4$           ;GO BACK, TRY AGAIN
781 005100 012600      6$:     MOV      (SP)+,RO    ;RESTORE RO
782 005102 000207      RTS      PC           ;RETURN TO CALLER
783
784      ;THIS ROUTINE TAKES CARE OF ALL LINKAGES FOR THE
785      ;EXECUTE ROUTINE. IT FORMS THE ADDRESS AND SETS UP ALL THE
786      ;REGISTERS FOR THE WRITE OPERATION
787
788 005104 105037 001316  WRLINK: CLAB     @#COMND    ;INDICATE WRITE OPERATION
789 005110 000240      NOP
790 005112 113701 001314      MOVB     @#DRIVE,R1    ;PICK UP THE DRIVE #
791 005116 006101      ROL      R1           ;MAKE IT A WORD INDEX
792 005120 016137 001206 001362  1$:     MOV      DRVD(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$:     MOV      @#PATRN,@#BUSADR ;GET THE ADDRESS OF THE OUTPUT
797 005144 013737 001342 001340      MOV      @#CYLCNT,@#WRDCNT ;GET THE WORD COUNT
798 005152 013737 001354 001332      MOV      @#WRITCS,@#CONTRL ;GET THE CONTROL + STATUS WORD
799 005160 004737 005402      JSR      PC,EXECUT    ;CALL EXECUTE
800 005164 032737 000020 001334      BIT      @#BIT4,@#DSKADR ;WAS THIS WRITE SURFACE "1"?
801 005172 001006      BNE      3$           ;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

```

```

818 005242 000240      NOP
819 005244 012703 000200  MOV    #200,R3      ;SET UP THE COMPARE MASK
820 005250 005004      CLR    R4           ;CLB THE INDEX COUNTER
821 005252 012705 001274  MOV    #CYLTBL,R5   ;GET THE CYLINDER TABLE ADDRESS
822 005256 030203      1S:  BIT    R2,R3    ;IS THE MASK BIT SELECTED IN BASE
823 005260 001401      BEQ    2S           ;IF NO BRANCH
824 005262 110425      MOV    R4,(R5)+    ;MOVE THE CYLINDER BASE TO THE TABLE
825 005264 105204      2S:  INCB   R4      ;INCREMENT THE BASE
826 005266 006003      ROR    R3          ;ROTATE THE COMPARE MASK
827 005270 103372      BCC    1S         ;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    #CYLTBL,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  CMP    #377,R4      ;IS IT THE TABLE TERMINATOR?
840 005322 001416      BASINC 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,D#DSKADR ;CLEAR ALL BUT DRIVE NUMBER
850 005352 050437 001334  BIS    R4,D#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    #CYLTBL,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: ADD    #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    D#TMR    ;CLEAR THE TIMER
864 005406 000240      NOP
865 005410 105037 001323  CLRB   D#TMR2      ;HALT HERE TO CHECK COMMAND ABOUT TO BE EXECUTED
866 005414 013777 001334 173754  MOV    D#DSKADR,D#RKDA ;CLEAR SECOND TIMER
867 005422 013777 001336 173744  MOV    D#BUSADR,D#RKBA ;LOAD THE DISK ADDRESS REGISTER
868 005430 013777 001340 173734  MOV    D#WRDCNT,D#RKWC ;LOAD THE BUS ADDRESS REGISTER
869 005436 013777 001332 173724  MOV    D#CONTRL,D#RKCS ;LOAD THE WORD COUNT REGISTER
870 005444 004737 005614      JSR    PC,SMTME    ;LOAD THE CONTROL REGISTER
871 005450 105777 173714  CHECK1: TSTB   D#KCS   ;KILL TIME FOR RK11-C
872 005454 100011      BPL    TIME        ;IS CONTROL READY SET
                        ;IF NO BRANCH

```

```

874 005462 005737 001360      TST      @ERRFLG      ;ERROR?
875 005466 001403      BEQ      IS          ;IF NO BRANCH
876 005470 005037 001360      CLR      @ERRFLG    ;CLEAR THE FLAG
877 005474 000742      BR       EXECUT     ;TRY AGAIN
878 005476 000207      IS:     RTS         PC
879 005500 005237 001346      TIME:   INC         @TIMR
880 005504 000240      NOP
881 005506 001360      BNE     CHECK1
882 005510 105737 001323      TSTB    @TIMR2      ;SECOND TIMEOUT?
883 005514 001003      BNE     IS          ;IF YES BRANCH
884 005516 105237 001323      INCB    @TIMR2      ;INDICATE SECOND TIMEOUT
885 005522 000752      BR      CHECK1     ;GO BACK
886 005524 004737 004524      IS:     JSR        PC,INITIL
887 005530 104401 005536      TYPE   655
888 005534 000426      BR      645        ;:TYPE ASCIZ STRING
889                                     ;:GET OVER THE ASCIZ
890 005612                                     ;:655: .ASCIZ <15><12>/TIMED OUT ON OPERATION RETRY IN PROGRESS/
891 005612 000673      645:   BR          EXECUT
892 005614 012737 000500 001346      SMTME: MOV         @500,@TIMR
893 005622 005337 001346      IS:     DEC         @TIMR
894 005626 001375      BNE     IS
895 005630 000207      RTS     PC

; THIS ROUTINE CHECKS TO SEE IF A DRIVE IS ACTIVE FROM A
; WRITE OPERATION. IT GETS THE READ MASK TO R3, AND THE
; EXPECTED DATA PATTERN TO "PATTERN", AND THEN CALLS ADDRESS
; CONTROL.

902 005632 010046      RDLINK: MOV        RD,-(SP)      ;SAVE RD
903 005634 000240      NOP
904 005636 152737 000377 001316      BISB    @377,@COMND ;***
905 005644 012701 001166      MOV     @LOGA,R1    ;INDICATE READ OPERATION
906 005650 012705 001305      MOV     @SECTBL,R5  ;GET THE TABLE ADDRESS TO R1
907 005654 000405      BR      RD2         ;GET THE SECTOR TABLE ADDRESS
908 005656 062701 000002      RD1:   ADD     @2,R1  ;SKIP OVER THE INDEX, FIRST PASS
909 005662 022701 001206      CMP     @DRVO,R1    ;ADD 2 TO THE ADDRESS
910 005666 001503      BEQ     EXIT2       ;ARE YOU THRU THE ENTIRE TABLE
911 005670 005711      RD2:   TST     (R1)   ;IF YES EXIT
912 005672 100001      BPL     RD3         ;IS THE DRIVE ACTIVE
913 005674 000770      BR      RD1         ;IF YES BRANCH
914 005676 011100      RD3:   MOV     (R1),RD ;IF NO GET NEXT WORD
915 005700 010002      MOV     RD,R2       ;GET THE ACTIVE WORD TO RD
916 005702 000300      SWAB   RD          ;COPY THE WORD
917 005704 042700 177770      BIC     @177770,RD  ;GET THE DRIVE # TO THE LOW BYTE
918 005710 110037 001317      MOVB   RD,@WRITBY  ;CLR ALL BUT THE DRIVE #
919 005714 006100      ROL    RD          ;SAVE THE DRIVE #
920 005716 016037 001206 001362      MOV     DRVO(RD),@PATTN ;MAKE IT AN INDEX
921 005724 004737 005234      JSR    PC,MASK     ;PICK UP THE DATA PATTERN
922 005730 004737 005302      JSR    PC,CYLADR   ;CALL THE MASK SUBROUTINE
923 005734 000401      BR      RD4         ;GO FORM A CYLINDER ADDRESS
924 005736 000747      BR      RD1         ;RETURN HERE IF NOT LAST BASE ADDR.
925 005740 053737 001352 001334      RD4:   BIS     @DSKTMP,@DSKADR ;IF LAST BASE ADDRESS, RETURN HERE
926 005746 152537 001334      RD5:   BISB   (R5)+,@DSKADR ;SET THE DRIVE # BITS IN DISK ADDR.
927 005752 012737 007356 001336      RD6:   MOV     @RDBUFF,@BUSADR ;SET THE SECTOR BITS IN DISK ADDR.
928 005760 000240      NOP                ;GET THE BUFFER ADDR.
;***

```



```

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,ROCHK         ;CHECK THE DATA
933 006006 042737 000017 001334      BIC      @17,@#DSKADR     ;CLR THE SECTOR BITS IN DISK ADDR.
934 006014 022705 001311      CMP      @#RCNT1,R5      ;WAS THIS THE LAST SECTOR?
935 006020 001353      BNE      RDS              ;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      RD7              ;IF YES, BRANCH
939 006036 052737 000020 001334      BIS      @BIT4,@#DSKADR  ;NO, SET SURFACE "1" BIT
940 006044 105137 001362      COMB    @#PATTRN        ;MAKE HEAD "1" PATTERN
941 006050 000736      BR       RDS              ;GO BACK AND EXECUTE
942 006052 042737 000020 001334      BIC      @BIT4,@#DSKADR  ;CLEAR THE SURFACE BIT
943 006060 105137 001362      COMB    @#PATTRN        ;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       RD4              ;IF HERE IT IS NOT THE LAST BASE ADDR
947 006074 000670      BR       RD1              ;IF HERE GET NEXT WORD-DRIVE FINISHED

948
949 006076 012600      EXIT2:  MOV      (SP)+,RO ;RESTORE RO
950 006100 000207      RTS      PC              ;RETURN TO MAIN LINE CODE

;THE ERROR CHECK ROUTINE CHECKS IF AN ERROR OCCURRED
;ON WRITING OR READING.

951
952
953
954
955 006102 032777 140000 173260      ERRCHK: BIT      @140000,@#KCS ;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,@#KCS       ;IF YES, ISSUE CNTRL RESET + GO
959 006122 004737 005614      JSR      PC,SMTHE
960 006126 012777 177777 173224      MOV      @-1,@#ERRFLG  ;FLAG AN ERROR (PREVENT UPDATE OF ADDR)
961 006134 105777 173230      15:     TSTB    @#KCS       ;CNTRL READY BIT SET (FROM CNTRL RESET)
962 006140 100375      BPL     15              ;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,@#KDS ;SEEK INCOMPLETE SET?
967 006162 000240      NOP                      ;***
968 006164 001530      BEQ     RETRN2          ;BRANCH IF NO
969 006166 012777 000015 173174      MOV      @15,@#KCS      ;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,SMTHE
972 006206 105777 173156      25:     TSTB    @#KCS
973 006212 100375      BPL     25
974 006214 032777 000100 173142      BIT      @100,@#KDS    ;"R/W/S READY" BIT SET?
975 006222 001771      BEQ     25              ;IF NO WAIT! (IF HUNG HERE RUN STATIC)
976 006224 105337 001322      DECB    @#CNTSIN        ;DECREMENT SEEK INCOMPLETE COUNTER
977 006230 001106      BNE     RETRN2          ;IF 3 'SIN' ERRORS FALL THROUGH
978 006232 105737 001321      RESTRT: TSTB    @#ECNT
979 006236 000240      NOP                      ;***
980 006240 001421      BEQ     15
981 006242 104401 006250      TYPE    65$            ;:TYPE ASCIZ STRING
982 006246 000415      BR      64$            ;:GET OVER THE ASCIZ
983
984 006302      64$:   .ASCIZ <15><12>/3 'SIN' ERRORS OCCURRED/

```

```

986 006304
987 006304 104401 006312
988 006310 000412
989
990 006336
991 006336
992 006336 104401 006344
993 006342 000422
994
995 006410
996 006410 105737 001316
997 006414 100006
998 006416 062706 000004
999 006422 012600
1000 006424 052710 100000
1001 006430 000207
1002 006432 052710 100000
1003 006436 012706 001100
1004 006442 000137 003324
1005 006446 000207
1006
1007
1008
1009
1010 006450 010446
1011 006452 010546
1012 006454 105037 001320
1013 006460 000240
1014 006462 012737 000005 001350
1015 006470 012704 007356
1016 006474 013705 001362
1017 006500 020524
1018 006502 001515
1019 006504 105737 001320
1020 006510 001046
1021 006512 104401 006520
1022 006516 000420
1023
1024 006560
1025 006560 152737 000377 001320
1026 006566 113746 001317
1027 006572 104403
1028 006574 001
1029 006575 000
1030 006576 104401 006604
1031 006602 000411
1032
1033 006626
1034 006626
1035 006626 104401 006634
1036 006632 000405
1037
1038 006646
1039 006646 013746 001334
1040 006652 104403

```

```

1S:
TYPE 67$          ;;TYPE ASCIZ STRING
BR 66$           ;;GET OVER THE ASCIZ
;;67$: .ASCIZ <15><12>/5 ERRORS OCCURRED/
66$:
2S:
TYPE 69$          ;;TYPE ASCIZ STRING
BR 68$           ;;GET OVER THE ASCIZ
;;69$: .ASCIZ / DRIVE DECLARED DOWN!! NOT TESTED !/
68$:
TSTB 2#COMND      ;TEST THE COMMAND
BPL 3$           ;IF WRITE, BRANCH
ADD 2#4,SP       ;POINT TO SAVE RD
MOV (SP)+,R0     ;GET RD BACK
BIS 2#BIT15,(R0) ;SET THE DOWN BIT
RTS PC          ;RETURN TO MAIN CODE
3S:
BIS 2#BIT15,(R0) ;SET THE DOWN BIT
MOV 2#STACK,SP  ;RESTORE THE STACK
JMP 2#G01
RETRN2: RTS PC

;THIS ROUTINE CHECKS A SECTORS WORTH OF DATA ON A READ OPERATION
;IT ALLOWS 5 ERROR PRINTOUTS PER SECTOR
RDCHK: MOV R4,-(SP) ;SAVE R4
MOV R5,-(SP) ;SAVE R5 FOR RDLINK
CLRB 2#HDRFLG ;CLEAR THE PRINT HEADER FLAG
NOP ;###
MOV 2#5,2#CHKCNT ;PUT ERROR COUNT IN CHECK COUNT
MOV 2#R0BUFF,R4 ;GET THE TABLE ADDRESS TO R4
MOV 2#PATTRN,R5 ;GET THE EXPECTED DATA TO R5
1S:
CMP R5,(R4)+ ;ARE THEY THE SAME
BEQ 3$ ;IF YES BRANCH
TSTB 2#HDRFLG ;IS THE HEADER FLAG CLEAR
BNE 2$ ;IF NO BRANCH
TYPE 65$ ;TYPE ASCIZ STRING
BR 64$ ;GET OVER THE ASCIZ
;;65$: .ASCIZ <15><12>/ERROR! DATA WRITTEN BY DRIVE /
64$:
BISB 2#377,2#HDRFLG ;SET THE HEADER FLAG
MOVB 2#WRITNBY,-(SP) ;PICK UP THE DRIVE # THAT WROTE
TYPOS
.BYTE 1
.BYTE 0
TYPE 67$ ;TYPE ASCIZ STRING
BR 66$ ;GET OVER THE ASCIZ
;;67$: .ASCIZ / CANNOT BE READ./
66$:
2S:
TYPE 69$ ;TYPE ASCIZ STRING
BR 68$ ;GET OVER THE ASCIZ
;;69$: .ASCIZ <15><12>/ ADDR=/
68$:
MOV 2#DSKADR,-(SP) ;PICK UP THE ADDRESS THAT FAILED
TYPOS

```

```

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

```

```

1098 007220          69$: MOV      R3,-(SP)      ;GET THE WORD COUNT ON THE STACK
1099 007220 010346   TYPOS
1100 007222 104403   .BYTE    6
1101 007224        006   .BYTE    0
1102 007225        000   TYPE    71$      ;;TYPE ASCIZ STRING
1103 007226 104401 007234 BR      70$      ;;GET OVER THE ASCIZ
1104 007232 000401   .ASCIZ  /=/
1105
1106 007236   70$: MOV      (R2)+,-(SP)    ;GET THE FIRST TO THE STACK
1107 007236 012246   TYPOS
1108 007240 104403   .BYTE    6
1109 007242        006   .BYTE    1
1110 007243        001   BIT      #BIT15,-2(R2)  ;WAS THIS THE TABLE TERMINATOR
1111 007244 032762 100000 177776 BNE     5$      ;BRANCH IF YES
1112 007252 001004   BIT      #BIT7,-2(R2)  ;TERMINATOR?
1113 007254 032762 000200 177776 BEQ     4$      ;IF NO BRANCH
1114 007262 001745   5$: TST      -(R0)
1115 007264 005740   HALT
1116 007266 000000
1117
1118 007270   RETFR2: TYPE    65$      ;;TYPE ASCIZ STRING
1119 007270 104401 007276 BR      64$      ;;GET OVER THE ASCIZ
1120 007274 000404   .ASCIZ  <15><12>/WORD=/  
1121
1122 007306   64$: RDOCT
1123 007306 104411   MOV      (SP)+,R2      ;GET THE WORD FROM SYSTEM 2 TO TABLE
1124 007310 012602   BIC     #177400,R2
1125 007312 042702 177400 BIC     #LOGA,R4      ;SET POINTER LOOK FOR FIRST "UP" DRIVE
1126 007316 012704 001166 MOV     #LOGA,R4      ;DRIVE UP?
1127 007322 005724   1$: TST      (R4)+      ;IF NO BRANCH
1128 007324 100776   BMI     1$
1129 007326 010437 001100 MOV     R4,#$PASS
1130 007332 014401   MOV     -(R4),R1
1131 007334 000240   NOP
1132 007336 004737 004224 JSR     PC,LDPLG      ;CALL D02+
1133 007342 004737 005632 JSR     PC,RDLINK     ;CALL READ CHECK
1134 007346 013700 001100 MOV     #,$PASS,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   64$: .ASCIZ  ??/
1150
1151
1152
.SBTTL  OSCILLATING SEEK ROUTINE

```

```

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 010177 170732      1$:     MOV      R1,DRKDA   ;DRIVE #'S IN A TABLE STARTING
1163 010444 105777 170714      TSTB    DRKDS          ;AT 'DRIVO'. BITS 15-13 CONTAINS
1164 010450 100001          BPL      2$            ;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
1171 010472 012777 000001 170670      MOV      #1,DRKCS      ;ISSUE CONTROL RESET + GO !
1172 010500 004737 005614          JSR      PC,SMTME
1173 010504 105777 170660      1$:     TSTB    DRKCS          ;DID CONTROL READY SET?
1174 010510 100375          BPL      1$            ;IF NO WAIT! (IF HUNG RUN STATIC)
1175 010512 012700 020614      MOV      #DRIVO,R0
1176 010516 012077 170654      3$:     MOV      (R0)+,DRKDA
1177 010522 012777 000015 170640      MOV      #15,DRKCS     ;ISSUE DRIVE RESET + GO!
1178 010530 004737 005614          JSR      PC,SMTME
1179 010534 105777 170630      2$:     TSTB    DRKCS
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 000176 001140 7$:     CMP      #SWREG,SWR     ;SOFTWARE SWITCH REGISTER IN USE ?
1192 010732 001002          BNE      9$
1193 010734 104405          GTSWR
1194 010736 000401          BR        8$
1195 010740 000000          9$:     HALT
1196 010742 117704 170172      8$:     MOV      #SWR,R4      ;WAIT FOR OPERATOR TO ENTER NEW SWR VALUE
1197 010746 001413          BEQ      4$
1198 010750 012700 020614      MOV      #DRIVO,R0     ;SW0 TO SW7 TO R4
1199 010754 005001          CLR      R1             ;NONE SET, SO TEST ALL
1200 010756 006004          6$:     ROR      R4            ;TABLE TO STORE DRIVE ADDRS
1201 010760 103001          BCC      5$
1202 010762 010120          MOV      R1,(R0)+
1203 010764 062701 020000      5$:     ADD      #20000,R1
1204 010770 001372          BNE      6$
1205 010772 012720 177777      MOV      #-1,(R0)+
1206 010776 105737 001326      4$:     TSTB    OSPFLG
1207 011002 001165          BNE      RWSRDY
1208 011004 104401 011012      TYPE      ,69$        ;TYPE ASCIZ STRING

```

```

1210          :69$: .ASCIZ <15><12>/TOGGLE THE "FIRST CYLINDER ADDRESS" (OUTER LIMIT)/
1211 011076 68$:          TYPE          71$          ;;TYPE ASCIZ STRING
1212 011076 104401 011104          BR          70$          ;;GET OVER THE ASCIZ
1213 011102 000441          :71$: .ASCIZ <15><12>/INTO THE LOW BYTE (BIT0-7) OF THE SWITCH REGISTER AND THE "LAST
1214          70$:          TYPE          73$          ;;TYPE ASCIZ STRING
1215 011206          BR          72$          ;;GET OVER THE ASCIZ
1216 011206 104401 011214          :73$: .ASCIZ <15><12>/CYLINDER ADDRESS" (INNER LIMIT) INTO THE HIGH/
1217 011212 000430          72$:          TYPE          75$          ;;TYPE ASCIZ STRING
1218          BR          74$          ;;GET OVER THE ASCIZ
1219 011274          :75$: .ASCIZ <15><12>/          BYTE (BIT8-15), THEN PRESS CONTINUE./
1220 011274 104401 011302          74$:          INCB          OSPFLG          ;;TYPE ASCIZ STRING
1221 011300 000424          BIT          #100,ARKDS          ;;GET OVER THE ASCIZ
1222          BEQ          RWSRDY          ;;READ/WRITE/SEEK READY" BIT SET?
1223 011352          CMP          #SWREG,SWR          ;;IF NO WAIT! (IF HUNG RUN STATIC)
1224 011352 105237 001326          BNE          IS          ;;SOFTWARE SWITCH REGISTER IN USE ?
1225 011356 032777 000100 170000 RWSRDY:          BR          IS          ;;BR IF NOT
1226 011364 001774          GTSWR          ;;GET NEW CONTENTS
1227 011366 022737 000176 001140 TRYAGN:          BR          CONTIN          ;;CONTINUE
1228 011374 001002          HALT          CONTIN          ;;CONTINUE
1229 011376 104405          MOV          #1,ARKCS          ;;CONTROL RESET
1230 011400 000401          TSTB          ARKCS          ;;CONTROL RESET
1231 011402 000000          BPL          -4          ;;CONTROL RESET
1232 011404 012777 000001 167756 IS:          MOV          SWR,R5          ;;GET THE ADDRESS OF THE SWITCH REG. TO R5
1233 011412 105777 167752          MOVB         (R5)+,R1          ;;GET A BYTE TO R1
1234 011416 100375          BIC          #177400,R1          ;;CLEAR THE UNUSED BITS
1235 011420 013705 001140          CMP          #312,R1          ;;IS ADDRESS LEGAL?
1236 011424 112501          BPL          2$          ;;BRANCH IF YES
1237 011426 042701 177400          TYPE         65$          ;;TYPE ASCIZ STRING
1238 011432 022701 000312          BR          64$          ;;GET OVER THE ASCIZ
1239 011436 100034          :65$: .ASCIZ <15><12>/INVALID ADDRESS IN SWITCH REGISTER TRY AGAIN/
1240 011440 104401 011446          64$:          BR          TRYAGN          ;;GO BACK FOR NEW PARAMETERS
1241 011444 000430          ROL          R1          ;;ROTATING THIS REGISTER
1242          ROL          R1          ;;MAKES THE BYTE FROM THE
1243 011526          ROL          R1          ;;SWITCH REGISTER LINE UP
1244 011526 000717          ROL          R1          ;;WITH THE CYLINDER BITS OF
1245 011530 006101          ROL          R1          ;;THE RKDA REGISTER.
1246 011532 006101          BIC          #160037,R1          ;;CLEAR THE UNUSED BITS
1247 011534 006101          BIT          #1,R5          ;;IS THIS THE LOW BYTE?
1248 011536 006101          BNE          3$          ;;BRANCH IF YES
1249 011540 006101          MOV          R1,#SEEKI          ;;STORE THE INNER LIMIT OF THE SEEK
1250 011542 042701 160037          BR          SEKSET          ;;START SEEKS
1251 011546 032705 000001          MOV          R1,#SEEKO          ;;STORE THE OUTER LIMIT OF THE SEEK
1252 011552 001003          BR          IS          ;;GO BACK AND GET HIGH BYTE.
1253 011554 010137 001402          3$:          MOV          #50,R3          ;;GET THE NUMBER OF SEEK CYCLES TO R3
1254 011560 000403          MOV          #SEEKO,R4          ;;ADD THE OUTER LIMIT CYLINDER ADDRESS
1255 011562 010137 001404          MOV          #SEEKI,R5          ;;ADD THE INNER LIMIT CYLINDER ADDRESS
1256 011566 000716          LDSEEK: MOV          #DRIVO,R0          ;;INITIALIZE POINTER
1257          SS:          MOV          R4,ARKDA          ;;GET INNER LIMIT CYL ADRES
1258 011570 012703 000050          BIS         (R0)+,ARKDA          ;;SET DRIVE ADRES
1259 011574 013704 001404
1260 011600 013705 001402
1261
1262 011604 012700 020614
1263 011610 010477 167562
1264 011614 052077 167556
    
```

```

1266 011626 004737 005614      JSR    PC,SMTME
1267 011632 105777 167532      TSTB  DRKCS
1268 011636 100375                BPL    JS
1269
1270 011640 022710 177777      CMP    #-1,(RO)      ;ALL DRIVES DONE?
1271 011644 001361                BNE                    ;NO
1272 011646 012700 020614      MOV    #DRIVO,RO
1273 011652 012077 167520      MOV    (RO)+,DRKDA
1274 011656 032777 000100 167500 6S:  BIT    #RWS,DRKDS      ;ADRES THE DRIVE
1275 011664 001774                BEQ    7S              ;SEEK DONE?
1276 011666 022710 177777      CMP    #-1,(RO)      ;NO, WAIT
1277 011672 001367                BNE                    ;ALL DRIVES DONE?
1278                                ;NO
1279 011674 012700 020614      MOV    #DRIVO,RO
1280 011700 010577 167472      MOV    R5,DRKDA      ;SET OUTER CYL ADRES
1281 011704 052077 167466      BIS    (RO)+,DRKDA   ;SET DRIVE ADRES
1282 011710 012777 000011 167452      MOV    #11,DRKCS     ;ISSUE SEEK+GO! (FOR OUTER LIMIT)
1283 011716 004737 005614      JSR    PC,SMTME
1284 011722 105777 167442      TSTB  DRKCS
1285 011726 100375                BPL    9S
1286
1287 011730 022710 177777      CMP    #-1,(RO)      ;ALL DONE?
1288 011734 001361                BNE                    ;NO
1289
1290 011736 012700 020614      MOV    #DRIVO,RO
1291 011742 012077 167430      MOV    (RO)+,DRKDA   ;SET DRIVE ADRES
1292 011746 032777 000100 167410 10S: BIT    #RWS,DRKDS     ;SEEK DONE?
1293 011754 001774                BEQ    11S
1294 011756 022710 177777      CMP    #-1,(RO)      ;ALL DONE?
1295 011762 001367                BNE                    ;NO
1296 011764 005303                DEC    R3              ;DONE 50 SEEK CYCLES (100 SEEKS)
1297 011766 001306                BNE                    ;IF NO BRANCH (KEEP CYCLING)
1298 011770 000605                BR     LDSEEK          ;CHECK SWR FOR CHANGE AND CONTINUE
1299
1300
1301
1302
1303
1304
1305

```

.SBTTL FORMATTER-SURFACE VERIFIER

```

1306 011772                BAD.IN:
1307 011772 104401 012000      TYPE  65S              ;;TYPE ASCIZ STRING
1308 011776 000401                BR    64S              ;;GET OVER THE ASCIZ
1309
1310                ;;65S: .ASCIZ  ???
1311                64S:
1312 012002 104401 012010      SECT.1: TYPE  65S              ;;TYPE ASCIZ STRING
1313 012006 000441                BR    64S              ;;GET OVER THE ASCIZ
1314                ;;65S: .ASCIZ <15><12>/FORMATTER-SURFACE VERIFIER,SET SW REG FOR DRV #'S. PRESS CONT./
1315                64S:
1316 012112 022737 000176 001140      CMP    #SWREG,SWR     ;SOFTWARE SWITCH REGISTER IN USE ?
1317 012120 001002                BNE                    ;BR IF NOT
1318 012122 104405                GTSWR                  ;GET SWITCH REGISTER VALUE
1319 012124 000401                BR     2S              ;CONTINUE
1320 012126 000000      1S:  HALT              ;WAIT FOR 'CONTINUE'

```

1322	012136	005037	020446			CLR	DRVCNT	; CLEAR DRIVE COUNT
1323	012142	033777	020444	166770	S13:	BIT	SHFCNT, JSWR	; IS THIS SW SET?
1324	012150	001011				BNE	S10	; YES FORMAT THIS DRIVE
1325	012152	006337	020444		S11:	ASL	SHFCNT	
1326	012155	005237	020446			INC	DRVCNT	
1327	012162	022737	000010	020446		CMP	#10, DRVCNT	; ALL DONE?
1328	012170	001561				BEQ	GO1	
1329	012172	000763				BR	S13	
1330	012174	013700	020446		S10:	MOV	DRVCNT, RO	
1331	012200	104401	001161			TYPE	, SCRLF	; TYPE 'DRIVE'
1332	012204	104401	020450			TYPE	, EM1	; TYPE DRIVE #
1333	012210	010046				MOV	RO, -(SP)	
1334	012212	104402				TYPOC		
1335	012214	000300				SWAB	RO	
1336	012216	006100				ROL	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, @#DSKTN.	
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		BIS	@#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.


```

1378 012502 104401 012510          TYPE      655          ;:TYPE ASCIZ STRING
1379 012506 000411                BR          645          ;:GET OVER THE ASCIZ
1380                                     ;:655: .ASCIZ <CR><LF>/PACK GOOD /
1381 012532                645:
1382 012532 000607                BR          511
1383 012534 000137 001440        GD1: JMP      0#START      ;RESTART
1384
1385                                     ;:
1386                                     ;:DISK I/O SUBROUTINE.
1387
1388                                     ;:
1389                                     ;:SET UP FOR A WRITE/FORMAT.
1390 012540 013777 001412 166624        10: MOV      WC,0RKWC      ;:SET UP THE WORD COUNT REG.
1391 012546 013777 001410 166622        MOV      DA,0RKDA      ;:SET UP THE DISK ADDRESS.
1392 012554 053777 001352 166614        BIS      0#DSKTMP,0RKDA ;:SET THE UNIT NUMBER UP.
1393 012562 013777 001406 166604        MOV      BA,0RKBA      ;:SET UP THE BUSS ADDRESS.
1394 012570 012777 000000 166572        MOV      00,0RKCS      ;:CLEAR THE CONTROL REG. FOR SET UP.
1395 012576 052777 006000 166564        BIS      #BIT10+BIT11,0RKCS ;:SET FORMAT&INHIBT INC. BITS.
1396 012604 052777 000002 166556        BIS      #BIT1,0RKCS    ;:SET UP WRITE FUN.
1397 012612 053777 001420 166550        BIS      EXTR,0RKCS     ;:SET UP 12OR16 SECTOR PACK.
1398 012620 052777 000001 166542        BIS      #BIT0,0RKCS    ;:GO DO THE WRITE FORMAT.
1399 012626 105777 166536        15: TSTB     0RKCS        ;:IS WRITE FORMAT DONE?
1400 012632 100375 166530        BPL      15             ;:NO SO WAIT.
1401 012634 005777 166530        TST      0RKCS         ;:WAS THERE A ERROR?
1402 012640 100541 166530        BMI      WFERR         ;:YES GO SERVICE IT.
1403 012642 012737 000000 001422        MOV      00,ERRWF      ;:CLEAR OUT THE ERROR COUNTER.
1404
1405                                     ;:
1406                                     ;:SET UP FOR A READ/FORMAT
1407 012650 013777 001416 166514        MOV      RWC,0RKWC     ;:SET UP WORD COUNT REG.
1408 012656 013777 001410 166512        MOV      DA,0RKDA      ;:SET UP DISK ADDRESS.
1409 012664 053777 001352 166504        BIS      0#DSKTMP,0RKDA ;:SET THE UNIT NUMBER
1410 012672 013777 001414 166474        MOV      RBA,0RKBA     ;:SET UP THE BUSS ADDRESS.
1411 012700 012777 000000 166462        MOV      00,0RKCS      ;:CLEAR THE CONTROL REG.
1412 012706 052777 002000 166454        BIS      #BIT10,0RKCS   ;:SET THE FORMAT BIT.
1413 012714 052777 000004 166446        BIS      #BIT2,0RKCS    ;:SET UP READ FUN.
1414 012722 053777 001420 166440        BIS      EXTR,0RKCS     ;:SET UP 12 OR 16 SECTOR PACK.
1415 012730 052777 000001 166432        BIS      #BIT0,0RKCS    ;:GO DO THE READ FORMAT.
1416 012736 105777 166426        25: TSTB     0RKCS        ;:IS THE READ FORMAT DONE?
1417 012742 100375 166426        BPL      25             ;:NO SO WAIT.
1418 012744 032777 040000 166416        BIT      #BIT14,0RKCS   ;:WAS TRERE A ERROR?
1419 012752 001134 166416        BNE      RFERR         ;:YES GO SERVICE IT.
1420 012754 012737 000000 001424        MOV      00,ERRRF      ;:CLEAR OUT THE ERROR COUNT.
1421
1422                                     ;:
1423                                     ;:SET UP FOR A WRITE CHECK.
1424 012762 013777 001412 166402        MOV      WC,0RKWC     ;:SET UP WORD COUNT REG.
1425 012770 013777 001410 166400        MOV      DA,0RKDA      ;:SET UP DISK ADDRESS.
1426 012776 053777 001352 166372        BIS      0#DSKTMP,0RKDA ;:SET UP THE UNIT NUMBER
1427 013004 013777 001406 166362        MOV      BA,0RKBA      ;:SET UP BUSS ADDRESS.
1428 013012 012777 000000 166350        MOV      00,0RKCS      ;:CLEAR THE CONTROL REG.
1429 013020 052777 004400 166342        BIS      #BIT11+BIT8,0RKCS ;:SET INHIBIT INCR.&STOP ON SOFT ERROR BITS.
1430 013026 053777 001420 166334        BIS      EXTR,0RKCS     ;:SET 12 OR 16 SECTOR PACK.
1431 013034 052777 000006 166326        BIS      #BIT1+BIT2,0RKCS ;:SET UP WRITE CHECK FUN.
1432 013042 052777 000001 166320        BIS      #BIT0,0RKCS    ;:GO DO THE WRITE CHECK.

```

```

1434 ;CHECK HEADERS READ BY THE READ/FORMAT.
1435 ;
1436 013050 013703 001416      MOV      RWC,R3      ;PUT NUMBER OF WORDS TO
1437 013054 005403              NEG      R3          ;CHECK IN REG 3.
1438 013056 063703 001414      ADD      RBA,R3      ;SET REG 3 TO THE LAST WORD TO BE CHECKED.
1439 013062 013702 001414      MOV      RBA,R2      ;SET REG 2 TO STARTING ADD. OF BUFF.
1440 013066 023722 001410      MORE:    CMP      DA,(2)+ ;CHECK THAT HEADER IS RIGHT.
1441 013072 001073              BNE     RFCERR       ;THIS HEADER WAS WRONG GO SERVICE IT.
1442 013074 020302              CMP      R3,R2       ;ARE WE DONE?
1443 013076 001373              BNE     MORE         ;NO GO CHECK THE NEXT ONE.
1444 013100 012737 000000 001426  NOV      #0,ERRRFC    ;CLEAR OUT THE ERROR COUNT.
1445 ;
1446 ;LETS CHECK ON THE WRITE CHECK WE STARTED.
1447 ;
1448 013106 105777 166256      IS:      TSTB     @RKCS ;THE CONTROLLER IS STILL BUSY.
1449 013112 100375              BPL     IS          ;WAS THERE A ERROR?
1450 013114 005777 166250      TST     @RKCS       ;YES GO SERVICE IT.
1451 013120 100407              BMI     WCERRR      ;YES GO SERVICE IT.
1452 013122 012737 000000 001430  MOV      #0,ERRWCH   ;CLEAR OUT THE
1453 013130 012737 000000 001432  MOV      #0,ERRWCS   ;ERROR COUNTERS.
1454 013136 000201              RTS     R1          ;RETURN TO THE MAIN LINE.
1455 013140 000137 013624      WCERRR: JMP      WCERR
1456 ;
1457 ;ERRORS FOR WRITE FORMAT.
1458 ;
1459 013144 005237 001422      WFERR:  INC      ERRWF ;ADD ONE TO THE ERROR COUNT.
1460 013150 022737 000004 001422  CMP      #4,ERRWF   ;HAS IT HAPPEND 4 TIMES ON THIS CYL.
1461 013156 001016              BNE     RETRY       ;NO.
1462 ;
1463 013160 104401 013166      SYSER:  TYPE     65$   ;:TYPE ASCIZ STRING
1464 013164 000410              BR      64$         ;:GET OVER THE ASCIZ
1465 ;
1466 65$: .ASCIZ <15><12>/SYSTEM ERROR/
1467 64$:
1468 013206 000000              HALT
1469 013210 000137 001440      JMP     START       ;LET THE TECH. BREATH.
1470 013214 012777 000000 166146  MOV      #0,@RKCS   ;RESTART THE TEST.
1471 013222 012777 000015 166140  MOV      #15,@RKCS ;CLEAR OUT THE CONTROL REG.
1472 013230 032777 000100 166126  IS:      BIT      @BIT6,@RKDS ;DO A DRIVE RESET.
1473 013240 000137 012540      BEQ     IS          ;IS IT DONE.
1474 ;
1475 ;ERRORS FOR READ/FORMAT.
1476 ;
1477 013244 005237 001424      RFERR:  INC      ERRRF ;ADDONE TO ERROR COUNT.
1478 013250 022737 000004 001424  CMP      #4,ERRRF   ;HAS IT HAPPEND 4 TIMES ON THIS CYL?
1479 013256 001356              BNE     RETRY       ;NO DO IT AGAIN.
1480 013260 000737              BR      SYSER       ;YES SO TELL HIM SO.
1481 ;
1482 ;READ/FORMAT ERRORS FOUND BY SOFTWARE CHECKS.
1483 ;
1484 013262 005237 001426      RFCERR: INC      ERRRFC ;ADD ONE TO ERROR COUNT.
1485 013266 105777 166076      IS:      TSTB     @RKCS ;WAIT FOR THE WRITE CHECK.
1486 013272 100375              BPL     IS          ;
1487 013274 022737 000004 001426  CMP      #4,ERRRFC  ;IS IT 4?
1488 013302 001401              BEQ     FAILED     ;PUT OUT FAILED MESSAGE.

```

```

1490 013306 042777 000037 166062 FAILED: BIC      #37,ARKDA      ;PUT WHICH SECTORS HEADER
1491 013314 042702 177740          BIC      #177740,R2
1492 013320 060277 166052          ADD      R2,ARKDA      ;FAILED IN RKDA FOR THE MESSAGE.
1493 013324          FAIL:
1494 013324 104401 013332          TYPE     655          ;;TYPE ASCIZ STRING
1495 013330 000417          BR       645          ;;GET OVER THE ASCIZ
1496          ;;655: .ASCIZ <15><12>/PACK FAILED AT (IN OCTAL) /
1497          645:
1498 013370 017701 166002          MOV     ARKDA,R1      ;GENERAT THE CYL,SECTOR,SURFACE
1499 013374 010102          MOV     R1,R2      ;MESSAGE FROM RKDA
1500 013376 042702 177770          BIC     #177770,R2
1501 013402 062702 000260          ADD     #260,R2
1502 013406 110237 013565          MOVB   R2,SEC+1
1503 013412 004337 013612          JSR    R3,SHF3
1504 013416 042702 177776          BIC     #177776,R2
1505 013422 062702 000260          ADD     #260,R2
1506 013426 110237 013564          MOVB   R2,SEC
1507 013432 004337 013616          JSR    R3,SHF1
1508 013436 042702 177776          BIC     #177776,R2
1509 013442 062702 000260          ADD     #260,R2
1510 013446 110237 013576          MOVB   R2,SUR
1511 013452 004337 013616          JSR    R3,SHF1
1512 013456 042702 177770          BIC     #177770,R2
1513 013462 062702 000260          ADD     #260,R2
1514 013466 110237 013554          MOVB   R2,CYL+2
1515 013472 004337 013612          JSR    R3,SHF3
1516 013476 042702 177770          BIC     #177770,R2
1517 013502 062702 000260          ADD     #260,R2
1518 013506 110237 013553          MOVB   R2,CYL+1
1519 013512 004337 013612          JSR    R3,SHF3
1520 013516 042702 177774          BIC     #177774,R2
1521 013522 062702 000260          ADD     #260,R2
1522 013526 110237 013552          MOVB   R2,CYL
1523 013532 104401 013542          TYPE     1$          ;TYPE OUT THE GENERATED MESSAGE
1524 013536 000137 013604          JMP     STOP          ;BYPASS THE INLINE MESSAGE
1525 013542 005015 054503 027114 1$: .ASCII <15><12>/CYL.
1526 013550 020040          CYL: .ASCII /000 /
1527 013552 030060 020060          SEC: .ASCII /SEC. /
1528 013556 042523 027103          SEC: .ASCII /00 /
1529 013564 030160 040          .ASCII /SURF. /
1530 013567 123 051125 027106          .ASCII /SURF. /
1531 013574 020040          SUR: .ASCIZ /0 /<15><12>
1532 013576 020060 005015 000 .EVEN
1533          STOP: HALT
1534 013604 000000          JMP     START          ;LET OPER DO HIS THING.
1535 013606 000137 001440          ;RESTART THE TEST.
1536          ;SHIFT SUBROUTINES.
1537          SHF3: ASR     R1          ;HERE FOR A SHIFT OF 3.
1538          SHF2: ASR     R1          ;HERE FOR A SHIFT OF 2.
1539 013612 006201          SHF1: ASR     R1          ;HERE FOR A SHIFT OF 1.
1540 013614 006201          MOV     R1,R2          ;PUT RESULTES IN THE WORKING REG.
1541 013616 006201          RTS    R3
1542 013620 010102
1543 013622 000203
1544

```

```

1546
1547 013624 032777 040000 165536 WCERR: BIT #BIT14,DRKCS ::WAS IT A HARD ERROR.
1548 013632 001010 BNE WCHERR' ::YES GO PROSSES 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 IO ::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 .SBTTL RK05 CONTROL PANEL TEST
1569
1570 013762 BAD.ON:
1571 013762 104401 013770 TYPE 65$ ::TYPE ASCIZ STRING
1572 013766 000401 BR 64$ ::GET OVER THE ASCIZ
1573 ::65$: .ASCIZ /?/
1574 013772 64$:
1575 013772 SECT.0:
1576 013772 104401 014000 TYPE 65$ ::TYPE ASCIZ STRING
1577 013776 000424 BR 64$ ::GET OVER THE ASCIZ
1578 ::65$: .ASCIZ <15><12>/RK05 CONTROL PANEL TEST, WHICH DRIVE?/
1579 64$:
1580 014050 RDCHR
1581 014052 104407 MOV (SP),RO
1582 014054 104403 TYPOS
1583 014056 001 .BYTE 1
1584 014057 000 .BYTE 0
1585 014060 162700 000060 SUB #60,RO
1586 014064 100736 BMI BAD.ON
1587 014066 022700 000010 CMP #10,RO
1588 014072 003733 BLE BAD.ON
1589 014074 110037 001314 MOVB RO,#DRIVE
1590 014100 000300 SWAB 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
1598 014124 000414 BR 66$ ::GET OVER THE ASCIZ
1599 ::67$: .ASCIZ <15><12>/MOUNT PACK ON DRIVE#/
1600 66$: 014156

```

```

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          70$:          BR 72$      ;;GET OVER THE ASCIZ
1613 014322 104401 014330   ;;73$: .ASCIZ <15><12>/MAKE DRIVE WRITE ENABLE PRESS CONTINUE/
1614 014326 000425          72$:          HALT
1615          ;;73$:          JSR PC,WRDCK      ;;GO WRITE 0'S, RD 0'S, CHECK
1616 014402 000000          TYPE 75$      ;;TYPE ASCIZ STRING
1617 014402 000000          BR 74$      ;;GET OVER THE ASCIZ
1618 014404 004737 016260   ;;75$: .ASCIZ <15><12><15><12>/WRITE PROTECT THE DRIVE THEN PRESS CONTINUE/
1619 014410 104401 014416   74$:          HALT
1620 014414 000430          JSR PC,WRPRO      ;;GO TRY OVERWRITE
1621          TYPE 77$      ;;TYPE ASCIZ STRING
1622 014476 000000          BR 76$      ;;GET OVER THE ASCIZ
1623 014476 000000          ;;77$: .ASCIZ <15><12><15><12>/CLEAR WRITE PROTECT THEN PRESS CONTINUE/
1624 014500 004737 016470   76$:          HALT
1625 014504 104401 014512   JSR PC,WRDCK      ;;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 000000          JSR PC,SMTME      ;;GO WAIT TIME FOR RK11-C
1629 014570 004737 016260   TSTB @RKCS        ;;IS CONTROL READY SET
1630 014574 013777 001352 164574   BPL 1$            ;;IF NO, BRANCH
1631 014574 013777 001352 164560   JSR PC,WRPRO      ;;GO TRY OVERWRITE OF IERO'S
1632 014602 012777 000017          TYPE 65$      ;;TYPE ASCIZ STRING
1633 014610 004737 005614   BR 64$      ;;GET OVER THE ASCIZ
1634 014614 105777 164550   1$:          .ASCIZ <15><12><15><12>/CAUTION! TRY TO OPEN THE DOOR, DO NOT FORCE:/
1635 014620 100375          64$:          TYPE 67$      ;;TYPE ASCIZ STRING
1636 014622 004737 016470   BR 66$      ;;GET OVER THE ASCIZ
1637 014626          66$:          .ASCIZ <15><12>/DOOR SHOULD NOT OPEN!/
1638 014626 104401 014634   ;;67$:          TYPE 69$      ;;TYPE ASCIZ STRING
1639 014632 000431          BR 68$      ;;GET OVER THE ASCIZ
1640          68$:          .ASCIZ <15><12>/PRESS CONTINUE WHEN FINISHED/
1641 014716          PRTTWO:      TYPE 71$      ;;TYPE ASCIZ STRING
1642 014716 104401 014724   BR 70$      ;;GET OVER THE ASCIZ
1643 014722 000414          ;;71$: .ASCIZ <15><12><15><12>/PUT DRIVE IN LOAD, WAIT FOR LOAD LIGHT/
1644          70$:          TYPE 73$      ;;TYPE ASCIZ STRING
1645 014754          BR 72$      ;;GET OVER THE ASCIZ
1646 014754 104401 014762   ;;69$:          TYPE 75$      ;;TYPE ASCIZ STRING
1647 014760 000420          BR 74$      ;;GET OVER THE ASCIZ
1648          74$:          HALT
1649 015022          TYPE 77$      ;;TYPE ASCIZ STRING
1650 015022 000000          BR 76$      ;;GET OVER THE ASCIZ
1651 015024 104401 015032   ;;77$: .ASCIZ <15><12><15><12>/WRITE PROTECT THE DRIVE THEN PRESS CONTINUE/
1652 015030 000426          76$:          HALT
1653          JSR PC,WRDCK      ;;GO WRITE 0'S, RD 0'S, CHECK
1654 015106          MOV @DSKTMP,@RKDA ;;SET UP DRIVE
1655 015106 104401 015114   MOV @17,@RKCS      ;;FUNCTION WRITE PROTECT
1656 015112 000420          JSR PC,SMTME      ;;GO WAIT TIME FOR RK11-C

```

```

1658 015154          72S:
1659 015154          HALT
1660 015156 01377 001352 164212      MOV      2#DSKTMP,ARKDA      ;SET UP DISK ADDRESS
1661 015164 012777 000015 164176      MOV      #15,ARKCS          ;ISSUE A DRIVE RESET
1662 015172 004737 005614          JSR      PC,SMTME           ;KILL TIME FOR RK11-C
1663 015176 105777 164166          TSTB    ARKCS              ;CONTROL READY SET?
1664 015200 100375          BPL     15                  ;IF NO, BRANCH
1665 015204 032777 100200 164154      BIT      #100200,ARKER      ;DRE SET
1666 015212 001057          BNE     25                  ;IF YES BRANCH
1667 015214 104401 015222          TYPE    75$                ;TYPE ASCIZ STRING
1668 015220 000427          BR      74$                ;GET OVER THE ASCIZ
1669          ;:75$: .ASCIZ <15><12>/DRE=BIT15 OF RKER DID NOT SET IF AN RK11-C/
1670 015300          74$:
1671 015300 104401 015306          TYPE    77$                ;:TYPE ASCIZ STRING
1672 015304 000422          BR      76$                ;:GET OVER THE ASCIZ
1673          ;:77$: .ASCIZ <15><12>/OR BIT07 DID NOT SET IF AN RK11-D/
1674 015352          76$:
1675 015352 032777 140000 164010      2S:    BIT      #140000,ARKCS ;DID HARD ERROR ON ERROR SET
1676 015360 001015          BNE     3$                  ;BRANCH IF YES
1677 015362 104401 015370          TYPE    79$                ;:TYPE ASCIZ STRING
1678 015366 000412          BR      78$                ;:GET OVER THE ASCIZ
1679          ;:79$: .ASCIZ <15><12>/ERROR DID NOT SET/
1680 015414          78$:
1681 015414 012777 000001 163746      3S:    MOV      #1,ARKCS          ;ISSUE A CONTROL RESET
1682 015422 004737 005614          JSR      PC,SMTME           ;WAST TIME
1683 015426 105777 163736          4S:    TSTB    ARKCS              ;CONTROL READY SET
1684 015432 100375          BPL     4$                  ;IF NO, BRANCH
1685 015434 032777 100000 163724      BIT      #BIT15,ARKER      ;'DRE' CLEAR
1686 015442 001425          BEQ     5$                  ;IF YES BRANCH
1687 015444 104401 015452          TYPE    #1$                ;:TYPE ASCIZ STRING
1688 015450 000422          BR      #0$                ;:GET OVER THE ASCIZ
1689          ;:81$: .ASCIZ <15><12>/CONTROL RESET DID NOT CLEAR 'DRE'/
1690 015516          80$:
1691 015516 032777 140000 163644      5S:    BIT      #140000,ARKCS ;ERROR BITS CLEAR
1692 015524 001431          BEQ     X$                  ;IF YES BRANCH
1693 015526 104401 015534          TYPE    #83$               ;:TYPE ASCIZ STRING
1694 015532 000426          BR      #82$               ;:GET OVER THE ASCIZ
1695          ;:83$: .ASCIZ <15><12>/CONTROL RESET DID NOT CLEAR 'ERROR', RKCS/
1696 015610          82$:
1697 015610          X:
1698 015610 104401 015616          TYPE    #65$               ;:TYPE ASCIZ STRING
1699 015614 000422          BR      #64$               ;:GET OVER THE ASCIZ
1700          ;:65$: .ASCIZ <15><12><15><12>/OPEN THE DOOR, PUT DRIVE IN RUN/
1701 015662          64$:
1702 015662 104401 015670          TYPE    #67$               ;:TYPE ASCIZ STRING
1703 015666 000425          BR      #66$               ;:GET OVER THE ASCIZ
1704          ;:67$: .ASCIZ <15><12>/CAUTION! IF RUN LIGHT ON ERROR! DEPRESS/
1705 015742          66$:
1706 015742 104401 015750          TYPE    #69$               ;:TYPE ASCIZ STRING
1707 015746 000426          BR      #68$               ;:GET OVER THE ASCIZ
1708          ;:69$: .ASCIZ <15><12>/LOAD IMMEDIATELY, CONTINUE WHEN FINISHED/
1709 016024          68$:
1710 016024 000000          XX:    HALT
1711 016026 104401 016034          TYPE    #65$               ;:TYPE ASCIZ STRING
1712 016032 000422          BR      #64$               ;:GET OVER THE ASCIZ

```

1714	016100				64S:				
1715	016100	104401	016106			TYPE	67S	::TYPE ASCIZ STRING	
1716	016104	000423				BR	66S	::GET OVER THE ASCIZ	
1717					::67S:	.ASCIZ	<15><12>/PUT DRIVE IN RUN, DRIVE SHOULD NOT/		
1718	016154				66S:				
1719	016154	104401	016162			TYPE	69S	::TYPE ASCIZ STRING	
1720	016160	000423				BR	68S	::GET OVER THE ASCIZ	
1721					::69S:	.ASCIZ	<15><12>/RUN...INTERLOCKS HAVE BEEN CHECKED/		
1722	016230				68S:				
1723	016230	104401	016236			TYPE	71S	::TYPE ASCIZ STRING	
1724	016234	000407				BR	70S	::GET OVER THE ASCIZ	
1725					::71S:	.ASCIZ	<15><12>/DONE!		
1726	016254				70S:				
1727	016254	000137	001440			JMP	2#START		
1728	016260	005037	001362		WRRDCK:	CLR	2#PATTRN	:MAKE A PATTERN OF ZERO'S	
1729	016264	013777	001352	163104		MOV	2#DSKTMP,2#RKDA	:SET UP DRIVE ADDRESS	
1730	016272	012777	000001	163070		MOV	#1,2#RKCS	:ISSUE A CONTROL RESET	
1731	016300	004737	005614			JSR	PC,2#TME		
1732	016304	105777	163060		5S:	TSTB	2#RKCS	:CONTROL READY SET	
1733	016310	100375				BPL	5S	:IF NO BRANCH	
1734	016312	013777	001352	163056		MOV	2#DSKTMP,2#RKDA	:SET UP DRIVE ADDRESS	
1735	016320	012777	001362	163046		MOV	2#PATTRN,2#RKBA	:GET BUSS ADDRESS	
1736	016326	013777	001344	163036		MOV	2#SECCNT,2#RKWC	:WORD COUNT 1 SECTOR	
1737	016334	013777	001354	163026		MOV	2#WAITCS,2#RKCS	:IBA + WRITE + GO	
1738	016342	004737	005614			JSR	PC,2#TME	:KILL TIME FOR RK11-C	
1739	016346	105777	163016		1S:	TSTB	2#RKCS	:CONTROL READY SET	
1740	016352	100375				BPL	1S	:IF NO BRANCH	
1741	016354	013777	001352	163014		MOV	2#DSKTMP,2#RKDA	:SET UP RK REGISTERS	
1742	016362	012777	007356	163004		MOV	2#RDBUFF,2#RKBA	:TO READ ONE SECTOR	
1743	016370	013777	001344	162774		MOV	2#SECCNT,2#RKWC	:TO THE READ BUFFER	
1744	016376	013777	001356	162764		MOV	2#READCS,2#RKCS		
1745	016404	004737	005614			JSR	PC,2#TME	:KILL TIME FOR RK11-C	
1746	016410	105777	162754		2S:	TSTB	2#RKCS	:CONTROL READY SET	
1747	016414	100375				BPL	2S	:IF NO BRANCH	
1748	016416	012704	007356			MOV	2#RDBUFF,R4	:GET BUFFER TO R4	
1749	016422	005005				CLR	R5	:SET UP TO COMPARE	
1750	016424	020524			3S:	CMP	R5,(R4)+	:FOR ZERO'S	
1751	016426	001414				BEQ	4S	:IF OK, BRANCH	
1752	016430	104401	016436			TYPE	65S	:TYPE ASCIZ STRING	
1753	016434	000410				BR	64S	:GET OVER THE ASCIZ	
1754					::65S:	.ASCIZ	<15><12>/WRITE FAILED/		
1755	016456				64S:				
1756	016456	000700				BR	WRRDCK	:GO BACK TRY AGAIN	
1757	016460	022704	010356		4S:	CMP	2#MANSEL,R4	:DONE ALL CHECKS	
1758	016464	001357				BNE	3S	:IF NO BRANCH	
1759	016466	000207				RTS	PC	:IF YES, RETURN	
1760	016470	032777	000040	162666	WRPRO:	BIT	2#BITS,2#RKDS	:BIT 5 ON	
1761	016476	001021				BNE	1S	:IF YES, BRANCH	
1762	016500	104401	016506			TYPE	65S	:TYPE ASCIZ STRING	
1763	016504	000416				BR	64S	:GET OVER THE ASCIZ	
1764					::65S:	.ASCIZ	<15><12>/WPS=BITS OF RKDS NOT SET/		
1765	016542				64S:				
1766	016542	012737	177777	001362	1S:	MOV	2#177777,2#PATTRN	:GO LOAD ALL	
1767	016550	013777	001352	162620		MOV	2#DSKTMP,2#RKDA	:RK REGISTERS	
1768	016556	012777	001362	162610		MOV	2#PATTRN,2#RKBA	:TO WRITE	

```

1770 016572 013777 001354 162570      MOV      2#WRITCS,2#RKCS      : OVER THE ZERO'S
1771 016600 004737 005614          JSR      PC,SMTME           : KILL TIME FOR RK11-C
1772 016604 105777 162560      2S:     TSTB      2#RKCS      : CONTROL READY SET?
1773 016610 100375          BPL      2S                : IF NO BRANCH
1774 016612 032777 020000 162546      BIT      2#BIT13,2#RKER     : WLO BIT SET
1775 016620 001032          BNE      3S                : IF YES BRANCH
1776 016622 104401 016630      TYPE    67S                : TYPE ASCIZ STRING
1777 016626 000427          BR       66S                : GET OVER THE ASCIZ
1778          ::67S: .ASCIZ <15><12>/EXPECTED WLO=BIT13 OF RKER BUT DID NOT SET/
1779 016706 66S:
1780 016706 012777 000001 162454      3S:     MOV      2#1,2#RKCS     : DO A CONTROL RESET
1781 016714 004737 005614          JSR      PC,SMTME           : KILL TIME FOR RK11-C
1782 016720 105777 162444      4S:     TSTB      2#RKCS      : CONTROL READY SET?
1783 016724 100375          BPL      4S                : IF NO BRANCH
1784 016726 032777 020000 162432      BIT      2#BIT13,2#RKER     : WLO BIT CLEAR
1785 016734 001431          BEQ      RDCHKD            : IF YES BRANCH
1786 016736 104401 016744      TYPE    69S                : TYPE ASCIZ STRING
1787 016742 000426          BR       68S                : GET OVER THE ASCIZ
1788          ::69S: .ASCIZ <15><12>/CONTROL RESET DID NOT CLEAR 'WLO' OF RKER/
1789 017020 68S:
1790 017020 013777 001352 162350      RDCHKD: MOV      2#DSKTMP,2#RKDA    : SET UP RK REGISTERS
1791 017026 012777 007356 162340      MOV      2#RDBUFF,2#RKBA    : TO READ SECTOR 0,
1792 017034 013777 001344 162330      MOV      2#SECCNT,2#RKWC    : CYLINDER 0, HEAD 0
1793 017042 013777 001356 162320      MOV      2#READCS,2#RKCS    : TO ENSURE NO WRITE TOOK PLACE
1794 017050 004737 005614          JSR      PC,SMTME           : KILL TIME
1795 017054 105777 162310      3S:     TSTB      2#RKCS      :
1796 017060 100375          BPL      3S                :
1797 017062 012703 000005      MOV      2#5,R3             : CHECK TO INSURE NO WRITE
1798 017066 012704 007356      MOV      2#RDBUFF,R4        : TOOK PLACE
1799 017072 005005          CLR      R5                 : WITH WRITE LOCK
1800 017074 020524          1S:     CMP      R5,(R4)+      :
1801 017076 001474          BEQ      2S                 :
1802 017100 005303          DEC      R3                 : DEC THE ERROR COUNT
1803 017102 001475          BEQ      4S                 : IF ZERO BRANCH
1804 017104 104401 017112      TYPE    65S                : TYPE ASCIZ STRING
1805 017110 000422          BR       64S                : GET OVER THE ASCIZ
1806          ::65S: .ASCIZ <15><12>/WRITE OCCURRED WITH WRITE PROTECT/
1807 017156 64S:
1808 017156 005744          TST      -(R4)              :
1809 017160 104401 017166      TYPE    67S                : TYPE ASCIZ STRING
1810 017164 000410          BR       66S                : GET OVER THE ASCIZ
1811          ::67S: .ASCIZ <15><12>/BUFFER ADDR=/
1812 017206 66S:
1813 017206 010446          MOV      R4,-(SP)           :
1814 017210 104403          TYPOS   6                   :
1815 017212 001          .BYTE   1                   :
1816 017213 001          .BYTE   1                   :
1817 017214 104401 017222      TYPE    69S                : TYPE ASCIZ STRING
1818 017220 000406          BR       68S                : GET OVER THE ASCIZ
1819          ::69S: .ASCIZ / EXPCTD=/
1820 017236 68S:
1821 017236 010546          MOV      R5,-(SP)           :
1822 017240 104404          TYPON   7                   :
1823 017242 104401 017250      TYPE    71S                : TYPE ASCIZ STRING
1824 017246 000406          BR       70S                : GET OVER THE ASCIZ

```


E05

MAINDEC-11-DZRKI-E MACY11 30A(1052) 24-MAR-78 09:23 PAGE 36
DZRKIF.P11 24-MAR-78 09:20 RK05 CONTROL PANEL TEST

SEQ 0056

1826 017264
1827 017264 012446
1828 017266 104404
1829 017270 022704 010356
1830 017274 001277
1831 017276 000207
1832
1833
1834

70S: MOV (R4)+, -(SP)
TYPON
2S: CMP #MANSEL, R4 ; FINISHED ALL CHECKS
BNE IS ; IF NO, BRANCH
4S: RTS PC ; RETURN

1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890

017300 000240
017302 012777 000001 162060
017310 105777 162054
017314 100375
017316 012700 020614
017322 005001
017324 005002

017326 010210
017330 010277 162042
017334 105777 162024
017340 100021

017342 104401 020450
017346 010146
017350 104402
017352 104401 020461
017356 052710 000300

017362 012777 000015 162000
017370 004737 005614
017374 032777 000100 161762
017402 001774

017404 005720
017406 005201
017410 062702 020000
017414 001344

017416 104401 001161

;THE FOLLOWING REVISION WAS MADE BY JIM KAPADIA
 .SBTTL CONTROL PANEL TEST # 2
 ;THIS IS THE ENTRY POINT INTO CONTROL PANEL TEST #2. ALL
 ;THE DRIVES THAT ARE PRESENT AND IN 'RDY' CONDITION ARE
 ;REPORTED (ON LINE).

```

SECT.4: NOP
MOV #1,DRKCS
3$: TSTB DRKCS
BPL 3$
MOV #DRIVO,R0
CLR R1
CLR R2

1$: MOV R2,(R0) ;SET UP ADDRESS TABLE
MOV R2,DRKDA ;ADDRESS THE DRIVE
TSTB DRKDS ;IS IT PRESENT?
BPL 2$ ;NO

TYPE EM1 ;TYPE 'DRIVE'
MOV R1,-(SP) ;TYPE OUT DRIVE #
TYPOC
TYPE EM2 ;TYPE 'ON LINE'
BIS #BIT6+BIT7,(R0) ;SET BITS INDICATING THIS
;DRIVE PRESENT

MOV #15,DRKCS ;ISSUE A DRIVE RESET
JSR PC,SMTME ;ALLOW SOME TIME
4$: BIT #RWS,DRKDS ;WAIT FOR RWS RDY
BEQ 4$

2$: TST (R0)+
INC R1
ADD #20000,R2 ;NXT DRIVE
BNE 1$ ;ALL DONE?

TYPE ,SCRLF
    
```

;THIS CODE CHECKS THE CONDITION OF 'DRY' BIT IN RKDS FOR EVERY
 ;DRIVE. IF 'DRY' IS SET DRIVE IS SAID TO BE 'ON LINE', OTHERWISE IT
 ;IS OFFLINE. IF THE 'DRY' BIT HAS CHANGED FROM LAST TIME, THEN
 ;IT IS REPORTED. IF THERE IS NO CHANGE NOTHING IS REPORTED.

```

BEGCT: MOV #DRIVO,R0 ;INITIALIZE POINTERS
CLR R1

BEGCT1: MOV (R0),DRKDA ;ADDRESS A DRIVE
BIC #17777,DRKDA ;MASK OUT NON DR# BITS
TSTB DRKDS ;IS THIS DRIVE ON LINE?
BPL 1$ ;NO
;YES
TSTB (R0) ;WAS IT 'ON LINE' LAST TIME?
BMI NXT1 ;YES, NO MESSAGE TO REPORT
    
```

```

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,DRKDS  ;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,DRKCS  ;WRITE PROT THE DISK
1903 017550 105777 161614          TSTB      DRKCS
1904 017554 100375          BPL        3$
1905 017556 000412          BR         NXT1
1906
1907 017560 105710          1$: TSTB      (R0)      ;WAS THIS DRIVE OFF LINE LAST
1908 017562 100010          BPL        NXT1      ;TIME? BRNCH IF YES
1909 017564 104401 020450          TYPE      EM1          ;IF NOT, REPORT THE CHANGE
1910 017570 010146          MOV        R1,-(SP)  ;TYPE DRIVE #
1911 017572 104402          TYPOC
1912 017574 104401 020473          TYPE      EM3          ;TYPE 'OFF LINE'
1913 017600 042710 000200          BIC        #BIT7,(R0) ;CLEAR BIT TO INDICATE THIS
1914                                ;DRIVE 'OFF LINE'
1915
1916                                ;THIS CODE CHECKS 'WPS' BIT FOR EVERY DRIVE THAT IS IN 'DRY'
1917                                ;CONDITION (ON LINE). IT REPORTS ANY CHANGE IN THE CONDITION OF
1918                                ;THE 'WPS' BIT. IF THERE WAS NO CHANGE FROM LAST TIME NOTHING
1919                                ;IS REPORTED. AT THE TIME OF ENTRY R0 POINTS TO DRIVE FLAG.
1920
1921 017604 105777 161554          NXT1: TSTB      DRKDS  ;IS THIS DRIVE PRESENT?
1922                                ;RKDA CONTAINS THE DRV #
1923 017610 100033          BPL        NXT2      ;NO, SKIP CHECKING
1924
1925 017612 032777 000040 161544          BIT        #WPS,DRKDS ;WPS BIT SET?
1926 017620 001014          BNE        1$        ;YES
1927                                ;WPS BIT CLEAR
1928 017622 032710 000004          BIT        #BIT2,(R0) ;WAS IT CLR LAST TIME ALSO?
1929 017626 001424          BEQ        NXT2      ;YES, NOTHING TO REPORT.
1930                                ;WPS CHANGED FROM 'SET'
1931 017630 104401 020450          TYPE      EM1          ;TO 'CLR', REPORT IT
1932 017634 010146          MOV        R1,-(SP)  ;TYPE DRIVE #
1933 017636 104402          TYPOC
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
1942 017660 104401 020450          TYPE      EM1          ;TYPE 'DRIVE'
1943 017664 010146          MOV        R1,-(SP)  ;TYPE DRIVE #
1944 017666 104402          TYPOC
1945 017670 104401 020506          TYPE      EM4          ;TYPE 'WPS SET'
1946 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 RD POINTS TO THE DRIVE FLAG.
1954
1955 017700 032710 000100      NXT2:  BIT      #BIT6,(R0)      ; WAS THIS DRIVE PRESENT AT BEGNG
1956 017704 001403              BEQ      4$              ; NO
1957 017706 105777 161452      TSTB    DRKDS          ; 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,DRKDA    ; RKDA ALREADY HAS THE DRV #
1962                                ; SET CYL 1 ADDRESS
1963 017726 012777 000011 161434  MOV      #11,DRKCS    ; SEEK, GO
1964
1965 017734 105777 161430      1$:     TSTB    DRKCS    ; WAIT FOR CONTROL RDY?
1966 017740 100375              BPL      1$              ; SOMETHING WRONG IF CNTAL RDY
1967                                ; DOES NOT COME BACK
1968 017742 032777 010000 161414  BIT      #DPL,DRKDS    ; 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      CLRDP1        ; YES, NOTHING TO REPORT.
1973                                ; DPL CHANGED, GOT SET THIS
1974 017760 104401 020450      TYPE    EM1           ; TIME, REPORT IT
1975 017764 010146              MOV      R1,-(SP)
1976 017766 104402              TYPOC
1977 017770 104401 020537      TYPE    EM6           ; TYPE 'POWER LO'
1978 017774 052710 000001      BIS      #BIT0,(R0)    ; SET FLAG BIT INDICATING THAT
1979                                ; DPL SET THIS TIME
1980 020000 000556              BR       CLRDP1
1981
1982 020002 032710 000001      2$:     BIT      #BIT0,(R0) ; 'DPL' BIT IS CLEAR
1983 020006 001410              BEQ      WATSK         ; WAS 'DPL' CLEAR LAST TIME ALSO?
1984                                ; YES, NOTHING TO REPORT
1985 020010 104401 020450      TYPE    EM1           ; REPORT THAT 'DPL' BIT CHANGED,
1986 020014 010146              MOV      R1,-(SP)     ; FROM SET TO CLEAR
1987 020016 104402              TYPOC
1988 020020 104401 020560      TYPE    EM7           ; TYPE 'POWER UP'
1989 020024 042710 000001      BIC      #BIT0,(R0)    ; SET FLAG BIT INDICATING THAT DPL
1990                                ; IS CLEAR THIS TIME
1991
1992      ; THIS CODE WAITS FOR THE SEEK (DONE ABOVE) TO FINISH. WAITING
1993      ; TIME IS APPROX. 50 MS (FOR THE WORST CASE). IF R/W/S RDY
1994      ; DOES NOT SET WITHIN 50 MS, THEN IT IS ASSUMED THAT A 'SIN'
1995      ; IS POSSIBLE AND THE PROGRAM WAITS FOR 1450 MS MORE, SO THAT
1996      ; THE 'SIN' CAN SET. IF 'SIN' DOES NOT SET WITHIN THIS
1997      ; TIME AN ERROR IS REPORTED:
1998      ; SIN DIDN'T OCCUR
1999      ; IF R/W/S RDY SETS WITHIN 50 MS THE PROGRAM PROCEEDS TO
2000      ; CHECK THE NEXT DRIVE.
2001
2002 020030 012705 164220      WATSK:  MOV      #-6000.,R5 ; SET COUNT TO WAIT FOR

```

```

2004 020034 032777 000100 161322 1$: BIT #RWS,ARKDS ;R/W/S. RDY SET?
2005 020042 001042 BNE 3$ ;YES
2006 020044 005205 INC R5 ;WAIT
2007 020046 001372 BNE 1$
2008 ;50 MS OVER, R/W/S RDY
2009 ;DIDN'T SET. WAIT FOR
2010 ;SIN TO SET.
2011 020050 005004 CLR R4
2012 020052 012705 177777 MOV #177777,R5 ;SET UP COUNT
2013 020056 032777 001000 161300 2$: BIT #SIN,ARKDS ;SIN SET?
2014 020064 001045 BNE SIN$ ;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 020102 4$:
2024 020102 104401 020110 TYPE 65$ ;TYPE ASCIZ STRING
2025 020106 000415 BR 64$ ;GET OVER THE ASCIZ
2026 ;65$: .ASCIZ <15><12>/SIN DIDN'T OCCUR, DRIVE/
2027 020142 64$:
2028 020142 010146 MOV R1,-(SP) ;TYPE DRIVE #
2029 020144 104402 TYPOC
2030 020146 000501 BR DN1DRV
2031 020150 032710 000002 3$: BIT #BIT1,(R0) ;DID R/W/S RDY SET LAST TIME?
2032 020154 001476 BEQ DN1DRV ;YES
2033 020156 042710 000002 BIC #BIT1,(R0) ;CLR FLAG INDICATING THAT SEEK IS OK
2034 020162 104401 020450 TYPE EM1 ;REPORT THAT SEEK IS OK
2035 020166 010146 MOV R1,-(SP)
2036 020170 104402 TYPOC
2037 020172 104401 020601 TYPE EM9
2038 020176 000465 BR DN1DRV
2039
2040 ;IF SIN SET, DO DRIVE RESET AND CLEAR IT
2041
2042 020200 032710 000002 SIN$: 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$: MOV ARKDA,R5 ;SAVE RKDA
2051 020232 012777 000001 161130 MOV #1,ARKCS ;DO CONTROL RESET
2052 020240 105777 161124 1$: TSTB ARKCS ;WAIT FOR CONTROL RDY
2053 020244 100375 BPL 1$
2054 020246 010577 161124 MOV R5,ARKDA
2055 020252 012777 000015 161110 MOV #15,ARKCS ;DO DRIVE RESET. RKDA
2056 ;ALREADY HAS THE DRIVE #
2057 020260 105777 161104 2$: TSTB ARKCS ;WAIT FOR CNTRL RDY
2058 020264 100375 BPL 2$

```

```

2060 020270 032777 000100 161066 3$: BIT #RWS,DRKDS ;R/W/S SET?
2061 020276 001025 BNE DN1DRV ;YES
2062 020300 005205 INC R5
2063 020302 001372 BNE 3$ ;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$: .ASCIZ <15><12>/RWS RDY NOT SET/
2068 64$:
2069
2070 020334 000406 BR DN1DRV
2071
2072 ;IF DPL SET CLEAR THE ERROR BY DOING CONTROL RESET.
2073
2074 020336 012777 000001 161024 CLRDPL: MOV #1,DRKCS ;CONTROL RESET
2075 020344 105777 161020 1$: TSTB DRKCS ;WAIT FOR CNTRL RDY
2076 020350 100375 BPL 1$
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 ;RD POINTS TO THE FLAG FOR THE DRIVE THAT WAS
2084 ;JUST CHECKED. BEFORE GOING ON TO THE NEXT
2085 ;DRIVE THE HD DS ARE BROUGHT BACK TO CYLINDER
2086 ;0 (FOR THE NEXT CYCLE).
2087
2088 020352 011077 161020 DN1DRV: MOV (RD),DRKDA ;GET DRIVE #
2089 020356 105777 161002 TSTB DRKDS ;DRIVE PRESENT?
2090 020362 100017 BPL 3$ ;NO
2091 020364 042777 017777 161004 BIC #1777,DRKDA ;CYL ADRES = 0
2092 020372 012777 000011 160770 MOV #11,DRKCS ;GO, SEEK
2093
2094 020400 105777 160764 1$: TSTB DRKCS ;WAIT FOR CNTRL RDY
2095 020404 100375 BPL 1$
2096
2097 020406 004737 005614 JSR PC,SMTME
2098 020412 032777 000100 160744 4$: BIT #RWS,DRKDS
2099 020420 001774 BEQ 4$
2100
2101 020422 005720 3$: TST (RD)+ ;INCREMENT POINTERS TO
2102 020424 005201 INC R1 ;NEXT DRIVE
2103 020426 020127 000010 CMP R1,#10 ;ALL DONE, THIS CYCLE?
2104 020432 001402 BEQ 2$ ;YES
2105 020434 000137 017430 JMP BEGCT1 ;GO DO NEXT DRIVE
2106 020440 000137 017422 2$: JMP 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 /

```

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	020550	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	045517	000		

```

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

```

2151	020614	020614				
2152	020616	000000	DRIV0:	.EVEN	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						

.SBTTL HEAD ALIGNMENT ROUTINE

2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216

: HEAD ALIGNMENT ROUTINE
: THIS MAINTAINANCE ROUTINE IS HELPFUL IN HEAD ALIGNMENT. UPON ENTRY
: THE QUESTION - DRIVE? - IS ASKED. THE USER SHOULD REPLY WITH THE
: DRIVE NUMBER THAT IS TO BE ALIGNED. IF THE DRIVE IS AN RK-05F
: THE LETTER 'F' IS ADDED AS A SUFFIX. FOR SELECTING SURFACE 0
: PUT SW0=0, FOR SELECTING SURFACE 1 PUT SW0=1. SET SW1 =1 TO SELECT
: CYLINDER 64. SET SW1=0 TO SELECT CYLINDER 105.
: IF THE DRIVE IS AN RK-05F, CYLINDER 64 BECOMES CYLINDER 130
: OF THE EVEN DRIVE, AND CYLINDER 105 BECOMES CYLINDER 5 OF THE ODD DRIVE
: THE HEADS ARE PLACED ON THE SELECTED CYLINDER AND DATA IS READ
: CONTINUOSLY FROM THE CYLINDER (SECTOR 0)
: THE UPPER OR LOWER HEAD AND CYLINDER CAN BE SELECTED
: DYNAMICALLY, IE. THE PROGRAM DOES NOT HAVE TO BE STOPPED TO SELECT THE
: UPPER OR LOWER HEAD OR CYLINDER.
: IN ORDER TO SELECT ANOTHER DRIVE, PUT ANY SWITCH FROM SW2 TO SW15 UP AND
: THE PROGRAM WILL AGAIN ASK THE QUESTION (DRIVE?).

020634 000240
020636 104401 020644
020642 000426

020720
020720 104401 020726
020724 000432

021012
021012 104401 021020
021016 000427

021076
021076 104401 021532
021102 005037 021526
021106 104410
021110 012601
021112 112100
021114 162700 000060
021120 002766
021122 022700 000067
021126 002763
021130 000241
021132 006000
021134 006000
021136 006000
021140 006000
021142 010037 020614
021146 112100
021150 001412
021152 020027 000106
021156 001347
021160 105711
021162 001345
021164 042737 020000 020614
021172 005237 021526

SECT.5: NOP
TYPE 655 ;:TYPE ASCIZ STRING
BR 645 ;:GET OVER THE ASCIZ
;:65\$: .ASCIZ <15><12>/SET SW0=0 FOR SURFACE 0, SW0=1 FOR SUR 1./
64\$:
TYPE 675 ;:TYPE ASCIZ STRING
BR 665 ;:GET OVER THE ASCIZ
;:67\$: .ASCIZ <15><12>/SET SW1=0 FOR CYLINDER 105, SW1=1 FOR CYLINDER 64/
66\$:
TYPE 695 ;:TYPE ASCIZ STRING
BR 685 ;:GET OVER THE ASCIZ
;:69\$: .ASCIZ <15><12>/PUT ANY SW FROM 2-15 HI TO SELECT NEW DRIVE/
68\$:
HDALGN: TYPE EM10 ;:ASK FOR DRIVE #
CLR FFLAG ;:FLAG FOR RK-05F
RDLIN ;:GET OPR INPUT
MOV (SP)+,R1 ;:ADDR OF COMMAND STRING
MOVB (R1)+,RO ;:FIRST CHAR
SUB #60,RO ;:0 TO 7
BLT HDALGN ;:TOO SMALL
CMP #67,RO ;:MUST BE 7 OR LESS
BLT HDALGN ;:TOO BIG

ROR RO
ROR RO
ROR RO
ROR RO
MOV RO,DRIVO ;:ADDRESS OF DRIVE
MOVB (R1)+,RO ;:NEXT INPUT CHAR
BEQ \$\$;:ALL DONE IF C.R.
CMP RO,#'F ;:IS IT F?
BNE HDALGN ;:NO, SO ERROR
TSTB (R1) ;:NEXT CHAR MUST BE C.R.
BNE HDALGN ;:ELSE ERROR
BIC #BIT13,DRIVO ;:USE EVEN DRIVE IF RK-05F
INC FFLAG ;:SHOW F TYPE DRIVE

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			15\$:	HALT			: WAIT FOR OPERATOR TO ENTER NEW SWR VALUE
2223	021220	017737	157714	021530	16\$:	MOV	2SWR, 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, 2RKDA		: ADDRESS DRIVE
2230	021254	012777	000017	160106		MOV	#17, 2RKCS		: WRITE PROTECT
2231	021262	105777	160102		8\$:	TSTB	2RKCS		
2232	021266	100375				BPL	8\$: WAIT FOR DRIVE READY
2233	021270	012777	000001	160072		MOV	#1, 2RKCS		: RESET CONTROLLER
2234	021276	105777	160066		9\$:	TSTB	2RKCS		
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 8 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	2RKCS		: ANY ERROR?
2248	021354	100006				BPL	11\$: NO CONTINUE
2249	021356	012777	000001	160004		MOV	#1, 2RKCS		: RESET
2250	021364	105777	160000		12\$:	TSTB	2RKCS		
2251	021370	100375				BPL	12\$: WAIT FOR READY
2252	021372	017702	157542		11\$:	MOV	2SWR, 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		6\$:	MOV	R0, 2RKDA		: ADDRESS THE DRIVE
2257	021414	012777	000017	157746		MOV	#17, 2RKCS		: WRITE PROTECT THE DRIVE
2258	021422	105777	157742			TSTB	2RKCS		: 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, 2SWR		: SW0 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, 2RKDA		: ADRES THE DRIVE
2267	021462	012777	177400	157702		MOV	#-400, 2RKWC		: READ 1 SECTOR
2268	021470	012777	007356	157676		MOV	#RDBUFF, 2RKBA		: INTO THIS BUFFER
2269	021476	012777	000005	157664		MOV	#5, 2RKCS		: READ, GO
2270									
2271	021504	105777	157660		3\$:	TSTB	2RKCS		: DONE?
2272	021510	100375				BPL	3\$: NO

```

2274 021512 032777 177774 157420 BIT #177774,DSWR ;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
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
    
```

.SBTTL DISK POWER FAILURE TEST

```

:(DISK)POWER FAILURE (DURING DISK WRITE) TEST
:THIS TEST CHECKS THAT THE INFORMATION WRITTEN ON THE DISK IS
:NOT DESTROYED WHEN THE DISK SENSES A LOSS OF POWER WHILE DOING A WRITE
:AND RETRACTS THE HEADS. UPON ENTRY THE PROGRAM FINDS OUT THE
:FIRST AVAILABLE DRIVE INDICATES IT (DRIVE XX) AND PROCEEDS TO TEST.
:CYLINDERS 0 TO 15 ARE WRITTEN WITH UNIQUE PATTERNS, THEN THE HEADS
:ARE POSITIONED ON CYLINDER 10 (DECIMAL) AND A MESSAGE (DROP
:POWER) IS GIVEN. AFTER RECEIVING THIS MESSAGE THE USER SHOULD
:DROP POWER FROM THE DRIVE. ON SENSING A LOSS OF POWER, THE
:PROGRAM ASKS THE USER TO PUT BACK THE POWER. THE ERRORS (DPL)
:ARE CLEARED AND A WRITE-CHECK IS PERFORMED TO CHECK IF THE
:UNIQUE PATTERNS ON THE DISK (CYLINDERS 0-9 AND 11-15) ARE STILL
:THERE. IF NOT A WRITE CHECK ERROR IS REPORTED.
.EVEN
    
```

```

021544
SECT.6: CLR R0 ;INITIALIZE DRIVE #
CLR DRIVO
8$: MOV R0,DRKDA ;IS IT PRESENT?
TSTB DRKDS ;IF NOT SKIP
BMI 12$
10$: INC DRIVO
ADD #20000,R0
BNE 8$
BR SECT.6
12$: MOV DRIVO, -(SP) ;GET DRIVE #
BIT #40,DRKDS
BEQ 9$
TYPE ,EM11
TYPOC
TYPE ,EM12
BR 10$
9$: TYPE ,EM1
MOV DRIVO, -(SP)
TYPOC
MOV #1,DRKCS ;CONTROL RESET
TSTB DRKCS
BPL -4
CLR R5 ;INITIALIZE PATTERN TO BE WRITTEN
;0 ON CYL 0, 1 ON CYL 1, ETC
MOV RKDA,R1
MOV RKWC,R2
MOV RKBA,R3
    
```

```

2330 021700 010011          MOV    R0,AR1
2331 021702 010537 022232 15:  MOV    R5,BUFR          ;FILL THE PATER IN DATA BUFFER.
2332 021706 012713 022232  MOV    #BUFR,AR3       ;BUS ADRES
2333 021712 012712 164000  MOV    #-14000,AR2    ;WRITE 1 CYL (256X12X2 WORDS)
2334 021716 012714 004003  MOV    #4003,AR4      ;WRITE GO, IBA SET
2335 021722 105714          TSTB   AR4             ;WAIT FOR CONTROL READY
2336 021724 100376          BPL    -2
2337
2338 021726 005205          INC    R5
2339 021730 020527 000020  CMP    R5,#20         ;DONE
2340 021734 001362          BNE    15             ;WRITTEN ALL 15 CYLINDERS?
2341
2342
2343 021736 010005          MOV    R0,R5
2344 021740 052705 000500  BIS    #500,R5        ;DRIVE #
2345 021744 012737 000012 022232  MOV    #12,BUFR       ;CYL 10
2346 021752 010511          MOV    R5,AR1        ;PATERM TO BE WRITTEN
2347 021754 012712 164000  MOV    #-14000,AR2    ;ADRES THE DISK
2348 021760 012713 022232  MOV    #BUFR,AR3      ;WORD COUNT= 1 CYLINDER
2349 021764 012714 004003  MOV    #4003,AR4      ;BUS ADRES
2350 021770 032777 000100 157366 25:  BIT    #RNS,AR4KDS    ;WRITE GO, IBA
2351 021776 001774          BEQ    25             ;WAIT FOR THE HEADS TO SETTLE
2352 022000 104401 022006          TYPE   65$           ;ON CYL 10
2353 022004 000406          BR     64$           ;TYPE ASCIZ STRING
2354
2355 022022          ;:65$: .ASCIZ /DROP POWER/
2356 64$:
2357
2358 022022 000407          BR     55
2359 022024 010511          MOV    R5,AR1
2360 022026 012712 164000  MOV    #-14000,AR2    ;ADRES THE DISK
2361 022032 012713 022232  MOV    #BUFR,AR3      ;WORD COUNT= 1 CYLINDER
2362 022036 012714 004003  MOV    #4003,AR4      ;BUS ADRES
2363 022042 105714          TSTB   AR4             ;WRITE GO, IBA
2364 022044 100376          BPL    -2             ;WAIT FOR CONTROL READY
2365 022046 005714          TST    AR4
2366
2367 022050 100365          BPL    35
2368
2369 022052 104401 022060          TYPE   67$           ;WAIT FOR DRIVE POWER TO GO DOWN,
2370 022056 000406          BR     66$           ;OTHERWISE, KEEP ON WRITING ON CYL 10
2371
2372 022074          ;:67$: .ASCIZ <15><12>/POWER ON/
2373 022074 105777 157264 66$:  TSTB   AR4KDS        ;IF DRIVE POWER LOSS WAS SENSED,
2374 022100 100375          BPL    -4             ;ASK TO PUT POWER ON.
2375 022102 012714 000001  MOV    #1,AR4        ;TYPE ASCIZ STRING
2376 022106 105714          TSTB   AR4             ;GET OVER THE ASCIZ
2377 022110 100376          BPL    -2
2378 022112 010077 157260  MOV    R0,AR4KDA
2379 022116 005005          CLR    R5             ;INITIALIZE PATTERN
2380 022120 010537 022232 65:  MOV    R5,BUFR
2381 022124 012713 022232  MOV    #BUFR,AR3
2382 022130 012712 164000  MOV    #-14000,AR2
2383 022134 012714 004007  MOV    #4007,AR4
2384 022140 105714          TSTB   AR4             ;WRITE CHECK, GO, IBA
2385 022142 100376          BPL    -2

```

```

2386 022144 005714
2387 022146 100023
2388 022150 104401 022156
2389 022154 000416
2390
2391 022212
2392 022212 011146
2393 022214 104402
2394 022216 005205
2395 022220 020527 000020
2396 022224 001335
2397
2398 022226 000137 021564
2399
2400 022232 000000
2401 022234 051104 053111 020105,
2402 022234 000040
2403 022234 051511 053440 044522
2404 022236 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
2427 022274 100002
2428 022276 000000
2429 022300 000407
2430 022302 010046
2431 022304 017600 000002
2432 022310 112046
2433 022312 001005
2434 022314 005726
2435 022316 012600
2436 022320 062716 000002
2437 022324 000002
2438 022326 122716 000011
2439 022332 001430
2440 022334 122716 000200

```

```

TST 3R4 ;ANY ERROR?
BPL 7S ;NO
TYPE 69S ;TYPE ASCIZ STRING
BR 68S ;GET OVER THE ASCIZ
;69S: .ASCIZ <15><12>/ERROR, ON PWR-UP, RKDA=/
68S:
MOV 3R1,-(SP)
TYPOC
7S: INC R5
CMP R5,#20
BNE 6S
JMP 10S

```

```

BUFR: .WORD 0
EM11: .ASCIZ /DRIVE /
EM12: .ASCIZ /IS WRITE PROTECTED /

```

.SBTTL TYPE ROUTINE

```

*****
#ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
#THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
#NOTE1: SNUL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
#NOTE2: SFILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
#NOTE3: SFILLC CONTAINS THE CHARACTER TO FILL AFTER.

```

```

#CALL:
#1) USING A TRAP INSTRUCTION
# TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
#OR
# TYPE
# MESADR
#

```

```

$TYPE: TSTB STPFLG ;: IS THERE A TERMINAL?
BPL 1S ;: BR IF YES
HALT ;: HALT HERE IF NO TERMINAL
BR 3S ;: LEAVE
1S: MOV R0,-(SP) ;: SAVE R0
MOV 32(SP),R0 ;: GET ADDRESS OF ASCIZ STRING
2S: MOVB (R0)+,-(SP) ;: PUSH CHARACTER TO BE TYPED ONTO STACK
BNE 4S ;: BR IF IT ISN'T THE TERMINATOR
TST (SP)+ ;: IF TERMINATOR POP IT OFF THE STACK
60S: MOV (SP)+,R0 ;: RESTORE R0
3S: ADD #2,(SP) ;: ADJUST RETURN PC
RTI ;: RETURN
4S: CMPB #HT,(SP) ;: BRANCH IF <HT>
BEQ 8S ;:
CMPB #CRLF,(SP) ;: BRANCH IF NOT <CRLF>

```

```
2440 022342 005726 TST (SP)+ ;: POP <CR><LF> EQUIV
2441 022344 104401 TYPE ;: TYPE A CR AND LF
2442 022346 001161 SCRLF
2443 022350 105037 022504 CLRB $CHARCNT ;: CLEAR CHARACTER COUNT
2444 022354 000755 BR 2$ ;: GET NEXT CHARACTER
2445 022356 004737 022440 5$: JSR PC,$TYPEC ;: GO TYPE THIS CHARACTER
2446 022366 123726 001156 6$: CMPB $FILLC,(SP)+ ;: IS IT TIME FOR FILLER CHARS.?
2447 022366 001350 BNE 2$ ;: IF NO GO GET NEXT CHAR.
2448 022370 013746 001154 MOV $NULL,-(SP) ;: GET # OF FILLER CHARS. NEEDED
2449 ;: AND THE NULL CHAR.
2450 022374 105366 000001 7$: DECB 1(SP) ;: DOES A NULL NEED TO BE TYPED?
2451 022400 002770 BLT 6$ ;: BR IF NO--GO POP THE NULL OFF OF STACK
2452 022402 004737 022440 JSR PC,$TYPEC ;: GO TYPE A NULL
2453 022406 105337 022504 DECB $CHARCNT ;: DO NOT COUNT AS A COUNT
2454 022412 000770 BR 7$ ;: LOOP
```

;HORIZONTAL TAB PROCESSOR

```
2455 022414 112716 000040 8$: MOVB #' (SP) ;: REPLACE TAB WITH SPACE
2456 022420 004737 022440 9$: JSR PC,$TYPEC ;: TYPE A SPACE
2457 022424 132737 000007 022504 BITB #',$CHARCNT ;: BRANCH IF NOT AT
2458 022432 001372 BNE 9$ ;: TAB STOP
2459 022434 005726 TST (SP)+ ;: POP SPACE OFF STACK
2460 022436 000724 BR 2$ ;: GET NEXT CHARACTER
2461 022440 105777 156504 $TYPEC: TSTB $STPB ;: WAIT UNTIL PRINTER IS READY
2462 022444 100375 BPL $TYPEC
2463 022446 116677 000002 156476 MOVB 2(SP),$STPB ;: LOAD CHAR TO BE TYPED INTO DATA REG.
2464 022454 122766 000015 000002 CMPB $CR,2(SP) ;: IS CHARACTER A CARRIAGE RETURN?
2465 022462 001003 BNE 1$ ;: BRANCH IF NO
2466 022464 105037 022504 CLRB $CHARCNT ;: YES--CLEAR CHARACTER COUNT
2467 022470 000406 BR $TYPEX ;: EXIT
2468 022472 122766 000012 000002 1$: CMPB $LF,2(SP) ;: IS CHARACTER A LINE FEED?
2469 022474 001402 BEQ $TYPEX ;: BRANCH IF YES
2470 022502 105227 INCB (PC)+ ;: COUNT THE CHARACTER
2471 022504 000000 $CHARCNT: WORD 0 ;: CHARACTER COUNT STORAGE
2472 022506 000207 $TYPEX: RTS PC
```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```
2473 ;: *****
2474 ;: *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
2475 ;: *OCTAL (ASCII) NUMBER AND TYPE IT.
2476 ;: *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
2477 ;: *CALL:
2478 ;: * MOV NUM,-(SP) ;: NUMBER TO BE TYPED
2479 ;: * $TYPOS ;: CALL FOR TYPEOUT
2480 ;: * .BYTE N ;: N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2481 ;: * .BYTE M ;: M=1 OR 0
2482 ;: * ;: 1=TYPE LEADING ZEROS
2483 ;: * ;: 0=SUPPRESS LEADING ZEROS
2484 ;: *
2485 ;: *$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2486 ;: *$TYPOS OR $TYPOC
2487 ;: *CALL:
2488 ;: * MOV NUM,-(SP) ;: NUMBER TO BE TYPED
```

```

2498          ;*
2499          ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2500          ;*CALL:
2501          ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2502          ;*      TYPOC      ;;CALL FOR TYPEOUT
2503
2504 022510 017646 000000          STYPOS: MOV      3(SP),-(SP)      ;; PICKUP THE MODE
2505 022514 116637 000001 022733 MOVVB 1(SP),SOFILL      ;; LOAD ZERO FILL SWITCH
2506 022522 112637 022735          MOVVB (SP)+,SOMODE+1      ;; NUMBER OF DIGITS TO TYPE
2507 022526 062716 000002          ADD      #2,(SP)      ;; ADJUST RETURN ADDRESS
2508 022530 000406          BR      STYPOC
2509 022534 112737 000001 022733 STYPOC: MOVVB #1,SOFILL      ;; SET THE ZERO FILL SWITCH
2510 022542 112737 000006 022735          MOVVB #6,SOMODE+1      ;; SET FOR SIX(6) DIGITS
2511 022550 112737 000005 022732 STYPOC: MOVVB #5,SOCNT      ;; SET THE ITERATION COUNT
2512 022556 010346          MOV      R3,-(SP)      ;; SAVE R3
2513 022560 010446          MOV      R4,-(SP)      ;; SAVE R4
2514 022564 010546          MOV      R5,-(SP)      ;; SAVE R5
2515 022568 113704 022735          MOVVB  SOMODE+1,R4      ;; GET THE NUMBER OF DIGITS TO TYPE
2516 022570 005404          NEG      R4
2517 022572 062704 000006          ADD      #6,R4      ;; SUBTRACT IT FOR MAX. ALLOWED
2518 022576 110437 022734          MOVVB  R4,SOMODE      ;; SAVE IT FOR USE
2519 022578 113704 022733          MOVVB  SOFILL,R4      ;; GET THE ZERO FILL SWITCH
2520 022580 016605 000012          MOV      12(SP),R5      ;; PICKUP THE INPUT NUMBER
2521 022582 005003          CLR      R3      ;; CLEAR THE OUTPUT WORD
2522 022584 006105          1$: ROL      R5      ;; ROTATE MSB INTO "C"
2523 022586 000404          BR      3$
2524 022588 006105          2$: ROL      R5      ;; GO DO MSB
2525 022590 006105          ROL      R5      ;; FORM THIS DIGIT
2526 022592 010503          ROL      R5,R3
2527 022594 006103          3$: ROL      R3      ;; GET LSB OF THIS DIGIT
2528 022596 105337 022734          DECB  SOMODE      ;; TYPE THIS DIGIT?
2529 022598 100016          BPL      7$
2530 022600 042703 177770          BIC      #177770,R3      ;; BR IF NO
2531 022602 001002          BNE      4$      ;; GET RID OF JUNK
2532 022604 005704          TST      R4      ;; TEST FOR 0
2533 022606 001403          BEQ      5$      ;; SUPPRESS THIS 0?
2534 022608 005204          4$: INC      R4      ;; BR IF YES
2535 022610 052703 000060          BIS      #'0,R3      ;; DON'T SUPPRESS ANYMORE 0'S
2536 022612 052703 000040          5$: BIS      #'',R3      ;; MAKE THIS DIGIT ASCII
2537 022614 110337 022730          MOVVB  R3,#$      ;; MAKE ASCII IF NOT ALREADY
2538 022616 104401 022730          TYPE  R3      ;; SAVE FOR TYPING
2539 022618 105337 022732          6$: DECB  $OCNT      ;; GO TYPE THIS DIGIT
2540 022620 003347          BGT      2$      ;; COUNT BY 1
2541 022622 002402          BLT      6$      ;; BR IF MORE TO DO
2542 022624 005204          INC      R4      ;; BR IF DONE
2543 022626 000744          BR      2$      ;; INSURE LAST DIGIT ISN'T A BLANK
2544 022628 012605          6$: MOV      (SP)+,R5      ;; GO DO THE LAST DIGIT
2545 022630 012604          MOV      (SP)+,R4      ;; RESTORE R5
2546 022632 012603          MOV      (SP)+,R3      ;; RESTORE R4
2547 022634 000002 000004          MOV      2(SP),4(SP)      ;; RESTORE R3
2548 022636 012616          MOV      (SP)+,(SP)      ;; SET THE STACK FOR RETURNING
2549 022638 000002          RTI
2550 022640          8$: .BYTE 0      ;; RETURN
2551 022642          .BYTE 0      ;; STORAGE FOR ASCII DIGIT
2552 022644          .BYTE 0      ;; TERMINATOR FOR TYPE ROUTINE

```

```

2554 022733 000
2555 022734 000000
2556
2557
2558
2559
2560 022736 000000
2561 022740 000000
2562 022742 000000
2563 022744 000001
2564 022745
2565 022746
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575 022746 005037 022736
2576 022752 012737 022744 022740
2577 022760 013737 022740 022742
2578 022766 012737 023016 000060
2579 022774 012737 000200 000062
2580 023002 005777 156140
2581 023006 012777 000100 156130
2582 023014 000207
2583
2584
2585
2586
2587
2588
2589 023016 117746 156124
2590 023022 042716 177600
2591 023026 021627 000007
2592 023032 001004
2593 023034 022737 000176 001140
2594 023042 001500
2595
2596 023044
2597 023044 022737 000001 022736
2598 023052 001004
2599 023054 104401 024066
2600 023060 005726
2601 023062 000451
2602 023064 021627 000023
2603 023070 001021
2604 023072 005077 156046
2605 023076 005726
2606 023100 105777 156040
2607 023104 100375
2608 023106 117746 156034

```

```

$OFILL: .BYTE 0          ;; ZERO FILL SWITCH
$OMODE: .WORD 0          ;; NUMBER OF DIGITS TO TYPE
.SBTTL TTY INPUT ROUTINE

;*****
.ENABL LSB
$TKCNT: .WORD 0          ;; NUMBER OF ITEMS IN QUEUE
$TKQIN: .WORD 0          ;; INPUT POINTER
$TKQOUT: .WORD 0         ;; OUTPUT POINTER
$TKQSRV: .BLKB 1         ;; TTY KEYBOARD QUEUE
$TKQEND=.
.EVEN

; *TK INITIALIZE ROUTINE
; *THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
; *SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
; *CALL:
; * JSR PC,$TKINT
; * RETURN
$TKINT: CLR $TKCNT          ;; CLEAR COUNT OF ITEMS IN QUEUE
        MOV $TKQSRV,$TKQIN ;; MOVE THE STARTING ADDRESS OF THE
        MOV $TKQIN,$TKQOUT ;; QUEUE INTO THE INPUT & OUTPUT POINTERS.
        MOV $TKSRV,$TKVEC  ;; INITIALIZE THE KEYBOARD VECTOR
        MOV #200,$TKVEC+2  ;; "BR" LEVEL 4
        TST $TKB           ;; CLEAR DONE FLAG
        MOV #100,$TKS     ;; ENABLE TTY KEYBOARD INTERRUPT
        RTS PC             ;; RETURN TO CALLER

; *TK SERVICE ROUTINE
; *THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
; *BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
; *IT IN THE QUEUE.
$TKSRV: MOVB $TKB,-(SP)    ;; PICKUP THE CHARACTER
        BIC #1C177,(SP)   ;; STRIP THE JUNK
1$: CMP (SP),#7           ;; IS IT A CONTROL G?
        BNE 2$            ;; BRANCH IF NO
        CMP #SWREG,SWR    ;; IS SOFT-SWR SELECTED?
        BEQ 6$            ;; GO TO SWR CHANGE
2$: CMP #1,$TKCNT         ;; IS THE QUEUE FULL?
        BNE 3$            ;; BRANCH IF NO
        TYPE $BELL        ;; RING THE TTY BELL
        TST (SP)+         ;; CLEAN CHARACTER OFF OF STACK
        BR 5$             ;; EXIT
3$: CMP (SP),#23         ;; IS IT A CONTROL-S?
        BNE 32$          ;; BRANCH IF NO
        CLR $TKS         ;; DISABLE TTY KEYBOARD INTERRUPTS
        TST (SP)+         ;; CLEAN CHAR OFF STACK
31$: TSTB $TKS           ;; WAIT FOR A CHAR
        BPL 31$          ;; LOOP UNTIL ITS THERE
        MOVB $TKB,-(SP)  ;; GET THE CHARACTER

```

```

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,$STKS   ;; REENABLE TTY KEYBOARD INTERRUPTS
2613 023132 000002          RTI          ;; RETURN
2614 023134 005237 022735 32$:  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$:  MOVB    (SP)+,$STKQIN ;; AND PUT IT IN QUEUE
2621 023164 005237 022740      INC      $TKQIN     ;; UPDATE THE POINTER
2622 023170 023727 022740 022745  CMP      $TKQIN,$STKQEND ;; GO OFF THE END?
2623 023176 001003          BNE      5$          ;; BRANCH IF NO
2624 023200 012737 022744 022740  MOV      #STKQSR,$STKQIN ;; RESET THE POINTER
2625 023206 000002 5$:  RTI          ;; RETURN
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664

```

:SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
:ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
:SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
:CALL WHEN OPERATING IN TTY INTERRUPT MODE.
\$CKSWR: CMP #SWREG,\$SWR ;; IS THE SOFT-SWR SELECTED
 BNE 15\$;; EXIT IF NOT
 TSTB \$STKS ;; IS A CHAR WAITING?
 BPL 15\$;; IF NOT, EXIT
 MOVB \$STKB,-(SP) ;; YES
 BIC #1C177,(SP) ;; MAKE IT 7-BIT ASCII
 CMP (SP),#7 ;; IS IT A CONTROL-G?
 BNE 2\$;; IF NOT, PUT IT IN THE TTY QUEUE
 AND EXIT

:CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
:ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
:CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
6\$: CMPB \$AUTOB,#1 ;; ARE WE RUNNING IN AUTO-MODE?
 BEQ 2\$;; BRANCH IF YES
 TST (SP)+ ;; CLEAR CONTROL-G OFF STACK
 JSR PC,\$TKINT ;; FLUSH THE TTY INPUT QUEUE
 CLR \$STKS ;; DISABLE TTY KEYBOARD INTERRUPTS
 MOVB #1,\$INTAG ;; SET INTERRUPT MODE INDICATOR

SGTSWR: TYPE ,SCNTLG ;; ECHO THE CONTROL-G (!G)
 TYPE \$MSWR ;; TYPE CURRENT CONTENTS
 MOV \$WREG,-(SP) ;; SAVE SWREG FOR TYPEOUT
 TYPOC ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
 TYPE ,SMNEW ;; PROMPT FOR NEW SWR
19\$: CLR -(SP) ;; CLEAR COUNTER
 CLR -(SP) ;; THE NEW SWR
7\$: TSTB \$STKS ;; CHAR THERE?
 BPL 7\$;; IF NOT TRY AGAIN
 MOVB \$STKB,-(SP) ;; PICK UP CHAR
 BIC #1C177,(SP) ;; MAKE IT 7-BIT ASCII


```

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 #6,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),2SWR ;; 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 #100,2$TKS ;; RE-ENABLE TTY KBD INTERRUPTS
2685 023430 003002 15$: RTI ;; RETURN
2686 023432 004737 022440 16$: JSR PC,$TYPEC ;; 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 > 7?
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 $QUES ;; TYPE ?<CR><LF>
2701 023510 000720 BR 20$ ;; SIMULATE CONTROL-U
2702 .DSABL LSB
2703
2704
2705 *****
2706 *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
2707 *CALL:
2708 * RDCHR ;; GET A CHARACTER FROM THE QUEUE
2709 * RETURN HERE ;; CHARACTER IS ON THE STACK
2710 * ;; WITH PARITY BIT STRIPPED OFF
2711 *
2712
2713 023512 011646 $RDCHR: MOV (SP),-(SP) ;; PUSH DOWN THE PC AND
2714 023514 016666 000004 000002 MOV 4(SP),2(SP) ;; THE PS
2715 023522 005066 000004 CLR 4(SP) ;; GET READY FOR A CHARACTER
2716 023526 005046 CLR -(SP) ;; PUT NEW PS ON STACK
2717 023530 012746 023536 MOV #64$,-(SP) ;; PUT NEW PC ON STACK
2718 023534 000002 RTI ;; POP NEW PC AND PS
2719 023536 64$: TST $TKCNT
2720 023536 005737 022736 1$: TST $TKCNT ;; WAIT ON A CHARACTER

```

```

2722 023544 005337 022736          DEC      STKCNT          ;; DECREMENT THE COUNTER
2723 023550 117766 177166 000004      MOV      @STKGOUT,4(SP) ;; GET ONE CHARACTER
2724 023556 005237 022742          INC      STKGOUT        ;; UPDATE THE POINTER
2725 023562 023727 022742 022745      CMP      STKGOUT,#STKGEND ;; DID IT GO OFF OF THE END?
2726 023570 001003          BNE      2$            ;; BRANCH IF NO
2727 023572 012737 022744 022742      MOV      #STKQSRRT,STKGOUT ;; RESET THE POINTER
2728 023600 000002          RTI                    ;; RETURN
2729
2730 *****
2731 *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2732 *CALL:
2733 *      RDLIN                ;; INPUT A STRING FROM THE TTY
2734 *      RETURN HERE          ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2735 *                          ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
2736
2736 023602 010346          $RDLIN: MOV      R3,-(SP)      ;; SAVE R3
2737 023604 005046          CLR      -(SP)         ;; CLEAR THE RUBOUT KEY
2738 023606 012703 024036      1$:  MOV      #STTYIN,R3   ;; GET ADDRESS
2739 023612 022703 024066      2$:  CMP      #STTYIN+30,R3 ;; BUFFER FULL?
2740 023616 101456          BLOS     4$            ;; BR IF YES
2741 023620 104407          RDCHR    (SP)+,(R3)    ;; GO READ ONE CHARACTER FROM THE TTY
2742 023622 112613          MOV      (SP)+,(R3)    ;; GET CHARACTER
2743 023624 122713 000177      10$: CMP      #177,(R3)    ;; IS IT A RUBOUT
2744 023630 001022          BNE      5$            ;; BR IF NO
2745 023632 005716          TST      (SP)          ;; IS THIS THE FIRST RUBOUT?
2746 023634 001007          BNE      6$            ;; BR IF NO
2747 023636 112737 000134 024034      MOV      #' \,9$       ;; TYPE A BACK SLASH
2748 023644 104401 024034          TYPE    9$            ;;
2749 023650 012716 177777          MOV      #-1,(SP)     ;; SET THE RUBOUT KEY
2750 023654 005303          6$:  DEC      R3           ;; BACKUP BY ONE
2751 023656 020327 024036      CMP      R3,#STTYIN   ;; STACK EMPTY?
2752 023662 103434          BLO      4$            ;; BR IF YES
2753 023664 111337 024034          MOV      (R3),9$      ;; SETUP TO TYPEOUT THE DELETED CHAR.
2754 023670 104401 024034          TYPE    9$            ;; GO TYPE
2755 023674 000746          BR       2$            ;; GO READ ANOTHER CHAR.
2756 023676 005716          5$:  TST      (SP)          ;; RUBOUT KEY SET?
2757 023700 001406          BEQ      7$            ;; BR IF NO
2758 023702 112737 000134 024034      MOV      #' \,9$       ;; TYPE A BACK SLASH
2759 023710 104401 024034          TYPE    9$            ;;
2760 023714 005016          CLR      (SP)         ;; CLEAR THE RUBOUT KEY
2761 023716 122713 000025      7$:  CMP      #25,(R3)    ;; IS CHARACTER A CTRL U?
2762 023722 001003          BNE      8$            ;; BR IF NO
2763 023724 104401 024072          TYPE    ,SCNTLU       ;; TYPE A CONTROL "U"
2764 023730 000726          BR       1$            ;; GO START OVER
2765 023732 122713 000022      8$:  CMP      #22,(R3)    ;; IS CHARACTER A "r"?
2766 023736 001011          BNE      3$            ;; BRANCH IF NO
2767 023740 105013          CLRB    (R3)          ;; CLEAR THE CHARACTER
2768 023742 104401 001161          TYPE    ,SCRLF        ;; TYPE A "CR" & "LF"
2769 023746 104401 024036          TYPE    ,STTYIN       ;; TYPE THE INPUT STRING
2770 023752 000717          BR       2$            ;; GO PICKUP ANOTHER CHACTER
2771 023754 104401 001160      4$:  TYPE    ,SQUES        ;; TYPE A '?'
2772 023760 000712          BR       1$            ;; CLEAR THE BUFFER AND LOOP
2773 023762 111337 024034      3$:  MOV      (R3),9$      ;; ECHO THE CHARACTER
2774 023766 104401 024034          TYPE    9$            ;;
2775 023772 122723 000015          CMP      #15,(R3)+   ;; CHECK FOR RETURN
2776 023776 001305          BNE      2$            ;; LOOP IF NOT RETURN

```

```

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      #STTYIN,4(SP)
2784 024032 000002          RTI                    ;; RETURN
2785 024034 000          9$: .BYTE 0          ;; STORAGE FOR ASCII CHAR. TO TYPE
2786 024035 000          .BYTE 0          ;; TERMINATOR
2787 024036 000030  STTYIN: .BLKB 30     ;; RESERVE 30 BYTES FOR TTY INPUT
2788 024066 177607 000377  SBELL: .ASCIZ <207><377><377> ;; CODE FOR BELL
2789 024072 052536 005015 000  SCNTLU: .ASCIZ /↑U/<15><12> ;; CONTROL "U"
2790 024077 136 006507 000012  SCNTLG: .ASCIZ /↑G/<15><12> ;; CONTROL "G"
2791 024104 005015 053523 020122  SMSWR: .ASCIZ <15><12>/SWR = /
2792 024112 020075 000
2793 024115 040 047040 053505  SMNEW: .ASCIZ / NEW = /
2794 024122 036440 000040
2795          .SBTTL READ AN OCTAL NUMBER FROM THE TTY
2796
2797          ;;*****
2798          ;;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2799          ;;*CHANGE IT TO BINARY.
2800          ;;*CALL:
2801          ;;* RDOCT          ;; READ AN OCTAL NUMBER
2802          ;;* RETURN HERE  ;; LOW ORDER BITS ARE ON TOP OF THE STACK
2803          ;;*          ;; HIGH ORDER BITS ARE IN $HIOCT
2804
2805 024126 011646          SRDOCT: MOV      (SP)-,(SP) ;; PROVIDE SPACE FOR THE
2806 024130 016666 000004 000002  MOV      4(SP),2(SP) ;; INPUT NUMBER
2807 024136 010046          MOV      R0,-(SP)    ;; PUSH R0 ON STACK
2808 024140 010146          MOV      R1,-(SP)    ;; PUSH R1 ON STACK
2809 024142 010246          MOV      R2,-(SP)    ;; PUSH R2 ON STACK
2810 024144 104410 1$: 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$: MOV      (R0)+,-(SP) ;; PICKUP THIS CHARACTER
2815 024156 001412          BEQ      3$          ;; 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$: TST      (SP)+        ;; CLEAN TERMINATOR FROM STACK
2826 024206 010166 000012  MOV      R1,12(SP)   ;; SAVE THE RESULT
2827 024212 010237 024226  MOV      R2,$HIOCT
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          $HIOCT: .WORD 0     ;; HIGH ORDER BITS GO HERE

```

```

2834
2835
2836
2837
2838
2839
2840
2841 024230 010046
2842 024232 016600 000002
2843 024236 005740
2844 024240 111000
2845 024242 006300
2846 024244 016000 024264
2847 024250 000200
2848
2849
2850
2851
2852 024252 011646
2853 024254 016666 000004 000002
2854 024262 000002
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 : -----
2864 $TRPAD: .WORD $TRAP2
2865 $TYPE ;;CALL=TYPE (TRAP+1(104401) TTY TYPEOUT ROUTINE
2866 $TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
2867 $TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
2868 $TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
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 $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 024356 000000          HALT
2891 024360 000776          BR      .-2          ;; HANG UP
2892
2893
2894
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  $PWRMG: .WORD $POWER ;; REPORT THE POWER FAILURE
2910 024444 024456          RTI   ;; POWER FAIL MESSAGE POINTER
2911 024446 000002          $SILLUP: HALT ;; THE POWER UP SEQUENCE WAS STARTED
2912 024450 000000          BR      .-2          ;; BEFORE THE POWER DOWN WAS COMPLETE
2913 024452 000776          $SAVR6: 0 ;; PUT THE SP HERE
2914 024454 000000          $POWER: .ASCIZ <15><12>"POWER"
2915 024456 005015 047520 042527
2916 024464 000122
2917          .EVEN

```


WCERRR	013140	1451	1455#						
WCERRR	013654	1548	1553#						
WFERR	013144	1402	1459#						
WPS	= 000040	291#	1896	1925					
WROCNT	001340	254#	797*	868	929*				
WRITCS	001354	260#	798	1737	1770				
WRLINK	005104	535	615	788#					
WRPRO	016470	1624	1636	1760#					
WRDCK	016260	1618	1630	1728#	1756				
WRTNBY	001317	239#	918*	1026					
X	015610	1692	1697#						
XX	016024	1710#							
SAUTOB	001134	177#	362*	2646	2795				
SBDADR	001122	172#							
SBDAT	001126	174#							
SBELL	024066	2599	2788#						
SCHARC	022504	2445*	2455*	2462	2471*	2476#			
SCKSMR	023210	2632#	2871						
SCHTAG	001100	160#	322	323					
SCM3	= 000000	190#							
SCNTLG	024077	2653	2790#						
SCNTLU	024072	2670	2763	2789#					
SCRLF	001161	191#	1331	1874	2444	2479	2681	2768	2788
SENDAD	001400	149	270#						
SERFLG	001103	163#							
SERMAX	001115	169#							
SERRPC	001116	170#							
SERRTB	001434	307#							
SERTTL	001112	167#							
SFILLC	001156	188#	2448	2479					
SFILLS	001155	187#	2479						
SGDADR	001120	171#							
SGDAT	001124	173#							
SGTSMR	023300	2654#	2869						
SHD	= 000003	11	12						
SHIOCT	024226	2827*	2832#						
SICNT	001104	164#							
SILLUP	024450	2879	2895	2912#					
SINTAG	001135	178#	2651*	2682	2795				
SITEMB	001114	168#							
SLF	001162	192#	2479	2778	2788				
SLPADR	001106	165#							
SLPERR	001110	166#							
SMAIL	= ***** U	349	358	2432					
SMNEW	024115	2657	2793#						
SMSMR	024104	2654	2791#						
SNLL	001154	186#	2450	2479					
SOCNT	022732	2511*	2540*	2553#					
SOMODE	022734	2506*	2510*	2515	2518*	2529*	2555#		
SPASS	001100	161#	1129*	1134					
SPOWER	024456	2910	2915#						
SPWRDN	024310	331	2879#	2907					
SPWRMG	024444	2910#							
SPWRUP	024362	2889	2895#						
SQUES	001160	190#	422	2479	2700	2771	2788		

\$RDDEC= ***** U	2875													
\$ROLIN 023602	2736#	2873												
\$RDOCT 024126	2805#	2874												
\$RDSZ = 000030	2729#													
\$R2A = ***** U	2875													
\$SAVRE = ***** U	2875													
\$SAVR6 024454	2888*	2896	2897*	2898*	2914#									
\$SETUP= 000114	314#	328	329	331	333	354	355	2591	2596	2597	2627	2795		
\$STUP = 177777	314#													
\$SVPIC = 000222	147#	152												
\$SMR = 160000	11	12#	190	2911										
\$TKB 001146	183#	2559	2580	2589	2608	2636	2663							
\$TKCNT 022736	2560#	2575*	2597	2614*	2720	2722*								
\$TKINT 022746	349	2575#	2649											
\$TKQEN= 022745	2564#	2622	2725											
\$TKQIN 022740	2561#	2576*	2577	2620*	2621*	2622	2624*							
\$TKQOU 022742	2562#	2577*	2723	2724*	2725	2727*								
\$TKQSR 022744	2563#	2576	2624	2727										
\$TKS 001144	182#	2559	2581*	2604*	2606	2612*	2634	2650*	2660	2684*				
\$TKSRV 023016	2578	2589#												
\$TN = 000001	11#													
\$TPB 001152	185#	2468*	2479											
\$TPFLG 001157	189#	2426	2479											
\$TPS 001150	184#	2466	2479											
\$TRAP 024230	329	2841#												
\$TRAP2 024252	2852#	2863												
\$TRP = 000012	2856#	2865#	2866#	2867#	2868#	2869	2870#	2871	2872#	2873#	2874#	2875#		
\$TRPAD 024264	2846	2863#												
\$TSTN 001102	162#													
\$TTYIN 024036	2738	2739	2751	2769	2783	2787#								
\$TYPBN= ***** U	2868													
\$TYPDS= ***** U	2868													
\$TYPE 022270	2426#	2856	2864											
\$TYPEC 022440	2447	2454	2461	2466#	2467	2686								
\$TYPEX 022506	2472	2474	2477#											
\$TYPOC 022534	2509#	2865												
\$TYPON 022550	2508	2511#	2867											
\$TYPOS 022510	2504#	2866												
\$OFILL = 024466	2505*	2509*	2519	2554#										
.	131#	135#	140#	147	148#	150#	152#	159#	193	196#	207#	208#	226#	
	248#	326	366#	378#	382#	396#	390#	394#	398#	402#	406#	410#	447#	
	451#	564#	572#	620#	717#	745#	890#	1033#	1052#	1098#	1137#	1158#	1234	
	1243#	1315#	1381#	1466#	1497#	1533#	1562#	1600#	1616#	1641#	1649#	1654#	1658#	
	1670#	1709#	1718#	1755#	1765#	1779#	1812#	1826#	1901#	2027#	2151#	2193#	2259	
	2300#	2323	2336	2354#	2362	2371#	2373	2376	2384	2391#	2479	2559	2563#	

.SDIV	1#	
.SEOP	1#	
.SERRO	1#	
.SERRT	1#	
.SMULT	1#	
.SPOWE	1#	2875
.SRAND	1#	
.SRDDE	1#	
.SRDOC	1#	2795
.SREAD	1#	2556
.SR2AZ	1#	
.SSAVE	1#	
.SSB20	1#	
.SSB20	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)

J07